CREATING STELLAR LESSONS WITH DIGITAL TOOLS

From Integration to Innovation in Technology-Enhanced Teaching

Kenneth J. Luterbach

Creating Stellar Lessons with Digital Tools

Creating Stellar Lessons with Digital Tools prepares teachers in training and in-service teachers to use technologies for design and development activities with middle and high school students. While software, open resources, handheld devices, and other tools hold great potential to enhance learning experiences, teachers themselves must model technology use in ways that inspire students to become producers and leaders rather than consumers and followers. Featuring concrete applications in social studies, English, mathematics, and science scenarios, this book provides pre-service teachers with seven paths to creatively integrate and innovate with computational thinking, datasets, maker spaces, visual design, media editing, and other approaches.

Kenneth J. Luterbach is Associate Professor in the College of Education at East Carolina University, USA.

Creating Stellar Lessons with Digital Tools

From Integration to Innovation in Technology-Enhanced Teaching

Kenneth J. Luterbach

Routledge
Taylor & Francis Group
NEW YORK AND LONDON

Cover image: © Getty Images

First published 2022
by Routledge
605 Third Avenue, New York, NY 10158

and by Routledge
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

Routledge is an imprint of the Taylor & Francis Group, an informa business

Library of Congress Cataloging-in-Publication Data
Names: Luterbach, Kenneth J., (Kenneth James), 1960- author.
Title: Creating stellar lessons with digital tools : from integration to innovation in technology-enhanced teaching / Kenneth J. Luterbach.
Description: First Edition. | New York : Routledge, 2022. | Includes bibliographical references and index. | Provided by publisher.
Identifiers: LCCN 2021053425 (print) | LCCN 2021053426 (ebook) | ISBN 9781032000121 (Hardback) | ISBN 9780367769703 (Paperback) | ISBN 9781003172284 (eBook)
Subjects: LCSH: High school teachers—Training of. | Education, Secondary—Computer-assisted instruction. | Curriculum planning—Technological innovations. | Instructional systems—Design.
Classification: LCC LB1777 .L87 2022 (print) | LCC LB1777 (ebook) | DDC 373.11—dc23/eng/20220215
LC record available at https://lccn.loc.gov/2021053425
LC ebook record available at https://lccn.loc.gov/2021053426

ISBN: 978-1-032-00012-1 (hbk)
ISBN: 978-0-367-76970-3 (pbk)
ISBN: 978-1-003-17228-4 (ebk)

DOI: 10.4324/9781003172284

Typeset in Baskerville
by codeMantra

Access the Support Material: www.routledge.com/9780367769703

To my Son

Contents

Acknowledgements

First, thanks to everyone who checks in with me to make sure I'm okay. Given my usual demeanor and virtual reality, I realize there's only so much you can do, but your efforts are paying off. You are succeeding, in my view, and I very much appreciate your love and encouragement.

Now, to my family, teachers, friends, colleagues, and students. Thank you for enlightening me. You are also succeeding; again, though, due to my limitations there is only so much you can do to instruct one like me.

I am also thankful to the folks at Routledge for their contributions, especially Daniel Schwartz, Katherine Tsamparlis, and Justine Bottles.

Part I

Teaching in Contemporary Times

1 Teaching in Middle Schools and High Schools

As we begin to consider the development of stellar lessons with digital tools, let's proceed with shared conceptions of teachers and their capacity to create stellar lessons in contemporary times. First, as a teacher, your contribution to society is tremendous. Creating a classroom or online environment in which students learn a lot is immensely beneficial. When students learn (joyfully), thanks to you, they gain knowledge and skills crucial to success in life, as well as gain confidence and self-esteem. Though students may not always regard learning as joyful, as teachers, we do well when making learning effective and pleasant. Now, we can turn to activities in this book to pique interest and to nurture discovery. Second, all teachers can create stellar lessons with digital tools. With mobile computers in our purses and pockets, when not in our hands, every teacher possesses technical skills. Creating stellar lessons with digital tools will draw on your technical skills, which you can enhance as you wish by proceeding through hands-on activities in this book. You need not develop a nerdy obsession with technology, though doing so is optional.

Teachers have been incorporating digital technologies into their lessons since Apple and IBM introduced microcomputers to the masses in the early 1980s. Those *technology integration* efforts continue to help students use computers for practical purposes. Indeed, all students graduating from high school can use word processing software. Further, virtually all high school graduates can use *Flipgrid*, and create a rectangle or a circle in a graphics program and include such shapes in their desktop-published documents. Students benefit when teachers help them master those software tools which enable them to complete routine tasks, but there is nothing exceptional or stellar about that. Teachers who go no further than demonstrating routine or consumer uses of software are stuck in what we might regard as the black hole of technology integration. Unfortunately, their students are more likely to be followers than leaders.

How can a teacher emerge from the black hole of technology integration and thereby provide students with greater capacity to lead? Consider the case of a teacher requiring students to combine digital images, text, sound, and video in order to make a presentation, whether in *Keynote*, *PowerPoint*, *Prezi*, or *Flipgrid*, for instance. In this case students need to produce an original work. Since this task requires a synthesis of content and development work, it requires skills at the top of Bloom's original cognitive taxonomy (Bloom, 1956) and revised cognitive taxonomy (Anderson & Krathwohl, 2001). That type of assignment has potential for making stellar use of technology for learning. However, many times the potential in this situation is not realized. Even though production work ensues when using multimedia development tools, lack of planning, disregard for fundamental visual design principles, and minimal time on task often yield terrible results. On the other hand, sometimes students perform admirably in this case. They take appropriate care to assemble media thoughtfully to advance cogent arguments in a consistent visual display, which results in the development of a presentation with persuasive arguments in support of a particular position. This case raises numerous key points, which provide the insights necessary to escape from the black hole of technology integration.

DOI: 10.4324/9781003172284-2

First, the mere specification of an assignment or instructional activity with a development task is insufficient for teaching students to make stellar use of technology. Many teenaged students make extensive use of consumer electronics, especially gaming devices, but comfort using digital technologies is insufficient preparation for successful completion of a development task. Students benefit from demonstrations and guidance. Watching others make exceptionally good use of technologies is important. When teachers model stellar use of technology through design and development activities, students are much more likely to become innovators themselves. Second, a synthesis or production activity creates potential for outstanding use of technology whereas use of a technology as a consumer is not likely to do so. For example, requiring students to engage in routine web searches in order to gain information needed to complete an assignment is respectable, but not stellar.

Teachers emerge from the black hole of technology integration when they help students transition from consumers to producers. In this book, producers are innovators, both designers and developers. Viewing an animation to learn about planetary orbits is a first step toward recalling facts, but creating the animation advances learning because it requires application of knowledge. Likewise, reading about cryptograms in *Book Scavenger* (Chambliss Bertman, 2016), for instance, is beneficial for learning, but creating a program to create and decode cryptograms fosters greater learning and creativity. Viewing graphs on a digital dashboard in order to learn about a particular cause–effect relationship may be helpful for learning, but creating a digital dashboard, for instance, alters the learning experience in favor of higher learning, which occurs at the top of Bloom's Taxonomy (Anderson & Krathwohl, 2001; Bloom, 1956). Using an instructional app to learn is sufficient for many, but designing an app, which requires a synthesis of knowledge and innovation, engages higher-order thinking skills. Designing is one of seven paths to stellar use of technologies. Proceeding along any one or more of the seven STELLAR paths in Figure 1.1 will lead teachers and their students toward innovative and stellar uses of technology. If you fancy the cosmic metaphor, proceeding along a STELLAR path will enable teachers to emerge from the black hole of technology integration toward supernova innovation.

For example, when modeling how to design an app, a teacher might use a program such as *Sketch* or *Adobe XD* to depict the visual design and functionality of the app. To demonstrate how to design and style a 3D character or a new product, a teacher might use *Blender*. Serious design work requires a synthesis of skills and creativity, both of which favor stellar use of digital technologies. After that design work, a teacher might tinker with a 3D printer or laser cutter in a maker space to produce the character or product. In the app development case, after designing the visual interface, a teacher might model engagement in computational thinking by writing and debugging the code for the app. Such modeling may help students gain insights into problem solving. In contemporary times, it is often helpful to leverage datasets, whether for app development or to seek a pattern in a dataset, perhaps using a spreadsheet. When leveraging a dataset, many of which are free of financial cost and easy to acquire, a teacher might demonstrate that visual representations of data sometimes reveal patterns in data. For such demonstrations, a teacher might create graphs in an electronic spreadsheet or use a software tool such as *matplotlib*. Alternatively, a teacher might implement a machine learning algorithm in a spreadsheet or call a Python function that invokes a machine learning algorithm to find patterns in data. Another path to supernova innovation involves use of application software for development work. For example, a teacher might use animation software, such as *Adobe Animate*, in order to depict motion, whether to animate a character or to simulate planetary motion, for instance. Alternatively, a teacher may use Cascading Style Sheet specifications to create the animation. When teachers demonstrate how to engage in design or production work that invites originality, creativity, and thoughtfulness of purpose, rather than show how to use a template or some other *quick and dirty cookie-cutter* approach, they act as innovators and model for students how to make stellar use of digital technologies.

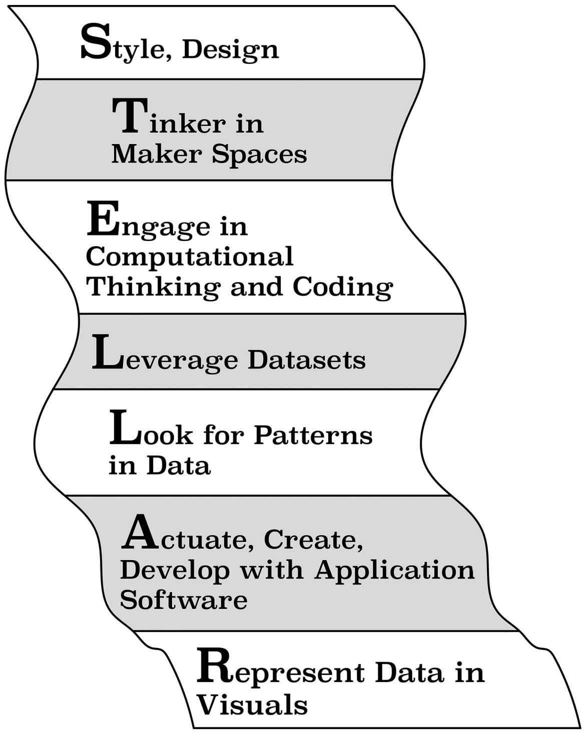

Style, Design
Tinker in Maker Spaces
Engage in Computational Thinking and Coding
Leverage Datasets
Look for Patterns in Data
Actuate, Create, Develop with Application Software
Represent Data in Visuals

Figure 1.1 STELLAR paths

Importantly, the seven STELLAR paths to stellar lessons offer multiple options. You may wish to pursue a single STELLAR path, such as the design path or the path to produce creative work using application software. Alternatively, you may choose to pursue a combination of two to seven STELLAR paths. Consider, for example, Social Studies teachers who want their students to recognize the distortion effects of map projections. Some of those Social Studies teachers will also want their students to realize that they can create maps themselves and specify the map projection. Those teachers will have varying amounts of time to learn the requisite map creation and map editing skills. Given this book, Social Studies teachers with five minutes can learn to acquire mapping data (which may also be called geographic data or geodata) and learn how to drop the data files on a web browser window at a particular website in order to produce accurate and detailed maps of the world or regions of the world. With another five minutes, they can learn to change the map projection, which will enable them to help their students attain the goal of recognizing the distortion effects of map projections. Teachers willing to spend a little more time can learn additional map editing skills, as revealed in this book, and then help their students gain those map editing skills, as well as skills to design an app with geodata and skills to create the app. The STELLAR paths and this book provide you the flexibility to spend as much time as you wish advancing your technological knowledge and your capability to model stellar use of digital technologies.

Technical skills are one requisite to stellar lessons with digital tools. As a teacher, you know about the planning involved in lesson development. We may consider lesson development as both a systematic and a systemic process. With respect to systematic lesson development, you may begin with front-end analysis, particularly learner analysis in order to discover the prior knowledge of your students and gain insights into their motivation to learn, as well as determine what instructional activities they prefer or even enjoy. Second, drawing on your pedagogic and subject matter expertise, you select particular instructional activities to help your students attain the instructional objectives. Third, you select or create instructional materials needed to implement the instruction. Fourth, you deliver the instruction, including opportunities for student engagement and practice. Then you evaluate learning gains and the effectiveness of the instruction. As time permits, you reflect on the lesson and improve it. In that systematic or step-by-step manner, you enhance your teaching practice. You may also consider lesson development as a systemic process. Indeed, teachers work in light of expectations of parents and generally respond to the rules and norms established by school administrators. Further, public school teachers and school administrators respond to school district administrators who are responsible for ensuring that schools operate in accordance with laws. Enactment of a new law or a change in the interpretation of an extant law can alter a teacher's instructional practice.

In this book, we consider both theory and practice of lesson development. As teachers, primarily through the development and delivery of lessons, we seek to help students learn. That is, we endeavor to help our students acquire a particular capability, which may involve the acquisition of facts, concepts, intellectual procedures (e.g., adding fractions, baking bread), physical skills (e.g., sewing, tying knots, skating), principles of cause and effect (e.g., effects of supply and demand on the cost of goods and services), problem-solving strategies (e.g., to write prose; to play scrabble, chess, or go; to design a dress), and/or attitudes (e.g., caring for others; caring for oneself; respecting genuine effort; valuing diversity), for instance. A teacher may also help students gain *metacognitive awareness*, which we may conceive as the capability to control one's own learning by specifying learning goals and monitoring progress toward their attainment (Bransford et al., 2000). Certainly, different terms could be used to describe forms of knowledge that constitute capabilities, and we will consider some additional characterizations of knowledge in Section 2.1, but in this introduction, we proceed with the terms above.

For each lesson, teachers need to determine how to help their students acquire a specific capability or a few specific capabilities. In K–12 schooling, those capabilities are stated explicitly in state standards in the United States, provincial standards in Canada, and national Standards in England. To determine how to help students attain a particular capability, teachers consider multiple factors, including the capability or capabilities to be learned, the prior knowledge of the learners, learner and teacher preferences regarding instructional methods, student safety, time available, and equipment available, for instance.

Ultimately, during lesson preparation, a teacher might decide to make a brief presentation and then have students engage in practice activities, first guided by teacher feedback, and then pursued through independent practice. Alternatively, a teacher might decide to have students engage in inquiry learning, perhaps by conducting an experiment. In either case, more detailed planning would be necessary before lesson delivery. In the first case, the teacher might choose to begin the presentation with a question or a paradox in order to stimulate curiosity (Keller, 1987) and then discuss a particular example or analogy to help students learn (Shulman, 1986). In the latter case, the teacher might direct the students to a book, a website, or other resource that displays the steps the students need to complete in order to conduct the experiment. Of course, too, there are many other instructional methods available to teachers seeking to help their students gain particular capabilities. Given the diversity of students, the large number of instructional objectives they pursue, and the wide variety of digital technologies available, among other factors, selecting instructional methods to help students learn diverse capabilities effectively is challenging, if not daunting. This book seeks to help you meet that challenge by increasing your technical skills and guiding you in the design and development of stellar lessons with digital tools.

Organization of this Book

This book is divided into three parts in order to move from consideration of instructional theory to the development of stellar lessons, which are effective, efficient, and engaging. Part I focuses on teaching in contemporary times. To ensure shared conceptions of critical factors that influence the development of lessons created by middle school and high school teachers, we begin with consideration of schools as instructional environments. In particular, commensurate with the sections in this chapter, we first consider the diversity of school settings, which suddenly, some would say drastically, reformed in order to (temporarily) deliver instruction online or in a hybrid combination of classroom and online instruction. Second, we reflect on the diversity of the primary stakeholders in schools, namely students, teachers, and administrators. Third, we take into account the range of subject matter and instructional goals. Fourth, we ponder the diversity of instructional methods.

In Chapter 2, we consider instructional theories that inform lesson design and development (Dick et al., 2015; Merrill, 1994, 2002; Reigeluth, 1983, 1999; Reigeluth & Carr-Chellman, 2009). In addition, we draw on work contributed by instructional technologists who have long contemplated teaching with technology, including Smaldino et al. (2019) and Mishra and Koehler (2006), in order to fathom the impact of technology on the design of instruction in schools. Then the STARS model for creating STELLAR lessons is introduced in order to facilitate the development of stellar lessons.

Part II of this book considers resources, technologies, and techniques for innovation to further illuminate pathways for developing stellar lessons, which make creative use of technologies and will help propel us beyond technology integration to supernova innovation. Specifically, Chapter 3 begins by fostering innovation through data analytics, which includes consideration of extant data repositories and techniques for analyzing and mining data, including incorporation of machine learning techniques. In addition, Chapter 3 promotes innovation through designing, with special attention paid to visual design, user experience design, and prototyping in *Adobe XD*. Explorations in innovation continue in Chapter 3 with hands-on activities that invite creative use of application software for developing and editing media as follows: (1) Creating Bitmapped Graphics in *GIMP*; (2) Creating Scalable Vector Graphics (SVG) with a Text Editor; (3) Creating Scalable Vector Graphics in *Inkscape*; (4) Audio Recording and Editing in *Audacity*; and (5) Video Editing in *Blender*. Multimedia explorations include development of web pages with HyperText Markup Language (HTML) and Cascading Style Sheets (CSS). The section on the design and production of 3D graphics and animation in *Blender* also fosters innovation. Further explorations in Chapter 3 invite innovative product development using 3D printers and laser cutters, which are available in maker spaces in some schools and in many urban areas.

Chapter 4 invites further expansion of innovative capacity by introducing problem solving through software design and then providing hands-on activities to gain conceptual and procedural knowledge necessary to code solutions in JavaScript and Python. Chapter 4 includes numerous introductory topics in coding for first-time programmers, as well as content for accessing and leveraging data using JavaScript Object Notation (JSON), Representational State Transfer (REST), Application Programming Interfaces (APIs), eXtensible Markup Language (XML), and Structured Query Language (SQL).

Part III applies instructional theory considered in Part I with technical skills acquired in Part II to the practice of lesson development in four particular instructional scenarios.

Each of Chapters 5–8 has one theme, which serves to anchor consideration of outstanding uses of technology to help middle school and high school students attain multiple instructional standards that pertain to the theme. Middle school and high school students in England, Canada, and the United States are expected to meet the instructional standards. Each scenario affords opportunities for you to gain additional practical knowledge of a wide variety of application software and technological tools for fostering creativity and innovation. Further, to exemplify how to create stellar lessons, each scenario demonstrates engagement in multiple STELLAR activities designed to help students attain the instructional standards. The following list identifies the subject areas and themes of the four instructional scenarios.

1. Social Studies: Mapping Our World
2. English: Ruminating Over Words
3. Mathematics: Building Dimensions through Linear Extensions
4. Science: Explorations in Space

The final section of the last chapter offers remarks on advancing your teaching practice.

Creative Thinking and Joyous Learning

There ought to be joy in learning. Students immerse themselves in fun instructional activities that foster creative thinking and lead to goal attainment. The level of challenge must be appropriate, within what Vygotsky (1978) called the *zone of proximal development*. When the instructional goal, activity, and the challenge are appropriate, students become completely absorbed in learning, which Csikszentmihalyi (1975) described as a *flow state*. In common parlance in sports, we might describe such an experience as being "in the zone." As you proceed through this book, here's hoping you experience the flow state and get in the zone. Enjoy the readings and the hands-on activities.

1.1 Diversity of Schools

Schools are not created equally, at least not in many high schools in the United States of America. Though one might expect some inequalities when comparing public and private schools, one might be prone to imagine public schools as equal. On key measures, schools might be relatively equal in some regions of the world, but equality in public schooling does not seem to be the goal in many school systems in the USA. In U.S. school systems, at least in high schools, flexibility appears to be the primary goal of schooling. To create public schools that accommodate people with diverse views on the purpose of schooling, flexibility is necessary. Rather than create high schools in which students pursue essentially the same learning goals, which would favor equality, high schools with multiple programs of study, implemented using distinct instructional methods, are created in attempts to serve students and parents who hold differing views about the purpose of schooling. If the fundamental goal of schooling is to prepare diverse students for the future, one may argue persuasively that flexibility is more important than equality, at least equality through uniformity.

School systems in the USA favor flexibility through local control. The federal government in the USA has limited involvement in academic programming in public schooling. Rather, each U.S. state is primarily responsible for creating and sustaining a system of free education for youth. For the most part, each U.S. state has multiple school districts in order to foster local control of schooling. Further, many U.S. states fund Charter Schools, which have authority to operate outside the rules that apply to conventional school districts and have increased in popularity over time. Specifically, enrollment in U.S. Charter Schools more than doubled between 2009 and 2018, moving from 3 to 7 percent of public school enrollment. In addition, many U.S. states support Magnet Schools or some form of schooling that enables the school to provide a program of study with an emphasis on a particular discipline, such as Fine Arts, Humanities, Science, or Technology. Some school systems also have Alternative Schools, such as middle schools for students over 13 years of age. Further, school systems must make accommodations in curriculum, instruction, and testing in order to serve students with special needs. Many school systems provide special instructional programs for students deemed academically or intellectually gifted. Moreover, numerous high schools offer academic programming that appeals to diverse students (or at least to the parents of the students). For example, in addition to the core subjects, English, Social Studies, Mathematics, and Science, high school programs may focus students on studies in television and film, healthcare, law enforcement, media communications, visual arts, culinary arts, digital arts, hospitality and tourism, environmental conservation, leadership, entrepreneurship, finance, economics, writing, government, urban planning, robotics and engineering, automotive technology, aviation maintenance, building construction, aerospace, music, dance, artistry, medicine, sports medicine, political science, law, ethics, architecture, journalism, and multilingualism, for instance. Further, some programs of study combine those pursuits, which enable students to pursue studies in Science, Technology, Engineering, and Mathematics (STEM), or STEM, Arts, and Humanities, for instance. Some school districts even compel students to apply for admission to middle school and to high school. Then, in a manner characteristic of private schools, the middle schools and high schools select students in accordance with their admission criteria. Ultimately, students denied admission to schools they favor and to which they applied do get placed in a public school in the district. Consider the case of public schools in New York City.

The New York City Department of Education (NYCDOE) places Grade 5 students into middle schools in accordance with their applications for admission to particular middle schools. Similarly, the NYCDOE places Grade 8 students into high schools in accordance with their applications for admission to particular high schools. Grades in academic subjects, attendance, and student preferences for particular schools are factors in admission decisions to public middle schools in NYC. Grade 5 students, accompanied by a parent, go on tours of middle schools in order to determine their preferred middle schools, which are ranked on the application for admission form. The application process for high school admission is somewhat more complex and may involve completion of one or more standardized tests. Admission to a specialized high school, such as The Bronx High School of Science, the Brooklyn Latin School, Queens High School for the Sciences at York College, and Stuyvesant High School, requires attainment of a (very) high score on the Specialized High School Admissions Test (SHSAT). Rather than the SHSAT, admission to the NYC specialized high school for the arts, namely Fiorello H. LaGuardia High School of Music & Art and Performing Arts, requires success in an audition. Even if not a specialized high school, NYC high schools may require students to submit an essay, reply to a series of open-ended questions, or take an admission test other than the SHSAT. In addition, students may arrange for teachers or others to submit letters of recommendation in support of their application for admission. Throughout the application process, which some regard as an ordeal, parents and students may seek information on NYCDOE web pages on middle school and high school admissions (NYCDOE, 2021a, 2021b), as well as consult with parent coordinators in schools, school counselors, school administrators, NYCDOE district personnel, and, at the high school level, a member of the admissions team. Popular high schools receive more than 5000 applications for their 400 seats. In the end, after the middle schools and high schools have made their selections, the NYCDOE assigns all remaining Grade 5 students to a middle school and all remaining Grade 8 students to a high school.

In addition to diverse programs of study in U.S. schools, there are great disparities in school funding in U.S. states and throughout the world. According to data collected in 2016, expressed in U.S. dollars for year 2018, average expenditure of grade schooling per pupil was $13,600 in the United States of America, $11,600 in the United Kingdom, and $11,100 in Canada. All three of those figures are greater than 13% higher than $9800, which was the average for the 37 countries in the Organisation for Economic Cooperation and Development (National Center for Education Statistics [NCES], 2020a, p. 269). Within the USA, for the year 2018, New York

State spent the most at $23,686 per pupil, while Utah spent the least at $7576 per pupil (NCES, 2020b, p. 15). The average (mean) per pupil expenditure for the top five K–12 spending states (New York, District of Columbia, New Jersey, Vermont, and Connecticut) was $21,491, and for the bottom five K–12 spending states (Utah, Idaho, Oklahoma, Arizona, and Mississippi) it was $8176 (NCES, 2020b, p. 15). Given such vast differences in expenditures, there is great diversity in facilities and resources available for instruction in U.S. schools. Those average figures per state do not capture actual spending within particular school districts.

Since the publication of *Savage Inequalities* decades ago (Kozol, 1991), inequities of school funding in the USA have been starkly evident. School funding based on local economies greatly favors wealthy urban areas while leaving lower-income communities unable to supply technology labs or even maintain school buildings (Build America's School Infrastructure Coalition, 2018; Katz, 2016). Kozol (1991) documented decrepit conditions in multiple school districts, including East St. Louis, Illinois, which struggles with local property taxes generating only about 9% of the school district's funds (Koziatek, 2019). Though the East St. Louis School District now receives considerable funds from the State of Illinois and the Federal government, there is still a budget deficit due to the low level of local funds. Further compounding problems for students in East St. Louis, according to Superintendent Arthur R. Culver, are children facing violence, poverty, and substance abuse in the community (Koziatek, 2019). Moreover, as one might expect, the East St. Louis School District faces teacher shortages.

Inadequate administration of schools also burdens students. After attending schools in the Detroit Public School system through 2016, multiple students sued the state of Michigan for failing to provide an adequate education. The students claimed that Detroit Public Schools had multiple deficiencies, including rat infestation, dangerous facilities, unqualified teachers, absent teachers, and invalid resourcing, such as biology textbooks in physics classes. Initially, a federal judge sided with the state of Michigan, but in a 2-1 ruling in the U.S. Court of Appeals for the Sixth Circuit, the students prevailed (Goldstein, 2020). In particular, judges Clay and Stranch stated that the students in the Detroit Public School system were denied "a basic minimum education, and thus have been deprived of access to literacy" (United States Court of Appeals for the Sixth Circuit, 2020). The case was remanded to the lower court for further proceedings. Hence, it is not clear whether any remedy could be forthcoming or even an apology from the Governor of Michigan, for instance. Inadequate physical buildings and facilities are not restricted to one state. A fact sheet from the White House promoting proposed legislation, namely, *The American Jobs Plan*, states, "Too many students attend schools and child care centers that are run-down, unsafe, and pose health risks." Further, "President Biden believes we can't close the opportunity gap if low-income kids go to schools in buildings that undermine health and safety, while wealthier students get access to safe buildings with labs and technology that prepare them for the jobs of the future" (White House, 2021). Schools in the United States are diverse and not created equally.

1.2 Diversity of Students, Teachers, and Administrators

Secondary School Students

Middle school students are early adolescents; high school students are middle and late adolescents. One could claim that all adolescents are special because adolescence is unique. Indeed, adolescence is a wondrous developmental period during which teens progress from relatively naïve and immature human beings to adults, ready to vote and, hopefully, in the spirit of the Roman poet Horace (aka Quintus Horatius Flaccus), seize the day (*carpe diem* everyone).

With the exception of physical characteristics, consider perhaps the greatest qualitative difference between students in elementary and secondary schools. Children in the early years of schooling accept that adults act in their best interest. In contrast, the preteens and teens in secondary school have come to realize that not all ideas conjured by adults are noble, and some of their ideas are to be dismissed as folly. Yes, the "jig is up!" Students in middle school and high school wonder, what's in it for me? Many of them want to assess alleged benefits of attaining particular learning goals before spending hours pursuing them. Some students will still work for extrinsic motivational purposes, such as earning a grade or a teacher's praise, but increasingly through secondary school, students want to invest time in goals they consider worthwhile. Hence, teaching practice throughout secondary school must increasingly take into account the perceived practical utility of instructional goals. Though not optimal for joyous learning, some teachers may disregard consideration of the practical benefit of instructional goals and play the power card, which might sound something like: "If you want to pass this course, you will do as I say." Not exactly comforting words… and possibly revealing of some measure of desperation.

In addition to considering variance in student motivation, teachers may also develop effective lessons after taking into account student differences in culture, language, and intelligence, as well as learner preferences or learning styles, for instance. To help culturally diverse students learn, teachers may engage in culturally responsive teaching (Gay, 2018; Herrera, 2016). To assist students with language differences, teachers may refer to pedagogical strategies offered by Rodriguez et al. (2014). Adjusting instruction for intellectually diverse learners takes multiple forms. In the traditional approach, students who find general classroom work either too simple or too difficult may be dispatched to another school or learning environment. For example, in rare instances, children 8–12 years of age may attend university (BBC News, 2018, 2019; CBS News, 2020; Entrepreneur, 2021). In contrast, inclusive approaches keep students in the general classroom and call for teachers to differentiate instruction in accordance with Universal Design for Learning (UDL) principles (CAST, 2021; Hall et al., 2012). Instruction that incorporates UDL principles provides multiple representations of content; multiple means of learner engagement; and invites learners to demonstrate what they know using multiple means of expression (Rose & Meyer, 2002).

With regard to learner preferences, teachers realize that student preferences for instruction do not always yield satisfactory learning gains. For example, consistent with research conducted by Salomon (1984), students may prefer instructional video to reading, but knowledge retention might favor reading over video. Another issue with learner preferences is variability. After engaging in a particular form of instruction for a few days, learners may prefer a change to the instructional method. Accordingly, teachers often vary instructional methods. Learner preferences somewhat overlap the notion of learning styles, which warrant separate treatment.

The notion of learning styles is controversial, even though educational researchers have been studying them for over four decades and millions of learners have completed online questionnaires on learning styles (e.g., Felder & Soloman, n.d.), which may be regarded as relatively stable cognitive and psychological indicators of perceptions of learning environments (Keefe, 1979; Rogowsky et al., 2015). Much of the controversy stems from the lack of research in support of what is called the *meshing hypothesis*, which holds that matching instruction to students' learning styles maximizes learning. Felder's (2020) defense of the use of learning styles in the design of instruction is sound, and, parenthetically, there is even some entertainment value in it. Felder and other proponents of learning styles reject the meshing hypothesis. They also regard as inappropriate any advice to a learner to pursue a particular field of study because they fit into a particular learning style category. Mainstream proponents of learning styles recognize that in a class of students, diversity in learning styles ought to be expected. Accordingly, they recommend that teachers sequentially address the variety of learning styles. For example, if following Kolb (Institute for Experiential Learning, 2020; Kolb, 2015), a teacher would plan instruction to cycle through concrete experience, reflective observation, abstract conceptualization, and active experimentation.

Consideration of students could consume multiple chapters or even books. The key point for teachers is to know your students. What motivates them? What makes them study? What makes them laugh? What are their personality traits? What support, if any, do their parents provide? Knowing your students is vital when creating and delivering lessons. Knowing your students makes it possible to teach them well.

Teachers

Plenty of teachers are plenty altruistic. Before each school day begins, many middle school and high school English Language Arts, Social Studies, Mathematics, and Science teachers have created or revised 3–7 lesson plans. During each school day, they implement those plans for approximately 75–175 adolescents in groups of 20–25. (During the pandemic, many teachers taught students in the classroom and online simultaneously). After school, secondary school teachers grade the work students have submitted and their lesson planning begins again. Many teachers are happy to do that, even after years of service, and even for modest salaries in many states. Apparently, too, teacher salaries have diminished over time. According to the NCES (2020a, p. 62), after adjusting for inflation, teachers across the nation earned $600 less in 2017–2018 than in 1999–2000. Further, as if working morning, afternoon, and night were insufficient, many teachers are so driven to help students that they engage in multiple hours of professional development. A considerable number of teachers even earn a graduate degree while employed full time as teachers. In some cases, as in North Carolina since 2013, teachers earn the graduate degree without the prospect of a salary increase. Of course, too, many teachers take time to collaborate with their colleagues and mentor novice teachers. Further, some teachers volunteer for various forms of service, whether impacting students directly or indirectly, through service work in professional educational organizations, for instance.

Of course, too, some teachers are not altruistic. There are a few teachers who are not a good match for teaching. In the worst cases, they have ulterior motives harmful to students. In less drastic cases, they don't like teaching young people. In either case, they ought not to be in a classroom nor teaching young people online. Additionally, a few teachers who may have been altruistic initially are now "burned out"; they want desperately to get out of the profession. In most cases, teachers are neither entirely altruistic nor entirely unhappy; they are generally content with the workload and benefits.

Middle school and high school teaching fits some people very well. They have a particular personality and teaching style that work well. They may be demonstrators, facilitators, delegators, or conductors, for instance (Gill, 2021; James, 2021). They are having a good time because most students respond well to their instruction and learn effectively. Here's the "sweet spot" they have come to realize: When seeking to optimize instructional effectiveness and joy of learning, successful teachers connect with their students, they know how to engage them. To assist, this book provides numerous activities for student engagement. Accept that time is needed for lesson preparation and grading; and enjoy interacting with young people in all their diversity. For novice teaches and teacher candidates, keep in mind that there are few superstar teachers in their rookie season. Smooth running of a class with 20–25 adolescents can take a couple of years to achieve, and even then, rebalancing is necessary over time.

In the traditional centralized model of teaching, in which groups of 20–25 students appear before one teacher multiple times a day, some teachers benefit from favorable working conditions. For example, some school administrators create teacher schedules with preparation periods. Further, school administrators often assign a teacher to multiple classes of the same subject and grade level in order to reduce the number of distinct classes to prepare. Some school administrators also favor particular teachers by assigning them to classes with fewer students than typical class sizes. Generally speaking, a supportive principal is a favorable working condition.

With respect to teacher salaries and student–teacher ratios, U.S. states differ, which makes some states more appealing to teachers than others. Similarly, resources available to teachers differ. For example, some schools and school districts purchase subscriptions to instructional content and purchase a digital device for each student. Earning a high salary and teaching smaller numbers of students in a school with abundant resources is a rare set of favorable working conditions, but each alone is favorable. There are also some non-traditional secondary schools in which students work through standalone tutorials, rather than attend classes. In those schools, teachers work as tutors, teaching students one at a time or in small groups of 3–4 students. Teachers in those schools also have time reserved to update the tutorials. Even in traditional schools, some balance can usually be achieved between students pursuing a project and teacher demonstration, which can ease the workload somewhat.

There ought to be joy in teaching. Whatever the teacher's race, culture, gender, and age, a good teacher finds a way to reach students, to help them feel comfortable in class and learn effectively. As a secondary school teacher, when students consistently engage in your instruction, learning and joy follow.

Administrators

Schools should be safe, as well as effective places of learning. Hopefully, students and teachers also experience joy in schools through social interactions and through learning and instruction. Of course, too, school administrators should also find joy in their schools. Through teacher hiring, teacher coaching, and communication, school principals have considerable impact on school safety and academic effectiveness. They also set the mood of a school. In large schools, the principal has assistants who exert more direct influence on operations, but ultimately the principal is responsible for all school operations. Since principles are diverse in personality, they go about their business variably.

According to Dunn and Brasco (2020), principals might be *collaborative, participative, bureaucratic, charismatic, laissez-faire, benevolent despotic*, or *autocratic*. Those categories could serve as a basis for discussing the effects of various type of principals, even though the categories are not all mutually exclusive, and *benevolent despot* seems like an oxymoron, yet some might accept that a nice person could be an absolute ruler. Alternatively, in work comparing the leadership styles of principals, Shepherd-Jones and Salisbury-Glennon (2018) considered three categories, namely *authoritarian, democratic*, and *laissez-faire* leaders. Even though a democratic principal might be authoritarian on rare occasions and laissez-faire in a few instances, those categories are sufficiently exclusive for many comparative purposes.

In particular, Shepherd-Jones and Salisbury-Glennon (2018) sought to discern the effect of authoritarian, democratic, and laissez-faire principals on teacher motivation. Since the stakes are high in employment decisions and conditions, a match between teaching style and principal leadership style is helpful. Sometimes schools are sufficiently large that a teacher may rarely encounter the principal, in which case compatibility of styles may not be of great concern, especially when teachers operate quite autonomously. In other cases, when the school principal influences teacher retention, transfer, and mentoring directly, a mismatch between teaching style and principal leadership style might lead to diminished respect, which in turn could make it very difficult on a teacher. Such a teacher might loathe coming to school on a daily basis and, especially if a new teacher, could fear job termination at the end of the school year (or earlier). In such a situation, a teacher might focus on helping students and work as autonomously as possible; in addition, a teacher might make an alternative plan in case it (*Plan B*) is needed at year's end (or earlier). In contrast, a match between teaching style and principal leadership style can buoy teachers to the point of experiencing joy in school even when some working conditions are not particularly favorable.

In addition to helping teachers help students reach academic objectives, principals influence the mood of the school. Many schools have a particular spirit often so positive that alumni long remember pleasant times at the school. Of course, the converse is also the case for some students. The principal, whatever her or his race, sense of humor, leadership style, or any other characteristic, should at least seek to make all teachers and students, in all their diversity, welcome. Hopefully, in collaboration with teachers, the principal will also create an environment that fosters learning and happiness. As a teacher, you can help the principal immensely by helping students attain learning standards and by fostering a positive school atmosphere. Remember to keep parents informed!

1.3 Diversity of Subject Matter and Instructional Goals

Historically, subjects taught in secondary school courses in the United States and elsewhere have been treated separately. Consequently, in the past, English, Social Studies, Science, and Mathematics teachers collaborated with colleagues in their subject area, rarely across disciplines. However, over the past decade there has been a growing trend toward integrating courses across disciplines, such as English and Social Studies to yield a course called *Humanities*, for instance. The course title *Integrated Science and Mathematics* is sometimes used to signal a mix of, well, Science and Mathematics. Supplementing that course blend with content pertaining to technology and engineering yields the acronym STEM, for Science, Technology, Engineering, and Mathematics. Adding Arts content to STEM yields STEAM. It is conceivable to add Humanities content to STEAM, which might yield the acronym STEAHM (Luterbach et al., 2018). Indeed, a small number of high schools offer specialized programs that include both Humanities courses and STEM courses.

Notably, integration of course content in secondary schools goes well beyond combinations of traditional core subjects and STEAM. For example, in addition to offering core subjects, some high schools specialize in content that combines multiple fields of study, such as media and communication; science and health; urban planning and engineering; aeronautics and mechanics; culinary arts, hospitality, tourism, and business; technology and business; technology and visual design; technology and film making; technology and music; law and technology; law and business; sports and business; liberal arts and science; English and Spanish (or another language or languages); culture, politics, and justice; philosophy, ethics, and engineering; philosophy and creative arts; and art, music, and dance. In such high schools, teachers are hired for their expertise in the integration of the fields of study. Even in middle schools and high schools that do not claim to offer integrated programs of study, urban students often have the opportunity to select from a wide variety of elective courses. A school or school system with small numbers of students and teachers will typically offer fewer course options than larger schools, unless virtual schooling is supported.

Multiple teachers have told me that virtual schooling will now be supported everywhere. They claim that after being forced to deliver instruction virtually during the pandemic, secondary school teachers and school administrators learned that virtual schooling is viable. Though not preferred by all students or parents, some students have thrived during mandated virtual schooling. More about this in the next section. We continue here with consideration of the diverse instructional goals pursued by students in the multitude of courses available to many of them.

Whereas many instructional goals in elementary school seek to help students remember and recall facts, concepts, and principles, instructional goals pursued by secondary school students are often toward the top of Bloom's revised taxonomy (Anderson & Krathwohl, 2001), which has students analyzing details, making evaluations, and creating. For example, consider the following sample of Common Core learning standards in English Language Arts for Grade 6 students (National Governors Association Center for Best Practices and Council of Chief State School Officers, 2010).

- Analyze in detail how a key individual, event, or idea is introduced, illustrated, and elaborated in a text (e.g., through examples or anecdotes) (p. 39).
- Compare and contrast one author's presentation of events with that of another (e.g., a memoir written by and a biography on the same person) (p. 39).
- Trace and evaluate the argument and specific claims in a text, distinguishing claims that are supported by reasons and evidence from claims that are not (p. 39).
- Conduct short research projects to answer a question, drawing on several sources and refocusing the inquiry when appropriate (p. 44).
- Gather relevant information from multiple print and digital sources; assess the credibility of each source; and quote or paraphrase the data and conclusions of others while avoiding plagiarism and providing basic bibliographic information for sources (p. 44).
- Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience (p. 43).
- Write arguments to support claims with clear reasons and relevant evidence (p. 42).
- Write informative/explanatory texts to examine a topic and convey ideas, concepts, and information through the selection, organization, and analysis of relevant content (p. 42).

Even though those learning standards pertain to one field of study, they demonstrate that by the beginning of middle school, teachers must develop lessons that challenge students to analyze, synthesize, judge, and evaluate.

1.4 Diversity of Instructional Methods

Teachers say and do a lot in order to help students learn. Some teachers might concede that they talk too much. There is an old teaching adage, which you have likely heard and perhaps have even voiced yourself, "the sooner the teacher stops talking, the sooner the students will start learning." There is a balance to strike between teacher speech and learner activity, whether teaching students in their homes or in a classroom, or in both places simultaneously. As raised in the previous section, some teachers are convinced that schooling will change post-pandemic because teachers and parents have learned that virtual instruction is effective and viable for a considerable portion of students.

The extent to which schooling changes will depend on multiple factors, including, but most certainly not limited to, willingness of teachers to change instructional practices, student maturity, and parent involvement. Given that high school students are more apt to be independent learners than elementary school students, reforms of teaching practices in high schools may be more likely than reform of elementary schools. I am aware of one high school in which the experience of teaching online during the pandemic has resulted in teachers becoming acutely aware of how they can utilize the vast quantity of digital tutorials and other online instructional resources to help students learn effectively throughout an entire academic year. Post-pandemic, those teachers are planning to restructure their school day to include more student activity and much less teacher talk. Such a change could reform that school from teacher centered to student centered. Students might be in the school building throughout the day working in groups at times and independently otherwise, rather than listening to teachers lecture. Alternatively, students might work in the school building at times and from home at other times. Such an approach would blend traditional schooling with home schooling. It is too soon to determine whether the reforms will endure at the school or occur in other high schools. Regardless of learning environment, teachers remain responsible for helping students attain learning standards by implementing instructional methods.

Regarding instructional methods, there is no single universally accepted definition of an *instructional method*. Consequently, educators conceive of vastly different practices as instructional methods. For example, some educators regard the statement of the learning objective or objectives at the beginning of a lesson as an effective instructional method. Other educators regard that single utterance as one component of an instructional method. As another example, some educators regard homework as an effective instructional method. Other educators scoff at the notion of homework as an instructional method because it lacks specific details about student or teacher activity. If we regard instruction as a goal directed and pre-planned teaching process, following Romiszowski's (1981) definition, and recognize the critical importance of practice in learning (Gagné, 1965, 1985; Merrill, 1983, 1994, 2002; Reigeluth, 1983, 1999; Reigeluth & Carr-Chellman, 2009), we can proceed with a shared conceptualization of an instructional method. In particular, the teacher needs to plan for student practice toward attainment of specific knowledge, which may be regarded as a fact, concept, capability, or disposition (Baumard, 1999; Gagné, 1985).

Many instructional technologists refer to instructional methods as instructional strategies or instructional tactics (Reigeluth, 1990). Whatever term you prefer, it is helpful to recognize the magnitude and diversity of instructional methods. Teachers may help students learn by directing them to compare and contrast entities, to summarize, to devise and to test hypotheses, to interpret a graphic organizer, to question, to respond to questions, and to defend their positions, for instance. Implementation options for those instructional methods and others abound. For example, after presenting a question, a teacher may direct students to write a reply and then to pair with another student to share their replies. You may know that instructional method as *Think-Pair-Share*. *Role playing* is another instructional method, as is the *Jigsaw* or *Expert Jigsaw* method in which each student in a small group masters (or becomes the expert in) a particular topic, then each group member teaches all other group members their specialty, which results (hopefully) in all group members mastering all of the subject matter. Web searches readily lead to lists of instructional strategies advanced by school districts and state or provincial departments of instruction. For example, the Washoe County School District in Nevada produced an annotated list of 49 teaching approaches they call instructional strategies (Community Training and Assistance Center and Washoe County School District, 2015). Similarly, the province of Alberta published descriptions of multiple instructional strategies (Alberta Education, 2002). In addition, the work of Robert Marzano (Marzano, 2003; Marzano et al., 2001; also, Dean et al., 2012) and John Hattie (2009) on teaching effectiveness in classrooms is well known to many educators.

In the next chapter, with recognition of the wide variety of instructional methods, instructional goals, and schools, as well as the diversity of students, teachers, and administrators, we continue our quest to develop stellar lessons through consideration of instructional design practices. Toward the end of the next chapter, the STARS model for the development of STELLAR lessons will be introduced, which will complete our grounding in theoretical foundations. Then our quest moves from theory to practice.

References

Alberta Education. (2002). *Instructional strategies*. https://education.alberta.ca/media/482311/is.pdf

Anderson, L. W., & Krathwohl, D. R. (Eds.). (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Longman.

BBC News. (2018). *Belgian boy Laurent Simons heads off to university aged 8*. https://www.bbc.com/news/world-europe-44668452

BBC News. (2019). *Laurent Simons: Belgian child prodigy drops out of university*. https://www.bbc.com/news/world-europe-50734000

Baumard, P. (1999). *Tacit knowledge in organizations* (S. Wauchope, Trans.). Sage.

Bloom, B. (1956). *Taxonomy of educational objectives. Book 1: Cognitive domain*. Longman.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (2000). *How people learn: Brain, mind, experience, and school*. National Academy Press.

Build America's School Infrastructure Coalition. (2018). *Education equity requires modern school facilities*. https://static1.squarespace.com/static/5a6ca11af9a61e2c7be7423e/t/5ecfc947d4d049452956b77d/1590675788253/BASIC+Education+Equity+-+New+Logo+Update-Just+logo_WhiteBack_Horizontal2.pdf

Rodriguez, D., Carrasquillo, A., & Lee, K. S. (2014). *The bilingual advantage: Promoting academic development, biliteracy, and native language in the classroom*. Teachers College Press.

CAST. (2021). *About universal design for learning*. https://www.cast.org/impact/universal-design-for-learning-udl

CBS News. (2020). 12-year-old genius on sailing through college: "I just grasp information quickly." https://www.cbsnews.com/news/12-year-old-genius-cruising-towards-university/

Chambliss Bertman, J. (2016). *Book scavenger*. Square Fish.

Community Training and Assistance Center and Washoe County School District. (2015). *Instructional strategies list*. https://www.washoeschools.net/cms/lib08/NV01912265/Centricity/Domain/228/Instructional%20Strategies%20List%20July%202015.pdf

Csikszentmihalyi, M. (1975). *Beyond boredom and anxiety: Experiencing flow in work and play*. Jossey-Bass.

Dean, C. B., Ross Hubble, E., Pitler, H. & Stone, B. J. (2012). *Classroom instruction that works: Research-based strategies for increasing student achievement* (2nd ed.). McREL.

Dick, W., Carey, L., & Carey, J. O. (2015). *The systematic design of instruction* (8th ed.). Pearson Higher Education.

Dunn, R., & Brasco, R. (2020). *Supervisory styles of instructional leaders*. The School Superintendents Association. https://www.aasa.org/SchoolAdministratorArticle.aspx?id=7886

Entrepreneur. (2021). *A young girl finished high school by the time she turned 8. Now, she's studying for two degrees in hopes of becoming an astronaut*. https://www.entrepreneur.com/article/366822

Felder, R. M. (2020). Uses, misuses, and validity of learning styles. *Advances in Engineering Education, 8*(1), 1–16. https://advances.asee.org/wp-content/uploads/vol08/issue01/Papers/AEE-Pathways-Felder.pdf

Felder, R. M., & Soloman, B. A. (n.d.). *Index of learning styles questionnaire*. NC State University. https://www.webtools.ncsu.edu/learningstyles/

Gagné, R. M. (1965). *The conditions of learning*. Holt, Rinehart and Winston.

Gagné, R. M. (1985). *The conditions of learning and theory of instruction* (4th ed.). Holt, Rinehart and Winston.

Gay, G. (2018). *Culturally responsive teaching: Theory, research, and practice (3rd ed.)*. Teachers College Press.

Gill, E. (2021). *What is your teaching style? 5 effective teaching methods for your classroom*. Resilient Educator. https://resilienteducator.com/classroom-resources/5-types-of-classroom-teaching-styles/

Goldstein, D. (2020, April 27). Detroit students have a constitutional right to literacy, Court rules. *The New York Times*. https://www.nytimes.com/2020/04/27/us/detroit-literacy-lawsuit-schools.html

Hall, T. E., Meyer, A., & Rose, D. H. (Eds.). (2012). *Universal design for learning in the classroom: Practical applications*. The Guilford Press.

Hattie, J. (2009). *Visible learning: A synthesis of over 800 meta-analyses relating to achievement*. Routledge.

Herrera, S. G. (2016). *Biography-driven culturally responsive teaching*. Teachers College Press.

Institute for Experiential Learning. (2020). *Kolb Learning Style Inventory 4.0*. https://experientiallearninginstitute.org/programs/assessments/kolb-learning-style-inventory-4-0/s0

James. (2021). *The 5 most common teaching styles*. ATutor. https://atutor.ca/teaching-styles/

Katz, D. (2016). *The inequalities are still savage*. https://danielskatz.net/2016/01/15/the-inequalities-are-still-savage/

Keefe, J. W. (1979). Learning style: An overview. In *Student learning styles: Diagnosing and proscribing programs* (pp. 1–17). National Association of Secondary School Principals.

Keller, J. M. (1987). Strategies for stimulating the motivation to learn. *Performance & Improvement, 26*(8), 1–7.

Kolb, D. A. (2015). *Experiential learning: Experience as the source of learning and development*. Pearson Education.

Koziatek, M. (2019). *East St. Louis school district cites progress, asks state for additional $6.5 million*. Belleville News-Democrat. https://www.bnd.com/news/local/article236235633.html

Kozol, J. (1991). *Savage inequalities*. Harper Perennial.

Luterbach, K. J., Xiao, W., Tang, H., & Chen, H. (2018). Reflections on World Education Day. *TechTrends, 62*(2), 143–145.

Marzano, R. J. (2003). *What works in schools: Translating research into action.* Association for Supervision and Curriculum development.

Marzano, R. J., Pickering, D. J., & Pollock, J. (2001). *Classroom instruction that works: Research-based strategies for increasing student achievement.* Association for Supervision and Curriculum Development.

Merrill, M. D. (1983). Component display theory. In C. M. Reigeluth (Ed.), *Instructional design theories and models: An overview of their current status.* Lawrence Erlbaum Associates.

Merrill, M. D. (1994). *Instructional design theory.* Educational Technology Publications.

Merrill, M. D. (2002). First principles of instruction. *Educational Technology Research & Development, 50*(3), 43–59.

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record, 108*(6), 1017–1054.

National Center for Education Statistics. (2020a). *The condition of education 2020.* https://nces.ed.gov/programs/coe/. https://nces.ed.gov/pubs2020/2020144.pdf

National Center for Education Statistics. (2020b). *Revenues and expenditures for public elementary and secondary education: Fiscal Year 2018.* https://nces.ed.gov/pubs2020/2020306.pdf

National Governors Association Center for Best Practices and Council of Chief State School Officers. (2010). *Common core state standards for English language arts and literacy in history/social studies, science, and technical subjects.* http://www.corestandards.org/wp-content/uploads/ELA_Standards1.pdf

New York City Department of Education. (2021a). *Welcome to NYC middle school admissions.* https://www.myschools.nyc/en/help/middle-school/

New York City Department of Education. (2021b). *NYC high school and specialized high school admissions guide.* https://www.schools.nyc.gov/enrollment/enroll-grade-by-grade/high-school/nyc-high-school-admissions-guide

Reigeluth, C. M. (Ed.). (1983). *Instructional-design theories and models: An overview of their current status.* Lawrence Erlbaum Associates.

Reigeluth, C. M. (1990). Instructional strategies and tactics. In T. Husen & T. Neville Postlethwaite (Eds.), *The international encyclopedia of education research and studies (Volume 2)* (pp. 314–319). https://www.researchgate.net/publication/330741196

Reigeluth, C. M. (Ed.). (1999). *Instructional-design theories and models: A new paradigm of instructional theory (Vol. II).* Lawrence Erlbaum Associates.

Reigeluth, C. M., & Carr-Chellman, A. (Eds.). (2009). *Instructional design theories and models, Volume III: Building a common knowledge base.* Routledge.

Rogowsky, B. A., Calhoun, B. M., & Tallal, P. (2015). Matching learning style to instructional method: Effects on comprehension. *Journal of Educational Psychology, 107*(1), 64–78. https://doi.org/10.1037/a0037478

Romiszowski, A. J. (1981). *Designing instructional systems: Decision making in course planning and curriculum design.* RoutledgeFalmer.

Rose, D. H., & Meyer, A. (2002). *Teaching every student in the digital age: Universal design for learning.* Association for Supervision and Curriculum Development.

Salomon, G. (1984). Television is "easy" and print is "tough": The differential investment of mental effort in learning as a function of perceptions and attributions. *Journal of Educational Psychology, 76*(4), 647–658.

Shepherd-Jones, A. R., & Salisbury-Glennon, J. D. (2018). Perceptions matter: The correlation between teacher motivation and principal leadership style. *Journal of Research in Education, 28*(2), 93–131. https://files.eric.ed.gov/fulltext/EJ1201598.pdf

Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher, 15*(2), 4–14.

Smaldino, S. E., Lowther, D. L., & Mims, C. (2019). *Instructional technology and media for learning* (12th ed.). Pearson.

United States Court of Appeals for the Sixth Circuit. (2020). *Gary B., Jessie K., Cristopher R., Esmeralda V., Paul M, and Jaime R. (minors) v. Gretchen Whitmer, et al.* Right to Literacy Detroit. https://www.detroit-accesstoliteracy.org/wp-content/uploads/2020/04/Sixth-Circuit-Opinion_2020-04-23.pdf

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes.* Harvard University Press.

White House. (2021). *FACT SHEET: The American jobs plan.* https://www.whitehouse.gov/briefing-room/statements-releases/2021/03/31/fact-sheet-the-american-jobs-plan/

2 Designing Instruction and Stellar Lessons

At times, like educational philosophers, teachers may contemplate the nature and goals of education. In those times, teachers might reflect on Behaviorism, Constructivism, Critical Theory, Democracy, Essentialism, Hermeneutics, Humanism, Multiculturalism, Perennialism, and Progressivism (Cahn, 2012; Nicholson, 2016; Noddings, 2018; Siegel et al., 2018), for instance. For scholarly treatises on those topics, teachers might turn to journals and other publications of organizations such as the Philosophy of Education Society of Great Britain (https://www.philosophy-of-education.org/) and the North American Association for Philosophy and Education (https://www.naape.org/). During your reflections on the nature and goals of education, you may return repeatedly to this one question: How to teach? Questions of who, what, where, and when to teach are evident in employment contracts, curriculum guides, and teaching assignments. Occasionally, teachers may wonder why they teach, but when focused on the job, the key question is how to teach.

Libraries in Schools of Education, as well as thousands of websites, videos, podcasts, and social media posts, are filled with advice on how to teach. Some of that advice pertains to instructional delivery while other guidance considers the process of lesson planning. Given our goal to create stellar lessons, we will focus on the planning process, which has been considered extensively by instructional designers and technologists. Let's draw on their work to inform pursuit of our goal, to create stellar instruction.

2.1 Instructional Design

Since the field of instructional technology has plenty of roots in behaviorism (Reiser & Dempsey, 2018), decisions concerning how to teach revolve around how best to help students attain a particular capability that can be observed. Again, owing to behaviorism, statements of capabilities are presented in terms of the behavior one can observe when the learner has attained the capability (Gagné, 1965; Mager, 1962, 1984). To produce effective and efficient instruction through which learners gain observable capabilities, instructional technologists recommend engaging in a systematic Instructional Design (ID) process. The general framework for systematic ID is the ADDIE process (Molenda, 2003), which includes the following phases or stages: ***A****nalysis,* ***D****esign,* ***D****evelopment,* ***I****mplementation*, and ***E****valuation*.

In the *Analysis* stage, the instructional designer seeks to become aware of the instructional context, particularly the learners, the observable capability the learners should attain, and the need for instruction. When time and funds permit, an instructional designer would conduct a needs analysis, a task or content analysis, and a learner analysis in order to obtain information needed to create effective instruction (Brown & Green, 2020). For guidance on ID processes, one may consult descriptions of general ID models (e.g., Carey et al., 2022; Gagné et al., 2005; Morrison et al., 2019; Romiszowski, 1981, 2016; Smith & Ragan, 2005). Additionally, one may draw on scholarly work pertaining to each stage of the ID process. For example, Saxena

DOI: 10.4324/9781003172284-3

(2011) discusses culturally competent learner analysis; Jonassen and Hannum (1986) analyze 30 task analysis procedures; Clark et al. (2008) and Van Geel et al. (2019) discuss Cognitive Task Analysis; and Rossett (1995) discusses needs analysis. In case you might regard the combination of learner analysis, task analysis, and needs analysis as sufficient preparation for creating instruction, note that Tessmer and Richey (1997) argue that those systematic approaches to instructional design are lacking because they might not take into account learning environment variables. Hence, for Tessmer and Richey (1997), instructional design ought to be systemic, which in the case of front-end analysis involves conducting a contextual analysis.

After front-end analyses, the *Design* stage begins, which involves planning, especially the selection of instructional methods and materials intended to help learners attain the behavioral objective or objectives articulated in the *Analysis* stage (Ertmer & Newby, 2013; Smith & Ragan, 2005). Then, in the *Development* stage, instructional materials are created, which could involve modification of existing materials. The instruction is delivered during the *Implementation* stage. Conceivably, learners may engage in the instruction with or without peers and with or without a teacher. Lastly, during the *Evaluation* stage, attainment of behavioral objectives might be observed and documented using valid and reliable assessment instruments.

ADDIE has become the common term for a systematic ID process. Derivation of the term is not attributed to any particular author but appears to have emerged through communications of instructional technologists over time in light of prior work on instructional design (Molenda, 2003). Through research and development, instructional design aims to make instruction effective and efficient by establishing which particular instructional methods are best suited to helping learners attain specific learning outcomes (Merrill, 1983, 1994, 2002; Reigeluth, 1983, 1999; Reigeluth & Carr-Chellman, 2009). One readily finds elements of ADDIE in Gagné's seminal works on requisite conditions for learning and the systematic design of instruction (Gagné, 1965; Gagné & Briggs, 1974). One important outcome of Gagné's research and scholarship was derivation and presentation of a sequence of nine events of instruction (Gagné et al., 1992). The first three of those events seem to prepare the learner for the instruction by gaining the learner's attention; stating the instructional objective or objectives; and stimulating recall of prior relevant knowledge. Fourth, instructional stimuli are presented. Fifth, the learner is provided some semantic guidance. Sixth, the learner is prompted to practice. Then feedback is provided, and the performance assessed. The last of the nine events of instruction involves enhancing retention and transfer. Whereas the first eight events of instruction are directly attributable to the influence of behaviorism, the last event is somewhat exceptional. Enhancing retention by repeated stimulus-response practice over time still brings behaviorism to mind, but enhancing transfer is more apt to bring constructivism to mind. Even though Gagné's work was heavily influenced by behaviorism, the Information Processing Theory (Atkinson & Shiffrin, 1968), a cognitivist notion that proposes a memory system and control processes, also informed the work. In such light, Gagné regarded learning as more than observable stimulus-response associations and encouraged instructors to enhance knowledge transfer.

There are multiple alternatives to the traditional ADDIE approach to instructional design. Constructivists, for instance, argued that the traditional behavioral approach to the systematic design of instruction is not helpful when teaching complex cognitive skills for problem solving in the real world (Duffy & Jonassen, 1992; Van Merriënboer et al., 1992). In one particular instance, Tessmer et al. (1990) disputed the claim that development of instruction for conceptual understanding should follow traditional notions of instructional design, which hold that learners need to acquire classification rules in order to understand concepts. Merrill and others who promoted traditional ID largely accepted the arguments that ID models founded on behaviorism are limited in their capacity to provide guidance on the development of instruction for particular learning outcomes (Merrill et al., 1990). One such limitation pertains to complex learning, such as statistical analysis, troubleshooting, and computer programming

(Van Merriënboer et al., 1992). Originally, Van Merriënboer et al. (1992) described the 4C/ID model as comprised of two phases: analysis and design. The analysis phase included both a skills analysis and a knowledge analysis. The skills analysis of expert performance revealed recurrent skills, such as text editing skills used when computer programming. In addition, the skills analysis, through cognitive task analysis, also revealed nonrecurrent skills, such as classifying computer programming tasks in accordance with an expert mental model, which distinguishes, for instance, between object-oriented, imperative, and functional programming paradigms. Analysis of recurrent skills and analysis of nonrecurrent skills were the first two components of the original 4C/ID model. The knowledge analysis produced outputs considered in the final two components of the original 4C/ID model, which were analysis of prerequisite knowledge for performance of recurrent skills, and analysis of support knowledge for performance of nonrecurrent skills. After multiple refinements over three decades, the 4C/ID model now regards learning tasks, pre-task practice, procedural information, and supportive information as the four components and prescribes a ten-step process for designing effective instruction for complex learning (Frerejean et al., 2021; Van Merriënboer et al., 1992; Van Merriënboer & Kirschner, 2017; for online descriptions and examples of the model, see https://www.4cid.org).

Whereas traditional ID models were influenced by behaviorism, the 4C/ID model employs cognitive task analysis to reveal mental models of experts and to derive authentic tasks, which are clear signs of its constructivist foundation. Both behavioral and constructivist ID models proceed from the view that knowledge of learning is an important, if not vital, factor necessary to create effective instruction. Even though many educators spend years learning about learning, one may still question the extent to which knowledge of learning is important in the development of effective instruction. Previously, B. F. Skinner (1950) also wondered about the utility of learning theories. One could take the view that each learner must have some tacit knowledge of learning because they know how they have learned. Such tacit knowledge might be sufficient for instructional development. Yet, even if sufficient, one could counter that instructional developers with explicit knowledge of learning create better instruction than other instructional developers. However you regard those claims, Allen (2012) posits that one who follows a Successive Approximation Model can not only develop effective instruction, but can create superior learning experiences.

According to Allen (2012), the development of instruction is too complex to expect production of effective instruction through immediate application of prior knowledge of the learning context. Rather, the first step is "to formulate our best guess, our first 'approximation' of an ideal design" (Allen, 2012, p. 16). Then refine the instruction successively through repeated formative evaluations. This notion to refine instruction iteratively and quickly is also captured in the rapid prototyping approach discussed by Tripp and Bichelmeyer (1990). Successive approximation models do not entirely eschew up-front analysis or information gathering about the instructional context. They collect only the information needed and remain focused on the task. This honors the *minimum commitment* principle of design (Asimow, 1962; Chitale & Gupta, 2013), which uses the fewest resources needed to complete each step of the process. Seemingly, the number of iterations needed will depend in large part on the effectiveness of the initial instructional prototype.

Applications of ADDIE and the 4C/ID model might be criticized for taking too much time in the analysis stage, in some cases bringing "paralysis by analysis" (Hites Anderson, 2010, p. 139) to mind. Further, expending extensive resources on front-end analyses before any development runs the risk of incurring substantial costs for instruction that might not deliver expected results. On the other hand, trivial analyses obtained from allocation of too few resources is likely to result in extensive and costly formative evaluations. Hence, instructional designers are well advised to honor the minimum commitment principle in order to balance time spent on front-end analyses and formative evaluation when seeking to create effective, efficient, and satisfying or (better) inspiring instruction.

2.2 Digital Technologies and the Design of Instruction in Schools

Since microcomputers became available commercially in the early 1980s, teachers have continually sought to integrate them into their teaching practices. This integration process was gradual for 40 years. Then the coronavirus pandemic struck, and teachers were suddenly forced to replace their classroom teaching practices with online instruction. In many cases, middle school and high school teachers actually engaged in classroom and online instruction simultaneously. In the next section we consider lessons learned through the evolution and subsequent revolution of instructional uses of computers in middle schools and high schools.

2.2.1 Technology Integration

First, consider the 40-year evolution of instructional uses of computers.

1980s: Schools purchased microcomputers and placed them in classrooms, sometimes one microcomputer per classroom; other times 15 or more microcomputers in one room to form a computer lab. Initially, teachers and students learned to use command line operating systems, as well as standalone computer applications for word processing and numerical processing (spreadsheets). Then teachers and students learned windows operating systems and a greater variety of standalone applications, such as programs to create graphics and desktop publishing applications. Very little use was made of drill and practice computer-based instruction. A small number of high school students took courses in introductory computer programming.

1990s: Microcomputers in schools were connected to high-speed networks – gone were the days of 300 baud modems and sitting in front of the screen waiting for content to arrive. Teachers began publishing and sharing instructional materials on the World Wide Web, and students began to access those instructional resources. To foster inquiry learning, some teachers directed students to engage in the discovery activities in web quests. Teachers instructed students to learn a greater variety of computer applications, such as web page editing, file transfer, email, and audio editing. Some knowledge building communities formed (Scardamalia & Bereiter, 1994), which began to move learning from a standalone or individual endeavor toward Computer-Supported Collaborative Learning. Also, in this decade, a small number of high school students learned to create web pages in the HyperText Markup Language (HTML), and a few high school students took courses in introductory computer programming.

2000s: This decade brought Web 2.0 technologies and greater collaboration among students. Students learned of blogging and crowd sourcing. Some teachers continued to direct students to web quests, and teachers continued to help students learn a variety of computer applications. A small number of high school students continued to learn to create web pages in HTML, and a few high school students took courses in introductory computer programming. Toward the end of this decade, mobile computing devices were introduced, and 1:1 computing in schools no longer referred only to one microcomputer per student, but to one mobile device per student. Students begin to use social media.

2010s: The massive acquisition of mobile devices for students increased throughout this decade, and many teachers responded by integrating those devices into their instructional practices. This resulted in students learning to use instructional apps on mobile devices and to access instructional resources on mobile devices. The increase in student use of mobile devices for communication heightened concerns about cybersecurity and ethical uses of computing devices. Curricula were developed to help students of all ages become aware of appropriate uses of computing devices. As in past decades, few students learned to create web pages in HTML. However, the movement to foster computational thinking began, and increasing numbers of students throughout the decade learned fundamentals of computer programming.

Throughout the four past decades, teachers have engaged in preservice and in-service professional development in order to integrate digital technologies into their instructional practice. Along their journey they have been guided by the International Society for Technology in Education (ISTE) Standards for Educators (2017) and ISTE Standards for Students (2016), which have moved from acquisition of basic computer skills, 1998, to using technology to learn, 2007, to transformation of learning with technology, 2016. In addition, teachers have considered numerous books (e.g., Cennamo et al., 2019; Hunter, 2015; Smaldino et al., 2019) and resources, such as the extensively researched *Concerns-Based Adoption Model*, which offers a method for measuring level of technology integration in schools (Hall et al., 2013), as well as blog posts, such as ones on technology integration in schools (Puentedura, 2013). Moreover, teachers have attended conferences, completed workshops, engaged in self-study, and even completed courses and graduate degrees in instructional technology. Due to such extensive professional development and technology integration efforts over the past four decades, several million students throughout the world have learned to use a wide variety of application software and digital devices to accomplish practical tasks.

The unanticipated revolution to distance learning in 2020, which could be described as teacher-guided and parent-supervised home schooling, has taught us that schooling can be different, particularly for high school and middle school students. Now, teachers are poised to guide students to new heights. Continued professional development remains important, and by turning from integration toward creative use of digital technologies for learning, we can guide students toward innovation and leadership.

Technological, Pedagogical, and Content Knowledge (TPACK)

Some people who teach are extraordinary subject matter experts, but they may not be competent teachers. Consider, for instance, an expert in statistics who can wax poetically about the Central Limit Theorem. Such an expert might speak flawlessly of a normalized sum of independent random variables tending toward a normal distribution. As part of the discussion, the expert might be keen to present a mathematical expression for the Central Limit Theorem, as in the expression below.

$$Z = \lim_{n\to\infty} \sqrt{n}\left(\frac{\bar{X}_n - \mu}{\sigma}\right) \textit{ is a normal distribution}$$

Then a proof might ensue, and some students might wonder whether to withdraw from the course.

One challenge for the subject matter expert in an instructional setting is to connect the content, in this case the Central Limit Theorem, to prior knowledge of the students. At the point in the course when this topic would be presented, students would already be familiar with the graphical depiction of the normal distribution, the bell curve. Recognizing this, the teacher might do well to present a scenario that would help students visualize values forming a bell curve. For example, imagine flipping a coin several times (say 1000) and counting the number of times the coin is heads. Maybe after the 1000 flips, the coin was heads 510 times. Now imagine 5000 such trials of 1000 coin flips. Most often, the number of heads would likely be around 500 (e.g., 499, 501, 490, 509), but sometimes the number of heads would be farther from 500 (e.g., 480, 485, 520, 537), and on rare occasions, the number of heads would be quite far from 500 (e.g., 451, 550). Plotting the number of heads obtained over all 5000 trials would approximate the image of the bell curve. According to the Central Limit Theorem, if the number of trials were infinite (rather than the 5000 used in this scenario), the normal distribution (bell curve) would result. With that explanation, students might find the Central Limit Theorem more comprehensible. Then the algebraic expression above could be explained, with reminders about the various symbols (e.g., $\bar{X}$ is a sample mean; μ is the population mean; and σ is the standard deviation). With limited pedagogical knowledge, the subject matter expert might overestimate the level of content knowledge the students possess. In contrast, an instructor with more pedagogical knowledge than the subject matter expert would know the target learners better or would first discern the extent of prior knowledge of the students. The point is that the combination of Pedagogical Knowledge and Content Knowledge (PCK) is necessary to teach effectively. This was the insight advanced by Lee Shulman (1986). In the case of teaching with technology, technological knowledge is also necessary. As discussed by Punya Mishra and Matthew Koehler, the combination of Technological, Pedagogical, And Content Knowledge (TPACK) is necessary to teach effectively with technology (Koehler, 2012; Koehler & Mishra, 2009; Mishra & Koehler, 2006).

Part I of this book considers elements of Pedagogical Knowledge (PK), particularly lesson development. Part II of this book helps build Technological Knowledge (TK) primarily. Part III of this book provides unique opportunities to build Technological, Pedagogical, And Content Knowledge (TPACK). This book invites you to add STELLAR activities to your technological, pedagogical, and content knowledge in order to foster student creativity, innovation, and leadership.

Realities of Lesson Development in Middle Schools and High Schools

In the real world, idealized implementations of the ADDIE model are impractical. Middle school and high school teachers do not need to conduct most front-end analyses. Information that would be obtained during a needs analysis is already evident to teachers. Through continuous assessment, teachers become increasingly aware of the knowledge and skills acquired by their students and what their students still need to learn to master state learning standards in the United States, provincial standards in Canada, and national Standards in England. Further, middle school and high school teachers do not need to conduct task analyses because, as subject matter experts in their fields, they are very familiar with the tasks their students need to master. Moreover, since most teachers are familiar with the resources and procedures in their schools, contextual analysis would typically reveal little. Exceptions to this include moving to a new school or when new procedures are implemented. Regarding learner analysis, many teachers benefit from getting to know what interests and motivates each student. Teachers gain that information through conversations with students and may also ask students to reply to an informal questionnaire at the beginning of the semester or school year. As discussed in Section 2.3, the STARS model for creating STELLAR lessons accounts for these contextual realities, as well as the time limitations that challenge middle school and high school teachers.

2.2.2 Supernova Innovation

As discussed in Chapter 1, perpetuating past technology integration practices will continue to help students learn to use digital technologies for routine consumer uses, but we should not stop there. Let's push technology integration efforts toward innovation to help students become critical thinkers and to enhance their leadership potential. Through development and implementation of STELLAR lessons, guide students to think deeply, to make creative and innovative use of digital technologies, and to enjoy learning.

Multiple educators have called for creativity in student work and continue to do so. Perhaps creative arts and liberal arts educators are chief among them (Gulla & Sherman, 2000; Knutson et al., 2021; Szekely, 2019). Some educators even speak of *creativity education* (Cropley, 2001; Snepvangers et al., 2018). Uniquely, through the features described below, this book seeks to help teachers embellish their lessons with activities that favor attainment of higher-order thinking skills, namely analyze, synthesize, evaluate, and create, through creative use of digital technologies. Implementing those lessons enables teachers to emerge from the black hole of technology integration to supernova innovation, which elevates student learning and leadership potential.

- **Recognition of the Importance of Design Work:** Designing is a form of innovating that requires analysis, synthesis, creativity, and evaluation. Accordingly, design activities will challenge students and present opportunities for leadership. Designers synthesize knowledge originating from multiple disciplines, such as engineering, computing, psychology, economics, and communication, and analyze a variety of factors, including human attitudes and capabilities, physical artifacts, digital technologies, and desired outcomes, in order to devise original solutions to problems. Articulating design solutions using software makes for stellar use of technology.
- **Articulation of STELLAR Paths for Stellar Use of Digital Technologies:** This book identifies seven particular pathways to foster learning through creative use of digital technologies:
  - **S**tyle, Design
  - **T**inker in maker spaces
  - **E**ngage in computational thinking and coding
  - **L**everage datasets
  - **L**ook for patterns in data
  - **A**ctuate, Create, Develop with application software
  - **R**epresent data in visuals
- **Presentation of Four Instructional Scenarios:** Part III of this book offers four instructional scenarios to exemplify stellar lessons, which draw on the combination of technological, pedagogical, and content knowledge. The scenarios address all four core subjects, English, Social Studies, Mathematics, and Science. Each scenario describes a theme in the field of study and provides examples of STELLAR activities for attaining curricular standards in middle school and high school. You may implement the STELLAR activities as described or modify and extend them to suit your students and circumstances.
- **STELLAR Connections to Humanities:** The chapters on Social Studies and English demonstrate the application of multiple STELLAR activities to attainment of particular instructional objectives in the Humanities. For example, data are leveraged and represented in visuals in order to convey populated places. Also, as exemplified, teachers may demonstrate to students how to design and develop apps for playing games to conceal communication and to learn about geography. For teachers inclined to design or develop those games, this work takes you step by step through each of those processes. Moreover, teachers are encouraged to guide students to leverage data to gain insights into language and to increase their understanding of the effects of natural resources on migration, for instance.
- **STELLAR Connections to STEM/STEAM:** In addition to chapters on Science and Mathematics, the STELLAR activities have direct connections to STEAM. Indeed, the first STELLAR activity pertains to styling and design, which connect to Arts and Engineering. Tinkering in maker spaces is also connected to design and engineering. Looking for patterns in data, leveraging datasets, and representing data in visuals are critical in Science and Mathematics. Technologies are often used to facilitate the completion of those tasks. Further, technologies are inherent in computational thinking and coding. Moreover, interacting with digital technologies is necessary for creative work with application software.

- **Diverse Examples for Diverse Teachers:** The examples of STELLAR uses of instructional technologies accommodate learners in different middle school and high school grades while pursuing a wide range of curricular standards across all four core subjects. In addition to addressing the variety of subject matter, this book recognizes the diversity of teachers with respect to technological knowledge. Accordingly, this book presents a variety of simple, moderate, and complex tasks.
- **Flexible Approach to Professional Development:** Since teachers are busy, you need flexibility to acquire technical knowledge and skills over time. Accordingly, you may consider the chapters in Parts II and III in the order you prefer. Further, in Part III, you may choose the STELLAR activities you wish to pursue in each chapter.
- **International Comparisons of Curricula:** Awareness of differences in schooling and curricular standards across countries enhances knowledge of the world and may inspire changes in one's own practices. This book presents differences in curricular standards between England, Canada, and the United States.
- **Annotated Resources:** While guiding students toward stellar use of digital technologies, teachers perform an important filtering function of the vast array of resources available for instruction. The items in the annotated resources section enable teachers to readily access curated resources, which I know to be valuable for particular purposes. The utility of the resource is evident in its annotation.
- **Utilizing Theory in Practice:** Drawing on technological, pedagogical, and content knowledge, teachers plan and implement lessons with digital devices. Chapters 1 and 2 provide theoretical background in the design of instruction for creative use of digital technologies. Leverage that knowledge, then enhance your technological knowledge through Part II, and draw on those knowledge bases, plus your subject matter expertise, in Part III as you consider the practical examples of STELLAR activities.

Technology integration was sufficient in the past, and technology integration practices will continue to help students master rudimentary uses of digital technologies. Push to move beyond routine technology integration; continue through this book and engage students in more innovative work with digital technologies. In so doing, you will guide your students toward critical thinking and leadership opportunities.

2.3 STARS Model for Creating STELLAR Lessons

To this point, discussions of secondary schools and instructional design have enabled us to recognize complexities in teaching, including lesson development. We have considered the diversity of students, teachers, administrators, instructional objectives, and instructional methods. We recognize that multiple technologies are used in a variety of ways for learning. We have also become aware of instructional design models that favor learner analysis, task analysis, and needs analysis prior to the development of instruction. Further, we have determined that contemporary teachers synthesize Technological Knowledge, Pedagogical Knowledge, and Content Knowledge in their teaching practices.

Given that instructional circumstances vary markedly, teachers need to consider the great diversity of students, instructional objectives, and learning environments when creating lessons. Some middle school and high school teachers work in classrooms; some work online; and others work in both environments simultaneously. Due to the variability in instructional circumstances and the need for teachers to draw on multiple knowledge bases simultaneously to function effectively, teaching may be regarded as a complex and ill-structured domain (Leinhardt & Greeno, 1986; Spiro et al., 1988, 1989). In endeavors as complex as teaching and lesson planning, it can help to have a model or framework that captures the essence of the process. The framework serves as a guide that facilitates engagement in the complex activity. To develop stellar lessons in middle school and high school, which foster creativity in learning with technology, consider the STARS model for creating STELLAR lessons:

Specify the Instructional Context
Target STELLAR Activity in the Lesson Plan
Activate the Lesson Plan
Reflect on the Lesson
Synthesize and Refine

Specify the Instructional Context

To create an effective lesson, knowledge of the learners and awareness of the instructional objective are imperative. Consider spending time at the beginning of a course collecting information on learner characteristics and a little time throughout the course gaining additional insights into your students. In particular, I recommend determining the following about your students.

- Capabilities
  - Prior knowledge of the subject matter
  - Linguistic skills
  - Special talents or needs
- Learning and instructional preferences
- Locus of motivation
- Cultural and personal traits

Knowledge of the capabilities of students is important for differentiating instruction (Armstrong, 2018; Doubet & Hockett, 2015; Tomlinson, 2014). By determining prior knowledge about the subject matter, teachers gain initial insights into the range of capabilities of their students. To gain additional insights, collect information about language skills in order to further differentiate instruction (Rodriguez et al., 2014). Students may be fluent in multiple languages, or they may be relatively new to the English language. Further, students may be at advanced, intermediate, or beginning reading levels. Determining fluency in a language requires information about reading, writing, speaking, and listening skills. Discovering special talents or needs of students is also important in order to provide appropriate supports for students with disabilities (Bryant et al., 2020; Krakower & LePage Plante, 2016) and to challenge students with special talents (Kaplan, 2022). Though controversial, Howard Gardner's Theory of Multiple Intelligences (Gardner, 2006, 2021) offers the view that special talents may reside in verbal-linguistic, logical-mathematical, visual-spatial, musical, naturalistic, bodily-kinesthetic, interpersonal, or intrapersonal intelligence.

Regarding learning preferences, it is helpful for teachers to recognize the diversity of learning styles or learning preferences of their students, as discussed in Section 1.2. Further, you may seek to identify the instructional preferences of your students in order to balance group and individual work, for instance. With respect to locus of motivation, identifying the extent to which each student is intrinsically and extrinsically motivated is beneficial. To gain insights into cultural and personal traits in order to personalize and differentiate instruction, some teachers ask students about their family background, as well as hobbies and activities outside of school (Gay, 2018; Herrera, 2016). Since knowledge of learners is critical for effective instruction, some middle school and high school teachers create a profile of each student, which may begin by directing each student to complete a biography card or directing each student to send you an email message containing specific information.

Each lesson seeks to help learners attain a particular capability or capabilities. When developing a lesson, it is helpful to state this explicitly. According to Mager (1962, 1984), the phrasing of the instructional objective makes explicit the behavior to be performed, the conditions for the performance, and the mastery level. For example, "given fractions with uncommon denominators, students will correctly calculate the sum and express it in simplest form 80% of the time." As another example, "given 20 vocabulary terms and four possible meanings for each term, students will select the correct definition 85% of the time." Alternatively, "given 20 vocabulary terms, students will write the correct definition 85% of the time." Some educators also like to identify the target learners explicitly in the statement of instructional objectives, thus forming an ABCD objective, which is the acronym for Audience, Behavior, Conditions, and Degree of mastery (Smaldino et al., 2019). For performances that yield correct or incorrect answers, such as the ones above, writing a pristine Mager objective is straightforward. However, the statement of the instructional objective will not be in such precise terms when students seek to gain capabilities that are not demonstrably correct or incorrect.

Consider, for instance, the goal for students to improve their capability to write a persuasive essay or to write an intriguing poem. In addition to learning to write well, students should continually improve their analytical and problem-solving capabilities. How should such instructional goals be captured in statements of instructional objectives? First, in the case of problem solving, it is important to be specific about the type of problem to be solved and how the problem-solving capability will be demonstrated. For example, solutions to some problems are demonstrated through production and presentation of a mathematical proof, a scientific laboratory report, a prototype for an app, an infographic that uniquely represents data, or a computer program, for instance. In the case of analytical skills, it is also important to be explicit about the object of analysis. For example, an instructional objective might compel students to judge the arguments advanced by opposing attorneys in a particular case, derive alternative game plans for opposing a particular team, or propose a plan for a particular manufacturer to reduce carbon dioxide emissions. When the teacher has determined the manner in which the problem-solving skill or analytical capability will be demonstrated, details about the conditions for the performance and assessment criteria need to be added to the statement of the instructional objective. For subjective ratings, teachers commonly use a scoring rubric that explicitly identifies evaluation criteria with variable levels of attainment in order to increase the validity and reliability of the assessment process. Combining all three elements (condition, behavior, and degree of mastery) for a problem-solving task might yield the following instructional objective: "Given data entry specifications for numbers and text, students will design and create a prototype for an app that presents a visually appealing graphical user interface and ensures accurate data entry, attaining at least a satisfactory rating on each element of the evaluation rubric." Subjective ratings will always be open to debate, especially in the case of artistry (e.g., writing a poem or composing a song), but the existence of a rubric can help to justify ratings and often serves as an instructional support for students.

In addition to knowledge of the learners and a statement of the instructional objective, recognition of the learning environment is critical to the development of effective lessons. Will students complete the lesson in a classroom, online at home, or will some students be in a classroom while others pursue the lesson at home? In the case of online instruction, one additional question arises: Will online students complete the lesson through real-time communication with the teacher and classmates (as in synchronous instruction) or will there be no real-time communication (as in asynchronous instruction)? What instructional resources are available in the learning environment? Those are the critical questions about the learning environment for lessons taught by middle school and high school English, Social Studies, Science, and Mathematics teachers. As discussed in the next section, the learning environment has implications for content delivery and student engagement, as well as resource availability.

As you exit this phase of the model, you should have the following:

1. A statement of the instructional objective;
2. Knowledge of the learners; and
3. Recognition of the learning environment.

Target STELLAR Activity in the Lesson Plan

When planning a lesson, the teacher is compelled to address this question: What presentations, if any, and activities will help the students attain the instructional objective(s)? In our case, since we seek to help learners attain capabilities through creative endeavors with technology, we will select at least one STELLAR activity in order to develop a stellar lesson, an effective, efficient, and at least satisfying lesson, though the best lessons will be joyous and inspirational. With the goal to develop such stellar lessons, we take time in this phase to ponder why students would want to engage in the lesson. From the previous step, recall what you discovered about the learning styles, instructional preferences, the locus of motivation, and personal traits of your students. This is the lesson development phase during which you draw on that knowledge.

Given your knowledge of the learners, the instructional objective, and the learning environment, devise a plan to help learners attain the instructional objective. If you plan a presentation, what will attract the attention of the learners? Consider providing a paradox, a riddle, or a question, for instance. Which STELLAR activity or activities will students pursue? What instructional materials will the students consider? What will students do with the materials? How will students access those materials within the learning environment? How will you determine the extent to which your students attained the instructional objective? How will you know which parts of the lesson the students enjoyed or were inspired by?

As you exit this phase of the model, you should have a lesson plan to help students gain the capability or capabilities identified in the instructional objective or objectives. Your lesson plan should include at least one STELLAR activity and an assessment plan for ascertaining the extent to which the students attained the instructional objective, whether they enjoyed the lesson, and how the lesson can be improved.

Activate the Lesson Plan

During this phase of the STARS model, students engage in the instruction in accordance with the lesson plan. In STELLAR lessons, we know that students will engage in some creative activity with a digital device. The instructor may assume various roles, which may include presenter, guide, and cheerleader. Importantly, as a teacher, you continually monitor student learning.

As you exit this phase of the model, you should have evidence of the extent to which students attained the instructional objective, information about the efficiency of the lesson, and indicators of student satisfaction.

Reflect on the Lesson

In this phase, draw on information collected in the previous step in order to reflect on the effectiveness and efficiency of the lesson. Also, consider student satisfaction. In the best-case scenario, students will have attained the instructional objective or objectives readily and will have been inspired to apply what they learned and seek to learn more. I refer to such effective, efficient, and inspiring lessons as *elegant instruction* (Luterbach, 2013). In some respects, the emotional reaction of the students to the lesson is the most important outcome. After all, inspired students seek to learn more. In contrast, the last thing dissatisfied students want to do is spend more time considering the subject matter. Beyond the lesson and the course, dissatisfied students drop out of school whereas satisfied students enjoy lifelong learning.

As you exit this phase of the model, you should be aware of the strengths and weaknesses of the lesson.

Synthesize and Refine

Teaching is great for learning! Students who have engaged in my lessons have taught me about various features of application software, for instance, and have inspired me to learn more about the topics I teach. They have also mentioned how my instruction could be improved (even, at times, without my prompting). In this phase, having reflected on strengths and weaknesses of the lesson, retain beneficial components of the lesson. In addition, synthesize your growing technological, pedagogical, and content knowledge in order to refine weak aspects of the lesson.

Over time, you might come to favor a small set of instructional methods and hold relatively stable views of education. You might even fall into one of the philosophical groups mentioned at the outset of this chapter. If you're an effective and happy teacher, you may remain in your philosophical camp. Alternatively, if you believe your diverse students might benefit from a greater variety of instructional methods, consider conversing with a fellow teacher or two who might enhance your teaching practice. Your colleagues might share with you an effective teaching practice, a particular STELLAR lesson, for instance.

References

Allen, M. W. (2012). *Leaving ADDIE for SAM: An agile model for developing the best learning experiences*. ASTD Press.

Armstrong, T. (2018). *Multiple intelligences in the classroom* (4th ed.). Association for Supervision and Curriculum Development (ASCD).

Asimow, M. (1962). *Introduction to design*. Prentice-Hall.

Atkinson, R. C., & Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. *Psychology of Learning and Motivation, 2*, 89–195. https://doi.org/10.1016/S0079-7421(08)60422-3

Brown, A. H., & Green, T. D. (2020). *The essentials of instructional design: Connecting fundamental principles with process and practice* (4th ed.). Routledge.

Bryant, D. P., Bryant, B. R., & Smith, D. D. (2020). *Teaching students with special needs in inclusive classrooms* (2nd ed.). Sage.

Cahn, S. M. (2012). *Classic and contemporary readings in philosophy of education* (2nd ed.). Oxford University Press.

Carrasquillo, A., Rodriguez, D., & Lee, K S. (2014). *The bilingual advantage: Promoting academic development, biliteracy, and native language in the classroom*. Teachers College Press.

Carey, J. O., Carey, L., & Dick, W. (2022). *The systematic design of instruction* (9th ed.). Pearson.

Cennamo, K. S., Ross, J. D., & Ertmer, P. A. (2019). *Technology integration for meaningful classroom use: A standards-based approach* (3rd ed.). Cengage.

Chitale, A. K., & Gupta, R. C. (2013). *Product design and manufacturing* (6th ed.). PHI Learning.

Clark, R. E., Feldon, D. F., Van Merriënboer, J. J. G., Yates, K. A., & Early, S. (2008). Cognitive task analysis. In J. M. Spector, M. D. Merrill, J. Van Merriënboer, & M. P. Driscoll (Eds.), *Handbook of research on educational communications and technology* (3rd ed., pp. 577–593). Routledge.

Cropley, A. J. (2001). *Creativity in education and learning: A guide for teachers and educators*. Routledge.

Doubet, K. J., & Hockett, J. A. (2015). *Differentiation in middle and high school: Strategies to engage all learners*. Association for Supervision and Curriculum Development (ASCD).

Duffy, T. M., & Jonassen, D. H. (1992). *Constructivism and the technology of instruction: A conversation*. Routledge.

Ertmer, P. A., & Newby, T. J. (2013). Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance Improvement Quarterly, 26*(2), 43–71.

Frerejean, J., Van Geel, M., Keuning, T., Dolmans, D., Van Merriënboer, J. J. G., & Visscher, A. J. (2021). Ten steps to 4C/ID: Training differentiation skills in a professional development program for teachers. *Instructional Science, 49*, 395–418. https://doi.org/10.1007/s11251-021-09540-x

Gagné, R. M. (1965). The analysis of instructional objectives for the design of instruction. In R. Glaser (Ed.), *Teaching machines and programmed learning, II: Data and directions* (pp. 21–65). National Education Association.

Gagné, R. M., & Briggs, L. J. (1974). *Principles of instructional design*. Rinehart & Winston.

Gagné, R. M., Briggs, L. J., & Wager, W. W. (1992). *Principles of instructional design (4th ed.)*. Harcourt Brace Jovanovich.

Gagné, R. M., Wager, W. W., Golas, K. C., & Keller, J. M. (2005). *Principles of instructional design* (5th ed.). Wadsworth/Thomson Learning.

Gardner, H. (2006). *Multiple intelligences: New horizons*. Basic Books.

Gardner, H. (2021). *Howard Gardner: Books*. https://www.howardgardner.com/books

Gay, G. (2018). *Culturally responsive teaching: Theory, research, and practice (3rd ed.)*. Teachers College Press.

Gulla, A. N., & Sherman, M. H. (2020). *Inquiry-based learning through the creative arts for teachers and teacher educators*. Palgrave Macmillan.

Hall, G. E., Dirksen, D. J., & George, A. A. (2013). *Measuring implementation in schools: Levels of use*. AIR/SEDL. https://sedl.org/cbam/lou_manual_201410.pdf. https://www.air.org/resource/concerns-based-adoption-model-cbam

Herrera, S. G. (2016). *Biography-driven culturally responsive teaching*. Teachers College Press.

Hites Anderson, J. (2010). Collecting analysis data. In K. H. Silber & W. R. Foshay (Eds.), *Handbook of improving performance in the workplace* (Vol. 1, Instructional Design and Training Delivery, pp. 95–143). Pfeiffer.

Hunter, J. (2015). *Technology integration and high possibility classrooms: Building from TPACK*. Routledge.

International Society for Technology in Education. (2016). *ISTE standards for students*. https://www.iste.org/standards/for-students

International Society for Technology in Education. (2017). *ISTE standards for educators*. https://www.iste.org/standards/for-educators

Jonassen, D. H., & Hannum, W. H. (1986). Analysis of task analysis procedures. *Journal of Instructional Development, 9*(2), 2–12.

Kaplan, S. N. (2022). *Differentiated curriculum and instruction for advanced and gifted learners*. Routledge.

Knutson, K., Okada, T., & Crowley, K. (2021). *Multidisciplinary approaches to art learning and creativity: Fostering artistic exploration in formal and informal settings*. Routledge.

Koehler, M. (2012). *TPACK explained*. http://tpack.org/. Click the *What is TPACK?* Button.

Koehler, M. J., & Mishra, P. (2009). What is technological pedagogical content knowledge? *Contemporary Issues in Technology and Teacher Education, 9*(1), 60–70.

Krakower, B., & LePage Plante, S. (2016). *Using technology to engage students with disabilities*. Corwin.

Leinhardt, G., & Greeno, J. G. (1986). The cognitive skill of teaching. *Journal of Educational Psychology, 78(*2), 75–95.

Luterbach, K. J. (2013). Elegant instruction. *Journal of Educational Technology Systems, 41*(2), 183–204.

Mager, R.F. (1962). *Preparing objectives for programmed instruction*. Fearon.

Mager, R. F. (1984). *Preparing instructional objectives* (2nd ed.). Lake.

Merrill, M. D. (1983). Component display theory. In C. M. Reigeluth (Ed.), *Instructional design theories and models: An overview of their current status*. Lawrence Erlbaum Associates.

Merrill, M. D. (1994). *Instructional design theory*. Educational Technology Publications.

Merrill, M. D. (2002). First principles of instruction. *Educational Technology Research & Development, 50*(3), 43–59.

Merrill, M. D., Li, Z., & Jones, M. K. (1990). Limitations of first generation instructional design. *Educational Technology, 30*(1), 7–11.

Mishra, P. & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record, 108*(6), 1017–1054.

Molenda, M. (2003). In search of the elusive ADDIE model. *Performance Improvement, 42*(5), 34–36.

Morrison, G. R., Ross, S. M., Morrison, J. R., & Kalman, H. K. (2019*). Designing effective instruction* (8th ed.). Wiley.

Nicholson, D. W. (2016). *Philosophy of education in action: An inquiry-based approach*. Routledge.

Noddings, N. (2018). *Philosophy of education* (4th ed.). Routledge.

Puentedura, R. (2013). *SAMR: Moving from enhancement to transformation*. Hippasus. http://www.hippasus.com/rrpweblog/archives/2013/05/29/SAMREnhancementToTransformation.pdf

Reigeluth, C. M. (Ed.). (1983). *Instructional-design theories and models: An overview of their current status*. Lawrence Erlbaum Associates.

Reigeluth, C. M. (Ed.). (1999). *Instructional-design theories and models: A new paradigm of instructional theory (Vol. II)*. Lawrence Erlbaum Associates.

Reigeluth, C. M., & Carr-Chellman, A. (Eds.). (2009). *Instructional design theories and models, Volume III: Building a common knowledge base*. Routledge.

Reiser, R. A., & Dempsey, J. V. (2018). *Trends and issues in instructional design and technology*. Pearson.

Romiszowski, A. J. (1981). *Designing instructional systems: Decision making in course planning and curriculum design*. RoutledgeFalmer.

Romiszowski, A. J. (2016). *Designing instructional systems: Decision making in course planning and curriculum design*. Routledge.

Rossett, A. (1995). Needs assessment. In G. Anglin (Ed.), *Instructional technology: Past, present, and future* (2nd ed., pp. 183–196). Libraries Unlimited.

Saxena, M. (2011). Learner analysis framework for globalized e-learning: A case study. *International Review of Research in Open and Distance Learning, 12*(5), 93–107.

Scardamalia, M., & Bereiter, C. (1994). Computer support for knowledge building communities. *The Journal of the Learning Sciences, 3*(3), 265–283.

Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher, 15*(2), 4–14.

Siegel, H., Phillips, D. C., & Callan, E. (2018). Philosophy of education. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2018 edition). https://plato.stanford.edu/archives/win2018/entries/education-philosophy/

Skinner, B. F. (1950). Are theories of learning necessary? *The Psychological Review, 57*(4), 193–216.

Smaldino, S. E., Lowther, D. L., & Mims, C. (2019). *Instructional technology and media for learning* (12th ed.). Pearson Education.

Smith, P. L., & Ragan, T. J. (2005). *Instructional design* (3rd ed.). Wiley.

Snepvangers, K., Thomson, P., & Harris, A. (Eds.). (2018). *Creative policy, partnerships and practice in education*. Palgrave Macmillan.

Spiro, R. J., Coulson, R. L., Feltovich, P. J., & Anderson, D. K. (1988). Cognitive flexibility theory: Advanced knowledge acquisition in ill-structured domains. In V. Patel (Ed.), *Tenth annual conference of the cognitive science society* (pp. 375–383). Lawrence Erlbaum.

Spiro, R., Feltovich, P., Coulson, R., & Anderson, D. (1989). Multiple analogies for complex concepts: Antidotes for analogy-induced misconception in advanced knowledge acquisition. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning* (pp. 498–531). Cambridge University Press. https://doi.org/10.1017/CBO9780511529863.023

Szekely, G. (2019). *Art rooms as centers for design education: Creativity and innovation in K-12 classrooms*. Routledge.

Tessmer, M., & Richey, R. C. (1997). The role of context in learning and instructional design. *Educational Technology Research and Development, 45*(2), 85–115.

Tessmer, M., Wilson, B., & Driscoll, M. (1990). A new model of concept teaching and learning. *Educational Technology Research and Development, 38*(1), 45–53.

Tomlinson, C. A. (2014). *The differentiated classroom: Responding to the needs of all learners*. Association for Supervision and Curriculum Development (ASCD).

Tripp, S., & Bichelmeyer, B. (1990). Rapid prototyping: An alternative ID strategy. *Educational Technology Research & Development, 38*(1), 31–44.

Van Geel, M., Keuning, T., Frerejean, J., Dolmans, D., Van Merriënboer, J., & Visscher, A. J. (2019). Capturing the complexity of differentiated instruction. *School Effectiveness and School Improvement, 30*, 51–67. https://doi.org/10.1080/09243453.2018.1539013.

Van Merriënboer, J. J. G., Jelsma, O., & Paas, F. G. W. C. (1992). Training for reflective expertise: A four-component instructional design model for complex cognitive skills. *Educational Technology Research and Development, 40*(2), 23–43. https://doi.org/10.1007/BF02297047

Van Merriënboer, J. J. G., & Kirschner, P. A. (2017). *Ten steps to complex learning: A systematic approach to four-component instructional design*. Routledge.

Part II

Resources, Technologies, and Techniques for Innovation

3 Analyzing, Designing, Making

Innovative teachers leverage a variety of free resources and technologies, as well as utilize multiple techniques. The remaining chapters in this book present a variety of free digital tools and technologies, as well as demonstrate multiple techniques. This chapter and the next one offer tasks that build fundamental design, development, and analysis skills, which are utilized when engaged in STELLAR activities. The four chapters in Part III also provide opportunities to build design, development, and analysis skills while also considering how STELLAR pursuits will engage students in creative endeavors as they attain learning goals.

Going by the order of the letters in the acronym, the first STELLAR path invites consideration of design thinking and pursuit of design tasks, which we attend to in Section 3.2. The second and sixth STELLAR paths focus on making, both digital creations and objects in maker spaces. Guided by the activities in Sections 3.3–3.10, we develop with text, images, sound, and video, as well as with combinations of those media. The third STELLAR path pertains to computational thinking. In Chapter 4, we consider computational thinking and pursue numerous coding activities, which are at beginner, intermediate, and advanced levels. The fourth, fifth, and seventh STELLAR paths invite reflections on data science and completion of data mining tasks, which we pursue next, in Section 3.1.

3.1 Data Science, Data Analytics, and Data Mining

In the present era, which features networked mobile devices, we have been enjoying and grappling with the growth and proliferation of data on demand. Our data privacy and piracy concerns are real, yet we embrace data and the services they enable, perhaps video streaming, social media communication, and multiplayer gaming chief among them. Presently, due to rapid transfers of digital data, many people work, learn, play, shop, and communicate online.

Data are critical to our lives, and innovating with data pays huge dividends. Companies that have innovated with data now dominate. Twenty years ago, General Motors, Ford, and Boeing were among the largest companies in the United States and the world, but they have been replaced by Apple, Microsoft, Amazon, Facebook, and Google's parent company Alphabet, all of which have innovated with data and/or digital devices. Individuals who innovate with data are also well compensated. A *data scientist* working in the United States in 2021 earned, on average, $96,500, with salaries ranging from $68,000 to $135,000 (PayScale, 2021a).

Like statistics, *data science* is a field of study that embraces data. Additionally, data scientists study computer technologies and seek to solve practical problems in health, humanities, science, sports, education, and business, for instance. *Data science* is closely related to *data analytics*. Indeed, many people use those terms interchangeably. Data science includes studies in *data mining*. Just as exercise enthusiasts invite people to go *spinning*, rather than *pedaling*, you may prefer to speak of *data analytics* or *data mining* because, somehow, they seem to sound more intriguing and fun than data analysis. Since data mining includes machine learning, there may be some level of

DOI: 10.4324/9781003172284-5

intrigue to the term, but *data mining* is an approach to data analysis. Indeed, whether speaking of *descriptive analytics*, *diagnostic analytics*, *prescriptive analytics*, or *predictive analytics* (Kachchi & Kothiya, 2021), data analysis is at the heart of the work. After all, *analysis* is the root word of *analytics*.

Over the past ten years, many universities have created programs of study in data science, and many have a department, or a center or institute focused on data science. For example, Yale University embraces both data science and statistics in their *Department of Statistics and Data Science*. New York University has the *Center for Data Science*; Columbia University, the University of California at Berkeley, the University of British Columbia, and Carlton University have an *Institute for Data Science*; the University of Cambridge has the *Center for Data-Driven Discovery*, the University of Oxford has the *Big Data Institute*, and the University of Edinburgh has the *Data Science Education Center of Excellence*. Data science and data analytics are also closely related to *information science* and *informatics*. Depending on the university, data science programs may be offered in academic units such as a School of Information Science, School of Informatics, School of Engineering, Department of Mathematics, or Department of Computer Science. Further, studies in data science may be offered by academic units within a specialty field, such as heath informatics, health data science, life science analytics, data arts and humanities, and marketing data analytics.

Given the potential for leadership through data analytics, three STELLAR activities make explicit mention of data. As a teacher, when you model for students how to leverage data, represent data in visuals, and look for patterns in data, you enhance their potential to become leaders.

Regarding the goal to leverage data, Section 3.1.1 identifies vast data repositories and discusses a particular search strategy for locating additional data repositories. Section 3.1.2 discusses data mining, including the use of electronic spreadsheets to represent data in visuals and to look for patterns in data.

Before proceeding to the next section, let's recall terms related to databases and terms for expressing quantities of data. In addition, let's seek to gain some perspective on the amount of data created and consumed in the world each year. Commonly, structured data appear in relational databases, such as the systems for student records and business transactions. Structured Query Language (SQL) is used to create, read, update, and delete data in relational databases, as exemplified in the next chapter (Section 4.5). Unstructured data, such as the text, images, audio, and videos in social media and other sites are stored in NoSQL systems, such as *MongoDB* and *Neo4j*. Those data stores, which retain data in raw formats, may be called *data lakes*. In some implementations, data lakes store raw data in a structured manner, which makes it possible to use SQL for certain types of data processing.

Discussions about quantities of data refer to multiples of bytes. Table 3.1 lists the terms used to quantify data using powers of 10. The exact number of bytes would be a power of 2, but since many people typically use powers of 10 and 2^{10} (1024) is relatively close to 10^3 (1000), Table 3.1 includes the number of bytes as a multiple of 10. As you consider each row in Table 3.1, recognize that the term for the row represents 1000 (10^3) times as many bytes as the term above it. If you wish to calculate the exact number of bytes for each term in Table 3.1, you may use a spreadsheet template, either *quantities-of-data.csv* or *quantities-of-data.xlsx* (both files are in the Electronic Resources for this book). Beginning with one byte, multiply each subsequent row by 1024 (2^{10}).

In 2020, International Data Corporation (2020) estimated that approximately 59 zettabytes (ZB) of data would be created and consumed throughout the world. Given that several microcomputers in 2020 shipped with one terabyte (TB) of data, it would take 59 billion such microcomputers to store all of that data. Alternatively, one could conceive of such a large quantity of data in terms of DVD storage. At 4.7 gigabytes per DVD, 1 TB of data could be stored on about 213 DVDs. Multiplying 213 by 59 billion yields 12.567 trillion, which is the number of DVDs needed to store 59 ZB of data. The yottabyte, 10^{24} bytes or, technically, 2^{80} bytes, follows the zettabyte.

Due to the exponential growth of data (International Data Corporation, 2017), conceptions of Big Data continue to shift. A decade ago, one might have perceived of one or two dozen

Table 3.1 Quantities of Data

Term	*Approximate Number of Bytes*		*Storage Capacity*
Byte (B)	1	10^0	one letter, digit, or punctuation mark
Kilobyte (KB)	1000	10^3	175 words
Megabyte (MB)	1,000,000	10^6	75 pages of text; a one-minute song
Gigabyte (GB)	1,000,000,000	10^9	250 songs; 45 minutes of video
Terabyte (TB)	1,000,000,000,000	10^{12}	200 DVDs; one microcomputer's SSD
Petabyte (PB)	1,000,000,000,000,000	10^{15}	200,000 DVDs; 5 billion photos
Exabyte (EB)	1,000,000,000,000,000,000	10^{18}	200 million DVDs; 250 billion songs
Zettabyte (ZB)	1,000,000,000,000,000,000,000	10^{21}	Data consumed worldwide in six days

terabytes of data as Big Data (Manovich, 2012). Even today, that would be plenty more data than most people could store on their microcomputers, which might lead one to regard a few dozen terabytes of data as Big Data. Yet, a small number of terabytes pales in comparison to what some others regard as Big Data, such as the data repositories of large retailers. According to Gour (2020) and Marr (2021), the data cloud at Walmart is capable of processing 2.5 petabytes of data each hour and permits modeling and visualization of 40 petabytes of data. Conceptions of Big Data are in the mind of the beholder and shift over time. At minimum, though, Big Data stores exceed the processing capacities and capabilities of common software and hardware.

Walmart is a gigantic retailer, but there are many other large, medium, and small companies leveraging data for profit. Data are so valuable that some economists speak of the *data economy* (Siegele, 2020). Though many datasets are available for purchase, free datasets are particularly beneficial to teachers and students. The next section identifies such datasets.

3.1.1 Data Repositories

Fortunately, numerous free and open datasets are available to teachers and others. In many cases, data may be downloaded with a single click on a button or a link in a web browser. Other times, data providers require an email address or account creation. For example, as noted in Chapter 5, button clicks on web pages at Natural Earth (https://www.naturalearthdata.com) will download geographic data for rendering maps of Earth. In addition to acquiring data at websites that distribute a particular type of data, teachers and others can acquire free datasets on a wide variety of topics available in massive data repositories. Further, datasets may be discovered by keyword search or by browsing categorized lists. See the website references below to pursue these three options for dataset acquisition.

Massive Data Repositories

Open Data in Google Cloud
https://cloud.google.com/bigquery
https://cloud.google.com/

Open Data on Amazon Web Services
https://aws.amazon.com/opendata/?wwps-cards.sort-by=item.additionalFields.sortDate&-wwps-cards.sort-order=desc

Azure (Microsoft) Open Datasets
https://azure.microsoft.com/en-us/services/open-datasets/

Kaggle
https://www.kaggle.com/

Machine Learning Repository
University of California – Irvine
https://archive.ics.uci.edu/ml/index.php

Open Data – Government of Canada
https://open.canada.ca/en/open-data
https://open.canada.ca/en

Open Data – Government of the United Kingdom
https://data.gov.uk/

Open Data – Government of the United States of America
https://www.data.gov/

Facebook
https://www.facebook.com/data/

DBPedia – A Knowledge Base Extracted from Wikipedia
https://www.dbpedia.org/

Keyword Search for Datasets

In addition to enabling keyword searching for web pages, Google provides a keyword search engine to locate datasets.

Dataset Search by Google
https://datasetsearch.research.google.com/

Categorized Lists

Tyagi (2020) lists data for locating images, audio, and databases for natural language processing, finance and economy, healthcare, scientific research, Aerospace and Defense, eCommerce, and sentiment analysis.

https://towardsdatascience.com/data-repositories-for-almost-every-type-of-data-science-project-7aa2f98128b

Numerous categorized lists of datasets are available in Keegan's (2019) article.

https://medium.com/information-expositions/list-of-lists-of-datasets-c9bf52370755

You may also acquire datasets, as well as contribute datasets, at *Awesome Public Datasets*.

https://github.com/awesomedata/awesome-public-datasets

Once you have a dataset of particular interest to you, enjoy mining it.

3.1.2 Data Mining

Searching for patterns in data is *data mining* (Jamsa, 2021). Both human beings and machine learning algorithms engage in the data mining process. Machine learning algorithms are pattern recognition programs that improve at a task without explicit coding for the task (Norrie, 2019). Using computer code for general pattern recognition and a dataset that includes digitized images of handwritten digits with specification of the digit in each image, a machine learning program can improve at recognizing handwritten digits, for instance. Such a case of machine learning, called *classification*, is known as *supervised learning* because the entity captured in the data is supplied in the training set. In *unsupervised learning*, algorithms group data without specification of prior classifications. This grouping process is referred to as *clustering*. Clustering algorithms may, for instance, learn to distinguish between benign and malignant cells, or may be used in search engines to return a group of web pages closest to the search query. *Reinforcement learning*, which a machine might use to learn a game, is a third approach to machine learning. In conversations about machine learning, you may also hear of *neural networks* and *deep learning*. Goodfellow et al. (2016) describe the good times and the bad times of research into what today is called *deep learning*, but in the 1940s–1960s was called *cybernetics*, and in the 1980s and 1990s was called *connectionism*. For a quick and accessible overview of the types of learning important to machine learning practitioners, see Brownlee (2019).

At the end of this section, we will leverage a free dataset and use a machine learning technique to predict prices of houses. In general, one would use Python, R, or some other programming language to engage in such data mining. We will use a spreadsheet to help conceptualize the input values, often called *features*, and, in this case, the single output value, which is called the *target*. We will also gain a sense for the limitations of spreadsheets for addressing machine learning problems. First, we consider fundamentals of electronic spreadsheets.

Creating a Loan Payment Schedule

The following example uses *Google Sheets*, but the steps can be completed in either *Microsoft Excel* or the *Numbers* app on a Mac. Complete the following steps to create the Loan Payment Schedule in Table 3.2.

1. In a web browser, go to https://sheets.google.com
2. Create a blank sheet by clicking the **Plus** button
3. In Cell A1, enter the title, `Loan Payment Schedule`
4. With the cursor still in Cell A1, set the font size to 14, and, for Bold text, press Ctrl-B on Windows or Command-B on a Mac
5. In Cells A3–A6, enter the following labels: `Initial Principal, Interest Rate, Monthly Payment, Loan Start Date`
6. Increase the width of Column A by positioning the mouse cursor over Column letter A; then click the triangle icon to display the items in the drop-down menu (or right-click in Windows or control-click on a Mac); select **Resize column**; enter a pixel value (e.g., 120); click the OK button
7. In Cells B3–B6, enter the following values: 5000, 0.1, 250, 1/1/2020
8. Select all four Cells B3–B6 (e.g., move the cursor to Cell B3, hold the Shift key down and press the Down arrow key three times); drop down the **Format** menu; then select **Align**; **Right** or click the Horizontal Alignment button and select right alignment
9. Move the cursor to Cell B4, drop down the **Format** menu; then select **Number**; **Percent**
10. In Cells A8–D8, enter the following labels: `Payment Number, Payment Date, Interest, Principal`
11. To enter the first payment number, enter 1 in Cell A9
12. To calculate the first payment date (loan start date plus 1 month), enter `=EDATE(B6, 1)` in Cell B9. (Note: EDATE is a function that increases a date, which in this example is in Cell B6, by a specific number of months, which in this example is 1.)
13. To calculate the first interest payment (principal * rate * time), enter `=B3*B4/12` in Cell C9

14. To calculate the principal after the first payment (principal + interest − monthly payment), enter `=B3+C9−B5` in Cell D9
15. Select Cells C9 and D9, drop down the **Format** menu, then select **Number**; **Number**
16. Select Cells A9–D9, drop down the **Format** menu, then select **Align**; **Right**
17. To automate the entry of Payment Number, Payment Date, Interest, and Principal, we will manually insert the entries for Cells A10–D10. The mathematical expressions for Payment Date, Interest, and Principal will refer to the previous row, Row 9, which will enable automation for all subsequent rows.

    i. Enter 2 in Cell A10
    ii. Enter `=EDATE(B9,1)` in Cell B10
    iii. Enter `=D9*B$4/12` in Cell C10 (see note below for explanation of the $)
    iv. Enter `=D9+C10−B$5` in Cell D10

18. Select Cells C10 and D10, drop down the **Format** menu, then select **Number**; **Number**
19. Select Cells A10–D10, drop down the **Format** menu, then select **Align**; **Right**
20. Now, complete the following steps to direct the spreadsheet to automatically calculate all of the remaining values for Payment Date, Interest, and Principal.

    i. Select Cells B10–D10 (e.g., press arrow keys to move the cursor to Cell B10, hold the Shift key down and press the Right arrow key twice)
    ii. Hover the cursor over the handle in the bottom-right corner of the highlighted cells; when the cursor changes to the Plus sign, click and drag the handle down to Row 30 and then release the mouse button. *Voila*! The values appear for Payment Date, Interest, and Principal in Cells B11–D30

21. Select Cells A9 and A10. Hover the cursor over the handle in the bottom-right corner of the highlighted cells; when the cursor changes to the Plus sign, double-click the mouse button (or your trackpad)

Upon consideration of the values in Row 30, notice that the loan would have been paid off after the 22nd payment, made on November 1, 2021. Further, the final payment should be $7.56 less than the $250 monthly payment.

Table 3.2 Design of the Loan Payment Schedule

	A	*B*	*C*	*D*
1	**Loan Payment Schedule**			
2				
3	Initial Principal	5000		
4	Interest Rate	10.00%		
5	Monthly Payment	250		
6	Loan Start Date	1/1/2020		
7				
8	Payment Number	Payment Date	Interest	Principal
9	1	2/1/2020	41.67	4,791.67
10	2	3/1/2020	39.93	4,581.60
11	3	4/1/2020	38.18	4,369.78

You may wonder why Steps 18iii and 18iv include cell references with and without the $ character. When an electronic spreadsheet copies a mathematical expression down one row, the row numbers in cells referenced in the mathematical expression are incremented by one unless a dollar sign appears in front of them. Hence, when the spreadsheet copied the expression in Cell B10 [i.e., `EDATE(B9,1)`] to Cell B11, the spreadsheet incremented the 9 in the reference to Cell B9, to 10, which set the mathematical expression in Cell B11 to `EDATE(B10,1)`. That expression adds one month to the Payment Date in the previous row. Also, when the spreadsheet copied the expression in Cell C10 (i.e., `D9*B$4/12`) to Cell C11, D9 became D10, but the $ in front of the 4 in B$4 ensured that the row reference, 4, remained the same, which set the mathematical expression in Cell C11 to `D10*B$4/12`. That expression calculates Interest by multiplying the principal balance, retrieved from Cell D10, by the annual interest rate, which is always in Cell B4. The annual interest rate is divided by 12 because the loan payment schedule calculates the state of the loan each month.

Steps 21 and 22 above copied mathematical expressions to subsequent rows. For cases in which a mathematical expression is copied one column to the right, column references are automatically incremented by one (A becomes B; B become C; C becomes D, etc.) unless a dollar sign appears in front of a column letter in a cell reference. With respect to terminology, copying a cell reference without a dollar sign is said to be *relative*, whereas a cell reference with a dollar sign is *absolute*. After all, cell references with a dollar sign remain constant, or *absolute*, whereas cell references without a dollar sign change *relative* to the cell of the original mathematical expression.

By creating the loan payment schedule, you proceeded through all the fundamental skills needed to "crunch" numbers in an electronic spreadsheet. You entered data, both text and numbers. You entered mathematical expressions to calculate the interest and the balance remaining on the principal. (If you wish, you could change the heading in Cell D8 from *Principal* to *Balance*.) You also used an expression with a built-in function, which automatically generated the payment dates. Importantly, you also directed the spreadsheet to copy the expressions you entered to subsequent rows, which involved both absolute and relative cell addressing. As a bonus, though you could have entered `=B9+1` into Cell A10 and then copied it along with the expressions in Cells B10–D10 in Step 21, you learned that spreadsheets can automatically increment row numbers.

Next, as adventurous data miners, we seek patterns in data.

Seeking Patterns in Data

One may use descriptive statistics, visualization, or machine learning algorithms to find patterns in data. To seek patterns using those techniques, we will leverage an extant dataset, namely the *student-por.csv* file from the *Student Performance Data Set*, which was collected and shared by Cortez and Silva (2008) to analyze grades in Portuguese and Math. Complete the following steps to acquire the data.

1. Open a web browser and go to https://archive.ics.uci.edu/ml/datasets/student+performance
2. Click the *Data Folder* link
3. Click the *student.zip* link (avoiding the *student.zip_old* link)
4. Unzip the file and locate the *student-por.csv* file. (The file contains grades in a course in Portuguese. Math grades are available in the *student-mat.csv* file. Descriptions of the fields in the dataset appear in the *student.txt* file.)

In the *student-por.csv* file, the data fields are separated (or delimited) by the semi-colon character. Open the file in *Google Sheets* by completing the following steps.

1. In a web browser, go to https://sheets.google.com
2. Click the Plus button to open a new spreadsheet

3. Drop down the **File** menu and select **Import**
4. Click the **Upload** tab
5. **Drag** and **Drop** the *student-por.csv* file as prompted (or click the **Select a file from your device** button to locate and select the *student-por.csv* file). When the *Import file* dialog displays, select Custom from the **Separator type** drop-down menu, then enter the semi-colon character; in the Custom separator field; and click the **Import data** button

The file contains 650 rows, one row of labels for each column followed by 649 rows of data, one row of data for each student. There are 33 columns, beginning with a school code, gender code, and age. In the steps below, though, we will focus on the final three columns, labelled G1, G2, and G3, which contain the grade attained by each student in the Portuguese course for Terms 1, 2, and 3. The third term, G3, is the final grade. In the Machine Learning section below, we will seek to predict student final grade using the grades attained in Terms 1 and 2. First, we seek patterns through descriptive statistics and visualization.

Descriptive Statistics

Observations recorded in many datasets tend to be close to a central value. Using descriptive statistics, we can identify three measures of central tendency, namely the *mean* (which is often called the average), *median*, and *mode*. In addition to those measures, patterns may emerge in the spread or dispersion of data. With respect to spread, descriptive statistics typically include the *range* of data, namely the minimum and maximum values, as well as the *standard deviation*. To gain a sense for the patterns in Term 1, Term 2, and final grades in the Portuguese course, complete the following steps, which will produce the values shown in Table 3.3.

1. Move the spreadsheet cursor to the bottom row (e.g., press Ctrl Down arrow on a Windows PC or Command Down arrow on a Mac)
2. Below the sheet, notice the **Add** button and a field for specifying the number of rows to add (you may need to press the Down arrow key once or twice more to see the button and field if not evident immediately)
3. Replace the default number of rows with 50 and click the **Add** button to add 50 rows
4. In Cell AD652, enter `Mean`; In Cell AD653, enter `Median`; In Cell AD654, enter `Mode`
5. In Cell AD656, enter `Minimum`; In Cell AD657, enter `Maximum`; In Cell AD658, enter `St. Deviation` (unlike the font sizes of these labels in Table 3.3, retain the default font size in *Google Sheets*)
6. Move the spreadsheet cursor to Cell AE652, which leaves one blank row between the last value in the final grade column (G3) and the cursor
7. Enter `=AVERAGE(AE2:AE650)`
8. To restrict the number of decimal places displayed, set the numeric format of the cell. For example, with the spreadsheet cursor in Cell AE652, drop down the **Format** menu and select **Number**; **Number**. In *Google Sheets*, the default is two decimal places, but sometimes you may prefer zero, one, or three decimal places, for instance. To set the number of decimal places displayed in *Google Sheets*, you create a custom format. Drop down the **Format** menu and select **Number**; **More Formats**; **Custom number format**; and change 0.00 to 0 for zero decimal places; 0.0 for one decimal place; and 0.000 for three decimal places. In Cell AE652, set the custom format to 0.0 and click the **Apply** button
9. In Cell AE653, enter `=MEDIAN(AE2:AE650)`
10. In Cell AE654, enter `=MODE(AE2:AE650)`
11. Select the three cells from Cell AE652–Cell AE654. Hover the cursor over the bottom-right handle; when it changes to the plus sign, drag the handle two columns to the right. Now

we have the Mean, Median, and Mode for the grades in Terms 1, 2, and 3. Does the mean increase? Does the mean increase each term? Does the median increase from the first to the last term? Does the mode increase from the first to the last term? Answers to those questions are yes, and they reveal patterns in the data.

12. In Cell AE656, enter `=MIN(AE2:AE650)`
13. In Cell AE657, enter `=MAX(AE2:AE650)`
14. In Cell AE658, enter `=STDEV(AE2:AE650)`
15. Set the format of the number in Cell AE658 to two decimal places
16. Select the three cells from Cell AE656 to Cell AE658 and drag the bottom-right handle two columns to the right in order to copy the mathematical expressions entered in Steps 10–12 to Columns AF and AG. Now we have the range and standard deviation for the grades in Term 1, Term 2, and the final grades. Do you notice any patterns? Given the increase in the standard deviation each term, in my view, variance in student knowledge of Portuguese increased as the course progressed.

Representing Data in Visuals

Visual representations of data often reveal patterns in data. First, we will create a pie chart in seeking a pattern in the gender of the participants. Then we will look for patterns in frequency distributions of grades. Complete the following steps to create the pie chart.

1. Select Cells B2 through B650 (e.g., move the cursor to Cell B2, then scroll down to Row 650; hold the Shift key down and click Cell B650)
2. Drop down the **Insert** menu and select **Chart**. A chart will appear, but the **Chart type** often needs to be changed. In the **Chart editor** dialog on the right, notice the two tabs, **Setup** and **Customize**. With the **Setup** tab selected, drop down the menu for **Chart type**, scroll down and select **Pie chart**. The relative sizes of the pie slices visually depict the unequal numbers of females and males in the study. The percentages reveal that in the study, the ratio of females to males is nearly 3:2.
3. Click the **Customize** tab to reveal the list of accordion style menus. Click **Chart & axis titles**. In the Title text field for Chart title, enter `Participants by Gender`
4. Once again, click **Chart & axis titles**, this time to close that menu. Then open the Pie chart options by clicking **Pie chart**. In the *Slice label* drop down menu, select **Value and Percentage**. (That duplicates the display of percentages, but that redundancy is eliminated in the next step.)

Table 3.3 Descriptive Statistics for Student Performance in Portuguese Course

	Z	*AA*	*AB*	*AC*	*AD*	*AE*	*AF*	*AG*
650	1	3	4	5	4	10	11	11
651								
652					Mean	11.4	11.6	11.9
653					Median	11	11	12
654					Mode	10	11	11
655								
656					Minimum	0	0	0
657					Maximum	19	19	19
658					St. Deviation	2.75	2.91	3.23

5. Close the **Pie chart** options and open the **Legend** options. In the *Position* drop-down menu, select **Top**. If you wish, you may close the **Legend** options. Optionally, you may open the **Pie slice** options and change the colors of each slice. To do so, select the slice from the drop down (in this case, either **F** or **M**); then click the **Color** button and select the color you wish. It is also possible to move a slice from the center in order to make it appear distinct, but this effect is better utilized when the pie chart contains three or more slices.
6. Charts sometimes cover a portion of the data; in those cases, you may drag the chart to a different location in the spreadsheet.

A frequency distribution of a field enables one to find any patterns in the dispersion of the data. Visual depictions of dispersion are especially helpful in large datasets. Complete the following steps to create frequency distributions for the grades the study participants earned in each term, as shown in Table 3.4.

1. In Cell AD660, enter `0`; in Cell AD661, enter 1
2. Select Cells AD660 and AD661; hover the cursor over the handle at the bottom-right of the selected cells; when the cursor changes to the plus sign, drag the handle to Row 680 and release the mouse button to view the numbers 0 through 20. Even though 19 is the maximum grade in the dataset, the documentation indicates that students may earn a grade of 20 at most.
3. In Cell AE660, enter `=COUNTIF(AE$2:AE$650,"=0")` in order to count the number of times zero appears in the cells from AE2 through AE650. Then drag the handle in the bottom-right corner of Cell AE660 across Cells AF660 and AG660. Notice the pattern: More students were assigned 0 as the course progressed. It appears that over time, the motivation to complete work diminished for some students. Also important are dispersion patterns within each term. The next step considers options for generating those frequencies.
4. Move the cursor to Cell AE660. Hover the cursor over the handle in the bottom-right corner; when the cursor turns to a plus sign, drag the handle down to Cell AE680. Move the cursor through Cell AE661, then press the Down arrow key a few times to proceed through Cell AE662, AE663, and so on to notice that the COUNTIF function remained precisely the same in every cell and hence the frequency count for zero is displayed in Cells AE660 through AE680. However, the number of times 1 appears in the AE2 through AE650 range should be calculated and displayed in Cell AE661. Hence, in Cell AE661, we could replace "=0" with "=1" and in Cell AE662, we could replace "=0" with "=2", and continue in that manner down to Cell AE680, replacing "=0" with "=20". Even though manually editing the search string would work in this case without expending too much time, such an inelegant (or "brute force") approach is not recommended when the process can be automated. Just imagine a case in which frequencies for 100 or 1000 values were sought. Instead, we will direct the spreadsheet to create the necessary search string using the built-in function, CONCAT (an abbreviation for concatenate), which in *Google Sheets* (and other spreadsheets) puts together two values. In our case, we seek to string together the equal sign and grade. In Cell AE660, replace `"=0"` with `CONCAT("=",$AD660)` in order to form `=COUNTIF(AE$2:AE$650,CONCAT("=",$AD660))` which will automatically create the necessary search string `"=0"` by putting together the values of the two parameters of the CONCAT function, the equal sign `("=")` and the value of Cell AD660 (`0`).
5. Now we merely direct the spreadsheet to copy and paste the revised COUNTIF function. Move the cursor to Cell AE660. Hover the cursor over the handle in the bottom-right corner; when the cursor turns to a plus sign, drag the handle down to Cell AE680 and release the mouse button. Now we have the frequency distribution for grades earned in the first term. To direct the spreadsheet to calculate the frequency distributions for the second and

final terms, select Cells AE660 through AE680. Hover the cursor over the handle in the bottom-right corner; when the cursor turns to a plus sign, drag the handle across to Cell AG680 and release the mouse button

The values in the frequency distributions in Columns AE, AF, and AG of the spreadsheet, as shown in Table 3.4, reveal patterns. In particular, relatively few students earned less than 6 or greater than 18 in any term. There is some variation between terms. For example, relatively few students earned more than 17 in the first term. Also, relatively few students earned less than 7 in the final term. The values in the frequency distributions also reveal that in the first term, increasing numbers of students earned 4 through 10, and then decreasing numbers of students earned 11 through 19. The increasing and decreasing patterns of values for the second and third terms are similar, though the peak is 11 for these terms, rather than the peak of 10 in the first term. The magnitude of the change from one grade to the next within each term can be estimated from the values, if given sufficient time. For example, in the first term, 9 more students earned 8 than 7 (42–33), which is a substantial increase since 9 of 33 represents an increase of more than 25%. From the raw values, each fluctuation from grade to grade takes some time to discern and is subject to error. On the other hand, when values are represented visually, nuances in data, such as the degree of fluctuation from one grade to the next, may seem easier to

Table 3.4 Frequency Distributions of Grades Earned by Term

	AC	*AD*	*AE*	*AF*	*AG*
660		0	1	7	15
661		1	0	0	1
662		2	0	0	0
663		3	0	0	0
664		4	2	0	0
665		5	5	3	1
666		6	9	7	3
667		7	33	16	10
668		8	42	40	35
669		9	65	72	35
670		10	95	83	97
671		11	91	103	104
672		12	82	86	72
673		13	72	80	82
674		14	71	54	63
675		15	35	38	49
676		16	22	25	36
677		17	16	20	29
678		18	7	14	15
679		19	1	1	2
680		20	0	0	0

recognize. For comparative purposes, complete the following steps to create visual representations of the frequency distributions.

1. Move the cursor to Cell AD660; then press the Shift key down and click Cell AE680
2. Drop down the **Insert** menu and select **Chart**. Next, view a variety of chart types. In the **Chart editor** dialog on the right, first select Line Chart from the **Chart type** drop-down menu. Notice the other settings in the **Chart editor**: The Data range is AD660:AE680. In this example, the x-axis values are set to possible grades, 0–20, which appear in Cells AD660–AD680. The values for Series (which are the values plotted on the y-axis) are in Cells AE660–AE680. Regarding the checkboxes at the bottom of the dialog, in this example, only the “Use column AD as labels” should be checked.
3. Drop down the **Chart type** menu again and select Area chart (scrolling may be necessary; hovering the mouse cursor over each chart image will display the name of the chart type). Then select Smooth line chart for Chart type. Next, once again, select Area chart. Note that the Area chart used a smooth line this time because the Smooth line chart was selected before Area chart. As experienced above, straight lines appear in an Area chart when Area chart is selected after Line chart. Lastly, for **Chart type**, select Column chart
4. Click the **Customize** tab. In *Chart & axis titles*, set the Chart title to `Distribution of Grades in Term 1`. Then set the *Horizontal axis title* to Grade and the *Vertical axis title* to `Frequency`
5. The relative heights of the frequency bars make comparisons of changes from grade to grade apparent without calculation. For example, given the heights of the bars for number of students who earned grades of 14 and 15, it appears that about twice as many students earned a grade of 14 rather than 15. (I recommend checking the values for those two frequency bars to confirm that claim.) Overall, the relative heights of the bars enable one to perceive the degree of fluctuation from grade to grade.

You may repeat steps 1–4 to create Column charts or other types of charts for the frequency distributions for the second term and the final term, or even all three terms simultaneously.

Machine Learning

In this section, we seek to determine whether the data exhibit a pattern useful for predicting final grade based on both term grades. To predict final grade based on either the first or second term grade, we could use the technique called *simple linear regression*, which is one machine learning method among many. Since there is one predictor variable in each of those cases, the linear regression technique is referred to as simple. However, in the multivariate case, the word *simple* is dropped because two or more predictor variables, commonly called *features* by data scientists, are used to predict the *target*, which in this case is final grade. Linear regression assumes that the input and output variable or variables have a linear relationship. When false, the statistical measure called R-squared (R^2) is low (e.g., closer to zero than one). The value of R-squared is quite useful because it represents the proportion of variance of the dependent variable (final grade in this example) that is accounted for by the independent variable or variables, which in this example are the predictor variables (i.e., *features*). Theoretically, if R-squared is 1.0, *features* will predict the *target* perfectly. In datasets of real observations, perfection is not expected. In this example, we should not expect to predict perfectly the final grades of students based on their first and second term grades. The question is whether the data will permit even decent predictions, which in linear regression, are made based on the coefficient calculated for each predictor variable.

Simple linear regression yields the slope (b_1) and the y-intercept (b_0). This enables prediction by multiplying the slope by the value of the predictor variable and adding the y-intercept, which is captured by the following equation.

$$\hat{y} = b_0 + b_1 x$$

Multivariate linear regression yields a coefficient for each of *n* input variables, $b_1, b_2, \ldots b_n$, and the y-intercept (b_0). This enables prediction by adding the y-intercept to the sum of the products of each predictor variable and its coefficient, which is captured by the following equation.

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n$$

We will apply that equation to make final grade predictions based on each student's grades in the first and second term. Rather than continue in the current sheet, we will repeat the steps to open the *student-por.csv* data in a new spreadsheet and then eliminate all but the final three columns.

1. Open a web browser and go to https://sheets.google.com
2. Click the **Plus** button to open a new spreadsheet
3. Drop down the **File** menu and select **Import**
4. Click the **Upload** tab
5. Drag and drop the *student-por.csv* file as prompted. When the Import file dialog displays, select Custom from the **Separator type** drop-down menu, then enter the semi-colon character; in the *Custom separator* field and click the **Import data** button
6. Select cells A1 through AD1 (e.g., with the cursor in Cell A1, hold the Shift key down and click the right arrow until arriving at Cell AD1)
7. Drop down the **Edit** menu and select **Delete columns A–AD**

In machine learning applications, the original dataset is divided into two datasets, one for training and the other for testing. To accomplish this, we will assign a random number between 1 and 100 inclusive to each row of data. Then we will filter the data such that about 80% of the data will be used for training and 20% of the data used for testing. We will also copy the training dataset and the testing dataset to separate spreadsheets.

1. In Cell D2, enter `=RANDBETWEEN(1,100)`
2. Hover the cursor over the handle in the bottom-right corner of Cell D2; when the cursor changes to the Plus sign, double-click the handle
3. To ensure that those random numbers do not change when you complete the steps below, make sure the cursor is in Cell D2; scroll to the bottom of the sheet; hold the Shift key down and click Cell D650; copy the values (e.g., use your keyboard shortcut, Ctrl-C or Command-C, or drop down the Edit menu and select Copy); drop down the Edit menu; select *Paste specia*l and then select *Paste values only*
4. Move the cursor to Cell E1 and insert a column to the right by dropping down the Insert menu and selecting Column right
5. Move the cursor to Cell F2, and enter `=FILTER(A2:D650,D2:D650>=21)`
6. With the cursor still in Cell F2, scroll down to the last row of data in Column H (which might be close to Row 520 because 80% of 650 is 520). As you scroll down you may wish to verify that the filtering function has selected only rows with a random number greater than or equal to 21 (i.e., every value in Column I should be greater than or equal to 21). When you reach the last row of data in Column H, hold the Shift key down and click the cell containing the last number in Column H

7. Copy the data (e.g., keyboard shortcut, Ctrl-C or Command-C, or drop down the **Edit** menu and select **Copy**)
8. At the bottom of the spreadsheet, notice the sheet tab labelled student-por. Click the **Plus** button to the left of that tab to add a new sheet
9. Move the cursor to Cell A2
10. Drop down the **Edit** menu; select *Paste special* and then select *Paste values only*
11. At the bottom of the spreadsheet, two tabs now appear, the original one and a second one with the default name of the new sheet. Replace the default name by clicking the triangle button next to the default name; select Rename and enter `TrainingSet`
12. Enter the field names, G1, G2, and G3 into Cells A1, A2, and A3 respectively
13. Click the student-por tab to return to the original spreadsheet
14. Move the cursor to Cell K2 and enter `=FILTER(A2:D650,D2:D650<21)`
15. With the cursor still in Cell K2, scroll down to the last row of data in Column M (which might be close to Row 130 because 20% of 650 is 130). As you scroll down you may wish to verify that the filtering function has selected only rows with a random number less than 21. When you reach the last row of data in Column M, hold the Shift key down and click the cell containing the last number in Column M
16. **Copy** the data
17. Add a new sheet. As before, at the bottom of the spreadsheet, click the **Plus** button to the left of tabs containing the sheet names
18. Move the cursor to Cell A2
19. Drop down the **Edit** menu; select *Paste special* and then select *Paste values only*
20. Replace the default name by clicking the triangle button next to the default name; select Rename and enter `TestSet`
21. Optionally, **Drag** and **Drop** the `TestSet` tab to the right of the `TrainingSet` tab
22. Enter the field names, G1, G2, and G3 into Cells A1, A2, and A3 respectively

It is customary in data science projects to explore the data before training the model (Lau, 2019). Since this case seeks to use linear regression, it would be helpful to determine whether the input and output variables have a linear relationship.

1. In the TrainingSet, move the cursor to Cell B2. Scroll down to the last row of data; while holding the Shift key down, click the cell containing the last value in Column C
2. Drop down the **Insert** menu and select **Chart**
3. For **Chart type**, select *Scatter chart* from the drop-down menu
4. Click the **Customize** tab and click *Series* to open those options; scroll down and click the Trendline checkbox

The data appear to suggest a linear relationship between the second term grade and the final grade. Additionally, you may insert the CORREL function to calculate the correlation coefficient. For example, if the last row of data in your training set is at Row 520, you could direct the spreadsheet to calculate the correlation coefficient between second term grade and final grade by entering the following into any empty cell (e.g., Cell D520).

```
=CORREL(C2:C520,B2:B520)
```

Replace both references to Row 520 in the expression above with the row number of the final row of data in your training set. You may also calculate the correlation coefficient between first term grade and final grade, as well as create a Scatter plot for those data.

Since there appears to be a linear relationship between at least one input variable (*feature*) and the output variable (*target*), we proceed with the regression analysis. As before, if the last row of data in your training set is not Row 520, replace the 520 in the cell references in Step 2 below to the row number of the final row of data in your training set.

1. Move the cursor to Cell E2
2. Enter `=LINEST(C2:C520,A2:B520,TRUE,TRUE)`

The first parameter in the LINEST function is the range of cells for the target and the second parameter is the range of cells for the features. The last two parameters are optional, but setting the third one to TRUE yields the y-intercept, and setting the fourth parameter to TRUE produces statistical results in addition to the regression coefficients and the y-intercept. In this example, with the LINEST function in Cell E2, the coefficients and y-intercept (i.e., b_2, b_1, and b_0) appear in Cells E2, F2, and G2, as displayed in Table 3.5. The standard error of each coefficient and the y-intercept appear one row below, in Cells E3, F3, and G3 in this example. As always, the smaller the error, the better. In this case, we can be optimistic about the accuracy of predictions because the errors of the coefficients are low (0.03653 and 0.03977). We can also be optimistic about predictions because R^2, which is in Cell E4, is relatively high at 0.844 (1.0 is the maximum). The other values are beyond the scope of this analysis. The characters #N/A which actually appeared in Cells G4–G6 in the spreadsheet, are to be ignored and hence were not included in Table 3.5.

Given the coefficients in Table 3.5 and recalling the prediction equation, $\hat{y} = b_0 + b_1x_1 + b_2x_2 + \cdots + b_nx_n$, we predict final grades as follows: $\hat{y} = -0.103813886 + 0.1525777799x_1 + 0.8904478892x_2$. In this example, x_1 is first term grade and x_2 is the second term grade, and those variables are in Columns A and B in the spreadsheets. For our purposes, it will be fine to round to the hundredths place. Complete the following steps to calculate the predicted scores, and to assess their accuracy.

1. Click the TestSet tab to open that sheet
2. Move the cursor to Cell D2
3. Enter `=-0.10+0.15*A2+0.89*B2`
4. Hover the cursor over the handle in the bottom-right corner of Cell D2; when the cursor changes to the Plus sign, double-click the handle
5. One technique to determine the extent to which the predicted values differ from the actual value is to take the absolute value of the difference between the actual and predicted scores, and then consider the ratio of error to actual score. In Cell E2, enter =ABS(C2-D2)
6. Hover the cursor over the handle in the bottom-right corner of Cell E2; when the cursor changes to the Plus sign, double-click the handle

Table 3.5 Results of the Linear Regression

	E	*F*	*G*
1			
2	0.8904478892	0.1525777799	-0.103813886
3	0.03653040461	0.03977261437	0.2452827869
4	0.8443641693	1.269313278	
5	1402.428584	517	
6	4519.063014	832.9677548	

7. In Cell F2, enter `=E2/C2`
8. Hover the cursor over the handle in the bottom-right corner of Cell F2; when the cursor changes to the Plus sign, double-click the handle
9. Lastly, use the AVERAGE function to calculate the mean of the values in Column F. As you scroll down Column F, delete all instances of #DIV/0!, if any, to ensure that the mean (average error) is calculated

Overall, the predictions were about 6% off the actual grades earned. One might hypothesize, then, that if the instructional methods are retained in the Portuguese course, final grades of future students might be quite accurately predicted as soon as second term grades are available.

3.2 Design Thinking

The renowned designer David Kelley – founder of d.school, the Hasso Plattner Institute of Design at Stanford University, and founder of the global design company, IDEO – speaks of design thinking as a method for generating breakthrough ideas that are new to the world. According to Kelley, breakthrough ideas typically emerge when designers are engaged in a complex project or when working to resolve a difficult problem (Camacho, 2016). In such light, design thinking is a form of creative problem solving. In the words of Tim Brown (n.d.), Executive Chair of IDEO: "Design thinking is a human-centered approach to innovation that draws from the designer's toolkit to integrate the needs of people, the possibilities of technology, and the requirements for business success."

Design thinking involves creativity, and everyone has creative capacity (Kelley & Kelley, 2013). Further, according to Kelley, design thinking requires engagement in a learning and discovery process with a multidisciplinary team. Even though Kelley regards a multidisciplinary team as an essential element of the design thinking process on large and complex projects, he appreciates the contrasting individual approach to design undertaken by one person, such as an architect (Camacho, 2016).

With respect to the design thinking process, Kelley is keen to emphasize the importance of a human centered approach in which the designers repeatedly seek input from a select group of primary stakeholders, first to understand how the stakeholders perceive the problem and then to engage with them toward a solution. In Kelley's experience, this human centered approach typically leads to a reframing of the problem (Camacho, 2016). Nevertheless, many designers seek to resolve problems different from the ones Kelley has encountered. Indeed, many design processes exclude reframing. To frame or reframe problems in design thinking is an open question in the design field (Beckman, 2020). Even though designers pursue resolution of vastly different problems, in myriad ways, there is plenty of overlap in design thinking processes offered by practitioners and researchers. For example, the Hasso Plattner Institute Academy (2021) at the University of Potsdam in Germany offers a design thinking process comprised of the following six phases.

- Understand
- Observe
- Define Point of View
- Ideate
- Prototype
- Test

Engagement in a particular phase might well invite a return to a prior stage before proceeding to a subsequent phase. For example, observation might lead to a new understanding of the stakeholders and the problem, which might then propel the designer to define the point of view and perhaps reframe the problem and return to the observation stage.

Similarly, the Nielsen Norman Group offers a process model for design thinking with six phases nested within three fundamental processes, namely *understand*, *explore*, and *materialize*. Though no mention of reframing appears in the elaboration of the Nielsen Norman process model of design thinking (Gibbons, 2016), the process includes many of the six phases above.

- Understand
    - Empathize
    - Define
- Explore
    - Ideate
    - Prototype
- Materialize
    - Test
    - Implement

As before, Gibbons (2016) speaks of the phases as non-linear and iterative, and also emphasizes a human centered approach, referred to variously as human-centric, user-centric, and user-centered. According to Gibbons (2016), resolving problems with a hands-on and user-centric mindset leads to innovation, which could result in a competitive advantage. Additional perspectives on the principles, practices, techniques, and mindsets of design thinking are available in Carlgren et al. (2016) and at the American Library Association (2018), for instance.

As an aside, one might expect that any type of design, such as fashion design, aircraft design, visual design, user experience design, and instructional design, would include elements of the general design processes noted above. Indeed, steps in the ADDIE model of instructional design, discussed in the previous chapter, are also in the design models above. In particular, the ADDIE model includes front-end analysis to gain insights into the problem, as well as a step during which the instructional designer or designers draw on their understanding of the problem to devise instruction, which is then developed, tested, and revised. In Section 4.1 of the next chapter, we will also find similarities in the phases of design processes and the stages in a general approach to problem solving.

3.2.1 Visual Design

Graphic designers take as input spoken and written expressions about information to be conveyed to a target audience and conjure up a plan to present the information visually and effectively. This might involve presentation of one or more pictures (figures or images), a display of text, or a layout including both pictures and text. As a teacher, it is helpful to create lesson plans and instruction in accordance with the following four visual design principles.

- Alignment
- Proximity
- Consistency
- Contrast

Regarding alignment, I chose to present the four principles in the bullet list above, not centered, as below.

Alignment
Proximity
Consistency
Contrast

Centering is popular among non-designers but is a poor choice for the list of items above. Adding a bullet point before each of the four centered items above makes the visual impression worse. The misalignment of the four words above is already troublesome; sticking a bullet point in front of each word in the list demonstrates exceedingly poor form.

- Alignment
    - Proximity
- Consistency
    - Contrast

In the three presentations of the list above, the items are in close proximity, which is good because the items are related, each one being the label of a visual design principle. Keeping related items close together favors effective message delivery.

Regarding contrast, the inclusion of a few bullet points on a page is often quite apparent because the many letters on the page are commonplace and hence unremarkable visually. When you want to be sure that something is noticed visually, go big on contrast, just as typesetters do, for instance, with contrasting font sizes and weights for headings versus prose.

Lastly, consistency is critical to visual appeal. In a bullet list, each bullet point should appear in the same column. In addition, the first letter beside each bullet point should be positioned in the same column. Even a tiny violation of consistent formatting is noticeable.

For a visual presentation to appear professional, it is important for beginners to adhere to all four of those principles. There are other visual design elements, such as balance and unity, plus a host of guidelines pertaining to color, but alignment, proximity, consistency, and contrast are a fine starting place. After gaining some visual design experience, one might intentionally bend a rule to achieve a particular effect, but novices can produce satisfying designs by upholding principles of alignment, proximity, consistency, and contrast. Novice designers might also keep in mind the adage "Keep it Simple." In *The Non-Designer's Design Book*, Robin Williams (2015) presents examples of those four principles concisely. Just note that she refers to *consistency* as *repetition*. After presenting those principles, Williams (2015) discusses matters pertaining to typography. Even though commercial word processing programs typically include a wide variety of fonts or typefaces, most designers will eventually seek a custom font for a specific purpose. Many fonts are free of financial cost and can be downloaded from sites such as dafont.com, fontspace.com, and fonts.google.com. To make such fonts available within your word processing software and other apps, the font file (which might have a file name extension such as .otf, .ttc, .ttf, .fnt) need only be included in the fonts folder. The path to the fonts folder in Windows is C:\ Windows\Fonts, and in MacOS the path is /System/Library/Fonts.

For visual design tips pertaining to color, consider the brief overviews provided by GCF Global (n.d.) and XtremeBrandMakeover (2010). For insights into research on color, see Jonauskaite et al. (2020), for instance.

3.2.2. User Experience Design

User Experience Design (UXD) professionals seek to create effective, efficient, and satisfying experiences. They realize this requires attention to human factors. Early studies of people using computers, conducted at Xerox Parc, resulted in the change from command line interfaces to graphical user interfaces (Hill, 2000). Such positive impact contributed to the rise of a discipline, pursued by both practitioners and researchers, focused on Human Computer Interfaces (HCI). Don Norman, a distinguished leader in the discipline, realized that satisfying user experiences involved more than the appearance of the visual interface. Users will gladly tap a couple of well-designed buttons to proceed through a cascading menu, but when doing so appears to freeze the device, perhaps owing to a network error not reported to the user, the experience will be unpleasant in the end. Rather than focus only on the human computer interface, Don Norman encouraged consideration of the overall user experience, and this prompted the change in the characterization of the discipline from HCI to User eXperience Design (UXD) or just UX (Norman et al., 1995). UXD is a dynamic multidisciplinary field that draws on psychology, sociology, cognitive studies, computer science, graphic design, sound design, and interaction design (Gothelf & Seiden, 2016; Travis & Hodgson, 2019). As discussed below, UX professionals follow a general design strategy and multiple heuristics ("rules of thumb") when designing an app. For their efforts, according to PayScale (2021b), UXD professionals in the United States earned, on average, $75,000 in 2021; the salary range was $50,000 to $109,000.

The User Experience Professionals Association International (2000) published a four-phase multistep guide for engaging in UX projects. In light of the guide, UX professionals appear to engage in the following activities within the four phases, recognizing that the real world imposes limitations that rarely permit pristine implementation.

1. **Analysis**: Complete a task analysis and user analysis in order to create user profiles, user scenarios, and user performance requirements, including statements of usability tasks
2. **Design**: Engage in brainstorming and multiple rounds of prototyping; refine prototypes in accordance with results gleaned from usability testing
3. **Implementation**: Work closely with developers while moving from the high-fidelity prototype to the fully functioning system; continue with usability testing and revise the system
4. **Deployment**: Conduct usability testing in the field and revise the system.

Given the emphasis on usability testing in UX projects, consider the following two conceptualizations of *usability*. According to the International Standards Organization (ISO 9241-11, 2018), the usability of a product is the extent to which it is effective, efficient, and satisfying to the target users. According to UXD leader, Jakob Nielsen (1993), usability includes the following five attributes: *Learnability, Memorability, Efficiency, Errors*, and *Satisfaction*. For Nielsen, then, a usable app is easy to learn and easy to use efficiently to accomplish intended goals. Further, for Nielsen, a useable app is memorable and satisfying, in part by facilitating recovery from errors when possible. During usability testing, UX professionals ask users to complete tasks identified in the analysis step. Even in the prototype stage, without a fully functioning system, UX professionals seek input from users in the target group to test the design, the layout, and navigation of menus and dialogs, for instance. Continually asking users for their input is critical to user centered design, which favors the design of apps that will ultimately meet their needs efficiently and satisfy them.

As prototypes and functioning systems are developed, UX professionals may follow specific guidelines for the visual design of windows, dialogs, sidebars, split views, popovers, menus, and toolbars. In addition, specific guidelines exist for the visual design of pop-up buttons, pull-down buttons, push buttons, radio buttons, switches, checkboxes, labels, text fields, combo boxes, sliders, steppers, date pickers, and progress indicators, for instance. One comprehensive source for such guidelines for particular devices or platforms, which engineers typically call *form factors*, is Apple's *Human Interface Guidelines*, available at https://developer.apple.com/design/human-interface-guidelines/. In addition to guiding the general design process and the design of specific elements in a user interface, UX professionals also follow heuristics ("rules of thumb"), such as Nielsen's heuristics for interface and interaction design (Molich & Nielsen, 1990; Nielsen, 2020), which offer guidance, such as display system status; provide a cancel option to exit a task no longer sought by the user; and consistently follow industry conventions. In real world practice, though usability testing is very much preferred, it might not always be possible to consult an actual user. When target users are not available, a designer might seek review by an expert who would conduct what Nielsen and Molich (1990) called *heuristic evaluation*. Obtaining a few objective opinions from UX professionals, and others with knowledge of visual design and app development, might help refine an interface. To repeat, though, usability testing with users in the target group is much preferred. As was the case with visual design principles, experienced UX designers might have justification to deviate from a guideline.

Even though many user experiences with computers involve visual interfaces, some interactions occur through audible conversations. Accordingly, UX professionals also design apps with voice interfaces (Deibel & Evanhoe, 2021; Hall, 2018). Lastly, since apps are designed for clients, UX professionals are keen to develop communication skills. Indeed, articulating a design is critical to a successful career in UXD (Greever, 2020).

3.2.3 Prototyping in Adobe XD

To facilitate discussions about a concept or idea, one might create a graphic. Indeed, for many people, "a picture is worth a thousand words." Tom and David Kelley extend that claim to convey the value of a prototype: "If a picture is worth a thousand words, a prototype is worth a thousand meetings." Indeed, a prototype eliminates much speculation and fanciful discussion about possibilities. We consider *Adobe XD* here because it is a free tool for creating prototypes of apps, whether native to a mobile device or a web app.

We will use *Adobe XD* to create a functional prototype of an app on artists in the Dutch Golden Age. From about 1580 to 1670, trade, art, and science flourished in the Netherlands. With perhaps five million Dutch paintings produced during the Dutch Golden Age, one could spend a lifetime seeking, recording, and documenting details about the paintings and the artists in the Dutch Golden Age. In this prototype, we will include the following artists: Rembrandt van Rijn, Frans Hals, Johannes Vermeer, Judith Jans Leyster, Bartholomeus van der Heist, Jan Steen, Rachel Ruysch, Salomon van Ruisdael, Jacob van Ruisdael, Maria van Oosterwijck, Aelbert Cuyp, and Gerard ter Borch. In an actual app, images of their work and, when available, an image of the artist would be expected. For our purposes, to learn how to articulate the visual design of the app and to learn how to simulate user interaction in the app (thus capturing its look and feel), we will use a combination of text and placeholder images.

The *Adobe XD* Starter version can be downloaded from the following link: https://www.adobe.com/products/xd/pricing/starter-plan.html?mv=affiliate&mv2=red Alternatively, one could use the trial version to become acquainted with the features of *Adobe XD*. For the trial version, go to https://www.adobe.com/products/xd.html and scroll down to the Free Trial button. Once installed, a new prototype can be created using the default dialog or by dropping down the File menu and selecting New. *Adobe XD* presents options for selecting a blank canvas, called an *artboard* in *Adobe XD*, suited to particular devices. You may also create a custom artboard, which is the option we will select here. Insert 400 for the width and 700 for the height. Then click the Custom Size button. The blank artboard appears in the center of the interface, with a toolbar on the far left, then a narrow window for document assets, and a properties window on the right. Hover the mouse over each icon in the toolbar to associate the icon with its tool (i.e., Selection, Rectangle, Ellipse/Circle, Polygon, Line, Pen, Text, Artboard, and Zoom).

To create the visual design depicted in Figure 3.1, complete the following steps.

1. Select the **Rectangle** tool; click on or near the upper-left corner of the artboard and drag a rectangle across the entire artboard such that the height is close to 135 pixels. You may watch the height field in the properties windows to get the exact size. Optionally, you can set the field to 135 if dragging the rectangle to the exact size seems tedious.
2. Select the **Text** tool; click near the upper-left corner of the artboard; type `Artists of the`, then press Return/Enter, then type `Dutch Golden Age`. In the properties window, click

the icon to center text; change the font to Zapfino (or Palatino, or whatever you wish); and change the font size to 20

3. Click the Arrow button in the toolbar to activate the **Selection** tool; adjust the textbox containing the title as necessary to center the title. Fortunately, the software shows a center guide and the number of pixels to the edges from the bounding box of the text. The software also snaps the top edge of the title's bounding box to the top of the artboard when the bounding box is close to the top of the artboard.
4. Rename the artboard by double-clicking its default name, which appears above the upper-left corner of the artboard, and replacing the default name with `Artist List`
5. Save the file, whether in *Adobe Cloud* (File; Save As…) or on your computer (File; Save As Local Document…)
6. **Drag** and **Drop** the silhouette thumbnail image (*personSilhouetteThumbnail.png* in this book's electronic resources) toward the left edge of the artboard under the title. If unsatisfied with your placement, select the image and set x to 0 and y to 135 in the Properties window
7. Select the **Text** tool; click to the right of the silhouette image; click the left-justification icon in the Properties window; and type Rembrandt van Rijn
8. With the **Text** tool still selected, click toward the right edge of the artboard about 25 pixels below the bottom of the title; and type the greater than sign (>). Change the font to Menlo and the font size to 16. If unsatisfied with your placement, drag the text box containing the greater than sign to 365 on the x-axis and 158 on the y-axis. The software will display a horizontal guideline when the text box is in the middle of the Rembrandt text box. If still unsatisfied with your placement, set x to 365 and y to 158 in the Properties window
9. Select the **Line** tool. Click and drag a horizontal line across the artboard underneath the entry for Rembrandt. Set y (the vertical position) to 205 if unsatisfied with the placement of the line.
10. Select the person silhouette image, both text boxes, and the line. Copy and paste to create a duplicate copy of the entry and drag below the original Rembrandt entry. Repeat the paste and drag operations six more times to fill the artboard.
11. Replace the names in rows 2–8 with those in Figure 3.1. Double-clicking a text box will highlight the text contained in it; then type the artist's name for that row
12. For each row, select the placeholder image; hold the Shift key down and select the textbox containing the name of the artist and the textbox containing the greater than sign; drop down the Object menu and select Group (or use the shortcut key, Ctrl-G or Command-G)
13. Save the file

Now we will create one artboard containing details about Rembrandt. The design of the artboard will be consistent with Figure 3.2, which includes a link for navigation to return to `Artist List`; Rembrandt's name (optionally, though excluded from Figure 3.2, you could include his birth year, 1606, and year of his death, 1669); a placeholder image for Rembrandt; and two placeholder images with captions for two sample paintings by Rembrandt. Then we will copy the artboard seven times and edit the copies in order to create the necessary eight artboards, one for each artist.

1. Select the entire artboard by clicking its name, which in Step 4 above, you changed to `Artist List`
2. **Copy** and **Paste** the artboard (e.g., use keyboard shortcuts, Ctrl-C; Ctrl-V on Windows or Command-C; Command-V on MacOS; or drop down the Edit menu, select Copy; then from the Edit menu, select Paste)
3. Rename the duplicate artboard by double-clicking its default name and replacing that text with `Rembrandt`

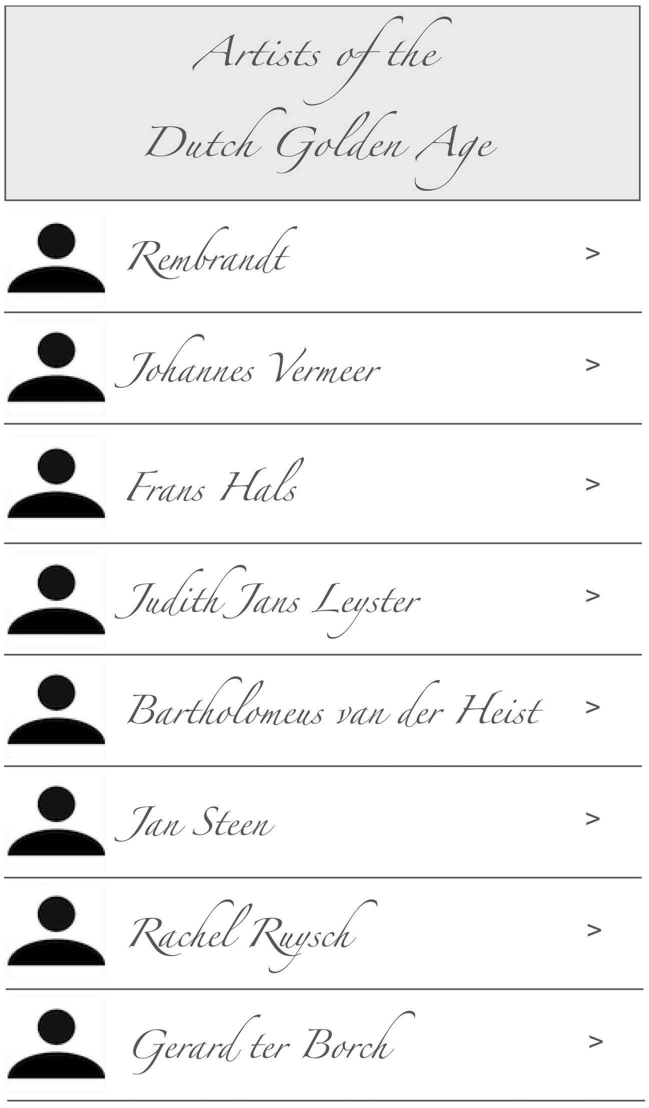


Figure 3.1 List of artists in prototype

4. Click the **Selection** arrow in the toolbar and click below the bottom-right edge of the `Rembrandt` artboard and drag across the left edge to the top row in the list. All objects in the rows from Gerard ter Borch to Rembrandt should be selected. Cut the selected objects, whether by pressing the shortcut key (Ctrl-X in Microsoft Windows; Command-X in Apple MacOS) or select Cut from the Edit menu
5. Replace the text with `Rembrandt`. Position the textbox in the middle of the artboard horizontally. For vertical alignment, set y = 60 in the Properties window
6. Select the **Text** tool in the toolbar and click inside the upper-left corner of the artboard. Type `< Artists` and change the font to Times New Roman; also, left-justify the text; set the

font size to 20; make the text appear bold by replacing Regular with Bold in the drop-down menu beside the font size; position the textbox at x = 15 and y = 15. (This text will become a clickable link, which will return the user to the list of artists.)

7. **Drag** and **Drop** the silhouette image (from this book's electronic resources) underneath the light gray rectangle (which is serving as the background for the navigation bar and the artist's name)
8. To present an artist biography and images of the artist's paintings, a vertical scrolling feature will be implemented. Currently, in *Adobe XD*, this is done by grouping objects and making the group scrollable. For our purposes, we will group a text field and two placeholder images. For alignment purposes, it will be helpful to view the biography text field and two placeholder images on the artboard before grouping them. Hence, we will temporarily set the height of the Rembrandt artboard to 1400 and return it to the device height of 700 after grouping the scrollable objects. Select the entire artboard by clicking its name, Rembrandt. In the Properties window, set the Height (H) to 1400 (or you could click and drag the center handle at the bottom of the artboard)
9. Select the **Text** tool in the toolbar and click below the silhouette image. Type `Brief Biography` and then press the Enter/Return key twice. Type `Artist bio begins here`
10. Set the font size of the heading, Brief Biography, to 20 and make it appear bold. The text of the biography appears as Regular text with font size 18. Move the left edge of the Biography textbox to Column 10 (e.g., drag the textbox to Column 10; or set x to 10 in the Properties window)
11. **Drag** and **Drop** the placeholder image for the first sample painting (from this book's electronic resources) underneath the textbox of the biography
12. Select the textbox of the biography. Copy and paste it. While holding the Shift key (to keep the column position constant), move the duplicate textbox underneath the placeholder image of the first sample painting. Double-click the duplicate textbox; replace all of the text with `Title of first painting, Year`
13. **Drag** and **Drop** the placeholder image for the second sample painting (from this book's electronic resources) underneath the textbox created in the previous step
14. Select the title/year textbox. **Copy** and **Paste** it. While holding the Shift key, move the duplicate textbox underneath the placeholder image of the second sample painting. Double-click the duplicate textbox; replace `first` with `second`
15. Save the file
16. Select the biography textbox, the two title/year textboxes and the placeholder images for the two sample paintings. Group the objects, whether by dropping down the Object menu and selecting Group or by pressing the shortcut key (Ctrl-G in Windows; Command-G in MacOS). Click the vertical scroll icon in the Properties window (which is positioned underneath the Y coordinate; hover over the icons to identify the vertical scroll icon)
17. Select the entire artboard by clicking its title, `Rembrandt` and set the Height (H) back to 700
18. Click the scrollable group to select it and drag the bottom scroll handle to the bottom of the artboard
19. Save the file
20. Test the scrolling by clicking the **Preview** button in the upper-right corner of the XD window (the button to the left of the zoom percentage; the button features a triangle, like a Play button). Close the **Preview** window to return to your prototype
21. Select the entire artboard by clicking its name, `Rembrandt`
22. **Copy** and **Paste** the artboard seven times; double-click the heading of each new artboard and replace it with the last name (or initials) of the next artist in the list; also replace the name Rembrandt with the name of the next artist in the list (i.e., Johannes Vermeer, Frans Hals, Judith Jans Leyster, Bartholomeus van der Heist, Jan Steen, Rachel Ruysch, Gerard ter Broch)

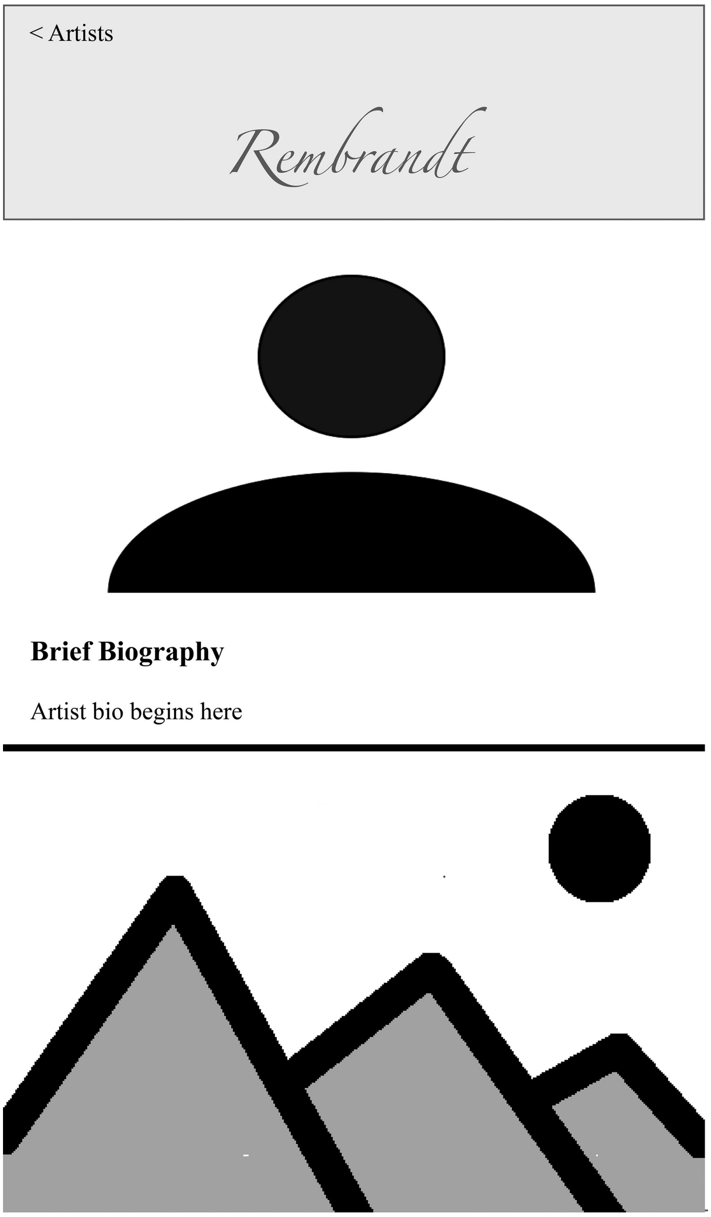


Figure 3.2 Prototype for artist details

Lastly, we will switch to *Adobe XD*'s **Prototype** mode and, by dragging from the greater than sign for each artist to the artboard for each artist, we will enable navigation. We will also use a series of eight drag operations to create links back to the `Artist List` artboard.

1. To show all the artboards, set the Zoom level to 23%. Select the original artboard by clicking its name, `Artist List`. Then click the greater than sign at the end of the row for Rembrandt

2. Click the **Prototype** tab, which is near the upper-left corner of the *Adobe XD* interface
3. The grouped objects are shown within a rectangle that has a blue circle on its right edge. Click on that circle and drag it anywhere over the Rembrandt artboard; then drop. A link is now depicted from the Rembrandt row in `Artist List` to the `Rembrandt` artboard. Now select the `< Artists` textbox in the `Rembrandt` artboard. Then click within the blue circle; drag back to the `Artist List` artboard and drop
4. Save the file
5. Test the navigation by clicking the **Preview** button
6. Repeat the linking actions in Step 3 for each of the seven other artists to create seven more navigation links from the row for each artist to each artist's artboard; also, create seven more navigation links back to `Artist List` from the artboards of the seven other artists
7. Save the file
8. Test the navigation by clicking the **Preview** button

Optionally, you may add more artists and convert the artist list to a scrollable group, which would make the app prototype appear more authentic. For even more authenticity, consider writing actual biographies and including them in the prototype, along with images and appropriate attributions.

3.3 Making

3.3.1 Creating Bitmapped Graphics in GIMP

Bitmapped images, which are also known as raster images, store the color of each picture element. In computer parlance, a picture element is called a pixel. Whereas closely packed dots of ink yield the text and images in newsprint, closely positioned squares illuminated in various colors yield the text and images displayed on computer screens. Images containing 640 pixels along the horizonal in each of 480 rows store color information for 307,200 (640 * 480) pixels.

In this section, we first use the GNU Image Manipulation Program (*GIMP*) to verify that digital images are displayed or rendered by tightly packed squares of various colors. *GIMP* is a free and open-source program for creating and editing images. You may download *GIMP* from gimp.org (gimp.org/downloads) by clicking one button. I found the download time to be less than one minute to acquire the approximately 170 MB file; installation time was longer.

1. To create a new image, drop down the **File** menu; select **New . . .** and click **OK**
2. Select the **Pencil** tool by clicking its button in the Tool Palette, which is in the top part of the left sidebar
3. Move the cursor over the canvas; then click, drag, and drop to create a few lines, curves, circles, or any figure you wish
4. Drop down the **View** menu; select Zoom; 400% (or press the short cut key 3). Look closely at the image to see each square in the image and notice how the variation in color of each tightly packed square creates a blur effect. Return to the default view of the image by clicking View; Zoom; Revert Zoom (or press the short cut key `).

Regarding the Tool Palette, over time, you may select each tool or hover the cursor over each icon to see the tool name and function. You will find that *GIMP* has the usual image editing tools, such as a pencil, brush, eraser, text, and crop tool, as well as selection tools, including the Fuzzy Select tool, which some may call a "magic wand" tool. The Fuzzy Select tool captures consecutive (or contiguous) pixels of a particular color. Such a tool is useful for eliminating the background color of an image. Eliminating the background of an image is critical in animations. Nothing conveys amateur work in animation faster than an image moving across a screen with its visible background. Four steps below can be used to avoid that rookie error. First, though, let's be sure we can change the foreground and background colors in *GIMP*.

On the left sidebar, immediately below the Tool Palette, are a pair of overlapping rectangles. Click the top-left rectangle (black by default) to change the foreground color. This will pop up a dialog (window) for changing the foreground color. Select any color you wish by clicking on the gradient rectangle of one color or you may click on a particular color in the rainbow color strip to its right and then on the gradient rectangle. The color change dialog offers numerous other methods for changing the color. For example, you may drag sliders to set the mix of red, green, blue, lightness, chroma, and hue. Alternatively, you may enter a hexadecimal color value (e.g., ffdd99, 3D5C8A). You may also click one of the icons in the upper-left corner of the color change dialog to select a color from sliders for cyan, magenta, and yellow; or to select a color from alternative color rectangles and wheels. After changing the foreground color, however you choose to do so, click OK. Then brush a stroke on the canvas, and zoom in to 400% or 800%, to notice the tightly packed squares in the various shades of the color you selected. To change the background color, in the pair of overlapping rectangles in the left sidebar, click the bottom-right rectangle; then select a color using any method you wish; and click OK. You may save the image or discard it before proceeding with the following technique for deleting a background color.

Locate an image with a single consistent background color and open that image in *GIMP* (**File**; **Open…**; navigate your computer's folder system; select the file; and click the **Open** button). If you prefer not to browse your images or wish to acquire one, you may download an image in the public domain. For example, you may download and then open in *GIMP* the image of a goose and duck on a white background, which is available at http://www.publicdomainfiles.com/show_file.php?id=13969387814852, and then proceed through the following steps.

1. Drop down the **Layer** menu; select **Transparency**; and then select **Add Alpha Channel**. (This adds a transparent layer to the image.)
2. Click the **Fuzzy Select** tool from the left sidebar; then click anywhere on the white background
3. Drop down the **Edit** menu and select **Cut**, or use the keyboard shortcut (Ctrl-X on Windows; Command-X on a Mac)
4. If the image has the PNG file extension, you may save it in the same format. Otherwise, export the image to the PNG format by dropping down the **File** menu; and selecting **Export As**…; click *Select File Type (By Extension)*; scroll and select PNG image; specify a file name; and click the **Export** button. A dialog pops up: Accept the default entries by clicking its **Export** button

Many backgrounds are not a single consistent color. In those cases, one needs to be more persistent to remove the multiple regions which comprise the background. You may hold the Shift key down while clicking the various background colors and then cut the background; or you may repeatedly select and cut background colors. Since pixels in bitmapped images are comprised of squares, removing background colors removes squares. When the squares positioned next to curved lines are removed, the curves appear jagged. Sometimes the resolution of the image is low, and the jagged curves are not visible; other times, one might blur the curve. In brief, removing the background color or colors of an image does not always yield a perfect result.

Additional approaches to retaining foreground objects, or removing backgrounds, are available through the *GIMP* documentation, which is available in multiple languages at https://www.gimp.org/docs/. See, for instance, the documentation for extracting a foreground object from its background, which is listed among the common tasks (https://docs.gimp.org/2.10/en/). In that case, the Free Select tool (lasso icon to the left of the Fuzzy Select tool) is used to select the foreground, and then the foreground selection is inverted to highlight and delete the background.

Before leaving this section, note that enlarging or reducing the size of a bitmapped image, which technicians call *scaling*, decreases the clarity of the image. In *GIMP*, one can view the effects of scaling by zooming in and out by pressing the + and – keys respectively. Alternatively, an image can be scaled by dropping down the Image menu; selecting Scale Image… and adjusting the width and height of the image in the dialog. Lastly, click the Scale button. Changing only the width or height will change the *aspect ratio*, resulting in a distorted image. The *aspect ratio* of an image or digital display device is width divided by height, which may be presented as width:height, as in 4:3, 3:2, 16:9, for instance. In *GIMP*, clicking the icon to the right of the width and height fields will toggle between states for retaining the aspect ratio of the image or permitting it to change. When the state is set to retain the aspect ratio, a change to the width will automatically change the height proportionally and vice versa.

3.3.2 Creating Scalable Vector Graphics (SVG) with a Text Editor

In contrast to raster graphics, a Scalable Vector Graphic (SVG) retains the clarity of the image when the size of the image is changed (i.e., *scaled*). Rather than store color information about individual pixels, an SVG image stores information about the shapes to be created, which are rendered on a Cartesian plane. The shapes may be lines, arcs, curves, circles, ellipses, or polygons. Shapes may also be specified by paths. The shapes can be displayed in a wide variety of colors. Comprehensive documentation for SVG markup is available at the following website: https://www.w3.org/TR/SVG/

Rendering a Circle

In accordance with SVG syntax, we first specify the width and height of the image. Then we will include specifications for the shape or shapes to be created. In the case of a circle, we specify the center and radius.

Open a text editor (e.g., *NotePad* on a PC running Microsoft Windows or *TextEdit* on a Mac)

Enter the following text and save the file as *circle.html*

```
    <svg width="400" height="400">
  <circle cx="200" cy="200" r="100" />
</svg>
```

By saving the file with the `.html` extension, web browsers will render SVG specifications.

Double-click the *circle.html* file icon to view the result. Perhaps you anticipated that the background would be white and the circle black.

In your text editor, edit the SVG as shown below to change the default color of the background and to specify a fill color.

```
    <svg width="400" height="400" style="background-color:lightgray">
  <circle cx="200" cy="200" r="100" fill="green" />
</svg>
```

Once again, save the *circle.html* file. Then redisplay it in your web browser by pressing Ctrl-R on a PC running Microsoft Windows or Command-R on a Mac.

The attributes of the circle are apparent: r is the attribute for radius; cx and cy are the x and y coordinates of the circle's center. In an SVG grid, the origin (0,0) is in the upper-left corner. Verify this by, for instance, changing r to 10, cx to 20, and cy to 300. Remember to save the text file and then refresh the web browser (Ctrl-R or Command-R).

After confirming that the origin of the grid is in the upper-left corner, return cx and cy to 200 and r to 100. Refresh the web page to restore the larger circle. Next, we will zoom in and out multiple times to check for any distortion. On a Windows PC, press Ctrl+ to Zoom In and Ctrl- to Zoom Out; on a Mac, press Command+ to Zoom In and Command- to Zoom Out. (Some web browsers may use different keyboard shortcuts to Zoom in and out; check the View menu if those keyboard shortcuts did not work in your web browser.) Notice the superb rendering of the circle's curvature when zooming both in and out. For contrast, you may take a screen shot of the circle (Command-Shift-4 on a Mac and drag over the circle; use Fn-Key+Windows-Logo-Key+Space-Bar on Windows); then open the circle in *GIMP* and zoom in and out or scale the image. Bitmapped images become blurry when scaled, unlike vector images.

Lastly, for the circle, if you would like a border around it, specify the stroke color and stroke width, as shown below.

```
    <svg width="400" height="400" style="background-color:lightgray">
  <circle cx="200" cy="200" r="100" stroke="red" stroke-width="4"
fill="green" />
</svg>
```

Vary the stroke color and width as you please, as well as the color of the background and the sizes of the circle and background.

Rendering Lines, Rectangles, and Ellipses

Open a new file in your text editor and name it *primitiveShapes.html*. Insert the following to produce a rectangle, a line, and two ellipses. As before, save the file after entering the text and open it in a web browser.

```
    <svg width="500" height="400" style="background-color:lightgray">
  <rect x="15" y="15" width="200" height="180" fill="green" stroke="red"
stroke-width="4" />
  <line x1="40" y1="215" x2="115" y2="380" stroke="gold" stroke-width="3" />
  <ellipse cx="350" cy="50" rx="100" ry="20" fill="rgb(250,204,0)"/>
  <ellipse cx="350" cy="200" rx="20" ry="100" fill="#FF99E8" />
</svg>
```

Vary the position of the rectangle by changing the values of x and y, which define the upper-left corner of the rectangle. Also vary the values of the rectangle's width and height. With respect to the line, x1 and y1 specify the coordinates of one endpoint; x2 and y2 specify the coordinates of the other endpoint. All SVG coordinates are relative to the origin, which has coordinates (0,0) and is located in the upper-left corner. Lastly, the ellipses are defined by a center, a horizontal radius (rx), and a vertical radius (ry). The fill colors for the ellipses use two different notations. Rather than state the word for a common color, a mix of red, green, and blue is specified. In the first ellipse, this is achieved by specifying values in the 0–255 range for each of red, green, and blue respectively. In the second case, two hexadecimal digits are used to specify the amount of red (FF), green (99), and blue (E8) to mix. If unfamiliar with hexadecimal numbers, valid digits include the decimal digits 0–9 and A through F. Vary the digits in both cases to produce colors you wish to view. Numerous websites provide red, green, and blue (rgb) values for colors in both decimal and hexadecimal, such as https://www.w3schools.com/colors/colors_picker.asp and trycolors.com. SVG color words are available at https://commons.wikimedia.org/wiki/File:SVG_Recognized_color_keyword_names.svg.

Those primitive shapes are sufficient for most images. Even though SVG also includes the polyline command for stringing many lines together (e.g., `<polyline points="75,215 100,220 120,235 140,325 170,350" fill="none" stroke="black" />`) and the polygon command for stringing segments with many angles together, the path command must be used to render the curves and edges in many images.

Rendering Paths

An SVG path contains a series of commands and points enclosed within quotation marks. The points are specified by their x and y coordinates. Many path commands permit points to be absolute or relative. Typically, the first command in a path specifies its start position. The remaining command or commands specify the manner in which the path is traversed. Let's start with a path that forms a staircase.

Create a new file in your text editor; enter the text below; and save the file as *staircase1.html*.

```
    <svg height="550" width="600" stroke="blue" stroke-width="3"
fill="none">
   <path d="M 50 500 L 100 500 L 100 450 L 150 450 L 150 400 L 200 400 L
200 350 L 250 350 L 250 300 L 300 300 L 300 250" />
</svg>
```

Double-click the file icon to view the staircase in a web browser. Notice that the path begins with a Move (M) command (i.e., M 50 500). Consequently, the path begins in Column 50 Row 500. Then the path contains a series of `Lineto` (L) commands, each of which draws a line from the current position to the coordinates specified. The uppercase L renders lines to absolute coordinates. Accordingly, from the start position (Column 50 Row 500), a line is drawn to Column 100 Row 500; then a line is drawn from that position to Column 100 Row 450. This pattern of moving 50 pixels right, followed by 50 pixels up continues until the final riser (line) in the staircase is drawn, which ends the path at Column 300 Row 250.

SVG syntax is such that a command letter need only be entered when a new command is specified. Hence, the L command need not be specified repeatedly, as shown above. Convince yourself of this by replacing the path above with the following one. As always, after making any change to the text, save the file and then reload the page in the web browser.

```
<path d="M 50 500 L 100 500 100 450 150 450 150 400 200 400 200 350 250
350 250 300 300 300 300 250" />
```

A variation on this staircase theme involves use of relative distances, rather than absolute coordinates. Notice that when using the absolute coordinates above, each horizontal line increases the column by 50 while retaining the value of the row. In the SVG coordinate system, a vertical line retains the value of the column and decreases the row by 50. Hence, to render a horizontal line in the staircase using distances relative to the current position, the column value must increase by 50 while adding zero to the value of the row. In this example, to render the horizontal lines relative to the current position, we use the lowercase l followed by the change in distances for the column and row (i.e., l 50 0). To render the vertical lines, zero is added to the column value while 50 is subtracted from the row value (i.e., l 0–50). Open a new file in your text editor; enter the text below; and save the file as *staircase2.html*.

```
    <svg height="550" width="600" stroke="red" stroke-width="3"
fill="none">
  <path d="M 50 500 l 50 0 l 0–50 l 50 0 l 0–50 l 50 0 l 0–50 l 50 0 l
0–50 l 50 0 l 0–50" />
</svg>
```

In this case, as before, there is no need to insert the command letter, lowercase l, repeatedly. Verify this by replacing the path above with the following one.

```
<path d="M 50 500 l 50 0 0–50 50 0 0–50 50 0 0–50 50 0 0–50 50 0 0–50" />
```

There is one more noteworthy variation to lines in SVG paths. When there is no change in the vertical dimension, the horizontal line command (H, h) may be used. Conversely, where there is no variation in the horizontal dimension, the vertical line command (V, v) may be used. To try this variation, open a new file in your text editor; enter the text below; and save the file as *staircase3.html*.

```
    <svg height="550" width="600" stroke="red" stroke-width="3"
fill="none">
  <path d="M 50 500 H 100 V 450 H 150 V 400 H 200 V 350 H 250 V 300 H 300
V 250" />
</svg>
```

After testing that version, which uses absolute coordinates, replace that path with the following one to render the staircase using distances relative to the current position.

```
<path d="M 50 500 h 50 v –50 h 50 v –50 h 50 v –50 h 50 v –50 h 50 v –50" />
```

After verifying that the path above renders the staircase in the same position as all previous versions, change the start position. For example, you might change the column from 50 to 0, 10, or 100. You might also change the row from 500 to 550 or 450, for instance. Notice that the staircase is rendered properly in its new position. However, if you went back and changed the start position in the previous version, which used absolute coordinates, you would find that the developer must also change the absolute coordinates of all the points in order to render the staircase properly.

Rendering Curves, Points, Text, and Two More Lines

This section provides specifications for a curve, points, and text, which produce the diagram in Figure 3.3.

Open a new file and enter the following SVG specifications to render a parabola.

```
    <svg height="550" width="600" stroke="black" stroke-width="1"
fill="black">
  <path d="M 50 50 Q 200 500 350 50" fill="none" />
</svg>
```

Save the file as *mathDiagram.html*. Then open it in your web browser. The Q command creates a quadratic Bezier curve. In the example above, the curve extends from the start point (50,50) to the end point (200,500), which are specified by the first two parameters of the command. The third and fourth parameters specify the control point (350,50 in this case), which controls the shape of the curve.

Next, one at a time, insert the 12 additional SVG specifications below. After adding each line below the path specification, save the file, and then reload the page in your web browser to determine the effect of each line.

```
    <svg height="550" width="600" stroke="black" stroke-width="1"
fill="black">
  <path d="M 50 50 Q 200 500 350 50" fill="none" />

  <circle cx="90" cy="153" r="3" />
  <text x="100" y="158" fill="black">A</text>

  <circle cx="126" cy="220" r="3" />
  <text x="136" y="226">B</text>
  <line x1="53" y1="84" x2="150" y2="265" />

  <circle cx="200" cy="275" r="3" />
  <text x="195" y="265">C</text>

  <circle cx="330" cy="105" r="3" />
  <text x="310" y="110">D</text>

  <circle cx="299" cy="178" r="3" />
  <text x="280" y="183">E</text>
  <line x1="351" y1="55" x2="266" y2="255" />
</svg>
```

Optionally, the `path` specification above, which contains absolute coordinates, could be replaced with the `path` below, which uses distances relative to the starting position.

```
<path d="M 50 50 q 150 450 300 0" fill="none" />
```

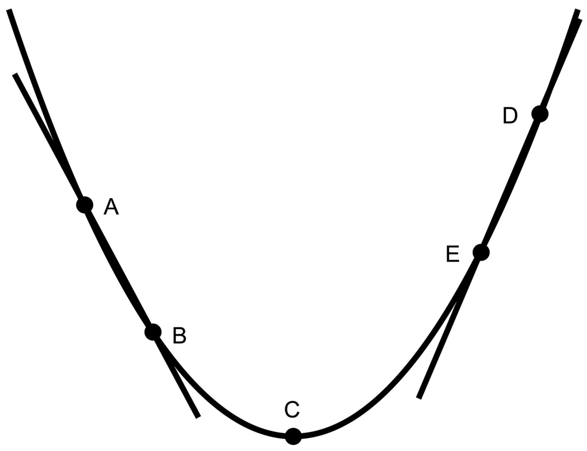


Figure 3.3 A conic section with intersecting lines

Exploring SVG Images

Consider downloading and viewing some Scalable Vector Graphics (SVG) to increase awareness of the images they produce through use of the path command. Once downloaded, open the image in your text editor. You may also copy the file and change the file extension to .html in order to view the image in a web browser. Alternatively, you may open SVG files in a vector graphics program, such as *Inkscap*e or *Adobe Illustrator*. Here are a few links to SVG images you may wish to consider.

https://freesvg.org/simple-dolphin
https://commons.wikimedia.org/wiki/File:Ferc-fish_ladder.svg#/media/File:Ferc-fish_ladder.svg
https://commons.wikimedia.org/wiki/File:Mona_Lisa_vectorized.svg
https://commons.wikimedia.org/wiki/File:English_pattern_playing_cards_deck.svg

To locate images of particular interest to you, use a web search engine such as images.google.com or commons.wikimedia.org; include SVG in addition to the words describing the image you wish to locate (e.g., svg mona lisa; svg dolphin; svg bird; svg musical instruments; svg world map). You may also conduct a key word search for free SVG images at freesvg.com.

As an alternative to creating your own SVG images by entering commands in a text editor, you may create your own images in *Inkscape*, which we pursue in the next section.

3.3.3 Creating Scalable Vector Graphics in **Inkscape**

Inkscape is a free program for creating Scalable Vector Graphics (SVG). As discussed in the previous section, SVG images store information about shapes to be rendered. Many developers enjoy using image editing programs that provide a visual interface. *Inkscape* is one such program. To explore *Inkscape*, we will create the staircase image and math diagram produced in the previous section using SVG commands. As we use the visual interface to create, place, and adjust objects on the canvas, *Inkscape* will automatically create the SVG commands needed to render the image. After utilizing the tools in the visual interface, we need only save the file.

Download *Inkscape* from inkscape.org by clicking the Download tab and selecting Current Version. Then click the button that corresponds to your computer's operating system, whether Microsoft Windows, Apple MacOS, or Linux. In my case, the download completed in about one minute; the installation process was slightly longer than one minute.

When *Inkscape* opens, a frame appears around an empty canvas. Buttons to select image editing tools appear as icons to the left of the canvas. Those vertically aligned icons comprise the *toolbox*. Buttons to adjust object properties appear as icons above the canvas, in what is often called a *ribbon*. By default, *Inkscape* has two ribbons above the canvas. The ribbon nearest the canvas includes fields to view and set the position of the selected object, as well as its width and height. A row of color swatches appears below the canvas and property dialogs appear to the right of the canvas. You may zoom in and out of the canvas using the + and - keys. An empty canvas appears by default. At any time, you can create another empty canvas by dropping down the File menu and selecting New.

Creating Lines for a Staircase

Like other image editing programs, the *Inkscape* Toolbox contains tools for creating primitive shapes. To identify the purpose of each tool, hover the cursor over each of the icons/buttons in the toolbox. Like other software, the tool at the top of the toolbox appears as an Arrow and is used for object selection. After identifying the tools, select the one for creating **Bezier curves**

and straight lines. Click inside the bottom-left corner of the canvas. Then move the cursor horizontally to draw a straight line (there's no need to drag); click again when the line is the desired length (about 80 pixels; perhaps one centimeter or half an inch, depending on the zoom level); then move the cursor vertically; click again when the line is the desired length (about the same length as the previous line). Continue to insert horizontal and vertical lines to complete a staircase, though the line lengths are likely unequal. Double-click to stop inserting lines. Save the file. (Optionally, you may open the file in a text editor to view the path command *Inkscape* created to render the staircase, along with numerous other entries for metadata and name spaces, for instance, all of which can be ignored.)

Let's try that again to learn more about *Inkscape*. This time we will create the staircase using primitive shapes. From the toolbox, select the rectangle/square tool and, optionally, click a color swatch below the canvas. Inside the bottom-left corner of the canvas, click and drag to create a rectangle positioned horizontally; minimize the height of the rectangle so that it appears to be a line. With the rectangle selected, use the Width (W) and Height (H) fields in the ribbon above the canvas to set the width to 33.0 pixels exactly and the height to 3.0 pixels exactly. Additionally, set the X and Y coordinates to 17.0 and 280.0. With the rectangle still selected, click the **Stroke paint** tab in the dialog positioned to the right of the canvas. (If the dialog is not present, drop down the **Object** menu and select **Fill and Stroke….**) Turn off the stroke by clicking the X button that appears to the left of the set of buttons used to set the Flat color, Linear gradient, Radial gradient, and Mesh gradient.

Now create another rectangle and orient it vertically above the right end of the first rectangle. Again, turn the **Stroke paint** off. Use the W and H fields to set the width to 3.0 pixels and the height to 30.0 pixels. Setting the X and Y coordinates to 47 and 250 respectively will position the rectangle immediately above the horizontal rectangle. We now have two lines (implemented as rectangles) of the staircase. Select both lines by clicking the Arrow button in the Toolbox (the Select tool); then click either rectangle; hold the Shift key down and click the second rectangle. Next, drop down the **Object** menu and select **Group** (or press the keyboard shortcut). **Copy** and **Paste** the grouped object. Reposition the object to the top-right of the existing structure to form the next component of the staircase. Repeat the paste and positioning process three times to complete the staircase. Save the file.

Consider recreating the staircase once again using the tool for making Bezier curves and straight lines. This time, though, create only the first horizontal line. Adjust the position and dimension parameters if you wish. Then **Copy** and **Paste** the horizontal line; drop down the **Object** menu and select *Rotate 90° CW* (or *Rotate 90° CCW*). Drag the now vertical line to the right edge of the horizontal line or set the X and Y parameters to position the vertical line. Once those two lines are positioned properly, group them. Then **Copy** and **Paste** the grouped object and reposition it; repeat until you are satisfied with the staircase.

Creating Curves, Points, Text, and Two More Lines

Proceed through the following two sets of steps to create a parabola with five points and two intersecting lines, as in Figure 3.3. First, the parabola.

1. Create a new image (From the **File** menu, select **New**)
2. Select the **Circle, Ellipse, Arc** Tool
3. Click within the canvas and drag to create an ellipse that has more height than width
4. Drag the round center handle to the opposite side
5. Turn off the fill color; Select the black stroke color; Set the stroke width to 1; Select the arc icon from the Tools Control Bar (which is directly above the canvas when the *Circle, Ellipse, Arc* object created is still selected)

6. Drop down the **Edit** menu and choose Deselect
7. Save the file

Now for the points, labels, and lines.

1. Ensure that the **Circle, Ellipse, Arc** Tool is selected; then drag the mouse over a portion of the parabola to make a circle
2. While the arc shape is still selected, click the Whole ellipse icon from the Tools Control Bar; then set the **Fill** color to black. If necessary or desired, click the **Selection** tool (Arrow button) and resize and/or move the circle to ensure that it appears as a distinct point on the parabola
3. To create the other four points, you may wish to repeat the point creation process four times (to practice creating circles); or you may click the object selection tool and Copy, Paste, and Drag the points to the five positions shown in Figure 3.3
4. Select the **Text** tool; click beside each point and type the appropriate letter to label the point
5. Using the tool to create **Bezier curves and straight lines**, create a line through Points A and B, and a line through Points D and E
6. Save the file

Options for Creating Functions and other Figures

In *Inkscape*, it is possible to create a parabola using the Function Plotter rather than the Arc tool. In a new document, first create a rectangle because *Inkscape* plots functions inside rectangles. With the rectangle selected, drop down the Extensions menu; and select Render; Function Plotter. For a parabola, you may set the Start X value to -2; the End X value to 2; the Y value of rectangle's bottom to -2; the Y value of rectangle's top to 2; and the number of samples to 200. Make sure *Use polar coordinates* is not checked. In the Function field, enter x * x. If you wish, click the *Remove rectangle* checkbox to hide the rectangle in which the parabola is rendered.

Beyond Function Plotter, you may also wish to explore the development of Spirographs, for instance. A bounding rectangle is not necessary to create a Spirograph. Just drop down the Extensions menu and proceed through Render to Spirograph. Try the default settings initially, and then adjust the parameters as you wish. For instance, you may set Ring Radius (R) to 200; Gear Radius (r) to 30; and Pen Radius (d) to 52 pixels. For Gear placement, select either Inside (Hypotrochoid) or Outside (Epitrochoid). Lastly, set Rotation to 45º and click the Apply button.

3.3.4 Audio Recording and Editing in **Audacity**

Audio recording and editing can be quite challenging; however, you may attain fine production quality by controlling two external factors. First, whenever possible, record in a quiet location. Second, use a decent microphone. Fortunately, many mobile devices include decent microphones, so you may be able to use recordings from your mobile device to produce satisfactory or even high-quality audio productions. Another option is to record audio using a microphone attached to, or built into, your desktop or laptop computer. The quality of microphones built into desktop and laptop computers varies greatly. Similarly, microphones attached to computers may produce horrible audio recordings or fabulous ones. You can test the quality of your microphone by recording in *Audacity*, which is a free and open-source program for Windows, MacOS, Linus, and other platforms. To download *Audacity*, go to https://www.audacityteam.org and select the operating system of your microcomputer, whether Windows, Mac, or Linux. In my case, the total time for downloading and installation was less than 40 seconds.

Audacity opens to a new project window. Additional projects can be started by dropping down the File menu and selecting New. The upper-left corner of the project window displays standard audio play, pause, rewind, and record buttons. In the toolbar below those buttons are sliders for setting microphone sensitivity and speaker volume. The toolbar below that one includes a drop-down menu for selecting among the microphones attached to your computer system, including any built-in microphone. The drop-down menu to the right indicates that stereo (two track) recording is set by default, which is fine if that is what you want. If recording your voice, a single track will do. A timeline appears below the toolbars and waveforms of audio tracks appear below the timeline. To delete an audio track, click the x in the upper-left corner of the dialog to the left of the waveform. A track can be added by dropping down the Track menu and proceeding through Add New to either Mono Track or Stereo Track. Alternatively, recording audio will add a track. Importing audio (File; Import; Audio) also adds an audio track or tracks. Let's begin with a voice recording and then perform key editing operations, namely amplifying, cutting, splicing, inserting.

1. Click the **Record** button and say a few words (e.g., "testing, 1, 2, 3,4, 5; testing, testing complete"); then stop the recording by clicking the **Stop** button
2. Notice that the two audio tracks have the same waveforms, which is redundant. (A single track suffices for a voice recording.)
3. Click the **Play** button to consider the quality of the recording. (Consider whether acquiring or using a different microphone would be beneficial.)
4. Save the project
5. Open a new project (From the **File** menu, select **New**)
6. Delete any default audio tracks in the waveform space
7. In the Recording Channel drop-down menu, switch from 2 (stereo) Recording Channels to 1 (mono) Recording Channel
8. Click the **Record** button and say a few words; at various times, speak loudly, softly, and normally ("testing, 1, 2, 3,4, 5; this is the second recording; testing, testing complete"); then stop the recording
9. Click the **Play** button and listen for words spoken quietly and loudly. Notice where in the waveform, the words become quiet and where in the waveform the words return to a normal volume
10. Click on the waveform at the start of the quiet section; then drag across the entire quiet section; release the mouse button (or trackpad) at the end of the quiet section
11. Drop down the **Effect** menu and select **Amplify**. A dialog opens with recommended amplification, but you are free to enter a different value or use the slider below the Amplification field to select a different value. Any negative value will reduce the volume level, whereas any positive value will amplify the sound. Click **OK** when ready (the default value is fine for your first attempt). Notice the change to the size of the waveform of the selected section. Press the **Play** button to hear the difference. Repeat this step multiple times, using various amplification values
12. Again, by dragging the cursor over the waveform, select a different section of the waveform. This time select a section you want to delete (perhaps a section with no sound, a flatline section in the waveform)
13. From the **Edit** menu, select **Delete** (or press the Delete key, the Backspace key, or a shortcut key for deletion, either Ctrl-X or Command-X)
14. Press the **Play** button
15. Save the project

Perform two more fundamental operations, namely splicing and audio insertion.

1. Click the **Rewind** button to return the play head to the start
2. Click the **Mute** button in the dialog beside the waveform
3. Drop down the **Tracks** menu and proceed through **Add New** to **Mono Track**
4. Click the **Record** button and say a few words ("how are you"); then stop the recording
5. Click the point inside the original waveform at which you want to insert audio
6. Drop down the **Edit** menu and proceed through **Clip Boundaries** to **Split** (or use the shortcut key, Ctrl-I or Command-I)
7. Hover your mouse over the icons to the right of the Record button to identify the **Time-Shift** button (below the Selection Tool is the Zoom Tool and to its immediate right is the Time-Shift tool); click the **Time-Shift** button to activate the tool. Notice the change to the appearance of the cursor
8. Click and drag the right-side of the split section to the right until there is sufficient space to insert the audio waveform in the track below
9. Click and drag the waveform in the lower audio track up to the top track; slide the waveform to connect to the left-side of the split section; drag the original right-side section flush with the inserted section; click the vertical lines at the joins to make them disappear
10. Click the **Play** button
11. Save the project

3.3.5 Video Editing in Blender

Video is produced for a variety of reasons using vastly different equipment and editing software. On the one hand, anyone with a contemporary mobile device can readily record video by opening the camera app; selecting video; and tapping the record button. Often, such video is not edited. In another case, one might record video, including audio, of a teacher playing a piece of music and providing some tips to novices. Such a video production might involve some video editing to add voice overs to include audio not spoken during recording. Editing might also add scrolling credits, for instance. In other video production scenarios to make a movie or a documentary, video production would encompass three phases, preproduction, production, and post-production. Preproduction would include scripting, location scouting, equipment acquisition, auditions, hiring personnel, contracting, and rehearsals, for instance. Once those time-consuming preliminary steps were complete, setup and video recording (shooting) would ensue. Lastly, the video would be edited in post-production. Much editing is done with expensive software, but *Blender* is free of financial cost and offers some fundamental video editing features. To download the *Blender* installation software, simply click the big *Download Blender* button from the download page (blender.org/download). Once you have *Blender* installed, start a new video editing session by dropping down the File menu and proceeding through New to Video Editing. Then complete the following sets of steps to acquire and import video; splice the video; add scrolling credits; and render the video in mp4 format.

Video Acquisition and Importing

You may have several videos on a mobile device or on your computer, which you can import and edit. If not, you may acquire free video in the public domain from websites such as pixabay.com, which also enables searching and acquisition of images and music, for instance. To search that site for video, go to pixabay.com, click the Video tab and enter a search term. Entering *drone*, for instance, yields more than 500 free videos. Many such videos are free for commercial use without attribution. For this work, after clicking the link to each video you wish to download and checking the Pixabay License, click the big green *Free Download* button, which pops up a dialog to select the resolution (1920 x 1080 will work very well here because that matches the default resolution for video in *Blender*, though changing that default is as easy as replacing the two numbers in the X and Y fields for video dimensions, which appear in the default properties dialog). For this work, download two videos, which need only be about 10–30 seconds each, but could be much longer.

The interface of the video editor contains, from left to right, a File Browser dialog, the Preview window, and the Properties dialog. Beneath those windows are the Sequencer and a right sidebar dialog (set to Active Tool by default). Editing occurs in the Sequencer, which displays a timeline of frames and icons for the media (video, sound, image, and text) assets. The Sequencer has a few drop-down menus, namely View, Select, Marker, Add, and Strip.

To import a video, drop down the **Add** menu and select **Movie**. Then navigate to the folder storing the video and double-click it or select it and click the **Add Movie Strip** button. The buttons to control the preview appear below the Sequencer. Notice that the right sidebar dialog has a new tab, called **Strip**. Click the **Strip** tab and scroll down to the Time information. You may need to tap **Time** to view the Time information. Look at the duration of the video, which appears in two forms, such as 00:00:08:01 and 241. The standalone integer is the total number of frames in the video. By default, the video editor plays the first 250 frames of the imported video. At this point, you could click the **Play** button under the Sequencer to view your video in the Preview window. Prior to that, though, you may wish to adjust the default End frame in the Scene dialog, which is beside the Preview window. Notice the Frame Start, End, and Step fields, as well as the Frame Rate field, in the Scene dialog. I would replace the 250 in the End field with the number of frames specified in the Duration field, which in this example is 241. Then click the **Play** button. The video loops back to the beginning until the Pause button is clicked (which only becomes visible after clicking the Play button). To return to the beginning of the video, press Shift-Left Arrow or click the *Jump to Endpoint* button below the Sequencer. Try all six video control buttons below the Sequencer, as well as different values for the End frame (and, if you wish, adjust the Start frame). As an alternative to clicking the **Play** button, video editors often drag the current frame marker right and left, which is called *scrubbing*. When scrubbing, you control the rate at which the video is viewed. In *Blender*, the current frame marker is a blue rounded rectangle over the timeline of the Sequencer. For example, when the frame marker is in the first frame, the timeline shows the blue rounded rectangle over 0+01. The plus sign is merely a visual separator of the two integers, the first of which is the number of seconds into the video and the second integer is the frame number. Go ahead and do some scrubbing (left and right) by dragging the current frame marker or by pressing the left and right arrow keys.

Video Splicing

Add your second video to the Sequencer (from **Add** menu, select **Movie**) and then complete the following steps to cut a section of video from one movie strip and insert it into the other.

1. Move the current frame marker to the position three seconds into the video (3+00)
2. Click the top video strip. If the video you just added to the Sequencer has an audio track (which many, but not all, videos have), hold the Shift key down and click the audio track strip
3. Drop down the **Strip** menu and select **Split** (or just press the k key)
4. Drag the strip (or strips if you have audio) on the right side of the splice two seconds down the timeline (to 5+00)
5 Now click the first video strip you inserted into the Sequencer and, if it has an audio track, hold the Shift key down and select the audio track as well

6. Press the k key (or drop down the **Strip** menu and select **Split**)
7. Move the current frame marker to the position five seconds into the video (5+00)
8. Press the k key (or drop down the **Strip** menu and select **Split**)
9. Drag the two-second strip (or strips) up and drop it (or them) into the two-second gap you created previously
10. Test by clicking the play button or by scrubbing. *Blender* seems to cache the video and audio data after playing through the splices (once or twice), so you may notice a pause at the splice points initially, but eventually they run through smoothly
11. Delete the strips that are not part of the final movie by clicking each strip and pressing the x key
12. Save the project

Scrolling Text

One could scroll a single line of text using the one-line text editing feature in *Blender* (Add; Text), but to scroll multiple lines we will insert an image containing the text to scroll (since *Blender* does not permit importing of documents from word processors) and then add a Transform to animate, or scroll, the image containing the text. The image with filename, *sampleTextToScroll.png*, which is available in the Electronic Resources for this book, will be used in the steps below. Even so, you are free to use your word processing software to create whatever text you wish. Then save your text as a PDF file and open it in *GIMP*, for instance, to set the width to 1912 pixels. Also, in *GIMP*, remove the background: Click the *Select with Color* button in the Tool Palette; click anywhere on the white background of your image; select Cut from the Edit menu; save as PNG.

1. In the *Blender* video editor, drop down the **Add** menu and select **Image/Sequence**
2. Drag the image strip to the 5-second point (5+00)
3. Drag the end of the strip to frame 241
4. Drop down the **Add** menu and proceed through **Effect Strip** to **Transform**
5. Double the size of the image by changing the **Scale** values for both X and Y to 2
6. With the current frame marker at 5+00, adjust the vertical position of the image (which contains text, so it appears the text is moving). In the Position field for Y, you may drag left to lower the position of the image of text until the text scrolls below the Preview window. Alternatively, you may set the Y position to -141
7. Next, to create an animation you first insert a keyframe by pressing the i key. Next, press Shift-Right Arrow to move the current frame marker to the final frame. Set the Y position to 137 (or adjust it by clicking in the Y position field and dragging to the right until the text scrolls above the top of the Preview window). To complete the animation, press the i key to insert another keyframe
8. Lastly, scroll down the Properties window to **Compositing**. Notice that the Blend property is set to Replace (by default). Using the drop-down menu for the Blend property, change this to **Alpha Over**
9. Test the animation by scrubbing or by clicking the **Play** button
10. Save the project

Rendering

Once you have completed the editing process and the project has been saved (per the steps above), there are some properties to set prior to rendering the movie.

1. In the **Scene properties** window (which is beside the Preview window), scroll down to the **Output properties**. Click the folder icon; navigate to the folder in which the rendered

movie will be saved; enter a file name; and click the **Accept** button. (*Blender* adds the frame number range to your file name.)

2. From the **File Format** drop-down menu, select FFmpeg video
3. Click **Encoding** to view the Encoding properties (if not already visible)
4. From the **Container** drop down menu, select MPEG-4
5. For **Video Codec**, ensure that H.264 is selected
6. For **Output Quality**, select High Quality
7. For **Encoding Speed**, select Realtime
8. In the **Audio** section (again, expand to view if necessary), select AAC for **Audio Codec**
9. Lastly, drop down the **Render** menu (upper-left corner of the Video Editing window) and select **Render Animation**. This will open a window enabling you to view your movie as it renders. Close the window to return to the video editing interface

3.3.6 Creating Web Pages in HyperText Markup Language (HTML)

Everyone who has used a web browser has displayed web pages and, consequently, has what I call the *consumer* conception of a web page. They know that contemporary web pages display text and images, as well as play audio and video. Further, they know that contemporary web pages enable interactivity. Just as users of web browsers have developed those consumer conceptions, drivers of automobiles have developed consumer conceptions of cars and trucks. Drivers of cars and web browsers know how to operate them, but unless they have studied engine manufacturing and HTML, they are not able to produce a car or web page from raw materials. We might say they lack a *producer* conception of cars and web pages.

Yet they might have partial *producer* conceptions of vehicles and web pages if they know something about the assembly of cars and web pages. For example, someone who has bolted a tire to a car, or observed that process, gains a partial conception of car assembly. One also gains a partial producer conception by assembling, or watching someone assemble, prefabricated furniture. In the case of web page development, using an app like *WordPress* or *Adobe Dreamweaver* to assemble a web page provides a partial producer conception. In that case, one learns that a web page can be produced by dragging and dropping an image into the app and typing text into the app. Since that assembly process conceals the inclusion of elements in the HyperText Markup Language (HTML), which enable the image and text to be rendered by web browsers, such web page assembly affords development of a partial producer conception. A complete producer conception of web page development begins with knowledge of HTML.

Even without knowledge of HTML, one may have a conception of a markup language. To convey recommendations for improving a paragraph, essay, or other written work, a proofreader inserts symbols that mark up the original text. In that sense, the proofreading symbols comprise a markup language. In HTML, *tags*, which are called *elements* in formal descriptions of HTML, are used to mark up text in order to specify how a web browser is to render the text and images in a web page. HTML tags have changed somewhat since their introduction in 1990 by Tim Berners-Lee (2000), yet many of the original tags are still used today, such as the ones used to underline and italicize text, as well as the tag used to make text appear bold. Most tags come in two parts, a start tag and an end tag. For example, to make text appear bold, one surrounds the text to be bold with `<b>` **and** `</b>` . HTML is not case sensitive, so one could use `<B>` **and** `</B>` , or even mix the case of the start and end tags, as in `<B>` **and** `</B>` but for human readability, consistency (either all lowercase or all uppercase) is much preferred. There is a formal, pristine structure to each web page, which we consider below, but to get started, open a text editor (e.g., *NotePad* in Windows or *TextEdit* in MacOS); enter the line below; save the file as *testPage.html*; and double-click the file icon in the operating system folder to view the web page as rendered by your web browser.

```
This text is <i>italicized</i>
```

The line above, which includes one markup tag in HTML, directed your web browser to display "italicized" in italics, whereas letters not within `<i>` and `</i>` were displayed in the default font used by your web browser. In HTML, the one difference in the syntax of the start and end parts of a tag is that the second character of the end part is /. Both start and end parts surround the tag letter, abbreviation, or name with <>. A small number of HTML tags have only one part, such as the tag for a line break, which is `<br>`. Edit *testPage.html* as shown below; save the file; and reload it in your web browser (Ctrl-R in Windows; Command-R in MacOS).

```
This text is <i>italicized</i><br>
This text is <b>bold</b><br>
This text is <u>underlined</u><br>
This text is <i><u>italicized and underlined</u></i><br>
This text is <b><i><u>bold, italicized, & underlined</u></i></b>
```

Given the HTML above and the output in your web browser, text within `<b>` and `</b>` appears bold. Further, text within the `<u>` and `</u>` appears underlined. Moreover, tags are nested to produce text in multiple styles. In those cases, the tag presented last must be closed first. Accordingly, the order in which the tags appear in the last line of the HTML above,`<b><i><u>`, is reversed, in order to close the tags properly.

Edit *testPage.html* again to try one or two more combinations of nested font styles. For example, you could add a line to display bold and italicized text. Also, remove one or more of the `<br>` tags to view how the page is displayed without line breaks. Then, in *testPage.html*, copy and paste the five lines above. In the pasted five lines, delete the <br> tags and surround each line with the paragraph tag (`<p> </p>`), as shown below.

```
<p>This text is <i>italicized</i></p>
<p>This text is <b>bold</b></p>
<p>This text is <u>underlined</u></p>
<p>This text is <i><u>italicized and underlined</u></i></p>
<p>This text is <b><i><u>bold, italicized, & underlined</u></i></p>
```

Given your progress to this point, you will have noticed that web browsers render web pages in accordance with tags that mark up text. Next, the pristine structure of web pages will become evident.

HTML documents begin with `<html>` and finish with `</html>`. Further, HTML documents include two sections, the head and body, as shown below. Continue to make changes to *testPage.html*, as shown below; then save the file and reload the page to view the rendered results in your web browser. Just note that changes outside the `<body> </body>` section will not change the appearance of the web page.

```
<html>

<head>
</head>

<body>
  This text is <i>italicized</i><br>
  This text is <b>bold</b><br>
  This text is <u>underlined</u><br>
  This text is <i><u>italicized and underlined</u></i><br>
  This text is <b><i><u>bold, italicized, & underlined</u></i></b>
```

```
        <p>This text is <i>italicized</i></p>
        <p>This text is <b>bold</b></p>
        <p>This text is <u>underlined</u></p>
        <p>This text is <i><u>italicized and underlined</u></i></p>
        <p>This text is <b><i><u>bold, italicized, & underlined</u></i></p>
</body>

</html>
```

Indenting is not required by web browsers, but it is helpful to human beings who seek to separate sections visually. The head section above is empty, but it usually contains a title tag. The text in the `title` tag of the head section displays in the browser tab above the web page you are creating. We will use the following HTML to specify the tab title within the head section.

```
<head>
  <title>My Test Page</title>
</head>
```

We will also include the standard language attribute in the `<html>` start tag. In this case, we will alert the web browser to use of English (end), which changes the `<html>` start tag to `<html lang="en">`. Lastly, for pristine HTML documents, the HTML document type is specified on the first line of the file by including `<!DOCTYPE html>`.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>My Test Page</title>
</head>

<body>
  This text is <i>italicized</i><br>
  This text is <b>bold</b><br>
  This text is <u>underlined</u><br>
  This text is <i><u>italicized and underlined</u></i><br>
  This text is <b><i><u>bold, italicized, & underlined</u></i></b>

  <p>This text is <i>italicized</i></p>
  <p>This text is <b>bold</b></p>
  <p>This text is <u>underlined</u></p>
  <p>This text is <i><u>italicized and underlined</u></i></p>
  <p>This text is <b><i><u>bold, italicized, & underlined</u></i></p>

</body>

</html>
```

The general structure of a pristine web page (as exemplified by W3C/WHATWG, the leading authorities on HTML) appears below, though the "en" language code may be replaced by a different code. The two-letter and three-letter primary language codes are available at the following web page: https://www.iana.org/assignments/language-subtag-registry/language-subtag-registry

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Text on Browser Tab</title>
</head>
<body>
  Content goes here
</body>
</html>
```

From this point, we will create web pages with that pristine structure.

For reference purposes, the following two links lead to the W3C/WHATWG documentation for HTML and its elements (tags).

https://html.spec.whatwg.org/multipage/

https://html.spec.whatwg.org/multipage/semantics.html#semantics

Information about the World Wide Web Consortium(W3C) is available at https://www.w3.org/. Details about the Web Hypertext Application Technology Working Group (WHATWG) are available at https://whatwg.org/. Click the button to their FAQ.

For information about the collaboration between W3C and WHATWG, see their agreements, which are available at the following web pages.

Memorandum of Understanding Between W3C and WHATWG

https://www.w3.org/2019/04/WHATWG-W3C-MOU.html

W3C/WHATWG Relationship Update

https://www.w3.org/2021/06/WHATWG-W3C-MOU_2021_update.html

Creating a Website

In this section, we create a two-page website about instruments in an orchestra. Two web pages are sufficient to learn how to hyperlink web pages, which is fundamental to website and web page development. When you have reached the end of this section, you may develop more than two web pages about orchestral instruments if you wish.

To begin, enter the HTML below into your text editor and save the file as *orchestra.html*. As before, double-click the file icon to open the page in your web browser. Then, as you proceed through this section, continue to save the file, and reload the page in your web browser whenever you make changes to the *orchestra.html* file.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Orchestral Instruments</title>
</head>
<body>
  <h1>Instruments in an Orchestra</h1>

  <p>Orchestral instruments appear in the following sections.
  <ul>
      <li>Strings</li>
      <li>Woodwind</li>
```

```
        <li>Brass</li>
        <li>Percussion</li>
    </ul>
    Let’s consider instruments in each section.
    </p>
</body>
</html>
```

The HTML above follows the pristine structure of a web page. The browser tab title is set to Orchestral Instruments. The first line of HTML in the body section, which defines the appearance of the web page and is “consumed” by the user, presents `Instruments in an Orchestra as` the title. In *orchestra.html*, the text of that title is surrounded by the `<h1>` start tag and `</h1>` end tag. In your web browser, notice the size of the title, which has been rendered in accordance with the h1 tag. Then change `<h1>` to `<h2>` and `</h1>` to `</h2>`. Save and reload the page. Notice that the title has decreased in size. Now change `<h2>` to `<h3>` and `</h2>` to `</h3>`. Save and reload the page again. Continue to reduce the heading number by one until you reach `h6`, which is the smallest heading size in HTML.

In addition to the title, a list of orchestral sections appears on the web page. Technically, in HTML, it is an unordered list (ul) because items in the list appear within the `<ul>` start tag and the `</ul>` end tag. Each item within that list is surrounded by the `<li>` start tag and `</li>` end tag. Given the display of the web page in your web browser, you can verify that the list is indented and that a bullet point appears in front of each item in the list.

To see the appearance of an ordered list (ol), change `<ul>` to `<ol>` and `</ul>` to `</ol>`. Notice that the bullet points have become digits starting with 1. Return the size of the title to the largest heading size (h1) and restore the unordered list. Then insert the following HTML into *orchestra.html* to add a heading for each orchestral section and a list of instruments within each section.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Orchestral Instruments</title>
</head>
<body>
    <h1>Instruments in an Orchestra</h1>

    <p>Orchestral instruments appear in the following sections.
    <ul>
        <li>Strings</li>
        <li>Woodwind</li>
        <li>Brass</li>
        <li>Percussion</li>
    </ul>

    Let’s consider instruments in each section.
    </p>

    <h2>Strings</h2>
    <ul>
        <li>Violins</li>
        <li>Violas</li>
        <li>Cellos</li>
        <li>Double Basses</li>
        <li>Harp</li>
    </ul>

    <h2>Woodwind</h2>
    <ul>
        <li>Piccolos</li>
        <li>Flutes</li>
        <li>Oboes</li>
        <li>Clarinets</li>
        <li>Bass Clarinets</li>
        <li>Bassoons</li>
    </ul>

    <h2>Brass</h2>
    <ul>
        <li>Horns</li>
        <li>Trumpets</li>
        <li>Trombones</li>
        <li>Tubas</li>
    </ul>

    <h2>Percussion</h2>
    <ul>
        <li>Marimba</li>
        <li>Xylophone</li>
        <li>Tubular Bells</li>
        <li>Glockenspiel</li>
        <li>Vibraphone</li>
        <li>Cymbals</li>
        <li>Snare Drum</li>
        <li>Bass Drum</li>
        <li>Timpani</li>
        <li>Miscellaneous (e.g., triangle, claves, guiro)</li>
    </ul>

    <hr>
</body>
</html>
```

Notice the `<hr>` tag, which renders a horizontal rule (line) below the list of percussion instruments. A style attribute may be added to various tags, including the `<hr>` tag, to alter the appearance of the element. For example, edit the `<hr>` tag to include the width attribute, as in `<hr style="width:80%">` and then save the file. After viewing the result, edit the tag again to include a height property, as in `<hr style="width:80%; height:3px">`.

Before proceeding to the final two elements in this section, you may alter or remove the style attribute in the `<hr>` tag if you wish.

Each image in a web page is inserted using the `<img>` tag, which includes two attributes. The first attribute, `src` (short for source), provides the path to the image file. Typically, the path to the image file is relative to the folder containing the web page. In some cases, though, an absolute path is used, but such use would greatly diminish the flexibility to port the web page to a different server. Accordingly, we will use a relative path. Store the image file, `orchestraLayout.png` (which is included in the electronic resources for this book), in the same folder as the web page.

In doing so, the `src` attribute need only be set to the image file name, `orchestraLayout.png`, as shown below. The second attribute, `alt` (short for alternative text), is a string of text used to describe the purpose of the image, which is provided to increase accessibility. Insert the following line after the `<hr>` tag and before the `</body>` tag.

```
<p>A Common Layout of Orchestral Instruments</p>
<img src="orchestraLayout.png" alt="in an orchestra, the conductor is
closest to the string section, followed by the woodwind, brass, and
percussion sections">
```

Lastly, to this web page, we will add two more lines of HTML in order to hyperlink this page to two other web pages, one link will enable users to go the home page of the Vienna Philharmonic Orchestra for more information about an orchestra. The second hyperlink will go to the web page we create next, which would offer more information about violins. The web page about violins, the second page in our website, will also link back to this page, *orchestra.html*.

The anchor tag, `<a> </a>`, is used to implement hyperlinking, which enables users to visit other web pages and websites. The anchor tag includes the `href` attribute, which is used to specify the target URL. Whenever a user clicks the link, the web browser fetches the file at the target URL and, in accordance with the HTML tags in the file, renders its content in the browser. The target URL in a hyperlink may be either absolute or relative. As is common practice, we will use an absolute URL to enable users to visit a website external to our two-page website. In our case, that external site will be the website of the Vienna Philharmonic Orchestra. Let's add that anchor tag now by inserting the following HTML after the `img` tag above and before the `</body>` tag.

```
<p>For more information about orchestral instruments, visit the website of
the <a href="https://www.wienerphilharmoniker.at/en">Vienna Philharmonic
Orchestra</a>.</p>
```

All absolute URLs begin http:// or https:// and are followed by the server name (or IP address). In this case, the presence of the server name, `www.wienerphilharmoniker.at`, confirms that the target URL uses absolute addressing. Server names often end with .com, .org, .net, or .edu, but can also end with a country code, which in this case is `.at` for Austria. In contrast to that use of absolute addressing, we use a relative reference to link to the second web page in our website.

As shown below, edit `<li>Violins</li>` to make the first list item in the Strings section a relative hyperlink.

```
<li><a href="violins.html">Violins</a></li>
```

If you test that link now, it will fail because we have yet to create the second web page. To make that link functional, open a new document in your text editor, enter the HTML below, and save the file as *violins.html*.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Violins</title>
</head>
<body>
  <h1>Violins</h1>
  Violins are great!

  <p><a href="orchestra.html">Home</a></p>
</body>
</html>
```

Note that either text or an image can be hyperlinked. In the example above, the word "Home" is hyperlinked. As an aside, such hyperlinked text is called *hypertext*, which explains the HT in HTML. Further, given your experience using markup tags, the ML in HTML has also become evident. Now, your *producer* conception of HTML includes comprehension of the acronym HTML. Returning to development of the web page for violins, replace Home with the line of HTML below. In addition, place the *homeIcon.png* file (which is included in the Electronic Resources for this book) in the same folder as the *violins.html* file.

```
<img src="homeIcon.png" alt="button to return to home page">
```

The following files are included in the Electronic Resources for this book: *orchestra.html, orchestraLayout.png, violins.html, homeIcon.png*

Additional HTML Tags

To learn more HTML tags, open a new document in your text editor and enter the HTML below, which displays a table of data. A table is created using the `<table>` start tag and `</table>` end tag. Each row in the table is surrounded by the `<tr>` start tag and `</tr>` end tag. Column headings are surrounded by `<th>` and `</th>`; each table cell withing a row is surrounded by `<td>` and `</td>`. Save the file using a file name that ends `.html` and then open the web page in your web browser. When creating file names for web pages, I recommend using only letters and digits; including even one special character, such as % ^ $!, and the character generated by pressing the space bar, often makes the page inaccessible.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Text on Browser Tab</title>
</head>
<body>
  <table>
    <tr><th>Product</th><th>Type</th><th>Price</th></tr>
    <tr><td>Panama</td><td>Hat</td><td>$20</td></tr>
    <tr><td>Bowler</td><td>Hat</td><td>$25</td></tr>
    <tr><td>Trilby</td><td>Hat</td><td>$25</td></tr>
    <tr><td>Fez</td><td>Hat</td><td>$20</td></tr>
    <tr><td>Ivy Cap</td><td>Hat</td><td>$15</td></tr>
    <tr><td>Beret</td><td>Hat</td><td>$20</td></tr>
    <tr><td>Floppy</td><td>Hat</td><td>$10</td></tr>
    <tr><td>Cowboy</td><td>Hat</td><td>$15</td></tr>
  </table>
</body>
</html>
```

HTML implements three types of lists, two of which you have already encountered, namely the ordered list and unordered list. In HTML, there is also a definition list (`<dl> . . . </dl>`), which contains a series of terms and definitions. Each term is surrounded by `<dt>` and `</dt>` and its definition is surrounded by `<dd>` and `</dd>`. Insert the following HTML into your web page; save the file and view the results.

```
<dl>
     <dt>Web browser</dt>
     <dd>Software that enables users to access and view web
       pages.</dd>
     <dt>Web server</dt>
     <dd>Software that fulfills web page requests.</dd>
     <dt>Client Server Technology</dt>
     <dd>Software that enables data transfers. A web browser is an
       example of client software because a web browser submits a
       request to a web server every time the user seeks web page
       content.</dd>
</dl>
```

One at a time, insert the `audio` and `video` tags below into the body section of your web page. Note that the command `autoplay` is optional and often absent. Include `autoplay` only when you want the media (audio or video) to play when the page opens.

```
<audio autoplay controls>
  <source src="soundFile.mp3" type="audio/mpeg">
</audio>

<br>
<video width="320" height="240" autoplay controls>
  <source src="movieFile.mp4" type="video/mp4">
</video>
```

At times, you may find that a web browser will not render playback controls, which will prevent the user from stopping the audio or video. In those cases, you may include both the `autoplay` and `controls` keywords, as in the examples above. For those tags to work, you will need to supply the *soundFile.mp3* and *moveFile.mp4* files, which you can do, for instance, by downloading an mp3 file and an mp4 file in the public domain and changing the file names. Store those files in the same folder as your web page file. Alternatively, you could create a folder or folders for media subordinate to the folder in which your web page is saved. Doing so is very helpful when creating websites that contain multiple images and web pages, for instance. If you do create a subordinate folder called `videos` and store `movieFile.mp4` in that folder, then be sure to change the `src` attribute. In the HTML above, the source file for the video is set using the following attribute.

```
src="movieFile.mp4"
```

With a folder called `videos` subordinate to the folder in which your web page is stored, the relative path to the video file changes to make explicit the folder name containing the video file. In this case, the `src` attribute changes to the following.

```
src="videos/movieFile.mp4"
```

In Chapters 5–8, additional HTML tags are introduced in order to make web pages interactive.

3.3.7 Enhancing Web Pages with Cascading Style Sheets (CSS)

Web pages are rendered in accordance with hundreds of default settings. For example, font sizes for the heading tags (H1–H6) are predetermined by default settings, as are margin settings and the amount of white space around paragraphs and lists, for instance. Those default settings are important for consistency, but web pages created entirely in HTML have limited visual appeal due to those settings, which cannot be altered with HTML alone. However, through use of Cascading Style Sheet (CSS) directives, web page developers can alter default settings and create web pages with enhanced visual appeal.

First, using CSS, let's add some visual appeal to a web page by adding styling directives to create a banner or header. Each styling directive assigns a value to a property, which overrides its default value and alters the appearance of the web page. Second, we will add an image and ensure that it is responsive by adding CSS styling directives that will scale (resize) the image automatically in order to fit variable screen dimensions. Third, we will add CSS styling directives to replace a banal list of hyperlinks with a navigation bar.

We begin with the following web page written entirely in HTML. Enter the following HTML in your text editor and save the file as *cellos.html*.

```
<!DOCTYPE html>
<html lang="en">
<head>
     <title>Cello</title>
</head>
<body>
     <h1>Cello</h1>
     <p>Cellos are great!</p>
</body>
</html>
```

There are three techniques a web page developer can use to apply CSS styling directives, *inline, internal*, and *external*. *Inline* styling directives are specified using the `style` attribute, which is inserted into HTML tags that appear in the `body` section of a web page. For example, edit the body, `h1`, and `p` tags as shown below.

```
<!DOCTYPE html>
<html lang="en">
<head>
     <title>Cello</title>
</head>
<body style="margin: 0px">
     <h1 style="font-size:70px; text-align:center;color:red">Cello</h1>
     <p style="margin-left:10px">Cellos are great!</p></body>
</html>
```

Altering the appearance of web pages by inserting *inline* directives combines HTML and CSS in the same tag, which makes it difficult to distinguish the CSS from the HTML. When the string of directives includes two or more properties, as in the h1 tag above, a semicolon is used to separate the properties. Unfortunately, it can be difficult to determine where one property ends and the next one begins when only a semicolon separates them. Hence, one may use *inline* directives when CSS styling directives need to be applied to a small number of tags. Since we want to add even more styling directives to the h1 tag than the three above and we want to be able to readily identify each styling property, we will restore the body section to its original form above. In addition, we will incorporate *internal* styling directives by adding the `<style>` start tag and `</style>` end tag to the head section of the web page and insert the styling directives within those tags, as shown below.

```
<!DOCTYPE html>
<html lang="en">
<head>
     <title>Cello</title>
     <style>
       body {
         margin: 0px;
       }
       h1 {
         font-size: 80px;
         text-align: center;
         color: #bb5500;
         background-color: #ffddbb;
         padding: 5px;
         margin: 0px;
         border-bottom: 5px solid #bb5500;
       }
       p {
         margin-left: 10px;
       }
     </style>

</head>
<body>
     <h1>Cello</h1>
     <p>Cellos are great!</p>
</body>
</html>
```

Notice that the `background-color`, `padding`, `margin`, and `border-bottom` properties have been added to the `h1` styling directive in addition to `font-size`, `text-align`, and `color`, which were present in the prior version of *cellos.html*. The effect of the `background-color` and `border-bottom` properties are readily apparent. To discern the effects of the `padding` and `margin` directives, delete them both and then enter them one at a time. With the header implemented through styling directives applied to the `h1` tag and to the `body` tag, we will now incorporate a responsive image.

Insert a blank line after the `</h1>` tag and before the `<p>` tag; then insert the following img tag.

```
<img src="celloImage.jpg" alt="front view of cello">
```

After saving the file and reloading the page in your web browser, notice that when you decrease the width of your web browser, part of the image disappears.

Insert the following styling directive to make the image responsive.

```
img {
  width: 100%;
  height: auto;
}
```

Sometimes it is helpful to insert the `max-width` property, as shown below to ensure that the image width does not exceed a particular value.

```
img {
  width: 100%;
  height: auto;
  max-width: 800px;
}
```

To further test the impact of the styling directives on the display of images, locate an image that is perhaps 2000 pixels wide and maybe half as high. Then edit the `src` attribute to display that image file. As before, decrease the width of your web browser to consider the utility of the styling directives. In this case, given the dimensions of *cellimage.jpg* (136 x 470), I recommend deleting the `img` styling directives.

In addition to visiting a web page about the cello, users would also want to view web pages about the violin, viola, double bass, and harp. To present a list of hyperlinks enabling navigation to those web pages, we nest five anchor tags within list elements of an unordered list. Presenting the unordered list without CSS styling will permit full functionality, but the visual appeal of the page leaves much to be desired. Insert the unordered list of hyperlinks between the heading and the image, as shown below.

```
<body>
    <h1>Cello</h1>

    <ul>
    <li><a href="violin.html">Violin</a></li>
    <li><a href="viola.html">Viola</a></li>
    <li><a href="cello.html">Cello</a></li>
    <li><a href="doubleBass.html">Double Bass</a></li>
    <li><a href="harp.html">Harp</a></li>
  </ul>

  <img src="celloImage.jpg" alt="front view of cello">
  <p>Cellos are great!</p>
</body>
```

Before adding web pages in order to make the hyperlinks functional, we will add CSS styling directives to improve the visual appeal of the page. The list of bullet points arranged vertically on the left has disrupted the visual flow from header to image. A horizontal orientation of the hyperlinks in a color palette and background color consistent with the header will much improve the visual appeal of the page. We will also add a hover color, which will serve to highlight each link for desktop and laptop users.

To implement the navigation bar, we will first place the unordered list in a separate division using the HTML `div` tag. Then we will apply styling directives to the division. Since multiple HTML divisions often appear within the body section of a web page, we will store the styling directives in a class and explicitly assign that styling class to the navigation bar division, as shown below.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Cello</title>
    <style>
```

```
        body {
          margin: 0px;
        }
        h1 {
          font-size: 80px;
          text-align: center;
          color: #bb5500;
          background-color: #ffddbb;
          padding: 5px;
          margin: 0px;
          border-bottom: 5px solid #bb5500;
        }
        p {
          margin-left: 10px;
        }

        .navbar {
          color: #bb5500;
          background-color: #ffddbb;
        }
      </style>

</head>
<body>
    <h1>Cello</h1>

    <div class="navbar">
    <ul>
      <li><a href="violin.html">Violin</a></li>
      <li><a href="viola.html">Viola</a></li>
      <li><a href="cello.html">Cello</a></li>
      <li><a href="doubleBass.html">Double Bass</a></li>
      <li><a href="harp.html">Harp</a></li>
    </ul>
    </div>

    <img src="violinImage.jpg" alt="front view of cello">
    <p>Cellos are great!</p>
</body>
</html>
```

We now have the navigation hyperlinks in an HTML division with the styling directives in our `navbar` class applied to the division. Most evident at this point is the proper background color for the navigation bar. Next, we will eliminate the white space between the header and navigation bar. We will also remove the circles that represent the bullet points. Insert the following CSS after the closing brace of the navbar class } and before the `</style>` tag.

```
ul {
    list-style-type: none;
    margin: 0px;
    padding: 0px;
    overflow: hidden;
}
```

Next, we will change the default vertical orientation of list elements by floating list elements left. This will position the navigation hyperlinks next to each other along a horizontal line. We will need to add some padding in order to separate the hyperlinks along the horizontal line. First, though, float the list elements left by inserting the following CSS after the closing brace of the styling directives for the `ul` tag `}` and before the `</style>` tag.

```
li {
    float: left;
}
```

Next, add the padding for the anchor tags within the list elements by inserting the following CSS after the closing brace of the styling directives for the `li` tag `}` and before the `</style>` tag.

```
li a {
    display: block;
    color: #bb5500;
    padding: 10px 14px;
    text-decoration: none;
}
```

Add the styling directive for the hover property of anchor tags by inserting the following CSS after the closing brace of the styling directives for the `li a` tag `}` and before the `</style>` tag.

```
li a:hover {
    background-color: #ffbb88;
}
```

Lastly, let's center the image and the caption, Cellos are great! Add the styling directive for a new class called center by inserting the following CSS after the closing brace of the styling directives for the `li a:hover` tag `}` and before the `</style>` tag. Also, to apply the styling directive in the `center` class to both the image and the caption, both the img tag and the caption must be enclosed by the paragraph tag to which the `center` class is applied, as shown below in the three lines of HTML immediately above the `</body>` end tag.

```
.center {
    text-align: center;
}
```

The HTML and CSS for the complete web page, *cellos.html*, appears below. In addition, *cellos.html* and *cellImage.jpg* are available in the Electronic Resources for this book.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Cello</title>
  <style>
      body {
        margin: 0px;
      }
      h1 {
        font-size: 80px;
        text-align: center;
        color: #bb5500;
```

```
          background-color: #ffddbb;
          padding: 5px;
          margin: 0px;
          border-bottom: 5px solid #bb5500;
        }
        p {
          margin-left: 10px;
        }

        .navbar {
          color: #bb5500;
          background-color: #ffddbb;
        }
        ul {
          list-style-type: none;
          margin: 0px;
          padding: 0px;
          overflow: hidden;
        }

        li {
          float: left;
        }
        li a {
          display: block;
          color: #bb5500;
          padding: 10px 14px;
          text-decoration: none;
        }
        li a:hover {
          background-color: #ffbb88;
        }
      </style>

</head>
<body>
      <h1>Cello</h1>

      <div class="navbar">
      <ul>
        <li><a href="violin.html">Violin</a></li>
        <li><a href="viola.html">Viola</a></li>
        <li><a href="cello.html">Cello</a></li>
        <li><a href="doubleBass.html">Double Bass</a></li>
        <li><a href="harp.html">Harp</a></li>
      </ul>
      </div>

      <p class="center">
      <img src="celloImage.jpg" alt="front view of cello"><br>
      Cellos are great!</p>
</body>
</html>
```

We now have a web page that respects visual design principles regarding contrast, alignment, proximity, and consistency (Williams, 2015). The final step is to make the links functional, which I encourage you to do by copying *cellos.html* to *violins.html*, *violas.html*, *doubleBasses.html*, and *harps.html*. You may do that copying and pasting of files in your operating system folder or within your text editor. Then open each of those four files in your text editor; change all instances of *cello* to the instrument name for that page; and save the page when complete. Test the links to ensure functionality.

3.3.8 Creating 3D Graphics and Animations in **Blender**

Like *GIMP* and *Inkscape*, *Blender* is free of financial cost and open source. Unlike *GIMP* and *Inkscape*, which facilitate two-dimensional image editing, *Blender* provides tools for the design and development of three-dimensional (3D) graphics. One may describe such design and development work as modeling, sculpting, or rigging, for instance. The *Blender* website (blender.org) includes a web page called Features (https://www.blender.org/features/), which describes how *Blender* facilitates modeling, sculpting, rigging, animation and simulation, and video editing. *Blender* even has a Python programming interface. The *Blender* website also includes a Support page (blender.org/support) from which one can access tutorials and documentation. To download the *Blender* installation software, simply click the big *Download Blender* button on the Download page (blender.org/download). For such a feature-rich program, one might expect lengthy download and installation times. In my case, the total time for both downloading and installation was about one minute. Of course, those times vary in accordance with Internet bandwidth and computer hardware speed.

After the one setup dialog, through which you may select a Dark Mode or Light Mode interface, *Blender* opens to the viewport with one cube in the center. The viewport is somewhat like a canvas in 2D editing software, but the viewport has a third dimension. Whereas a 2D graphics canvas is always viewed from the same fixed perspective, the 3D viewport in *Blender* enables the developer to rotate around objects, as well over and under objects. With a trackpad, one rotates around the *Blender* viewport using a two-finger drag. With a three-button mouse, one rotates around objects using a middle button drag. Rotation can also be achieved by clicking and dragging either the X, Y, or Z circle of the axes interface element in the upper-right corner of the viewport. This rotation capability for viewing a 3D object immediately provides one with a sense for how 3D graphics editing is fundamentally different than editing 2D graphics. In addition to rotating the viewport, it is also helpful to zoom and pan. To zoom in or out, hold the Ctrl key and use a two-finger drag on a trackpad or hold the Ctrl key and use a middle mouse button drag. Alternatively, one may click and drag the zoom icon (the magnifying glass with a + sign in it), which is very close to the axes interface element used for rotation. To pan, which enables one to slide the viewport around, hold the Shift key down while moving two fingers on the trackpad or dragging with the middle mouse button down. Alternatively, one may click and drag the hand icon, which is beside the zoom icon. If, at any time, the cube or other object disappears from the screen, you may find it by zooming out and, once visible again, slide it back toward the middle of the viewport before zooming back in. Alternatively, press the Home key if your keyboard has one; otherwise, press the Function key (fn) and the Left arrow key. Parenthetically, if you ever find your computer not responding to keyboard input, press the Escape key once or twice before pressing the key or key sequence again.

Having mastered those viewport navigation techniques, let's proceed with manipulating and modifying 3D graphics. Fundamentally, one may move, rotate, or scale an object. The toolbar for such transformations appears over the left side of the Viewport. The Arrow button at the top of the toolbar is the selection tool. Hover the cursor over each of the other buttons in the toolbar to discover the Move, Rotate, and Scale tools. Select the Move tool, if not already selected; then move the cursor over the cube and tap the mouse button or trackpad. Notice that you can move the selected object, the cube in this case, along either the X, Y, or Z axis by clicking and dragging the red, green, or blue ray (line with arrow) that appears over the cube. You can move the object in any direction by clicking inside the white circle from which the rays extend; then drag the object to the desired position. In *Blender*, you are free to Undo and Redo operations (Edit; Undo; or Edit: Redo; or use the standard keyboard shortcuts Ctrl-Z or Command-Z). In addition to movement (translations), objects can also be rotated or scaled in a similar manner. Select the Rotate tool in the toolbar. Rather than rays, the *Blender* interface displays three arcs and one white circle over the object. Click and drag each of the red, green, and blue arcs to rotate the object around the X, Y, and Z axes respectively. Click and drag the white circle to rotate in any direction. Now try resizing the object by clicking the Scale tool in the toolbar; then click and drag either the red, green, or blue extension (which is similar to a ray, except that a cube appears at the end of the line segment rather than the arrow used for translations). Dragging the white circle scales the object along the X, Y, and Z axes simultaneously.

When moving, rotating, or scaling the cube, you may have noticed numerical values changing in the Properties window to the right of the Viewport. There is also an optional sidebar that displays the same numbers, which can be viewed by dropping down the View menu and clicking the checkbox beside Sidebar. These numbers pertain to the X, Y, and Z axis values for the object's location, rotation, and scale. At times, rather than dragging an object to change its location, rotation, or scale, you may find it helpful to change the corresponding numerical values.

Also, at times, you may wish to adjust the Viewport to view your models without the default entities displayed. To hide the camera and the light, as well as the origin, axes, and the *Blender* cursor, click the Viewport Overlays button (which is in the group of buttons in the upper-right corner of the Viewport), and uncheck the entities you wish to hide. Check them again when you want them to reappear.

At this point, you can already navigate the Viewport and transform objects. Now we consider adding and deleting objects. Then we will apply fundamental skills to build a tractor. To add an object, drop down the **Add** menu, which is in the upper-left portion of the Viewport, and proceed through **Mesh** to the 3D object you wish to add, such as the cube, cylinder, or UV sphere, for instance. Add each of them; then move, rotate, and scale them. It is especially helpful to move each object after adding it because *Blender* inserts each new object at the same location (unless you move the origin). The insertion point is depicted by a circle, with alternating red and white segments, which surround the + sign. Unfortunately, *Blender* calls it the cursor. I will call it the *Blender* cursor to distinguish it from the cursor arrow of your computer. By default, when a new document is opened, the *Blender* cursor is positioned at the origin, which is 0,0,0 on the X, Y, and Z axes. You can move the *Blender* cursor by clicking its icon in the Toolbar and then dragging the *Blender* cursor. If you want to return the *Blender* cursor to the World origin, press Shift-S; w.

To delete an object in *Blender*, you might expect the key sequence Ctrl-X (on Windows) or Command-X (on a Mac) to work. However, *Blender* dropped the need to hold down the Ctrl or Command key, so after selecting the object to be deleted, press the x key, and then confirm deletion by pressing the d key (or Enter/Return).

To save your work, drop down the **File** menu and select **Save**. Navigate to the folder in which the file will be saved; specify the file name (*Blender* appends the .blend extension to the file name); and click the **Save Blender File** button. *Blender* may also store a backup file, with the blend1 file name extension. If you want to share your work with someone, you need only send the .blend file.

Now, capitalize on your fundamental skills to create a tractor like the one in Figure 3.4.

1. Open a new document by dropping down the **File** menu and proceeding through **New** to **General**

2. Rotate the viewport until the x-axis is positioned horizontally across the screen (i.e., parallel to the base of the viewport). When the x-axis is in position, the y-axis will split the viewport into two sides, left and right. The green line for the y-axis will appear bright at the base of the viewport and fade toward the top of the screen
3. With the cube selected, scale the x-axis to be three times the original size
4. Add a cube; scale its z-axis to 1.2; then move the cube to -2 on the x-axis and 2.2 on the z-axis
5. Add a cylinder; rotate both the y-axis and the z-axis 90 degrees; move the cylinder to -1.5 on the x-axis and to -1 on the z-axis; scale both the x-axis and the y-axis to 0.7 times the original size and scale the z-axis to 1.4 times the original size
6. Click the Selection Arrow in the toolbar. Only the cylinder should be selected. Drop down the **Object** menu and select **Copy Objects**; Drop down the **Object** menu again and select **Paste Objects**; Move the cylinder to 1.5 on the x-axis
7. Save the file

As an alternative to Copy Objects and Paste Objects, it is possible to Duplicate selected objects. To try this, select an object or objects to be copied; then drop down the **Object** menu and select **Duplicate Objects**. Notice that this feature automatically goes to Move mode after duplicating (copying and pasting) the selected object or objects. Any mouse or trackpad movement then alters the position of the copied object or objects, which is unexpected since duplicate does not imply movement. If you tried this option, simply delete the copied object or objects.

Now you have created a tractor, but it is looking rather "square." Let's bevel some edges.

1. Rotate the viewport about 90 degrees, so that the x-axis extends from the top-left to the bottom-right corners of the viewport and the y-axis extends from the bottom-left to top-right corners of the viewport
2. Select the tractor cabin (top cube). Notice that the drop-down menu in the upper-left corner of the viewport is set to Object Mode. Switch that to **Edit Mode** by pressing the **Tab** key, or by dropping down the menu and selecting Edit Mode. Some new icons appear in the toolbar on the left sidebar along with three icons to the immediate right of the drop-down menu, which now shows Edit Mode. Those three icons are buttons for selecting a vertex, edge, or face. Click the middle of those three buttons to enable edge selection. Of the two edges at the top of the cube, select the one that is at the front of the tractor
3. **Hover** over each of the new icons in the toolbar (the left sidebar) to identify the **Bevel** tool
4. Select the **Bevel** tool. In the viewport, a pin appears in the middle of the edge selected. Drag the circular end of that pin down until the edge becomes a rectangle about 25 pixels in width. Upon releasing the drag operation, a dialog appears. Increase the number of segments from 1 to 7
5. Click any empty place on the viewport to view the beveled edge
6. Repeat Steps 2 and 4 to bevel the trailing edge of the tractor cabin
7. Return to Object Mode and select the base of the tractor. Then switch back to Edit Mode to bevel the top and bottom edges of the front of the tractor, as well as the bottom edge at the back of the tractor
8. Save the file

Animation by Gravity

Some development environments enable simulations that uphold the laws of physics. To view this type of simulation in *Blender*, complete the following steps.

1. Open a new document by dropping down the **File** menu and proceeding through **New** to **General**

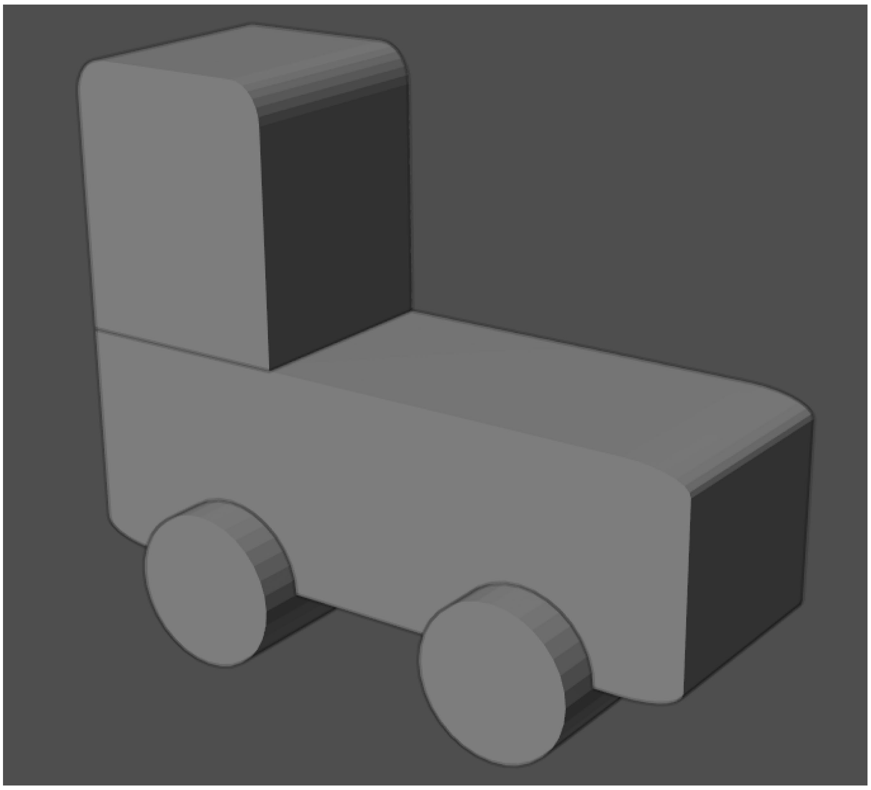


Figure 3.4 A tractor with beveled edges

2. Move the cube to -8 on the x-axis; 2 on the y-axis; and 4 on the z-axis
3. **Add a plane**; scale it to 5 on both the x-axis and the y-axis; tilt it by changing the rotation of the y-axis to 25 degrees; move it to -5 on the x-axis
4. Move around the viewport and notice the alignment of the cube relative to the plane
5. **Add a UV Sphere**; move it to -8 on the x-axis; -2 on the y-axis; and 4 on the z-axis
6. Hover the cursor over the tools in the Properties window, which borders the right side of the viewport. Notice that the Object Properties (including location, rotation, and scale) are displayed by default
7. With the UV Sphere still selected in the Viewport, click **Rigid Body**; set **Type** to Active. Notice the default settings for Friction and Bounciness in the Surface Response section. Set Bounciness to 0.8
8. **Select the plane** in the Viewport. Click **Rigid Body**; set **Type** to Passive. In the Surface Response section, set Bounciness to 3.0
9. **Select the cube** in the Viewport. Click **Rigid Body**; set Type to **Active**. There is no need to alter the Friction and Bounciness at this time
10. Save the file
11. Click the Play Animation button, which appears in the center of the bar immediately below the Viewport. Click the Pause button when the sphere and cube disappear
12. Click the Rewind button and adjust the Friction and Bounciness as you wish. Click the Play button again. Repeat this step as you wish, trying new values for Friction and Bounciness
13. **Select the plane**; duplicate (or copy and paste) it; set the rotation of the y-axis to -25 degrees and move it to 5 on the x-axis

14. Save the file
15. Rewind and play the animation. Reset values, including the angles of the planes (if you wish), and replay the animation until you are satisfied with the ricochet of the sphere and cube between the planes

3.3.9. 3D Printing

Many maker spaces include at least one 3D printer. Budgets at some schools permit the purchase of multiple 3D printers. Given the variety of features in 3D printers, multiple printers might enable fabrication of 3D models in multiple materials and at greater speeds than a single 3D printer (unless it is new and expensive). Regarding printing materials, plastics remain an inexpensive option, which makes them a good option for beginners. Commonly, 3D printers fabricate models in ABS (Acrylonitrile Butadiene Styrene) or PLA (Polylactic Acid) plastic. Through other materials are often more expensive, certain 3D printers can produce nylon, sandstone, ceramic, wood filament, and metal products, for instance. Note that metal products include tiny particles of metal that are fused by a resin. Wood products are produced by 3D printers by combining wood filaments, PLA plastic, and glue. Ceramic products fabricated by 3D printers use fine particles of clay combined with minerals and water. After fabrication, ceramic products are placed in a kiln. Some 3D printers fabricate products using various resins and polymers. Manufacturers of 3D printers provide comprehensive lists of the materials that their devices use for fabrication (e.g., see https://www.makerbot.com/stories/design/3d-printing-materials/).

As is common with technologies, the quality of 3D printers has increased over time while costs have decreased. For multiple years, budgets in many middle schools and high schools have permitted the purchase of one or more low-cost 3D printers. Even so, teachers may consider submitting a grant to various agencies to buy a 3D printer if money has been allocated to other instructional resources, or to purchase a higher-cost 3D printer. Before you proceed to the next paragraph and prepare to 3D print the tractor model you created in Section 3.3.8, it is worthwhile to distinguish between additive and subtractive manufacturing. Since 3D printers assemble products layer upon layer, they exemplify additive manufacturing. In contrast, chisels, saws, and laser cutters, for instance, remove unneeded materials from a chunk of raw material, which is regarded as subtractive manufacturing (Flynt, 2021).

Section 3.3.8 provides an overview of the 3D modeling program, *Blender*, as well as two sets of steps to create a 3D model of a tractor. After completing those two sets of steps to create your model tractor, you need only complete one additional step to prepare your model for submission to a 3D printer. After creating and saving your tractor model, drop down the File menu; select Export; and then select Stl, which will save your file in the stereolithography format (.stl), a format regarded by various groups as *Standard Triangle Language* and *Standard Tessellation Language*. The interpretation of the acronym is far less important than recognition that the Stl file format can be submitted directly to many 3D printers. Prior to printing your tractor, and other 3D models you create, 3D printing software generally provides an opportunity to modify a few parameters. You may wish to scale the model, for instance. Additionally, some 3D printers enable you to select colors for materials. Enjoy your model tractor!

3.3.10 Laser Cutting and Engraving

A well-ventilated device for laser cutting and engraving is essential in a contemporary maker space. The number and variety of such devices has greatly increased. Consequently, features and cost vary widely. Before purchasing a laser cutter/engraver for a maker space, one of the first features to consider is the software interface to the device. Some devices require use of proprietary software supplied with the device, which might be fine. Other devices cut and engrave in accordance with paths in a Scalable Vector Graphics (SVG) file. Laser cutting and engraving devices that support the popular SVG file format enable one to use free software, such as *Inkscape*, to use the device. Other key features to consider are cost and size of the cutting bed. The types of materials that can be cut and engraved is another factor to consider, while recognizing that many of these devices cut and engrave the same materials, such as wood, leather, acrylic, and soft metals (e.g., copper and aluminum). To experience both cutting and engraving, you may complete the steps below in *Inkscape* in order to create a nameplate.

1. Open *Inkscape*
2. Drop down the **File** menu and select **Document Properties…**; set the **Custom size** width and height to match the width and height of the cutting bed of the laser cutting/engraving device
3. Select the **Rectangle** tool; then click and drag across the canvas. Ensure that the width and height of the rectangle are the dimensions you want for your nameplate
4. If you wish to round the corners, drag the circle handles along the edges of the rectangle. Alternatively, you may add a Path Effect as follows: Drop down the **Path** menu; select **Path Effects…**; click the + button to add an effect; select **Corners (Fillet/Chamfer)**; then set the Radius to 25 pixels, for instance
5. Drop down the **Object** menu and select **Fill and Stroke…** (or click the Fill and Stroke icon) to open the Fill and Stroke Properties dialog
6. On the **Fill** tab, click the No paint button, which is an x. (Alternatively, one could fill the rectangle with a background color.) On the **Stroke paint** tab, click the Flat color button and click the RGB tab; set red to 255, green to 0, and blue to 0. On the **Stroke style** tab, set the Width to 0.2 mm. Now we have implemented the conventional settings for a laser cutter to cut, which has a maximum stroke width of 0.2 mm and a red color.
7. To enter your name, click the **Text** tool (the icon with the uppercase A and a vertical line) in the tool palette on the left. Click inside the upper-left corner of your rectangle and drag down and across to the lower-right corner. Type your name. Highlight your name and change the font to Zapfino, for instance, and change the font size to fit your name within your rectangle
8. Click the Arrow button in the Tool Palette and drag your name as necessary to position it as you wish. You may also move the rectangle if necessary. It may be helpful to select both the rectangle object and your name object (just hold the Shift key down while clicking the second object) and align the objects as follows: Drop down the **Object** menu; select **Align and Distribute…**; then click the buttons to center horizontally and vertically. You can also align the rectangle and name objects relative to the canvas by selecting Page from the top drop-down menu labelled **Relative to**
9. With the Arrow button active in the Tool Palette, click only your name. Open the **Stroke and Fill** properties (as in Step 5 above)
10. On the **Stroke paint** tab, click the Flat color button and set the RGB settings to 255 for blue and 0 for red and green. On the **Stroke style** tab, the default width of 1 mm is fine. By convention, for engraving with a laser cutter, the path color is set to blue and the width must be at least 0.5 mm
11. Save your file, perhaps as *namePlate.svg*
12. Before submitting the file to a laser cutting and engraving device, ensure that the device is well ventilated

Place your nameplate on the teacher's desk, or anywhere you wish, except the principal's desk.

References

American Library Association. (2018). *Design thinking.* http://www.ala.org/tools/future/trends/designthinking

Beckman, S. L. (2020). To frame or reframe: Where might design thinking research go next? *California Management Review, 62*(2), 144–162. https://doi.org/10.1145/223355.223477

Berners-Lee, T. (2000). *Weaving the web: The original design and ultimate destiny of the world wide web*. Harper Business.

Brown, T. (n.d.). *Design thinking defined*. https://designthinking.ideo.com/

Brownlee, J. (2019). *14 different types of machine learning.* Machine Learning Mastery. https://machinelearningmastery.com/types-of-learning-in-machine-learning/

Camacho, M. (2016). David Kelley: From design to design thinking at Stanford and IDEO. *She Ji: The Journal of Design, Economics, and Innovation, 2*(1), 88–101. https://doi.org/10.1016/j.sheji.2016.01.009

Carlgren, L., Rauth, I., & Elmquist, M. (2016). Framing design thinking: The concept in idea and enactment. *Creativity and Innovation Management, 25*(1), 38–57.
https://doi.org/10.1111/caim.12153

Cortez, P., & Silva, A. (2008). Using data mining to predict secondary school student performance. In A. Brito & J. Teixeira (Eds.), *Proceedings of the Fifth Future Business Technology Conference.* http://www3.dsi.uminho.pt/pcortez/student.pdf
Dataset is available as a semi-colon delimited file at the first URL below and as a comma delimited file at the second URL below:
https://archive.ics.uci.edu/ml/datasets/student+performance
https://www.kaggle.com/larsen0966/student-performance-data-set

Deibel, D., & Evanhoe, R. (2021). *Conversations with things: UX design for chat and voice*. Rosenfeld.

Flynt, J. (2021). *3D printing and woodworking – A perfect combination.* 3D Insider. https://3dinsider.com/3d-printing-and-woodworking/

GCF Global (n.d.). *The power of color.* https://edu.gcfglobal.org/en/beginning-graphic-design/color/1/

Gibbons, S. (2016). *Design thinking 101.* https://www.nngroup.com/articles/design-thinking/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* The MIT Press.

Gothelf, J. & Seiden, J. (2016). *Lean UX: Designing great products with agile teams*. O'Reilly Media.

Gour, R. (2020). *Big data in retail: Faster than airmail [2 astonishing case studies].* https://medium.com/dataflair/big-data-in-retail-faster-than-airmail-2-astonishing-case-studies-a06c4afd17dc

Greever, T. (2020). *Articulating design decisions: Communicate with stakeholders, keep your sanity, and deliver the best user experience* (2nd ed.). O'Reilly Media.

Hall, E. (2018). *Conversational design*. A Book Apart.

Hill, D. R. (2000). Give us the tools: A personal view of multi-modal computer-human dialogue. In M. M. Taylor, F. Néel, & D. G. Bouwhuis (Eds.), *The structure of multimodal dialogue II* (pp. 25–62). John Benjamins Publishing.

Hasso Plattner Institute Academy. (2021). *What is design thinking?* https://hpi-academy.de/en/design-thinking/what-is-design-thinking.html
The Six Phases of the Design Thinking Process
https://hpi.de/en/school-of-design-thinking/design-thinking/background/design-thinking-process.html

International Data Corporation. (2017). *Data age 2025: The evolution of data to life-critical*. https://www.import.io/wp-content/uploads/2017/04/Seagate-WP-DataAge2025-March-2017.pdf.

International Data Corporation. (2020, May 8). *IDC's Global DataSphere Forecast shows continued steady growth in the creation and consumption of data*. https://www.idc.com/getdoc.jsp?containerId=prUS46286020

International Standards Organization. (2018). *ISO 9241-11:2018(en). Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts*. https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en

Jamsa, K. (2021). *Introduction to data mining and analytics.* Jones & Bartlett Learning.

Jonauskaite, D., Alejandro Parraga, C., Quiblier, M., & Mohr, C (2020). Feeling blue or seeing red? Similar patterns of emotion associations with colour patches and colour terms. *iPerception, 11*(1), 1–24. https://doi.org/10.1177/2041669520902484

Kachchi, V., & Kothiya, Y. (2021, May 8). 4 types of data analytics every analyst should know – Descriptive, diagnostic, predictive, prescriptive. *Medium*.
https://medium.com/co-learning-lounge/types-of-data-analytics-descriptive-diagnostic-predictive-prescriptive-922654ce8f8f

Keegan, B. (2019). List of lists of datasets. *Medium*.
https://medium.com/information-expositions/list-of-lists-of-datasets-c9bf52370755

Kelley, D., & Kelley, T. (2013). *Creative confidence: Unleashing the creative potential within us all*. Currency Books. https://www.creativeconfidence.com/

Lau, C. H. (2019). *5 steps of a data science project lifecycle*. Towards Data Science.
https://towardsdatascience.com/5-steps-of-a-data-science-project-lifecycle-26c50372b492

Manovich, L. (2012). Trending: The promises and challenges of big social data. In M. K. Gold (Ed.), *Debates in the digital humanities* (pp. 460–475). University of Minnesota Press.
https://doi.org/10.5749/minnesota/9780816677948.003.0047
https://www.jstor.org/stable/10.5749/j.ctttv8hq
http://manovich.net/index.php/projects/trending-the-promises-and-the-challenges-of-big-social-data

Marr, B. (2021). *Walmart: Big data analytics at the world's largest retailer*.
https://bernardmarr.com/walmart-big-data-analytics-at-the-worlds-biggest-retailer/

Molich, R., & Nielsen, J. (1990). Improving a human-computer dialog. *Communications of the ACM, 33*(3), 338–348.

Nielsen, J. (1993). *Usability engineering*. Academic Press.

Nielsen, J. (2020). *10 usability heuristics for user interface design*. https://www.nngroup.com/articles/ten-usability-heuristics/

Nielsen, J., & Molich, R. (1990, April). Heuristic evaluation of user interfaces. In J. C. Chew & J. Whiteside (Eds.), *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, Seattle, WA (pp. 249–256). Association for Computing Machinery. https://doi.org/10.1145/97243.97281

Norman, D., Miller, J., & Henderson, A. (1995). What you see, some of what's in the future, and how we go about doing it: HI at Apple Computer. In *Proceedings of CHI '95*.
https://doi.org/10.1145/223355.223477

Norrie, M. A. (2019). An introduction to machine learning. In M. Ahmed & A.-S. Khan Pathan (Eds.), *Data analytics: Concepts, techniques, and applications* (pp. 3–32). CRC Press.

PayScale (2021a). *Average data scientist salary*. https://www.payscale.com/research/US/Job=Data_Scientist/Salary

PayScale (2021b). *Average UX designer salary*.
https://www.payscale.com/research/US/Job=UX_Designer/Salary

Siegele, L. (2020, February 22). Mirror worlds: A deluge of data is giving rise to a new economy. *The Economist*.
https://www.economist.com/special-report/2020/02/20/a-deluge-of-data-is-giving-rise-to-a-new-economy

Travis, D., & Hodgson, P. (2019). *Think like a UX researcher: How to observe users, influence design, and shape business strategy*. CRC Press.

Tyagi, H. (2020). *Data repositories for almost every type of data science project*.
https://towardsdatascience.com/data-repositories-for-almost-every-type-of-data-science-project-7aa2f98128b

User Experience Professionals Association International. (2000). *Designing the user experience*.
Discussion at https://uxpa.org/about-ux/
Poster at https://mprove.de/script/00/upa/_media/upaposter_11x17.pdf

Williams, R. (2015). *The non-designer's design book* (4th ed.). Peachpit Press.

XtremeBrandMakeover. (2010. *Color meaning chart*.
https://cdn.ymaws.com/www.ewald.com/resource/resmgr/Docs/Ewald-LogoColors.pdf

4 Computational Thinking

In 1980, Seymour Papert's revolutionary book, *Mindstorms: Children, Computers, and Powerful Ideas*, was published. That book, which continues publication today, encourages student inquiry with computers, in part by using the computer programming language Logo as a tool for problem solving. In the early 1980s, many sessions at teachers' conventions discussed strategies for teaching Logo, often by encouraging students to solve problems. The most popular type of problem involved the development of instructions that directed a virtual turtle to successfully draw a particular pattern. Whereas many sought for computers to provide instruction, Papert encouraged students to instruct the computer. He described such inquiry as *procedural learning*, whereas many educators today prefer to speak of *computational thinking*.

Beginning in 2006, with Jeannette M. Wing's landmark paper "Computational Thinking," educators interested in teaching inquiry with computers have been drawing on her ideas and adapting them. In the quarter century from 1980 to 2005, Wing had observed numerous cases of researchers and practitioners applying computers to practical problems. She asserted that this application process, which she dubbed *computational thinking*, ought to be perceived as a basic skill, like reading, writing, and arithmetic (Wing, 2006). According to Wing, we engage in computational thinking each day. For instance, acquiring ingredients from a grocery store in preparation to cook a particular meal or, as Wing (2006) notes specifically, a daughter putting items in her backpack in preparation for school exemplifies engagement in *prefetching* and *caching*. In such light, planning and preparation are elements of computational thinking and are useful in everyday life. In 2006, Wing did not provide a definition of computational thinking, but she characterized it this way (p. 33): "Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science." In addition, Wing (2006) held that computational thinking, or thinking like a computer scientist, includes a variety of mental tools, as noted below.

- Thinking recursively
- Parallel processing
- Type checking
- Assessing resource constraints
- Using abstraction and decomposition
- Planning, learning, and scheduling
- Mathematical thinking
- Engineering thinking
- Problem solving
- Heuristic reasoning
- Conceptualizing beyond application of computer programming skills
- "[Computational thinking] is interpreting code as data and data as code" (p. 33).

DOI: 10.4324/9781003172284-6

Those initial notions have been adapted by many teachers, curriculum developers, educational technologists, and computer scientists. Wing's own adaptations are revealing. In 2008, Wing provided some elaborations on the notions above. In particular, Wing (2008, p. 3717) called *abstraction* the "essence of computational thinking." Specifically, according to Wing (2008, p. 3718): "An algorithm is an abstraction of a step-by-step procedure for taking input and producing some desired output." In 2008, Wing continued to emphasize the importance of computational thinking for everyone directly and indirectly. Further, Wing (2008) remarked about the need to reflect continually on teaching and learning strategies with respect to development of computational thinking. Wing (2008) envisioned computational thinking as an integral part of childhood education. By 2014, Wing (after giving thanks to Al Aho, Jan Cuny, and Larry Snyder) was willing to offer this definition: "Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer – human or machine – can effectively carry out." Elaborating on that definition, Wing (2014) noted that people can develop computational thinking without a computer (e.g., see Bell & Lodi, 2019). The key for Wing is developing language capability sufficient to express solutions to people or machines. Also, with respect to the definition, Wing (2014) sought to convey that computational thinking is not strictly about problem solving, but also involves problem formulation, such that the problem becomes subject to a computational solution.

Wing's (2006, 2008, 2010, 2014) papers on computational thinking launched a movement, which has built upon her foundational ideas in order to advance computational thinking by students in schools throughout the world (Gallup & Google, 2016; Vegas & Fowler, 2020; Wing, 2016). This global movement has resulted in the production of numerous books and papers; the delivery of several conference presentations; the development of new K-12 Computational Thinking (CT) and K-12 Computer Science (CS) curricula, lessons, and resources; and the delivery of many workshops for teacher professional development.

One might begin to explore K-12 curricula and resources for CT/CS at ISTE Standards for Students (www.iste.org/standards/for-students) or ISTE CT Competencies for Educators (www.iste.org/standards/computational-thinking); the K-12 Computer Science Framework (k12cs.org); or the Computer Science Teachers Association (csteachers.org). Alternatively, one might start exploring CT and CS at Code.org, which promotes computer science instruction in all schools and provides a catalog of courses, lessons, and tutorials (https://studio.code.org/courses). Another starting point for exploring K-12 CT/CS resources would be the Hour of Code website (hourofcode.com), which is a Code.org project. Alternatively, one could begin exploration into CT and CS in schools through reviews of implementation models, such as those found in Heintz et al. (2016) or Webb et al. (2019). Yet another alternative starting point would be the series of lessons on CS and CT provided by the BBC (formally, by Royal Charter, the British Broadcasting Corporation) at https://www.bbc.co.uk/bitesize/subjects/zvc9q6f. Based on my analyses of those sources on computational thinking, the following two themes emerged.

1. Computational thinking involves problem formulation through decomposition and abstraction (i.e., break complex tasks into subtasks; discern key information in order to devise input-processing-output models that solve the subtasks)
2. Computational thinking involves iterative development of algorithms that meet problem specifications (i.e., create and debug instructions to solve the subtasks, which leads directly to problem resolution)

One must draw on a variety of conceptual knowledge and skills in order to engage in those two activities. Brennan and Resnick (2012) capture particular concepts and practices within their three dimensions of computational thinking, which they labelled *computational concepts, computational practices*, and *computational perspectives*. For Brennan and Resnick (2012), computational concepts include sequences, loops, parallelism, events, conditionals, operators, and data. They demonstrate those concepts using examples of *Scratch* code (Resnick et al., 2009) but note that the concepts can be transferred to other programming languages and to non-programming contexts. With respect to practices involved in computational thinking, Brennan and Resnick (2012, pp. 7–9) identify the following: Being incremental and iterative; Testing and debugging; Reusing and remixing; and Abstracting and modularizing. With respect to reusing and remixing, Brennan and Resnick (2012, p. 8) regard building on prior work as "a longstanding practice in programming" and note that a key function of the online *Scratch* community is to support reuse and remixing of extant *Scratch* projects. Regarding computational perspectives, Brennan and Resnick (2012) describe computational thinking as a form of expression. Further, engagement in computational thinking often leads to face-to-face and online social connections, as well as questioning the implications of technology in the world.

Other researchers, including Corradini et al. (2017) and Duncan et al. (2017), have also analyzed elements of computational thinking. In definitions and conceptions of computational thinking, they also found multiple concepts and practices consistent with the characterizations by Wing (2006, 2014) and Brennan and Resnick (2012), including problem formulation, abstraction, decomposition, pattern recognition, and application of logic in algorithm development. Though Brennan and Resnick placed elements of computational thinking into three categories, and Corradini et al. (2017) placed those elements into four categories (i.e., *mental processes, methods, practices*, and *transversal skills*), there is much overlap in their frameworks. Regarding substantive difference, only the Corradini et al. framework makes explicit mention of perseverance and tolerance for ambiguity.

A review by Zhang and Nouri (2019) of 55 empirical studies of students around the world who engaged in program design and development using the visual drag-and-drop interface in *Scratch* found that students acquired all of the computational thinking concepts, practices, and perspectives in the Brennan and Resnick (2012) framework. Zhang and Nouri (2019) also found that students acquire concepts of input and output and gain perspectives on user interactions, which may also be regarded as human-computer interaction or user experience design. Beyond the skills identified in the Brennan and Resnick (2012) framework, Zhang and Nouri (2019) found that students gained predictive thinking skills and multimedia design skills, as well as learned how to read, interpret, and communicate code. In related work (Nouri et al., 2019), interviews of 19 Swedish teachers of students in grades one through nine learning about programming, further supported the Brennan and Resnick (2012) framework and described student acquisition of general cognitive skills for logical thinking and abstraction, as well as attitudes favoring accuracy. According to the teachers, students also gained language skills related to coding and increased their ability to explain how code works. Further, the teachers perceived students as gaining collaborative skills and problem-solving skills related to remixing code, as well as increasing their capacity for creativity.

It is important to note that some claims about the universal benefits of computational thinking, such as beliefs in gains in general cognitive processing due to problem solving with computers, are disputed (Denning, 2017; Nardelli, 2019). Some proponents of alternative views object to the ambiguity of the expression *computational thinking*. Accordingly, they seek to clarify it (Aho, 2011; Corradini et al., 2017; Denning, 2017) or replace it with another term, such as *informatics* (Nardelli, 2019). To some extent, cultural differences account for linguistic preferences in style and word choice, but word choice also carries important curricular implications. For discussion of both "CS for All" and "Informatics for All," see Casperen et al. (2019). For a comprehensive treatment of the "Informatics for All" initiative, see Casperen et al. (2018) and Forlizzi et al. (2018).

Multiple books also explore definitions and conceptions of computational thinking, as well as provide insights into research on computational thinking. For example, Bers (2021) discusses how to introduce students in early childhood to coding. Williams (2017) presents strategies for teaching coding and computational thinking in elementary school. Beecher's (2017) comprehensive treatment of computational thinking moves from theory to practice and provides a Python tutorial for beginners. Denning and Tedre (2019) approach computational thinking from two perspectives, which they call *designing* and *explaining*. The design perspective proceeds from the engineering tradition and involves crafting computations to accomplish tasks on a computer. The explanation perspective draws on the science perspective to understand computation and how computers affect people. This is reminiscent of the questioning perspective of Brennan and Resnick (2012). Denning and Tedre (2019) regard computational thinking as much more than a set of programming concepts and treat the subject from these six dimensions: methods, machines, software engineering, design, computational science, and computing education. Wang (2016) offers a deep technical treatment of computers and computer processing in order to inspire computational thinking. Edited volumes by Rich and Hodges (2017) and Kong and Abelson (2019) convey the importance of disciplined inquiries into computational thinking across the continents and provide results of research and practice in teaching computational thinking to learners in diverse cultures. Further, these works consider computational thinking with respect to issues pertaining to equity, gender, underrepresentation of girls in computing, and impact on quality of workforce.

4.1 Problem Formulation and Problem Solving

As discussed in the previous section, computational thinking involves problem formulation and problem solving. Problem formulation is a matter of problem specification through decomposition and abstraction. Problem solving involves development of an algorithm to meet the problem specification. Much has been written about software design and development, and one does well to consider lessons learned by software engineers and architects (Chemuturi, 2018; Felleisen et al., 2018; Hanson & Sussman, 2021; Kleppmann, 2017; Petre & van der Hoek, 2016; Richards & Ford, 2020). We will incorporate those lessons learned as we proceed through the general problem-solving strategy advanced by Polya (1945, 2004), which is comprised of the following steps.

1. Understand the problem
2. Devise a plan
3. Carry out the plan
4. Look back

Consider this software design scenario: A group of teachers would like to engage teenagers in an activity to help them learn about heads of state, such as presidents and prime ministers. If you fancy different content, feel free to replace that subject with a different group of people or an entity. For example, you may know a group of teachers who would like teenagers to learn about musicians, sculptors, actors, movie directors, poets, authors, chefs, mathematicians, astronomers, or astronauts. Alternatively, you may prefer to replace heads of state with chemical elements, planets, polyhedra, types of hats, or book titles, for instance. To help teenagers learn about heads of state (or whatever subject you choose), we will create a game that presents clues about the identity of the head of state. The sooner the student identifies the head of state, the more points earned. In the case of heads of state, we will present clues in the form of birthdate, birthplace, margin of electoral victory, inauguration date, and years in office. In a game format, teachers may encourage students to improve their score over time or teachers may have students compete against classmates.

To address Polya's first step, we will begin with the initial conception of the problem scenario, as described in the previous paragraph, which calls for developing a game to help teenagers learn about heads of state, and develop a more precise understanding of the problem. To envision a solution or solutions to this problem, we need to know about the capabilities of the learners, including language ability. Further, we need to know how to access the data pertaining to heads of state. In addition, for any digital solution, we need to be clear on the technology available to the target learners.

With respect to the capabilities of the learners, we will solve this problem for teenagers who comprehend English at Grade 6 level or higher. Further, we will proceed with the view that the learners are adept at handling index cards and responding to prompts on a screen. For solutions that do not involve any computing device, the clues about heads of state could be written on index cards and a partner could question the other and tally points. A solution for individual game play could instruct the player to pass a cardboard sheet over each index card. Instructions to the player would direct the player to cover the entire card to start, then slide down the card to the first horizontal line. Once at that line, the first clue (i.e., the head of state's birthdate) would have been revealed. If the player answers correctly (the head of state's name would be on the reverse side of the index card), the player's score would increase by 50 points, for instance. If incorrect, the player's score remains unchanged and play proceeds to the next card. Otherwise, the player reveals the second clue by sliding the cardboard cover to the next horizontal line. Once again, the player may respond or proceed to the next clue. After the first clue, which is worth 50 points, 10 points could be deducted for each clue revealed. After five clues, the player either earns 10 points by answering correctly or does not score any points for that card. Either way, play would continue with the next card. Certainly, one may devise variations to that game, which you are free to do.

To fathom digital solutions, let's stipulate that the target learners will have a device running a web browser. Further, data about each head of state will be formatted as a semicolon delimited string containing the head of state's name, birthdate, birthplace, margin of electoral victory, inauguration date, and years in office, as exemplified below.

```
Jimmy Carter; October 1, 1924; Plains, Georgia; 297/537 (53.3%); January 20,
1977; 4
```

To devise a plan that solves this problem, note that game play involves two fundamental activities: (1) Presentation of a clue; and (2) Evaluation of the user's response. Those two activities need to be repeated until no clues remain or the user enters the correct answer. The general plan for solving this game development problem is presented in pseudocode below and in Figure 4.1. Those representations of the plan recognize rounds of play. During each round of play, the program will present the clues for one head of state, one clue at a time. After presenting each clue, the learner may enter the head of state's name or just press the Enter/Return key to view the next clue.

```
Initialize data and total score

While another round to play
        While another clue to display
                Display clue
                Enable learner input

                Get learner response

                If response is correct
                        Update total score
                        Display total score
                        Start next round
                Else
                        Present next clue
```

The general plan for solving the game development problem provides a broad overview of the method to be implemented. Importantly, the plan has decomposed the problem into subtasks. In particular, the game involves rounds of play; within each round, clues are presented, and the learner's input is retrieved and processed.

The general plan is a decent starting place, but we need to refine it before moving to implementation. For a digital solution, we need to consider elements of software design. Development of nontrivial programs usually involves consideration of structured data. This case is no exception. We need two lists and hence will use two structured variables (arrays): (1) `dataStrings` will contain the strings of data about the heads of state; and (2) `dataFields` will contain one

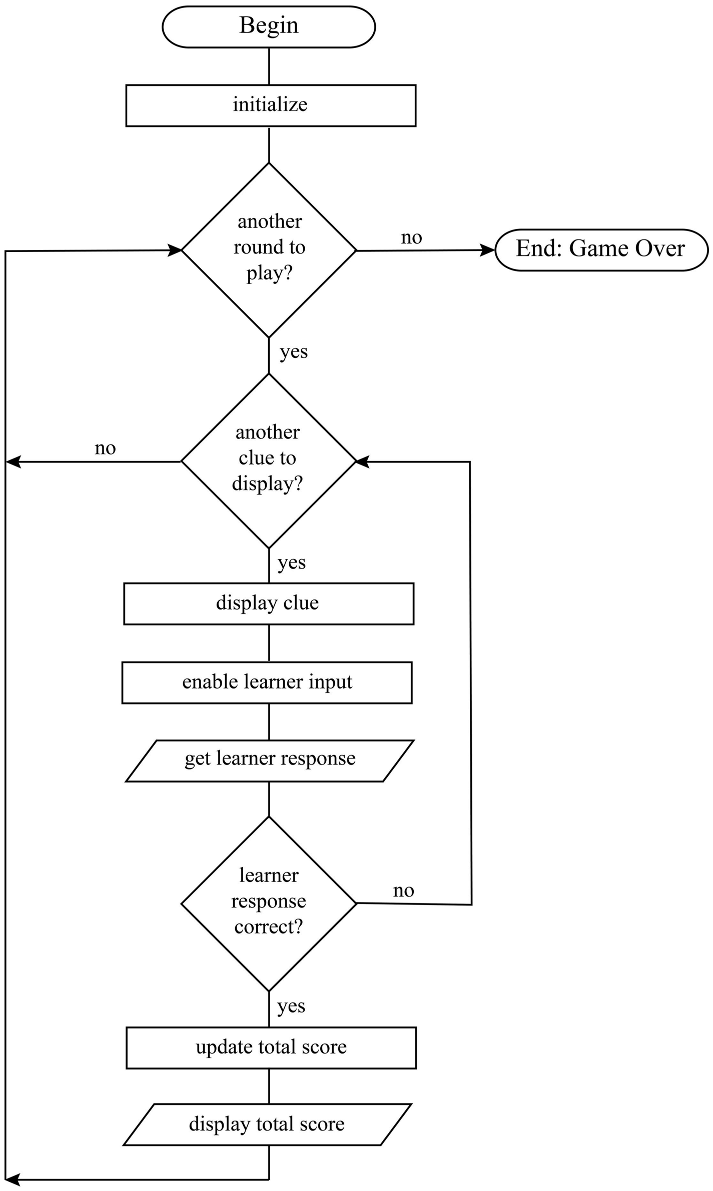


Figure 4.1 Flowchart of steps in game to name heads of state

head of state's name, birthdate, birthplace, margin of electoral victory, inauguration date, and years in office. The `dataFields` variable will be updated each round of play. In addition to the structured variables, we will use three integer variables, namely `totalScore`, `clueNumber`, and `indexDataStrings`. As we prepare to refine the pseudocode above, notice the benefit of iteration. The first line of pseudocode above initializes two variables, but three variables actually need to be initialized. This became evident with declaration of the structured variable, `dataStrings`. As a structured variable, it stores more than one entity and consequently needs to be indexed. This realization has given rise to the variable `indexDataStrings`. The pseudocode below refines the first version of pseudocode. In the refined version below, notice that `dataStrings`, `totalScore`, and `indexDataStrings` are initialized. Also, in the pseudocode below, notice that reflection on the general plan and the two structured variables has enabled precise specifications of "another round to play" and "another clue to display."

```
dataStrings = ["name;birthdate;birthplace;victoryMargin;inaugDate;years
InOffice", "...", "..."]
  indexDataStrings = 0
  totalScore = 0
  While indexDataStrings < length(dataStrings)
      dataFields = dataStrings[indexDataStrings].split-on-semicolon
      clueNumber = 0
      While clueNumber < length(dataFields)
            Display clue and empty text field

            Get user response

            If user response is correct
                  totalScore += 50 – 10 * (clue number – 1)
                  print(totalScore)
                  Start next round
            Else
                  Present next clue
```

We could further refine the pseudocode above, for implementation in a particular programming language, but its current state serves well to inform implementation in any one of multiple programming languages. In Section 4.2.4, we will draw on the refined pseudocode above to produce JavaScript, which will enable game play in web browsers, thus meeting the requirements of the scenario. You might find a couple of surprises as we convert the pseudocode above to JavaScript. In any case, implementing the plan will fulfill Polya's third step. Then, also in Section 4.2.4, we will complete Polya's final step by looking back at this process of problem formulation and problem solving. Reflecting on the JavaScript implementation will reveal advantages and disadvantages of various programming languages and development environments. Sometimes, the similarities between pseudocode and real code are quite striking.

4.2 Coding Examples

Computer programming languages have been categorized for multiple purposes. One common categorization distinguishes between *low-level* and *high-level* languages. Low-level languages provide instructions for a particular Central Processing Unit (CPU) and may be referred to as machine code or assembly languages. Those languages include instructions for manipulating registers in the CPU and addressing memory directly, which are necessary in applications that must read and respond to data received at microsecond intervals. In contrast, high-level languages relieve the programmer from consideration of CPU architecture. In this section we consider statements and programs in the high-level programming language JavaScript. When we run the code, software called an interpreter will rapidly convert the source code to machine code and the program will run on the CPU in your computer, provided the syntax of your source code is correct.

Different types of high-level programming languages have been developed to foster multiple forms of problem solving. There are, for instance, *functional, logic, object-oriented,* and *imperative* programming languages (Louden & Lambert, 2012). Those types of programming languages enable problem solving in particular programming paradigms. Some programming languages, such as JavaScript, Python, and Scheme (a dialect of LISP), enable development of programs across multiple paradigms. We consider JavaScript here because it is ubiquitous (bundled in web browsers) and has imperative programming features, which enable structured or procedural programming. Consequently, we can develop programs to solve problems by decomposing them into subtasks, and then implementing the subtasks in JavaScript functions. In JavaScript, and other programming languages with imperative features, one can solve a wide variety of problems using three types of statements (assignment, conditional, and iterative). If new to programming, running the sample JavaScript code below in a web browser will enable you to gain insights into programming. The JavaScript code and example programs in the upcoming three sections will prepare you to implement the *Heads of State* game in accordance with the plan developed in Section 4.1 and guidance provided in Section 4.2.4. In Section 4.2.5, a new programming language and development environment are introduced. Section 4.2.5 provides opportunities to practice coding in Python and prepares you to implement the *Heads of State* game in Python. The activities in those overviews of computer programming introduce key concepts and techniques, which will help you form conceptions of the fundamental building blocks of languages that enable imperative programming, such as C, Perl, Lua, Pascal, Python, and JavaScript. In addition, the activities will enable you to gain coding skills.

Prior to your forays or excursions into JavaScript, consider the inputs and outputs of computer programs, as well as their functionality. A messaging app, for instance, repeatedly sends, receives, and displays text and images, as well as plays audio and videos. A web browser repeatedly retrieves numbers, text, images, sounds, and video from a web server on the Internet and displays and plays the information received. A word processing app repeatedly accepts keyboard input and displays the characters typed in real time. Once a line has been filled with text, the software automatically moves the insertion point to the next line, along with any characters in the word that could not fit on the previous line. Also consider the app at electrical utility companies that repeatedly performs the same set of calculations to send bills to customers. For each customer, usage is multiplied by cost per kilowatt hour, which varies for residential, commercial, and industrial customers. The usage cost is added to any previous balance, positive or negative; the resulting balance is displayed on the invoice, along with customer name, address, and account number. Lastly, the bill may be printed or sent by email, for instance. Upon consideration of those apps, the following themes are evident.

1. Apps process a variety of data, which human beings perceive as numbers, text, images, audio, and video
2. Apps obtain data from multiple sources (e.g., people, files, databases, web servers)
3. Apps output data, which may be viewed or heard by people in real time or after a delay (e.g., printed invoices), or data may be stored in a file or sent to another computer
4. Apps often perform the same series of steps repeatedly
5. Apps adjust to different inputs (e.g., use different rates for different types of customers)

Those themes have implications for app developers. Given the first theme, variables in computer programs store different types of data, which may be numeric, character, Boolean (true or false), or binary. Data in a binary variable might be an image, sound, or video. Further, unstructured variables store a single value whereas structured variables store multiple values. Given Themes 2 and 3, app developers must become aware of multiple Input/Output (I/O) methods. Further, in light of Item 4 above, app developers must learn how to repeat a set of statements (i.e., implement loops). Given Theme 5, an app developer must also learn how to branch to sets of statements (i.e., implement a conditional statement, typically an if statement). Though not readily apparent in the five themes, it is also important for novice app developers to recognize that programmers decompose complex data processing tasks into manageable subtasks and code the subtasks in *subroutines*, which are also known as *functions, procedures,* and *methods.*

With awareness of those implications for learning about imperative computer programming, you can build your mental model of programming around concepts of **variables, data types, I/O,** and **subroutines**, as well as data processing techniques involving **assignment statements, if statements, and loops**. Now enhance your knowledge and mental model of programming though practice. Fortunately, a tool for practicing programming in JavaScript is readily available to everyone with a contemporary web browser (e.g., Chrome, Firefox, Opera, Safari, Edge). That tool is called the *JavaScript Console*, which enables the entry and execution of JavaScript statements and programs.

4.2.1 Initial Forays into JavaScript

Open a web browser and display the JavaScript console by pressing one of the following key combinations or menu options.

- Google Chrome: Option-Command-J (on MacOS) or Ctrl-Shift-J (on Windows); alternatively, through the View menu, select Developer; JavaScript Console
- Mozilla Firefox: Option-Command-I (on MacOS) or Ctrl-Shift-I (on Windows); alternatively, through the Tools menu, select Browser Tools; Web Developer Tools
- Opera: Option-Command-I (on MacOS) or Ctrl-Shift-I (on Windows); alternatively, through the Developer menu, select Developer Tools
- Microsoft Edge: F12 key; alternatively, through the Edge Browser Menu (· · ·), select Developer Tools
- Microsoft Internet Explorer: F12 key; alternatively, through the IE Browser Menu (Gear icon), select Developer Tools

Each JavaScript expression or statement is entered at the prompt, which is the > sign, and executed by pressing the Enter/Return key.

In the JavaScript console in your web browser, enter and run each of the following mathematical expressions.

```
3 + 2
3 – 2
4 * 3 – 2
4 * (3 – 2)
–2 / 3
```

When you may wish to clear the console, press Ctrl-L (in Windows or MacOS); Command-K also works on a Mac; or click the Clear Console icon in the toolbar. The Clear Console icon is a circle with a diagonal line through it. You may hover your mouse cursor over each toolbar icon to view the tool name. Also notice the various tabs. Ensure that the Console tab is selected. At times, you may be surprised that a different tab has been activated.

Your web browser may evaluate and display the result of a mathematical expression even before you press the Enter/Return key. If you wish, disable Eager Evaluation in the Console Settings.

Now, continue your explorations into JavaScript by entering the following assignment statement, which assigns the value `4` to the variable `a`.

```
a = 4;
```

Now that the variable `a` has been defined, it can be used in expressions. Try each of the following, for instance.

```
a
3 + a
a + 3
1 − a − a
```

Tip: When typing a letter in the JavaScript console, you may need to press the Escape key to dismiss the name completion pop up.

With the variable `a` set to 4 above, the four expressions above yield 4, 7, 7, and -7 respectively. Now enter the following expression, which will generate an error message.

```
A + 4
```

Since JavaScript variable names are case sensitive, the attempt above to add 4 to the undefined variable `A` generates a Reference Error. Retry that expression after setting `A` to a value, as in the following example.

```
A = −3;
A + 4
A + a
```

Now the expression `A + 4` is valid and the result was calculated and displayed. The expression `A + a` is also valid, but it is often better to select variables names that contain more than one letter. For example, enter the following two assignment statements.

```
Celsius = 5;
Fahrenheit = 9 / 5 * Celsius + 32;
```

The second statement reveals that 41 degrees Fahrenheit is equal to 5 degrees Celsius.

In general, statements in JavaScript programs are not entered one at a time. To enter a program with two or more lines of code into the JavaScript console, hold the Shift key down before pressing the Enter/Return key for all but the last line. Use that technique to enter the two lines of code below as one program.

```
Fahrenheit = 50;
Celsius = 5 / 9 * (Fahrenheit −32);
```

When entered as a single program, the JavaScript console displays the last result calculated, which in the case above is 10, the Celsius equivalent of 50 degrees Fahrenheit.

To this point, we have written and entered six assignment statements in JavaScript. All six of those statements begin with a variable name followed by the equal sign `(=)` and an expression. That pattern conforms to the rules defining a syntactically correct **assignment statement** in JavaScript. The equal sign is the assignment operator, which conveys that the variable on the left side of the equal sign is assigned (or becomes) the result of the expression on the right side of the equal sign. Sometimes, the expression on the right is one number, as in the following four assignment statements.

```
a = 4;
A = −3;
Celsius = 5;
Celsius = 10;
```

Sometimes, though, variables contain strings of characters. Enter the following assignment statement into the JavaScript console.

```
greeting = "Hello, World!";
```

Now, one at a time, enter the following three lines of code.

```
greeting = "Hello, ";
person = "John";
greeting + person
```

When applied to variables that contain character strings, as in `greeting + person`, the plus sign concatenates (joins) the strings. Enter the following statements as a single program.

```
greeting = "Hi, ";
person = "Jack";
fullGreeting = greeting + person;
```

As mentioned above, the JavaScript console displays the final result calculated, which in the case above displays `"Hi, Jack"`. However, most tools for computer programming, namely interpreters and compilers, do not display any results unless explicitly directed to do so in the code. In JavaScript, explicit output to the console is generated using the `console.log` function, as in the following example.

```
message = "Rise Up!";
console.log(message);
```

The JavaScript console is implemented such that `undefined` appears after execution of `console.log.` Technically, in JavaScript, a console is an object with a function called log, which returns a null result, and since the JavaScript console displays the last result, `undefined` appears. Simply ignore it.

4.2.2 Branching and Looping

In the previous section, we considered numeric and string data types; we also created some assignment statements and used the `console.log` function to generate output. In this section, we consider `if` statements and loops.

The code below adds one `if` statement to the Celsius conversion program above. Each `if` statement contains a Boolean expression, which evaluates to either `true` or `false`. The condition in the `if` statement below is `Fahrenheit < 0`, which must be enclosed in parentheses in JavaScript. After executing the second of the two assignment statements below, the variable `Fahrenheit` holds the number that is equivalent to −20° Celsius. If that number is less than zero, a comment is displayed and the program terminates; otherwise, no additional lines of code remain in the program, so the program terminates. Enter the six lines of code below as one program and notice the output.

```
Celsius = −20;
Fahrenheit = 9 / 5 * Celsius + 32;
if (Fahrenheit < 0) {
     outputString = Fahrenheit.toString() + "° Fahrenheit is very cold";
     console.log(outputString);
}
```

Since the variable `Fahrenheit` became −4 at the second line of code, the condition in the `if` statement evaluated to `true`. Hence, the *true task*, which in this case consists of the following two lines, was executed.

```
outputString = Fahrenheit.toString() + "° Fahrenheit is very cold";
console.log(outputString);
```

In the assignment statement above, `Fahrenheit.toString()` converts the value of the variable `Fahrenheit`, which is the number –4, to the string `"−4"`. The string `"−4"` is thsen joined to `"° Fahrenheit is very cold"` in order to set the variable `outputString` to `"−4° Fahrenheit is very cold"`. Lastly, in the true task, `console.log(outputString)` displays the output below.

```
4° Fahrenheit is very cold
```

To run the Celsius conversion program again, you can copy and paste code into the JavaScript console, or you can press the Up arrow key (one or more times) to re-run prior code. In this case, with one press of the Up arrow key in the JavaScript console, the Celsius conversion program above is displayed again; then you can click the cursor over the -20 and replace it with -15. The Left and Right arrow keys can also be used to position the cursor at -20. Run the Celsius conversion program with Celsius set to -15. If you wish, run the program with some other values for Celsius (e.g., −25, 0, 100).

Since no output is generated by the Celsius conversion program when the Fahrenheit temperature is greater than or equal to zero, the user would never become aware of the Fahrenheit temperature that is equivalent to Celsius temperatures greater than -17.778°. That is a serious flaw. Here's the remedy. Modify the `if` statement, as shown below, to include an `else` part (called the *false task*), which will run when the condition (`Fahrenheit < 0`) is false. The false task includes one statement, a call to the `console.log` function to ensure that the Fahrenheit temperature is also displayed when the Fahrenheit temperature is greater than or equal to zero.

```
Celsius = −15;
Fahrenheit = 9 / 5 * Celsius + 32;
if (Fahrenheit < 0) {
     outputString = Fahrenheit.toString() + "° Fahrenheit is very cold";
     console.log(outputString);
}
```

```
else {
     console.log(Fahrenheit);
}
```

That version of the app does at least provide some feedback, but a program that describes the temperature for multiple temperature ranges would be more robust. Enter and run the following version of the Celsius conversion program. If you wish, you can copy the code from the *celsiusConversionComment.txt* file in the Electronic Resources for this book and paste the code into the JavaScript console.

```
Celsius = -15;
Fahrenheit = 9 / 5 * Celsius + 32;
if (Fahrenheit < 0) {
     outputString = Fahrenheit.toString() + "° Fahrenheit is very cold";
}
else if (Fahrenheit < 30) {
     outputString = Fahrenheit.toString() + "° Fahrenheit is cold";
}
else if (Fahrenheit < 60) {
     outputString = Fahrenheit.toString() + "° Fahrenheit is cool";
}
else if (Fahrenheit < 75) {
     outputString = Fahrenheit.toString() + "° Fahrenheit is warm";
}
else if (Fahrenheit < 90) {
     outputString = Fahrenheit.toString() + "° Fahrenheit is hot";
}
else {
     outputString = Fahrenheit.toString() + "° Fahrenheit is very hot";
}
console.log(outputString);
```

Use of the if `. . . else if . . . else` structure in the code above has enabled the inclusion of more descriptors, which was the goal. However, the utility of the program is still quite limited. After all, the program provides a descriptor for only one temperature. A table that displayed both Celsius and Fahrenheit temperatures from -40C to 40C, along with a suitable descriptor, would be more beneficial to someone learning a new temperature scale. In this case, we need to repeatedly calculate the conversion of Celsius temperatures to Fahrenheit, which we know is implemented by one assignment statement (i.e., `Fahrenheit = 9 / 5 * Celsius + 32;`). Imperative programming languages provide loops to repeat one or more statements. Before writing the temperature conversion loop, enter and run the following loops (as programs, not one line at a time) in the JavaScript console.

```
for (i = 0; i <= 10; i++) {
     console.log(i);
}
```

In JavaScript, the `for` loop begins by setting the initial value of the loop control variable; in the case above, the loop control `i` is set to `0 (i = 0)`. Second, the `for` loop specifies the condition that must be true for the statements in the loop body to be executed. In the case above, as long as the value of `i` is less than or equal to `10 (i <= 10)`, the statement or statements in the *loop body* will be executed. The third part of the `for` loop specifies the value by which the loop control variable changes after executing the loop body. In the loop above, `i++` increments the loop control variable `i` by one, after executing the loop body. The fourth part of the `for` loop is the loop body, which includes the statement or statements within the braces, `{ }`. In the loop above, only the one output statement appears, `console.log(i);`. The output of the loop above demonstrates that the loop control variable is incremented by 1 each time through the loop.

Now run the following loop.

```
for (i = 0; i <= 40; i += 5) {
     console.log(i);
}
```

The loop above demonstrates that `i += 5` increments the loop control variable by 5 each time through the loop, which starts at 0 and runs to 40. Next, per the code below, change the initial value to -40 to display the multiples of 5 from -40 to 40.

```
for (i = −40; i <= 40; i += 5) {
     console.log(i);
}
```

We now have a loop that generates the numbers in the first column of Table 4.1. Let's add the assignment statement for calculating the Fahrenheit temperature and replace `Celsius` with `i`, the loop control variable. Also, add `Fahrenheit` to the list of output values.

```
for (i = −40; i <= 40; i += 5) {
     Fahrenheit = 9 / 5 * i + 32;
     console.log(i, Fahrenheit);
}
```

Table 4.1 Celsius to Fahrenheit Conversion Table

-40	-40	Very cold
-35	-31	Very cold
-30	-22	Very cold
-25	-13	Very cold
-20	-4	Very cold
-15	5	cold
-10	14	cold
-5	23	cold
0	32	cool
5	41	cool
10	50	cool
15	59	cool
20	68	warm
25	77	hot
30	86	hot
35	95	very hot
40	104	very hot

Alternatively, the loop control variable `i` could have been replaced by `Celsius`, thus preserving the original assignment statement, as in the code below.

```
for (Celsius = -40; Celsius <= 40; Celsius += 5) {
     Fahrenheit = 9 / 5 * Celsius + 32;
     console.log(Celsius, Fahrenheit);
}
```

The loop above generates the numbers in the first two columns of Table 4.1. We need only add the descriptor, and then some padding in order to align the numbers properly. To add the descriptor, we will use the prior `if . . . else if . . . else` structure, replacing `outputString` with `descriptor` and retaining only the descriptive words. The following code produces the values in the three columns of Table 4.1. The code below is available in the *celsiusConversionTable1.txt* file in the Electronic Resources for this book.

```
for (Celsius = -40; Celsius <= 40; Celsius += 5)

{
     Fahrenheit = 9 / 5 * Celsius + 32;

     if (Fahrenheit < 0) {
          descriptor = "very cold";
     }
     else if (Fahrenheit < 30) {
          descriptor = "cold";
     }
     else if (Fahrenheit < 60) {
          descriptor = "cool";
     }
     else if (Fahrenheit < 75) {
          descriptor = "warm";
     }
     else if (Fahrenheit < 90) {
          descriptor = "hot";
     }
     else {
          descriptor = "very hot";
     }
     console.log(Celsius, Fahrenheit, descriptor);
}
```

In JavaScript, padding is added to the start of a string using the `padStart` function, which requires two parameters: (1) The length of the string with padding included; and (2) The character to use for the padding. We will pad with a blank space, but sometimes a leading zero or some other character is used. Since the `padStart` function requires a string data type, the numeric variables, `Celsius` and `Fahrenheit`, are converted to strings using the `toString` function. The code below is available in the *celsiusConversionTable2.txt* file in the Electronic Resources for this book.

```
for (Celsius = -40; Celsius <= 40; Celsius += 5) {
     Fahrenheit = 9 / 5 * Celsius + 32;
     if (Fahrenheit < 0) {
          descriptor = "very cold";
     }
     else if (Fahrenheit < 30) {
          descriptor = "cold";
     }
     else if (Fahrenheit < 60) {
          descriptor = "cool";
     }
     else if (Fahrenheit < 75) {
          descriptor = "warm";
     }
     else if (Fahrenheit < 90) {
          descriptor = "hot";
     }
     else {
          descriptor = "very hot";
     }

     paddedCelsius = Celsius.toString().padStart(5, ' ');
     paddedFahrenheit = Fahrenheit.toString().padStart(5, ' ');
     paddedDescriptor = descriptor.padStart(11, ' ');

     console.log(paddedCelsius, paddedFahrenheit, paddedDescriptor);
}
```

While loops

In a `for` loop, the number of times the loop body will execute is determined and fixed before the looping begins. Due to that limitation, the `for` loop is incapable of solving many problems. For example, the characters in a file of text are processed until the end of the file is reached. In JavaScript (and many other programming languages), the `while` loop is used to execute the statements in a loop body until a condition is met. Run the code below to consider the functionality of a `while` loop.

```
n = 20;
while (n < 30) {
     console.log(n);
     n = n + 2;
}
```

Run the following program, which draws random numbers between 0 and 10 and determines whether your lucky number (assumed to be 7) is drawn. Note that the random number function in JavaScript (`Math.random`) generates a pseudorandom number between 0 and 0.99999. Hence, to obtain a random integer between 0 and 10 inclusive, the code below multiplies the random number generated by 10 (`Math.random() * 10`) and then rounds that result to the nearest integer using JavaScript's `Math.round` function.

```
maxDraws = 5;
drawNumber = 0;
luckyNumber = 7;
randomNumber = -1;
```

```
while (randomNumber!= luckyNumber && drawNumber < maxDraws) {
     drawNumber++;
     randomNumber = Math.round(Math.random() * 10);
}
if (randomNumber == luckyNumber) {
     console.log("Today is your lucky day!");
     console.log("Your lucky number was drawn on Draw ", drawNumber);
}
else {
     console.log("Your lucky number was not drawn.");
}
```

To view the random numbers as they are drawn, add the following line as the last statement in the loop body.

```
console.log(randomNumber);
```

The `while loop` above contains two conditions, `randomNumber!= luckyNumber` and `drawNumber < maxDraws`. The `!= symbol in randomNumber!= luckyNumber` is the not equal operator. To test for equality, == is used, as in the `if` statement above. When the random number generator (`Math.random`) produces the number 7, which is the value assigned to the variable, `luckyNumber`, `randomNumber != luckyNumber` evaluates to `false` and the loop is terminated. To avoid the chance of an infinite loop, which would occur if the random number generator never picked the number 7, the variable `drawNumber` counts the number of times a random number is drawn. If `drawNumber` becomes greater than 5, the value of the variable `maxDraws`, the `while` loop terminates. Both conditions, `randomNumber!= luckyNumber and drawNumber < maxDraws`, must be true for the loop body to execute because those two conditions are joined by the symbol &&, which is the `and` operator. When either condition is false, the loop terminates. The or operator is `||`, but use of that operator in the program above would very likely create an infinite loop. Infinite loops do not terminate on their own. An infinite loop executing in a web browser would require the programmer to terminate the web browser to break free of the error (though you could also try various key combinations, Ctrl-C or Command-C, or Ctrl-Z or Command-Z, for instance).

We will need a `while` loop in the next section as well because the number of passes needed to sort a list is not known until the list has been sorted.

4.2.3 Sorting Data in a List

Numerous applications sort a list of numbers or strings. We will focus on numbers here, but the coding algorithm for strings remains the same; the only difference is in the type of data in the list to be sorted (a task below prompts you to verify this). There are numerous algorithms for sorting, which vary in speed and memory requirements. Commonly, the data to be sorted reside in one list, which is often referred to as an *array*. To sort a list, a developer must know how to access the items in the list. In JavaScript, the loop control variable of a `for` loop is often used to access each value in a list. One might imagine that the first value in at list is at location 1; the second value at location 2, the third value at location 3, and so forth. Even though some computer-programming languages are implemented in that manner, JavaScript begins lists at index 0. To access values in a list, the variable name of the list is followed by an index in brackets (e.g., `myFirstList[0]`, `myFirstList[1]`). Enter the following assignment statement into the JavaScript console.

```
myFirstList = [25, –1, 10, –2, 70];
```

Then, one at a time, run the expressions below.

```
myFirstList[0];
myFirstList[1];
myFirstList[2];
myFirstList[3];
myFirstList[4];
myFirstList[5];

myFirstList.length;

n = 2;
myFirstList[n];
myFirstList[n−2];

console.log(myFirstList[0]);
```

Since lists can have thousands of values, it would not be practical to address each value in a separate statement with a constant for the index. Instead, the control variable of a `for` loop is often used to access all of the values in a list, as in the code below. Enter the following code into the JavaScript console as a single program.

```
theList = [5, 2, −2, 17, 10, 25, 1];
for (i = 0; i < theList.length; i++) {
     console.log(theList[i]);
}
```

Now we proceed to sort the list. One method for implementing an ascending sort involves inspecting consecutive pairs of numbers and swapping the numbers if the first number is greater than the second number. Given the initial state of the list, the first pair of numbers contains 5 and 2 respectively. Since 5 is greater than 2, those numbers would be swapped to produce the list below.

```
[2, 5, −2, 17, 10, 25, 1]
```

Now consider the next pair of numbers, the second and third numbers in the list. Given the state of the list above, the second and third numbers in the list are 5 and -2 respectively. Since 5 is greater than -2, those numbers would be swapped to produce the list below.

```
[2, −2, 5, 17, 10, 25, 1]
```

Now consider the next pair of numbers, the third and fourth numbers in the list. Since 5 is less than 17, those numbers would not be swapped. Proceed to the fourth and fifth numbers in the list; since 17 is greater than 10, those numbers would be swapped to produce the list below.

```
[2, −2, 5, 10, 17, 25, 1]
```

The fifth and sixth numbers in the list, 17 and 25, would not be swapped, but the last pair of numbers, 25 and 1, would be swapped to produce the list below.

```
[2, −2, 5, 10, 17, 1, 25]
```

After that one pass through all six pairs of consecutive numbers, the list is not yet sorted, so we would make another pass through the list, repeating the process of comparing consecutive pairs of numbers in the list and swapping them if the first number is larger than the second number. In the case above, the list will still not be sorted after the second pass. If there was even one swap during a pass, the list may not be sorted. Hence, a pass is made through all consecutive pairs of numbers in the list until no swaps are made. The following pseudocode captures that sorting algorithm, which is called the *bubble sort*.

```
while the list is not sorted
      for each consecutive pair of numbers
            if first number > second number
                  swap the numbers
```

To control the outer loop, we need to determine whether or not the list is sorted. Given that the list can only be in one of two possible states, sorted or unsorted, we could use a Boolean variable to keep track of that state. Let's call that variable `sorted`; assume that it is true (i.e., the list is sorted) at the start of each pass through the list and set `sorted` to false when a swap is made. When no swaps are made through an entire pass through the list, `sorted` will remain true and the sorting algorithm will terminate because the list has been sorted. The following code implements the bubble sort in JavaScript, using a numeric variable to count swaps.

```
swapNumber = 1;
while (swapNumber != 0) {
    swapNumber = 0;
    for (i = 0; i < theList.length -1; i++) {
       if (theList[i] > theList[i+1]) {
         temp = theList[i];
         theList[i] = theList[i+1];
         theList[i+1] = temp;

         swapNumber++;
       }
    }
}
```

The outer loop is a `while` loop that stops the sorting process when the list is sorted. The inner loop, the `for` loop, implements a complete pass through all consecutive pairs of numbers. Note that the number swapping process takes three steps: (1) Save the first number in the pair of numbers; (2) Replace the first number with the second number; and (3) Replace the second number with the saved first number. Since the numbers are in consecutive locations in the list, we have access to their list locations using the loop control variable, `i`, and `i + 1`. The JavaScript code below implements the bubble sort algorithm captured in the pseudocode above. Note that in JavaScript, the negation operator is the exclamation mark, `!`. Hence, in JavaScript, ! sorted means not sorted and != means not equal.

To display the initial values of the list and the values in the list after each pass, we need to run the following code, which we wrote earlier.

```
for (i = 0; i < theList.length; i++) {
    console.log(theList[i]);
}
```

However, including the same code in multiple places in a program should be avoided. Code that needs to be reused should be placed into a function and the function called when necessary. Here's the final version of the program to sort numbers in a list. The code below is available in the *bubbleSort.txt* file in the Electronic Resources for this book.

```
function displayList() {
     for (i = 0; i < theList.length; i++) {
          console.log(theList[i]);
     }
}

theList = [2, -2, 5, 10, 17, 1, 25];
console.log("Initial List");
displayList();

swapNumber = 1;
while (swapNumber != 0) {
     swapNumber = 0;
     for (i = 0; i < theList.length -1; i++) {
          if (theList[i] > theList[i+1]) {
               temp = theList[i];
               theList[i] = theList[i+1];
               theList[i+1] = temp;

               swapNumber++;
          }
     }
     console.log("Finished Pass through List");
     displayList();
}
```

After running the code above to verify its functionality, replace the list of numbers with the following list of character strings (or other strings of your choosing, perhaps strings with uppercase and lowercase letters).

```
theList = ["turtle", "dove", "dog", "cat", "fox", "duck"];
```

Make one last change to the bubble sort program. Implement a descending sort by replacing one character in the statement that compares two consecutive numbers in the list. Currently, that statement checks whether the first number is greater than the second number, but in a descending sort, that statement should determine whether the first number (or string) is less than the second number (or string). Then the swap will move smaller numbers (or strings) toward the right end of the list, which will ultimately sort them in descending order.

In the development of two programs, one to covert temperatures from Celsius to Fahrenheit and the other to sort a list of numbers, we utilized the fundamental building blocks of imperative programming languages. We used numeric, character, and Boolean variables. We used unstructured variables, which store a single value, and, in the second program, we used a structured variable to store and process multiple values. In the first program, we manipulated the data in variables in order to produce a conversion table of Celsius and Fahrenheit temperatures. In the second program, our data manipulations sorted a list of numbers. In the two programs, we implemented assignment statements, `if` statements, and loops in order to process the data. Further, in the bubble sort program, we decomposed the problem into subtasks, addressing all aspects of the sorting problem before considering the output problem. We solved the sorting problem by devising a plan to sort numbers; articulating the plan in code; and then implementing it in JavaScript. We solved the output problem by determining when to display the values in the list. Further, we realized that we could reuse code developed earlier for displaying numbers in a list. Moreover, consistent with decomposition, we implemented the output function in a subroutine. This modular approach ensured that the program would contain no redundant code and added flexibility, which enabled us to output the list at any point in the program. Further development of those fundamentals would be helpful to novices. In addition, more exploration into forms of input should be considered. We used inputs generated through the code (i.e., the multiples of five for the Celsius temperatures) or declared in the code (i.e., the list of numbers to be sorted). Of course, users are an important source of input. The next two sections consider user input in two different computing contexts, specifically web apps and desktop/laptop microcomputers.

4.2.4 JavaScript Implementation of the Heads of State Game

In Section 4.1, we developed a plan to implement the Head of State game. The specifications require implementation of the game as a web app. Given that requirement (which is relaxed in the Python implementation below), we will embed JavaScript code into a web page. To begin, enter the following HTML and JavaScript into a text editor. If unfamiliar with HTML, first proceed through Section 3.3.6. Consistent with the plan established in Section 4.1, the solution begins with initialization. Notice the inclusion of the `onload` attribute in the body tag (`<body onload="init()">`). Since that attribute is set to `init()`, the `init` function in the JavaScript program will run when the web page is opened in a web browser. The `init` function initializes the `dataStrings` variable to the strings of information about each of the heads of state. The `indexDataStrings` and `totalScore` variables have already been set in the variable declarations. Lastly, the init function calls the `startRound` function. Notice that the program currently has three empty functions, namely `startRound`, `presentClue`, and `getLearnerResponse`, which will be implemented next. Consistent with the plan in Section 4.1, those three functions will implement game play.

```
<html>
<head>
    <title>Heads of State Game</title>

<script type="text/javascript">
    var dataStrings = [];
    var indexDataStrings = -1;
    var dataFields = [];
    var clueNumber;
    var totalScore = 0;
    var gameSpaceString;


    function init() {
        dataStrings = ["Barack Obama; August 4, 1961; Honolulu, Hawaii;
        365/538 (67.8%); January 20, 2009; 8", "Ronald Reagan; February
        6, 1911; Tampico, Illinois; 489/537 (91.0%); January 20, 1981; 8",
        "Jimmy Carter; October 1, 1924; Plains, Georgia; 297/537 (53.3%);
```

```
          January 20, 1977; 4", "George W Bush; July 6, 1946; New Haven, CT;
          271/537 (50.5%); January 20, 2001; 8"];

          startRound();
     }
     function startRound() {
     }

     function presentClue() {
     }

     function getLearnerResponse(e) {
     }

</script>

</head>

<body onload="init()">

<h3>Identify the Heads of State</h3>
<div id="scoreBoard"><p>Score: 0</p></div>
<div id="gameSpace"> </div>

</body>
</html>
```

The `startRound` function keeps track of the index to the data strings by incrementing the `indexDataStrings` variable by one each time the function is called. That index is used to retrieve the string of information, from the variable `dataStrings`, pertaining to the head of state currently under consideration. Consistent with the outer loop depicted in Figure 4.1 and the pseudocode in Section 4.1, the `if` statement in the `startRound` function controls whether the game proceeds with data for the current head of state or the game ends because all rounds of play have been completed. If there is another round to play, the `dataFields` array is set to the data for the head of state currently under consideration by splitting the current string of data. In addition, `clueNumber` is set to zero; `gameSpaceString` is set to the empty string; and the `presentClue` function is called. When there are no more rounds to play, the program displays the final score and the program terminates.

In the `presentClue` function, the `clueNumber` variable, which is used as the index to the `dataFields` array, is incremented by one. Then, if there are more clues to display, the current clue is displayed along with an input field for the learner's response. The input field has an onKeyPress attribute (`onKeyPress="getLearnerResponse(event)"`), which is used to identify the function to be called when the learner enters the name of a head of state or presses only the Enter/Return key in order to proceed with game play. The input field also has an id attribute (`id="learnerResponse"`), which is accessed in the `getLearnerResponse` function to retrieve the learner's response. When there are no more clues to display in the current round, the function calls `startRound()` in order to begin the next round of play.

The `getLearnerResponse` function reads the learner's input and controls the inner loop depicted in Figure 4.1 and in the pseudocode of the plan. When a JavaScript program runs in a web browser, user input is obtained through an input field in the web page. The input field sends “key pressed” events as the user types characters into the field. The Enter/Return key is recognized by its key code, which is 13. The `getLearnerResponse` function ignores all keys pressed until key code 13 (the Enter/Return key) is detected. When that key code is detected (`e.keyCode == 13`), the name entered by the learner, if any, is stored in the `learnerResponse` variable (`learnerResponse = document.getElementById("learnerResponse").value`). The learner’s response is compared to the name of the head of state, which is in the first element of the `dataFields` array (`dataFields[0]`). If the learner’s response matches the name of the head of state, the `totalScore` variable is incremented by the points earned in the round; the score is displayed on the web page; and the `startRound()` function is called to begin the next round of play. On the other hand, if the learner’s response does not match the director’s name, whether the learner entered an incorrect name or pressed only the Enter/Return key in order to get the next clue, the `presentClue` function is called in order to proceed with the next clue in the current round. The complete script for the game, which is embedded in a web page, appears below. The code below is also in the *headsOfStateGame.html* file in the Electronic Resources for this book.

```
<html>
<head>
     <title>Heads of State Game</title>

<script type="text/javascript">
     var dataStrings = [];
     var indexDataStrings = –1;
     var dataFields = [];
     var clueNumber;
     var totalScore = 0;
     var gameSpaceString;

     function init() {
          dataStrings = ["Barack Obama; August 4, 1961; Honolulu, Hawaii;
          365/538 (67.8%); January 20, 2009; 8", "Ronald Reagan; February
          6, 1911; Tampico, Illinois; 489/537 (91.0%); January 20, 1981; 8",
          "Jimmy Carter; October 1, 1924; Plains, Georgia; 297/537 (53.3%);
          January 20, 1977; 4"];

          startRound();
     }

     function startRound() {
          indexDataStrings += 1;
          if (indexDataStrings < dataStrings.length) {
               dataFields = dataStrings[indexDataStrings].split(";");
               gameSpaceString = "";
               clueNumber = 0;
               presentClue();
          }
          else {
               gameRating = totalScore / (dataStrings.length * 50);
               gameRating = Math.round(gameRating * 100);
               document.getElementById("gameSpace").innerHTML = "Game over:
               You earned " + gameRating + "% of the points over the " +
               dataStrings.length + " rounds.";
          }
     }

     function presentClue() {
          clueNumber += 1;
          if (clueNumber < dataFields.length) {
               gameSpaceString += dataFields[clueNumber] + "<br />";
               document.getElementById("gameSpace").innerHTML =
               gameSpaceString + '<input onKeyPress="getLearnerRespon-
               se(event)" id="learnerResponse" type="text" size="40">';
               document.getElementById("learnerResponse").focus();
          }
          else {
               startRound();
          }
     }

     function getLearnerResponse(e) {
          if (e.keyCode == 13) {
               learnerResponse = document.getElementById("learnerResponse").
               value;
               if (learnerResponse == dataFields[0]) {
                  totalScore += 50-10 * (clueNumber -1);
                  document.getElementById("scoreBoard").innerHTML =
                  "<p>Score: " + totalScore + "</p>";
                  startRound();
               }
               else {
                  presentClue();
               }
          }
     }

</script>

</head>

<body onload="init()">

<h3>Identify the Heads of State</h3>
<div id="scoreBoard"><p>Score: 0</p></div>
<div id="gameSpace"> </div>

</body>
</html>
```

Now that we have implemented the game, we can complete Polya's problem-solving process by looking back and reflecting on the solution. The JavaScript solution does implement the plan depicted in Figure 4.1. However, the nested `while` loops that appear in the pseudocode suggest an iterative solution, rather than a recursive one. In web pages, JavaScript code cannot prevent the web page from being rendered and displayed. Hence, the JavaScript solution above calls functions recursively. For example, the `startRound` function calls the `presentClue` function and, when no more clues remain to be presented, the `presentClue` function calls the `startRound` function. In the interim, the `getLearnerResponse` function will have been invoked at least once. For implementation in web pages, this "non-blocking" form of program execution is necessary in order to render the entire web page. In contrast, apps running in environments other than web browsers can block, pause, or suspend execution until the user enters data. In those cases, it is possible to implement an iterative solution very much like the pseudocode. This is demonstrated in Section 4.2.6, which implements a solution to the Heads of State game in an interactive Python programming environment.

The JavaScript solution above also raises questions about access to data beyond the data strings in the code. A more robust solution would access a data file or files that would permit game play to learn about other topics. Again, the programming environment has a profound effect on implementation. In a web environment, a data file must be fetched in a manner that does not block program execution. Such implementation requires multiple lines of code and goes beyond the scope of this section. In contrast, a Python implementation would require only one line of code to read the file, as demonstrated in Section 4.2.6. In preparation for the Python implementation of the Heads of State game, the next section introduces Python programming in an interactive environment.

4.2.5 Interactive Python

Python is a flexible multipurpose programming language. As an open-source project, Python is freely available and very popular. Python can be downloaded from https://www.python.org/ and installed on Windows, MacOS, Linux, and many other platforms (https://www.python.org/download/other/). It is possible to run Python in an interactive mode or submit files to the Python interpreter from either the Windows or MacOS command line console. Alternatively, you may run Python code in an Integrated Development Environment (IDE), such as Microsoft Visual Studio Code or Spyder. Commonly, the Spyder IDE is acquired through Anaconda (https://www.anaconda.com/), which is a popular bundle of software tools for data science and machine learning. The Individual Edition of Anaconda is an open-source distribution, which can be installed on Windows, MacOS, or Linux. To begin working in Python, I recommend trying the Spyder IDE in Anaconda.

By default, the Spyder IDE displays three windows: (1) The text editor is on the left; (2) The variable explorer is on the top right; and (3) The console is on the bottom right. After typing Python code into the text editor, click the Run button (the green triangle) in the toolbar, or Press F5 or drop down the Run menu and select Run. The program runs in the Console window. The values of variables can be viewed in the Variable Explorer window, provided the Variable Explorer tab has been selected. (The Help tab is the default tab for the window in the upper-right portion of the Spyder IDE, so be sure to click the Variable Explorer tab to view the values of variables). At times, you may want to run a single line of code or two or more consecutive lines of code, in which case you can highlight the line or lines in the editor and press the F9 hotkey. (Hotkeys can be changed in Preferences; Keyboard Shortcuts.) Executing a particular line or lines of code can help in the development/debugging process.

Python is a flexible programming language which may be used for functional programming, as well as imperative programing (e.g., variables may contain numeric, character, or Boolean data types; the language includes assignment and `if` statements, as well as loops; and functions support decomposition). Unlike many imperative programming languages, which use braces { } to surround blocks of code, Python uses indentation. For example, type the following two lines of Python code into the editor and run the program.

```
for i in range(0, 10):
     print(i)
```

Also, run the following variations on the `for` loop.

```
for i in range(0, 10):
     print(i, i * 5)

for i in range(50, 0, -5):
     print(i)

for i in range(0, 10):
     print(i)
     print(i * 5)

for i in range(0, 10):
     #print(i)
     #print(i * 5)
     print(i, i * 5)
```

Enter the following lines of code to consider the syntax of the `if` statement in Python. Notice that `else if` is `elif` in Python. Instead of the JavaScript `toString` function, Python uses the `str` function.

```
Celsius = -15
Fahrenheit = 9 / 5 * Celsius + 32
if Fahrenheit < 0:
     outputString = str(Fahrenheit) + "° Fahrenheit is very cold"
     print(outputString)
elif Fahrenheit < 30:
     print(str(Fahrenheit) + "° Fahrenheit is cold")
else:
     print(Fahrenheit)
```

At this point, I encourage you to write a Python program that includes both a `for` loop and an `if` statement to implement the display of a temperature conversion table that mimics the JavaScript version in Section 4.2.2.

After writing a Python program to produce a temperature conversion table, run the code below and notice that a list in Python is both syntactically and functionally similar to JavaScript.

```
theList = [2, -2, 5, 10, 17, 1, 25]
for i in range(0, len(theList)):
     print(theList[i]);
```

Open a new Python file and run the following program. Note that random numbers can only be generated in Python after importing the `random` library. One of the functions in the Python `random` library is `randint`, which generates a random integer in a specified range, as shown below.

```
import random
maxDraws = 5
```

```
drawNumber = 0
luckyNumber = 7
randomNumber = −1
while randomNumber != luckyNumber and drawNumber < maxDraws:
     drawNumber = drawNumber + 1;
     randomNumber = random.randint(0,10);

if randomNumber == luckyNumber:
     print("Today is your lucky day!");
     print("Your lucky number was drawn on Draw ", drawNumber);
else:
     print("Your lucky number was not drawn.");
```

At this point, I encourage you to write a Python program that implements the bubble sort algorithm.

4.2.6 Python Implementation of the Heads of State Game

Unlike JavaScript, Python permits user input directly from the console in which the program is running. Hence, the following Python program implements the Heads of State game, which is strikingly similar to the pseudocode in Section 4.1. The following code is available in the *headsOfStateGame.py* file in the Electronic Resources for this book.

```
dataStrings = ["Barack Obama; August 4, 1961; Honolulu, Hawaii; 365/538
(67.8%); January 20, 2009; 8", "Ronald Reagan; February 6, 1911; Tampico,
Illinois; 489/537 (91.0%); January 20, 1981; 8", "Jimmy Carter; October 1,
1924; Plains, Georgia; 297/537 (53.3%); January 20, 1977; 4"]
indexDataStrings = 0
totalScore = 0
while indexDataStrings < len(dataStrings):
     dataFields = dataStrings[indexDataStrings].split(";")
     clueNumber = 1
     while clueNumber < len(dataFields):
          learnerResponse = input(dataFields[clueNumber] + "\n Name: ")
          if learnerResponse == dataFields[0]:
               totalScore += 50−10 * (clueNumber −1)
               print("Total Score:", totalScore)
               clueNumber = 100
          else:
               clueNumber += 1
     indexDataStrings += 1

gameRating = round(100 * (totalScore / (len(dataStrings) * 50)))
print("Game Over: Your earned", gameRating, "percent of the points over
the", len(dataStrings), "rounds.")
```

Looking back, the Python implementation of the Heads of State game contains fewer lines of code than the web app in JavaScript. This is largely due to the differences in the implementation context: web browser versus console of personal computer. Running code in one's computer console reduces security risks considerably and enables access to multiple resources. For example, using the code below, the input data could be stored in a text file named `headsOfStateData.txt`, with one line of information for each Head of State, and the contents of the file could be read and assigned to the list of data strings (`dataStrings`).

```
inputFile = open("headsOfStateData.txt", "r")
fileContent = inputFile.read()
dataStrings = fileContent.split("\n")
```

Storing the input data in a file and initializing the `dataStrings` list to each line of the file, using the `split` function, as shown above, the Python code never needs to be changed to play the game, even after the input file is edited. By editing the input file using a text editor, the game content could be changed without a computer programmer.

Consider replacing the assignment statement (`dataStrings = ["Barack . . . "]`) in the Heads of State code above with the three lines of Python code above the previous paragraph. A sample *headsOfStateData.txt* file is available in the Electronic Resources for this book, along with the *headsOfStateGame2.py* file, which provides a sample solution of this version of the game.

Lastly, the input file, *headsOfStateData.txt* in this case, could even be moved to a web server to make it available globally. Then the Python script could be sent to users in order to play the game from anywhere in the world. In this scenario, the three lines of code used to access the data file in the previous version would be replaced by the following five lines of Python code to read the file from a web server, provided a complete URL were provided. As exemplified above, the `import` command loads a library of code. In the code below, the `requests` library is used to facilitate HyperText Transfer Protocol (HTTP) requests.

```
import requests

url = 'https://'
webObject = requests.get(url, allow_redirects=True)
fileContent = webObject.text
dataStrings = fileContent.split("\n")
```

4.3 Structured Data Transfers in JavaScript Object Notation (JSON) with REST and APIs

JavaScript Object Notation (JSON) is a popular standard that defines rules for structuring numeric, character, and Boolean data transferred between computers. When Application Programming Interfaces (APIs) and RESTful web services facilitate those data transfers, the data are often formatted in either JSON or XML. We consider XML in Section 4.4, and before proceeding with discussion about JSON in this section, it will be helpful to proceed with common conceptions of APIs and REST.

Just as a *user interface* provides menus, data entry fields, buttons, and other such components, which enable users to interact with a machine to collectively accomplish a goal, an Application Programming Interface (API) facilitates interactions between computers to accomplish a goal (Lauret, 2019). For example, in data exchanges with mobile apps, APIs fetch and transfer weather information, game scores, social media posts, maps, and much more. When the goal involves the transfer of data using a web service, which follows the HyperText Transfer Protocol (HTTP) or its more secure form, HTTPS, developers often establish a RESTful service. REST is the acronym for REpresentational State Transfer, which compels adherence to six guiding principles (https://restfulapi.net/). In this work, we need not dwell on any one of those principles; we need only recognize that a RESTful API operates like a web page request. When a web server receives a URL that uniquely addresses one of its resources, the web server dutifully sends the resource back to the client software that initiated the request (sometimes on behalf of a human being who clicked a hyperlink in a web browser). Similarly, a RESTful API facilitates the flow of data from client to server and back to the client. Data exchanged through a RESTful web service usually requires some formatting work on the part of the client app before presentation to a human being. For example, enter the following URL, which makes a REST request for data, into your web browser.

https://api.publicapis.org/entries

At the time of writing, that REST request returned 940 descriptions of APIs in JSON. The integer for the "count" field, the first field in the response, identifies the number of APIs described in the results you received.

Try one more REST request; enter the following URL into your web browser.

http://api.open-notify.org/astros.json

Again, the response is in JSON, which is not ordinarily intended for human perception in its raw form, but when we add the following four lines of Python code to the program below, the software will readily extract, and present in a tidy list, the names of the people currently in space.

```
peopleInSpace = responseObject["people"]
for listIndex in range(len(peopleInSpace)):
     person = peopleInSpace[listIndex]
     print(person["name"])
```

Before we get to the code below, which calls a RESTful service and displays the data returned by the service, it will be helpful to learn some fundamental JSON specifications. JSON requires that data be formatted as *name/value* pairs, where *name* is an attribute of an object and *value* is the number, character string, or Boolean value of the attribute for a particular instance of the object (https://www.json.org/json-en.html). For example, the following JSON could be used to depict an employee named George who is currently working as the manager in the sales department and who previously worked as a salesperson and as an assistant manager.

```
{
     "id": 83,
     "name": "George",
     "title": "Manager",
     "department": "Sales",
     "priorpositions": ["Salesperson", "Assistant Manager"],
     "terminated": false
}
```

An actual app processing employee data would likely need information on multiple employees, in which case the data would include a list of employees, as exemplified in the following JSON.

```
{
     "employees": [
          {
          "id": 43,
          "name": "George",
          "title": "Manager",
          "department": "Sales",
          "priorpositions": ["Salesperson",
"Assistant Manager"],
          "terminated": false
          },
          {
          "id": 5,
          "name": "Maria",
          "title": "President",
          "department": null,
          "priorpositions": ["salesperson",
"Assistant Manager",
               "Manager", "Vice President"],
          "terminated": false
          },
          {
          "id": 99,
          "name": "John",
          "title": "Clerk",
          "department": "Mailroom",
          "priorpositions": [],
          "terminated": true
          }
     ]
}
```

The two JSON examples above demonstrate the use of *name/value* pairs. In addition, the second example shows that a value can be null, as in Maria's department. Further, a list can be null, as is the case for John who held no prior positions. JSON is also used to capture more complex objects, which contain lists of objects nested within lists of objects. The fundamental rules for valid JSON are described at json.org. Also, at json.org, those rules are presented in flow diagrams and a grammar specification.

If you completed Section 4.2.5, enter the following program into the interactive Python environment you installed. The program below will enable you to retrieve and display data from REST APIs that return JSON data. At some point, if you decide to access an API that requires authentication, the code below will remain the same; you will only need to add the key and password you created during account creation to the URL assigned to the `restfulURL` variable.

```
import requests

restfulURL = 'https://open.er-api.com/v6/latest/GBP'
response = requests.get(restfulURL)
responseObject = response.json()

for key in responseObject.keys():
     print(key, "\n ", responseObject[key], "\n")
)
```

Once you have that code running, you may add the four lines above to display a tidy list of the people currently in space. Remember to change `restfulURL` to the following URL before running the revised program.

http://api.open-notify.org/astros.json

If you wish to find additional open APIs, one starting place is the Big List of Free Open APIs at https://mixedanalytics.com/blog/list-actually-free-open-no-auth-needed-apis/

The examples above retrieve data from open APIs, which do not require authentication using a key code and a password, but many APIs do require account creation for authentication purposes, and potentially for billing, as well.

When developing apps using JSON, one can verify the validity of the formatting using a software tool, such as *JSONlint* (https://jsonlint.com/) or *JSON Editor* (https://jsonlint.com/). Also, note that the online tool *quicktype* (https://app.quicktype.io) automatically converts JSON to record structures in numerous programming languages (e.g., JavaScript, Python, and Swift).

4.4 Structured Data Transfers in eXtensible Markup Language (XML) with REST and APIs

Like the HyperText Markup Language (HTML), the eXtensible Markup Language (XML) is a derivative of the Standard Generalized Markup Language (SGML). Whereas HTML is used to display information for human beings, XML provides a standard for structuring data exchanged between computers using Application Programming Interfaces (APIs) and, commonly, RESTful web services, as described in Section 4.3. The app that received the computer exchange of information might then process the information and present it in some manner favorable to human perception. In that respect, XML is like JSON. Multiple websites enable conversion of XML to JSON and JSON to XML (e.g., https://www.oxygenxml.com/xml_json_converter.html). Consider the following XML, which documents one sculpted figure and four paintings in a museum.

```
<?xml version="1.0" encoding="UTF-8"?>
<gallery>
     <sculpture>
          <title>Cupid</title>
          <artist>Michelangelo Buonarroti</artist>
          <year>circa 1490</year>
          <medium>Marble</medium>
     </sculpture>
     <painting>
          <title>Portrait of a Prelate</title>
          <artist>Lavinia Fontana</artist>
          <year>circa 1580</year>
          <medium>Oil on copper</medium>
     </painting>
     <painting>
          <title>Portrait of a Man</title>
          <artist>Rembrandt van Rijn</artist>
          <year>1632</year>
          <medium>Oil on wood</medium>
     </painting>
     <painting>
          <title>Portrait of a Woman</title>
          <artist>Rembrandt van Rijn</artist>
          <year>1633</year>
          <medium>Oil on wood</medium>
     </painting>
     <painting>
          <title>Flora</title>
          <artist>Rembrandt van Rijn</artist>
          <year>circa 1654</year>
          <medium>Oil on canvas</medium>
     </painting>
</gallery>
```

To retrieve the data above and display the information about the art works, enter the HTML and JavaScript below using a text editor. Then save the file as *getXMLData.html*.

```
<html>

<head>

<title>Get My XML Data</title>

<script>

function getXMLData() {
     fetch('http://localhost:8080/artGallery.xml')
          .then(response => response.text())
          .then(str => (new window.DOMParser()).parseFromString(str,
               "text/xml"))
          .then(data => {
               galleryNodes =
                   data.getElementsByTagName("gallery")[0].childNodes;
               outputString = '  ';
               for (i = 0; i < galleryNodes.length; i++) {
                   crntNode = galleryNodes[i];
                  if (crntNode.nodeType == Node.ELEMENT_NODE) {
                   outputString += "<h3>" + crntNode.tagName + "</h3>";
                   outputString += crntNode.childNodes[1].tagName + ": " +
                       crntNode.childNodes[1].textContent + "<br>";
                   outputString += crntNode.childNodes[3].tagName + ": " +
                       crntNode.childNodes[3].textContent + "<br>";
                   outputString += crntNode.childNodes[5].tagName + ": " +
                       crntNode.childNodes[5].textContent + "<br>";
                   outputString += crntNode.childNodes[7].tagName + ": " +
                       crntNode.childNodes[7].textContent;
               }
          }
          document.getElementById("contentSpace").innerHTML =
               outputString;});
     }

</script>
</head>

<body onload="getXMLData()">
```

```
<h2>Gallery</h2>
<div id="contentSpace"></div>
</body>
</html>
```

The JavaScript code above consists of one function, which fetches the data using an asynchronous call to the built-in function, `response.txt()`. Then the xml file is parsed, first using the built-in utility function, `DOMParser()`, and then manually in the `for` loop, which steps through the child nodes of `<gallery>`. By attending only to child nodes that are XML elements, the information about each artifact in the gallery is accessed through the `tagName` (which in this case are `title`, `artist`, `year`, and `medium`) and `textContent` attributes.

In order to run the web app above on your computer, you need to open a web server. Accordingly, you may install Node.js from https://nodejs.org/en/. Then open a terminal window and install a web server using the following command.

```
npm install http-server -g
```

Then, in the terminal window, change to the directory in which you have stored your *getXMLData.html* and artGallery.xml files (both of which are available in the Electronic Resources for this book). Then run the web server by entering the following command.

```
http-server -o --cors
```

Once you have the server running, open *getXMLData.html* by double-clicking its file icon. If you happen to be in New York City one day, you may find the artifacts listed in your web browser window in the Museum of Metropolitan Art (https://www.metmuseum.org/).

4.5 Controlling Databases with Structured Query Language (SQL)

Apps store data in multiple forms. Text editors, for instance, store alphanumeric characters (a combination of letters, numbers, punctuation marks, and various other symbols, such as * ^ < > | * $), which human beings can view easily outside the app used to create the file. A word processing program stores both text and formatting information to make text appear bold, italicized, and underlined, as well appear in a particular typeface and size. The storage method for such data may be propriety and known only to a particular company, or the data may be stored in an open format. For instance, in 2007, Microsoft switched from storing documents in the proprietary.doc format to an open XML format (.docx). Hence, programs such as *Google Docs* can read. docx files. In addition to unformatted and formatted text, images, sounds, and video are stored in a variety of proprietary and open formats containing bytes of data, which human beings do not view one byte at a time, but perceive the data as rendered by software that produce images, sounds, and video. Collections of multimedia data may be stored as *document databases*, *key-value databases*, *wide-column stores*, and *graph databases*, which may be categorized as NoSQL database formats. Those formats contrast the storage of multimedia data in *relational databases*, which contain electronic records of multimedia data stored in tables related by one or more common fields. Relational databases provide access to records using Structured Query Language (SQL) or a proprietary app (e.g., Microsoft Access, LibreOffice Base, dBase).

SQLite is a free and open-source tool for creating and controlling relational databases. To try *SQLite*, open a web browser and go to https://www.sqlite.org/. Then click the Download tab and then click the link to the precompiled binary for your operating system, whether MacOS, Windows, or Linux. Decompress the.zip file to unpack the *SQLite* tools and double-click the icon to open sqlite3, which will open in a terminal or console window with the `sqlite>` prompt.

In SQL, a table is created using the Create table command, as exemplified below. The SQL commands below are available in the *sqlcommands.txt* file in the Electronic Resources for this book. You may copy the commands from that file and paste them after the `sqlite>` prompt or you may type the commands. For single-line commands, just press the Enter/Return key after typing the line. For multi-line commands, like the one below, hold the Shift key at the end of each line except the last line. Remember to include the semicolon at the end of each command, otherwise *SQLite* will produce a blank line until you enter the semicolon.

```
create table person
(id int primary key,
firstname text,
lastname text,
birthdate datetime);
```

When the command above has been entered and executed, the `sqlite>` prompt returns. Unless an error is encountered, expect only that the `sqlite>` prompt will reappear, which indicates that the system is ready for the next SQL command.

After creating a relational database table, complete control over the records in the table are managed using four operations: Create, Read, Update, and Delete (CRUD). In SQL, a record is created using the `insert` command, as in the following examples. For a `datetime` field, the date takes the form YYYY-MM-DD and the time, which is optional, takes the form HH:MM:SS. SSS. The seconds can be entered with three decimal places at most, though the decimal portion is optional.

Enter the following three commands, one at a time. Again, expect no visual indication that the command executed, except the return of the `sqlite>` prompt. Evidence that the commands have executed is forthcoming.

```
insert into person (id, firstname, lastname, birthdate) values
(1, 'Julio', 'Cordero', '2000–01–03');

insert into person (id, firstname, lastname, birthdate) values
(2, 'Jane', 'Smith', '2001–12–31 11:34:21.456');

insert into person (id, firstname, lastname) values (3, 'John', 'Smith');
```

The `select` command is used to read (retrieve) records from a database table. To retrieve the data in particular fields, specify the field names, as shown below. Run the command, which will retrieve and display the first and last names of the three records inserted into the table.

```
select firstname, lastname from person;
```

To view the data in all fields, the asterisk is used to represent all field names, as in the following example. Run the SQL command below, which will retrieve and display the data in all fields in all three records.

```
select * from person;
```

Records can be retrieved selectively by including a `where` clause, as in the following example. Give it a try.

```
select firstname, lastname from person where id < 3;
```

Appropriately, the `update` command alters one or more fields in one or more records. Run the following command. Again, expect only the return of the `sqlite>` prompt.

```
update person set firstname = 'Jan' where id = 2;
```

To verify that the SQL command executed properly, you could enter and run the following command. Alternatively, in the *SQLite* console window, press the Up arrow key twice and the command below, which you entered earlier, will reappear. Then you need only press the Enter/Return key.

```
select firstname, lastname from person where id < 3;
```

Lastly, the delete command is used to remove one or more records. Always use a where clause to delete records selectively. Without a `where` clause to limit record deletions, every record in the table will be deleted! Verify this for yourself.

```
delete from person where id = 2;

select firstname, lastname from person;

delete from person;

select firstname, lastname from person;
```

Now the `person` table is empty, and will need to have records inserted again, if you wish to continue using the table.

References

Aho, A. (2011, January). Computation and computational thinking. *Ubiquity*. https://dl.acm.org/doi/10.1145/1922681.1922682

Beecher, K. (2017). *Computational thinking: A beginner's guide to problem-solving and programming*. BCS Learning & Development.

Bell, T., & Lodi, M. (2019). Constructing computational thinking without using computers. *Constructivist Foundations, 14*(3), 342–351.

Bers, M. U. (2021). *Coding as a playground: Programming and computational thinking in the early childhood classroom (2nd ed.)*. Routledge.

Brennan, K., & Resnick, M. (2012, April). *New frameworks for studying and assessing the development of computational thinking* [Paper presentation]. 2012 Annual Meeting of the American Educational Research Association, Vancouver, BC, Canada.

Casperen, M. E., Gal-Ezer, J., McGettrick, A., & Nardelli, E. (2018). *Informatics for all: The strategy*. ACM Europe. https://acct-europe.acm.org/binaries/content/assets/public-policy/acm-europe-ie-i4all-strategy-2018.pdf

Casperen, M. E., Gal-Ezer, J., McGettrick, A., & Nardelli, E. (2019). Informatics as a fundamental discipline for the 21st century. *Communications of the ACM, 62*(4), 58–63. https://doi.org/10.1145/3310330

Chemuturi, M. (2018). *Software design: A comprehensive guide to software development projects*. CRC Press.

Corradini, I., Lodi, M., & Nardelli, E. (2017). Conceptions and misconceptions about computational thinking among Italian primary school teachers. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 136–144). https://dl.acm.org/doi/10.1145/3105726.3106194

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33–39. https://doi.org/10.1145/2998438

Denning, P. J., & Tedre, M. (2019). *Computational thinking.* The MIT Press.

Duncan, C., Bell, T., & Atlas, J. (2017). What do teachers think? Introducing computational thinking in the primary school curriculum. In *Proceedings of the Nineteenth Australasian Computing Education Conference* (pp. 65–74). https://doi.org/10.1145/3013499.3013506

Felleisen, M., Findler, R. B., Flatt, M., & Krishnamurthi, S. (2018). *How to design programs: An introduction to programming and computing (2nd ed.).* The MIT Press.

Forlizzi, L., Lodi, M., Lonati, V., Mirolo, C., Monga, M., Montresor, A., Morpurgo, A., & Nardelli, E. (2018). A core informatics curriculum for Italian compulsory education. In *Eleventh International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 141–153). St. Petersburg, Russia. https://hal.inria.fr/hal-01913057/document

Gallup & Google. (2016). *Trends in the state of computer science in U.S. K-12 schools.* https://services.google.com/fh/files/misc/trends-in-the-state-of-computer-science-report.pdf

Hanson, C., & Sussman, G. J. (2021). *Software design for flexibility: How to avoid programming yourself into a corner.* The MIT Press.

Heintz, F., Mannila, L., & Färnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in k-12 education. In *Proceedings of the IEEE Frontiers in Education conference* (pp. 1–9). https://ieeexplore.ieee.org/document/7757410

Kleppmann, M. (2017). *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems.* O'Reilly Media.

Kong, S.-C., & Abelson, H. (Eds.). (2019). *Computational thinking education.* Springer. Open access: https://link.springer.com/book/10.1007/978-981-13-6528-7

Lauret, A. (2019). *The design of web APIs.* Manning Publications.

Louden, K. C., & Lambert, K. A. (2012). *Programming languages: Principles and practice.* Course Technology/Cengage.

Nardelli, E. (2019). Do we really need computational thinking? *Communications of the ACM, 62*(2), 32–35. https://doi.org/10.1145/3231587

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2019). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry, 11*(1), 1–17. https://doi.org/10.1080/20004508.2019.1627844

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* Basic Books.

Petre, M., & van der Hoek, A. (2016). *Software design decoded.* The MIT Press.

Polya, G. (1945). *How to solve it: A new aspect of mathematical method.* Princeton University Press.

Polya, G. (2004). *How to solve it: A new aspect of mathematical method.* Princeton University Press.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67. https://dl.acm.org/doi/10.1145/1592761.1592779

Rich, P. J., & Hodges, C. B. (Eds.). (2017). *Emerging research, practice, and policy on computational thinking.* Springer.

Richards, M., & Ford, N. (2020). *Fundamentals of software architecture: An engineering approach.* O'Reilly Media.

Vegas, E., & Fowler, B. (2020). *What do we know about the expansion of K-12 computer science education? A review of the evidence.* Brookings. https://www.brookings.edu/research/what-do-we-know-about-the-expansion-of-k-12-computer-science-education/

Wang, P. S. (2016). *From computing to computational thinking.* The CRC Press.

Webb, M., Bottino, R. M., Passey, D., Kalas, I., Bescherer, C., Malyn Smith, J., Angeli, C., Katz, Y., Micheuz, P., Rosvik, S., Brinda, T., Fluck, A., Magenheim, J., Anderson, B. B., & Fuschek, G. (2019). *Coding, programming and the changing curriculum for computing in schools.* Report of UNESCO/IFIP TC3 meeting at OCCE, June 27, 2018, Linz, Austria.

Williams, H. (2017). *No fear coding: Computational thinking across the K-5 curriculum.* International Society for Technology in Education.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. http://www.cs.cmu.edu/~wing/publications/Wing06.pdf

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society, 366*, 3717–3725. http://www.cs.cmu.edu/~wing/publications/Wing08a.pdf

Wing, J. M. (2010). *Computational thinking: What and why?* theLink. https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Wing, J. M. (2014, January 10). Computational thinking benefits society. *Social Issues in Computing*. http://socialissues.cs.toronto.edu/2014/01/computational-thinking/

Wing, J. M. (2016). Progress in computational thinking. *Communications of the ACM, 59*(7), 10–11. https://dl.acm.org/doi/pdf/10.1145/2933410

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education, 141*, 1–25.

Part III

Instructional Scenarios

5 Social Studies

Mapping Our World

5.1 Theme

When engaged in Social Studies, we might focus on individuals (psychology) or groups of people (sociology). Alternatively, we might study human societies and cultural development (anthropology), artifacts produced by people (archaeology), places inhabited by people (geography), languages created by people (linguistics), spiritual beliefs of people (religion), the fundamental nature of people (philosophy), or trading behaviors among people (economics). We might also study law and politics, which could be conceived as subsets of anthropology. Those fields of study within Social Studies may be considered distinctly or in combination, such as studies of geography and politics (geopolitics). At times we might study those specialties separately, while at other times we might consider them collectively and recognize that history encompasses them all. Of the multiple specialty fields within Social Studies, this chapter pertains to geography in general and mapping in particular.

Maps of Earth are images that represent areas of land and sea. Social scientists use maps as a tool for understanding and explaining human activity. Accordingly, quick and easy access to free maps is beneficial. Thankfully, those who study geodesy and cartography have made mapping data and rendering software widely available for free. Consequently, after downloading free mapping data using a web browser, one can view a detailed map of the world in about five seconds by dragging and dropping a few file icons on top of a web browser window at mapshaper.org. In Section 3, we will download geographic data on the web and render some maps using that approach, as well as create maps in *QGIS*, free software for Geographic Information Systems (GIS). First, though, let's acknowledge the wide variety of maps and recognize that all two-dimensional maps of Earth are somewhat distorted. Further, let's consider curricular goals and curricular standards pertaining to mapping.

First, there are so many types of maps that cartographers categorize them. Australia's Intergovernmental Committee on Surveying and Mapping classifies maps into five categories: (1) General Reference (or planimetric) maps depict important physical features of an area (e.g., geopolitical maps; street maps; tourist maps); (2) Topographical (or relief) maps show differences in landscape height using contour lines; (3) Thematic maps convey information about a particular topic, such as weather, geology, or vegetation; thematic maps may be choropleths, which show map regions shaded in accordance with a numeric attribute, such as population density; (4) Navigation charts are used primarily by mariners and pilots; and (5) Cadastral maps identify regions based on land ownership. Maps may be contemporary or historical. Mapping ontology is one topic pursued by cartographers. For an overview of the diverse topics of interest to them, you may wish to consider the article "Mapping the World" (Cartwright & Ruas, 2015), which is published in the first issue of the *International Journal of Cartography*.

Second, we often benefit from two-dimensional maps of Earth because they enable us to view places on the planet without rotating a globe. However, a map projection from points on a sphere to a plane incurs some level of distortion. In some cases, a two-dimensional plane can be derived

DOI: 10.4324/9781003172284-8

from a three-dimensional object. For example, imagine that you have enclosed Earth in a cylinder. Further, the Earth is hollow, but the land masses still exist on the surface of the planet. Positioned within the hollowed Earth, you have a device that produces straight beams of light. You use that device to project points along the edges of Earth's land masses to the cylinder. After projecting points from Earth to the cylinder, you cut the cylinder along a perpendicular and unfold it to produce a two-dimensional (rectangular) map depicting Earth's land masses. That scenario is simulated in a video by Johnny Harris (2016). One could project points from Earth to a solid other than a cylinder, but whatever the shape of the solid, distortion occurs. As discussed in the Harris video, history identifies Carl Friedrich Gauss (1777–1855) as the first person to prove that all projections from a sphere to a two-dimensional surface have some distortion. In maps rendered using the Mercator projection, for instance, areas near the poles are greatly enlarged compared to regions near the equator. The Mercator projection, created in 1569 by Gerardus Mercator, enables maritime navigation along a line of constant course (a rhumb line), which made the projection very popular for centuries. However, Antarctica, Russia, Greenland, and the Nunavut territory in Canada appear much larger than their actual sizes. As noted by Routley (2017), maps using the Mercator projection depict Canada and Russia as occupying about 25% of Earth's land, but collectively they actually occupy about 5% of the land. Robert Israel (2003) discusses the Mercator projection, which includes a map depicting the distortion of the Mercator projection, as well as a map with a sample rhumb line. Israel's (2003) presentation (a single web page) considers the mathematics underlying the Mercator projection from both historic and contemporary perspectives. In light of the Harris (2016) video and Israel's (2003) presentation, an informed interpretation of a map requires awareness of the distortion effects of its map projection.

5.2 Sample Learning Goals

Before considering differences in learning goals for students in England, British Columbia, Canada, and North Carolina, USA, it is helpful to recognize how students are grouped in their respective school systems. In England, groups of grades are categorized by four Key Stages, as shown in Table 5.1. In contrast, school grades in Canada and the United States are grouped into three categories, elementary school, middle or junior high school, and high school, as shown in Table 5.2. Since schooling is primarily a regional matter in Canada and the USA, provinces create laws and curricula pertaining to education in Canada, and states do the same in the USA. Further, public school districts within provinces and states have some flexibility to group students according to local norms. Hence, for instance, one will find some middle schools in Canada.

Table 5.1 Key Stages in Schools in England

	Key Stage 1	*Key Stage 2*	*Key Stage 3*	*Key Stage 4*
Age	5–7	7–11	11–14	14–16
Year groups	1–2	3–6	7–9	10–11

Table 5.2 Groupings of Grades in Schools in Canada and the USA

Canada	*Elementary School*						*Junior High School*			*High School*		
Age	6	7	8	9	10	11	12	13	14	15	16	17
Grade	1	2	3	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	12
USA	Elementary School					*Middle School*			*High School*			

In England, Canada, and the USA, studies in English, Mathematics, and Science are required in all grades. In England, three other subjects are compulsory in all Key Stages, namely Computing, Religious Education, and Physical Education (Department for Education, 2014a). The treatment of Social Studies differs between regions. In England, for Key Stages 1, 2, and 3, Geography is a distinct compulsory subject, as is History. In contrast, through middle and junior high schools in Canada and the USA, studies in geography and history are embedded in the compulsory subject, Social Studies. In England, instruction in Geography, including mapping, ends with Key Stage 3, when students are about 14 years of age. In Key Stage 4, students in England study Citizenship, which is a compulsory subject. In high schools in Canada and the USA, Geography and History are distinct subjects. Commonly in the USA, American History and World History courses are mandatory in high school. High school students in Canada study Canadian History and World History.

Learning goals pertaining to geography, and mapping in particular, appear in programs of study for children and adolescents. Children in the early grades are presented maps and begin to encounter representations of places. They become aware of a variety of maps (e.g., political, physical, ordnance survey, and topographic) and gain rudimentary skills needed to use them. In adolescence, emphasis shifts from mere awareness of maps and attainment of rudimentary skills to use of maps for acquiring information that can be used to justify positions about the movement of goods and people, for instance.

The sample learning goals pertaining to geography and mapping, which appear below, have been copied from the curriculum guides used in England; British Columbia, Canada; and North Carolina, USA.

England

Department for Education (2014b)
2014–Present
National Curriculum in England: Framework document

Geographical skills and fieldwork (Key Stage 3, Years 7–9)

- build on their knowledge of globes, maps and atlases, and apply and develop this knowledge routinely in the classroom and in the field (p. 244)
- interpret Ordnance Survey maps in the classroom and the field, including using grid references and scale, topographical and other thematic mapping, and aerial and satellite photographs (p. 244)
- use Geographical Information Systems (GIS) to view, analyse and interpret places and data (p. 244)
- use fieldwork in contrasting locations to collect, analyse and draw conclusions from geographical data, using multiple sources of increasingly complex information (p. 244)

Notice that the first and second objectives above refer to maps explicitly. The third objective requires use of GIS, and the fourth objective refers explicitly to geographical data.

British Columbia, Canada

Ministry of Education (2016)
Grades 7, 8, 9
Use Social Studies inquiry processes and skills to ask questions; gather, interpret, and analyze ideas; and communicate findings and decisions (pp. 33, 38, 43)

Key skills:

- Compare the advantages and disadvantages of various graphic forms of communication (e.g., graphs, tables, charts, maps, photographs, sketches). (pp. 34, 39)
- Demonstrate an ability to interpret scales and legends in graphs, tables, and maps (e.g., climograph, topographical map, pie chart). (pp. 34, 39)
- Compare maps of early civilizations with modern maps of the same area. (pp. 34, 39)
- Select an appropriate graphic form of communication for a specific purpose (pp. 34, 39)
- Construct graphs, tables, and maps to communicate ideas and information, demonstrating appropriate use of grids, scales, legends, and contours. (p. 44)

In British Columbia (BC), students in junior high school (Grades 7, 8, and 9) are required to gain mapping skills, as stated explicitly above in the *Key skills* section of the learning standards. With respect to content elaboration, the BC curriculum requires that students analyze the following.

- human responses to particular geographic challenges and opportunities, including climates, landforms, and natural resources (p. 35)
  - Create maps to show the key physical environmental characteristics of a selected ancient culture. (p. 36)
- the credibility of multiple sources and the adequacy of evidence used to justify conclusions (evidence): (p. 39)
  - What do different systems of mapping and cartography indicate about the cultures from which they emerged? (p. 39)

Grades 10 and 11 Social Studies

Ministry of Education (2018a, 2018b)

Big Idea: The development of political institutions is influenced by economic, social, ideological, and geographic factors. (2018a, p. 1)

Grade 12 Human Geography

Ministry of Education (2018c)

Use geographic inquiry processes and skills to ask questions; gather, interpret, and analyze data and ideas; and communicate findings and decisions. (p. 1)

Sample topics:

- Map skills (p. 2)
  - Use a map for navigation. (p. 2)
  - Understand a map legend. (p. 2)
  - Use map scales. (p. 2)
  - Understand latitude and longitude. (p. 2)
  - Understand topographic maps and contour lines. (p. 2)
- Mapping software and GIS tools (p. 2)
- Interpreting satellite images and photos (p. 2)

In Grades 10 and 11 Social Studies courses, no explicit mention is made of mapping, though consideration of geographic factors on political institutions is required. The *Human Geography* course in Grade 12 requires attainment of map skills, including the use of mapping software and GIS tools.

North Carolina, USA

North Carolina State Board of Education and Department of Public Instruction (2012a, 2012b, 2012c)

North Carolina Essential Standards: Sixth Grade Social Studies
North Carolina Essential Standards: Seventh Grade Social Studies
North Carolina Essential Standards: Eighth Grade Social Studies
Grade 6 (2012a, p. 3)
Geography and Environmental Literacy
6.G.2. Apply the tools of a geographer to understand the emergence, expansion and decline of civilizations, societies and regions.

6.G.2.1. Use maps, charts, graphs, geographic data and available technology tools to draw conclusions about the emergence, expansion and decline of civilizations, societies and regions.

6.G.2.2. Construct maps, charts and graphs to explain data about geographic phenomena (e.g., migration patterns and population and resource distribution patterns).

Grade 7 (2012b, p. 3)

7.G.2. Apply the tools of a geographer to understand modern societies and regions.

7.G.2.1. Construct maps, charts, and graphs to explain data about geographic phenomena (e.g. migration patterns and population and resource distribution patterns).

7.G.2.2. Use maps, charts, graphs, geographic data and available technology tools (i.e. GPS and GIS software) to interpret and draw conclusions about social, economic, and environmental issues in modern societies and regions.

Grade 8 (2012c, p. 4)

8.G.1. Understand the geographic factors that influenced North Carolina and the United States.

8.G.1.1. Explain how location and place have presented opportunities and challenges for the movement of people, goods, and ideas in North Carolina and the United States.

In North Carolina, students in Grades 6 and 7 are required to use and construct maps, as stated above. The curriculum for Grade 8 emphasizes analysis and explanation over attainment of map skills, as noted in the essential standard on geographic factors.

North Carolina State Board of Education and Department of Public Instruction (2010a, 2010b, 2010c)

North Carolina Essential Standards: Social Studies – American History Course I
North Carolina Essential Standards: Social Studies – American History Course II
North Carolina Essential Standards: Social Studies – World History Course

The objectives below for high school students in North Carolina identify requirements to analyze maps, but no standard requires attainment of mapping skills or use of GIS software.

High School American History I, American History II, and World History
Use Historical Comprehension to: (2010a, 2010b, p. 3; 2010c, p. 2)

- Analyze data in historical maps.
- Analyze visual, literary and musical sources.

5.3 STELLAR Activities

In this section we leverage public data in order to create a world map using *Mapshaper*, which is a freely available web app. In addition, we use a free program called *QGIS* to render GIS data and look for patterns in data. After converting the GIS data to Scalable Vector Graphics (SVG) in *Mapshaper*, we modify the map colors using a text editor. For those who favor visual apps over text editing, we also modify the map colors using a free program called *Inkscape*. (Links to *Mapshaper*, *QGIS*, and *Inkscape* are available in the Annotated Resources section.) In Section 5.3.3, we create a globe of Earth in *Blender*, which you are encouraged to send to a 3D printer. In Section 5.3.4, we leverage SVG data to design a web app and then create the app in JavaScript in Section 5.3.5.

Acquiring Free GIS Data

In order to create accurate digital maps, we must first obtain geographic data. Fortunately, numerous organizations make geographic data available at no financial cost. As a bonus, the datasets have already been prepared for immediate use. For the map rendering in this section, we will retrieve data from the Natural Earth website, which contains numerous sources of data described below and in the Annotated Resources section.

1. Open a web browser to https://www.naturalearthdata.com/ (also, see note below)
2. Click the Features tab and scroll down to the **Countries** section under the heading, **Cultural Vector Data Themes**
3. Click the yellow rectangle labelled **110**
4. Click the **Download countries** button in order to retrieve the data

Alternatively, you could retrieve the data by opening a web browser to the following URL:

https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/110m/cultural/ne_110m_admin_0_countries.zip

Note: When the Natural Earth file server was not functional, a message was posted on the home page of their web server (https://www.naturalearthdata.com) to retrieve the files from the following URL: https://gist.github.com/DanielJWood/b71237cc200831acf8e637c05ce2c375-#file-natural_earth_s3_links-md

Once the download completes, the file named **ne_110m_admin0_countries.zip** will appear in your folder for downloaded files. Unzip the file (double-click the file on a Mac; or in Windows, double-click the file and extract the contents to a folder). You now have the data needed for the initial map renderings, but as long as we are downloading data from the Natural Earth website, let's also download files for the activities in the *Second Round of Map Renderings* and the *Design and Create Unique Web App* sections below.

1. Open a web browser to https://www.naturalearthdata.com/
2. Click the Features tab and scroll down to the **First order admin** section under the heading, **Cultural Vector Data Themes**
3. Click the blue rectangle labelled **50**
4. Click the **Download states and provinces** button
5. Go back to the previous web page (Click back arrow icon or press Ctrl-left arrow key on Windows or Command-left arrow key on a Mac)
6. Beside **Populated places** (the first label below First order admin), click the blue rectangle labelled **50**
7. Click the **Download populated places** button
8. Go back to the previous web page
9. Scroll down to the **Ocean** label in the **Physical Vector Data Themes**
10. Click the green rectangle labelled **10**
11. Click the **Download ocean** button
12. Go back to the previous web page
13. Scroll down to the **Lakes** label in the **Physical Vector Data Themes**
14. Click the green rectangle labelled **10**
15. Click the **Download lakes** button
16. Go back to the previous web page
17. Scroll down to the **Gray Earth** label in the **Raster Data Themes**
18. Click the green rectangle labelled **10**
19. Under the **Gray Earth with Shaded Relief, Hypsography, and Ocean Bottom** title, click the **Download medium size** button

Alternatively, you could open a web browser to each of the following URLs in order to retrieve the data:

https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/50m/cultural/ne_50m_admin_1_states_provinces.zip

https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/50m/cultural/ne_50m_populated_places.zip

https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/physical/ne_10m_ocean.zip

https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/physical/ne_10m_lakes.zip

https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/10m/raster/GRAY_LR_SR_OB.zip

5.3.1 Rendering and Styling Maps in **Mapshaper** *and* **Inkscape**

Here is a quick four-step method to view geographic data using a tool called *Mapshaper*.

Rendering GIS Data in Mapshaper

1. Open a web browser to https://mapshaper.org/
2. In the **ne_110m_admin_0_countries** folder, select *ne_110m_admin_0_countries.dbf*, *ne_110m_admin_0_countries.prj*, and *ne_110m_admin_0_countries.shp*
3. Drag and drop the three selected files on the bottom portion of the browser window
4. Click the **Import** button

You now have an accurate and detailed world map. By leveraging free GIS data, you have produced a map in a few seconds. The map conforms to the projection selected by Natural Earth, which is referred to by its World Geodetic System (WGS) code, WGS84 (84 refers to 1984). The WGS84 projection is also coded in the EPSG (European Petroleum Survey Group) geodetic system as EPSG:4326. In addition to applications for cartography and geodesy, WGS84 is the reference coordinate system for the popular Global Positioning System (GPS).

In addition to the data needed to display the world map on your screen, the data you downloaded and rendered in *Mapshaper* have several hidden attributes (which are also known as properties or fields). You can view those attributes. Notice the four icons that comprise a sidebar on the right side of the *Mapshaper* window. The home icon at the top brings you to the current display; the plus sign (+) permits you to zoom in; the minus sign (-) permits you to zoom out (a mouse or track pad also permits zooming); and the bottom icon, the cursor arrow, permits you to view the hidden attributes, such as the names, in multiple languages, and abbreviations of the countries.

View Hidden Data in Mapshaper

1. Click the **Arrow** icon in the sidebar on the right of the *mapshaper* app
2. Hover the cursor over a country to view the values of the hidden attributes for that country. Rather than hover, you may click the country to display the hidden data.
3. Look for patterns in the data. For example, locate regions of the world classified as high, medium, and low-income by looking at the value in the `INCOME_GRP` field for multiple countries. Alternatively, or additionally, you may look for regions of the world with countries grouped according to other economic classifications using the `ECONOMY` field or the `GDP_MD_EST` field. You may also be interested in discerning a relationship between population and population

rank using the `POP_EST` or `POP_RANK` fields. In the steps below for filtering data, we will consider a much faster method for revealing patterns in data, but it is important to know that GIS data include multiple fields, many of which do not affect the visual display of the map.

Click the **Arrow** icon in the sidebar again to stop viewing hidden data.

At times, you may want to view maps in *Mapshaper* using a projection other than the default. For example, complete the two steps below to change from the default WGS84 projection to the Robinson projection.

Change Projection in Mapshaper

1. Click the **Console** Tab
2. At the $ prompt, enter the following
   ```
   -proj robinson
   ```

To view a list of all possible projections, enter the following at the $ prompt.

```
-projections
```

In light of the list of projections, change the projection again. Note that switching projections may cause line intersection warnings and display spurious lines. Restoring the display is as easy as dropping the three original files (i.e., *ne_110m_admin_0_countries.dbf, ne_110m_admin_0_countries.prj*, and *ne_110m_admin_0_countries.shp*) on top of the browser window displaying the map. Doing so adds a layer. You can go back to previous layers by clicking the current layer name, which appears in the center of the title bar. A triangle icon appears after the layer name to convey that it is a drop-down menu.

Clicking the **Console** tab a second time will close the console window and return the map to the full size of the browser window.

Filtering Data in Mapshaper

1. Open the console by clicking the **Console** tab if the console is not already open
2. At the $ prompt, enter the following to focus on countries classified as high income
   ```
   -filter 'INCOME_GRP == "1. High income: OECD"'
   ```
3. In an instant, the regions of the world with countries classified as high income become evident in the map.
4. Drag the three original files on top of the browser window to restore the full set.
5. At the $ prompt, enter
   ```
   -filter 'INCOME_GRP == "3. Upper middle income"'
   ```
6. Again, instantly, the regions of the world with countries meeting the filtering criterion become evident.
7. Drag the three original files on top of the browser window to restore the full set
8. At the $ prompt, enter the following to focus on countries classified as high income
   ```
   -filter 'INCOME_GRP == "5. Low income"'
   ```
9. This time, instantly, the regions of the world with countries classified as low income become evident.

The filter command is followed by an expression enclosed in single quotes. In the cases above, the filter expression checks the value of the `INCOME_GRP` field which appears entirely in uppercase letters because that is the form defined by the dataset, for being equal to a particular string of characters. As the developer, you can filter the data using any field, any comparison operator, and any value. You may also filter on other economic indicators, such as the `ECONOMY` field or the `GDP_MD_EST` field. As in the manual inspection of data above, you may also filter the data based on the `POP_EST` or `POP_RANK` fields.

Next, we will filter the data to focus on one particular country. Data filters do not eliminate countries, but their borders appear faded, and they are disabled (so hovering over them will not show hidden data). The first two steps below put the focus on Canada by fading the borders of the other countries. The third step below makes the other countries completely disappear from view.

1. Drag the three original files (*ne_110m_admin_0_countries.dbf, ne_110m_admin_0_countries.prj* and *ne_110m_admin_0_countries.shp*) on top of the browser window to restore the full set.
2. At the $ prompt, enter the following to focus on Canada
   `filter 'ADMIN == "Canada"'`
3. At the $ prompt, enter
   `–clean`

The converse of the filter above would search the ADMIN field for country names not equal (!=) to Canada, as shown below.

```
–filter 'ADMIN!= "Canada"'
```

To search by three-letter country codes assigned by the International Organization for Standardization (ISO 3166-1:2013), use the ISO_A3 field, as in the examples below.

```
–filter 'ISO_A3 == "GBR"'
–filter 'ISO_A3 == "USA"'
```

As before, drag the three original files on top of the browser window to restore the full dataset.

Another important feature of *Mapshaper* is data simplification, which is actually data reduction. Note that reducing the amount of data diminishes the level of detail in the map. Give the following steps a try.

Data Reduction in Mapshaper

1. Click **Simplify**
2. Move the slider from left to right (or you could change the value in the percentage field). Notice that as you move the slider to the right, countries begin to disappear, and the shape of visible countries is simplified as the percentage of detail goes down from 100%.

At times, you may want to create a unique app using data exported from *Mapshaper*. In *Mapshaper*, you can export the GIS data to other types of data, including SVG format.

Exporting GIS Data to SVG Format

1. Click the **Export** tab
2. Select one layer (if you dropped the files over the window multiple times, you will have multiple layers; in that case, uncheck the boxes so that only one layer is checked)
3. Click the **SVG** radio button
4. Click the **Export** button

Since the exported file was saved in SVG format, you can open the .svg file in a web browser. Beware, the color scheme is not the same as in *Mapshaper* and the color contrast between country borders (the stroke color) and the country fill color is minimal, so look carefully to see the country borders.

Once you are convinced that the default stroke and fill colors are unsatisfactory, you may change them, either by editing the SVG tag in *NotePad* on Windows or in *TextEdit* on a Mac. The second line of the SVG file contains the SVG tag below, absent the text in bold. In *NotePad* or *TextEdit*, type the text in bold below in order to specify the stroke and fill colors. (The text you type in your text editor will not appear bold.) Save the file and reload it (press Ctrl-R or Command-R) in your web browser in order to view a clear map.

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
width="800" height="387" viewBox="0 0 800 387" stroke="white" fill="gray"
stroke-linecap="round"
stroke-linejoin="round">
```

If you prefer not to edit text, you can change the stroke and fill colors in a vector graphics editing program, such as the free open-source program *Inkscape*. As described in the Annotated Resources section, no registration is required to download *Inkscape*; you can go directly to the current version of the software through the Download tab at the *Inkscape* website (https://inkscape.org/). Installation is generally straightforward on contemporary computers with adequate memory, speed, and a recent version of the operating system.

Changing Stroke and Fill Colors in Inkscape

1. In *Inkscape,* open the SVG file you created in the previous step
2. Drop down the **Object** menu and select **Fill and Stroke…**
3. Click the map to select it
4. Click a color swatch along the bottom (a 50% or 60% gray provides sufficient contrast)
5. If you wish, you can also change the stroke color by clicking the **Stroke Paint** tab in the **Properties** window; then click the **Flat color** icon (the rectangle immediately to the right of the X in the **Properties** window below the **Stroke Paint** tab); lastly, click a color swatch along the bottom or adjust the Red, Green, and Blue quantities, either by clicking within the red, green, and blue gradient rectangles or by changing their numeric values
6. Save the SVG file
7. Open the SVG file in a web browser to confirm the color changes

See the Annotated Resources section for links to the Inkscape documentation and to their online tutorial.

5.3.2 Rendering and Styling Maps in QGIS

In this section we will use *QGIS*, an open-source GIS to create two distinct maps. The link to the *QGIS* website appears in the Annotated Resources section, along with information about downloading and installing the software. Of the two maps we will create in *QGIS*, one will be a physical map of the world and the other a political map of the world. The physical map will depict land elevations and ocean depths worldwide. We will also add lakes and an ocean layer to the physical map of the world. We will consider styling options, which you may implement. The political map will include country borders, as well as regional borders for Australia, Brazil, Canada, and the United States of America. In addition, the political map will include major cities and towns. We will style the map to highlight particular features. We will also filter the map in order to focus on particular countries and change some of the default colors.

Creating the Physical Map

1. Open *QGIS*
2. Start a new project by dropping down the **Project** menu and selecting **New**
3. From the folder containing the Gray Earth files, drag the *GRAY_LR_SR_OB.tif* file and drop it on the empty canvas in *QGIS*
4. From the folder containing the Natural Earth data for Lakes, drag the *ne_10m_lakes.shp* file on top of the current map
5. To view the data pertaining to the lakes, drop down the **Layer** menu and select **Open Attribute Table**
6. Scroll through the list of data to view the field names and the values in the fields. You may use the arrow keys or the tab key. The largest lakes have scalerank = 0. Let's filter the data to display only those lakes. Close the attribute table.
7. To display the **Query Builder** dialog, drop down the **Layer** menu and select **Filter …**
8. In the **Fields** section, double-click the scalerank; In the **Operators** section, click the = button; on your keyboard, tap the digit 0. Notice that the Filter Expression (`"scalerank" = 0`) is now complete. Click the **Test** button at the bottom of the dialog. The data have now been filtered (temporarily) and the map updated to display only the largest lakes. You may reject the filter by clearing the filter expression, or by clicking the **Cancel** button (which would also close the dialog). Accept the filter and close the dialog by clicking **OK**
9. To change the default color of the lakes, click the **Styling Panel** icon (left-most icon) in the **Layers** panel (which is in the sidebar on the left). Alternatively, drop down the **View** menu; hover over **Panel**s; and select **Layer Styling**
10. Beside the **Color** label, either click on the colored bar or click the triangle to the right of the colored bar in order to select a color. (You may try both methods to view the differences in the dialogs.) There are several options for selecting a color, whether by moving sliders; clicking on a color palette; or entering hue, saturation, and value, or by entering red, green, blue (e.g., for blue lakes, you might try red: 60, green: 150, and blue: 225). When you have finished selecting the lake color, close the **Layer Styling Panel** (e.g., click the x in the top-right corner of the panel)
11. Save your work

Make any additional changes you wish. In the next set of steps, we will add an ocean layer, but at this point, you may wish to bring some or all of the lakes back. Just return to the Query Builder and adjust the Filter Expression (e.g., you could try "scalerank" < 2).

1. From the folder containing the Natural Earth data for oceans, drag the *ne_10m_ocean.shp* file on top of the current map
2. Sometimes, default colors are unsatisfactory. If the oceans appear in an undesirable color, you could remove the layer (right-click or control-click on the layer name in the **Layers** panel; and select **Remove Layer**…). Alternatively, you could toggle the layer off by clicking the checkbox containing the checkmark, which is beside the layer name and color swatch in the Layers panel. I recommend neither of those options. Rather, in the next step, let's make the ocean color a dark blue and reduce the opacity to bring back the rich depiction of the ocean depths.

3. As before, open the **Layer Styling Panel**. For color, try red: 60, green: 60; and blue 235. Also, try setting the opacity to 37% and to 27% and to other values until you are satisfied with the appearance of the map. Be sure to return the map to full size before making your final adjustments.
4. Save your work

Creating the Political Map

As you pursue the following steps, save the file at various points, as you wish.

1. Open *QGIS*
2. Start a new project by dropping down the **Project** menu and selecting **New**
3. From the folder containing the Natural Earth data for countries, drag the *ne_100m_admin_0_countries.shp* file and drop it on the empty canvas in *QGIS*
4. At times, you may wish to zoom in or zoom out. When you wish to do so, select **Zoom In** or **Zoom Out** from the **View** menu (or select the **Zoom In** or **Zoom Out** icon from the tool bar). Then click the canvas in order to zoom in or out in accordance with the feature you selected. When finished zooming, select the **Pan Map** icon (hand in the tool bar) or select **Pan Map** from the **View** menu. If the entire map appears off screen due to zooming, you can bring the map back into view by selecting **Zoom Full** from the **View** menu
5. From the folder containing the Natural Earth data for states and provinces, drag the *ne_50m_admin_1_states_provinces.shp* file and drop it on top of the map of countries, which is currently displayed in *QGIS*
6. From the folder containing the Natural Earth data for populated places, drag the *ne_50m_populated_places.shp* file and drop it on top of the current map (of countries, provinces, and states) in *QGIS*
7. Notice the **Layers** window in the sidebar on the left. (If not visible for some reason, select **Panels**; **Layer**s from the **View** menu.) You can hide or display a layer by clicking the checkbox beside the name of the layer. Also, in the **Layers** window, you can re-order the layers by dragging a layer to a different position. For example, if you drag **ne_100m_admin_0_countries** above **ne_50m_admin_1_states_provinces**, the regional boundaries of the states and provinces will be hidden. After experimentation, readjust the layers to make the states and province boundaries visible
8. You may wish to change the default colors of the layers. For instance, to change the color of the countries, click the **ne_100m_admin_0_countries** layer. Then click the **Styling Panel** icon (left-most icon) in the **Layers** toolbar. (Alternatively, you could drop down the **Layer** menu and select **Layer Properties**; **Symbology**.). Beside the **Color** label, you may click on the color bar or click the triangle to the right of the color bar and then select a color. (You may try both methods in order to view the differences in the dialogs.) For the countries, you might select a gray (equal parts red, green, and blue). For instance, try 212 for each of red, green, and blue. If you want a slight tint to the gray, retain the 212 for red and blue, and set green to 195, for instance.
9. Now click the **ne_50m_admin_1_states_provinces** layer. Then drop down the **Layer** menu and select **Layer Properties …**. Click the **Symbology** tab and change the color to light yellow (e.g., Red: 250; Green: 250; Blue: 220). Of course, you may select any color you wish, including the same color you chose in the previous step for the countries layer, so as not to highlight Australia, Brazil, Canada, and the USA.
10. In this step, you direct *QGIS* to display the names of the states and provinces. With the **Layer Properties** window still open, click the **Labels** tab. The default is No Labels.

Click the No Labels drop down menu and select **Single Labels**. The field for provincial and state names is selected by default; and another sidebar appears. On the **Text** tab of the nested sidebar, you may change the font, as well as the font style, size, and color. You may leave the default settings or adjust them as you wish. When ready, click the **Formatting** tab. For type case, select All Uppercase. Click the **Apply** button at the bottom of the window and then close the **Layer Properties** window.

11. You may wish to change the color of the circle marker used to identify each populated place or add the names of the populated places. If so, click the layer for populated places. Then repeat Steps 8, 9, and 10, though for these labels you may not want to select the All Uppercase option.

At this point, you might be satisfied with your map. Save it and return to it as you wish. Alternatively, proceed with the following filtering steps in order to isolate a particular group of countries. Also, consider changing the projection.

1. Click the **ne_100m_admin_0_countries** layer. Then drop down the **Layer** menu and select **Filter ...**
2. To focus on Canada and the USA, enter (or build) the filtering criteria below. Use the **Test** button after entering or building the filtering criteria. Two rows should be returned. Click **OK** to remove the alert dialog displaying the number of rows returned. Lastly, click **OK** to close the **Query Builder**.
   ```
   "ADM0_A3" = 'CAN' OR "ADM0_A3" = 'USA'
   ```
3. At this point, populated places throughout the world are still visible, which draws attention away from the countries of focus in this example. To restrict the display of populated places to locations in Canada and the USA, click the ne_50m_populated_places layer. Then enter the following filtering criteria.
   ```
   ("ADM0_A3" = 'CAN' OR "ADM0_A3" = 'USA') AND "FEATURECLA" LIKE '%capital'
   ```
4. Brazil and Australia still appear. To display only Canada and the USA, click the ne_50m_admin_1_states_provinces layer and enter the following filtering criteria.
   ```
   "ADM0_A3" = 'CAN' OR "ADM0_A3" = 'USA'
   ```
5. With respect to projection, consider trying EPSG:3347, the Statistics Canada Lambert projection. Click the default projection for Natural Earth Data, EPSG:4326, which appears toward the lower-right corner of the *QGIS* window. A dialog pops up when you click EPSG:4326. In the Filter field of that dialog, enter 3347; select the projection returned by the filter and click OK.

Remember that you can open the Attribute Table (through the Layer menu, for instance) in order to inspect the field names and values. Since the mapping data contain multiple fields, there are often many options for filtering data. For example, in Step 2 above, countries can be selected using either country name, three-letter abbreviations, or two-letter abbreviations, for instance. Further, country names can be filtered using any one of multiple field names (e.g., `SOVEREIGNT`, `ADMIN`, `GEOUNIT`, `NAME`, `FORMAL_EN`). Similarly, one can create filters containing abbreviations with any one of multiple field names (e.g., `ADM0_A3`, `GU_A3`, `BRK_A3`, `ABBREV`, `POSTAL`, `ISO_A2`, `ISO_A3`, `ISO_A2`, `WB_A2`, `WB_A3`). When creating filtering criteria, note that field names and values are not case sensitive. As such, in Step 4 above, though not recommended due to the importance of consistency in use of field names and values, the following filtering criteria could have been used.

```
"adm0_a3" = 'CAn' OR "adm0_a3" = 'usa'
```

In the ne_50m_populated_places table, values in the `FEATURECLA` field include, for instance, `"Admin-0 capital"`, `"Admin-1 capital"`, and `"Populated place"`. The `Admin-0` level refers to countries whereas the `Admin-1` level refers to the first subunit in a country, which are states in the USA, and provinces and territories in Canada. In Step 3 above, the filtering criterion, `"FEATURECLA" LIKE '%capital',` uses the % symbol as a wild card in order to match all characters in values preceding the letters `capital`. In Step 3, the filtering criterion limits the places selected to capital cities, both country capitals (Ottawa and Washington, DC) and the capitals of U.S. states, as well as Canadian provinces and territories.

5.3.3 Creating a 3D Model of Earth

To create a three-dimensional model of Earth in *Blender*, we will wrap a two-dimensional map around a sphere. Select or create a map for the type of globe you want to create. You may want your globe to depict political regions, climate zones, typography, bathymetry, both typography and bathymetry, or something completely different. To begin, considering retrieving one or more of the following two-dimensional maps.

For a globe depicting both topography and bathymetry, the National Aeronautics and Space Administration (NASA) provides a 5400x2700 pixel image at this URL: https://visibleearth.nasa.gov/images/73801/september-blue-marble-next-generation-w-topography-and-bathymetry/73803l

NASA provides numerous alternative images of Earth in what they call the Blue Marble collection, which is available at this URL: https://visibleearth.nasa.gov/collection/1484/blue-marble

For an alternative view of topography and bathymetry, you may consider retrieving a 7964 x 3980 pixel map by Demis (https://www.demis.nl/about/introduction/), which was uploaded to Wikimedia commons at this URL: https://commons.wikimedia.org/wiki/File:WorldMap-A_non-Frame.png

For a globe displaying climate zones, consider retrieving a 1600 x 800 pixel map by Peel et al. (2007) as a png image from this URL: https://commons.wikimedia.org/wiki/File:K%C3%B6ppen-vereinfacht.svg

With the cursor over the image, right click (Windows) or Ctrl-click (Mac) to pop up a contextual menu and save the image in the Portable Network Graphics (png) format.

After creating or obtaining at least one of the images above, proceed through the following steps to create a globe.

1. Open *Blender*
2. Delete the cube
3. Click **Add**; **Mesh**; **UV Sphere**
4. Double the number of **Segments** and **Rings**
5. Switch to **Edit** mode
6. Click **Mesh**; **Shading**; **Smooth Faces**
7. Switch back to **Object** mode
8. For **Viewport Shading**, click the icon for either **Look Dev** or **Render**
9. Wrap your map around the globe by adding it as an image texture to a new material. Click the **Material** icon; then click **New**; **Base Color**; **Image Texture**; navigate to the image file you have created or selected; select **Open Image**

Your globe is ready for 3D printing. As described in Chapter 3 (Section 3.3.9), the model can be exported to the stereolithographic format by dropping down the File menu and selecting Export, Stl. Then submit the.stl file to a 3D printer.

5.3.4 Designing a Web App to Highlight Regions of a Country

In this section, we design a web app to help middle school students and others become aware of the primary administrative divisions of a country, which may be called provinces, states, regions, parishes, or cantons, for instance. The app will enable users to become aware of the relative positions of the administrative divisions within the country. In addition, the app will enable users to view the division's population, flag, and official flower.

Since learning the relative positions of administrative divisions is important, the app will display a map of the country, including visual boundaries of the primary administrative divisions. When the user positions the cursor over an administrative division and clicks (or taps), the app will highlight the administrative division by changing its color and, in a sidebar, will display the name, population, area, flag, and official flower of the division.

To depict the visual design of the app and to demonstrate its functionality, we will create a prototype in *Adobe XD*, which has a free version. After downloading and installing *Adobe XD*, proceed through the following three sets of steps to create the prototype.

A. Import GIS Data into *mapshaper* and Export to SVG
B. Insert the SVG Map into Adobe XD and Create Sidebar
C. Add a Tapped State to Administrative Divisions

A. Import GIS Data into Mapshaper and Export to SVG

1. Open a web browser to https://mapshaper.org/
2. In the **ne_50m_admin_1_states_provinces** folder, select *ne_50m_admin_1_states_provinces.dbf, ne_50m_admin_1_states_provinces.prj,* and *ne_50m_admin_1_states_provinces.shp*
3. Drag and drop the three selected files on the top portion of the browser window
4. Click the **Import** button

Notice that the data include primary administrative divisions for Australia (AUS), Brazil (BRA), Canada (CAN), and the United States of America (USA). Filter the data using the `adm0_a3` field and a three-letter code to focus on one country. For example, you would enter the following into the *Mapshaper* console in order to select Canada.

```
-filter 'adm0_a3=="CAN"'
-clean
```

The default projection used by Natural Earth (WGS84) may be retained for Brazil and Australia, but the distortion might be too much to tolerate for maps of Canada and the USA. In *Mapshaper*, there is a projection for the USA called *albersusa*, which preserves the relative sizes of the states, and positions Alaska and Hawaii underneath the 48 contiguous states. If you filtered the data for the USA, the albersusa projection can be applied by entering the following.

```
-proj albersusa
```

For Canada, you could select the common Mercator projection, if you wish, but the distortion for the Northern part of the country is large. You may want to consider each one of the following six projections.

```
-proj merc
-proj gall
```

```
-proj mill
-proj times
-proj epsg:6623
-proj epsg:3347
```

The last two entries demonstrate that *Mapshaper* accepts projections using EPSG codes.

In order to include the map into your app prototype, export the map by clicking the Export tab; selecting the SVG radio button, and entering the following into the export options field in order to include the name of each administrative division in the SVG file.

```
id-field=name
```

As noted in the *First Map Renderings* section above, use a text editor to insert `stroke="white" fill="gray"` into the SVG declaration; then save the file. You may wish to verify those changes to the appearance of the map by opening your SVG file in a web browser.

B. Insert the SVG Map into Adobe XD and Create Sidebar

1. Open *Adobe XD*
2. From the available form factors (devices), select Web (1920 x 1080 pixels).
3. Drag and drop the SVG file on the empty artboard
4. Click the rectangle tool in the sidebar
5. Position the cursor to the right of the map; then click and drag the cursor in order to create a rectangle as shown in Figure 5.1

C. Add a Tapped State to Administrative Divisions

1. Add a rectangle 300 pixels wide to the upper-right section of the artboard; make the rectangle the full length of the artboard; and retain the default white background

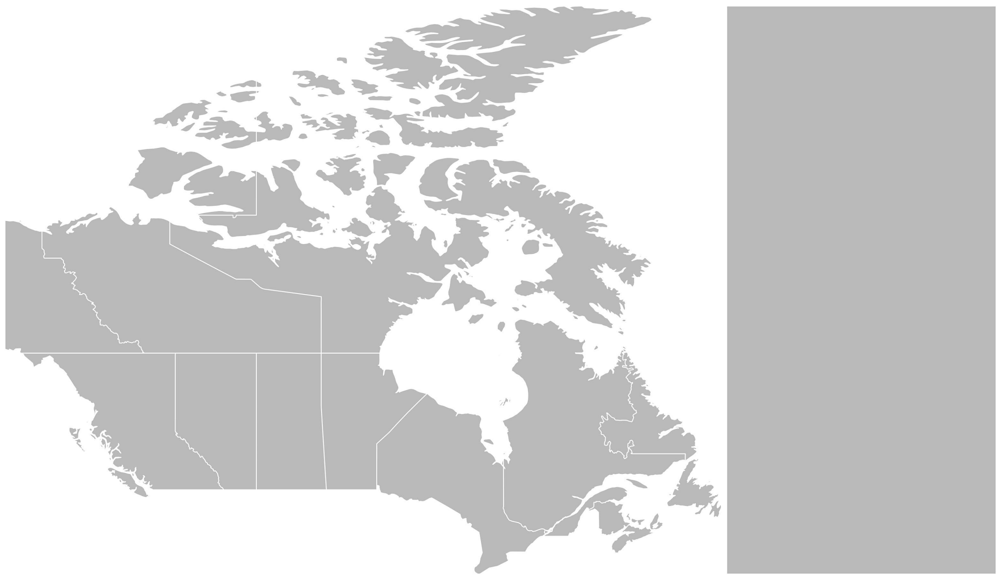


Figure 5.1 Visual design of app with map and sidebar

2. Copy and paste that rectangle to create a new rectangle; fill this rectangle with blue (#0057BB)
3. Add a text asset for the provincial title, British Columbia; fill color is gold (#FFCD00), arrange center and toward the top of the blue rectangle
4. Add a text asset, also in gold, for the population (approximately 5.1 million in 2021)
5. Add a text asset, also in gold, for the area, which is 929,730 square kilometers
6. Add a placeholder image for the flag of British Columbia
7. Add an image or a placeholder image for BC's official flower, the Pacific Dogwood
8. Double click the path for British Columbia
9. Select all assets
10. Drop down the Object menu; select Make Component (or use the Keyboard shortcut, Ctrl-K on Windows; Command-K on a Mac)
11. Add a new state under the Default state; name the new state, BC Tapped
12. Select the Default state and arrange assets to display the map with the white rectangle
13. Select the BC Tapped state and arrange the assets to show the layout in Figure 5.2
14. Click the Prototype tab; select the BC path in the map; set a trigger to self-animate to the BC Tapped state
15. Select the Default state; play the prototype; tap BC in the map to ensure that details for BC are displayed

You may repeat those steps for each province and territory (i.e., Alberta, Saskatchewan, Manitoba, Ontario, Quebec New Brunswick, Nova Scotia, Prince Edward Island, Newfoundland and Labrador, Yukon, Northwest Territories, Nunavut). You will need to ungroup the component in order to add more text assets and rectangles. Alternatively, you may repeat those steps for one or two more provinces in order to demonstrate the functionality across multiple provinces. Of course, too, you may add more information about the provinces. Adding the mottos of the provinces, which appear below, might increase interest among members of the target audience.

British Columbia: *Splendor sine occasu* (Splendor without diminishment)

Alberta: *Fortis et liber* (Strong and free)

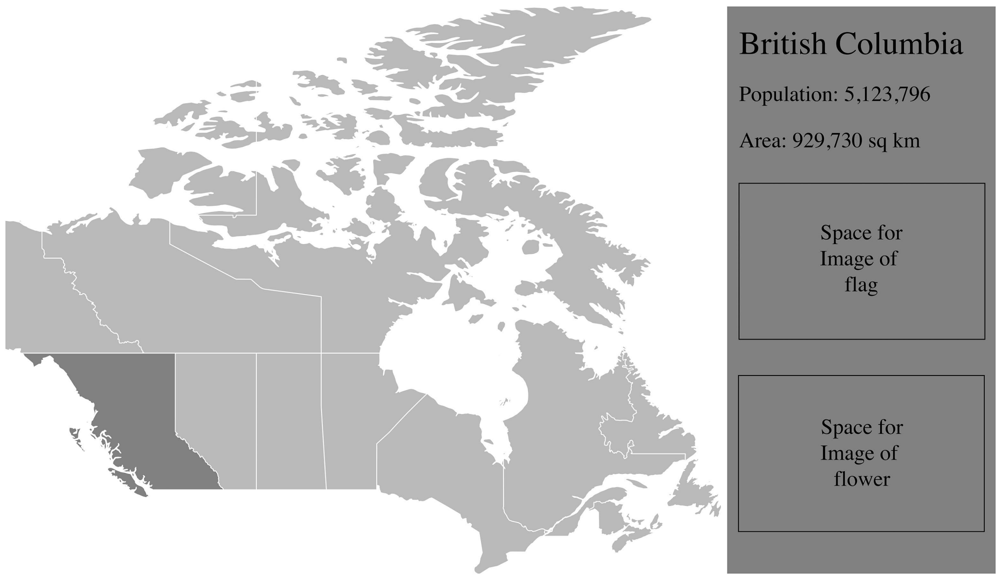


Figure 5.2 Visual design of app with map focused on British Columbia

Saskatchewan: *Multis e gentibus vires* (From many people's strength)
Manitoba: *Gloriosus et liber* (Glorious and free)
Ontario: *Ut incepit fidelis sic permanent* (Loyal she began and loyal she remains)
Quebec: *Je me souviens* (I remember)
New Brunswick: *Spem reduxit* (Hope restored)
Nova Scotia: *Munit haec et altera vincit* (One defends and the other conquers)
Prince Edward Island: *Parva sub ingenti* (The small under the protection of the great)
Newfoundland and Labrador: *Quaerite prime regnum dei* (Seek ye first the kingdom of God)
Nunavut: *Nunavut sanginivut* (Nunavut, our strength)
The Yukon Territory and the Northwest Territories do not have official mottos.

Consider including the capital city of each province (from west to east, Victoria, Edmonton, Regina, Winnipeg, Toronto, Quebec City, Fredericton, Halifax, Charlottetown, St. John's) and the capital city of each territory (also from west to east, Whitehorse, Yellowknife, Iqaluit). You may also wish to include information about the official animal, bird, or fish, for instance. You might even include curling as the official sport of Saskatchewan. See the annotated resources for links to additional information about Canada.

5.3.5 Developing the Web App to Highlight Regions of a Country

Consistent with the design in the previous section, we will create a web app that enables the user to click or tap various locations on the map in order to view details about the administrative division selected. We will develop the app in three parts. First, using CSS, we will create a web page that has two columns positioned horizontally. The column on the left will display the map while the column on the right will display details of the administrative division selected. Initially, we will simplify the SVG map by creating two rectangular paths for the regions. Second, we will make the app functional by adding an event listener that responds to user clicks within regions of the map. Regional details will appear in the column to the right of the map. Third, we will replace the simplified SVG with the SVG of the actual map, which was generated in the Export step above. We will also replace the phony regional data with actual data for the ten Canadian provinces and three territories.

Part 1: CSS for Side by Side Columns with Map on Left Side and Details on Right Side

```
<!DOCTYPE html>
<html>
<head>
<style>
    .map {
       float: left;
       padding: 10px;
    }

    .details {
       float: left;
       padding: 10px;
       color: black;
       background-color: gray;
       display: block;
    }
</style>
</head>
```

```
<body>
   <div class="map">
      <svg width="600" height="200" viewBox="0 0 600 200" stroke="white"
      fill="gray">
            <path d="M 100 0 500 0 500 100 100 100 Z" id="region1" />
            <path d="M 100 100 500 100 500 200 100 200 Z" id="region2" />
      </svg>
   </div>

   <div class="details" id="detailsDiv">
      Title of Region<br />
      Population<br />
      Image of Flag
   </div>
</body>
</html>
```

When you have verified that this web page displays the simplified map (in accordance with the two paths in the SVG tag) and the placeholder text for details, replace `block` with `none` in order to suppress the display of details initially. Regional details will appear only when the user selects a region of the map by clicking or tapping it.

Part 2: Add Event Listener to Implement Interactivity

To implement interactivity, first surround the map (the SVG tag) with a button tag.

```
      <div class="map">
      <button class="tapped">
         <svg width="600" height="200" viewBox="0 0 600 200" stroke="white"
         fill="gray">
         <path d="M 100 0 500 0 500 100 100 100 Z" id="region1" />
         <path d="M 100 100 500 100 500 200 100 200 Z" id="region2" />
         </svg>
      </button>
</div>
```

Then, as shown below, insert a little CSS to style the button, and add opening and closing script tags below the closing style tag. Within the script tags, include the initial `pathTapped` function and the JavaScript statement that adds the Event Listener.

```
<head>

<style>
   .map {
      float: left;
      padding: 10px;
   }

   .details {
      float: left;
```

```
        padding: 10px;
        color: black;
        background-color: gray;
        display: none;
    }
    .tapped {
        background-color: transparent;
        border: 0;
        padding: 0;
        outline: none;
    }
</style>

<script>
        function pathTapped(event) {
            if (event.target.closest('.tapped')) {
                event.target.style.fill = "red";
            }
        }
        document.addEventListener('click', pathTapped);
</script>

</head>
```

At this point, save your script and ensure that mouse clicks are detected. When you have entered the code above, each map region will turn red when clicked.

When a click is detected, the `pathTapped` function will be actuated (activated). If the tap occurs with the mouse cursor over a map path or finger on the map path, the if statement in the `pathTapped` function will evaluate to true. This will result in execution of the assignment statement, which will make the path tapped turn from gray (the default color) to red because the fill color, which is a style attribute, is set to red.

The present implementation of the `pathTapped` function serves well to test the event listener. While verifying that click detection is working properly, you will have noticed that subsequent clicks do not remove the red highlight from the path tapped previously. Let's take care of that issue first because multiple highlighted paths would confuse users.

First, we will add a global variable, called `prevTarget`, in order to keep track of the previous path tapped. Each time a path is clicked, `prevTarget` will be set to the current path, which is stored in `event.target`. Update your script as shown below; test it; and debug as necessary to ensure that each time a path is selected, the previous path returns to gray, which is the default appearance of all unselected paths.

```
<script>
        var prevTarget = null;

        function pathTapped(event) {
            if (event.target.closest('.tapped')) {
                if (prevTarget!= null) {
                        prevTarget.style.fill = "gray";
                }
                event.target.style.fill = "red";
                prevTarget = event.target;
            }
        }

        document.addEventListener('click', pathTapped);
</script>
```

Now, the one remaining feature to implement is the display of region details in the space to the right of the map. First, we will create a list of objects to store the region name, population, area, as well as a link to an image of the region's flag and a link to an image of the region's official flower. In addition, we will store foreground and background colors.

To create the list of objects, insert the following variable declaration after the declaration of prevTarget.

```
var regionInfo = {"Region 1" : {population: "123,456", area: "789 sq km",
uriFlag: "flag1.png", uriFlower: "flower1.jpg", foregroundColor: "black",
backgroundColor: "yellow"}, "Region 2" : {population: "345", area: "1,000 sq
km", uriFlag: "flag2.jpg", uriFlower: "flower2.png", foregroundColor: "white",
backgroundColor: "#0000FF"}};
```

Second, delete the placeholder text from `detailsDiv`. That is, delete the following.

```
Title of Region<br />
Population<br />
Image of Flag
```

Third, insert HTML tags into the `detailsDiv` in order to display the region name, population, area, flag, and flower, as shown below.

```
    <div class="details" id="detailsDiv">
    <h2 id="regionName">Bogus Name</h2>
    <h3 id="population">Bogus Population</h3>
    <h3 id="area">Bogus Area</h3>
    <img id="uriFlag" src="" />
    <br />
    <img id="uriFlower" src="" />
</div>
```

Fourth, update the `pathTapped` function to display the region details, which involves setting the display to "block" (thereby making the region details visible); setting the foreground and background colors; and setting the values for region name, population, area, flag, and flower.

```
<script>
    var prevTarget = null;

    var regionInfo = {"Region 1" : {population: "123,456", area: "789 sq
km", uriFlag: "flag1.png", uriFlower: "flower1.jpg", foregroundColor: "black",
backgroundColor: "yellow"}, "Region 2" : {population: "345", area: "1,000 sq
km", uriFlag: "flag2.jpg", uriFlower: "flower2.png", foregroundColor: "white",
backgroundColor: "#0000FF"}};
    function pathTapped(event) {
        if (event.target.closest('.tapped')) {
            if (prevTarget!= null) {
                prevTarget.style.fill = "gray";
        }
            event.target.style.fill = "red";
            document.getElementById("detailsDiv").style.display = "block";
            document.getElementById("detailsDiv").style.color =
regionInfo[event.target.id].foregroundColor;
            document.getElementById("detailsDiv").style.backgroundColor =
regionInfo[event.target.id].backgroundColor;
            document.getElementById("regionName").innerHTML = event.target.id;
            document.getElementById("population").innerHTML = "Population: " +
regionInfo[event.target.id].population;
            document.getElementById("area").innerHTML = "Area: " +
regionInfo[event.target.id].area;
            document.getElementById("uriFlag").src  =  regionInfo[event.target.
id].uriFlag;
            document.getElementById("uriFlower").src =
regionInfo[event.target.id].uriFlower;
            prevTarget = event.target;
        }
    }
    document.addEventListener('click', pathTapped);
</script>
```

Part 3: Fully Functional App

Replace the bogus regional data with the following.

```
var regionInfo = {"British Columbia" : {population: "5123796", area: "929,730
sq  km",  uriFlag:  "https://www.canada.ca/content/dam/pch/images/services/
provincial-territorial-symbols-canada/british-columbia/drapeau_c_b-b_c_flag.
jpg",  uriFlower:  "https://www.canada.ca/content/dam/pch/images/services/
provincial-territorial-symbols-canada/british-columbia/cornouiller_nuttall_
c_b-b_c_pacific_dogwood.jpg",  foregroundColor:  "#FFCD00",  backgroundColor:
"#0047BB"}};
```

Also, replace the bogus SVG map with the SVG exported in the section titled, *Import GIS Data into Mapshaper and Export to SVG*.

Debug as necessary to ensure functionality. This app is available in the *CanadianProvincesApp.html* file in the Electronic Resources for the book.

5.4 Challenges and Pursuits

1. To gain an overview of the field of cartography, read "Mapping the World" by William Cartwright and Anne Ruas (link to article appears below). Cartwright and Ruas are the founding editors of the *International Journal of Cartography*, and in "Mapping the World" they convey their feelings about the importance of mapping. They begin by questioning whether anything could be more important than mapping. Then they dream about what human beings will make of the world. Subsequently, they focus on matters cartographers actually address, such as map projections, symbols and signs (semiotics), and map design. As described by Cartwright and Ruas, topics pertinent to cartographers are informed by spatial analysis and visual analytics.

Further, in addition to technical matters, such as data features, ontologies, and formats, cartographers reflect on history, art, and education. Lastly, Cartwright and Ruas introduce the eight papers in the first issue of the *International Journal of Cartography*, which were presented at the 2015 International Cartographic Conference in Rio de Janeiro, Brazil. The diversity of the articles is noteworthy; the papers discuss the following topics: (1) Use of cartograms in school cartography; (2) Use of machine learning algorithms to automate the identification of buildings based on data in topographic databases; (3) Determining projection parameters when the radius of the underlying sphere of the map is unknown; (4) Usability of a digital support tool for hand-drawn maps; (5) Influence of color on human performance with choropleth and chorochromatic maps; (6) Use of simulations and visualizations to inform designers of accessible wayfinding technologies for people with disabilities; (7) Emergence of irrigation mapping through analyses of water estates in Taiwan from 1901–1921; and (8) Recognition of place names (toponyms) through data parsing of geographical dictionaries (gazetteers). https://www.tandfonline.com/doi/pdf/10.1080/23729333.2015.1062608

2. Browse the *Free GIS Data* resource at http://freegisdata.rtwilson.com/. Determine whether any data were produced by your country. Also, consider which datasets would be helpful to your students.
3. Retrieve data from the *Database of Global Administrative Areas* (GADM) (https://gadm.org/download_world.html), which enables map generation of all countries and their subdivisions. You have the option to retrieve the data per country (https://gadm.org/download_country_v3.html), but to gain insight into the enormity of the project, download the data as a single database in the GeoPackage format (https://biogeo.ucdavis.edu/data/gadm3.6/gadm36_gpkg.zip). After unzipping the file, you can drop the gadm36.gpkg file into *QGIS*. Alternatively, if you download the data as a compressed Shapefile (https://biogeo.ucdavis.edu/data/gadm3.6/gadm36_shp.zip), you can unzip the file and drop the *gadm36.dbf, gadm36.prj,* and *gadm36.shp* files into *Mapshaper*. Filter the data to focus on a particular region of the world, one or more specific countries or subdivisions within a country of particular interest to you. Focusing on a particular region, country, or subdivision will reduce the amount of data, which will increase processing speed. The alternative to filtering is to retrieve the data for each country. Consider which datasets would be helpful to your students.
4. Select any GIS dataset you have acquired, and then color subdivisions to make a choropleth based on a numeric marker, such as population. For example, the greater the population, the darker the shade of blue (red, green, or any color you wish) for the subdivision. If you use *QGIS*, you can alter colors directly. If you use *Mapshaper*, export the data to SVG and change the subdivision colors in *Inkscape*.
5. Using any SVG dataset you have acquired, create a web app that automates choropleth coloring in accordance with a numeric marker of your choosing (e.g., births per 1000 population, income per capita, hours worked per week by country).
6. GEOData from the United Nations contains free geographic data and images on multiple topics. For this exploration, consider data from the Global Human Settlement project of the European Commission (https://ghsl.jrc.ec.europa.eu/)

   a. Open a web browser to https://datacore-gn.unepgrid.ch/geonetwork//srv/eng/catalog.search;jsessionid=3FD1A35F6A0389D6E61AD45A13DC4483#/home
   b. Click the **Society** button
   c. Browse the various descriptions of the datasets on population density, population distribution, settlements, and protected lands
   d. Click Population Distribution – 250m (GHSL - JRC)
   e. Click the **Open Link** button for Population Distribution (GHSL – JRC)
   f. Click the **V1-0/** link
   g. Click the GHS_POP_GPW42015_GLOBE_R2015A_54009_250_v1_0.zip link

- h. Once the.zip file has downloaded, unzip it to produce the data folder.
- i. Drop the GHS_POP_GPW42015_GLOBE_R2015A_54009_1k_v1_0 file into *QGIS*
- j. Open the Layer Styling panel and, using the dropdown box, replace Singleband gray with Multiband color.
- k. Behold the distribution of the world population

Alternatively, the data can be obtained from the following URL.

http://cidportal.jrc.ec.europa.eu/ftp/jrc-opendata/GHSL/GHS_POP_GPW4_GLOBE_R2015A/GHS_POP_GPW42015_GLOBE_R2015A_54009_1k/V1-0/GHS_POP_GPW42015_GLOBE_R2015A_54009_1k_v1_0.zip

7. To make all provinces and territories in the *Canadian Provinces App* clickable, complete the `regionInfo` data structure in *CanadianProvicesApp.html* (which is available in the Electronic Resources for the book). Content pertaining to Canadian provinces may be located through the websites on Canada in the Annotated Resources below.
8. Extend the design of the mapping app to make it a game. For example, the app might include two text labels at the top of the screen in order to display the name of an administrative division (e.g., state, province) of a country in one label and the user's score in the other label. The division name could be left-justified and the score right-justified. After viewing the name of the administrative region, the user would click a place on the map and the app would tally the number of times the user tapped the correct administrative region.
9. Create the app designed in the previous step. You may wish to add a sound effect for a correct answer.
10. You have gained familiarity with multiple sources of free geographic data by progressing through this chapter, including Challenges 2, 3, and 6. In addition, you may have initiated your own searches for geographic data using a web search engine. Retrieve whatever geographic data interest you and enjoy creating unique maps. Export the maps you created to SVG and develop unique apps.

5.5 Annotated Resources

Why All World Maps are Wrong

https://www.youtube.com/watch?v=kIID5FDi2JQ

This six-minute video describes the historic significance of the Mercator projection and provides a dynamic simulation of points on Earth projected on to the surface of a cylinder. This video also includes key historic notes, such as the mathematical proof by Carl Friedrich Gauss in 1827 on the distortion of projections. Importantly, a distortion can be acceptable when its advantages and disadvantages are evident. As noted in the video, though the Mercator projection depicts Greenland as 14 times its actual size, the projection preserves shape and direction quite well, which enabled navigation of ships across seas.

Mercator's Projection

https://www.math.ubc.ca/~israel/m103/mercator/mercator.html

On this web page, Robert Israel describes the trigonometry used by Edward Wright in 1599 to explain the Mercator projection. The explanation requires calculation of a limit in order to account for the stretching of lines of constant latitude (parallels). In the Mercator projection, all parallels (except the equator) are stretched in order to make them equal in length to the equator. As Israel notes, given the mathematics of the day, Wright approximated that limit using sums. Israel then describes how the introduction of logarithms and calculus have changed the mathematics used to explain the Mercator projection.

Map Projections

Knowledge of the distortion effects of map projections enables one to make informed interpretations of maps. You may wish to pursue any of the following sources to learn more about map projections.

Cartography & Visualization

(CV) 06 – Map Projections

Battersby, S. (2017). Map Projections. The Geographic Information Science & Technology Body of Knowledge (2nd Quarter 2017 Edition), John P. Wilson (ed.)

https://gistbok.ucgis.org/bok-topics/map-projections

In the article at the link above, Battersby defines and exemplifies map projection, and then succinctly describes construction of projections, distortion effects, and projection selection.

Understanding Map Projections

http://downloads2.esri.com/support/documentation/ao_/710Understanding_Map_Projections.pdf

This is a comprehensive book on cartography by the Environmental Systems Research Institute (1994–2000). Projection types are on Pages 13–19.

Designing Across Map Use Contexts: A Research Agenda

Amy L. Griffin, Travis White, Carolyn Fish, Beate Tomio, Haosheng Huang, Claudia Robbi Sluter, João Vitor Meza Bravo, Sara I. Fabrikant, Susanne Bleisch, Melissa Yamada, & Péricles Picanço (2017). *International Journal of Cartography, 3*(1).

https://www.tandfonline.com/doi/full/10.1080/23729333.2017.1315988

This scholarly work discusses numerous issues pertaining to the design of maps, such as map use, map context, individual differences of target users, spatial abilities, disability, emotional responses, and privacy.

EPSG and WGS Codes

For a very brief overview of EPSG codes, consider the following web page.

https://support.virtual-surveyor.com/en/support/solutions/articles/1000261353-what-is-an-epsg-code#:~:text=EPSG%20stands%20for%20European%20Petroleum,spheroids%2C%20units%20and%20such%20alike

For a brief and insightful overview of the WGS84 geodetic standard, consider the following web page.

https://support.virtual-surveyor.com/en/support/solutions/articles/1000261351-what-is-wgs84

Free GIS Data

http://freegisdata.rtwilson.com/

This web page contains links to over 500 websites from which one can obtain GIS data free of cost. The links are organized into three primary categories, namely Physical Geography (general, land and ocean boundaries, elevation, weather and climate, hydrology, snow/ice, natural disasters, land cover, ecology, mineral resources/oil and gas), Human Geography (general, administrative boundaries, environmental boundaries, land use; lakes, oceans, and other water sources; wars, conflict and crime; population; buildings, roads and points of interest; transport and communications; gazetteers-place/feature names; miscellaneous), and Datasets for Individual Countries (approximately 45 countries are included in this section). Most websites can be accessed freely; a suitable label also appears when registration is required to access the site. A suitable label also appears when a language other than English is necessary to comprehend the site.

Natural Earth
https://www.naturalearthdata.com/

This website has organized datasets into three categories (called themes on the website), cultural, physical, and raster (bitmapped). Datasets for maps of countries, including regional boundaries within countries and major cities and towns, appear in the Cultural category. Datasets in the Physical category pertain to entities such as rivers, lakes, oceans, bathymetry (ocean depths), polar circles, and tropical circles. The Raster category contains datasets to depict land surfaces and ocean bottoms, for instance. Since GIS data files can be very large, some datasets come in two or three levels of detail in order to balance detail with file size. The level of detail is clearly marked with color codes and numbers, 10 for the most detailed and largest file size, 50 for moderate detail and file size, 110 for course detail and smallest file size.

Datasets at this website are also available at the GitHub site.

https://gist.github.com/DanielJWood/b71237cc200831acf8e637c05ce2c375#file-natural_earth_s3_links-md

International Cartographic Association (ICA)
https://icaci.org/

The ICA seeks to promote cartography and geographic information science through a variety of methods, including multinational research and knowledge transfer. With over 70 national members (https://icaci.org/national-members/), the ICA sponsors conferences, workshops, and edits the *International Journal of Cartography*, which is a relatively young journal, publishing its first issue in 2015. Current and archived issues are available through Taylor & Francis Online (https://www.tandfonline.com/loi/tica20#.VFfqb_mG98G).

Mapshaper
https://mapshaper.org/

This web app renders GIS data and permits editing. There is no registration requirement and there are no popups, no legal notifications, and no advertisements; just drag and drop your data on the browser window. Documentation is available through the Wiki link. Open-source code is available through GitHub.

QGIS
https://www.qgis.org/en/site/

QGIS is a free and open-source Geographic Information System (GIS).

The *QGIS* website includes four tabs: (1) *Discover QGIS* enables one to become aware of the features in *QGIS*; (2). *For Users* enables one to get started with *QGIS*, which includes a **Get the installer** button to download the software (no registration necessary); (3) *Get Involved* provides a link to report bugs and provides information about participating in chats and forums, as well as invites participation in programming, translating, or writing documentation. A link about supporting *QGIS* financially also appears on the web page for this tab; and (4) The *Documentation* tab facilitates access to the User Guide, the Training Manual, and to a page called A Gentle Introduction to Geographic Information Systems, which is a rather extensive introduction.

QGIS is a large program with hundreds of features, including a Python programming console.

Downloading and installing *QGIS* is straightforward but takes some time (perhaps 5–10 minutes, depending on the speed of your Internet connection and computer processor). The size of the installer file is well over 500 MB.

The Q in *QGIS* is a remnant of a prior version of the software; the Q stood for Quantum.

A Guide to Making SVG Maps with *Mapshaper*
https://simplemaps.com/resources/guide-to-mapshaper

This website presents a tutorial on rendering GIS data in *Mapshaper*. The example pursued in the tutorial pertains to regions of the United States of America.

International Organization for Standardization (ISO) List of Country Codes
https://www.iso.org/obp/ui/#search/code/

This page provides a list of ISO 3166-1 country names, in English and French, along with their two-letter, three-letter, and numeric codes.

Inkscape
https://www.iso.org/obp/ui/#search

The *Inkscape* website includes a DOWNLOAD tab from which you can obtain, without registration, the current version of the software for your computer platform. In addition, the LEARN tab contains links to tutorials, such as the basic tutorial at https://inkscape.org/doc/tutorials/basic/tutorial-basic.html

The **Fill and stroke** section of the basic tutorial pertains directly to the task described in this chapter, which involves changing both the stroke and fill colors of the world map in the SVG file.

The Provinces and Territories of Canada
http://publications.gc.ca/collections/collection_2016/pch/S2-211-2002-1-eng.pdf

This is a foldout chart with information and images for each province and territory of Canada. In addition to the capital city, and the year the province or territory joined confederation, the chart includes a landmark image, and images of the flag, coat of arms, official flower, and official bird of each province and territory.

Provincial and Territorial Symbols
https://www.canada.ca/en/canadian-heritage/services/provincial-territorial-symbols-canada/british-columbia.html

This web page displays the symbols of British Columbia. The sidebar on the left of the page enables one-click navigation to the other Canadian provinces and territories. In addition to the symbols of each province and territory, there is a brief discussion about the origin of the name and a brief history of the region.

British Columbia (B.C.) Symbols
https://www2.gov.bc.ca/gov/content/governments/celebrating-british-columbia/symbols-of-bc

This web page, produced by the Government of British Columbia, contains images of the B.C. Coat of Arms, Shield, Great Seal, and Flag, as well as images of the official flower, bird, mammal, gemstone, tree, and tartan. In 2013, the B.C. government declared an official fish, the Pacific Salmon, but did not decide upon an image to represent the multiple species of Pacific Salmon, including Chinook, Chum, Coho, Pink, and Sockeye. This web page also identifies two types of trout as belonging to the Pacific Salmon category, but this is controversial (see, this article, for instance: *How many species of salmon are there and how large can they get?* https://www.usgs.gov/faqs/how-many-species-salmon-are-there-and-how-large-can-they-get?qt-news_science_products=0#qt-news_science_products).

References

Battersby, S. (2017). Map projections. In *The Geographic Information Science & Technology body of knowledge* (2nd Quarter 2017 ed.), J. P. Wilson (ed.). doi: 10.22224/gistbok/2017.2.7

Cartwright, W., & Ruas, A. (2015). Mapping the world. *International Journal of Cartography,* 1(1), 1–4. https://doi.org/10.1080/23729333.2015.1062608

Department for Education. (2014a). *National curriculum in England: Framework for key stages 1 to 4.* Government of the United Kingdom (gov.uk). https://www.gov.uk/government/publications/national-curriculum-in-england-framework-for-key-stages-1-to-4/the-national-curriculum-in-england-framework-for-key-stages-1-to-4

Department for Education. (2014b). *National curriculum in England: Framework document.* Government of the United Kingdom (gov.uk). https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/381344/Master_final_national_curriculum_28_Nov.pdf

Griffin, A. L., White, T., Fish, C., Tomio, B., Huang, H., Robbi Sluter, C., Meza Bravo, J. V., Fabrikant, S. I., Bleisch, S., Yamada, M. & Picanço, M. (2017). Designing across map use contexts: A research agenda. *International Journal of Cartography, 3*(1), 90–114. https://www.tandfonline.com/doi/full/10.1080/23729333.2017.1315988

Harris, J. (2016). *Why all maps are wrong.* YouTube. https://www.youtube.com/watch?v=kIID5FDi2JQ

Israel, R. (2003). *Mercator's projection.* University of British Columbia. https://www.math.ubc.ca/~israel/m103/mercator/mercator.html

Ministry of Education. (2016). *Grades K–9 Learning Standards: Social Studies.* Province of British Columbia. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/social-studies/en_social-studies_k-9_elab.pdf

Ministry of Education. (2018a). *Grade 10 Social Studies: Canada and the world, 1914 to the present.* Province of British Columbia. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/social-studies/en_social-studies_10_elab.pdf

Ministry of Education. (2018b). *Grade 11: Explorations in Social Studies.* Province of British Columbia. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/social-studies/en_social-studies_11_explorations-in-social-studies_elab.pdf

Ministry of Education. (2018c). *Grade 12 Social Studies: Human geography.* Province of British Columbia. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/social-studies/en_social-studies_12_human-geography_elab.pdf

North Carolina State Board of Education and Department of Public Instruction. (2010a). *North Carolina Essential Standards: Social Studies – American History Course I.* https://files.nc.gov/dpi/documents/curriculum/socialstudies/scos/american-history-1.pdf

North Carolina State Board of Education and Department of Public Instruction. (2010b). *North Carolina Essential Standards: Social Studies – American History Course II.* https://files.nc.gov/dpi/documents/curriculum/socialstudies/scos/american-history-2.pdf

North Carolina State Board of Education and Department of Public Instruction. (2010c). *North Carolina Essential Standards: Social Studies – World History Course.* https://files.nc.gov/dpi/documents/curriculum/socialstudies/scos/world.pdf

North Carolina State Board of Education and Department of Public Instruction. (2012a). *North Carolina Essential Standards: Sixth grade Social Studies.* https://files.nc.gov/dpi/documents/curriculum/socialstudies/scos/6.pdf

North Carolina State Board of Education and Department of Public Instruction. (2012b). *North Carolina Essential Standards: Seventh grade Social Studies.* https://files.nc.gov/dpi/documents/curriculum/socialstudies/scos/7.pdf

North Carolina State Board of Education and Department of Public Instruction. (2012c). *North Carolina Essential Standards: Eighth grade Social Studies.* https://files.nc.gov/dpi/documents/curriculum/socialstudies/scos/8.pdf

Peel, M. C., Finlayson, B. L., & McMahon, T. A. (2007). Updated world map of the Köppen-Geiger climate classification. *Hydrology and Earth Systems Science, 11,* 1633–1644. https://doi.org/10.5194/hess-11-1633-2007

Routley, N. (2017, June). The problem with our maps. *Business Insider.* https://www.businessinsider.com/the-mercator-projection-distorts-countries-2017-6

6 English

Ruminating Over Words

6.1 Theme

Thankfully, we have the fruits of those who have labored over words. More specifically, we have poets, lyricists, playwrights, novelists, essayists, historians, and philosophers to thank for thinking deeply about words. Teachers, instructional designers, scholars, lawyers, and newspaper columnists also ruminate over words at times. We should thank them, as well as other authors, but there is a big difference between a legal brief, a scholarly journal article, a newspaper column, and the work of literary artists. I am sure you could create a long list of such artists, which might include Geoffrey Chaucer (1343), William Shakespeare (1564), John Milton (1608), William Wordsworth (1770), John Keats (1795), Elizabeth Barrett Browning (1806), Walt Whitman (1819), Emily Dickinson (1830), Rudyard Kipling (1865), Federico García Lorca (1898), Maya Angelou (1928), Sylvia Plath (1932), Leonard Cohen (1934), Margaret Atwood (1939), and Amanda Gorman (1998), for instance. Many people today, and in the past, have greatly enjoyed the poems, lyrics, essays, short stories, and novels resulting from their ruminations over words, and in the future many more will relish their works of art, unique combinations of words that engender lasting memories.

In some cases, individual words stand alone, but many times they are building blocks to phrases, sentences, and larger works, such as verses, paragraphs, poems, essays, and novels. Margaret Atwood captured this in her poem *Spelling*: “A word after a word after a word is power.” For Leonard Cohen, words bring light: “There’s a blaze of light in every word.” You may recognize that line from the lyrics of Cohen’s most popular song, *Hallelujah*. There are numerous versions of that song, and the verse in which that line appears is not always present, but Cohen’s original version includes it. As you may imagine, Cohen ruminated over many words when writing the lyrics to *Hallelujah*. According to Alan Light (2012), Cohen wrote 80 verses for the song, selecting only four of them for his original recording in 1984. The words and music of *Hallelujah* are so intriguing that approximately 300 artists have followed Cohen in recording the song (Light, 2012; Partridge, 2019), yet the tune was slow to gain popularity initially (Light, 2012).

When we sequence words in accordance with particular grammatical rules, we comprise phrases and sentences in a language. Linguists dedicate their professional lives to the study of language. Speaking generally, a language is a system of communication, which human beings often use to great advantage, though we must admit that some communications are confusing. In such cases, when the stakes are perceived as low, we just move on. In contrast, when the stakes are perceived as high, disagreement ensues, and resolution is uncertain. The time needed to resolve ambiguities in language depends in part on whether the language is spoken or written. Human beings capable of hearing and speaking learn spoken language quite naturally, but learning the written form of a language takes considerably more mental effort. Spoken and written forms of communication also have huge implications for computer processing of language (Dickinson

DOI: 10.4324/9781003172284-9

et al., 2013; Gudivada & Rao, 2018), but we will not delve into computational linguistics here. We will briefly consider some differences in written languages and become acquainted with some of the branches of linguistics, as well as a few linguists.

Written languages may be alphabetic, syllabic, or logographic. Both alphabetic and syllabic systems map characters to sounds, whereas in logographic writing systems, each logograph (symbol) represents a unit of meaning. A true logographic system is rare today. Chinese, in many cases, uses logographs in combination with phonetic information. Returning to alphabetic systems, there are essentially two types. Phonemic alphabets produce all sounds with their letters, either individually or in combination, as in English and Spanish. In contrast, abjads or consonant alphabets, such as Arabic, Aramaic, and Hebrew, contain only consonants. Vowels are generally inferred from context. Alphabetic, syllabic, and logographic languages may be written and read from left to right, which you are doing now, but if you were reading or writing Hebrew, you could go right to left. We could also go top to bottom or bottom to top, or even agree to alternate directions, as in the boustrophedon style, writing left to right on one line and right to left on the next. Apparently, there are some ancient Etruscan texts in that style.

Returning to English, as you know, the alphabet has a set of 26 letters, one symbol per letter. Spanish matches those 26 letters and adds *ch*, *ll*, *ñ*, and *rr* to the alphabet. Spanish is remarkably consistent in the pronunciation of letter combinations, which form syllables, the building blocks of all languages. Even though Spanish letters may contain accents (technically ligatures), such as é and í, consistency of pronunciation remains. In contrast, English has many exceptions to the pronunciation of syllables. For example, *off* rhymes with *scoff* and *trough*. In English, then, the *gh* letter combination can sound like f. Yet, *trough* does not rhyme with *through*, which makes the *gh* letter combination silent at times. Further, in English, the f sound can also be produced by *ph*, as in *phone*. Further, *phone* (fōn) rhymes with *Dijon* (arguably), but certainly with *own* (ōn), *groan* (grōn,), and *bemoan* (optionally bē-mōn or bi-mōn). In summary, there are multiple spellings for the same sound in English. This yields many homophones, words with different spellings that are pronounced precisely the same, such as *no* and *know*, *complement* and *compliment*, *sweet* and *suite*, and *sight*, *site*, and *cite*. Another aspect of homonymy is the set of homographs, which contains words with the same spelling but different meanings, such as right, ring, rock, match, mean, and row (argument), row (align objects), and row (to move a boat). In addition, a letter in English may be silent in certain letter combinations, as is the k in *knot* and *knock*. Pronunciation of words in English is nontrivial.

Study of word formation or word structure is the component of linguistics called morphology. Within that subfield of linguistics, one might seek to identify the number of homophones in a language, or the number of letter combinations in a language that sound the same. To resolve questions about speech sounds, consider phonetics and phonology, though such inquiry might raise questions about prosody, phonemes, and allophones, for instance. Perhaps the better-known subfields of linguistics are semantics, which pertains to word meanings, and syntax, which considers sentence structure. When Robert Berwick considered syntax in 1985, he presented a computational model of language acquisition, which learns the rules of English syntax given grammatically correct sentences (Berwick, 1985). In 2011, Berwick was still reflecting on syntax, and with a group of collaborators, discussed the remarkable behavioral and neural similarities between auditory-vocal learning in birds and babies. This was done to inform studies of the brain and cognitive evolution. You may find comfort in knowing that birdsong syntax is nothing compared to the sophistication of human syntax (Berwick et al., 2011). In the linguistics subfield known as *pragmatics*, which is also a subfield of semiotics, researchers study how context influences meaning. Some linguists study language and culture (sociolinguistics), neurology and linguistics (neurolinguistics), and evolution and linguistics, for instance. Further, some linguists study languages in a particular region, such as South America (Adelaar, 2004; Dixon & Aikhenvald, 2008), as well as language families (e.g., Afroasiatic, Dravidian, and Indo-European). Categorizing languages is helpful in order to bring some manner of organization to the approximately 6500 languages in the world today. Of course, some languages are spoken by very few people, such as Busuu, which is known to eight people, while other languages, such as English, Mandarin Chinese, Hindi, Spanish, French, Arabic, Bangla/Bengali, Russian, Portuguese, Indonesian, Urdu, and German, are used by several million people (Klappenbach, 2021).

Some specialists apply linguistics to language translation and to language acquisition, as well as in clinics to discern speech pathology, and in legal matters, such as criminal investigations and in court proceedings.

For an introduction to linguistics, you may peruse or pursue the book *For the Love of Language* (Burridge & Stebbins, 2016). In addition, you may consider publications at the Linguistic Society of America (https://www.linguisticsociety.org/), the Canadian Linguistic Association (https://cla-acl.ca/index.html), and the Linguistic Association of Great Britain and Northern Ireland (http://www.lagb.org.uk/), for instance. You may also consider the work of some or all of the following distinguished linguists.

Pāṇini	Somewhere between the 7th and 4th centuries BCE, Pāṇini presented a grammar with generative rules that defined classic Sanskrit. The grammar contains generative rules akin to context-free grammars today.
Ferdinand de Saussure	Swiss linguist (born in 1857) who established tenets of structural linguistics and was a major contributor to the founding of semiotics
Edward Sapir	A linguist and anthropologist (born in 1884) who uniquely established relationships between language and culture
Leonard Bloomfield	Born in 1887, advanced linguistics as a science by establishing structures and elements of language, especially phonological processes through observation
Noam Chomsky	Prolific scholar (born in December 1928) who published such influential work that some regard him as the founder of modern linguistics
Eve V. Clark	Prolific scholar with unique contributions to knowledge and the study of first language acquisition
Robin Lakoff	Prolific scholar with unique contributions to knowledge and the study of language and gender
Kathryn Burridge	Prolific scholar with unique contributions to knowledge and the study of contextual uses of language, as well as contributions to the history of English language use

Certainly, there are many other noteworthy linguists to add to that list, such as Dan Jurafsky, James H. Martin, Julia Hirschberg, Regina Barzilay, Robert Berwick, James R. Martin, Daniel Klein, and Martha Stone Palmer. You may also want to add to your list of linguists more scholars who study language in combination with other disciplines, such as cognition and philosophy (e.g., George Lakoff and Ludwig Wittgenstein).

6.2 Sample Learning Goals

Beginning in the first year of schooling and continuing throughout elementary school, teachers instruct students to listen, speak, read, and write. The curriculum guides for England and British Columbia, Canada, as well as the U.S. Common Core standards make such instruction mandatory. With respect to secondary school, sample learning standards pertaining to English language knowledge, skills, and attitudes appear below for adolescents in England; British Columbia, and approximately 40 U.S. states.

England

Department for Education (2014)
2014–Present
National Curriculum in England: Framework document
The sample learning standards below have been copied from the curriculum guides used in England; British Columbia, Canada; and the USA, except where learning standards have been merged for this presentation, as indicated.

Reading

- read, understand and critically evaluate texts (Years 7–11, merged, p. 83, p. 86)
- develop an appreciation and love of reading, and read increasingly challenging material independently (Years 7–9, p. 83)
- read and appreciate the depth and power of the English literary heritage (Years 10–11, p. 86)

Writing

- apply their growing knowledge of vocabulary, grammar and text structure, gained through reading and listening, to write accurately, fluently, effectively and at length for pleasure and information for a wide range of purposes and audiences (Years 7–9, merged, pp. 83–84)
- make judicious use of vocabulary, grammar, form, and structural and organisational features, including rhetorical devices, gained through reading and listening, to write accurately, fluently, effectively and at length for pleasure and information for a wide range of purposes and audiences in order to describe, narrate, explain, instruct, give and respond to information, and argue (Years 10–11, merged, pp. 86–87)
- plan, draft, edit and proof-read (Years 7–9, p. 84)
- make notes, draft and write, revise, edit and proof-read (Years 10–11, merged, p. 87)

Vocabulary and Grammar

- consolidate and build knowledge of grammar and vocabulary; discuss reading, writing and spoken language with precise and confident use of linguistic and literary terminology (Years 7–11, merged, pp. 84–85, p. 87)

Spoken English

- speak confidently and effectively (Years 7–11, p. 85, p. 88)

The curriculum guide for English courses includes the International Phonetic Alphabet (p. 74) and an excellent glossary of terms for the study of English (Department for Education, 2014, pp. 89–107). Phonetic spellings in those sources provide insights into why particular words are pronounced differently in England compared to Canada and the United States, except for the New England states (e.g., see the phonetic spellings for *born* on p. 74, as well as the phonetic spellings for *butter* and *doctor* on p. 103).

British Columbia, Canada

British Columbia Ministry of Education (2019a)
English Language Arts K–9 Curricular Competencies
British Columbia Ministry of Education (2019b)
English Language Arts K–9 – Big Ideas

Grades 7, 8, and 9

BIG IDEAS (PP. 1–2)

- Developing our understanding of how language works allows us to use it purposefully. (Grades 6–7)
- Language and text can be a source of creativity and joy. (Grades 4–9)

Grades 7, 8, and 9

COMPREHEND AND CONNECT (READING, LISTENING, VIEWING) (PP. 4–6)

- Access information and ideas for diverse purposes and from a variety of sources and evaluate their relevance, accuracy, and reliability (Grades 6–9)
- Apply appropriate strategies to comprehend written, oral, and visual texts, guide inquiry, and extend thinking (Grades 6–9)
- Synthesize ideas from a variety of sources to build understanding (Grades 5–9)
- Recognize and appreciate how different features, forms, and genres of texts reflect different purposes, audiences, and messages (Grades 6–9)
- Think critically, creatively, and reflectively to explore ideas within, between, and beyond texts (Grades 6–9)
- Explain how literary elements, techniques, and devices enhance and shape meaning (Grades 7–9)
- Recognize an increasing range of text structures and how they contribute to meaning (Grades 6–9)
- Recognize and appreciate the role of story, narrative, and oral tradition in expressing First Peoples perspectives, values, beliefs, and points of view (Grades 6–9)

Create and Communicate (writing, speaking, representing) (pp. 4–6)

- Use writing and design processes to plan, develop, and create engaging and meaningful literary and informational texts for a variety of purposes and audiences (Grades 6–9)
- Use an increasing repertoire of conventions of Canadian spelling, grammar, and punctuation (Grades 6–9)
- Use and experiment with oral storytelling processes (Grades 6–9)
- Select and use appropriate features, forms, and genres according to audience, purpose, and message (Grades 6–9)
- Transform ideas and information to create original texts (Grades 4–9)

Grades 10, 11, and 12

Composition, Creative Writing, Literary Studies, New Media, Spoken Language
British Columbia Ministry of Education (2019c)

BIG IDEAS (P. 2)

- The exploration of text and story deepens our understanding of diverse, complex ideas about identity, others, and the world.
- Texts are socially, culturally, geographically, and historically constructed.
- Language shapes ideas and influences others.
- Engagement with writing processes can support creativity and enhance clarity of expression.
- Voice is powerful and evocative.

COMPREHEND AND CONNECT (READING, LISTENING, VIEWING) (PP. 2–3)

- Recognize and appreciate the role of story, narrative, and oral tradition in expressing First Peoples perspectives, values, beliefs, and points of view
- Read for enjoyment and to achieve personal goals
- Access information for diverse purposes and from a variety of sources to inform writing
- Explore the relevance, accuracy, and reliability of texts
- Apply appropriate strategies to comprehend written, oral, visual, and multimodal texts
- Recognize and appreciate how different forms, formats, structures, and features of texts enhance and shape meaning and impact
- Think critically, creatively, and reflectively to explore ideas within, between, and beyond texts
- Explore how language constructs personal and cultural identities
- Construct meaningful personal connections between self, text, and world
- Identify bias, contradictions, and distortions

CREATE AND COMMUNICATE (WRITING, SPEAKING, REPRESENTING) (PP. 2–3)

- Use writing and design processes to plan, develop, and create engaging and meaningful texts for a variety of purposes and audiences
- Use the conventions of Canadian spelling, grammar, and punctuation proficiently and as appropriate to the context
- Transform ideas and information to create original texts, using various genres, forms, structures, and styles
- Demonstrate speaking and listening skills in a variety of formal and informal contexts for a range of purposes
- Express and support an opinion with evidence
- Assess and refine texts to improve clarity and impact
- Use acknowledgements and citations to recognize intellectual property rights

Common Core Standards in English Language Arts

40 of 50 States in the USA

National Governors Association Center for Best Practices and Council of Chief State School Officers (2010)

http://www.corestandards.org/wp-content/uploads/ELA_Standards1.pdf

The sample learning standards below have been copied from the Common Core standards, except where learning standards have been merged for this presentation, as indicated.

Reading Standards for Literature and Informational Text

- Determine a theme or central idea of a text and analyze its development over the course of the text, including its relationship to the characters, setting, and plot; provide an objective summary of the text. (Grades 7–10, pp. 36, 38)
- Determine two or more themes or central ideas of a text and analyze their development over the course of the text, including how they interact and build on one another to produce a complex account; provide an objective summary of the text. (Grades 11–12, p. 38)
- Analyze how a drama's or poem's form or structure (e.g., soliloquy, sonnet) contributes to its meaning. (Grade 7, p. 36)
- Compare and contrast the structure of two or more texts and analyze how the differing structure of each text contributes to its meaning and style. (Grade 8, p. 36)
- Analyze how an author's choices concerning where to begin or end a story, and how to structure a text, order events (e.g., parallel plots), and manipulate time (e.g., pacing, flashbacks) create such effects as mystery, tension, or surprise, and provide a comedic or tragic resolution, for instance. (Grades 9–12, merged, p. 38)

Writing Standards

- Write arguments to support claims with clear reasons and relevant evidence. (Grades 7–12, pp. 42, 45)
- Develop and strengthen writing as needed by planning, revising, editing, rewriting, or trying a new approach, focusing on addressing what is most significant for a specific purpose and audience. (Grades 7–12, pp. 43, 46)
- Use technology, including the Internet, to produce and publish writing; to interact and collaborate with others; to cite sources and link to them. (Grades 7 & 8, merged, p. 43)
- Use technology, including the Internet, to produce, publish, and update individual or shared writing products; take advantage of technology's capacity to link to other information and to display information flexibly and dynamically; in response to ongoing feedback, including new arguments or information. (Grades 9–12, merged, p. 46)
- Write routinely over extended time frames (time for research, reflection, and revision) and shorter time frames (a single sitting or a day or two) for a range of tasks, purposes, and audiences. (Grades 6–12, pp. 44, 47)

Speaking and Listening Standards

- Delineate a speaker's argument and specific claims; evaluate the soundness of the reasoning and the relevance and sufficiency of the evidence; recognize and disregard irrelevant arguments. (Grades 7–8, merged, p. 49)
- Evaluate a speaker's point of view, reasoning, and use of evidence and rhetoric; assess the stance, premises, links among ideas, word choice, points of emphasis, and tone used; identify any fallacious reasoning or exaggerated or distorted evidence. (Grades 9–12, merged, p. 50)
- Present claims and findings, emphasizing salient points in a focused, coherent manner with pertinent descriptions, facts, details, and examples; use appropriate eye contact, adequate volume, and clear pronunciation. (Grades 7–8, merged, p. 49)
- Present information, findings, and supporting evidence clearly, concisely, distinctively, and logically such that listeners can follow the line of reasoning, and the organization, development, substance, and style are appropriate to purpose, audience, and a range of formal and informal tasks. (Grades 9–12, merged, p. 50)

Language Standards

- Demonstrate command of the conventions of standard English grammar and usage when writing or speaking. (Grades 6–12, pp. 52, 54)
- Demonstrate understanding of figurative language, word relationships, and nuances in word meanings. (Grades 6–12, pp. 53, 55)
- Acquire and use accurately grade-appropriate general academic and domain-specific words and phrases; gather vocabulary knowledge when considering a word or phrase important to comprehension or expression. (Grades 6–8, p. 53)
- Acquire and use accurately general academic and domain-specific words and phrases, sufficient for reading, writing, speaking, and listening at the college and career readiness level; demonstrate independence in gathering vocabulary knowledge when considering a word or phrase important to comprehension or expression. (Grades 9–12, p. 55)

The three sets of standards are similar with respect to the inclusion of standards for reading across multiple genres and writing clearly for multiple purposes with correct grammar and spelling. All three sets of standards also include goals for speaking and listening. Over time, the quality of writing and speaking is expected to improve through a growing vocabulary and increasing capabilities to communicate creatively, to convey information succinctly, and to argue persuasively and respectfully. The Common Core standards for writing make explicit mention of digital technologies for writing, editing, and publication, which is not the case for the standards in England and British Columbia.

6.3 STELLAR Activities

In this section we leverage data to create visuals, word clouds in particular. In addition, we create 3D models of language symbols in *Blender*. We also engrave a quotation using *Inkscape* and a laser cutter. Further, we look for patterns in data in order to decode cryptograms, as was necessary in the young adult novel *Book Scavenger* (Chambliss Bertman, 2016). Moreover, we design and create a web app to create cryptograms.

6.3.1 Creating Word Clouds with Data in Literature

To create word clouds, we are going to leverage data available in free online books. In particular, we will copy and paste paragraphs of words from novels by Jane Austen, which are available through Project Guttenberg (https://www.gutenberg.org/). Alternatively, for experimentation, you are welcome to select words in other literary works available from free Google Books (https://books.google.com/googlebooks/about/free_books.html), for instance, or you may select words from any of thousands of websites or subscription services in order to create word clouds of particular interest to you. There are critics of word clouds, objections by Harris (2011), for instance, seem to stem from the notion that visualization is reporting. Certainly, one viewing a word cloud might extract some meaning from it, especially if aware of the text entered to produce the word cloud. However, developers of word clouds have a role, which Harris (2011) did not consider. If a developer seeks to capture a story in a word cloud, editing the word list is the developer's responsibility to ensure an accurate story is told. On the other hand, when seeking to convey a theme, rather than a visual representation of word frequencies in a bar graph, a word cloud serves well.

Word cloud programs tally the number of instances of each word entered. The greater the tally, the larger the word appears in the word cloud. Software for creating word clouds eliminate definite and indefinite articles and display the remaining words at various angles. Though optional to pursue, one of the challenges in Section 6.4 is to write a program to create a word cloud (which, given a particular Python library, may be accomplished with fewer than 10 lines of code). In the following steps, we will use the Edwordle word cloud generator (http://www.edwordle.net/create.html). Alternatively, one might consider using any of several online programs, such as WordItOut (https://worditout.com/word-cloud/create) or, for young children, the word cloud generator at ABCya (https://www.abcya.com/games/word_clouds).

Complete the following steps in order to generate a word cloud, which you may edit after its initial display.

1. In a web browser, go to https://www.gutenberg.org/
2. In the **Quick Search** field, enter Jane Austen
3. Click the link titled, Complete Project Guttenberg Works of Jane Austen
4. Click the link titled, Read this book online: HTML
5. Scroll down a little and click the link titled Emma, which will display the novel
6. Scroll down to Chapter 1 and copy the text in Chapter 1
7. In another web browser tab, go to http://www.edwordle.net/create.html
8. Into the text field, paste the text copied two steps ago
9. Click the button titled, **First Generate a Wordle**
10. At this point, you may save the word cloud by clicking **Fil**e in the right sidebar and selecting either **Save Image** (to generate a.png file) or **Save As PDF**. Alternatively, you may make changes to the appearance of the word cloud by dragging and dropping words or by clicking the various sidebar items. (See Item 1 in Section 6.4: Challenges and Pursuits.)

Repeat those steps as you wish with text from other chapters or other Jane Austen novels. Alternatively, if you prefer to leverage other data, you may submit words from other literary sources.

6.3.2 Creating 3D Language Symbols

One aspect of linguistics is *semiotics*, the study of symbols. Certainly, we use a variety of symbols to communicate in writing. You may follow the steps below to create 3D language symbols, such as uppercase and lowercase letters, punctuation marks, graphemes (e.g., oo, ow, ee), ligatures and diacritics (e.g., ā, ċ, č̣, é, ñ, æ), proofreading marks (e.g., ¶, #), and phonetic markings (e.g., *uh*, the epiglottal plosive, ʡ, and the voiced epiglottal fricative, ʢ).

1. Open *Blender*
2. Delete the cube (press the X key; then press the D key)
3. Add text (e.g., click the Add button and select Text)
4. Zoom in to make the default text appear closer; then move the viewport to make the x-axis parallel with the base of your screen in order to view the text as normally positioned when read
5. Rotate the text 90° on the x-axis (e.g., Press r; x; 90; Enter/Return)
6. Switch to **Edit** mode (e.g., Press the Tab key). A cursor appears at the end of the default text.
7. Press the Backspace/Delete key four times to delete the four characters in the default text. Then hold down the Option key and press 7 in order to display the proofreading mark to begin a new paragraph, ¶
8. Switch back to **Object** mode (e.g., Press the Tab key)
9. In the Properties panel, click the **Object Data Properties** icon (lowercase **a** in green)

10. In the **Geometry** section, set Extrude to 0.2; and for Bevel, set Depth to 0.02 and Resolution to 3
11. Optionally, you may change the font by clicking the folder icon for the Regular font; navigating to the Fonts folder on your computer and selecting a font file
12. Save the file as *newParagraphSymbol.blend*
13. For 3D printing, drop down the **File** menu; select **Export** and the **Stl** format

For other symbols, repeat Steps 6 and 7; then drop down the File menu and select Save As; edit the file name, and click the Save As button. At Step 7, you may use the keyboard to enter a letter or punctuation mark. You may also copy a symbol (e.g., æ) from your word processing software and paste it into *Blender*. For 3D printing, repeat Step 13.

If the symbol you seek to create is not available in the default font in *Blender*, you may change the font per Item 2 in Section 6.4 (Challenges and Pursuits).

6.3.3 Engraving a Quotation

Select a quotation in a poem, lyrics, or prose, and then choose the material on which you want the quotation engraved. The material you select might be wood, copper, or acrylic, for instance. Then, open *Inkscape*; select the Text tool; click and drag in the canvas to create a text box; and type your quotation. Select the font and resize the text to fit your material. If you have not yet installed *Inkscape*, you can acquire it free of cost and without providing any information. Click the *Download* button at https://inkscape.org/ and select *Current Version*. The download will commence when you click the button for your computer's operating system (Mac OS, Microsoft Windows, or Linux). After saving your quotation in *Inkscape*, you will have an SVG (Scalable Vector Graphics) file. Submit your SVG file to the laser cutter/engraver after positioning your material. Laser cutters and engravers are available in maker spaces, whether in schools, homes, or civic facilities. If you wish, you may refer to Section 3.3.10 for additional steps regarding document setup in *Inkscape*.

6.3.4 Designing a Web App to Create Ciphers

In the young adult novel *Book Scavenger* (Chambliss Bertman, 2015), the 12-year-old protagonist and her companion create and decode ciphers in order to solve mysteries. In this section, we design a web app to create ciphers. The app, implemented in Section 6.3.5, will replace the letters in the words to be encrypted with alternative characters, specified by the user in the conversion string. The visual design of the app could look like Figure 6.1, which includes suitable labels for the two text entry fields, and a submit button. The conversion string will consist of 27 characters, the replacement character for the space bar followed by the 26 characters that will replace the letters a–z. After entering the conversion string and the words to be encrypted, the user taps the Encrypt button.

The basic design in Figure 6.1 would be fully functional, even though the form lacks visual appeal. We will improve the visual design of the app by framing the input fields, the title, and the button, which will unify the visual elements of the form, as in Figure 6.2. In addition to unifying the visual elements, the enhanced design of the app in Figure 6.2 sets the title apart from the input fields with contrasting colors for their backgrounds (the book shows gray scale contrasts, but the design in the Electronic Resources for the book presents the color contrast). The large title size set in a background color that contrasts the white web page helps to create a feeling that the form "pops off the page." Further, the rounded corners blend the title, the input fields, and the button into one form.

Completing the following steps in *Adobe XD* will yield the app design in Figure 6.2, though we will make the backgrounds of the title and the button red rather than black. We will also add

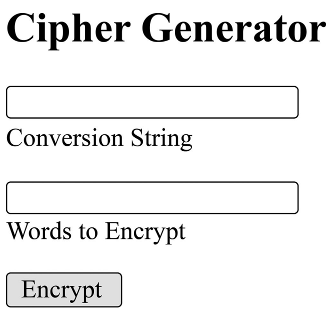


Figure 6.1 Rudimentary design of the cipher app

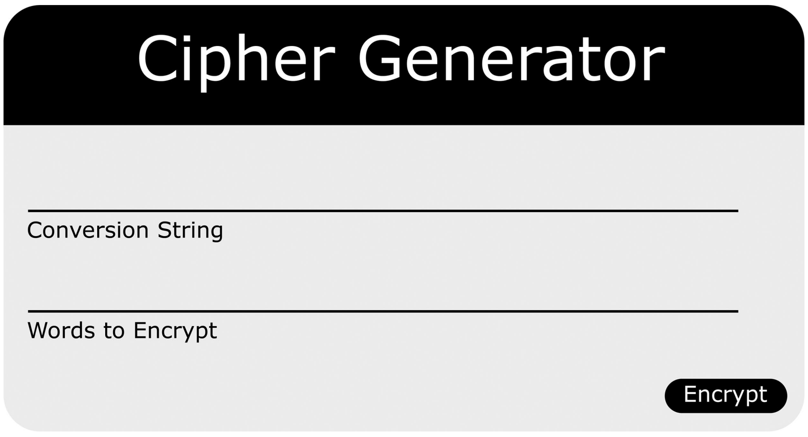


Figure 6.2 Enhanced design of the cipher app

text strings to depict sample input and output. Moreover, we will create a functional prototype of the app, which will display the output cipher after a single click of the button.

1. Open *Adobe XD*
2. Create a new custom-sized artboard that is 500 pixels wide and 300 pixels high
3. Select the **Rectangle Tool** from the palette and position a rectangle 480 pixels wide and 80 pixels high in the upper-left corner of the artboard. In the Properties window you can set the width and height dimensions precisely, as well as set the X and Y position to 10 in order to place the rectangle.
4. To round the corners, click and drag any one of the four interior circles, which appear near the handles on the four corners of the rectangle, toward the middle of the rectangle. All four corners will be rounded after this step. Now, to return the lower two corners to their original positions, hold the Alt key down and click and drag each of the lower two interior circles to its respective corner
5. **Copy and Paste** the rectangle (using Ctrl-C-V or Command-C-V, or the Edit menu drop-down items)
6. Drop down the **Object** menu and proceed through **Transform** to **Flip Vertically**
7. Drag the new rectangle, which now has rounded corners on the bottom, below the original rectangle. Make sure there is no gap between the two rectangles (the Y coordinate should be 89). Resize the height of the lower rectangle to 200 pixels

8. To set the fill color of the lower rectangle to light gray, click the **Fill** swatch in the Properties window in the right sidebar and enter **F0F0F0** for the Hex value. To remove the border, tap the checkbox beside the color swatch for the border, which will remove the default checkmark
9. Click the top rectangle to select it and set the fill color to red (**FF0000**). Also remove the border
10. To create the form title, select the **Text Tool** from the palette and drag inside the top rectangle. Type the title of the form, `Cipher Generator`
11. To change the font size and type, click the **Arrow** at the top of the Tool Palette and then click the text field. In the Properties window on the right, select the Verdana font and set the size to 30. Also set the Fill color to white. (You may need to scroll down the Properties window to view the fill color setting.). You may drag the text field to center it within the red rectangle, but to align it horizontally within the rectangle, instead, drop down the **Object** menu, proceed through **Align** to **Center (Horizontally)**
12. Save the file as *CipherAppDesign.xd*

The outline of the form is now complete, and the white title in a red background stands out, just as it will in the actual app. Now to the form elements.

1. Select the **Line Tool** from the palette and drag a line across the lower rectangle. Set the line height to 1 pixel and the width to 420 pixels. Set the X and Y positions to 25 and 150 respectively
2. **Copy and Paste** the line and move the second line to X and Y positions, 25 and 210 respectively
3. Select the **Text Tool** from the palette and drag under the top line to enter the label for the top field. Type the label, `Conversion String`. Set the font size to 12 and the color to light gray (**505050**). Drag the text label as necessary or use the align technique to position the label flush left with the start of the line, or just set the X value to 25. The Y value should be about 153.
4. **Copy and Paste** the text label and position the new text label at X and Y positions 25 and 213 respectively. Replace the text in the label to `Words to Encrypt`
5. To create a sample conversion string, **Copy and Paste** the text label created in the previous step and set the Y value to 130. Replace the text with the following and ensure that the width is at least 195 pixels: `c*defghijklmnopqrstuvwxyzab`
6. Save your work
7. To create a sample set of words to encrypt, **Copy and Paste** the conversion string (created in the previous step) and set the Y value to 190. Replace the text with the following: `my secret message`
8. To create the button for the form, select the **Rectangle Tool** and create a rounded rectangle with width 70 pixels and height 20 pixels. Place the rectangle at X and Y positions 400 and 260 respectively. Set the fill color to red (**FF0000**)
9. Select the **Text Tool** and type Encrypt. Place the text over the rounded rectangle of the button (X and Y positions 412 and 262 should work well). Set the fill color to white (FFFFFF)
10. Select the button's rectangle; hold the Shift key down and select the Encrypt text field. Drop down the **Object** menu and select **Group**
11. Save your work

The visual design of the form is complete at this point, and the bulk of the work is done. In the remaining steps we copy the form, add the cipher, and link the Encrypt button of the original form to the new form in order to display the cipher.

1. Click the **Arrow** in the Tool Palette and double-click the artboard name, which appears above the artboard. Replace the artboard name (Custom Size -1) with `Encryption Form`

2. With the name of the artboard still selected, **Copy and Paste** the artboard
3. In the new artboard, select the text, *my secret message*. **Copy and Paste** that text and set the Y value to 260. Replace the text with `oacugetgvcoguu*ig`
4. Click the arrow in the Tool Palette and then click *Encryption Form*, the name of the original artboard in order to select it
5. Save your work
6. Click the **Prototype** tab
7. Drag the handle from the **Encrypt** button over top of the second artboard
8. With the original artboard selected, click the icon for the **Play** button in the upper-right corner of the XD window in order to test the functionality of the button. Once you click the **Encrypt** button, the cipher appears, and the demonstration is complete.
9. Save your work. You may return to the **Design** tab and make any adjustments you wish. Continue refining and retesting as you wish

A functional prototype of this cipher generator is available in the *CipherAppDesign.xd* file in the Electronic Resources for the book.

6.3.5 Developing the Web App to Create Ciphers

Given the design in Figure 6.2, we will develop the app to create ciphers in HTML, CSS, and JavaScript. In case you may prefer to implement the design in Figure 6.1, which uses HTML and JavaScript (no CSS), we will implement that visual design in HTML in Part 1 below. Then you can skip Part 2, which adds the CSS, and complete the JavaScript implementation in Part 3. To implement the Design in Figure 6.2, proceed through Parts 1, 2, and 3. You may recall from the previous section that the method or algorithm for creating ciphers will replace the letters in the original words with alternative characters entered by the user in the field labeled, *Conversion String*. In Part 3, *Implementing the Encryption Algorithm in JavaScript*, we will ensure that the conversion string is exactly 27 characters. The first character will replace all occurrences of the space bar character and the subsequent 26 characters will replace all occurrences of the letters a–z in the words to be encrypted.

Part 1. Implementing the Rudimentary Design in HTML

The HTML below uses the conventions and tags discussed in Section 3.3.6. Two additional HTML tags are introduced below, namely `input` and `button`. Using your favorite text editor, enter the HTML below and save the file as *cipherGenerator1.html*.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Rudimentary Design</title>
</head>

<body>
    <h1>Cipher Generator</h1>

    <p>
      <input type="text" id="conversionString" size="30"><br />
      Conversion String
    </p>
```

```
     <p>
       <input type="text" id="inputWords" size="30"><br />
       Words to Encrypt
     </p>

     <button type="button" onclick="encryptText()">Encrypt</button>

     <p><div id="cipher"></div></p>
</body>

</html>
```

The body section of the HTML above displays seven visual elements. The first element displays the title using the largest heading tag (`h1`). Then the first input text field is displayed above its label, `Conversion String`. Next, the second input text field is displayed above its label, `Words to Encrypt`. The sixth element displays the `Encrypt` button. The final element displays the cipher when the user clicks the `Encrypt` button. JavaScript code in Part 3 generates the cipher and displays it.

The following line, copied from the HTML above, displays the first text field.

```
<input type="text" id="conversionString" size="30">
```

In the `input` tag above, notice the three attributes, `type`, `id`, and `size`. The `type` attribute is set to `text` in order to generate a text field into which the user enters a string of characters. In this case, the user enters the conversion string; hence, the `id` attribute is set to `conversionString`. The `size` attribute is set to `30` in order to make the input field sufficiently wide to view the entry of 27 characters. The `size` attribute does not limit the number of characters that the user can enter into the field. For that, we would use `maxlength="27"`. Add `maxlength="27"` as the fourth attribute in the input tag.

The following line, copied from the HTML above, displays the button, which has the label `Encrypt`.

```
<button type="button" onclick="encryptText()">Encrypt</button>
```

The `button` tag above has two attributes, `type` and `onclick`. The `type` attribute is set to `button`, which is redundant because `button` is also the tag name. In this case, the `type` attribute is used to distinguish the button from the `reset` and `submit` buttons. (Optionally, you may delete `type="button"` in your version of the app.) The `onclick` attribute is required in order to specify the name of the function to execute when the user clicks the button. As specified below in the line of HTML above, the button click in the app will execute the `encryptText()` function (see Part 3 for its implementation).

Part 2. Implementing the Enhanced Design in HTML and CSS

To retain the rudimentary design, copy *cipherGenerator1.html* to *cipherGenerator2.html*. Make all of the following edits in *cipherGenerator2.html.*

In the `title` tag, replace Rudimentary with Enhanced. Also, insert the `style` tag, as shown below.

```
<head>
     <title>Enhanced Design</title>
     <style>

     </style>
</head>
```

Now, consistent with the example that uses internal styling directives in Section 3.3.7, insert the following CSS after the closing `</title>` tag and before the closing `</head>` tag.

```
<style>
   body {
      font-family: Verdana, sans-serif;
      font-size: 12px;
   }
   .formTitle {
      font-size: 37px;
      font-weight: 300;
      width: 35%;
      height: 40px;
      border-radius: 10px 10px 0 0;
      text-align: center;
      background-color: #FF0000;
      color: white;
      padding: 13px 20px 22px 20px;
      margin-bottom: 0;
   }
</style>
```

To apply those styling directives to the title, insert the class attribute in a div tag on the first line of HTML in the body of the web page, as shown below. Notice that the `<h1></h1>` tags have been removed because styling is now controlled by the directives in the `formTitle` class.

```
<div class="formTitle">Cipher Generator</div>
```

Save the file *cipherGenerator2.html* and open the page in a web browser. If already open in a web browser, reload the page (press Ctrl-R or Command-R) in order to view the white title against its red background.

Next, insert the following styling directives before the closing `</style>` tag.

```
   .formContainer {
    width: 35%;
    height: 150px;
    border-radius: 0 0 10px 10px;
    background-color: #F0F0F0;
    padding: 20px;
   }

   label {
    padding: 0;
    width: 125px;
    display: inline-block;
   }

   input[type=text] {
    font-family: Verdana, sans-serif;
    font-size: 12px;
```

```
     background-color: transparent;
     padding: 0px 10px 10px 10px;
     display: block;
     border: none;
     border-bottom:1px solid #505050;
     width: 90%;
    }

    input[type=text]:focus {
     outline-width: 0;
     border-bottom: 1px solid #FF0000;
    }
```

To implement the styling directives for the form container, enclose the following lines of HTML within a new `div` tag that includes the `formContainer` class.

```
        <p>
          <input type="text" id="conversionString" size="30"><br />
          Conversion String
        </p>

        <p>
          <input type="text" id="inputWords"><br />
          Words to Encrypt
        </p>
```

Also, eliminate the `size` attribute in the `input` tag and surround the text field labels (`Conversion String` and `Words to Encrypt`) with the `label` tag, as shown below.

```
<div class="formContainer">
    <p>
    <input type="text" id="conversionString" maxlength="27">
    <label>Conversion String</label>
    </p>

    <p>
    <input type="text" id="inputWords">
    <label>Words to Encrypt</label>
    </p>
</div>
```

Save the file *cipherGenerator2.html* and reload the page in your web browser. The visual design of the form is now complete, and the input text fields are functional. However, the Encrypt button and the cipher display space are currently outside the form container. In the rudimentary design, the Encrypt button appears before the cipher display space, but we will reverse that order in accordance with the enhanced design prototype, which positions the cipher flush left at the bottom of the form container and the Encrypt button flush right at the bottom of the form container. We implement that design by creating three CSS classes, namely `footer`, `flush-left`, and `flush-right`. Insert the following styling directives before the closing style tag, `</style>`.

```
    .footer {
        margin-top: 25px;
    }

    .flush-left {
        float: left;
    }

    .flush-right {
        float: right;
    }
```

Also, include the following styling directives before the closing style tag, `</style>`, in order to style the button.

```
    buttons {
        width: 65px;
        border-radius: 10px;
        padding: 4px;
        background-color: #FF0000;
        color: #FFFFFF;
        border: none;
    }
```

Then nest the cipher display space flush left and the button flush right within the footer, as shown below.

```
<div class="formContainer">
    <p>
    <input type="text" id="conversionString">
    <label>Conversion String</label>
    </p>

    <p>
    <input type="text" id="inputWords">
    <label>Words to Encrypt</label>
    </p>

    <div class="footer">
        <div class="flush-left" id="cipher"></div>
        <div class="flush-right">
          <button type="button" onclick="encryptText()">Encrypt</button>
        </div>
    </div>
</div>
```

Save the file *cipherGenerator2.html* and reload the page in your web browser. Next, in Part 3, we insert JavaScript code in order to generate and display the cipher for the user's input. If you want a sneak preview of text in the cipher display space, insert a few characters for a fake cipher (e.g.,

<div class="flush-left" id="cipher">sblkmdsblkmdsblkmd</div>) and then delete those phony cipher characters before proceeding to the next section.

Part 3. Implementing the Encryption Algorithm in JavaScript

In this part, we need to write the algorithm that generates a cipher in accordance with the two strings entered by the user. The encryption algorithm has two parts. The first part associates the space bar character and each letter in the English alphabet to consecutive characters in the conversion string entered by the user. Due to the inclusion of the id attribute in the input tags we used in Part 1 (which remained unchanged in Part 2), we obtain the conversion string submitted by the user by retrieving the value of the HTML element with id="conversionString". We do this with the following JavaScript assignment statement.

```
conversionString = document.getElementById("conversionString").value;
```

To establish the mapping between the letters in the English alphabet, plus the space bar character, we declare a constant called alphabet using the following JavaScript declaration.

```
const alphabet = " abcdefghijklmnopqrstuvwxyz";
```

Notice that the first character in the variable alphabet is the space bar character and then each letter in the English alphabet follows. The first part of the encryption algorithm will extract, from left to right, each character in the variable alphabet and use each extracted character to create a key in conversionObject that associates the key with the corresponding character in conversionString. For example, consider the mapping for the following conversion string, which contains 27 characters, as required.

```
*zyxwvutsrqponmlkjihgfedcba
```

The space bar character, the first character in alphabet, maps to *, the first character in conversionString. The letter a, the second character in alphabet, maps to z, the second character in conversionString. The letter b, the third character in alphabet, maps to y, the third character in conversionString (and so on). In JavaScript, a character is retrieved from a string using the substring function with consecutive indices. For example, substring(0,1) returns the first character in a string; substring(1,2) returns the second character in a string; substring(2,3) returns the third character in a string (and so on). Regarding JavaScript syntax, the substring function is preceded by the name of the variable (or constant) from which the substring is being extracted. For example, alphabet.substring(2,3) extracts the letter b because alphabet is set to " abcdefghijklmnopqrstuvwxyz". Using the substring function to extract corresponding characters from alphabet and conversionString, we can use two assignment statements nested in a loop, as shown below, to implement the mapping part of the encryption algorithm.

```
for (i = 0; i < 27; i++) {
    crntChar = alphabet.substring(i, i + 1);
    conversionObject[crntChar] = conversionString.substring(i, i + 1);
}
```

The second part of the encryption algorithm is also implemented with a loop in order to extract each character to be encrypted. To retrieve the words to be encrypted, we use the following

JavaScript assignment statement, which employs the same technique used to retrieve the conversion string.

```
inputWords = document.getElementById("inputWords").value;
```

With the variable `inputWords` set, we could complete the encryption algorithm with the following loop.

```
for (i = 0; i < inputWords.length; i++) {
    crntChar = inputWords.substring(i, i + 1);
    encryptedWords += conversionObject[crntChar];
}
```

We would do well, though, to set `encryptedWords` to the null string before the loop begins (i.e., insert `encryptedWords = ";`). In addition, since human beings are imperfect, we can't be trusted to enter the conversion string properly. Hence, we will add an `if` statement to ensure that the only valid characters (lowercase letters and the space bar character) are encrypted, bypassing any other characters and alerting the user whenever an invalid character is encountered. The loop above, then, is refined as follows.

```
encryptedWords = ";
for (i = 0; i < inputWords.length; i++) {
    crntChar = inputWords.substring(i, i + 1);
    if (alphabet.includes(crntChar)) {
        encryptedWords += conversionObject[crntChar];
    }
    else {
        alert("'" + crntChar + "' bypassed - not valid input");
    }
}
```

We will insert one additional `if` statement in order to ensure that the conversion string entered by the user is exactly 27 characters. If the conversion string is not 27 characters, no attempt is made to produce a cipher and a suitable alert is displayed for the user.

We could also ensure that all 27 characters in the conversion are unique and terminate the program with a suitable alert when that is not the case. You may implement that feature when completing Item 3 in Section 6.4 (Challenges and Pursuits).

Since we set the `onclick` attribute to `encryptText` (in Part 1 and retained it in Part 2), the encryption algorithm must be enclosed within the following function definition, which is nested in the `<script>` tag, as shown below.

```
<script>
    function encryptText() {

    }
</script>
```

Two final notes before presenting the complete function. Just as we initialized the variable `encryptedWords` to the null string, the code below initializes `conversionObject` to null (i.e., `var` `conversionObject = { };`). Secondly, the code below contains the following statement in order to clear the output space before a cipher is displayed.

```
document.getElementById("cipher").innerHTML = "";
```

Inserting the following JavaScript code immediately before the closing body tag, `</body>` in either *cipherGenerator1.html* or *cipherGenerator2.html* (or both scripts) will yield a functional app. Save the file and reload the page in your web browser. Then try it out (debugging as necessary).

```
<script>
   function encryptText() {
      const alphabet = " abcdefghijklmnopqrstuvwxyz";
      var conversionObject = { };

      document.getElementById("cipher").innerHTML = "";
      conversionString =
        document.getElementById("conversionString").value;

      if (conversionString.length == 27) {
        for (i = 0; i < 27; i++) {
           crntChar = alphabet.substring(i, i + 1);
           conversionObject[crntChar] =
              conversionString.substring(i, i + 1);
        }

        inputWords = document.getElementById("inputWords").value;
        encryptedWords = ";
        for (i = 0; i < inputWords.length; i++) {
           crntChar = inputWords.substring(i, i + 1);
           if (alphabet.includes(crntChar)) {
              encryptedWords += conversionObject[crntChar];
           }
           else {
              alert("'" + crntChar + "' bypassed - not valid input");
           }
        }

        document.getElementById("cipher").innerHTML = encryptedWords;
      }
      else {
        alert("Conversion string must be 27 characters");
      }
   }
</script>
```

Both *cipherGenerator1.html* and *cipherGenerator2.html* are included in the Electronic Resources for the book.

6.4 Challenges and Pursuits

1. Using a literary source of your choosing, generate a word cloud at EdWordle (http://www.edwordle.net/create.html). Then adjust the appearance of the word cloud by clicking the **Re-Layout** button in the side. You may also drag and drop words, as well as zoom in and out in order to adjust the size of a word. You may click and rotate words (though rotating precisely 90 degrees was not possible) by pressing the R or E key after selecting a word or words. In addition, you may use the buttons in the sidebar to add a word or words. To delete one or more words, click the **Words List** button; then select a word and press the Delete button in the Words List dialog. With a word selected, you may also change its font and color by clicking the appropriate button in the Words List dialog. Rather than change the font or color of words individually, you may click sidebar buttons to change the font and foreground color scheme for the entire word cloud. Entering a hexadecimal color code will change the background color.
2. When seeking to insert a symbol in *Blender* for subsequent 3D printing, some language symbols are not available in the default font in *Blender*. You may add a font in *Blender* through the **Properties** dialog or by setting the **Fonts path** in *Blender* (drop down the **Edit** menu; select **Preferences**; then select **File Paths** from the sidebar of the Blender Preferences dialog; click the folder icon for **Fonts** and navigate to the folder containing your fonts). After adding a custom font, add a symbol from the custom font, and 3D print it.
3. Unlike apps with Graphical User Interfaces (GUIs), Virtual Personal Assistants (VPAs), such as Siri, Cortana, Alexa, and Google Assistant, respond to spoken language. VPAs also have capability to combine data sources, as documented and discussed by Tom Gruber, who led the original Siri design team (https://tomgruber.org/, https://tomgruber.org/technology/siri.htm). In light of a sample of Tom Gruber's publications and presentations (https://tomgruber.org/, https://tomgruber.org/a-conversation-about-conversational-ai), write a succinct statement about language, learning, and artificial intelligence. Use *Inkscape* and a laser cutter to engrave your statement.
4. Copy either *cipherGenerator1.html* or *cipherGenerator2.html* to a new file. Then update the Cipher Generator app to ensure that the conversion string contains 27 unique characters. When the user has entered one or more duplicate characters, display an alert to inform the user that 27 unique characters must be entered in the conversion string in order to create the cipher. One approach to this would set a Boolean variable to false when a key already exists for a character in the user's conversion string. In JavaScript, the `in` operator can be used to determine whether an object contains a particular key (e.g., `if crntChar in conversionObject`). If a key for a character in the conversion string already exists, alert the user and terminate the app.
5. Modify the design of the web app to display a table, with two rows and 27 columns, which shows the original characters and the replacement characters as entered in the conversion string. The visual design of the table with data for the conversion string, c*defghijklmnopqrstuvwxyzab, appears below.

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
c	*	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b

Inclusion of such a table would require an increase in the width and length of the original form.

6. Copy your current version of the Cipher Generator app to a new file. Then implement the enhanced design produced through completion of the previous item.
7. When looking for patterns in ciphers, you may develop an app to display the frequencies of the symbols that comprise the cipher (or you could use an extant online tool). Alternatively, you may prefer to decode ciphers unassisted. Decode the following cipher.

   uibolt*gps*dsfbujoh*tufmmbs*mfttpot

   Then complete the following steps to develop an app that decodes your ciphers. Copy either *cipherGenerator1.html* or *cipherGenerator2.html* to *decodeCipher.html* and modify the interface and JavaScript in order to decode the encrypted messages. Regarding the interface, the input text field for the conversion string remains the same. The label for the second input field changes from "Words to Encrypt" to "Encrypted Words." Regarding the JavaScript code, in the loop to initialize the `conversionObject`, the roles of the `alphabet` string and the `conversionString` are reversed. Hence, in the first assignment statement, crntChar is set to `conversionString.substring(i, i + 1)` and in the second assignment statement, the `crntChar` key in the `conversionObject` is set to `alphabet.substring(i, i + 1)`. I also recommend changing a couple of variables names in the second loop. A sample solution is provided in the *decodeCiphers.html* file in the Electronic Resources for this book.
8. Locate websites to eBooks, poetry, and song lyrics. Begin with the Annotated Resources below for some suggested websites. Then create a web page (Section 3.3.6 and 3.3.7) or eBook (see *Calibre* in annotated resources below) to share the sources of eBooks, poetry, and song lyrics you located.
9. For over half a century, linguists have been developing corpora by collating, analyzing, and tagging collections of words, both written and spoken. Explore some of those corpora and then design and develop a computer game that leverages the data in a corpus to improve the player's knowledge of language. To begin your exploration of corpora, you may consider the entries in the Annotated Resources for WordNet, British National Corpus, English Corpora, University of Pennsylvania Corpora of Historical English and Corpora for Multiple Languages, and corpora distributed by the Linguistic Data Consortium.
10. In Python, write a program to create a word cloud. To display the words, you may leverage the Python word cloud package, which may be installed using either one of the following two commands.

```
pip install wordcloud
```

or

```
conda install —c conda—forge wordcloud
```

6.5 Annotated Resources

Association for Computational Linguistics (ACL)
https://www.aclweb.org/portal/

About ACL
https://www.aclweb.org/portal/what-is-cl

In addition to a quick overview of ACL, this page provides a succinct definition of computational linguistics and makes clear both scientific and technological motivations for study in the field.

North American Computational Linguistics Open Competition (NACLO) https://www.nacloweb.org/

The sidebar of the ACL website contains a cascading menu titled *education*, which links to information about NACLO, a contest for high school students that requires no prior knowledge of linguistics. Sample problems from prior competitions offer numerous opportunities to learn about language (https://www.nacloweb.org/practice.php) and to gain insights into the scope of problems considered by linguists.

ACL Anthology
https://www.aclweb.org/anthology/

The sidebar of the ACL website also contains a cascading menu titled *anthology*, which links to a collection of over 65,000 papers on computational linguistics. In such a vast number of papers, a broad range of topics is considered. Analyses of sentiments expressed in social media is a relatively common theme, which includes detection of sarcasm in social media, for instance. Prediction is a relatively common goal in some work on computational linguistics, such as predicting ratings of Ted Talks. Pedagogical practice is another consideration of computational linguists, such as strategies for teaching text-mining online. This web page enables one to link to lists of proceedings papers and then to retrieve pdf files of the papers. In addition, one may download the full reference to the papers, with or without abstracts. These reference lists are .bib files, which may lead to discovery of apps designed to facilitate searches in bibliographies.

Frequently Asked Questions about Computational Linguistics
https://aclweb.org/aclwiki/Frequently_asked_questions_about_Computational_Linguistics

The sidebar of the ACL website also contains a link to the ACL Wiki, which contains links to multiple topics pertaining to computational linguists, as well as links to numerous resources (e.g., tutorials, books, corpora, datasets). The page of frequently asked questions about computational linguistics provides a brief history of the field and identifies key journals and applications in the field.

The National Museum of Language
https://languagemuseum.org/

This is a virtual museum, though it had a physical presence at one time. The virtual exhibits and resources offer a variety of content, such as cultural stories, a history of the emergence of American language, the first book of jokes, and comparisons of written languages.

Cambridge Language Surveys
https://www.cambridge.org/core/series/cambridge-language-surveys/ED52BCCA8BAFABE00AE30AD2AD992E6C

This series of books offers overviews of major language families. There is a book for each of the following language families: Australian, Andes, Dravidian, Germanic, Mainland Southeast Asia, Mesoamerican Indian, Slavic, and Turkic, as well as an edited volume on Sign language.

Project Gutenberg
https://www.gutenberg.org/

From this website, one can browse and access 60,000 free eBooks.

The Online Books Page
http://digital.library.upenn.edu/books/

This website enables access to over three million eBooks on the web.

Free Books in Google Books
https://books.google.com/googlebooks/about/free_books.html

This web page provides a search engine to access more than 10 million free eBooks.

Calibre eBook Management
https://calibre-ebook.com/

This website enables one to download Calibre, which is free and open-source eBook management software. Calibre provides features to organize and read eBook and eZine content. The software also converts between eBook formats and provides a feature to create eBooks. Calibre runs on Microsoft Windows, Mac OS, and Linux.

WordNet
A Lexical Database for English

https://wordnet.princeton.edu/
This corpus includes a simple web interface that enables word searching for free and without user registration. Output of a word search includes the definition, semantic relations, and word relations. Clicking links in the output reveals hypernyms, meronyms, derivations, and domain categories, for instance. The database can also be downloaded and is supported by extensive documentation. Cognitive psychologist George A. Miller (1995) initiated this project in the 1980s. Development continued for a few decades (Fellbaum, 2005). At this time, the database appears to be in a steady state.

British National Corpus
http://www.natcorp.ox.ac.uk/

This corpus contains 100 million words of written and spoken English collected from books, magazines, brochures, memoranda, letters, diaries, and transcripts of speeches, plays, and broadcasts. Work began in 1991 to create the corpus after an initial planning phase. Corpus development involved permissions clearance, followed by collection, encoding, and annotation of text. Documentation was also produced. In keeping with the plan to use open technologies, the corpus is rendered in XML, which is available for download for free without registration.

English Corpora
https://www.english-corpora.org/

This collection includes about 20 corpora, such as The TV Corpus, The Movie Corpus, Corpus of Contemporary American English, British National Corpus, Strathy Corpus (Canada), Wikipedia Corpus, Corpus of US Supreme Court Opinions, and Time Magazine Corpus. The two largest corpora are iWeb: The Intelligent Web-based Corpus, which contains 14 billion words, and News on the Web, which contains over 12.8 billion words. Registration is free and necessary to access the corpora.

Penn Parsed Corpora of Historical English and Corpora for Multiple Languages
https://www.ling.upenn.edu/hist-corpora/index.html

https://www.ling.upenn.edu/hist-corpora/other-corpora.html

Researchers at the University of Pennsylvania have a history of developing corpora. They have also developed software for constructing and searching corpora. Corpora and software are available for download free of cost, though some corpora are now distributed by the Linguistic Data Consortium.

Linguistic Data Consortium
https://www.ldc.upenn.edu/

The Linguistics Data Consortium includes members from universities, libraries, corporations, and governments. The repository includes nearly 900 corpora and covers more than 90 languages. The number of corpora increases by approximately 35 per year. The catalog can be searched freely, but access to each corpus requires registration and may include fees.

References

Adelaar, W. F. H. (2004). *The languages of the Andes*. Cambridge University Press.
Authored in collaboration with Pieter C. Muysken.
https://doi.org/10.1017/CBO9780511486852
Berwick, R. C. (1985). *The acquisition of syntactic language*. MIT Press.
Berwick, R. C., Okanoya, K., Beckers, G. J. L., & Bolhuis, J. J. (2011). Songs to syntax: The linguistics of birdsong. *Trends in Cognitive Sciences, 15*(3), 113–121.
British Columbia Ministry of Education. (2019a). *English Language Arts K–9 curriculum competencies*. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/continuous-views/en_ela_k-9_curricular_competencies.pdf
British Columbia Ministry of Education. (2019b). *English Language Arts K–9 – Big ideas*. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/continuous-views/en_ela_k-9_big_ideas.pdf
British Columbia Ministry of Education. (2019c). *Curriculum guides for Grades 10, 11, and 12 composition, creative writing, literary studies, new media, spoken language*.
https://curriculum.gov.bc.ca/curriculum/english-language-arts
Burridge, K., & Stebbins, T. N. (2016). *For the love of language: An introduction to linguistics*. Cambridge University Press.
Chambliss Bertman, J. (2016). *Book scavenger*. Square Fish.
Department for Education. (2014). *National curriculum in England: Framework document*. Government of the United Kingdom (gov.uk). https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/381344/Master_final_national_curriculum_28_Nov.pdf
Key Stage 3
https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/244215/SECONDARY_national_curriculum_-_English2.pdf
Key Stage 4
https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/331877/KS4_English_PoS_FINAL_170714.pdf
Dickinson, M., Brew, C., & Meurers, D. (2013). *Language and computers*. Wiley-Blackwell.
Dixon, R. M. W., & Aikhenvald, A. Y. (2008). *The Amazonian languages*. Cambridge University Press.
Fellbaum, C. (2005). WordNet and wordnets. In K. Brown et al. (Eds.), *Encyclopedia of Language and Linguistics, 2nd ed.* (pp. 665–670). Elsevier.
Gudivada, V. N., & Rao, C. R. (2018). *Computational analysis and understanding of natural languages: Principles, methods and applications*. Elsevier.
Harris, J. (2011). *Word clouds considered harmful*. NiemanLab.
https://www.niemanlab.org/2011/10/word-clouds-considered-harmful/
National Governors Association Center for Best Practices and Council of Chief State School Officers. (2010). *Common Core State Standards in English language arts*.

http://www.corestandards.org/wp-content/uploads/ELA_Standards1.pdf

Klappenbach, A. (2021). *The 12 most spoken languages in the world.* https://blog.busuu.com/most-spoken-languages-in-the-world/

Light, A. *The holy or the broken: Leonard Cohen, Jeff Buckley, and the unlikely ascent of "Hallelujah"*. Atria Books.

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM, 38*(11), 39–41.

Partridge, K. (2019). *The many lives of Leonard Cohen's "Hallelujah"*. Mental Floss. https://www.mentalfloss.com/article/609945/leonard-cohen-hallelujah

7 Mathematics

Building Dimensions through Linear Extensions

7.1 Theme

Children learn to count quite early in life. In so doing, they acquire a conception of quantity and the ability to express magnitude using numbers. They also acquire a conception of sequential order. Those early experiences in counting commonly involve tactile manipulation of objects and, for sighted individuals, visual observations. Through manipulations of objects, children begin to gain concepts of addition and subtraction. In general, as children develop number sense, they move from actual manipulation of objects to recognizing abstract representation of quantities. That is, they realize that the symbols 1, 2, 3, 4, 5, 6, 7, 8, and 9 represent specific quantities that increase from 1 through 9. Further, they realize that the digit 0 represents the complete absence of an object and as such it appears at the beginning of the ordered list of digits, as depicted below.

0 1 2 3 4 5 6 7 8 9

That list conveys the correct order of all the digits in the base 10 number system, but it may seem lacking as a visual representation. There is no visual cue connecting the numbers, even though they are in close proximity with equal white space between them. The beauty of the number line is that it connects the numbers visually, as depicted in Figure 7.1.

Since middle school and high school students can conceive of numbers less than zero, they may relate to the number line in Figure 7.2.

Drawing a second number line through the middle of the first and perpendicular to it creates a second dimension, as shown in Figure 7.3.

The space created by crossing two number lines at a 90-degree angle forms a two-dimensional space, often called the *Coordinate plane* or the *Cartesian plane*, owing to René Descartes. In 1637, Descartes produced a famous philosophical treatise which includes the words, *I think, therefore I am*. That treatise also includes an appendix, *La Gèomètrie*, which proposes problem solving by combining geometry and algebra. Both the original French version of the famous work and an

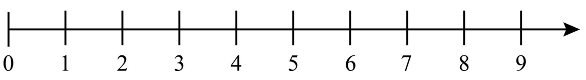


Figure 7.1 A number line with whole numbers

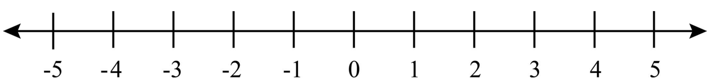


Figure 7.2 A number line with integers

DOI: 10.4324/9781003172284-10

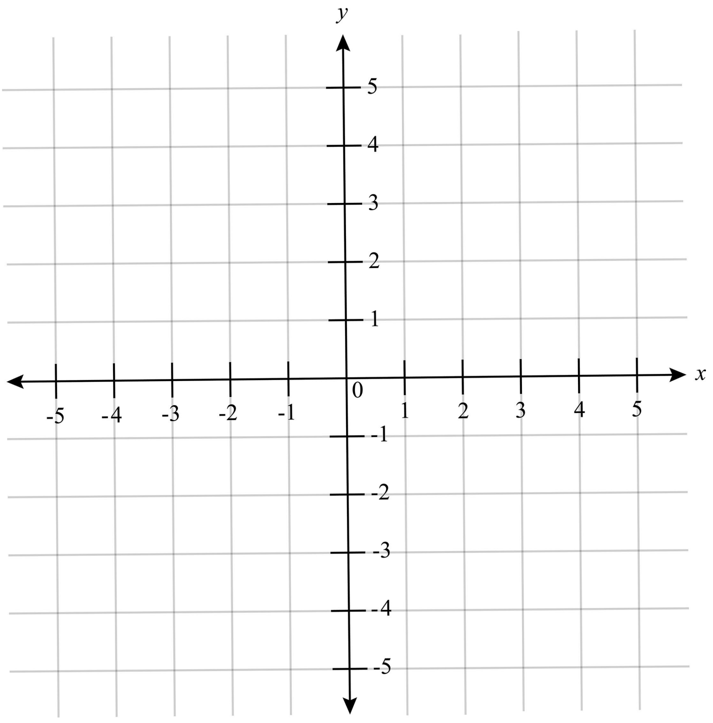


Figure 7.3 Two perpendicular number lines forming the Cartesian plane

English translation are available online for no cost (through amazon.com, for instance). Unfortunately, the appendix is not available for free, but the entry Descartes' Mathematics in the *Stanford Encyclopedia of Mathematics* (https://plato.stanford.edu/entries/descartes-mathematics/) provides important details. For our purposes, we can gain sufficient insight into problem solving through the combination of algebra and geometry by first recognizing that a point in the Cartesian plane is specified by the pair (x, y) where x is the distance from the origin along the x-axis and y is the distance from the origin along the y-axis. Then, to seek the distance between two points, (x_1,y_1) and (x_2,y_2), we could use the expression below which applies the Pythagorean Theorem to the algebraic representations of the two points.

$$d = \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2}$$

Drawing a third number line through the origin of the first two and perpendicular to the plane they form creates a third dimension. This can be gleaned from the two-dimensional Figure 7.4 if you imagine the z-axis perpendicular to the plane formed by the x and y axes.

A point in three dimensions (3D) is specified by the triple (x, y, z) where x is the distance from the origin along the x-axis, y is the distance from the origin along the y-axis, and z is the distance from the origin along the z-axis. To seek the distance between two points, (x_1,y_1,z_1) and (x_2,y_2,z_2), we could use the expression below which applies the Pythagorean Theorem to the algebraic representations of the two points in 3D space.

$$d = \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2 + \left(z_2 - z_1\right)^2}$$

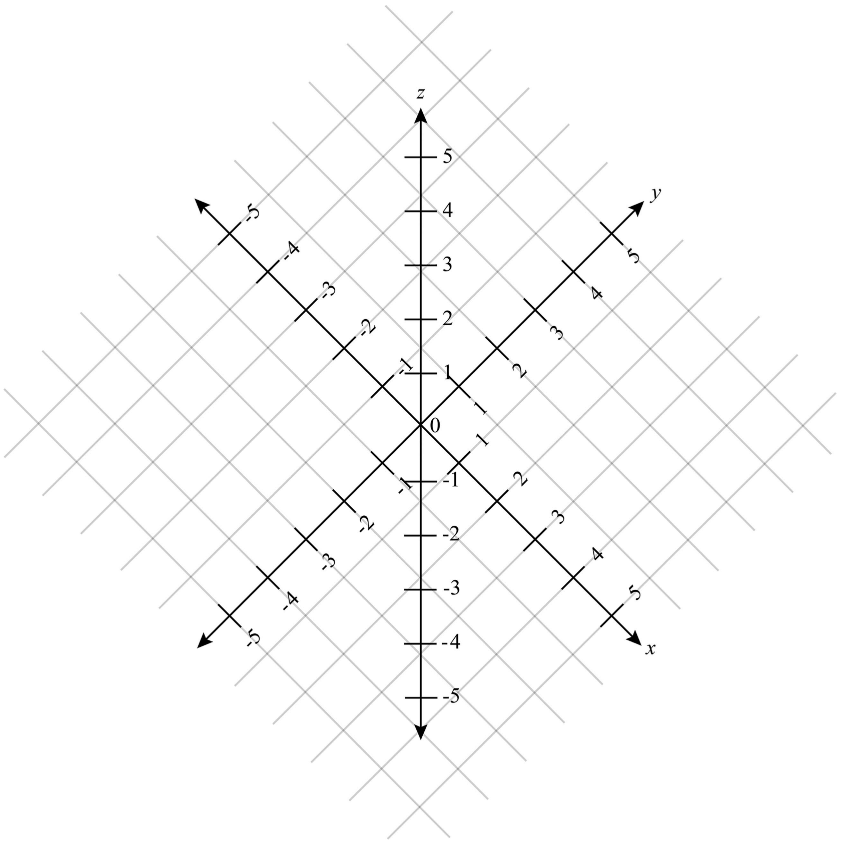


Figure 7.4 A number line perpendicular to the x-y plane forming a third dimension

In the language of mathematics, one finds precise and concise statements; one may also find beauty and joy. Thankfully, we have the proofs, results, and techniques of mathematicians to apply in problem solving.

7.2 Sample Learning Goals

Learning goals pertaining to the number line begin in Grade 2 and continue through elementary school in both British Columbia (Ministry of Education, 2016) and in Common Core standards in Mathematics (National Governors Association Center for Best Practices and Council of Chief State School Officers, 2010). In the United States, Common Core standards apply to approximately 40 of the 50 states. In England, the curriculum includes learning goals pertaining to the number line in Years 1–6 (Department for Education, 2014).

The following curricular standards pertain to the Cartesian plane and to algebraic conceptions of 3D objects and are pursued by middle school and high school students in England, British Columbia, Canada, and the United States.

England

The sample learning goals below, which pertain to the coordinate plane and 3D objects, have been copied from the curriculum guide used in England (Department for Education, 2014).

Year 6 – Geometry (p. 150)

- describe positions on the full coordinate grid (all four quadrants)
- draw and translate simple shapes on the coordinate plane, and reflect them in the axes.

Key Stage 3 – Algebra (p. 156)

- recognise, sketch and produce graphs of linear and quadratic functions of one variable with appropriate scaling, using equations in x and y and the Cartesian plane

Also, in Key Stage 3, pupils are expected to "construct and interpret plans and elevations of 3D shapes" (p. 165).

British Columbia, Canada

According to curricular standards regarding the Cartesian plane, Grade 6 students in British Columbia are expected to master "plotting points on Cartesian plane using whole-number ordered pairs" (Ministry of Education, 2016, p. 46). In addition, with respect to the History of Mathematics, covered in Grade 11, students are expected to know the Cartesian plane with respect to patterns and algebra (Ministry of Education, 2018a, p. 1).

Regarding competencies pertaining to three dimensions, Grade 8 students are expected to recognize that the "relationship between surface area and volume of 3D objects can be used to describe, measure, and compare spatial relationships" (Ministry of Education, 2016, p. 54). Exploration of that relationship may involve measuring and calculating surface area of 3D objects, as well as use of "design software to create 3D objects from nets" (Ministry of Education, 2016, p. 59).

In addition, students in Workplace Mathematics in Grades 10 and 11 are expected to know the following.

- 3D objects can be examined mathematically by measuring directly and indirectly length, surface area, and volume (Ministry of Education, 2018b, p. 1)
- 3D objects: angles, views, and scale diagrams (Ministry of Education, 2018c, p. 1)

Further, students in Apprenticeship Mathematics in Grade 12 are expected to master the following (Ministry of Education, 2018d, p. 1).

- 2D and 3D shapes: including area, surface area, volume, and nets
- 3D objects and their views (isometric drawing, orthographic projection)

Common Core Standards in Mathematics

National Governors Association Center for Best Practices and Council of Chief State School Officers (2010)

http://www.corestandards.org/wp-content/uploads/Math_Standards1.pdf

The sample learning goals below, which pertain to the coordinate plane begin in Grade 5 and continue through middle school and high school, have been copied from the Common Core curriculum guide for Mathematics.

> Grade 6 Ratios and Proportional Relationships (6.RP.3a)
>
> Make tables of equivalent ratios relating quantities with whole-number measurements, find missing values in the tables, and plot the pairs of values on the coordinate plane. Use tables to compare ratios. (p. 42)

Grade 6 The Number System (6.NS.6b)

Understand signs of numbers in ordered pairs as indicating locations in quadrants of the coordinate plane; recognize that when two ordered pairs differ only by signs, the locations of the points are related by reflections across one or both axes. (p. 43)

Grade 6 The Number System (6.NS.6c)

Find and position integers and other rational numbers on a horizontal or vertical number line diagram; find and position pairs of integers and other rational numbers on a coordinate plane. (p. 43)

Grade 6 The Number System (6.NS.8)

Solve real-world and mathematical problems by graphing points in all four quadrants of the coordinate plane. Include use of coordinates and absolute value to find distances between points with the same first coordinate or the same second coordinate. (p. 43)

Grade 6 Geometry (6.G.3)

Draw polygons in the coordinate plane given coordinates for the vertices; use coordinates to find the length of a side joining points with the same first coordinate or the same second coordinate. Apply these techniques in the context of solving real-world and mathematical problems. (p. 45)

Grade 7 Ratios and Proportional Relationships (7.RP.2a)

Decide whether two quantities are in a proportional relationship (e.g., by testing for equivalent ratios in a table or graphing on a coordinate plane and observing whether the graph is a straight line through the origin). (p. 48)

Grade 8 Expressions and Equations (8.EE.6)

Use similar triangles to explain why the slope m is the same between any two distinct points on a non-vertical line in the coordinate plane; derive the equation $y = mx$ for a line through the origin and the equation $y = mx + b$ for a line intercepting the vertical axis at b. (p. 54)

Algebra: Reasoning with Equations and Inequalities (A.REI.10)

Understand that the graph of an equation in two variables is the set of all its solutions plotted in the coordinate plane, often forming a curve (which could be a line). (p. 66)

Functions: Trigonometric Functions (F.TF.2)

Explain how the unit circle in the coordinate plane enables the extension of trigonometric functions to all real numbers, interpreted as radian measures of angles traversed counterclockwise around the unit circle. (p. 71)

Geometry: Expressing Geometric Properties with Equations (G.GPE.4)

Use coordinates to prove simple geometric theorems algebraically. For example, prove or disprove that a figure defined by four given points in the coordinate plane is a rectangle; prove or disprove that the point $(1, \sqrt{3})$ lies on the circle centered at the origin and containing the point (0, 2). (p. 78)

In Common Core standards for Mathematics, the characters 3D do not appear, but following is the one learning goal that pertains to three dimensions, which is a goal for students in Grade 8.

Grade 8 Geometry (8.G.7)

Apply the Pythagorean Theorem to determine unknown side lengths in right triangles in real-world and mathematical problems in two and three dimensions. (p. 56)

I found no Common Core standards in Mathematics for high school students that make explicit mention of three dimensions.

7.3 STELLAR Activities

First, in this section, we will acquire and leverage population data. More specifically, we create a bar graph and pie chart in *Google Sheets* to represent the data in visuals. Second, we will create four of the five regular convex polyhedra in *Blender*; we would have created all five, but the cube is available by default. For classroom display, I encourage you to produce all five polyhedra by exporting the files to Stl format and sending them to a 3D printer. While in *Blender*, we will also explore the production of 2D and 3D graphs using mathematical functions. Third, we will create polyhedral nets, which may be cut and folded to create paper models. A variation to each net will also make possible laser cutting through wood or acrylic, for instance. Then you could fasten those pieces to construct the polyhedra. Fourth, we will design a web app to represent data in visuals. Fifth, we will develop that web app. Lastly, in this section, we will leverage the population data again in order to render a bar graph and pie chart using *matplotlib*, which offers a library of graphing tools accessible in Python programing environments. We will also consider 3D plotting in *matplotlib*.

7.3.1 Rendering Population Data in Spreadsheets

First, acquire extant population data. In this case, you may download data from the Office for National Statistics in the UK (https://www.ons.gov.uk/), the United States Census Bureau (https://www.census.gov/en.html), or Statistics Canada (https://www.statcan.gc.ca/eng/start), for instance. For 2019–2020, population data were retrieved as noted below. You may prefer to use the data for the most recent year.

- UK: At the following URL, I clicked the link titled, **Mid-2019: April 2020 local authority district codes edition of this dataset** to expose the download button for the. xls file and then clicked that download button.
  https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationestimates/datasets/populationestimatesforukenglandandwalesscotlandandnorthernireland
- USA: At the following URL, under the **Tables** headings, I clicked the link titled, **Annual Estimates of the Resident Population for the United States, Regions, States, and Puerto Rico: April 1, 2010 to July 1, 2019 (NST-EST2019-01).** https://www.census.gov/data/tables/time-series/demo/popest/2010s-state-total.html#par_textimage
- Canada: At the following URL, scroll down slightly; after scrolling down slightly, click the **Download options** button; and select the top button, CSV: Download as displayed.
  https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1710000901

To generate the bar graph in Figure 7.5, complete the following steps in *Google Sheets*.

1. Open *Google Sheets* in a web browser by going to https://sheets.google.com
2. Click the **+** button to open a blank spreadsheet
3. Drop down the **File** menu; click **Open**; click the **Upload** tab; navigate to the downloaded file, ukmidyearestimates20192020ladcodes.xls, or **Drag and Drop** the file within the designated space
4. The dataset contains multiple tabs of information and data. Click the triangle icon to scroll the spreadsheet tabs to the right. Then click the **MYE1** tab to open the data sheet titled, **MYE1: Population Estimates: Summary for UK, mid-2019**
5. Since Row 7 is empty, delete it by moving the cursor to Row 7; dropping down the **Edit** menu; and selecting **Delete row 7**
6. In Cells E6–H6 to change the country names from all caps to first letter capitalized (i.e., type England, Wales, Scotland, and Northern Ireland into Cells E6–H6, respectively)

7. Highlight cells E6–H7 (e.g., drag the cursor from cell E6 to H7, or position cursor in Cell E6; hold down shift key; and press the Right arrow key three times and the Down arrow key once)
8. Drop down the **Insert** menu; select **Chart** (or click the Insert Chart icon in the ribbon)
9. A column chart is generated by default, which looks quite reasonable because there are four bars, one for each country in the UK. However, in the case of 50 States in the USA, a bar graph, which has horizontal bars, would be preferred. To change the default chart to a horizontal bar graph, notice the **Chart editor** in the right sidebar. From the Chart editor, drop down the **Chart Type** menu; scroll down to the Bar graphs and select the left-most one, the **Bar chart**
10. In the Chart Editor, click the **Customize** tab. **Click Chart & axis titles**; for **Chart title**, set the **Title text** to *UK Population by Country (2019)*. For **Title font**, select Roboto; for alignment in **Title format**, select the **Center** icon. From the drop-down menu showing **Chart title**, select **Chart subtitle** and set the **Title text** to *Data from the Office for National Statistics*. Also, select Roboto for **Title font** and select the **Center** icon for **Title format**. Now, from the drop-down menu showing **Chart subtitle**, select **Horizontal axis title**. Set **Title text** to *Number of Residents*. Set the **Title font** to Roboto
11. You may close the Chart & axis titles section by clicking **Chart & axis titles**. In the Chart editor, with the Customize tab still selected, click **Series**. Here, you may change the color of the bar if you wish. Click the **checkbox** for **Data labels**. The population for each country is now displayed.
12. You may close the Series section by clicking **Series** or just scroll down and click **Horizontal axis**. In this section, click **the checkbox** for **Show axis line**. Also, set the Minimum value to 0 and the Maximum value to 65000000 (which is 65,000,000, but commas are not permitted)
13. Click the bar chart to select it. Then, at the top right of the chart, click the vertical ellipsis button (⋮); proceed through **Download** to select your desired export format (PNG, PDF, or SVG) if you wish to try this feature
14. With the chart still selected, drag it below the data (e.g., move the top of the chart to Row 35)

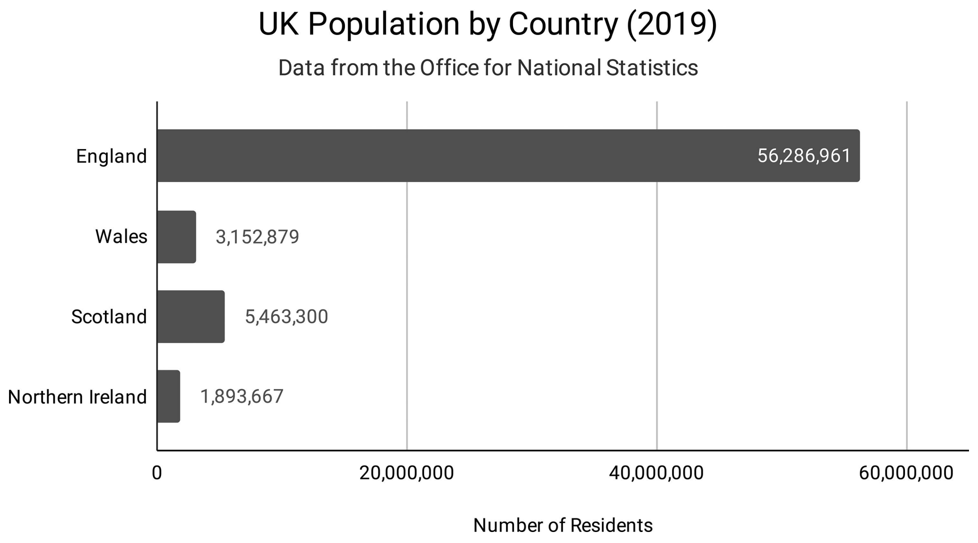


Figure 7.5 Sample bar graph depicting population estimates of countries in the UK in 2019

To produce the pie chart in Figure 7.6, complete the following steps in *Google Sheets*.

1. Drop down the **Insert** menu; select **Chart** (or click the Insert Chart icon in the ribbon)
2. Again, a Column chart is generated by default. With the new Column chart still selected and the **Setup** tab selected in the Chart editor, for **Chart type**, scroll down to select the left-most pie chart
3. In the Chart Editor, click the **Customize** tab. Click **Chart & axis titles**; for **Chart titl**e, set **Title text** to *UK Population by Country (2019)*. For **Title font**, select Roboto; for alignment in **Title format**, select the **Center** icon. From the drop-down menu showing **Chart title**, select **Chart subtitle** and set **Title text** to *Data from the Office for National Statistics*. Also, select Roboto for **Title font** and select the **Center** icon for **Title format**. Click **Chart & axis titles** to close it
4. To change the color of each slice, click **Pie Slice**. Notice that England appears in the drop-down menu. You may change the color of the England pie slice by clicking the **Colo**r drop-down menu from which you may select a circular color swatch. Notice that you may also separate the pie slice from the others by entering a **Distance from center**, either by typing a value into the field or by selecting 0%, 25%, 50%, 75%, or 100% from the drop-down menu. To change the appearance of the pie slice for the other countries, select the country name from the drop-down menu and repeat this step
5. As before, you may download the chart in PNG, PDF, or SVG format. At the top-right of the chart, click the vertical ellipsis button (⋮); then proceed through **Download** to select your desired export format (either PNG, PDF, or SVG)
6. Also, as before, you may wish to drag the chart to another location in the spreadsheet to reveal the bar graph created previously

7.3.2 Creating Models of the Five Regular Convex Polyhedra in **Blender** *for 3D Printing*

The following steps enable the development of all five regular polyhedra, which some refer to as the five planar solids, namely the tetrahedron, hexahedron (cube), octahedron, dodecahedron,

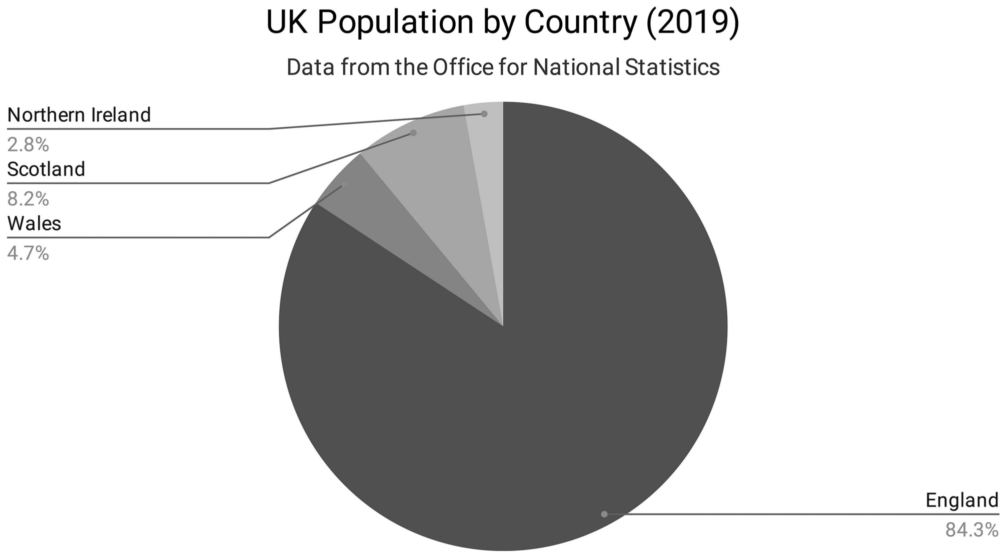


Figure 7.6 Sample pie chart depicting the population of countries in the UK in 2019

and icosahedron. The prefix of the name of each solid is based on the number of faces it has, as evident in Table 7.1, which also lists the number of vertices and edges in each solid, as well as the polygon that comprises the solid. If assigned the task of creating these five 3D models, students should be able to provide the information in Table 7.1 and verify Leonhard Euler's formula for convex polyhedra, faces plus vertices minus edges equals two ($F + V - E = 2$). In addition, students may learn about angles of polygons and gain insights into 3D coordinates. Students may also be challenged to use the Pythagorean Theorem to calculate missing side lengths of right triangles. For 3D printing purposes, after creating each polyhedron in *Blender*, the file needs to be exported to the stereolithographic (Stl) format, which can be accomplished by dropping down the **File** menu and proceeding through **Export** to **Stl**.

As it turns out, recent versions of *Blender* have a feature which can be activated through a Preference setting (noted at the end of this section), which enables automatic generation of the regular polyhedra. However, since we learn by doing, students will benefit by creating the regular polyhedra, as well as increase their self-esteem when they succeed.

Hexahedron (Cube)

1. As mentioned previously, since *Blender* creates a cube by default, you need only open a new General file in *Blender*
2. Rotate the cube to count the faces, vertices, and edges. For reference purposes, it may help to select one face when counting faces, vertices, and edges. To select a face, press the **Tab** key to enter **Edit** mode (or select Edit mode from the drop-down box); then click the **Face select** button (third button to the right of the drop-down menu for selecting Object or Edit mode); then click a face of the cube (or any other polyhedron or object)

Tetrahedron

1. Open a new General file in *Blender*. You may wish to zoom into the cube a little
2. Press the **Tab** key to enter **Edit** mode (or select Edit mode from the drop-down box); then click the **Vertex select** button (the button to the immediate right of the drop-down menu for selecting Object or Edit mode)
3. On the top of the cube, select any vertex; then, while holding the Shift key down, click the diagonally opposite vertex. On the bottom of the cube, also while holding the Shift key down, click the two vertices that are on the opposite diagonal formed by the top two vertices selected
4. From the *Blender* ribbon, drop down the **Vertex** menu and select **Bevel Vertices** (or click Shift-Ctrl-B on Windows; Shift-Command-B on a Mac)
5. Drag the Bevel handle down and release. This reveals the Bevel dialog. (Though in some cases it appears closed in the lower left portion of the window and needs to be expanded by

Table 7.1 Characteristics of the Five Regular Polyhedra

Name	*Polygon*	*Faces*	*Vertices*	*Edges*
Tetrahedron	Triangle	4	4	6
Hexahedron (cube)	Square	6	8	12
Octahedron	Triangle	8	6	12
Dodecahedron	Pentagon	12	20	30
Icosahedron	Triangle	20	12	30

clicking it.) Ultimately, set the **Width** to 2. To realize the visual effect of the vertex bevel, you may repeatedly set the bevel width. For instance, you may initially set the width to 0.2, and then reset it to 0.5, 1.0, 1.5, and lastly to 2.0.

6. You now have a tetrahedron. Rotate the tetrahedron to count the faces, vertices, and edges
7. Optionally, return to Object mode
8. Save the file
9. For 3D printing, drop down the **File** Menu, proceed through **Export** to **Stl**

Octahedron

1. Open a new General file in *Blender*. You may wish to zoom into the cube a little
2. Press the **Tab** key to enter **Edit** mode (or select Edit mode from the drop-down box)
3. From the *Blender* ribbon, drop down the **Edge** menu and select **Bevel Edges** (or click Ctrl-B on Windows; Command-B on a Mac)
4. Drag the Bevel handle down and release. This will reveal the Bevel dialog. Ultimately, set the **Width** to 1. To realize the visual effect of the edge bevel, you may repeatedly set the bevel width. For instance, you may initially set the width to 0.1, and then reset it to 0.2, 0.3, 0.5, 0.8, and lastly to 1.0. (You could even go beyond 1.0, but you would no longer have an octahedron.)
5. Rotate the octahedron to count the faces, vertices, and edges
6. Optionally, return to Object mode
7. Save the file
8. For 3D printing, drop down the **File** Menu, proceed through **Export** to **Stl**

Icosahedron

1. Open a new General file in *Blender*
2. Delete the cube (press the X key; then press the Enter/Return key)
3. From the *Blender* ribbon, drop down the **Add** menu; proceed through **Mesh** to select **Ico Sphere**
4. Set **Subdivisions** to 1. You may wish to zoom into the icosahedron a little.
5. Rotate the icosahedron to count the faces, vertices, and edges
6. For 3D printing, drop down the **File** Menu, proceed through **Export** to **Stl**

Dodecahedron

The first four steps below are the same as the first four steps in the Icosahedron, which you may bypass if you save your icosahedron file to another file name.

1. Open a new General file in *Blender*
2. Delete the cube
3. From the *Blender* ribbon, drop down the **Add** menu; proceed through **Mesh** to select **Ico Sphere**
4. Set **Subdivisions** to 1. You may wish to zoom into the icosahedron a little
5. At this point, it is possible to use the Bevel tool to reform the icosahedron into a dodecahedron (multiple websites describe this process), but in my experience the edge lengths and angles were not sufficiently consistent. Hence, I recommend adding a subsurface modifier and working with it. To add this modifier, click the **Modifier Properties** button (the blue wrench in the right sidebar). From the **Add Modifier** drop-down box, select **Subdivision Surface** (look toward the bottom of the Generate column). Then apply the

subdivision surface either by pressing Ctrl-A; G on Windows or Command-A; G on a Mac (or, from the subdivision surface properties, there is a new drop-down box with an inverted carat (or v) from which Apply can be selected). In the key sequence, Ctrl-A or Command-A may display a context-sensitive menu, from which you could select Visual Geometry to Mesh (but pressing the G key works just as well).

6. Press the **Tab** key to enter **Edit** mode (or select Edit Mode from the drop-down box); then click the **Vertex select** button (the button to the immediate right of the drop-down menu for selecting Object or Edit mode)
7. Rotate slightly upward and select the top-most vertex. (Actually, you could have selected the vertex at the center of any pentagon. Tilting the model upward can help to visualize the vertex in the center of the pentagon at the top of the forming dodecahedron.)
8. From the *Blender* ribbon, drop down the **Select** menu; proceed through **Select Similar** to **Amount of Connecting Edges**. (This selects the vertex at the center of the eleven other pentagons.)
9. Press the **X** key and select **Dissolve Vertices**
10. Press **A** to select the entire model (or from the *Blender* ribbon, click Select; All)
11. Press the **X** key and select **Limited Dissolve** and set the **Max Angle** to 25
12. You now have a dodecahedron. Rotate it to count the faces, vertices, and edges
13. Optionally, return to Object mode
14. Save the file
15. For 3D printing, drop down the **File** Menu, proceed through **Export** to **Stl**

As noted above, additional functionality can be added to *Blender* through a preference setting, which enables a feature to automate the production of the regular polyhedra. Including this additional functionality also activates a feature in *Blender* that enables plotting of graphs by entering one or more mathematical functions.

To enable automatic generation of the regular polyhedra, as well as gain additional capability to create 2D and 3D images of mathematical functions, select the **Add-ons** tab in **Preferences** (check the Edit menu, or the other menus if not there); scroll down a little and then click the checkbox for **Add Mesh: Extra Objects**. After closing the Preferences window, more entries will be available when you drop down the Add menu and select Mesh. The following cascading sequence generates a polyhedron: Drop down the **Add** menu; select **Mesh**; **Math Function**; **Regular Solid**. The production of the default solid can be changed by selecting any one of the five regular polyhedra, which appear in the drop-down menu labelled Source.

You may also want students to generate graphs of mathematical functions using the function generation features (Drop down the **Add** menu; select **Mesh**; **Math Function**; Drop down the **Add** menu; or drop down the **Add** menu; select **Mesh**; **Math Function**; **XYZ Math Surface**).

In late middle school or early high school, many students plot $y = x^2$ on the Cartesian plane. *Blender* renders that graph in three dimensions. Students may learn more by viewing 3D versions of the following graphs.

$y = x^2$
$y = 1 + x^2$
$y = 1 - x^2$
$y = x^3$

In the function plotting feature in *Blender*, exponents are entered using consecutive asterisks, so the equations above would be rendered in 3D through this sequence: **Add**; **Mesh**; **Math**

Function; **Z Math Surface**; and then enter the following in the **Z Equation** field. For plotting purposes, *Blender* provides default values for x.

```
x**2
1 + x**2
1 – x**2
x**3
```

Blender also has a feature for entering functions for each of the x, y, and z axes. For example, students studying trigonometry may appreciate the image generated by this set of functions.

x = u
y = v
z = cos(u) + cos(v)

To plot that series of functions, proceed through this sequence: **Add**; **Mesh**; **Math Function**; **XYZ Math Surface**; and then enter the following for the x, z, and z equations. For plotting purposes, *Blender* provides default values for u and v. For viewing purposes, it may be helpful to switch to Object Mode if *Blender* renders the image in Edit mode.

```
u
v
cos(u) + cos(v)
```

Even if one is not studying trigonometry, one may appreciate the intricacy and symmetry of the visual rendered by those equations. For additional learning, the z equation above could be changed to one of the following, for instance.

```
-cos(u) - cos(v)
cos(u) + sin(v)
```

7.3.3 Creating 2D Nets of the Regular Polyhedra in **Inkscape** *for Laser Cutting or Folding*

Complete the steps below in *Inkscape* to make a template, called a net, for constructing a tetrahedron.

1. Open *Inkscape*
2. Select the tool for creating **Bezier curves and straight lines**, which we will use to create an equilateral triangle. First, click the canvas and create a vertical line. Set the height to 200 pixels. Rotate the line 30 degrees (e.g., drop down the **Object** menu; select **Transform**; click the **Rotate** tab; set **Angle** to 30 degrees and click the **Apply** button)
3. **Copy** and **Paste** the line. Flip the new line horizontally (e.g., drop down the **Object** menu and select **Flip Horizontal**). Drag the new line to the original line such that the tops of the lines meet
4. Create a line that joins the bottom points of the two lines. Change the stroke style of the base of the triangle to a dashed line (e.g., drop down the **Objec**t menu; select **Fill and Stroke**; click the **Stroke style** tab; drop down the **Dashes** menu and select a **dashed line**)
5. From the **Edit** menu, choose **Select All**. Then, from the **Object** menu, select **Group**

6. **Copy** and **Paste** the triangle; **Rotate** it 120 degrees; and position the tip of the triangle at the lower-right corner of the original triangle
7. **Copy** and **Paste** the triangle; **flip** the new triangle **horizontally**; and position the tip of the triangle at the lower-left corner of the original triangle

After completing those seven steps, you will have the net in Figure 7.7(a), which can be submitted to a laser cutter. When creating a net to be printed and then folded, add tabs as shown in Figure 7.7(b). Create the first tab using the arc tool by aligning the start position with a vertex and then extending across to the opposite side. Set the depth to 85 pixels. To add subsequent tabs, copy and paste the arc; then flip or rotate the new arc and move it into position. Once printed, fold on the dotted lines and apply glue under each tab.

Once the net is complete, if you wish, you can **Select all** and scale the net (Object; Transform; Scale) to adjust its size for any printing or cutting device.

Complete the steps below in *Inkscape* to make a net for constructing a cube (hexahedron).

1. Open *Inkscape*
2. Select the tool for creating **Bezier curves and straight lines**, which we will use to create four lines that form a square. First, click the canvas and create a vertical line. Set the height to 200 pixels
3. From the top point of the line, create a horizontal line. Set the width to 200 pixels
4. Click the right-most point of the horizontal line and move down to create the second vertical line. Set the height to 200 pixels
5. Create a line that joins the bottom points of the two vertical lines. Change the stroke style of the base of the square to a dashed line
6. From the **Edit** menu, choose **Select All**. Then, from the **Object** menu, select **Group**
7. **Copy** and **Paste** the square; **Rotate** it 90 degrees; and position the top-right corner of the square to the bottom-left corner of the original square
8. Repeat twice to form the crossing pattern in Figure 7.8(a)
9. Select the bottom square of the crossing pattern. Copy and paste that square and move it to a side momentarily

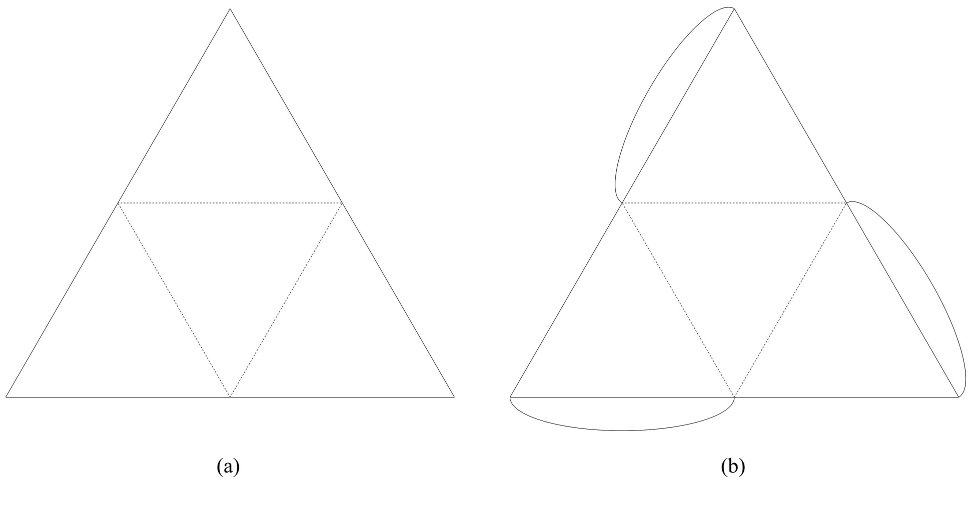


Figure 7.7 Nets of the tetrahedron for cutting and printing

10. Once again, select the bottom square of the crossing pattern. Ungroup it; select the bottom line and delete it
11. Select the square you moved aside and drag it under the bottom square of the crossing pattern

As noted above for the tetrahedron, arcs may be added to the cube using the arc tool followed by a series of duplications, flips or rotations, and movements. Then, if desired, the entire net can be selected and scaled to adjust its size for any printing or cutting device.

7.3.4 Designing a Web App to Represent Data in Visuals

In this section we design a web app that will display a two-dimensional scatter plot for a dataset entered by the user. The web app will also enable the user to display or hide a line of best fit.

Given those specifications, the app must include a data entry space and a space for the scatter plot. In addition, a button is needed to display or hide the line of best fit. Also, as depicted in Figure 7.9, the app will include a button to activate the plot. If new to *Adobe XD* and unfamiliar with software featuring a Rectangle Tool, a Circle/Ellipse Tool, and a Text Tool, consider completing the activity in Section 3.2.3 before proceeding with the steps in this section.

1. Open *Adobe XD*
2. Create new custom document 670 pixels in width and 500 pixels in height
3. Create a rectangle in Column 10 Row 10 that is 140 pixels wide and 450 pixels high
4. Create a rectangle in Column 160 Row 10 that is 500 pixels wide and 450 pixels high
5. Create a rectangle in Column 10 Row 465 that is 140 pixels wide and 30 pixels high; round the corners 5 pixels; set background color to a light gray (#EEEEEE)

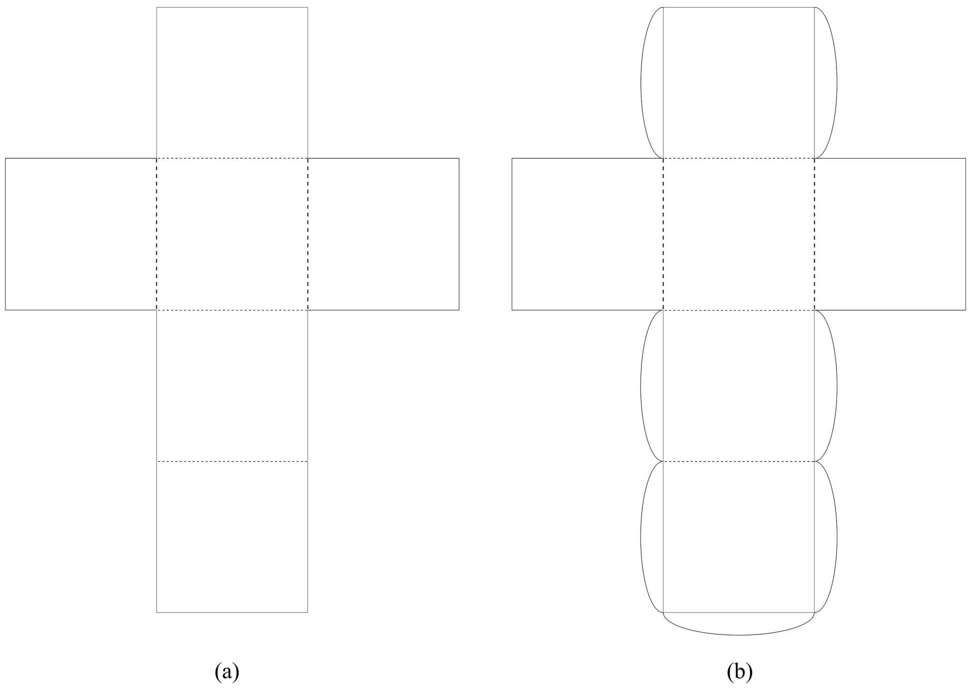


Figure 7.8 Nets of the cube for cutting and printing

6. Using the **Text Tool**, type `Plot` and center it in the rectangle created in the previous step; for the font size and type, select 20-point Times New Roman
7. Create a rectangle in Column 160 Row 465 that is 500 pixels wide and 30 pixels high; set the background color to gray (#CCCCCC)
8. Make a second button by selecting the word Plot and the rectangle underneath it; **Copy** and **Paste**; position the new button at the right edge of the rectangle created in the previous step; replace the word `Plot` with `Show Line`

Now that we have created the structural components of the app's interface, we can proceed with the steps necessary to demonstrate the functionality of the app. We will accomplish this by duplicating the art board and creating a scatter plot for sample data. Then we will duplicate this new artboard and add a line of best fit. Lastly, we will activate the Plot and Show Line buttons in order to display the appropriate artboard.

1. **Copy** and **Paste** the artboard; name the new artboard *plotBoard*
2. In *plotBoard*, for the x-axis, create a 360-pixel horizontal line at Column 270 Row 395
3. To add the major tick marks, create a 21-pixel vertical line at Column 305 Row 385; then copy and paste that line and move the new line right 35 pixels (to Column 340). Repeat that process to add and place another eight major tick marks.
4. Select the horizontal line and the ten major tick marks; **Group** them to create a single object; **Copy** and **Paste** it; then rotate it 90° and position the lowest point of the line at the left edge of the horizontal line. Now we have both the x and y axes
5. Use the **Text Tool** to add the numbers to label the major tick marks on both the x and y axes. Select 14-point Courier for these numbers
6. Using the **Text Tool**, add the data to be plotted in the rectangle above the Plot button, as shown in Figure 7.10. As before, select 14-point Courier for these numbers

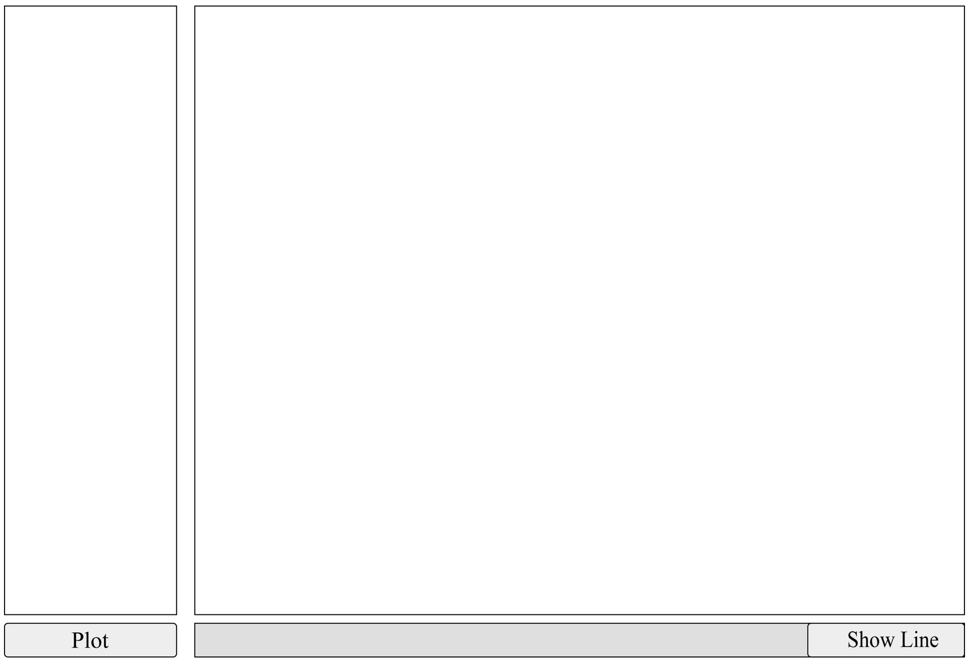
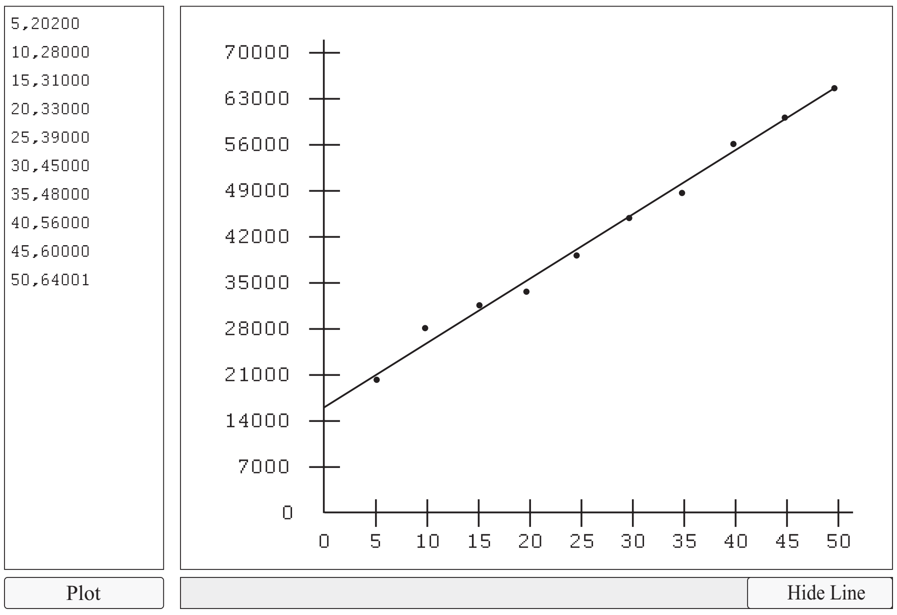


Figure 7.9 Web app interface to enable users to plot data

7. Using the **Ellipse Tool**, create a filled black circle with a 5-pixel diameter at Column 303 Row 293. (*Adobe XD* uses width and height measures; hence, a 5-pixel diameter has both the width and height set to 5.) **Copy** and **Paste** that circle and move it to Column 336 Row 251. Repeat that process to create and position the remaining eight data points, as shown in Figure 7.10. The (column, row) positions for the remaining points should be close to (373,237), (405,226), (440,198), (475,170), (511,151), (546,115), (581,93), and (616,72)
8. Select *plotBoard*; **Copy** and **Paste** it; name the new artboard *plotLineBoard*
9. In *plotLineBoard*, for the line of best fit, create a diagonal line from a point above the 14000 mark on the y-axis to the top-right plotted point, as shown in Figure 7.10. The bounding box of the line should have width 350; height 245; x position 270; and y position 73
10. Replace the word `Show` with `Hide`, which will change the button text to `Hide Line`
11. With `Hide Line` still selected, hold the Shift key, and select the rectangle beneath it; **Group** them to create one object (which will become an active button)
12. In *plotBoard*, select the text, `Show Line`, and the rectangle beneath it; **Group** them to create one object (which will also become an active button)
13. Still in *plotBoard*, select the text box containing the data and **Copy** it
14. Select the original artboard, and **Paste**. Now the data appear in the original art board
15. Still in the original artboard, select the word, Plot, and the rectangle beneath it; **Group** them to create one object (which will become an active button in the next step)
16. Click the **Prototype** tab; select the Plot button; then **Drag** and **Drop** its handle over any part of *plotBoard*. You may find that the Transition action with no animation works well for this button and for the buttons activated in the next two steps.
17. In *plotBoard*, select the Show Line button; then **Drag** and **Drop** its handle over any part of *plotLineBoard*
18. In *plotLineBoard*, select the Hide Line button; then **Drag** and **Drop** its handle over any part of *plotBoard*
19. Save the file
20. Select the original art board and click the **Preview** button to test the buttons



```
5,20200
10,28000
15,31000
20,33000
25,39000
30,45000
35,48000
40,56000
45,60000
50,64001
```

Figure 7.10 Web app displaying the scatter plot with line of best fit

7.3.5 Developing the Web App to Represent Data in Visuals

In the previous section, we created a functional prototype of a web app that displays a scatter plot. That prototype established the visual design of the app and demonstrated its functionality for a particular dataset. In this section, we use that design to guide development of the actual web app. We will approach the development task in three parts: (1) Create the visual elements of the user interface and provide a sample dataset; (2) Display the X and Y axes with custom labels; and (2) Display the points and line of best fit.

Part 1. Create the Visual Elements of the User Interface with Sample Data

Fundamentally, the user interface in this app contains one input space, one output space, and two buttons. To facilitate the entry of multiple rows of data, the input space will be implemented with the HTML `textarea` tag. The output space requires the display of lines for the X and Y axes, as well as lines for the tick marks on the axes, and the line of best fit. In the same output space, numbers to label the tick marks must be displayed, as well as circles to represent the data points. In HTML, a `canvas` must be implemented to meet those output requirements. Lastly, one HTML `button` tag is used for each of the two buttons. Since the `button` tag specifies the function to be called when the button is clicked, the code below includes two JavaScript functions.

Enter the following HTML and JavaScript into your text editor and save the file as *scatterPlot.hml*.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Scatter Plot App</title>
</head>

<body>

<div style="width: 100%; height: 450px">
  <div style="width: 140px; height: 98.8%; float: left">
    <textarea id="data" rows="20" cols="15" style="resize: none; height:
    100%">5,20200
10,28000
15,31000
20,33000
25,39000
30,45000
35,48000
40,56000
45,60000
50,64001</textarea>
    <br />
    <button id="plot" type="button" style="width: 140px" onclick="
    make-ScatterPlot()">Plot</button>
  </div>

  <div style=" height: 100%; margin-left: 150px">
```

```
      <canvas id="plottingCanvas" width="500" height="450" style="border:1px
      solid #909090"></canvas>
      <br />
      <div id="buttonbar" style="width: 500px; background-color: lightgray">
        <button id="toggle" type="button" style="width: 100px; margin-left:
        400px" onclick="toggleRequestLine()">Show Line</button>
      </div>
    </div>
</div>

<script>

var requestLine = false;
var canvas = document.getElementById("plottingCanvas");
var canvasContext = canvas.getContext("2d");

function toggleRequestLine() {
  requestLine =! requestLine;

  nextState = (requestLine)? "Hide Line" : "Show Line";
  document.getElementById('toggle').innerText = nextState;

  console.log("From toggleRequestLine: Calling makeScatterPlot with
  requestLine", requestLine);
  makeScatterPlot();
}

function makeScatterPlot() {
  console.log("In makeScatterPlot");
}

</script>

</body>
</html>
```

After saving *scatterPlot.hml* in your text editor, open the file in your web browser. At this point, data have been added to the design presented in Figure 7.9. Also, the two functions, `toggleRequestLine` and `makeScatterPlot`, print a status message for the programmer to view in the JavaScript console. Accordingly, open the JavaScript console in your web browser to view state information.

Within the `body` tag above, notice the nested `<div>` tags, which implement the two-column design. The column on the left (implemented with `float: left` in the `<div>` tag) contains the `<textarea>` tag, which in this app enables the user to enter the data to be plotted. The `<textarea>` tag contains the `id` attribute (`id="data"`). Later, we will use that `id` in the JavaScript function, `makeScatterPlot`, to retrieve the data entered by the user. I have entered 10 lines of default data in the `textarea`, which you are free to change as you wish. In accordance with the HTML above, immediately below the data input area is the Plot button. The following line of HTML renders the Plot button.

```
<button id="plot" type="button" style="width: 140px" onclick="makeScatterPlot
()">Plot</button>
```

The setting of the `onclick` attribute ensures that the app will call the `makeScatterPlot` function when the user taps the Plot button.

The following tag begins the HTML specification of the column on the right, with its left margin set to 150.

```
<div style="height: 100%; margin-left: 150px">
```

In the design step, we positioned the Column at 160 pixels, but in HTML the left margin is set to 150 because the web page has a default left margin of 10 pixels. Importantly, the width and height dimensions of the plotting space, which occupies the vast majority of space in the column on the right, are 500 pixels and 450 pixels respectively, precisely as designed. The space used for plotting is implemented with the `canvas` tag in the line below. The `style` attribute in the `canvas` tag is used to make a visible border; otherwise, the plotting space would not be evident until a graph was plotted.

```
<canvas id="plottingCanvas" width="500" height="450" style="border:1px solid
#909090"></canvas>
```

The `id` attribute of the `canvas` tag is set to `plottingCanvas`, which is used in the following global variable declaration in JavaScript.

```
var canvas = document.getElementById("plottingCanvas");
```

That JavaScript variable declaration provides access to the HTML canvas. There is one level of indirection, though, which necessitates the following variable declaration.

```
var canvasContext = canvas.getContext("2d");
```

The `canvasContext` variable will be used to plot points and to draw lines on the HTML `canvas`.

Immediately below the plotting canvas is the `<div>` tag that specifies the width of the button bar, which is rendered as a light gray rectangle 500 pixels wide.

```
<div id="buttonbar" style="width: 500px; background-color: lightgray">
</div>
```

That `<div>` tag contains only the following line of HTML, which renders one button flush-right in the button bar, as previously designed.

```
<button id="toggle" type="button" style="width: 100px; margin-left: 400px"
onclick="toggleRequestLine()">Show Line</button>
```

The `onclick` attribute is set to `toggleRequestLine`, which ensures that function will be called when the user taps the Show Line button. Since this button is a toggle, the Boolean variable `requestLine` is used to switch between false and true, to determine whether the user wishes to view the line of best fit. The default value of this variable is false, due to the following variable declaration.

```
var requestLine = false;
```

A Boolean variable is toggled by negating its current value. In JavaScript, the exclamation mark is the negation operator. Hence, the following line of code toggles `requestLine`.

```
requestLine = ! requestLine;
```

As the value of `requestLine` changes from `false` to `true` and back again, the text of the button changes from `Show Line` to `Hide Line`. This is accomplished with the following two assignment statements in the `toggleRequestLine` function.

```
nextState = (requestLine)? "Hide Line" : "Show Line";
document.getElementById('toggle').innerText = nextState;
```

In this first iteration of the app, the following line is included to display the current state of the `requestLine` variable. You are encouraged to view that output in the JavaScript console.

```
console.log("From toggleRequestLine: Calling makeScatterPlot with requestLine",
requestLine);
```

Lastly, in the `toggleRequestLine` function, a call is made to the `makeScatterPlot` function to display the scatter plot, with or without the line of best fit, as the user wishes.

To facilitate monitoring the functionality of this version of the app, there is one `console.log` statement in the `makeScatterPlot` function to provide feedback when that button is tapped.

Part 2. Displaying the X and Y Axes with Custom Labels

A scatter plot displays X and Y axes with appropriate labels on the tick marks. The tick marks divide the axis into a suitable number of intervals, which encompass the data range. The label for each tick mark is a multiple of the interval. Calculation of the axis intervals is a challenge. Since we will use 10 tick marks on each axis and the minimum value will always be 0, the interval could be calculated as the maximum value in the dataset divided by 10. That is a good start, but if the maximum number is not a multiple of 10, the interval includes a decimal part, which in some cases would be fine. However, in many cases the decimal portion would be insignificant. In those cases, we could round up to the one's place to ensure inclusion of all the data. Still, in many datasets, with values in the thousands or millions, the value of the one's place is also insignificant. Since we are dividing each axis into 10 intervals, a better approach is to round to the place value one less than the highest place value of the maximum number.

Consider the utility of this approach for the sample data, in which 50 is the maximum number of the values plotted on the x-axis and 64,001 is the maximum number of the values plotted on the y-axis. In the case of 50, the ten's place is the highest place, so this approach rounds to one place less, the one's place. Since 50 is a multiple of 10, no rounding occurs; the interval for the x-axis is 5 because 50 divided by 10 equals 5. In the other case, 64,001 is rounded up to the thousands place because that is one place value less than the ten-thousands place, which is the highest place value of 64,001. To begin, 64,001 divided by 10 is 6400.1, which would yield 6401 if rounded up to the one's place, but that is an unsuitable interval. Instead, rounding up to the thousands place makes the interval 7000. This is done by dividing 6401 by 1000 to yield 6.401 and rounding up using the `ceiling` function (`Math.ceil` in JavaScript) and then multiplying by 1000. To determine the highest place value of a positive number, x, one could count the number of digits in x. Then, with n representing the number of digits in x, 10^n yields the highest place value of x. Since we need the place value one less than the highest place value of the maximum number of the dataset, we need to calculate 10^{n-1} to determine the place value. The maximum number of the dataset could be converted to a string and the length function called on the string to return the number of digits in the string. Subtracting one from that string length would yield the needed exponent. Alternatively, to streamline processing, we will truncate the result of the Math.log10 function to return the needed exponent. The Math.log10 function returns the logarithm to the Base 10 of a number, x, the ceiling of which yields the number of digits in x.

Ignoring any decimal part of that logarithm yields the number of digits in x minus 1, which is the precise result needed. The following JavaScript code will be used to calculate the intervals for the x-axis and the y-axis, with the maximum number of the values plotted on the x-axis stored in the variable, `maxX`., and the maximum number of the values plotted on the y-axis stored in the variable, `maxY`.

```
intervalX = Math.ceil(maxX / 10);
intervalY = Math.ceil(maxY / 10);
exponentX = Math.trunc(Math.log10(intervalX));
exponentY = Math.trunc(Math.log10(intervalY));
intervalX = Math.ceil(intervalX / 10 ** exponentX) * 10 ** exponentX;
intervalY = Math.ceil(intervalY / 10 ** exponentY) * 10 ** exponentY;
```

Those calculations require the maximum number of the values on the x-axis and the maximum number of the values on the y-axis, which the app will determine by looping through the data. Each time through the loop, using the `Math.max` function, the current value of the dataset will be compared to the maximum value encountered so far. The maximum value will be updated when the current value is greater than the maximum value encountered so far. We will use the following assignment statement to retrieve the dataset, which arrives as a single string and is stored in the variable, `dataString`.

```
dataString = document.getElementById("data").value;
```

Then we will use the following assignment statement, which uses the JavaScript `split` function, to parse `dataString`. The `split` function below detects each new line character (`"\n"`) and returns an array, each location of which contains one line of data.

```
dataStrings = dataString.split("\n");
```

In JavaScript, array indices start at zero. We could use the following code to loop through each element of the array.

```
for (i = 0; i < dataStrings.length; i++) {
    console.log(dataStrings[i]);
}
```

Given the default dataset for this app, the first element of the `dataStrings` array (`dataStrings[0]`) will be the string, `"5,20200"`.

Inserting the following line of code into the loop above splits each line of data into the array `values`. The value on the x-axis is stored in `values[0]` and the value on the y-axis is stored in `values[1]`.

```
values = dataStrings[i].split(", ");
```

Hence, the first time through the loop above, `values[0]` will be `"5"` and `values[1]` will be `"20200"`. Since both of those values are strings, they will be converted to numbers using the two assignment statements below, which stores the numbers as `currentX` and `currentY`.

```
currentX = Number(values[0]);
currentY = Number(values[1]);
```

As shown below, we now have all the code needed to calculate the interval for each of the two axes.

```
maxX = -1;
maxY = -1;
for (i = 0; i < dataStrings.length; i++) {
   values = dataStrings[i].split(", ");
   currentX = Number(values[0]);
   currentY = Number(values[1]);

   maxX = Math.max(maxX, currentX);
   maxY = Math.max(maxY, currentY);
}

intervalX = Math.ceil(maxX / 10);
intervalY = Math.ceil(maxY / 10);
exponentX = Math.trunc(Math.log10(intervalX));
exponentY = Math.trunc(Math.log10(intervalY));
intervalX = Math.ceil(intervalX / 10 ** exponentX) * 10 ** exponentX;
intervalY = Math.ceil(intervalY / 10 ** exponentY) * 10 ** exponentY;
```

The code above is placed into the function `calculateIntervals`. Note that when we calculate the slope and y-intercept of the line of best fit, we will need the arithmetic mean of the values on the x-axis and the arithmetic mean of the values on the y-axis, as captured in the following mathematical expressions.

$$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad \bar{Y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

Since those expressions require the sum of the values on the x-axis and the sum of the values on the y-axis, we include the following two assignment statements in the loop above to calculate those required sums.

```
sumX += currentX;
sumY += currentY;
```

After the loop, we will calculate the two arithmetic means using the following two statements.

```
meanX = sumX / dataStrings.length;
meanY = sumY / dataStrings.length;
```

Insert the following function into *scatterPlot.html*.

```
function calculateIntervals(dataStrings) {

   sumX = 0;
   sumY = 0;
   maxX = -1;
   maxY = -1;
   for (i = 0; i < dataStrings.length; i++) {
        values = dataStrings[i].split(", ");
        currentX = Number(values[0]);
        currentY = Number(values[1]);
        console.log(currentX, currentY);

        maxX = Math.max(maxX, currentX);
        maxY = Math.max(maxY, currentY);

        sumX += currentX;
        sumY += currentY;
    }
    console.log("maxX =", maxX, " maxY =", maxY);

    intervalX = Math.ceil(maxX / 10);
    intervalY = Math.ceil(maxY / 10);
    exponentX = Math.trunc(Math.log10(intervalX));
    exponentY = Math.trunc(Math.log10(intervalY));
    console.log("exponentX =", exponentX, " exponentY =", exponentY);
    intervalX = Math.ceil(intervalX / 10 ** exponentX)* 10** exponentX;
    intervalY = Math.ceil(intervalY / 10 ** exponentY)* 10** exponentY;
    console.log("intervalX =", intervalX, " intervalY =", intervalY);

    console.log("sumX =", sumX, " sumY =", sumY);
    meanX = sumX / dataStrings.length;
    meanY = sumY / dataStrings.length;
    console.log("meanX =", meanX, " meanY =", meanY);
}
```

Also, insert the following three lines of code into the `makeScatterPlot` function.

```
dataString = document.getElementById("data").value;
dataStrings = dataString.split("\n");

calculateIntervals(dataStrings);
```

Further, even though the app would execute properly without the following global variable declarations, I recommend adding them after the `<script>` tag at this time.

```
var intervalX = 0;
var intervalY = 0;
var meanX = 0;
var meanY = 0;
var maxX = -1;
```

Then save the file and run the code. You may wish to verify the accuracy of the values displayed in the JavaScript console window for `maxX` (50), `maxY` (64,001), `intervalX` (5), `intervalY` (7000), `meanX` (27.5), and `meanY` (42420.1), for instance.

Now that we have the intervals, we can move toward displaying the axes, complete with tick marks and labels. Since the tick marks are lines, we can display them by moving to a particular point on the HTML canvas, using the `moveTo` method, and then draw a line from that position to the other end point of the line, using the `lineTo` method. Both of those methods require a canvas context. In this app, we have initialized `canvasContext` in a variable declaration for the purpose of enabling output to the canvas. Hence, we could display a diagonal line on the canvas from (10,15) to (110,115) using the following four lines of code. Since the origin (0,0) of the HTLM canvas is in the upper-left corner of the canvas, this line would begin at the point 10 pixels from the left and 15 pixels below the top of the canvas and finish at the point 110 pixels from the left and 115 pixels from the top of the canvas.

```
canvasContext.beginPath();
canvasContext.moveTo(10, 15);
canvasContext.lineTo(110, 115);
canvasContext.stroke();
```

If we added a second line, using the code below, the second line would go from (110,115) to (120,230).

```
canvasContext.beginPath();
canvasContext.moveTo(10, 15);
canvasContext.lineTo(110, 115);
canvasContext.lineTo(120, 230);
canvasContext.stroke();
```

If the two lines were not supposed to be joined, the starting point of the second line could be moved with an intervening `moveTo`, as in the code below. This technique of moving to a point on the canvas (`moveTo`) before drawing a line (`lineTo`) is used to display tick lines through the axes. Of necessity, variables are used rather than the constants that appear in the following statements.

```
canvasContext.beginPath();
canvasContext.moveTo(10, 15);
canvasContext.lineTo(110, 115);
canvasContext.moveTo(100, 155);
canvasContext.lineTo(120, 230);
canvasContext.stroke();
```

To add text to the canvas, we will use the `fillText` method, which requires three parameters. The text to be displayed is first parameter; the second and third parameters are the x and y coordinates of the location to display the text. Technically, the x and y coordinates specify the lower-left corner of the bounding box containing the text to be displayed. For example, the JavaScript statement below displays `"Hello"` in the upper-left corner of the canvas because 10 pixels down from the top of the canvas is sufficient to display the default font size.

```
canvasContext.fillText("Hello", 0, 10);
```

The font type and font size of text displayed on the canvas are set in the code below.

```
canvasContext.font = "16px Courier";
canvasContext.fillText("Hello", 0, 10);
```

Use of a fixed-width font, such as Courier, is helpful when aligning numbers vertically, which is necessary for the labels on the y-axis.

Now, let's go ahead and display the axes, along with their tick marks and labels. Insert the following function into *scatterPlot.html*.

```
function makeTickMarkLabel(startX, startY, endX, endY, label, labelX, labelY)
{
   canvasContext.moveTo(startX, startY);
   canvasContext.lineTo(endX, endY);
   canvasContext.fillText(label, labelX, labelY);
}
```

We will call that function 10 times to add tick marks and labels to the x-axis and 10 times to add tick marks and labels to the y-axis. First, we will create the x-axis and y-axis. Insert the following function into *scatterPlot.html*.

```
function makeAxes() {
   canvasContext.clearRect(0, 0, canvas.width, canvas.height);
   canvasContext.font = "16px Courier";

   canvasContext.beginPath();
   canvasContext.moveTo(455, 400);
   canvasContext.lineTo(100, 400); // x-axis
   canvasContext.lineTo(100, 45); // y-axis

   canvasContext.stroke();
}
```

Then add the following call to that function as the last line in the `makeScatterPlot` function.

```
makeAxes();
```

Save *scatterPlot.html* and execute the program by reloading the page in your web browser. Tap the Plot button to view the axes. (If the numbers appear in a lower resolution than expected, there is nothing wrong with your code. The explanation and fix appear at the end of the third part of this section. For now, continue to focus on drawing lines and text on the HTML canvas.)

During app development, programmers seek to write robust code without compromising development time greatly. Robust code is flexible and makes future development easier. In the code above we used the following three statements to position the axes using absolute coordinates.

```
canvasContext.moveTo(455, 400);
canvasContext.lineTo(100, 400); // x-axis
canvasContext.lineTo(100, 45); // y-axis
```

That code is functional and was easy to write, but if we ever change the size of the HTML canvas, we will need to rewrite the code. We can add flexibility be replacing those three lines of code with the following three lines and then adding three global constants.

```
canvasContext.moveTo(xOffset + 355, yOffset);
canvasContext.lineTo(xOffset, yOffset); // x-axis
canvasContext.lineTo(xOffset, yOffset - pixelDistance * 10-5); // y-axis
```

Now, after the variable declarations at the start of the script, add the following constants. With the three constants below, the three statements above yield the same values as the absolute coordinates used earlier.

```
const xOffset = 100;
const yOffset = 400;
const pixelDistance = 35;
```

Save *scatterPlot.html* and execute the program by reloading the page in your web browser. Click or tap the Plot button to view the axes. The display of the axes is precisely the same. However, we are in much better position for the next step, the display of the tick marks and labels. In addition, we now have robust code for future development.

As noted above, we will call the `makeTickMarkLabel` function repeatedly to make a tick mark and label. The code below, when parameters are supplied, will place the 10 tick marks and their labels on the x and y axes.

```
for (i = 1; i <= 10; i++) {
       makeTickMarkLabel(, , , , , ,); // x-axis
       makeTickMarkLabel(, , , , , ,); // y-axis
}
```

The first four parameters to the `makeTickMarkLabel` function specify the end points of the tick mark. Each of the two endpoints has an x coordinate and a y coordinate. Accordingly, the first four parameters in the `makeTickMarkLabel` function are `startX, startY, endX,` and `endY`.

Since the tick marks on the x-axis are vertical lines, the x coordinate of the two endpoints is the same. Each tick mark is separated by 35 pixels (`pixelDistance`) beginning at 35 pixels from the origin. We have set at HTML Canvas position (100,400), which we access through two constants, namely `xOffset` and `yOffset`. The loop control variable, `i`, acts as a counter for the tick marks. Hence, for the x-axis tick marks, we use the following expression for both `startX` and `endX`.

```
xOffset + i * pixelDistance
```

Inserting that expression for both the `startX` and `endX` parameters in the first call to the `makeTickMarkLabel` function above, which specifies the parameters for the tick marks on the x-axis, yields the following.

```
for (i = 1; i <= 10; i++) {
    makeTickMarkLabel(xOffset+i*pixelDistance,,xOffset+i*pixelDistance,,,,);
    // x-axis
    makeTickMarkLabel(, , , , , ,); // y-axis
}
```

Since the x-axis tick marks are vertical lines 21 pixels in length, we set `startY` to `yOffset - 10` and `endY` to `yOffset + 10`. Inserting those expressions for the x-axis `startY` and `endY` parameters, respectively, yields the following.

```
for (i = 1; i <= 10; i++) {
    makeTickMarkLabel(xOffset + i * pixelDistance, yOffset - 10, xOffset + i
    * pixelDistance, yOffset + 10, , ,); // x-axis
    makeTickMarkLabel(, , , , , , ); // y-axis
}
```

The endpoints of the tick marks on the y-axis are calculated similarly, but since y-axis tick marks are horizontal lines, the roles of the x and y coordinates are reversed. Hence, we subtract 10 from `startX`, add 10 to `endX`, and use the following expression for both `startY` and `endY`.

```
yOffset - i * pixelDistance
```

Inserting those expressions for the first four parameters into the second call to the `makeTick-MarkLabel` function, which specifies the parameters for the tick marks on the y-axis, yields the following.

```
for (i = 1; i <= 10; i++) {
    makeTickMarkLabel(xOffset + i * pixelDistance, yOffset – 10, xOffset + i
    * pixelDistance, yOffset + 10, , ,); // x–axis
    makeTickMarkLabel(xOffset –10, yOffset - i * pixelDistance, xOffset + 10,
    yOffset - i * pixelDistance, , ,); // y–axis
}
```

The labels on the x-axis for the sample data are 5, 10, 15,..., 50, which are multiples of `inter-valX`, which makes `intervalX * i` the fifth parameter of the first call to `makeTickMarkLabel`. Similarly, the labels of the y-axis are multiples of `intervalY`, which makes `intervalY * i` the fifth parameter of the second call to `makeTickMarkLabel`, as shown below.

```
for (i = 1; i <= 10; i++) {
    makeTickMarkLabel(xOffset + i * pixelDistance, yOffset – 10, xOffset + i
    * pixelDistance, yOffset + 10, intervalX * i, ,); // x–axis
    makeTickMarkLabel(xOffset –10, yOffset - i * pixelDistance, xOffset + 10,
    yOffset - i * pixelDistance, intervalY * i, ,); // y–axis
}
```

The two remaining parameters provide the x and y coordinates of the lower-left corner of the bounding box of the text labels. The y coordinate for the x-axis labels is constant at 30 pixels below the x-axis, which is `yOffset + 30`. The y coordinate for the y-axis label varies, as per the case for the tick marks on the y-axis. Since we are setting the lower-left corner of a bounding box rather than the endpoint of a line, we use `yOffset + 5- pixelDistance * i` (a slight variation of 5 pixels from the expression for the y coordinate of the tick mark on the y-axis).

To determine the x coordinate, it is necessary to adjust for the number of digits in the label. The following expressions calculate the x coordinate of the text labels on the x-axis and y-axis respectively.

```
xOffset + i * pixelDistance - (4 * (Math.trunc(Math.log10(intervalX * i)) + 1))
xOffset - 20 - (10 * (Math.trunc(Math.log10(intervalY * i)) + 1))
```

Inserting those expressions for the label positions into the calls to the `makeTickMarkLabel` function yields the following.

```
for (i = 1; i <= 10; i++) {
    makeTickMarkLabel(xOffset + i * pixelDistance, yOffset - 10, xOffset + i *
    pixelDistance, yOffset + 10, intervalX * i, xOffset + i * pixelDistance -
    (4 * (Math.trunc(Math.log10(intervalX * i)) + 1)), yOffset + 30); // x–axis
    makeTickMarkLabel(xOffset - 10, yOffset - i * pixelDistance, xOffset + 10,
    yOffset - i * pixelDistance, intervalY * i, xOffset - 20 - (10 * (Math.
    trunc(Math.log10(intervalY * i)) + 1)), yOffset + 5- pixelDistance * i);
    // y-axis
}
```

Insert that loop into the `makeAxes` function at this time; then save and run. The tick marks and labels will appear as expected. However, no labels appear for the origin. If you changed the start value of the loop above to 0, the tick marks at the origin would appear, but no labels would appear. The presence of tick marks at the origin may seem odd because numbers less than zero are never plotted in this app. Further, the labels do not appear because Log10(0) cannot be resolved, so the label cannot be placed anywhere along the x-axis. For labels at the origin, include the following two statements in the `makeAxes` function.

```
canvasContext.fillText(0, xOffset - 5, yOffset + 30);
canvasContext.fillText(0, xOffset - 30, yOffset + 5);
```

Save *scatterPlot.html* and execute the program by reloading the page in your web browser. Tap the Plot button to view the axes.

Part 3. Plotting Points and Displaying the Line of Best Fit

To address to the two fundamental tasks remaining, we will first add code to plot the points. Then we will add code to calculate the line of best fit, which will appear when requested by the user.

Insert the function below into *scatterPlot.html*. Notice that the `plotPoint` function has two parameters, which are the x and y coordinates of the center of the point being plotted. Also note that the `arc` method is used to create the circle representing the point. The `arc` method has five parameters. The first two parameters are the x and y coordinates of the center of the arc (a complete circle in our case). The third parameter is the radius, which in the code below is set to 3 pixels, but you may change the radius if you wish. The fourth and fifth parameters specify the arc's start and end angles in radians. By going from 0 radians to 2 radians, a complete circle is formed. Lastly, the final statement in the `plotPoint` function uses the `fill` method to give the appearance of a point, rather than an empty circle. (An unfilled circle would use the `stroke` method, which in this case would be `canvasContext.stroke()`.)

```
function plotPoint(x, y) {
    canvasContext.beginPath();
    canvasContext.arc(x, y, 3, 0, 2 * Math.PI);
    canvasContext.fill();
}
```

With the `plotPoint` function, we need only call it with each data point. The loop to accomplish this is familiar because we developed it above to calculate the intervals for the x and y axes. As before, the x and y coordinates of each point must be translated in accordance with the x and y axes we positioned in the HTML canvas. In the case of data points, for each x coordinate and y coordinate, we need only calculate the ratio of the coordinate to its interval and then account for the offset of its axis. This is evident below in the parameters of the `plotPoint` function.

```
    for (i = 0; i < dataStrings.length; i++) {
        values = dataStrings[i].split(", ");
        currentX = Number(values[0]);
        currentY = Number(values[1]);
        plotPoint(Math.trunc(currentX / intervalX * pixelDistance + xOffset),
        Math.trunc(yOffset - currentY / intervalY * pixelDistance));
    }
```

Before inserting that loop in the `makeScatterPlot` function, we will add four assignment statements. Two of those statements set the variables `sumDeltaXDeltaY` and `sumDeltaXsq` to zero. The other two assignment statements calculate results needed for the line of best fit. Ultimately, to determine the line of best fit, we calculate the slope, *m*, and the y-intercept, *b*, as follows.

$$m = \frac{\sum_{i=1}^{n}\left(x_i - \overline{X}\right)\left(y_i - \overline{Y}\right)}{\sum_{i=1}^{n}\left(x_i - \overline{X}\right)^2}$$

$$b = \overline{Y} - m\overline{X}$$

In the code below, `sumDeltaXDeltaY` stores the sum of the products of the differences in x from `meanX` and the differences in y from `meanY` for the entire dataset. The variable `sumDeltaXsq` stores the sum of the sqaures of the differences in x from `meanX` for the entire dataset.

Insert the following two functions into *scatterPlot.html*.

```
function makeScatterPlot() {
    console.log("In makeScatterPlot");

    dataString = document.getElementById("data").value;
    dataStrings = dataString.split("\n");

    calculateIntervals(dataStrings);
    makeAxes();

    sumDeltaXDeltaY = 0
    sumDeltaXsq = 0
    for (i = 0; i < dataStrings.length; i++) {
        values = dataStrings[i].split(", ");
        currentX = Number(values[0]);
        currentY = Number(values[1]);

        plotPoint(Math.trunc(currentX / intervalX * pixelDistance + xOffset),
        Math.trunc(yOffset - currentY / intervalY * pixelDistance));

        sumDeltaXDeltaY += (currentX - meanX) * (currentY - meanY);
        sumDeltaXsq += (currentX - meanX) ** 2;
    }

    if (requestLine) {
        displayLineBestFit();
    }
}

function displayLineBestFit() {
    console.log("In displayLineBestFit");
}
```

Even though the app would execute properly without the following two global variable declarations, I recommend adding them after the `<script>` tag at this time.

```
var sumDeltaXDeltaY = 0;
var sumDeltaXsq = 0;
```

Save *scatterPlot.html* and execute the program by reloading the page in your web browser. Tap the Plot button to view the scatter plot. Change the data as you wish and then click the Plot button again. You may also click the Show Line button and view the placeholder text in the JavaScript console.

Since we took care to calculate `sumDeltaXDeltaY` and `sumDeltaXsq` previously, we can use those results now to calculate the slope and the y-intercept, as shown below.

```
slope = sumDeltaXDeltaY / sumDeltaXsq;
yIntercept = meanY - slope * meanX;
```

Lastly, we use the now familiar **moveTo** and **lineTo** methods to draw the line from `(0, yIntercept)` to `(maxX, slope * maxX + yIntercept)`. As before, the x and y coordinates of those two points are translated in accordance with the x and y axes we positioned in the HTML canvas. This is evident below in the parameters of the `moveTo` and `lineTo` functions. Edit `displayLineBestFit` to include the code below.

```
function displayLineBestFit() {
    slope = sumDeltaXDeltaY / sumDeltaXsq;
    yIntercept = meanY - slope * meanX;
    console.log("slope yIntercept");
    console.log(slope, yIntercept);

    canvasContext.beginPath();
    canvasContext.moveTo(xOffset, Math.trunc(yOffset - yIntercept / intervalY
    * pixelDistance));
    canvasContext.lineTo(Math.trunc(maxX / intervalX * pixelDistance +
    xOffset), Math.trunc(yOffset - (slope * maxX + yIntercept) / intervalY *
    pixelDistance));
    canvasContext.stroke();
}
```

Save *scatterPlot.html* and execute the program by reloading the page in your web browser. Tap the Plot button and the Show Line button. Change the dataset as you wish.

The HTML canvas is implemented for both low-resolution and high-resolution screens. Unfortunately, web browsers do not detect the resolution of the screen automatically. Inserting the following code after the declarations of the canvas and canvasContext will remedy the blurry text and images in the canvas.

```
const HTML_CANVAS_WIDTH = 500;
const HTML_CANVAS_HEIGHT = 450;
const SCALE_FACTOR = 2;
canvas.style.width = HTML_CANVAS_WIDTH + "px";
canvas.style.height = HTML_CANVAS_HEIGHT + "px";
canvas.width = HTML_CANVAS_WIDTH * SCALE_FACTOR;
canvas.height = HTML_CANVAS_HEIGHT * SCALE_FACTOR;
canvasContext.scale(SCALE_FACTOR, SCALE_FACTOR);
```

The *scatterPlot.html* file is available in the Electronic Resources for the book.

7.3.6 Developing Python Apps to Visualize Data Using **matplotlib**

In this section we write Python code to represent data in visuals, in two-dimensional graphs and plots first, and then in three-dimensional renderings of functions. In Python, developers use a flexible library of functions called *matplotlib* to present data visually. The visuals may be bar charts, histograms, pie charts, scatter plots, box plots, ellipses, polygons, Bezier curves, SVG paths, Hinton diagrams, images, or 3D surfaces, for instance. In addition, text may be added to visuals and colors specified. Further, the visuals may be static, animated, or interactive.

If you have installed the Anaconda interactive Python environment (as described in the Python example in Section 4.2.5), matplotlib is included in the bundle. Otherwise, installation of matplotlib and its dependencies may be accomplished on macOS, Windows, and Linux by the following two commands.

```
python -m pip install -U pip
python -m pip install -U matplotlib
```

If installation did not complete, try adding `--prefer-binary` above (https://matplotlib.org/3.3.4/users/installing.html).

In the Anaconda environment, the two statements below produce a diagonal line that passes through the following points: (0,0), (1,2), (2,4), (3,6), (4,8). The first statement below loads the plotting library, `matplotlib.pyplot`. Per convention, `matplotlib.pyplot` is imported with the alias, `plt`. Without the alias, `matplotlib.pyplot` would need to prefix every function call in the plotting library.

```
import matplotlib.pyplot as plt

plt.plot([0, 2, 4, 6, 8])
```

Many Python environments require a call to the `show` function to display the plot. The code below includes a call to the `show` function.

```
import matplotlib.pyplot as plt

plt.plot([0, 2, 4, 6, 8])
plt.show()
```

The following examples exclude the call to the `show` function, but if it is needed to display plots in your Python implementation, include `plt.show()` in your code. In the code above, the call to the `plot` function determines the visual representation of the data. In this case, given the submission of only one list of data (i.e., [0, 2, 4 6, 8]), Python assumes those values to be y coordinates for the x coordinates 0, 1, 2, 3, . . ., N − 1, where N is the number of points in the dataset. Commonly, two lists of data are submitted to the plot function in order to make explicit the values for both the x and y coordinates. In addition, a format string often appears as the third parameter in calls to the `plot` function. First, though, insert the # character in front of `plt.plot([0, 2, 4, 6, 8])` in order to treat it as a comment (to prevent its execution) and add the following statement in order to verify that it produces the same plot.

```
plt.plot([0, 1, 2, 3, 4], [0, 2, 4, 6, 8])
```

Now, add the format string, `'o-k'`, as shown below. Running the line below adds a visual representation of each data point and changes the color of the plot to black.

```
plt.plot([0, 1, 2, 3, 4], [0, 2, 4, 6, 8], 'o-k')
```

A format string offers a shortcut approach to the specification of some attributes. In particular, the format string above specifies the `marker`, `linestyle`, and `color` parameters. Verify this by replacing the format string above with the three parameters shown below.

```
plt.plot([0, 1, 2, 3, 4], [0, 2, 4, 6, 8], marker='o', linestyle='-', color='k')
```

With `marker` set to `'o'`, a filled circle appears at each data point. Since `linestyle` was set to `'-'`, the solid line style was used. (The solid line style is also the default line style.) With color set to `'k'`, the plot displayed in black. In addition to single-letter color codes, the `color` parameter can be set using an RGB or RGBA tuple with values in the range 0 to 1, such as `color=(1,0.5,1.0)` or `color=(1,0.5,1.0,0.5)`); a hexadecimal RGB or RGBA string, such as `color='#ff80ff'` or `color='#ff80ff30'`; a color name (or its color letter), either blue (b), green (g), red (r) cyan (c), magenta (m), yellow (y), black (k), or white (e.g., `color='red'` or `color='r'`); or a color name identified in the xkcd color name survey (https://xkcd.com/color/rgb/), such as `color='xkcd:teal'` or `color='xkcd:dull teal'`).

There are four line styles, namely solid, dashed, dash-dot, and dotted, which are activated in a format string as -, --, -., and :, respectively. There are approximately 20 markers, which include the asterisk (*), the plus sign (+), the X (x), a diamond (D), a smaller diamond (d), triangle up (^), and triangle down (v). The complete list of markers appears in the documentation for the `plot` function (https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html). To gain more experience with the effects of format strings, consider replacing the format string in the statement below with each of the following: `'og', '*b', '^', 'd', 'vm', 'o--y', 'o-.', '+:'`, as well as any format strings you wish to create.

If you set the color to white (w), you will need to replace the default white background color in order to view the plot, which you can do using either of the following statements, for instance.

```
plt.axes().set_facecolor('black')

plt.axes().set_facecolor('xkcd:dark magenta')
```

Pattern Seeking in Scatter Plots

Enter the code below to display a scatter plot. Determine whether there is a linear relationship in the fictitious height and weight data. Notice that the last three statements in the code below add a title to the plot, as well as labels for the x and y axes.

```
import matplotlib.pyplot as plt
from statistics import mean
from math import ceil, floor
```

```
height = [66.1, 72.0, 70.1, 67.2, 67.5, 67.7, 70.2, 69.3, 68.4, 65.9, 65.6, 68.2,
68.1, 66.5, 70.2, 70.9, 66.2, 67.9, 70.0, 65.8, 66.9, 67.8, 62.0, 68.3, 65.1,
68.4, 71.1, 68.1, 66.0, 63.4]
weight = [122.1, 140.7, 138.0, 129.5, 134.3, 123.3, 141.9, 135.2, 114.1, 121.3,
128.5, 113.7, 125.2, 119.3, 115.9, 132.1, 128.4, 123.2, 135.7, 121.0, 133.6,
143.5, 105.2, 125.3, 139.8, 133.1, 139.4, 129.9, 110.8, 114.8]

plt.plot(height, weight, 'o')
#plt.scatter(height, weight)

plt.title("Growth Data")
plt.xlabel('Height (in)')
plt.ylabel('Weight (lbs)')
```

Note that *matplotlib* can render a scatter plot with the `scatter` function, or with the `plot` function when the format string contains a marker and no line style. Hence, the following two statements are functionally equivalent.

```
plt.plot(height, weight, 'o')
plt.scatter(height, weight)
```

By default, the y-axis label is rotated 90° and there is little white space (or padding) between the axis label and the tick mark labels. If you wish to change the default settings of those attributes, parameters can be inserted, as shown below.

```
plt.xlabel('Height (in)', labelpad=10)
plt.ylabel('Weight (lbs)', rotation=0, labelpad=40)
```

Lastly, for this plot, note that more than one dataset can be plotted on the same axes. Add the code below to add the line of best fit to the plot. The code below uses the same calculations to calculate the slope and y-intercept of the data used in Section 7.3.5.

```
meanHeight = mean(height)
meanWeight = mean(weight)

sumDeltaHDeltaW = 0
sumDeltaHsq = 0
for i in range(len(height)):
    sumDeltaHDeltaW += (height[i] - meanHeight) * (weight[i] - meanWeight)
    sumDeltaHsq += (height[i] - meanHeight) ** 2

slope = sumDeltaHDeltaW / sumDeltaHsq
yIntercept = meanWeight - slope * meanHeight

minHeight = floor(min(height))
maxHeight = ceil(max(height))

plt.plot([minHeight, maxHeight], [slope * minHeight + yIntercept, slope *
maxHeight + yIntercept])
```

Since many points appear far from the line of best fit, it appears that there is no linear relationship in the fictitious data for height and weight. I encourage you to download free data for height and weight and run this regression analysis again in order to determine whether there is a pattern between height and weight.

Visual Depictions of Population Estimates for the United Kingdom

In *matplotlib*, calling the `bar` function with a list of labels and a list of data will produce a vertical bar graph. To display the bars horizontally, use the `barh` function, as shown below. Optionally, you may include the `color` parameter in the function call. As per Section 7.3.1, the population data were provided by the Office for National Statistics in the UK (https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationestimates/datasets/populationestimatesforukenglandandwalesscotlandandnorthernireland/).

```
import matplotlib.pyplot as plt

countriesUK = ['England', 'Wales', 'Scotland', 'Northern Ireland']
population = [56286961, 3152879, 5463300, 1893667]

plt.figure(figsize=(8,5))

plt.barh(countriesUK, population, color='xkcd:light gray')

for i, v in enumerate(population):
    if v > 50000000:
        plt.text(v -9000000, i -0.05, format(v, ', d'))
    else:
        plt.text(v + 500000, i -0.05, format(v, ', d'))

plt.title('UK Population Estimates\nMiddle 2019', fontweight='bold')
plt.xticks([10000000,20000000,30000000,40000000,50000000,60000000],
['10,000,000', '20,000,000', '30,000,000', '40,000,000', '50,000,000', '60,000,000'])
plt.xlabel('Population', labelpad=10, fontweight='bold')
plt.ylabel('Country', rotation=0, labelpad=40, fontweight='bold')
```

Those population data may also be represented in a pie chart. As shown in the code below, calling the `pie` function with a list of data and a list of labels will produce a pie chart. Run the following code to produce a pie chart that displays the population of the countries in the United Kingdom.

```
import matplotlib.pyplot as plt

countriesUK = ['England', 'Wales', 'Scotland', 'Northern Ireland']
population = [56286961, 3152879, 5463300, 1893667]

plt.pie(population, labels=countriesUK)
plt.legend(bbox_to_anchor=(1,1))

plt.axis('equal')
plt.title('UK Population Estimates\nMiddle 2019')
```

Optionally, to produce an exploded pie slice, provide a list of offset values that specify distances from the center of the pie. When the offset value is greater than 0, the slice will be offset from the center by the value specified, which will create the appearance of an exploded pie slice. As shown below, add the `explode` parameter and set it to `[0.1, 0, 0, 0]` in order to offset the slice for England. The order of offset values corresponds to the order of values in the lists for the data and labels, which provide the data for England first, followed by Wales, Scotland, and Northern Ireland.

```
plt.pie(population, labels=countriesUK, explode=[0.1, 0, 0, 0])
```

After running that program to view the exploded pie slice, change the values in the list to offset the slice for a country (or countries) other than England.

Three-Dimensional Depictions of Functions

Plotting a function renders a visual representation that conveys the purpose of the function. At times, the visual representation may also be regarded as visually appealing, especially when the plots are in three dimensions.

As before, the code below imports `pyplot` from `matplotlib`. In addition, the following two examples import the 3D plotting library, `mplot3d`, as well as the `numpy` library. In the following two examples, `numpy` provides access to a function for calculating the square root of a number, as well as access to sine and cosine functions. Further, the two examples below call the `linspace` and `meshgrid` functions in `numpy`. By default, the `linspace` function returns an array of values evenly spaced between inclusive start and stop values. For example, `linspace(0,10,11)` returns `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` and `linspace(0,10,5)` returns `[0, 2.5, 5.0, 7.5, 10]`. In general, `linspace(start, stop, samples)` returns the `start` and `stop` values with intermediate values cumulatively incremented from the `start` value by `(stop – start) / (samples – 1)`. The `meshgrid` function returns coordinate matrices from coordinate arrays. Consider the following call to the `meshgrid` function in which `numpy` has been imported as `np`.

```
X, Y = np.meshgrid([0,1,2], [0,1,2,3])
```

That call to `meshgrid` yields the following results.

```
X = [[0, 1, 2],
     [0, 1, 2],
     [0, 1, 2],
     [0, 1, 2]]

Y = [[0, 0, 0],
     [1, 1, 1],
     [2, 2, 2],
     [3, 3, 3]]]
```

Those are the exact matrices needed to represent all 12 of the Cartesian coordinates on a plane with x-values, 0, 1, 2 and y-values, 0, 1, 2, 3. Notice that the values of X[0,0] and Y[0,0] form the ordered pair (0,0); X[0,1] and Y[0,1] form the ordered pair (1,0); and X[0,2] and Y[0,2] form the ordered pair (2,0). Also, the values of X[1] and Y[1] form the ordered pairs (0,1), (1,1), and (2,1). Further, X[2] and Y[2] form the ordered pairs (0,2), (1,2), and (2,2), and X[3] and Y[3] form the ordered pairs (0,3), (1,3), and (2,3).

In the two example programs below, after creating matrices of values for x and y coordinates using the combination of `linspace` and `meshgrid` functions, the z coordinate for each pair of (x, y) coordinates is calculated by calling the `zCoordinate` function. That function, in the first case, calculates the z coordinate as $x^2 + y^2$. By commenting that statement out and deleting the comment character (#) in front of the second statement in the `zCoordinate` function, the z coordinate will be calculated as the square root of $x^2 + y^2$. In the third case, the z coordinate is calculated as the sine of the square root of $x^2 + y^2$. Enter the code below; save the file as *plot3Dsurface1.py*; and then run the program as described above to view each plot.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

def zCoordinate(x, y):
    return x ** 2 + y ** 2
    #return np.sqrt(x ** 2 + y ** 2)
    #return np.sin(np.sqrt(x ** 2 + y ** 2))

x = np.linspace(-5, 5, 40)
y = np.linspace(-5, 5, 40)

X, Y = np.meshgrid(x, y)
Z = zCoordinate(X, Y)

ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z, cmap='Blues')
```

After viewing the initial plots, I encourage you to change the range in the `np.linspace` function from -5, 5 to other ranges, such as -6, 6 and -3.1, 3.1, for instance. In addition, you may enjoy the results of different color maps. For instance, in the parameter below, you may replace `Blues` with `autumn`, `PuRd`, `ocean`, or `gist_rainbow`, for instance.

```
cmap='Blues'
```

The following program uses the same fundamental structure as the previous one, but polar coordinates are used in the program below. As a reminder, polar coordinates specify a distance from a reference point, and an angle from a reference direction. In the program below, those points are mapped into x and y coordinates. Then, as before, each z coordinate is calculated as a function of x and y. This program presents four trigonometric functions to be considered one at a time. Enter the code below; save the file as *plot3Dsurface2.py*; and run the program multiple times to view the various plots.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

def zCoordinate(x, y):
    return np.cos(x) + np.cos(y)
    #return -np.cos(x) - np.cos(y)
    #return np.cos(x) + np.sin(y)
    #return np.sin(np.sqrt(x ** 2 + y ** 2))
```

```
r = np.linspace(0, 2, 40)
theta = np.linspace(-0.9 * np.pi, 0.9 * np.pi, 40)
r, theta = np.meshgrid(r, theta)

X = r * np.sin(theta)
Y = r * np.cos(theta)
Z = zCoordinate(X, Y)

ax = plt.axes(projection='3d')

ax.plot_surface(X, Y, Z, cmap='Greens')
```

Once again, after viewing the initial plots, I encourage you to change the range in the `np.linspace` function from 0, 2 to other ranges, such as 0, 3 and 0, 5.8, for instance. Again, you may enjoy changing the color map. I also encourage you to determine how to alter the position of the gap, as well as enclose the gap. Item 9 in the next section draws attention to the line in the code that influences the gap size. Your experimentations may reveal that line before you even consider the next section.

7.4 Challenges and Pursuits

1. To gain additional experience creating and customizing graphs and charts in spreadsheets, leverage data of interest to you to create the following.
   a. Column chart
   b. 3D Pie chart
   c. Donut chart
   d. Histogram
   e. Line graph
   f. Scatter plot
2. Why is it not possible to have more than five regular polyhedra?
3. In *Inkscape*, develop the nets for the octahedron, dodecahedron, and icosahedron. Print and fold the nets. In addition, submit the nets to a laser cutter and then fasten the pieces to construct the polyhedra.
4. In *Adobe XD*, enhance the design of the web app for plotting data by enabling the user to enter a title, as well as labels for the x and y axes. Recommendation: Increase the height of the artboard in order to make space for three text-entry fields.
5. Implement the enhanced design created in the previous item by increasing the size of the HTML canvas in the web app in order to display the additional text-entry fields, as well as increase the size of the plotting space to accommodate the inclusion of the title and x and y axes.
6. In *Adobe XD*, design a web app that enables the user to specify the visual representation to be generated for a particular dataset, which is also entered by the user. For example, the user may select to view a vertical or horizontal bar chart, a line graph, a scatter plot, or a histogram. The user should not be able to make an invalid selection. Hence, when the user seeks to represent categorical data, the scatter plot and histogram options should not be available.
7. Create a web app that implements the design created in the previous item.
8. In *plot3Dsurface1.py* and *plot3Dsurface2.py*, insert additional functions for the z coordinate. For example, you may calculate the z coordinate with the following function.

   ```
   np.cos(np.sqrt(x ** 2 + y ** 2))
   ```

9. In *plot3Dsurface2.py*, the following statement displays a 3D image with a gap. Replace the values -0.9 and 0.9 in order to move the gap, increase or decrease the size of the gap, and eliminate the gap.

   ```
   theta = np.linspace(-0.9 * np.pi, 0.9 * np.pi, 40)
   ```

10. The function np.sin(np.sqrt(x ** 2 + y ** 2)) is used in both *plot3Dsurface1.py* and *plot3 Dsurface2.py*, but the images rendered are different. Can the parameters to Numpy's `linspace` function in each program be changed in order to make *plot3Dsurface1.py* and *plot3Dsurface2.py* render the same image?

7.5 Annotated Resources

GeoGebra
https://www.geogebra.org/

GeoGebra, a combination of **Geo**metry and Al**gebra**, is an open-source project that provides instructional activities, as well as online and mobile device apps, for learning mathematics. More specifically, the instruction and apps pertain to geometry, algebra, arithmetic, statistics, trigonometry, probability, and calculus, and may involve spreadsheets and graphing. There are more that 300 instructional activities for elementary school students and over 2200 instructional activities for middle school and high school students. The online calculator apps are available from the "Start Calculator" button on the Home page. Then, from the dropdown, select either the Graphing, 3D, Geometry, or Computer Algebra System (CAS) calculator. The Resources tab leads to the instructional activities for elementary and secondary school students. This site may be considered without an account. Optionally, one may create an account or sign in using a social media account (for instance) in order to access more resources.

Desmos
https://www.desmos.com/

Like GeoGebra, Desmos offers online apps for engaging in mathematical activities. Under the *Math Tools* drop-down menu, this website provides a graphing calculator, a scientific calculator, a matrix calculator, a simple four-function calculator, and a geometry tool. The geometry app enables the plotting of points, segments, rays, lines, arcs, and figures, which can be transformed (i.e., reflected, translated, rotated, and dilated). The website also offers a preview of their Mathematics curricula for middle school students, which include approximately 160 days of content for each of Grades 6, 7, and 8.

Paul's Online Notes
https://tutorial.math.lamar.edu/

This online book by Dr. Paul Dawkins, a mathematician at Lamar University (Beaumont, Texas), offers notes and problem sets for classes in Algebra, Calculus I, II, and III, and Differential Equations. In the Calculus III class, Dr. Dawkins presents the 3D coordinate system (https://tutorial.math.lamar.edu/Classes/CalcIII/3DCoords.aspx). As part of these notes, Dr. Dawkins presents examples of equations graphed in 2D and 3D space. Clicking the Practice Problems button displays a set of questions, each with a link to its solution. Book chapters are also available in PDF format. In addition to chapters of content, this online book also includes tips for studying math; lists of common math errors; and reviews of fundamental topics in algebra and trigonometry; a tutorial on complex numbers; and summary sheets of mathematical properties, facts, and formulas. According to the Terms of Use (https://tutorial.math.lamar.edu/Terms.aspx), this online book can be used for private non-commercial use.

Matplotlib

https://matplotlib.org/

Matplotlib is a flexible library of functions enabling rapid creation of visuals in Python. Visuals may be two-dimensional graphs or plots, such as bar charts, histograms, pie charts, scatter plots, or box plots. Alternatively, the visuals may be ellipses, polygons, Bezier curves, SVG paths, Hinton diagrams, images, or 3D surfaces, for instance.

This comprehensive website provides installation instructions, documentation in the form of a user guide, examples, and multiple tutorials at the introductory, intermediate, and advanced levels (https://matplotlib.org/3.3.4/tutorials/index.html). Over 300 examples are available, which may be downloaded individually if interested in a particular plot or visualization. Alternatively, all of the examples can be downloaded in one collective set. For the entire set, scroll to the bottom of the web page at https://matplotlib.org/3.3.4/gallery/index.html, or click the particular example of interest and then download the code.

Canva

https://www.canva.com/

This is a general-purpose design site through which one may create postcards, invitations, logos, t-shirts, mugs, and graphs, for instance. Canva's motto is "design anything." STELAR activities in this chapter drew particular attention to some sophisticated tools for graphing. In Canva, a graph may be designed from an empty frame or by modifying a template. To get started, drop down the **Features** menu and select **Graphs & Charts**, or go directly to the chart-making page at https://www.canva.com/graphs/. After entering a title and clicking the **Create my graph now** button, you may pursue either the guided or unguided path to graph creation. When it comes to entering data, click the default bar or pie slice (or other rendering) to manually enter data or copy and paste data from a spreadsheet. Using data of interest to you, create any of the following to represent data in visuals: Bar graph, Comparison chart, Donut chart, Pie Chart, T-Chart, or a Venn Diagram with 3, 4, or 5 circles.

References

Department for Education. (2014). *National curriculum in England: Framework document*. Government of the United Kingdom (gov.uk). https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/381344/Master_final_national_curriculum_28_Nov.pdf

Ministry of Education. (2016). *Grades K–9 learning standards: Mathematics*. Province of British Columbia. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/en_mathematics_k-9_elab.pdf

Ministry of Education. (2018a). *Grade 11 history of mathematics learning standards*. Province of British Columbia. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/en_mathematics_11_history-of-mathematics.pdf

Ministry of Education. (2018b). *Grade 10 workplace mathematics learning standards*. Province of British Columbia. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/en_mathematics_10_workplace-mathematics.pdf

Ministry of Education. (2018c). *Grade 11 workplace mathematics learning standards*. Province of British Columbia. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/en_mathematics_11_workplace-mathematics.pdf

Ministry of Education. (2018d). *Grade 12 apprenticeship mathematics learning standards*. Province of British Columbia. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/mathematics/en_mathematics_12_apprenticeship-mathematics_elab.pdf

National Governors Association Center for Best Practices and Council of Chief State School Officers. (2010). *Common core state standards in mathematics*. http://www.corestandards.org/wp-content/uploads/Math_Standards1.pdf

8 Science

Explorations in Space

8.1 Theme

Thankfully, astronomers and physicists have discovered what is going on in our solar system, at least in some fundamental respects. Critically, the Sun is massive and at the center of our solar system, which formed about 4.6 billion years ago. About 500 years ago Nicolaus Copernicus advanced the heliocentric view, that spherical planets orbit the Sun rather than Earth. Copernicus further held that Earth's circular orbit around the Sun resulted in the cycle of seasons and that Earth's rotation accounts for day and night. Galileo supported heliocentrism, which put him in opposition with the Catholic Church. Ultimately, Galileo's defense of heliocentrism resulted in a trial and he was convicted of "vehement suspicion of heresy" in 1633. Galileo was forced to recant, to state that the Copernican view of heliocentrism is erroneous. Further, Galileo was subjected to house arrest for the remaining nine years of his life, which he spent mostly in his villa (Machamer & Miller, 2021). Meanwhile, Johannes Kepler correctly claimed that planets move along elliptical paths, which contradicted the circular path view held by Copernicus and earlier by Ptolemy. In addition, the oblate (flattened at the poles) spheroid that is Earth wobbles as it spins, as contended by Sir Isaac Newton. In the late 17th century, Newton held that gravitational forces of the Sun and Moon acting on Earth cause it to wobble. That was a unique and remarkable offering for 1687, which he sought to justify scientifically. Even though Newton's revised value for the force of the Moon was still off by a factor greater than two in 1713 (Smith, 2021), Earth's axis of rotation does vary or wobble, albeit slowly, presently taking 25,772 years to complete the cycle. Astronomers, astrophysicists, and cosmologists (those who study the origin of the universe and its physical structure) continue to seek resolution to unanswered questions, such as what causes reversals of Earth's magnetic poles (Cooper et al., 2021; Shah, 2021), as well as make new discoveries, which sometimes lead to revisions of measurements about our solar system and beyond. Parenthetically, our solar system is one component of the heliosphere, which may be regarded as the bubble of space influenced by the Sun (NASA, 2013a). The information in Tables 8.1 and 8.2 about the planets in our solar system is factual due to sourcing from NASA (https://solarsystem.nasa.gov/planet-compare/). More than factual, the data are phenomenal. The data in the tables inform us that we are on a tilted spheroid cruising non-stop at 66,662 miles per hour (107,218 km/h) around the Sun while spinning about 1000 miles per hour (1700 km/h). Enjoy the ride!

The planet comparison web page, https://solarsystem.nasa.gov/planet-compare/, does not include temperatures for Jupiter, Saturn, Uranus, and Neptune. However, on a different web page, NASA (2018) does report average temperatures for those planets as follows: Jupiter -162° F (-108° C); Saturn -218° F (-138 ° C); Uranus -320° F (-195° C); Neptune -331° F (-201° C). For Earth, NASA (2018) lists the average temperature as 61° F (16° C). Since a single value cannot capture the variance in temperature between day and night, nor convey the variability in temperature around the planet during different seasons, an average temperature seems

DOI: 10.4324/9781003172284-11

Table 8.1 Characteristics of Planets

Planet	*Equatorial Radius (miles kilometers)*	*Primary Atmospheric Constituents*	*Surface Temp Fahren. Celsius*	*Surface Gravity (ft/s^2 m/s^2)*	*Rotation Period (Earth Days)*	*Axial Tilt*
Mercury	1516.0 2439.7		279/801 173/427	12.1 3.7	58.646	0.0°
Venus	3760.4 6051.8	Carbon Dioxide Nitrogen	864 462	29.1 8.9	-243.018	177.3°
Earth	3958.8 6371.0	Nitrogen Oxygen	126/136 -88/58	32.0 9.8	0.997	23.4°
Mars	2106.1 3389.5	Carbon Dioxide Nitrogen Argon	-225/70 -153/20	12.2 3.7	1.026	25.2°
Jupiter	43,440.7 69,911.0	Hydrogen Helium		81.3 24.8	0.414	3.1°
Saturn	36,183.7 58,232.0	Hydrogen Helium		34.3	0.444	26.7°
Uranus	15,759.2 25,362.0	Hydrogen Helium Methane		29.1 8.9	-0.718	97.8°
Neptune	15,299.4 24,622.0	Hydrogen Helium Methane		36.6 11.2	0.671	28.3°

unsatisfactory. Further, in the case of the gas giants, Jupiter and Saturn, and the ice giants, Uranus and Neptune, data collected in the late 1970s and the 1980s by the Voyager probes indicated that those planets are much hotter than expected. Subsequent observations have confirmed this. In the case of Jupiter, for instance, one expecting to find temperatures around -162° F (-108° C) would actually find a scorching 615° F (325° C) in the lower latitudes. The cause for this anomaly, friction caused by disparities in wind speed and plasma flow due to auroras, was discovered quite recently, as discussed by Andrews (2021). Jupiter may receive 125 times more heat from that source of friction than from the Sun.

In Table 8.1, the negative rotation values for Venus and Uranus denote rotation opposite the direction of the Sun's rotation (i.e., retrograde rotation). In Table 8.2, the average distance from the Sun in Astronomical Units (AU) for each planet was calculated by dividing the planet's average distance to the Sun by Earth's average distance to the Sun.

In attempts to fathom long distances, such as a million or more kilometers or miles, one may seek to relate long distances to shorter ones. For example, we might better comprehend one million kilometers as a factor of a smaller distance, such as the circumference of Earth at the equator. According to Table 8.1, the radius of Earth is 6371 kilometers. Curiously, a different group at NASA, the Goddard Space Flight Center, regards that equatorial radius as 6378 kilometers (https://imagine.gsfc.nasa.gov/features/cosmic/earth_info.html), which is the radius also identified by the European Space Agency (https://sci.esa.int/web/solar-system/-/35649-earth), but the Canadian Space Agency identifies the equatorial diameter as 12,742 km (https://www.asc-csa.gc.ca/eng/astronomy/solar-system/earth.asp), which makes the radius 6371 km, as per Table 8.1 and the original NASA group. The 6371 km figure appears to be something of a global average since the radius to various points on the surface of Earth varies from 6357 km to 6378 km. For our purposes, we will split the difference to obtain 6374.5 and then multiply by 2 and pi (3.14159) to calculate the circumference as 40,052 km. At 40,052 kilometers

Table 8.2 Planetary Orbits

Planet	*Average Distance from the Sun (miles/km)*	*Average Distance from the Sun (AU)*	*Orbit Inclination*	*Orbit Eccentricity*	*Mean Orbit Velocity (miles/km) per hour*	*Orbit Period (Earth Years)*
Mercury	35,983,125 57,909,227	0.390	7.000°	0.20563593	105,946 170,503	0.2408467
Venus	67,238,251 108,209,475	0.723	3.390°	0.00677672	78,339 126,074	0.6151973
Earth	92,956,050 149,598,262	1.000	0.000°	0.01671123	66,622 107,218	1.0000174
Mars	141,637,725 227,943,824	1.524	1.850°	0.0933941	53,858 86,677	1.8808476
Jupiter	483,638,564 778,340,821	5.203	1.304°	0.04838624	29,205 47,002	11.8626150
Saturn	886,489,415 1,426,666,422	9.539	2.490°	0.05386179	21,562 34,701	29.4474980
Uranus	1,783,744,300 2,870,658,186	19.180	0.770°	0.04725744	15,209 24,477	84.0168460
Neptune	2,795,173,960 4,498,396,441	30.060	1.770°	0.00859048	12,158 19,566	164.7913200

for each rotation, about 25 trips would equal one million kilometers (1,000,000 / 40,052 = 24.97). In miles, that radius is 6374.5 * 0.62137 = 3960.923, which makes each rotation around Earth approximately 24,887.2 miles. Then a little more than 40 trips around Earth accounts for 1,000,000 miles (1,000,000 / 24,887.2 = 40.18). The average distance from Earth to the Sun is about 93 times one million, which would require about 3737 rotations (40.18 * 93 = 3736.74). In contrast, the average orbit distance of Earth's Moon is 238,855 miles or 384,400 km (https://solarsystem.nasa.gov/moons/earths-moon/by-the-numbers/ or https://www.asc-csa.gc.ca/eng/astronomy/solar-system/moon.asp), which is equivalent to fewer than 10 trips around Earth (238,855 / 24887.2 = 9.598). Alternatively, we may conceive of the distance between Earth and the Moon in terms of the diameter of Earth (7922 miles or 12,749 km), which puts the Moon about 30 Earths away (238,855 / 7922 = 30.15).

In further attempts to comprehend long distances, we may also consider travel times to distant places. Race cars, whether Indy, NASCAR, or Formula 1, can go in excess of 200 miles per hour (320 km per hour). If we could drive a race car around the circumference of Earth, 24 hours a day without stopping, we would complete one revolution in a little over 124 hours (24,887.2 / 200 = 124.436) or about five days and four and a half hours (124.436 / 24 = 5.185). If we could drive at that speed to the Moon, we would arrive in about 1200 hours (238,855 / 200 = 1194.275), which is less than 50 days (1194.275 / 24 = 49.76). At that same speed, the trip to the Sun would take about 53 years and 21 days, 208 days for each million-mile segment (1,000,000 / 200 / 24 = 208.333; then 208.333 * 92.95605 / 365 = 53.057; or 92,956,050 / 200 / 24 / 365 = 53.057).

If you can imagine a drive to the Moon or the Sun, you may also imagine traveling at the speed of light, which is nearly 300,000 km per second, or approximately 186,000 miles per second. At that speed, the trip around Earth takes about 13 hundredths of one second (40,000 / 300,000 = 0.133), which would enable you to make about 7 and a half trips around Earth in one second. At the speed of light, you would travel from the Sun to Earth, as light does, in about 8 minutes 20 seconds (93,000,000 / 186,000 = 500 seconds and 500 / 60 = 8.333). Again, travelling at the speed of light, it would take about 4.25 years to reach the star closest to Earth (other than the Sun), Proxima Centauri, because it is approximately 4.25 light years away (NASA, 2020a), which is equivalent to 1.3 parallactic seconds (parsecs). Proxima Centauri is one of perhaps 400 billion stars in our galaxy, the Milky Way, which is approximately 100,000 light years wide. If we could travel at the speed of light, it would take us 100,000 years to cruise across the Milky Way. We might even imagine going to stars beyond our galaxy, such as galaxies more than ten billion light years from Earth (Gohd, 2020; NASA, 2020b; Overbye, 2020).

In light of Einstein's theory of general relativity, which came a dozen years after the 1905 publication of his five "miracle year" papers on molecular dimensions, Brownian motion, special relativity, and transformation of light (Stachel, 1998), travel at the speed of light does not seem possible because an object moving at the speed of light would have infinite mass and zero length. Having sought to fathom long distances, our exploration of space now turns to consideration of facts evident in data returned by various instruments on fly-by probes, such as NASA's Voyager 1 and 2, the only two objects created by human beings in interstellar space (the region beyond the heliopause where the pressures of the solar wind and interstellar wind are in balance (NASA, 2013a, 2021a).

The primary mission of the two Voyager probes was to explore Jupiter and Saturn, including their larger moons, and Saturn's rings. Then, while Voyager 1 was to proceed upward from our solar system's plane toward interplanetary space, Voyager 2 was to explore the two planets farthest from the Sun, Uranus and Neptune. Both probes were launched from Cape Canaveral, Florida in the summer of 1977, and both continue to return data to Earth today. Their current distances from the Sun and Earth appear on the home page of the Voyager website (https://voyager.jpl.nasa.gov/) and are updated each second. Numerous instruments were installed on the probes in the quest to meet their missions.

Each probe is equipped with a Magnetometer (to detect magnetic fields), Ultraviolet and Infrared Spectrometers, and an Interferometer and Radiometer (to detect atmospheric conditions, including temperature and measurements of hydrogen and helium), as well as devices for imaging and detection of particles (to measure particle charge composition in the magnetosphere of planets, for instance). The complete list of instruments and an image of their configuration appears at the Voyager website (https://voyager.jpl.nasa.gov/mission/spacecraft/instruments/), along with descriptions of each instrument and the goals of experiments planned to analyze the data sent back to Earth. Researchers at NASA's Jet Propulsion Laboratory at California Institute of Technology and elsewhere continue to analyze data and publish results of their work (https://voyager.jpl.nasa.gov/news/). For example, data of plasma waves detected in interstellar space document persistent interstellar turbulence rather than the silent and serene environment one might imagine in space (Ocker et al., 2021). In addition to the article, the plasma wave data and example plots are available to the public (https://space.physics.uiowa.edu/voyager/data/). In another study, researchers retrieved and analyzed decades-old data from the Uranus fly-by, made by Voyager 2 in January 1986, to discover new details about the odd planet, which rotates on its side. Apparently, Uranus has a plasmoid in the tail of its wobbly magnetosphere (DiBraccio & Gershman, 2019; Gershman & DiBraccio, 2020; https://voyager.jpl.nasa.gov/news/details.php?article_id=119).

In addition to learning about space from data retrieved by telescopes and instruments on probes, astronauts from a growing number of countries dare to explore space in person. Due in part to successful launches of Sputnik on October 4, 1957, and on November 3, 1957, with a dog on board (https://history.nasa.gov/sputnik-timeline.html; https://history.nasa.gov/sputnik.html), the Soviets developed a program to train people, called cosmonauts, to go into space. The first person in space was Soviet cosmonaut Yuri Gagarin, who made one Earth orbit in 108 minutes on April 12, 1961, before, as planned, jettisoning from the Vostok vessel to return to Earth by parachute (https://www.nasa.gov/mission_pages/shuttle/sts1/gagarin_anniversary.html).

The venture for human beings to take the first steps on a cosmic surface other than Earth was accomplished by Neil Armstrong and Edwin "Buzz" Aldrin after landing the lunar module, Eagle, on the Moon's surface on July 20, 1969 (https://www.nasa.gov/mission_pages/apollo/missions/apollo11.html). Meanwhile, Michael Collins orbited the Moon in the command service module, Columbia, which carried the astronauts and Eagle to an orbit about 69 miles above the surface of the Moon. On the 27^{th} orbit, Eagle reconnected with Columbia. Once Armstrong and Aldrin were back on Columbia, Eagle was released into lunar orbit and Columbia began the journey back to Earth, ultimately, as planned, splashing down near the recovery ship in the Pacific Ocean on July 24, 1969. In total, 12 astronauts have walked on the Moon and over 550 people have been in space. The actual number of people who have traveled to space depends on definitions of where space begins, which is 100 km (62 miles) according to the *Fédération Aéronautique Internationale*, but only 50 miles (about 80.47 km) according to the US Department of Defense. McDowell (2018) considers multiple arguments regarding the boundary between Earth's atmosphere and space, including the original Kármán Line, or what could be called the Kármán-Haley Line of 84 km (52.195 miles). In light of historical, physical, and technological perspectives, McDowell (2018) concludes that 80 km is an appropriate designation for the altitude at which space begins.

In 2021, the James Webb Space Telescope is scheduled to launch, which will enable study of distant galaxies, including study of quasars, in hopes of discovering how galaxies are formed and take shape (https://jwst.nasa.gov/). In 2022, the European Space Agency plans to launch the Jupiter ICy moons Explorer (JUICE) probe, for arrival at Jupiter in 2029 and spend at least three years exploring Jupiter and its moons with ice, namely Ganymede, Callista, and Europa (https://sci.esa.int/web/juice). Also in 2022, in collaboration with the Russian Space Agency, Roscosmos, the European Space Agency plans to launch their first rover as part of their ExoMars project to determine whether life has ever existed on Mars (https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/Exploration/ExoMars). For a history of Mars exploration probes and rovers, see https://mars.nasa.gov/mars-exploration/missions/historical-log/. Other planned space projects include NASA's Artemis program, which has goals to land the first woman and the first person of color on the Moon by 2024 (https://www.nasa.gov/specials/artemis/). Further, with global and corporate partners, the Artemis programs seeks to begin sustained exploration and development of the moon, one component of which is NASA's Gateway project (NASA, 2021b). That research and development work is expected to lead to a human mission to Mars. Collaboration with partners on the International Space Station also continues (https://www.nasa.gov/mission_pages/station/main/index.html). Space tourism is also beginning to become a reality (Whitman Cobb, 2021).

Observations are critical in explorations. As described above, human beings have made direct observations on the Moon on rare occasions. In addition, astronauts on the International Space Station (https://www.nasa.gov/mission_pages/station/main/index.html) observe space directly at times. Yet the vast majority of explorations in space rely on observations collected from sensors on probes and rovers, as well as the various Earth-based telescopes and wave sensors.

Vast quantities of data about space have been and continue to be collected. Fortunately, tens of thousands of datasets about space are freely available for educational purposes and for personal explorations. Source code from various projects is also available. Further, vast quantities of new data from spacecraft projects are collected each day. For example, NASA collects more than four terabytes of new Earth Science data each day from aircraft field experiments (https://www.nasa.gov/open/data.html). In addition to NASA, the Canadian Space Agency and the European Space Agency archive data and make them available to the public without financial cost. See the Annotated Resources section for links to data repositories, which lead to numerous types of data, including images and video.

8.2 Sample Learning Goals

Science curricula for England, British Columbia, Canada, and in the United States of America, as advocated by the National Science Teaching Association, include learning standards pertaining to experimental methodology, laboratory skills, and content knowledge. In addition, communication skills are also included in science programs.

The sample learning goals below, which pertain to space, have been copied from curriculum guides used in England; British Columbia, Canada; and the USA.

England

Science Programmes of Study: Key Stage 3
National Curriculum in England
Department for Education (2013a)

Geography Programmes of Study: Key Stage 3
National Curriculum in England
Department for Education (2013b)

Science Programmes of Study: Key Stage 4
National Curriculum in England
Department for Education (2014)
Key Stage 3 – Forces (p. 11)

- Non-contact forces: gravity forces acting at a distance on Earth and in space, forces between magnets and forces due to static electricity

Key Stage 3 – Space Physics (p. 13)

- Gravity force, weight = mass x gravitational field strength (g), on Earth g=10 N/kg, different on other planets and stars; gravity forces between Earth and Moon, and between Earth and Sun (qualitative only)
- Our Sun as a star, other stars in our galaxy, other galaxies
- The seasons and the Earth's tilt, day length at different times of year in different hemispheres
- The light year as a unit of astronomical distance

Key Stage 3 – Geography (p. 2)

- Physical geography relating to geological timescales and plate tectonics; rocks, weathering and soils; weather and climate, including the change in climate from the Ice Age to the present; and glaciation, hydrology and coasts

Key Stage 4 – Space Physics (p. 224)

- The main features of the solar system

British Columbia, Canada

British Columbia Ministry of Education (2016)
K-9 Learning Standards for Science

British Columbia Ministry of Education (2018a)
Grade 10 Learning Standards for Science

British Columbia Ministry of Education (2018b)
Grade 11 Learning Standards for Earth Science

British Columbia Ministry of Education (2018c)
Grade 11 Learning Standards for Physics

British Columbia Ministry of Education (2018d)
Grade 11 Learning Standards for Geology

Grade 6 Science *(British Columbia Ministry of Education, 2016, p. 12)*

Big Idea: The solar system is part of the Milky Way, which is one of billions of galaxies

- The overall scale, structure, and age of the universe
- The position, motion, and components of our solar system in our galaxy

Grade 7 Science *(British Columbia Ministry of Education, 2016, p. 14)*

Big Idea: Earth and its climate have changed over geologic time

- Evidence of climate change over geological time and the recent impact on humans

Grade 9 Science *(British Columbia Ministry of Education, 2016, p. 18)*

Big Idea: The biosphere, geosphere, hydrosphere, and atmosphere are interconnected, as matter cycles and energy flows through them

- Effects of solar radiation on the cycling of matter and energy ("solar radiation provides the energy required for most life on Earth, and is the root cause of wind and ocean currents, which distribute energy and nutrients around the planet, as well as the energy sources for the water cycle", https://curriculum.gov.bc.ca/curriculum/science/9/core)

Grade 10 Science *(British Columbia Ministry of Education, 2018a, p. 1)*

Big Idea: The formation of the universe can be explained by the Big Bang theory

- Big bang theory, components of the universe over time
- Astronomical data and collection methods

Grade 11 Earth Sciences *(British Columbia Ministry of Education, 2018b, pp. 1–2)*

Big Idea: Astronomy seeks to explain the origin and interaction of Earth and its solar system

- Earth as a unique planet within its solar system due to presence of water, life, protective magnetic field, temperature, and atmosphere (https://curriculum.gov.bc.ca/curriculum/science/11/earth-sciences)
- Stars as the center of a solar system (stellar classification, life cycle, magnitude, brightness, https://curriculum.gov.bc.ca/curriculum/science/11/earth-sciences)
- Impacts of the Earth-Moon-Sun system (e.g., tides, eclipses, seasonal variation, albedo, precession, moon phases, solar winds, https://curriculum.gov.bc.ca/curriculum/science/11/earth-sciences)
- application of space technologies to the study of changes in Earth and its systems

Grade 11 Physics *(British Columbia Ministry of Education, 2018c, pp. 1–2)*

Big Idea: An object's motion can be predicted, analyzed, and described

- Projectile motion (1D and 2D, including vertical launch, horizontal launch, and angled launch, https://curriculum.gov.bc.ca/curriculum/science/11/physics)

Grade 12 Geology *(British Columbia Ministry of Education, 2018d, p. 1)*

Big idea: Earth's geological and biological history is interpreted and inferred from information stored in rock strata and fossil evidence

- The geologic time scale and major events in Earth's history (e.g., formation of oldest rocks, earliest recorded life, domination of invertebrates, first land plants, domination of reptiles, appearance of flowering plants, Rocky Mountain orogeny, mass extinctions https://curriculum.gov.bc.ca/curriculum/science/12/geology)
- The local and global fossil record (e.g., Foraminifera, Mollusca, Brachiopoda, Echinodermata, Arthropoda-trilobites, Coelenterata-corals, Vertebrata, Graptolithina, Conodonta, algae, plants, reptiles, https://curriculum.gov.bc.ca/curriculum/science/12/geology):
  - evidence of evolution
  - methods of fossil formation
  - First Peoples perspectives
- Methods for relative and absolute dating of rocks, fossils, and geologic events
- Reconstruction of Earth's past through correlation of fossil data and rock strata

National Science Teaching Association, USA

According to the National Science Teaching Association, only five U.S. states, Texas, Ohio, Pennsylvania, North Carolina, and Florida, do not have science curricula based on standards created by the National Research Council of the National Academies (https://www.nsta.org/science-standards), as documented in *A Framework for K-12 Science Education* (National Research Council, 2012). That framework identifies Earth and Space Science as a Disciplinary Core Idea (DCI). The Earth and Space Science (ESS) DCI is divided into three topics, Earth's Place in the Universe (ESS1), Earth's Systems (ESS2), and Earth and Human Activity (ESS3, National Research Council, 2012, p. 171). Of particular note here is ESS1, which is subdivided into three components, The Universe and its Stars, Earth and the Solar System, and The History of Earth (National Research Council, 2012, p. 171). The framework provides expectations for student achievement in those three components by the end of grades 2, 5, 8, and 12. Based on my synthesis of that framework (National Research Council, 2012, pp. 174, 176, 178–179), it appears that students should learn the following by the end of Grade 8 and Grade 12.

Grade 8

- The universe originated with what is called the Big Bang, which is marked by a period of extreme and rapid expansion
- Cosmic bodies, such as planets and moons, can be observed and described
- Motions of cosmic bodies can be observed, predicted, explained, and modeled
- A set of planets, including Earth, are in a galaxy called the Milky Way, which is one of many galaxies in the universe
- In the solar system, the Sun's gravitational pull keeps a group of objects, such as planets and asteroids, in orbit around the sun

- The solar system model accounts for tides, eclipses, and the positions of planets relative to the stars
- Earth's axis of rotation is tilted relative to the plane of its orbit around the sun, which accounts for seasons and variance in intensity of light on different regions of the planet
- Rock layers, called strata, provide an accurate method for determining historical events on Earth, such as periods of glaciation, volcanic eruptions, formation of mountain ranges and ocean basins, the continual succession of life through evolution and extinction of organisms
- By analyzing rock strata and fossils, one may determine the order of major events, not absolute dates of the events

Grade 12

- The Sun is a star, which is constantly changing and will burn out in about 10 billion years
- The Milky Way contains more than 200 billion stars, including the Sun
- The Milky Way is one galaxy among hundreds of billions of galaxies in the universe
- By studying the brightness and light spectra of stars, compositional elements of stars, their movements, and distances from Earth can be determined
- Kepler's laws account for motions of orbiting objects, which includes elliptical paths of objects orbiting the sun
- Due to gravitational effects, or collisions of objects in space, orbital paths may change
- In cycles over long time periods, on the order of tens or hundreds of thousands of years, Earth's orbit around the Sun has changed, which has altered the intensity of sunlight on Earth
- Such cycles cause climate changes, sometimes as profound as an ice age
- Some continental rocks on Earth are over four billion years old whereas rocks on the ocean floor are less than 200 million years old
- Rocks on Earth change due to erosion and plate tectonics, which has made the study of rocks on Earth difficult, but rocks on the Moon, as well as on asteroids and meteorites have been largely unchanged over billions of years
- Rocks can be dated based on isotopes of their elements and radioactive decay, which helps cosmologists study the origin of the universe

8.3 STELLAR Activities

The STELLAR activities in this chapter involve looking for patterns in solar system data; creating a 3D model of the solar system; leveraging data to create visuals of interplanetary entities; and designing and implementing a web app to provide insights into space travel.

8.3.1 Seeking Patterns in Solar System Data

First, this section presents a dataset and discovery activities which secondary school students could pursue to enhance their conceptions of our solar system. Then this section provides instructions for creating column/bar charts and scatter plots in *Google Sheets*, which may be used to visualize the data and answer the questions.

First Discovery Activity for Students

The data in Table 8.3 include eight criteria for each of the eight planets in our solar system (https://solarsystem.nasa.gov/planet-compare/ and https://nssdc.gsfc.nasa.gov/planetary/factsheet/). For each criterion, how would you group the planets in order to convey similarities

Table 8.3 Solar System Data

Planet	*Radius (km)*	*Mass (10^{21} kg)*	*Average Distance from Sun (Mm)*	*Mean Orbit Velocity (km/h)*	*Orbit Period (Earth days)*	*Escape Velocity (km/h)*	*Number of Moons*	*Rings*
Mercury	2440	330	57.9	170,503	88	15,300	0	No
Venus	6052	4867	108.2	126,074	225	37,296	0	No
Earth	6371	5972	149.6	107,218	365	40,284	1	No
Mars	3390	642	227.9	86,677	687	18,108	2	No
Jupiter	69,911	1,898,130	778.3	47,002	4331	216,720	79	Yes
Saturn	58,232	568,319	1426.7	34,701	10,747	129,924	83	Yes
Uranus	25,362	86,810	2870.7	24,477	30.589	76,968	27	Yes
Neptune	24,622	102,410	4498.4	19,566	59,800	84816	14	Yes

and differences? Justify your groupings. If you find no basis for grouping planets, describe the uniqueness of each planet with respect to each criterion.

Some students may look at the data and immediately address the question, while other students may prefer to consider visual representations and use them to justify their groupings.

1. In a web browser, open *Google Sheets* (http://sheets.google.com/)
2. Click the Plus icon to open a new (blank) sheet
3. Drop down the **File** menu; select **Import**; click the **Upload** tab; and **Drag** and **Drop** the file, *Solar-System-Data.csv*, on to the File Browser window. When prompted, click the **Import Data** button. (The *Solar-System-Data.csv* file is available in the Electronic Resources for this book.) You may wish to format the title or headings. For example, wrapping the text of the column headings, as in Table 8.3, reveals the full label for the heading and the unit of measurement. To format the headings, select cells A4 through I4; then drop down the **Format** menu; cascade through **Text wrapping**; and select **Wrap**
4. In the spreadsheet, select **Chart** (whether through the **Insert** menu or the tool bar button). For **Chart Type**, select *Column chart* (or *Bar chart*). For *Data Range*, enter `A4:A12,B4:B12`. For X-axis, enter A4:A12. For Series, enter B4:B12
5. For styling, click the **Customize** tab, in *Chart and axis titles*, enter `Size of Planets` for *Chart title*. Then click the drop-down menu for *Chart title* and select *Horizontal axis title*; enter `Planet`. Lastly, in the drop-down menu, replace *Horizontal axis title* with *Vertical axis title*, and enter `Equatorial Radius (km)`
6. The size of the bar for each planet suggests grouping Mercury, Venus, Earth, and Mars, as well as grouping Jupiter, Saturn, Uranus, and Neptune. If the chart covers the data, click the chart to make it the active object; then drag it to move the chart to the side

For other Column (or Bar) charts, I recommend copying and pasting the current chart, and completing the following steps.

1. Edit the chart settings by double-clicking the chart or by selecting Edit Chart from the chart menu, which is displayed by clicking the vertical ellipsis button (⋮) in the upper-right corner of the chart
2. With the **Setup** tab selected, enter different ranges of cells for *Data* and *Series*. For a Column Chart of Planetary Mass; enter `A4:A12,C4:C12` for *Data range* and enter `C4:C12` for *Series* range
3. With the **Customize** tab selected, within *Chart and axis titles*, set *Chart title* to `Mass of Planets` and set Vertical axis to `Mass`

4. Since the mass of the planets varies greatly, it may be necessary to drag the handle in the middle of the bottom line of the bounding box down to increase the height of the Column chart and view the columns of Mercury, Venus, Earth, and Mars

Second Discovery Activity for Students

What relationships exist, if any, between the following features of planets?

- Average distance from the Sun and Mean Orbit Velocity
- Average distance from the Sun and Orbit Period
- Average distance from the Sun and Escape Velocity
- Average distance from the Sun and Number of Moons
- Average distance from the Sun and Presence of Rings
- Average distance from the Sun and Mass
- Mass and Escape Velocity
- Orbit Period and Orbit Velocity
- Orbit Period and Escape Velocity

Again, some students may perceive whether a relationship exists by viewing the raw data. Creating a scatter plot, though, enables one to visually discern any relationship. In addition, spreadsheets often provide a feature to fit a trend line through the data, which may help resolve whether the relationship is linear, exponential, or logarithmic, for instance. The steps below create a scatter plot and a trend line that fit the data for average distance from the Sun and Mean Orbit Velocity, which answers the first question by establishing an exponential relationship between the data.

1. Create a new chart by dropping down the **Insert** menu and selecting Chart or by clicking the Chart button in the tool bar. For Chart Type, select **Scatter**. For *Data range*, enter `D4:D12,E4:E12`. For *X-axis*, enter `D4:D12`. For *Series*, enter `B4:B12`
2. For styling, click the **Customize** tab, in *Chart and axis titles*, enter `Relationship between Orbit Distance and Orbit Velocity` for *Chart title*. Then click the drop-down menu for *Chart title* and select **Horizontal axis title**; enter `Distance from Sun (Mm)`. Lastly, in the drop-down menu, replace **Horizontal axis title** with **Vertical axis title**, and enter `Orbit Velocity (km/h)`
3. Close the settings for **Chart and axis titles** by clicking its label. Open the settings for the next customization feature by clicking **Series**. Scroll down to *Trendline* and click its checkbox. The default trendline is Linear, but that does not fit these data well. In the drop-down menu, replace **Linear** with **Exponential**. Then replace **Exponential** with **Logarithmic**
4. In light of the visual representation of the fit of the data to the logarithmic trendline, students should conclude that the shorter the distance to the Sun, the faster the orbit velocity.

The chart can be copied and pasted, as before, and the data ranges edited in order to determine whether the other relationships exist.

8.3.2 Creating and Displaying Planetary Orbits

In this activity, we will complete the following steps.

1. Derive the equations for the orbits of the eight planets
2. Plot the orbits using GeoGebra
3. Export the image as an SVG file; and
4. Modify the image in *Inkscape* for final presentation

Deriving the Equations of the Planetary Orbits

As Kepler claimed in the early 1600s, each planetary orbit is an ellipse with the Sun at one focus. With knowledge of the length of the semi-major axis (i.e., mean distance of planet to the Sun) and the measure of orbital eccentricity (i.e., the extent to which an ellipse deviates from a circle), we can derive the equation of an ellipse. The data needed are in Columns 2 and 5 of Table 8.2. For convenience, those values have been rounded in Table 8.4. Also, for convenience, we set the center of each ellipse to the origin of the coordinate system. Further, recognizing that the major axis of all eight planetary orbits is horizontal, the equation of each orbital ellipse will have the general form, $\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$, where *a* is the length of the semi-major axis (provided in Table 8.4 in billions of meters, gigameters), *b* is the length of the semi-minor axis. To calculate b, we first calculate *c*, the length from a focus to the elliptical center. To calculate *c*, we use eccentricity (ε), which is provided in Table 8.4. Since $\varepsilon = c/a$, $c = a\varepsilon$. With Table 8.4 in a spreadsheet (provided in the *OrbitCalculations.csv* file in the Electronic Resources for this book), calculate c for Mercury by setting Cell D2 to `=B2*C2`. Then copy the expression in Cell D2 to Cells D3–D9 for the remaining seven planets.

In an ellipse, $b^2 = a^2 - c^2$. Hence, $b=\sqrt{a^2-c^2}$. In the spreadsheet, set Cell E2 to `=SQRT(B2*B2-D2*D2)`. Then copy the expression in Cell E2 to Cells E3 through E9.

For Mercury, $c = 58 * 0.21 = 12.18$ and $b = \sqrt{58^2 - 12.18^2} = 56.71$. Hence, the orbital equation is $\frac{x^2}{58^2}+\frac{y^2}{56.71^2}=1$.

Plotting Orbits in GeoGebra

1. Open a web browser and go to https://www.geogebra.org/calculator
2. Enter $\frac{x^2}{58^2}+\frac{y^2}{56.71^2}=1$ into the field for an equation
   Clicking the keyboard icon in the lower-left corner of the app displays buttons for exponentiation. You may need to press the Enter/Return key after entering the equation and zoom out to view the plot.
3. Repeat the second step seven times. Use the values in your spreadsheet for *a* and *b* for each of the remaining seven planets

If you wish to hide an orbit, click the circle icon to the left of its equation. Click the circle again to make the orbit reappear.

Table 8.4 Data for Calculating Planetary Orbits

	Length of Semi-Major Axis (gigameters)	*Eccentricity*
Mercury	58	0.21
Venus	108	0.01
Earth	150	0.02
Mars	228	0.09
Jupiter	778	0.05
Saturn	1427	0.06
Uranus	2871	0.05
Neptune	4498	0.01

Exporting the Image as an SVG File

1. For each of the eight equations, click the vertical ellipsis (⋮) at the end of the equation. Then click Settings; and uncheck the box for *Show label*. Optionally, you may also select the Color tab and change the color of the ellipse
2. Click the gear icon in the upper-right corner of the grid. Select *Show Axes* to toggle it off. Also, select **No Grid** in the Show Grid drop-down menu
3. Open the hamburger menu (≡) to the left of the web app title, GeoGebra. Click *Download as* and select *SVG image (.svg)*
4. Also, in the hamburger menu, you can select **Save**. The prompt for login credentials can be bypassed by clicking the *Continue without signing in now* link. Then enter a file name (which takes the.ggb file name extension) and click the **Save** button
5. If you wish, you may also click the right edge of the equation entry panel and drag it left in order to create more space for viewing the plot. In the Gear menu, there is also a *Zoom to fit* option. If you select that option, the orbits may look more like ellipses, depending on the shape of your web browser window. The greater the ratio of browser window width to height, the more like an ellipse the orbit appears. Also, the larger the ratio of browser window height to width, the more like an ellipse the orbit appears.

Modifying the Image in Inkscape

1. Open the SVG file in Inkscape. Before presenting the top view of orbital paths, I recommend adding circles to depict the planets with circle sizes scaled to actual sizes, which appear in the second column of Table 8.1. In *Inkscape*, click the circle icon in the left sidebar and then click a color swatch. Next, drag the circle over the orbit and adjust its size accordingly. Zoom as necessary in order to view the inner orbits
2. I also recommend adding a background color by clicking the paint bucket icon; selecting a dark color; and then clicking the white space
3. Save the file

Due to scaling issues, you may wish to create two images, one with the inner four orbits (Mercury, Venus, Earth, and Mars) and one with the outer four planets (Jupiter, Saturn, Uranus, and Neptune).

Rather than use *Inkscape* to modify an SVG file, you may wish to edit the file in a text editor, per Section 3.3.2.

8.3.3 Creating a 3D Model of the Solar System in **Blender**

Acquire a free raster image of earth from Natural Earth by completing the following steps (Plan A) or see Plan B steps if the server at Natural Earth is down.

Plan A

1. Open a web browser and go to https://www.naturalearthdata.com/downloads/50m-raster-data/
2. Click the link labelled, Natural Earth 1
3. Click the Download small size button for the image with Shaded Relief and Water (84.32 MB)
4. Unzip the file, NE1_50M_SR_W.zip
5. In the NE1_50M_SR_W folder, rename the file NE1_50M_SR_W.tif to Earth_land_oceans.tif

Plan B

1. Open a web browser and go to https://gist.github.com/DanielJWood/b71237cc200831ac-f8e637c05ce2c375#file-natural_earth_s3_links-md
2. Click the following link to download the data https://naturalearth.s3.amazonaws.com/50m_raster/NE1_50M_SR_W.zip
3. Unzip the file, NE1_50M_SR_W.zip
4. In the NE1_50M_SR_W folder, rename the file NE1_50M_SR_W.tif to Earth_land_oceans.tif

Create the Sun

1. Open *Blender*
2. Press the X key and then the D key in order to delete the default cube
3. Click the **Add** button; then select **Mesh; UV Sphere**
4. In the Object Properties, set the Scale values for X, Y, and Z to 7.0. Then zoom out
5. Change the default shading of the Viewport from Solid to Material Preview by clicking the **Material Preview** button in the upper-right corner of the Viewport
6. Click the **Object** button and select Shade Smooth
7. In the Properties panel on the bottom right, click the button for **Material Properties**. Then click the **New** button. Click the Yellow dot beside the Base Color label. From the second column of the pop-up window, select **Gradient Texture**. Using the drop-down menu immediately below Base Color: Gradient Texture, replace the default Linear setting with Spherical. Scroll down the Material Properties panel to Emission. Click its color swatch and replace it with orange (e.g., in RGB tab, set Red to 1.0, Green to 0.4, and Blue to 0.0). Below the Emission swatch, you may also change Emission Strength to 7.0, or to a value between 2 and 10, for instance.
8. In the Scene Collection panel (in the upper-right corner), double-click Sphere and replace it with `Sun`. Press Enter
9. Save the file as *solarSystem.blend*

Create the Spheroid for Earth

1. Click the **Add** button; then select **Mesh; UV Sphere**
2. In the Object Properties window, set the X value for Location to 20. You may want to zoom into the Viewpoint and adjust it to focus on the new sphere. Also, in the Object Properties window, retain the **Scale** values for X and Y at 1.0 and set the **Scale** value for Z to 0.98 (since gravity contracts Earth at the poles). Also, for now, set the Y value for Rotation to 23.4°. [Later, we will set this to 156.6° (180-23.4) because the ellipse of the orbital path will be rotated 180° for counterclockwise rotation.]
3. In the Properties panel on the right, click the button for **Material Properties**. Then click the **New** button. Click the yellow dot beside the Base Color label. From the second column of the pop-up dialog, select **Image Texture**. Below Base Color: Image Texture, notice the two buttons, New and Open. Click the Open button; in the File Browser, navigate to the *Earth_land_oceans.tif* file; select it; and click the **Open Image** button. Adjust the Viewport in order to look at the entire sphere
4. Click the Object button and select Shade Smooth. Again, you may wish to adjust the Viewport to look at the entire sphere
5. In the Scene Collection panel, double-click Sphere and replace it with `Earth`. Press Enter/Return
6. Save the file

Create the Animation to Model Earth's Orbit around the Sun

For Earth's orbit around the Sun, we will create an ellipse and configure animation settings such that the Earth will follow the path of the ellipse. The first step is a preparation step; it is necessary to account for the inversion of the elliptical path, which is inverted to create the proper counterclockwise rotation of the Earth around the Sun.

1. In the Object Properties window for Earth, replace the 23.4 in the Y value for Rotation with 156.6. The inversion of the poles is temporary.
2. Zoom out of the Viewport in order to include both the Sun and the Earth
3. Click the **Add** button; then select **Curve**; **Circle**
4. In the Object Properties window, set the **Scale** values for X, Y, and Z to 21. Adjust the Zoom level of the Viewport as necessary to view the entire circle, as well as the Sun and the Earth. Since we need an elliptical path, set the Scale value of X to 22
5. Also, in the Object Properties window, set the Rotation value for Y to 180. (You could try values other than 180 in order to view the effect of this rotation, then set the value for Y back to 180 before proceeding to the next step)
6. In the Scene Collection panel, double-click BezierCircle and replace it with `EarthOrbit`
7. Click **Earth** to select it. Set the X value for Location to 0. In addition, ensure that the Location values for Y and Z are also 0. This temporarily places the Earth inside the Sun, but **Earth** must be at Location 0,0,0 on the X, Y, and Z axes for *Blender* to set Earth on the proper orbital path.
8. In the Properties Panel on the right, with **Earth** still selected, click the button for Object Constraint Properties. Click **Add Object Constraint**. From the Fourth column, select **Follow Path**. Click the field for Target; from the pop-up menu, select **EarthOrbit**. Click the **Animate Path** button. (Notice that Earth's poles have returned to their original positions.)
9. Save the file
10. Click the **Play** button (or press the Space Bar) to view your model of Earth orbiting around the Sun. Once the animation begins, the Play button changes to a Pause button, which enables you to pause the animation (or press the Space Bar again). To restart an animation, click the left-most of the six buttons in the play bar, the set of six buttons in the top-center of the Timeline Panel
11. In the Scene Collection window, click the **Eyeball** button for the **EarthOrbit** to hide the ellipse. Then restart the animation. Also, adjust the Viewport to consider the animation from multiple perspectives. Click the **Eyeball** button again if you wish to view the ellipse again

At this point, the orbit looks odd because the **Earth** is not rotating on its axis. To remedy that, complete the steps below to animate Earth's axial rotation.

1. Raise the Timeline Panel by dragging the boundary between the Viewport and the Timeline Panel up an inch (a couple of centimeters). Ensure that Frame 1 is selected. If Frame 1 is not selected, drag the blue rectangle in the timeline, which is the current frame indicator, to Frame 1, or click the left-most button in the play bar
2. The top-right corner of the Timeline Panel has fields for the Start and End frames. The default Start frame is 1, which is fine, but replace the End frame with 1400
3. Adjust the Zoom level of the Timeline by positioning the mouse cursor over the **Timeline**, which is just below the frame numbers. In order to view all frames from 1 to 1400, zoom out by pressing the Ctrl key and mouse scrolling. Alternatively, if using a trackpad, press the Ctrl key and use a two-finger swipe to zoom out. You may also move the Timeline left and

right using the scrollbar at the bottom of the Timeline Panel, or with mouse movement or two-finger swipes without the Control key pressed.

4. In the Viewport, click **Earth** to select it
5. In the Timeline panel, click the **Record** button (the button with the black circle immediately to the left of the play bar, also called the *Auto Keying* button). Upon clicking this button, the button's background turns blue, and the circle turns white.
6. Move the mouse cursor over the Viewport and press the I key in order to display the **Insert Keyframe** menu; select **Rotation**
7. Go to the 1400th frame by clicking the right-most button in the play bar. (You can use the left or right arrow key to get to Frame 1400 if, instead of following the direction, you clicked near the number 1400 on the Timeline, but not right on it.)
8. In the Object Properties window, change the Z value for **Rotation** to -16800. The negative value rotates Earth in the proper direction and the number results in a rate of rotation suitable for this simulation. After some test runs, you may redo this animation with a different number for either faster or slower spin.
9. Once again, move the mouse cursor over the Viewport. Then Press the I key to display the **Insert Keyframe** menu and select **Rotation**
10. Click the **Record** button again, this time to turn it off
11. Save the file
12. Click the **Play** button to view your model of Earth orbiting the Sun and spinning on its axis. Play and pause the animation as you wish. Alternatively, you can ***scrub*** the animation by dragging the blue frame marker back and forth.
13. You may wish to adjust the speed at which Earth orbits the Sun, as well as adjust its starting position. Two parameters work in tandem to do this. For example, with Frame 1 selected and **Earth** selected, set Offset to 650 in the Object Constraint Properties. Then select the **EarthOrbit** and in the Object Data Properties, set Frames to 700. In the Object Data Properties window, you may need to expand (by clicking on the triangle), the properties for Path Animation. You may experiment with different values for Frames and Offset. I chose to set the Offset to 650 and the Frames to 700 in order to make the Earth orbit around the Sun appear to occur slowly (since it takes one year for a complete orbit) and to position the Earth at its location on July 25, 2021, a little past the summer solstice in the Northern Hemisphere. You may notice that the North Pole is closer to the Sun than the South Pole on that date and remains so until the autumnal equinox in the Northern Hemisphere.
14. Click the **Sun** and set the X value for Location to 0.75 because the Sun is at one focus of the ellipse, not at its center.
15. Save the file

Lastly, in a simplified manner, I recommend modeling the Sun's rotation on its axis. The speed of the Sun's rotation varies, with the fastest rotation at the equator and the slowest rotation at the poles (NASA, 2013b). Representing the Sun as one solid sphere, as we have, does not permit variable rotation speeds, but we can model a single rate of rotation. In *Blender*, with the Sun selected and the Timeline at Frame 1, repeat Steps 5 through 10 above, but when repeating Step 8, change the Z value for Rotation to 3000 (a slower rotation than Earth). Save the file; play and pause as you wish.

Modeling other planets or moons is left as an exercise.

8.3.4 Designing a Web App for Space Travel

When designing an app for space travel, one may include a variety of features organized into multiple parts. For example, the app could include a section pertaining to space vehicles, which could be divided into rocket structure (e.g., nose, body tube, and propulsion system) and electronic control systems. Another section of the app might facilitate the planning of excursions to specific destinations, which would include a mission statement. For example, the overall mission might be to go to the Moon and return to Earth. Additional goals for the mission might involve conducting particular experiments. Another, more ambitious, mission might first go to the Moon and then proceed to Mars before returning to Earth. A third section of the app might pertain to astronaut training, a fourth section might concern fund raising, and a fifth section might contain tools for space calculations. We will proceed to design that app.

The web app will include five tabs, namely *Rocketry, Mission, Training, Funding*, and *Tools*. Within the *Tools* tab, we will design a calculator to determine escape velocity. In order to travel into space, building or acquiring a rocket capable of attaining the escape velocity of planetary bodies seems like a good idea. Calculations of escape velocity are based on Newton's Universal Constant of Gravitation, which might not actually be entirely universal (Quinn & Speake, 2014), but experimental results have determined the constant to be approximately 6.67 X 10^{-11}. Escape velocity (v_e), the minimum speed needed to escape the gravitational field of a celestial body, is given by $v_e = \sqrt{\frac{2GM}{r}}$, where G is Newton's universal constant of gravitation, M is the mass of the celestial body from which to escape, and r its radius (European Space Agency, n.d.; Farage, n.d.). Table 8.3 provides escape velocities for the eight planets in our solar system, but our calculator will determine the escape velocity for any celestial body, given its mass and size (equatorial radius). The *Tools* tab will also include an orbit calculator.

Figure 8.1 presents the overall structure of the web app in gray scale, but we will design the app with a blue color palette. Figure 8.2 displays the Tools menu bar, which has tabs for the two calculators. Figure 8.3 displays the escape velocity calculator with input and output values for Earth.

First, complete the steps below to create the visual design depicted in Figure 8.1.

1. Open *Adobe XD*
2. Create a new custom size artboard with height 900 pixels and width 700 pixels
3. Double-click the default artboard name and replace it with `SpaceTravelHome`
4. Select the **Rectangle** tool from the left sidebar and create a rectangle in the upper-right corner with height 100 pixels and width 700 pixels
5. In the Properties dialog, click the color swatch for **Fill** and set the Hex value to #0077CC. Uncheck the box for **Border** to eliminate the default border
6. Select the **Text** tool from the left sidebar and enter the title, `Space Travel`. Click the color swatch for **Fill** and choose white by setting the Hex value to #FFFFFF. Set the font to

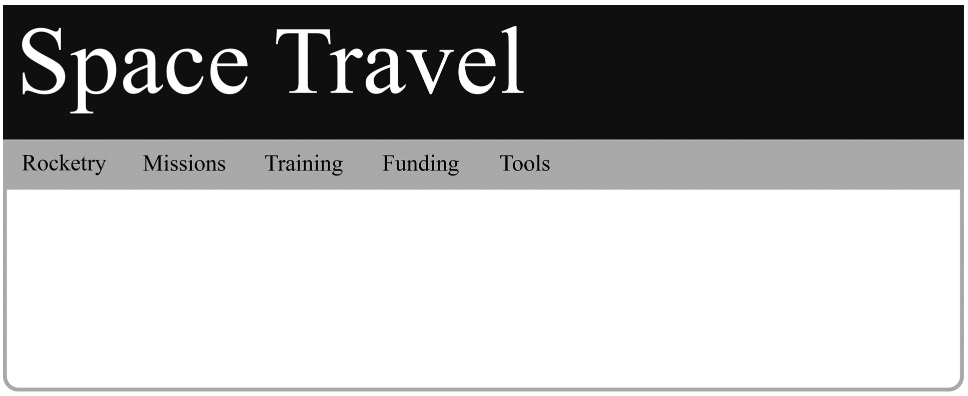


Figure 8.1 Design of web app for space travel

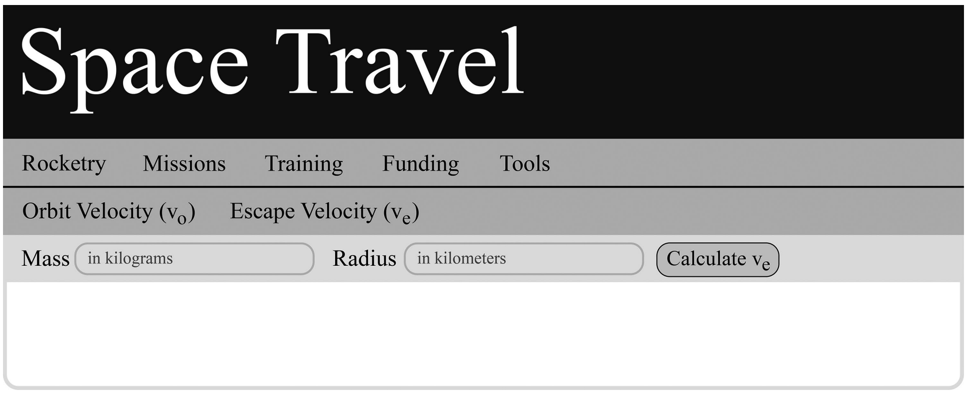
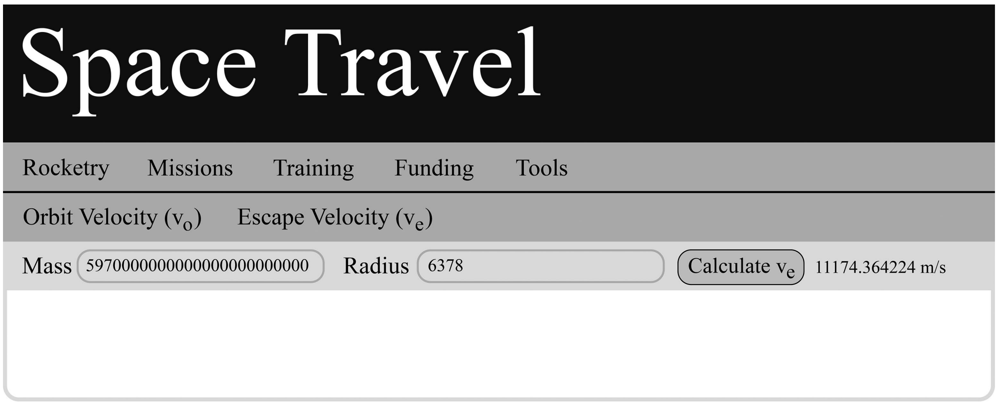


Figure 8.2 Design of Tools menu and escape velocity calculator



Mass 5970000000000000000000000 Radius 6378 Calculate v_e 11174.364224 m/s

Figure 8.3 Earth's escape velocity calculated and displayed

Times New Roman and the font size to 70. For the bounding box of the text, set x to 10 and y to 5

7. Save the file as *spaceTravelApp.xd*
8. Select the **Rectangle** tool from the left sidebar and create a rectangle with height 40 pixels and width 700 pixels immediately below the header
9. Click the color swatch for **Fill** and set the Hex value to #BBDDFF. Uncheck the box for **Border**
10. Select the **Text** tool from the left sidebar and enter the label for the first menu item, `Rocketry`. Click the color swatch for **Fill** and set the Hex value to #505050. The font should still be Times New Roman; set the font size to 17. For the bounding box of the text, set x to 15 and y to 110. Also, ensure that the width is set to 70
11. **Copy** and **Paste** the Rocketry text label. Hold the Shift key down and press the right arrow 9 times (or set x to 105). Replace Rocketry with `Missions`
12. **Copy** and **Paste** the Missions text label. Hold the Shift key down and press the right arrow 9 times (or set x to 195). Replace Missions with `Training`
13. **Copy** and **Paste** the Training text label. Hold the Shift key down and press the right arrow 9 times (or set x to 285). Replace Training with `Funding`

14. One last time: **Copy** and **Paste** the Funding text label. Hold the Shift key down and press the right arrow 9 times (or set x to 375). Replace Funding with `Tools`
15. Save the file

You may wish to select a different color palette and to make other design choices. Indeed, I encourage redesign. See Items 6, 7, & 9 in the Challenges and Pursuits section for particular redesign options.

Complete the steps below in order to implement the Tools menu and the initial state of the escape velocity calculator, as shown in Figure 8.2.

1. Select the existing artboard by clicking its name, `SpaceTravelHome`. **Copy** and **Paste** the artboard. Rename the new artboard, `SpaceTravelTools`
2. Click the rectangle that serves as the background for the menu items. **Copy** and **Paste** it. Hold the Shift key down and press the Down arrow four times (or set y to 140)
3. Select the **Line** tool from the left sidebar and extend a line across the artboard. The line should be 1 pixel in height and 700 pixels in width. If the line is not positioned perfectly (a likely occurrence), you may set x to 0 and y to 140. Click the **Border** color swatch and set the Hex value to #0077CC
4. Select the Rocketry menu item. **Copy** and **Paste** it. Hold the Shift key down and press the Down arrow 4 times (or set y to 150). Replace *Rocketry* with *Orbit Velocity (vo)* Highlight the single letter o in (vo) and click the subscript button (T_1), which is in the Properties dialog below the alignment buttons
5. **Copy** and **Paste** *Orbit Velocity* (v_o). Hold the Shift key down and press the right arrow 16 times (or set x to 175). Ungroup the object and replace Orbit with `Escape`. Also, replace the lowercase o with lowercase e to form *Escape Velocity* (v_e)
6. Click the Prototype tab
7. In the *SpaceTravelHome* artboard, select the Tools menu item. **Drag** and **Drop** its handle anywhere over the *SpaceTravelTools* artboard
8. In the *SpaceTravelTools* artboard, select the title, Space Travel. **Drag** and **Drop** its handle anywhere over the *SpaceTravelHome* artboard
9. Save the file
10. Click the **Play** button to simulate app execution. Test both buttons. Close the simulator (which *Adobe* calls Desktop Preview)
11. Click the **Design** tab
12. Click the *SpaceTravelTools* artboard. **Copy** and **Paste** it. Replace the default name of the new artboard with `SpaceTravelToolsEV`
13. In the *SpaceTravelToolsEV* artboard, select the rectangle that serves as the background for the Tools menu. **Copy** and **Paste** it. Hold the Shift key down and press the Down arrow key 4 times. Click the color swatch for **Fill** and set the Hex value to #DDDDDD. This gray rectangle serves as the background for the escape velocity calculator
14. Over the gray rectangle created in the previous step, add the text label `Mass` with font size 17 in Times New Roman. The width of the text label should be 40 pixels and its height 19 pixels. To position this text label, set x to 15 and y to 192
15. For the input text field for mass, add a **Rectangle** 160 pixels wide with height 25 pixels. To position this rectangle, set x to 55 and y to 188. To round the corners, replace the default corner radius of 0 with 8. (In the Properties window, the corner radius icons and the corner radius field are immediately above the Fill color swatch.)
16. Add the text label, `in kilograms` with font size 12 in Times New Roman. Set the width of this text label to 150 pixels and, if necessary, set its height to 14 pixels. To position this text label, set x to 60 and y to 194

17. Click the `Mass` text label to select it; also select, by holding the Shift key down, the rectangle created in Step 15 and the `in kilograms` text label. Release the Shift key. Copy and paste the three selected items. Hold the Shift key down and press the right arrow 22 times (or set x to 235). Select the copied Mass text label and replace it with `Radius`. Set the width of this text label to 150. Move the copied rectangle and the copied `in kilograms` text label 10 pixels to the right. Replace grams with `meters` to make the new text label `in kilometers`
18. To make the button, create a rounded corner rectangle, 100 pixels wide and 25 pixels in height; set **Fill** color to Hex value #BBDDFF and set the 2-pixel border color to Hex value #0077CC. To position this button, set x to 460 and y to 188. For the button label, create the text label, `Calculate` v_e in Times New Roman with font size 17. This text label should be 90 pixels wide and 24 pixels in height. To position this text label, set x to 466 and y to 189. With the label now appearing inside the rounded rectangle, select both the text label and its enclosing rounded rectangle; **Group** the objects (e.g., drop down the **Object** Menu and select **Group**)
19. Save the file
20. Click the **Prototype** tab
21. In the *SpaceTravelTools* artboard, select *Escape Velocity* (v_e); **Drag** and **Drop** its handle over the *SpaceTravelToolsEV* artboard
22. In the *SpaceTravelToolsEV* artboard, select the title, Space Travel. **Drag** and **Drop** its handle over the *SpaceTravelHome* artboard
23. Also, in the *SpaceTravelToolsEV* artboard, select the Tools menu items. **Drag** and **Drop** its handle over the *SpaceTravelTools* artboard
24. Save the file
25. Click the **Play** button to simulate app execution. Test the three new buttons. Close the simulator

Complete the steps below to implement the Escape Velocity calculator shown in Figure 8.3.

1. In the *SpaceTravelToolsEV* artboard, click the `in kilograms` text label to select it. **Copy** and **Paste** it. Replace the copied text with `5970000000000000000000000` (that's 597 and 22 zeros). You may want to lower the text field temporarily (e.g., by holding the shift key and pressing the Down arrow key five times) in order to edit the text without simultaneously viewing the original `in kilograms` text label over top of the text label you are trying to edit. If you use that technique, raise the field again to its original position after editing. Another strategy is to view the two text fields in the Assets panel (though *Adobe* calls it the Layers panel), which is opened by clicking the Layers button (the middle of the three buttons in the lower-left corner of the *XD* window). Once the Layers (Assets) panel is open, click any single asset to select it. Hover the mouse cursor over the top of any asset to reveal buttons, including the eyeball button, which is used to hide or show the asset. This comes in handy in the next step.
2. In the Layers (Assets) panel, click to select the `in kilograms` text label; hold the Ctrl key (or Command key on a Mac) down and click the `5970000000000000000000000` text label in order to select it as well. Drop down the **Object** menu and select **Make Component**
3. Since *Adobe XD* does not (yet) enable simulation of text field entry without a third-party plugin, we will implement a tap (click) gesture to replace the `in kilograms` text label with the `5970000000000000000000000` text label. The Layers (Assets) panel now shows the two text labels enclosed in a component. In the Layers (Assets) panel, click the diamond icon to reveal the assets included in the component. On the other side of the *Adobe XD* window is the Properties panel. With the newly created component selected, notice the Default State entry in the Properties panel, and to its right notice the + sign. Click the + sign and select **New State**. Replace the default State name (State 2) with `Data Entered State`

4. Currently, the text labels appear in both states of this component. To make this component functional, the default state must show only the `in kilograms` label and the Data Entered State must show only the `5970000000000000000000000` text label. We will accomplish this by hiding one of the two text labels in each state and ensuring that the text labels are ordered properly in the Layers (Assets) panel. In the Properties panel, click *Default State* to select it. In the Layers (Assets) panel, if the assets for the component do not appear, click the diamond icon. In the Layers (Assets) panel, click *5970000000000000000000000*. Then, in the properties panel, click the **Fill** swatch and set the Hex value to #FFFFFF. Now, in the default state, only the `in kilograms` text appears.
5. In the Layers (Assets) panel, click the component in order to select it and ensure that its two assets are visible in the Layers (Assets) panel. In the Properties Panel, click *Data Entered State* to select it. In the Layers (Assets) panel, click the diamond icon and then click *in kilograms*. In the Properties panel, click the **Fill** swatch and set the Hex value to #FFFFFF. In the Layers (Assets) panel, click *5970000000000000000000000* and drag it above *in kilograms*. With *5970000000000000000000000* still selected in the Layers (Assets) panel, in the Properties panel, click the **Fill** swatch and set the Hex value to #000000. Now, in the *Data Entered State*, only the `5970000000000000000000000` text appears.
6. In the Layers (Assets) panel, click the component in order to select it. In the Properties panel, click the *Default State* to select it. Click the **Prototype** tab to configure the settings, which will make the component functional
7. In the Properties panel, there is a section labeled **Interaction**. In the Interaction section, click the + sign. Ensure that the **Trigger** drop-down menu is set to **Tap**. Also, ensure that the drop-down menu for Action type is set to Transition. In the drop-down menu for Destination, select *Data Entered State*
8. In the Properties panel, click the *Data Entered State* to select it. In the **Interaction** section, click the + sign. Ensure that the **Trigger** drop-down menu is set to **Tap**. Also, ensure that the drop-down menu for Action type is set to Transition. In the drop-down menu for Destination, select *Default State*
9. Save the file
10. Click the **Play** button to simulate app execution. Test the functionality of the component. Close the simulator
11. Click the **Design** tab. To make the input field for Radius functional, you may repeat Steps 1 through 8, but 6378 will be the second text label, rather than *5970000000000000000000000*, and interpret all references to *in kilograms* as references to *in kilometers*. Alternatively, you may delete the `in kilometers` text label; select the component you just created; copy and paste it; then hold the Shift key down and press the right arrow key 23 times (or set the x value to 290). In the Properties panel, select Default State. Then, in the artboard, replace `in kilograms` with `in kilometers`. Select the component once again; this time, in the Properties panel, select Data Entered Sate. Then, in the artboard, replace `5970000000000000000000000` with `6378`
12. Save the file
13. Click the **Play** button to simulate app execution. Test the functionality of the new component. Close the simulator
14. To make the *Calculate v_e* button functional, first create the `11174.364224 m/s` text label in 15-point Times New Roman. To position this text label, set x to 576 and y to 193
15. Select both the *Calculate v_e* button and the text label created in the previous step. Drop down the **Object** menu and select **New Component**. In the Properties panel, click the + sign beside Default State; select **New State**; and name it `Result State`
16. In the Properties panel, click *Default State* to select it. In the artboard, double-click the `11174.364224 m/s` text label. In the Properties panel, click the **Fill** swatch and set the Hex

value to #DDDDDD. This hides the result initially. Select the component again (whether through the artboard or the Layers/Assets panel)

17. In the Properties panel, click *Result State* to select it. In the artboard, double-click the hidden `11174.364224 m/s` text label (or select the text label through the Layers/Assets panel). In the Properties panel, click the **Fill** swatch and set the Hex value to #000000. Select the component again (whether through the artboard of the Layers/Assets panel)
18. In the Properties panel, click *Default State* to select it. Click the **Prototype** tab. In the **Interaction** section, click the + sign. Ensure that the **Trigger** drop-down menu is set to **Tap**. Also, ensure that the drop-down menu for Action type is set to Transition. In the drop-down menu for Destination, select *Result State*
19. Save the file
20. Click the **Play** button to simulate app execution. Test the functionality of the button, and the app overall. Revise as necessary and as you wish

Redesign the user interface and functionality of the app as you wish. As noted above, Items 6, 7, & 9 in the Challenges and Pursuits section recommend consideration of particular design options.

8.3.5 Developing the Web App for Space Travel

We will develop this three-page web app in three phases, one phase for each page. Using a combination of HMTL and CSS, we will first create the header and the menu bar. Second, for the Tools menu page, we will add a few lines of HTML and CSS, which will display two buttons and enable the Escape Velocity button; and in the third web page of the app, we will complete the app by adding HTML and CSS to display the necessary input fields and one button. In addition, we will implement the JavaScript function that calculates and displays the escape velocity.

Part 1. Create the Header and Menu Bar

To implement the web app design in Figure 8.1, we will first create the header background and title. Then we will add the menu bar.

In your text editor, enter the HTML and CSS below; then save the file as *spaceTravelApp.html*.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <style>
        body {
            margin: 0px;
        }

        .header {
            height: 100px;
            font-size: 70px;
            color: #FFFFFF;
            background-color: #0028CC;
            margin-top: 0px;
            margin-bottom: 0px;
            padding-left: 10px;
        }
    </style>
</head>

<body>
    <div class="header">Space Travel</div>
</body>
</html>
```

After saving the file, open *spaceTravelApp.html* in a web browser (e.g., double-click the file icon). Notice the effects of the styling directives in the `header` class. The title, Space Travel, appears rather large (`font-size: 70px`) and white (`color: #FFFFFF`) against a 100-pixel high (`height: 100px`) dark blue background (`background-color: #0028CC`).

To implement the menu bar, insert the following CSS after the closing brace of the `header` class definition and before `</style>`.

```
        ul {
            list-style-type: none;
            margin: 0px;
            padding: 0px;
            overflow: hidden;
        }

        li {
            float: left;
        }

        li a {
            display: block;
            color: #505050;
            padding: 10px 15px;
            text-decoration: none;
        }
        li a:hover {
            background-color: #66BBFF;
        }

        .menuBar {
            background-color: #BBDDFF;
        }
```

Also, insert the following HTML after the `</div>` and before `</body>`.

```
    <div class="menuBar">
        <ul>
            <li><a href="">Rocketry</a></li>
            <li><a href="">Missions</a></li>
            <li><a href="">Training</a></li>
            <li><a href="">Funding</a></li>
            <li><a href="toolsMenu.html">Tools</a></li>
        </ul>
    </div>
```

The CSS directives above to style the `ul` and `li` elements, along with the `menuBar` background color (Hex value `BBDDFF`) create a horizontal menu bar with a light blue background. Additional styling for the anchor tags within list elements, defined in the `li a {... }` section, adds padding around the menu items and, with `text-decoration` set to `none`, eliminates the underline that appears beneath hyperlinks by default. Setting the `background-color` to Hex value 66BBFF in the `li a:hover` styling directive adds a little polish to the user interface and serves to identify Rocketry, Missions, Training, Funding, and Tools as menu selections. In this early development stage, only the Tools page will be implemented. You may develop this app further, as you wish; some recommendations for subsequent design and development appear in the Challenges and Pursuits section.

Next, we will create a web page to implement the submenu for the Tool calculators. In accordance with the design established in the previous section, the Tools submenu contains two items, Orbit Velocity (v_o) and Escape Velocity (v_e).

Part 2. Create the Tool Bar

First, make a copy of *spaceTravelApp.html* and name the new file *toolsMenu.html*. Open *toolsMenu.html* in your text editor. Insert the following CSS after the styling directives for the `header` class and before the styling directives for the `ul` tag.

```
.header a {
   color: #FFFFFF;
   text-decoration: none;
}
```

Then, edit the first line of the HTML body to make it appear as follows.

```
<div class="header"><a href="spaceTravelApp.html">Space Travel</a></div>
```

Save *toolsMenu.html* and test the page. You should be able to navigate between the two web pages by clicking the *Tools* menu item in *spaceTravelApp.html* and the title, *Space Travel*, in *toolsMenu.html*.

To implement the Tool bar, first insert the following CSS after the `menuBar` class and before `</style>`.

```
.submenuBar {
   background-color: #BBDDFF;
   border-top: 1px solid #0077CC;
   height: 38px;
}
```

Then insert the following HTML immediately before the `</body>` tag.

```
<div class="submenuBar">
   <ul>
      <li><a href="">Orbit Velocity (vo)</a></li>
      <li><a href="EVcalculator.html">Escape Velocity
         (ve)</a></li>
   </ul>
</div>
```

Margins in your text editor are likely set by default to display the second list item (`<li>`) above as one line, which is preferred and shown below.

```
<li><a href="EVcalculator.html">Escape Velocity (ve)</a></li>
```

To resolve any questions that you may have about indenting and other formatting of CSS and HTML, refer to the source files (*spaceTravelApp.html*, *toolsMenu.html*, and *EVcalculator.html*), which are available in the Electronic Resources for this book.

Save *toolsMenu.html* and reload the web page to ensure that the *Tools* submenu is displayed. Next, we will implement the escape velocity calculator.

Part 3. Enable Numeric Input and Calculate Escape Velocity

First, make a copy of *toolsMenu.html* and name the new file *EVcalculator.html*. Open *EVcalculator.html* in your text editor. Insert the following CSS after the `submenuBar` class and before `</style>`.

```
.calculatorBar {
    background-color: #DDDDDD;
    padding-top: 10px;
    padding-bottom: 10px;
}

label {
    margin-left: 20px;
}

input[type=text] {
    font-family: Verdana, sans-serif;
    font-size: 12px;
    padding: 5px;
    border: 1px solid #AAAAAA;
    border-radius: 7px;
    resize: vertical;
    width: 250px;
}

input[type=text]:focus {
    outline-width: 0;
    border: 1px solid #0077CC;
}

button {
    font-family: Verdana, sans-serif;
    font-size: 12px;

    margin-top: 0px;
    margin-left: 15px;
    height: 26px;
    background-color: #BBDDFF;
    color: #505050;
```

```
            padding: 3px;
            border: 2px solid #0077CC;
            border-radius: 10px;
            cursor: pointer;
        }

        .resultSpace {
            display: inline-block;
            margin-left: 12px;
        }
```

Also, insert the following HTML immediately before the `</body>` tag.

```
<div class="calculatorBar">
        <label>Mass</label>
        <input type="text" id="mass" placeholder="in kilograms">

        <label>Radius</label>
        <input type="text" id="radius" placeholder="in kilometers">

        <button type="button" onclick="calculateEV()">Calculate
            ve</button>

        <div class="resultSpace" id="resultSpace"></div>
    </div>
```

Save *EVcalculator.html* and test the app. Clicking the Escape Velocity (v_e) tab in the Tool bar should display the Escape Velocity Calculator, by loading the *EVcaculator.html* page, and you should be able to navigate between all three web pages. The CSS includes many styling directives we have encountered before (e.g., `color`, `background-color`, `margin-top`, `margin-bottom`). The HTML above creates two pairs of labels and input fields, one for mass, the other for radius. The HTML above also creates a button, which will execute a function called `calculateEV` when clicked by the user. In addition, the HTML above designates a space on the web page for displaying the result of the escape velocity calculation. To make the calculator function properly, implement the `calculateEV` function by inserting the following JavaScript immediately before the `</body>` tag.

```
    <script>
        var radiusField = document.getElementById("radius");
        radiusField.addEventListener("keyup", function(event) {
            if (event.keyCode === 13) {
                event.preventDefault();
                calculateEV();
            }
        });

        function calculateEV() {
            const gravityConstant = 6.67 * Math.pow(10,-11);

            document.getElementById("resultSpace").innerHTML = "";
        mass = Number(document.getElementById("mass").value);
        radius = Number(document.getElementById("radius").value) *
           1000;

        if (mass > 0 && radius > 0) {
           escapeVelocity = Math.sqrt(2 * gravityConstant * mass /
              radius);
           document.getElementById("resultSpace").innerHTML =
              escapeVelocity.toFixed(6) + " m/s";
        }
        else {
           alert("Both mass and radius must be greater than zero");
        }
     }
  </script>
```

The first seven lines in the JavaScript code above, which include the variable declaration for `radiusField` and the creation of a keyboard event listener, are optional. For convenience, after entering a number into the *radius* field, users may prefer to press the Enter/Return key rather than click (or tap) the *Calculate* v_e button.

Calculation of escape velocity occurs in the `calculateEV` function. After declaring `gravityConstant`, clearing the `resultSpace` of any prior calculation, and retrieving the mass and radius from their respective input fields, the code in the `calculateEV` function checks for numeric input values greater than zero. If the input values for both mass and radius are greater than zero, the escape velocity is calculated and displayed. Otherwise, a message alerts the user to the invalid input.

Enjoy refining the app and creating stellar lessons!

8.4 Challenges and Pursuits

1. Continue the work started in Section 8.3.1. to seek relationships in solar system data. In particular, what relationships exist, if any, between the following features of planets?
   - Average distance from the Sun and Orbit Period
   - Average distance from the Sun and Escape Velocity
   - Average distance from the Sun and Number of Moons
   - Average distance from the Sun and Presence of Rings
   - Average distance from the Sun and Mass
   - Mass and Escape Velocity
   - Orbit Period and Orbit Velocity
   - Orbit Period and Escape Velocity
2. Using *GeoGebra* and *Inkscape*, add the orbit of a comet to your image of the solar system. Data for the semi-major axis and orbital eccentricity of Halley's comet appears in the next item, and data for additional comets may be found at the following URL. https://nssdc.gsfc.nasa.gov/planetary/factsheet/cometfact.html
3. In *Blender*, model the orbit of a comet around the Sun. For example, the orbit of Halley's comet can be modeled in the manner described in Section 8.3.2, given that the orbital eccentricity of Halley's comet is 0.967 and the length of its semi-major axis is 17.94 AU. See the URL in Item 2 for a link to data for the semi-major axis and orbital eccentricity of 20 comets.

4. In *Blender*, add more planets and a moon or moons to your solar system simulation.
5. Implement the Orbit Velocity calculator by copying *spaceTravelEVcalculator.html* to *spaceTravelOVcalculator.html*. Then edit *spaceTravelOVcalculator.html* as necessary to calculate orbital velocity (v_o) as $v_o = \sqrt{\frac{GM}{r}}$. Compare the equations for v_e and v_o and note that $v_e = \sqrt{2} * v_o$.
6. Redesign the header of the Space Travel app. Consider replacing the background of the header with an image that you create. You might include one or more cosmic bodies (e.g., sun, planets, moons, comets) and perhaps an ellipse (or two) to depict an orbit (or two). Also, in your image, consider whether to include a rocket or a probe, for instance.
7. Redesign the Space Travel app to provide flexibility regarding the manner in which users enter the mass. In particular, rather than enter 20+ zeros, some users may prefer to use exponential notation. In that case, two input fields would be helpful, one for the coefficient and one for the exponent. Since base 10 is assumed, no input field is necessary for the base; your visual design should make clear that the exponent will apply to base 10.
8. Implement the Space Travel app as redesigned in Item 7.
9. Redesign the Tools submenu of the Space Travel app. In particular, combine the separate menu items for calculating orbit velocity and escape velocity into one menu item. Since both velocities are calculated using mass and radius, presenting both results would spare the user the burden or re-entering data. Further, consider flipping the visual design of the calculator from horizontal to vertical. In a vertical design, the input fields and their labels would be stacked rather than spread across the web page. Lastly, you may add items to the Tools submenu. For example, one tool might calculate where a planet, moon, or comet is in its orbit based on the date entered by the user. Consider searching for space calculators on the web in order to generate ideas.
10. Implement the Tools submenu of the Space Travel app as redesigned in Item 9.

8.5 Annotated Resources

Solar System Exploration
https://solarsystem.nasa.gov/

This website provides both overviews and detailed information about planets, moons, asteroids, comets, and meteors, as well as the Kuiper Belt and Oort Cloud. Presentations of cosmic bodies include extensive image galleries. In addition, this website features a simulation of objects orbiting the sun, including all eight planets, as well as some dwarf planets and comets.

Astronomy
https://openstax.org/details/books/astronomy

Andrew Fraknoi
David Morrison
Sidney C. Wolff
This free and open book, published in 2018, is available as a PDF file, an iBook, a Kindle book, and on the web. The book contains 30 chapters and provides a comprehensive introduction to astronomy.

Smithsonian National Air and Space Museum
https://airandspace.si.edu/

This website includes multiple virtual exhibits on aviation and space. Of particular relevance to the theme of this chapter is the following exhibition, which pertains to exploration.

Exploring the Planets
https://airandspace.si.edu/exhibitions/exploring-the-planets/online/
This exhibit describes discoveries of our solar system from ancient times to the present. In addition, this exhibit includes descriptions of tools for exploration. Such tools enable observations on Earth through telescopes for capturing light, as well as radio signals and radio waves. In addition, tools for exploration include airborne and orbital telescopes, fly-by probes, orbiters, landers, and rovers.

Universities Space Research Association (USRA)
https://www.usra.edu/

This website describes collaborations between a group of US universities and NASA to advance knowledge of astronomy and astrophysics, heliophysics, lunar and planetary science, earth science, computer science and information technology, aeronautics, microgravity science, space technology, and science facility management and operations, as well as contribute to student and workforce development.

One of the drop-down menu items at the USRA website pertains to educational opportunities. Within that drop-down menu is a section on K–12 STEM Education.

International Space University
https://www.isunet.edu/

This website describes the programs of study and research at the university, as well as admission procedures.

International Astronomical Union (IAU)
https://www.iau.org/

This website contains information about the association (located in Paris, France), as well as menu selections leading to educational resources. The IAU offers some workshops on astronomical themes, as well as resources pertaining to astronomy in education. For instance, see the web pages linked to themes within the drop-down menu titled *Astronomy for the Public*.

Canadian Space Agency
https://www.asc-csa.gc.ca/eng/default.asp

This website provides numerous resources for educators, researchers, and others. Sections of the website are dedicated to Junior Astronauts (adolescents 11–15 years old), to youth and education (including multiple digital games and activities), to open data, code and Application Programming Interfaces (https://www.asc-csa.gc.ca/eng/open-data/default.asp), and to careers, internships, and student jobs, for instance. After accessing various resources from the home page of the website, I found the Resources page below particularly beneficial.
https://www.asc-csa.gc.ca/eng/resources.asp

European Space Agency (ESA)
This website is also packed with information and resources about science and technology, including open campaigns for ideas on space exploration and safety, for instance. In addition, this website includes descriptions of the activities of each of the 22 Member States of the ESA. Much of the information is readily available through the sidebar menu. ESA's Gaia data archive is available at the following links.

https://gea.esac.esa.int/archive/
https://www.cosmos.esa.int/web/gaia/
Click the Help links for information on data retrieval.

National Aeronautics and Space Administration (NASA)
https://www.nasa.gov/

A comprehensive website on science, technologies, and careers in space. This website includes the solar system exploration website referenced in Section 8.1, as well as many additional resources, including open data. The website interface is easy to navigate given the drop-down menus for information about NASA Topics, Missions, Galleries, and Downloads, including apps and eBooks. Social media connections to NASA are available through the *Follow NASA* drop-down menu. Another useful drop-down menu at the website is *NASA Audiences*, which enables educators and students to go directly to information of relevance to them.

NASA's open data resources are available through the following web page.
https://www.nasa.gov/open/data.html

The open data page is organized into multiple topics, including *Space Science Data* (heliophysics, space physics, solar analysis, planetary data system, astrobiology, exoplanet), *Life Science Data*, and *NASA People Data*. In addition, there are separate sections on Earth Science, Space Science, and Life Science Information and Tools. There is also a separate section on Educational Resources. Further, there are links to aggregate data, code, and API archives, which are at https://data.nasa.gov/, https://code.nasa.gov/, and https://api.nasa.gov/.

8.6 Refining Your Teaching Practice

Congratulations for reaching this point! We have certainly covered some ground, and just as I am wrapping up this book, you are seeking to advance your teaching practice to better prepare students, to make their learning journeys more efficient and pleasant. Importantly, too, as you refine your teaching practice, you give your students a better shot at a fulfilling life.

As discussed in the first chapter, though teaching students is challenging, perhaps especially so for teachers of adolescents, as wonderfully diverse as they are, you are in a position to add immensely to their lives. Every school day, I am sure that you and other teachers help their students in numerous ways. At times, a child or, in your case, an adolescent, just needs a teacher to listen. Thank you for being that teacher on those occasions, for helping students cut through the noise to focus on what matters most.

As a secondary school teacher, you also help prepare students for life in contemporary times. This book asserts that you can better position your students for leadership positions in the world today by fostering their creativity and innovation with digital tools as you help them attain the learning goals in curricular standards. In pursuit of that goal, this book offers guidance in the form of the STARS model for developing stellar lessons and numerous opportunities to enhance your technology knowledge and skills, as well as your capacity to problem solve through design.

In Chapter 2 we gained awareness of instructional development theories and prior work on technology integration in schools. In the end, the STARS model emerged to guide the development of stellar lessons. **S**pecify the instructional context. Yes, know your students; be clear on the instructional objectives; and recognize the school and community resources available for instructional purposes. Next, the pivotal instructional design step. To foster creativity and innovation with digital tools, select or **T**arget one or more STELLAR activities. Recalling Figure 1.1, students can be challenged to problem solve through design; tinker in maker spaces; engage in computational thinking; develop with application software; and capitalize on digital data – leverage data, mine the data to look for patterns, and represent data in visuals. **A**ctivate your stellar lesson plan; **R**eflect on it; **S**ynthesize the impacts of the lesson on learning and learner state of mind; then refine the lesson and your teaching practice. As we spin on Earth at 1000 miles an hour and cruise around the Sun at 66,000 miles an hour, enjoy the ride – seek joy in teaching.

All the best to you and your students!

References

Andrews, R. G. (2021, June 22). Gas giants' energy crisis solved after 50 years. *Quanta*. https://www.quantamagazine.org/cassini-data-solves-jupiter-and-saturns-energy-mystery-20210622/

British Columbia Ministry of Education. (2016). *K-9 learning standards for science*. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/science/en_science_k-9.pdf

British Columbia Ministry of Education. (2018a). *Grade 10 learning standards for science*. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/science/en_science_10_core.pdf

British Columbia Ministry of Education. (2018b). *Grade 11 learning standards for Earth science*. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/science/en_science_11_earth-sciences.pdf

British Columbia Ministry of Education. (2018c). *Grade 11 learning standards for physics*. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/science/en_science_11_physics.pdf

British Columbia Ministry of Education. (2018d). *Grade 12 learning standards for geology*. https://curriculum.gov.bc.ca/sites/curriculum.gov.bc.ca/files/curriculum/science/en_science_12_geology.pdf

Cooper, A., Turney, C. S. M., Palmer, J., Hogg, A., McGlone, M., Wilmshurst, J., et al. (2021). A global environmental crisis 42,000 years ago. *Science, 371*(6351), 811–818. https://doi.org/10.1126/science.abb8677

Department for Education. (2013a). *Science programmes of study: Key stage 3*. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/335174/SECONDARY_national_curriculum_-_Science_220714.pdf

Department for Education. (2013b). *Geography programmes of study: Key stage 3*. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/239087/SECONDARY_national_curriculum_-_Geography.pdf

Department for Education. (2014). *Science programmes of study: Key stage 4*. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/381380/Science_KS4_PoS_7_November_2014.pdf

DiBraccio, G. A., & Gershman, D. J. (2019). Voyager 2 constraints on plasmoid-based transport at Uranus. *Geophysical Research Letters, 46*(19). https://doi.org/10.1029/2019GL083909

European Space Agency. (n.d.). *3… 2… 1… Liftoff! Building your own paper rocket.*
See Question 11 on Page 20.
https://esamultimedia.esa.int/docs/kids/P17_3.2.1.lift_off.pdf
Secondary Classroom Resources
https://www.esa.int/Education/Teachers_Corner/Secondary_classroom_resources

Farage, T. (n.d.). *Let's do the math: Escape velocity*. https://personal.utdallas.edu/~TFARAGE/escapevelocity/transcripts/math%20video%20transcript.pdf

Gershman, D. J., & DiBraccio, G. A. (2020). Solar cycle dependence of solar wind coupling with giant planet magnetospheres. *Geophysical Research Letters, 47*(24). https://doi.org/10.1029/2020GL089315

Gohd, C. (2020, December 22). Scientists think this may be the farthest galaxy in the universe. *Scientific American*. https://www.scientificamerican.com/article/scientists-think-this-may-be-the-farthest-galaxy-in-the-universe/

Machamer, P., & Miller, D. M. (2021). Galileo Galilei. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy*. https://plato.stanford.edu/archives/sum2021/entries/galileo/

McDowell, J. C. (2018). The edge of space: Revisiting the Karman line. *Acta Astronautica, 151*, 668–677. https://arxiv.org/abs/1807.07894

NASA. (2013a). *The heliosphere*. https://www.nasa.gov/mission_pages/sunearth/science/Heliosphere.html

NASA. (2013b). *Solar rotation varies by latitude*. https://www.nasa.gov/mission_pages/sunearth/science/solar-rotation.html

NASA. (2018). *Solar system temperatures*. https://solarsystem.nasa.gov/resources/681/solar-system-temperatures/

NASA. (2020a). *The nearest neighbor star*. https://imagine.gsfc.nasa.gov/features/cosmic/nearest_star_info.html

NASA. (2020b). *The farthest visible reaches of space*. https://imagine.gsfc.nasa.gov/features/cosmic/farthest_info.html

NASA. (2021a). *Heliosphere*. https://science.nasa.gov/heliophysics/focus-areas/heliosphere

NASA. (2021b). *Gateway*. https://www.nasa.gov/gateway/overview

For information about the initial instruments on the sustained orbiter, see https://www.nasa.gov/feature/nasa-selects-first-science-instruments-to-send-to-gateway

National Research Council. (2012). *A framework for K-12 science education*. The National Academies Press. https://doi.org/10.17226/13165

Ocker, S. K., Cordes, J. M., Chatterjee, S., Gurnett, D., Kurth, B., & Spangler, S. (2021). Persistent plasma waves in interstellar space detected by Voyager 1. *Nature Astronomy, 5*, 761–765. https://doi.org/10.1038/s41550-021-01363-7

https://voyager.jpl.nasa.gov/news/details.php?article_id=122

Overbye, D. (2020, July 10). Beyond the Milky Way, a galactic wall. *The New York Times*. https://www.nytimes.com/2020/07/10/science/astronomy-galaxies-attractor-universe.html

Quinn, T., & Speake, C. (2014). The Newtonian constant of gravitation – A constant too difficult to measure? An introduction. *Philosophical Transactions of the Royal Society, 372*, 1–3. https://doi.org/10.1098/rsta.2014.0253

Shah, K. (2021, February 18). Earth's magnetic field flipping linked to extinctions 42,000 years ago. *NewScientist*. https://www.newscientist.com/article/2268520-earths-magnetic-field-flipping-linked-to-extinctions-42000-years-ago/

Smith, G. (2021). Newton's Philosophiae Naturalis Principia Mathematica. In E. N. Zalta (Ed.). *The Stanford encyclopedia of philosophy*. https://plato.stanford.edu/archives/sum2021/entries/newton-principia/

Stachel, J. (Ed.). (1998). *Einstein's miraculous year: Five papers that changed the face of physics*. Princeton University Press.

Whitman Cobb, W. (2021, September 11). Space tourism: SpaceX Inspiration4 mission will send 4 civilians with minimal training into orbit. *SciTechDaily*.
https://scitechdaily.com/space-tourism-spacex-inspiration4-mission-will-send-4-civilians-with-minimal-training-into-orbit/

Index