

AP COMPUTER SCIENCE PRINCIPLES CURRICULUM



AP Computer Science
Principles



Grade Level(s):10-12

Curriculum Author(s): Raymond Robillard

Course Description: The AP Computer Science Principles course opens up the world of Computer Science to all students by providing a CS course with a greater focus on fundamental concepts in CS without getting bogged down in the exact details of syntax for a specific programming language. This course covers both universal concepts in programming but also focuses on concepts such as: digital copyright issues, closing the technology gap created by economic inequity, how the internet functions, ethical issues in CS, data abstraction, and algorithms. The course has a heavy emphasis on collaboration. The AP Computer Science Principles course curriculum is compatible with many CS1 courses in colleges and universities. Students will be able to obtain college credits by successfully meeting a benchmark score on the Advanced Placement Exam. Block coding is utilized in the course through the online platform at code.org.

Year At A Glance

Unit Title	Overarching Essential Question	Overarching Enduring Understanding	<u>Vision of A Learner “I Can” Statements</u>
<u>Digital Information</u>	How can we use 1s and 0s to represent something complex like a video of the marching band playing a song?	The way a computer represents data internally is different from the way the data are interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.	TI3(9-12); AA1(9-12)
<u>The Internet</u>	What is the internet and how does it work?	The internet is a connected network of computing devices that uses an agreed upon series of protocols for the transfer of information.	TCC2(9-12); TI3(9-12)
<u>Intro to App Design</u>	How does the interactive design of programs and apps impact its user base?	Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.	CCE2(9-12); CCE3(9-12)
<u>Variables, Conditionals, and Functions</u>	In what ways can we reduce code complexity through the use of conditional statements and functions?	Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.	TI1(9-12); AA1(9-12)
<u>Data</u>	When is it more appropriate to use a computer to analyze data than to complete the analysis by hand?	We live in an era where data is so plentiful as to seem overwhelming and it has become increasingly important to know how to process and interpret it correctly.	TCC2(9-12); TCC4(9-12)



Lists, Loops, and Traversals	What types of abstractions are available to programmers and how are they used?	To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.	TI1(9-12); AA1(9-12)
Parameters, Return, and Libraries	How do programmers utilize the work on one another to improve software?	Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.	TI1(9-12); AA1(9-12)
Cybersecurity and Global Impacts	Are innovators responsible for the harmful effects of their computing innovations, even if those effects were unintentional? Why or why not?	While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.	TCC1(9-12); CCE1(9-12); AA4(9-12)
Algorithms	What common real-world problems can be solved with computer processed algorithms?	There exist real-world problems that computers cannot solve, and even when a computer can solve a problem, it may not be able to do so in a reasonable amount of time.	TCC3(9-12); AA1(9-12); AA2(9-12)

Unit 1 - Digital Information

Desired Results - Goals, Transfer, Meaning, Acquisition

Established Goals:

- 3B-IC-27 - Predict how computational innovations that have revolutionized aspects of our culture might evolve.
- 3A-AP-21 - Evaluate and refine computational artifacts to make them more usable and accessible.
- 2-DA-07 - Represent data using multiple encoding schemes.
- 3A-DA-09 - Translate between different bit representations of real-world phenomena, such as characters, numbers, and images.
- 3A-CS-02 - Compare levels of abstraction and interactions between application software, system software and hardware layers.
- 3A-DA-10 - Evaluate the tradeoffs in how data elements are organized and where data is stored.
- 3A-IC-28 - Explain the beneficial and harmful effects that intellectual property laws can have on innovation.

Vision of A Learner Attributes: Students will be able to independently use their learning to... ("I can" statements to be demonstrated)

TI3(9-12): I can formulate and investigate probing questions to further my learning.

AA1(9-12): I can evaluate different approaches and justify the best pathway to success.

Understandings: Students will understand that...

- The way a computer represents data internally is different from the way the data are interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.
- Programs can be used to process data, which allows users to discover information and create new knowledge.

Essential Questions:

- How can we use 1s and 0s to represent something complex like a video of the marching band playing a song?
- What types of problems can be solved more easily with a computer, and what types can be solved more easily without a computer? Why?

Students will know...

- Computing innovations change the way people interact and share information.
- All data can be reduced to binary representations.
- Binary representations of decimal numbers follow a pattern based on place value.
- Decimal numbers can be represented in binary with some limitations.
- 8 bits = 1 byte
- Text can be abstracted first as a number assigned to each character, and then to a binary representation.

Students will be able to...

- Communicate with classmates about computing innovations in their lives.
- Describe positive and negative effects of computing innovations.
- Explain how the same piece of information can be represented in a variety of different ways.
- Use a device to represent different pieces of information
- Use patterns to represent information
- Explain the challenges of creating a clear set of rules for ordering patterns
- Follow a set of rules for ordering sets of patterns

- | | |
|---|---|
| <ul style="list-style-type: none"> • Pixels in a monochrome image are represented by a bit, but colors require many more bits of memory per pixel. • The greater the sampling size, the more detailed an image will be. • Lossless compression means that data can be compressed and reassembled with perfect accuracy. • Lossy compression means that data loses some detail as part of the compression process. • Intellectual property laws, including copyrights, are governed by rules and expectations that face unique issues in a digital era. • Creative Commons Copyrights allow creators to choose the ways in which their work can be shared. | <ul style="list-style-type: none"> • Explain how the position of each binary digit determines its place value and numeric value • Represent binary numbers using combinations of decimal (base 10) digits 0-9 • Represent decimal numbers using combinations of binary (base 2) digits 0 and 1 • Understand that overflow and roundoff errors result from real-world limitations in representing place value. • Describe the challenges in representing text when using a fixed number of bits for each character • Develop a system for using numbers to represent text • Explain how bits are grouped to represent abstractions like numbers and text. • Explain how bits can be used to represent the individual pixels of a black and white image • Explain how sampling is used to create a digital form of an analog image • Explain how bits can be used to represent the individual pixels of a color image • Explain how digital data is used to approximate real-world analog data • Analyze patterns in data to determine compression strategies • Create lossless compressions of text files • Examine the effects of lossy compression on text & images • Given a piece of media, decide whether to use lossy or lossless compression based on the needs of a situation • Argue if current copyright laws are helping or harming society using evidence from an article • Explain how copyright and Creative Commons Licenses can be applied to digital works of creativity |
|---|---|

Key Vocabulary: Binary, Bit, Byte, Decimal, Octal, Hexadecimal, Overflow, Round-off Error, Text, Ascii, Abstraction, Pixel, RGB, Digital/Analog, Sampling, Lossy Compression, Lossless Compression, Intellectual Property, Copyright, Creative Commons

Assessment Evidence

Performance Tasks:

Digital Information Dilemmas Project: **TI3(9-12)** and **AA1(9-12)**

- Students tackle the question of whether digitizing information has made the world a better or worse place.
- Students place stickies on a spectrum of "worse" to "better" to state their opinion prior to doing the activity.
- Students then choose an article they are interested in reading using a process the class completed collectively in the previous lesson.
- Students design and make an artifact to present their findings.
- Students identify the benefits and harms highlighted in the article and make an overall claim about whether the world has been made better or worse through the digitization of information.
- Students update the position of their stickies on the "better" to "worse" spectrum and discuss whether and why they changed their opinion.

Other Evidence:

Unit 1 Quiz - This quiz consists of a series of released problems from AP CS Principles Exams, and requires students to assess what snippets of code do, debug code, or to evaluate different ways of accomplishing the same task.

Code.org End of Lesson Checks for Understanding - Part of the code.org curriculum materials include 1-3 questions that summarize learning from the lesson in an open ended format. These questions can easily be shared out by the course instructor to create further discussion.

Code.Org Unit 1 Assessment - This end-of-the-unit task assesses student understanding through a variety of multiple choice questions that cover topics from this unit.

Learning Plan

This unit explores the technical challenges and questions that arise from the need to represent digital information in computers. Learn how complex information like numbers, text, images, and sound are represented in bits, how compression works, and the broader social impacts of digitizing the world's information.

- Welcome to CSP- Welcome to Computer Science Principles! The first lesson is about getting students excited about the course and connecting their own personal interests to computer science. Students are asked to share something they know a lot about and teach it to a small group. Groups make a "rapid" prototype of an innovative idea and share it. Students watch a brief video about computing innovations. The lesson ends with students logging into the Code.org CSP course web site, and answering a brief prompt about what "computer science" means to them. This activity plants the initial seed for students to think about the ways in which they might be able to solve some problems relevant to their lives with technological innovations. **TI3(9-12)**
- Representing Information- Using everyday materials, students create devices for sending information to a partner, which introduces the concept of sending bits of information from one place to another. Each group then uses its device to send an answer to a question. Following this, students modify their devices to answer more complex answers, responding with one of four possible messages, then one of eight possible messages, then one of sixteen possible messages. This lays the foundation for understanding how complex information is represented in computers using a combination of bits. **TI3(9-12)**
- Circle Square Patterns- Students will create rules for ordering patterns of circles and squares. Students generate all possible messages

with three place values, then create rules that explain how they ordered each message. Emphasis is placed on creating clear rules so that, if another group were to follow the rules, they would generate the same list in the same order. Using these rules, students then try to list all possible messages with four place values. As the lesson concludes, students share their rules with classmates. This lesson acts as a bridge to the next lesson where binary is formally introduced and practiced.**TI3(9-12)**

- Binary Numbers- In this lesson, students will practice representing numbers in binary (base 2), transitioning from the circle-square representations they made in the last lesson. Students will create and use a "Flippy Do", a manipulative which helps students convert between binary (base 2) and decimal (base 10) numbers. They will practice converting numbers and explore the concept of place value in the context of binary numbers.**AA1(9-12)**
- Overflow and Rounding- Students extend their understanding of the binary number system by exploring errors that result from overflow and rounding. They use the binary odometer widget and develop their own systems for representing fractional amounts using the Flippy Do. Students discover the limitations of creating numbers that are "too big" or "too small" to count. They learn that, while a number system is infinite, the physical representation of numbers requires place values -- which are finite, and limit the ability to represent numbers.**AA1(9-12)**
- Representing Text- In this lesson, students create a system for representing text using only numbers while communicating with each other. They are only allowed to send numbers back-and-forth, so they must create a system to translate between number and character. Most importantly they should understand the challenges or principles that led to the creation of these systems and that these systems are somewhat arbitrarily created to solve a shared problem. They work not because there was "one right answer" but because many people agreed to use them. At the end of the main activity they briefly review the ASCII system for representing text. The wrap up discussion emphasizes how all of the concepts thus far have 'built' on each other and introduces the concept of abstraction to describe this progression.**AA1(9-12)**
- Black and White Images- Students explore how black and white images are represented. Students use the black and white pixelation widget to represent each pixel of an image with black or white light. They learn how to sample an analog image using small squares of uniform size (each represented with a black or white value) and reflect on the pros and cons of choosing a smaller or larger square size when sampling.**AA1(9-12)**
- Color Images- This is a second opportunity for students to interact with the Pixelation Widget, but this time they will work with color pixels. Students start off learning that each pixel uses red, green, and blue lights that can be turned on or off using bits. They will create more color variants using an increasing amount of bits per pixel, and apply their learning by approximating an analog color image using the widget. These images are produced using layers of abstraction, with each layer relying on the other to perform its process.**AA1(9-12)**
- Lossless Compression- Students use the Text Compression Widget to experiment with compressing songs and poems and try to find their 'personal best' compression. A video introduces important vocabulary for the lesson and demonstrates the full features of the widget. Students pick a text they think will be 'easy' to compress and one they think will be 'difficult', paying attention to why some texts might be more compressible than others. As a wrap-up, students discuss what factors make some texts more compressible than others. The widget is just one example of lossless compression and students aren't expected to master specific compression strategies - instead, they should understand that lossless compression uses less data and still lets them re-create the original information.**TI3(9-12)**
- Lossy Compression- Students are introduced to lossy compression via the Lossy Text Compression widget. They apply this concept and their prior knowledge of sampling to create their own lossy compressions of image files using the Lossy Image Widget. A theme throughout the lesson is that lossy compression can greatly reduce the file size, but it can also greatly reduce the quality and it's important to find that balance between quality and file size. The real challenge here is finding where that line is - how much can we compress but still

keep it recognizable? Students then discuss several practical scenarios where they need to decide whether to use a lossy or lossless compression algorithm. The lesson ends with a discussion of the situations where lossless compression is important and the situations where lossy compression is important. **TI3(9-12)**

- Intellectual Property- Students are asked to reflect on who owns their creative works from this class, such as their pixel images, before reading an article describing how ownership can become complicated as analog works become digital artifacts. After reading the article, students watch several videos explaining copyright and introducing them to the Creative Commons. Students then re-read the article answering three questions about the benefits, harms, and impacts of current copyright policy. Students use their new understanding of copyright to form an opinion about current copyright policies and create a small poster justifying their opinion with a quote from the article. This lesson is also a scaffold to the larger project that begins after this lesson which includes several tasks that are also a part of this lesson such as, annotating an article, answering questions, and forming an opinion using the article as evidence. **TI3(9-12)**

Teacher Resources:

Code.org online curriculum

<http://chortle.ccsu.edu/Java5/index.html#07>

Java Runtime Environment

Java Development Kit

Unit 2 - The Internet

Desired Results - Goals, Transfer, Meaning, Acquisition

Established Goals:

2-NI-04 - Model the role of protocols in transmitting data across networks and the Internet.

3A-NI-04 - Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.

3B-NI-03 - Describe the issues that impact network functionality (e.g., bandwidth, load, delay, topology).

Vision of A Learner Attributes: Students will be able to independently use their learning to... ("I can" statements to be demonstrated)
TCC2(9-12): I can evaluate evidence from multiple perspectives, and recognize their limitations and implications, in order to justify new conclusions.

TI3(9-12): I can formulate and investigate probing questions to further my learning.

Understandings: Students will understand that...

- The internet is a connected network of computing devices that uses an agreed upon series of protocols for the transfer of information.
- Computer systems and networks facilitate the transfer of data.
- Parallel and distributed computing leverage multiple computers to more quickly solve complex problems or process large data sets.

Essential Questions:

- What is the internet and how does it work?
- When an internet service outage occurs in a different part of your town or city, how are you still able to access the internet?

Students will know...

- The internet is a network of computing devices.
- Like postal addresses, IP addresses allow computing devices to identify one another.
- DNS matches names like www.google.com with the ip address at which the data for the site is located.
- Protocols are the key to making sure all computing devices can interact with others on the internet.
- Packets contain data sent over the internet.
- An email or download could require very many packets of data.
- Redundancy allows systems to be fault tolerant.
- Different protocols for sending data exist and are more or less appropriate in different situations.

Students will be able to...

- Identify questions they have about how the Internet works
- Use the Internet Simulator to communicate information with a partner
- Explain how computing devices can be connected to form a network
- Identify the path(s) connecting two devices in a simulated network
- Describe the way the Internet Protocol helps uniquely identify one another on the Internet
- Explain the need for open and shared protocols for communicating on the Internet
- Describe how the redundant nature of networks can lead to

	<p>dynamic, fault tolerant routes</p> <ul style="list-style-type: none"> ● Explain how data is routed through the Internet ● Describe how information flows through the Internet as a datastream of packets ● Explain how packet numbering and re-ordering can allow for large messages to reliably be sent even if packets are dropped or arrive out of order ● Explain the differences between the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) ● Describe how HTTP is used for sharing the files and pages that make up the World Wide Web ● Describe how the Domain Name System helps the Internet scale by allowing devices to find the IP addresses associated with a domain name ● Explain how different layers of protocols on the Internet build upon and rely on one another
<p>Key Vocabulary: Computing Device, Computing System, Computing Network, Path, Bandwidth, Internet, Protocol, IP Address, Internet Protocol, Router, Redundancy, Fault Tolerant, User Datagram Protocol(UDP), Transmission Control Protocol (TCP), Datastream, Packet, Packet Metadata, Scalability, Domain Name System(DNS), World Wide Web, Hypertext Transfer Protocol(HTTP)</p>	
<p style="text-align: center;">Assessment Evidence</p>	
<p>Performance Tasks: Internet Dilemmas Project:TCC2(9-12) and TI3(9-12)</p> <p>Students work on a two-day project exploring a dilemma at the intersection of the Internet and society. In this project, students explore a relevant Internet dilemma: Net Neutrality, Internet Censorship, or the Digital Divide. Students apply their knowledge of how the Internet works to address the core question related to their chosen dilemma. This project addresses the "so what" question - why is it important to learn about how the Internet works?</p>	<p>Other Evidence:</p> <p>Unit 2 Quiz - This quiz consists of a series of released problems from AP CS Principles Exams, and requires students to assess what snippets of code do, debug code, or to evaluate different ways of accomplishing the same task.</p> <p>Code.org End of Lesson Checks for Understanding - Part of the code.org curriculum materials include 1-3 questions that summarize learning from the lesson in an open ended format. These questions can easily be shared out by the course instructor to create further discussion.</p> <p>Code.Org Unit 2 Assessment - This end-of-the-unit task assesses</p>

student understanding through a variety of multiple choice questions that cover topics from this unit.

Learning Plan

This unit reveals how the Internet was designed to connect billions of devices and people to one another. Learn how the different protocols of the Internet work and actually build them yourself using the Internet Simulator. Then consider the impacts the Internet has had, both good and bad, on modern life.

- Welcome to the Internet- After a short transition from representing information in Unit 1 to communicating information in Unit 2, students take time to think about their knowledge of the Internet and how it works. Following this, students are introduced to a new widget: The Internet Simulator which they will use throughout this unit to explore the inner workings of the Internet. Students should leave this lesson primed to know more about the Internet. **TCC2(9-12)**
- Building a Network- In this lesson, students are formed into groups of 5-7 and given string so they can connect themselves together to form a computer network. Students are given several specific networks to form, along with several guidelines for how to best form computer networks. Students are also forced to wrestle with conflicting guidelines in determining the 'best' way to connect together to form a network, and will need to justify why they chose the networks that they did. **TCC2(9-12)**
- The Need for Addressing- Students complete a scheduling challenge three times, once unplugged, and twice on the Internet Simulator, to explore the need for addressing messages online. Students first complete a challenge where they are allowed to talk to one another to fill out a weekly schedule. They then move on to a version of the Internet Simulator where all of their messages are "broadcast" or sent to everyone in the same simulator room. This challenge forces students to develop shared rules for communicating to complete the scheduling activity a second and then third time. The wrap up helps students connect their experiences to real-life rules, or protocols, used on the Internet for addressing messages. **TI3(9-12)**
- Routers and Redundancy- Students spend most of today's lesson in an updated Internet Simulator that lets students send messages with a dedicated To and From IP Address. Students start by connecting to a dedicated router and sending messages only to each other. They look at the router logs to find other students on different routers, then send messages to those students. They look at the router logs again to notice that messages are being passed between routers in order to reach their destination. Students continue to send messages and view the logs one last time to notice that the messages are also taking different paths to reach the same destination. The lesson wraps-up by introducing new vocabulary and using these words to summarize today's activity. **TI3(9-12)**
- Packets- Students learn that large messages sent over the Internet are actually divided into individual packets and explore the challenges this creates. First students explore a version of the Internet Simulator that breaks messages into packets to get a sense for how it works. Students then use an activity guide that walks them through the challenges that dropped or out-of-order packets create. They then design their own protocol that addresses these challenges. At the end of the lesson students watch a video and learn the User Datagram Protocol (UDP) and The Transmission Control Protocol (TCP), two different protocols for sending messages broken into packets. **TI3(9-12)**
- HTTP and DNS- Students conclude their study of the layers of the Internet by learning about two top-level protocols, HTTP and DNS. The lesson begins with a review of the layers students have already learned, namely the physical Internet, IP, and TCP / UDP. The lesson then dives into the core of the unit which is "What happens when I type an address into my browser?" Students will complete two brief activities,

one unplugged, and one plugged, that explore how the DNS works. Students then watch videos explaining how both DNS and HTTP work, taking notes in a provided activity guide. The lesson concludes with a review of how the different layers of the Internet work. **TCC2(9-12)**

Teacher Resources:

Code.org online curriculum; <http://chortle.ccsu.edu/Java5/index.html#07>; Java Runtime Environment; Java Development Kit

Unit 3 - Intro to App Design

Desired Results - Goals, Transfer, Meaning, Acquisition

Established Goals:

- 3A-AP-16 - Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.
- 2-AP-17 - Systematically test and refine programs using a range of test cases.
- 3A-AP-13 - Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
- 3A-AP-21 - Evaluate and refine computational artifacts to make them more usable and accessible.
- 3A-AP-22 - Design and develop computational artifacts working in team roles using collaborative tools.
- 3A-AP-23 - Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.
- 3A-CS-03 - Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.

Vision of A Learner Attributes: Students will be able to independently use their learning to... ("I can" statements to be demonstrated)
 CCE2(9-12): I can give and receive actionable and relevant feedback with openness to be able to determine meaningful revisions for success.
 CCE3(9-12): I can show initiative in prompting group discourse and fostering collaboration among others, providing actionable feedback, and working with others to solve problems and/or design products.

Understandings: Students will understand that...

- Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.
- Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.

Essential Questions:

- How does the interactive design of programs and apps impact its user base?
- What are some ways you can collect additional feedback on your program to use for improvements?

Students will know...

- Collaboration is an important element of designing computer software.
- User feedback will result in computer software that best meets the needs of its intended audience.
- A user interface is the most common field for input and output on today's modern day applications.
- Using variable names and element ids makes coding, and reading code, easier.

Students will be able to...

- Identify the inputs of an app
- Identify the outputs of an app
- Identify the purpose of an app
- Set up the User Interface of an app including buttons, text, and images
- Use meaningful names to for element ids
- Explain the qualities that differentiate natural languages and programming languages

<ul style="list-style-type: none"> • Many programming languages exist, though many of the core elements are the same. • Comments clarify the purpose of code segments • Debugging is critical to making sure that software works as intended when released. • Sequential code executes in a top-down order. • Event-driven code executes as conditions are met. 	<ul style="list-style-type: none"> • Justify the existence of programming languages to precisely communicate instructions • Define a program as a sequence of commands that are executed or run by a computer • Define comments as notes or documentation into a program that do not affect how the program executes • Explain the differences between how sequential and event-driven programs execute • Debug simple sequential and event-driven programs • Use the debugging process and Identify specific best practices for debugging programs • Use the speed slider, break points, and documentation as part of the debugging process
<p>Key Vocabulary: App, User Interface, Input, Output, Design, Implementation, Program Statement, Program, Sequential Programming, Event Driven Programming, Debugging, Documentation, Comment, Pair Programming</p>	
<p style="text-align: center;">Assessment Evidence</p>	
<p>Performance Tasks: Designing an App Project:CCE2(9-12) and CCE3(9-12)</p> <ul style="list-style-type: none"> • Students brainstorm app ideas and sketch out user interfaces • Students follow the app development process outlined in their App Development Guide • Students apply Pair Coding • Students watch other groups test their apps and collect feedback that will be used to make updates. • Students review other apps that classmates made and complete a short set of reflection prompts before submitting their projects. 	<p>Other Evidence:</p> <p>Unit 3 Quiz - This quiz consists of a series of released problems from AP CS Principles Exams, and requires students to assess what snippets of code do, debug code, or to evaluate different ways of accomplishing the same task.</p> <p>Code.org End of Lesson Checks for Understanding - Part of the code.org curriculum materials include 1-3 questions that summarize learning from the lesson in an open ended format. These questions can easily be shared out by the course instructor to create further discussion.</p> <p>Code.Org Unit 3 Assessment - This end-of-the-unit task assesses student understanding through a variety of multiple choice questions that cover topics from this unit.</p>

Learning Plan

This unit is an introduction to programming and app design with a heavy focus on important skills like debugging, pair programming, and user testing. Learn how to design user interfaces and write event-driven programs in App Lab and then design a project that teaches your classmates about a topic of your choosing.

- Introduction to Apps- Students explore and investigate what makes an app an app. They begin by looking at and discussing five different apps. Following this, students watch a video explaining the basics of how computers work. Finally students return to the apps and consider the various inputs and outputs. **CCE2(9-12) and CCE3(9-12)**
- Introduction to Design Mode- Students work through a progression of levels to build an understanding of how to use Design Mode to layout an app. The final level has students setting up the screen of an app by attempting to copy an image of an app. **CCE2(9-12) and CCE3(9-12)**
- The Need for Programming Languages- In this lesson students explore the challenges of clearly communicating instructions. They build a small arrangement of blocks (LEGO® pieces or paper cutouts) and then create text instructions a classmate could follow to construct the same arrangement. Groups then trade instructions to see if they were clear enough to allow reconstruction of the original arrangement. The wrap-up discussion is used to highlight the inherent ambiguities of human language and call out the need for the creation of a programming language which leaves no room for interpretation. **CCE2(9-12) and CCE3(9-12)**
- Intro to Programming- Students use and modify a series of simple apps to get familiar with a small set of programming commands. They observe the way the code runs by slowing down the code and compare programs that run all at once to those that respond to user actions like button clicks. At the end of the lesson students discuss what they observed and are introduced to some key vocabulary for describing the running of programs. **CCE2(9-12) and CCE3(9-12)**
- Debugging- In this lesson students practice using the different programming concepts that they were introduced to in the last lesson. To begin, however, they are introduced to the concept of debugging and are encouraged to use and reflect on this practice throughout the lesson. At the end of the lesson students share their experiences debugging as well as new realizations about programming. **CCE2(9-12) and CCE3(9-12)**

Teacher Resources:

Code.org online curriculum; <http://chortle.ccsu.edu/Java5/index.html#07>; Java Runtime Environment; Java Development Kit

Unit 4 - Variables, Conditionals, and Functions

Desired Results - Goals, Transfer, Meaning, Acquisition

Established Goals:

- 2-AP-11 - Create clearly named variables that represent different data types and perform operations on their values.
- 2-AP-19 - Document programs in order to make them easier to follow, test, and debug.
- 3B-AP-23 - Evaluate key qualities of a program through a process such as a code review.
- 2-AP-10 - Use flowcharts and/or pseudocode to address complex problems as algorithms.
- 3A-AP-15 - Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance and explain the benefits and drawbacks of choices made.
- 2-AP-12 - Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- 3B-AP-21 - Develop and use a series of test cases to verify that a program performs according to its design specifications.
- 3A-AP-17 - Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
- 3B-AP-14 - Construct solutions to problems using student-created components, such as procedures, modules and/or objects.

Vision of A Learner Attributes: Students will be able to independently use their learning to... ("I can" statements to be demonstrated)

TI1(9-12): I can implement a realistic plan and adapt when necessary to achieve my goals.

AA1(9-12): I can evaluate different approaches and justify the best pathway to success.

Understandings: Students will understand that...

- The way statements are sequenced and combined in a program determines the computed result.
- Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.

Essential Questions:

- How can we store data in a program to solve problems?
- In what ways can we reduce code complexity through the use of conditional statements and functions?

Students will know...

- Numbers and strings are two different types of values
- Expressions evaluate to a single new value
- Global Variables are permanent and can be referenced anywhere in program code.
- Local variables are temporary and only exist within the scope of their declaration.
- A boolean value is a data type that is either true or false.
- Each side of a Boolean expression is reduced to a single value.

Students will be able to...

- Evaluate expressions that include numbers, strings, arithmetic operators, Boolean values, comparison operators, and logical operators
- Trace simple programs that use variables, expressions, and variable assignment.
- Use appropriate vocabulary to describe variables, expressions, variable assignment, Boolean expressions, conditional statements, functions

- A truth table can allow us to determine all truth values for a Boolean expression.
- A flowchart can illustrate the steps of making a decision with a boolean expression.
- A conditional statement is used to choose between two or more options.
- Conditional statements can be nested to handle multiple possible results.
- Functions can be used to avoid unneeded code reuse.
- Functions are an abstraction for multiple commands in code.
- Event based functions occur only when specific conditions are met.

- Explain the purpose of those programming patterns with variables both in terms of how they work and what they accomplish
- Identify common programming patterns when using variables as part of an app
- Modify apps that make use of common programming patterns with variables to adjust their functionality
- Debug programs that use variables and expressions
- Write programs that use variables and expressions with the support of sample code.
- Implement programming patterns with variables to develop a functioning app
- Recognize the need for programming patterns with variables as part of developing a functioning app
- Use debugging skills as part of developing an app
- Write comments to clearly explain both the purpose and function of different segments of code within an app
- Explain the purpose of those programming patterns with boolean expressions and conditional statements both in terms of how they work and what they accomplish
- Identify common programming patterns using boolean expressions and conditional statements
- Modify apps that make use of common programming patterns with boolean expressions and conditional statements to adjust their functionality
- Debug programs that use boolean expressions, conditional statements, and functions
- Write programs that use boolean expressions, conditional statements, and functions with the support of sample code.
- Implement programming patterns with boolean expressions and conditionals statements to develop a functioning app
- Recognize the need for programming patterns with Boolean expressions and conditional statements as part of developing a functioning app
- Write comments to clearly explain both the purpose and function of different segments of code within an app
- Describe the way a function call interrupts the normal flow of execution within a program

	<ul style="list-style-type: none"> • Modify programs that declare and call functions to adjust their functionality • Trace the flow of execution in programs that declare and call functions • Identify opportunities to use functions to reduce repeated code within a program • Implement a function using programming patterns while developing a functional app • Recognize the need for a function to reduce repeated code while developing a functional app
Key Vocabulary: Expressions, Variable, Assignment Operator, String, Concatenation, Global Variable, Local Variable, Boolean Expression, Comparison Operator, Logical Operators (&&, , !), Truth Table, If Statement, If Else-If Statement, Modulus Operator (%), Syntax Error, Logic Error, Function, Function Call, Event Based Function	
Assessment Evidence	
<p>Performance Tasks: Decision Maker App Project: TI1(9-12) and AA1(9-12)</p> <p>Using a Project Planning Guide, students work through the stages of creating an app from scratch.</p> <ul style="list-style-type: none"> • Students complete the planning phase. • Students translate the plans they documented in Part 1 of the Practice PT to a working program in App Lab through a series of steps. • Students work with classmates to review and update the functionality of their apps before submitting the final project. 	<p>Other Evidence:</p> <p>Unit 4 Quiz - This quiz consists of a series of released problems from AP CS Principles Exams, and requires students to assess what snippets of code do, debug code, or to evaluate different ways of accomplishing the same task.</p> <p>Code.org End of Lesson Checks for Understanding - Part of the code.org curriculum materials include 1-3 questions that summarize learning from the lesson in an open ended format. These questions can easily be shared out by the course instructor to create further discussion.</p> <p>Code.Org Unit 4 Assessment - This end-of-the-unit task assesses student understanding through a variety of multiple choice questions that cover topics from this unit.</p>
Learning Plan	
<p>This unit explores how variables, conditionals, and functions allow for the design of increasingly complex apps. Learn how to program with these three new concepts through a sequence of collaborative activities. Then build your own decision maker app to share with friends and help them make a decision.</p>	

- Variables Explore- To begin the lesson students explore sample apps similar to the ones they'll be able to build by the end of the unit. Then students complete an unplugged activity with plastic baggies and sticky notes to build a mental model of how variables are used to move and store information. The lesson ends with a synthesizing discussion and students adding key vocabulary to their journal.**AA1(9-12)**
- Variables Investigate- In this lesson students work with partners to investigate several versions of the "Thermostat App" to understand how variables store and update information. To begin, students examine a version of the app where the temperature displayed changes each time a button is clicked. The next two versions of the app demonstrate how variables can store strings. Students learn about the patterns they are observing, specifically "Counter Pattern with Event" and "Variables with String Concatenation Pattern". To conclude the lesson, students review and discuss the programming patterns that they will make use of in the programs they write. **AA1(9-12)**
- Variables Practice- In this lesson students spend most of their time practicing using the skills and processes they have learned about variables. At the conclusion of the lesson students discuss the main things they realized and still have questions about at the conclusion of this lesson.**TI1(9-12)**
- Variables Make- Using Programming Patterns and a step-by-step approach students make their own version of a Photo Liker app. At the beginning of the lesson students are able to explore a working version of the app. They are then given the design elements of the app but begin with a blank screen. A progression of levels guides students on the high level steps they should use to develop their app but leaves it to them to decide how to write the code. At the end students submit their apps which can be assessed using a provided rubric.**TI1(9-12)**
- Conditionals Explore- Students learn the basics of conditionals through an unplugged activity using the sticky notes and plastic baggies from the Variables Explore lesson. The activity provides students a physical mental model they will be able to use when they start programming with conditionals in the subsequent lessons. Flowcharts are introduced as a way to understand how computers make decisions using Boolean expressions. **AA1(9-12)**
- Conditionals Investigate- In this lesson students work with partners to investigate three versions of the "Lemon Squeeze" app to understand how boolean expressions and conditional statements allow programs to make decisions. In each guided investigation students first watch a short video on a concept, then use a working app to predict how new features work, then investigate the code to see how those features are implemented, and finally modify the code to add expanded features. To conclude the lesson, students review and discuss common programming patterns with conditionals.**AA1(9-12)**
- Conditionals Practice- In this lesson students spend most of their time practicing using the skills and processes they have learned about conditionals. At the conclusion of the lesson students discuss the main things they realized and still have questions about at the conclusion of this lesson.**TI1(9-12)**
- Conditionals Make- Using Programming Patterns and a step-by-step approach students make their own version of a Museum Ticket Generator app. At the beginning of the lesson students are able to explore a working version of the app. They are then given the design elements of the app but begin with a blank screen. A progression of levels guides students on the high level steps they should use to develop their app but leaves it to them to decide how to write the code. At the end students submit their apps which can be assessed using a provided rubric.**TI1(9-12)**
- Functions Explore/Investigate- Students begin the lesson by considering two ways to write out the lyrics of a song, one that includes a lot of repeated text and one that does not. After exploring this example students complete a series of investigation activities in which functions have been used to remove repeated code from a program. At the conclusion of the lesson students discuss the concept of a function to synthesize their learning and add definitions to their journal.**AA1(9-12)**
- Functions Practice- In this lesson students spend most of their time practicing using the skills and processes they have learned about

functions. At the conclusion of the lesson students discuss remaining questions in anticipation of their Make project in the following lesson. **TI1(9-12)**

- Functions Make- Using Programming Patterns and a step-by-step approach students make their own version of a Quote Maker app. At the beginning of the lesson students are able to explore a working version of the app. They are then given the design elements of the app but begin with a blank screen. Students use an Activity Guide to go through the high level steps they should use to develop their app but leaves it to them to decide how to write the code. At the end students submit their apps which can be assessed using a provided rubric. **TI1(9-12)**

Teacher Resources:

Code.org online curriculum; <http://chortle.ccsu.edu/Java5/index.html#07>; Java Runtime Environment; Java Development Kit

Unit 5 - Data

Desired Results - Goals, Transfer, Meaning, Acquisition

Established Goals:

- 3A-DA-11 - Create interactive data visualizations using software tools to help others better understand real-world phenomena.
 3B-DA-05 - Use data analysis tools and techniques to identify patterns in data representing complex systems.
 3B-DA-06 - Select data collection tools and techniques to generate data sets that support a claim or communicate information.
 3A-DA-10 - Evaluate the tradeoffs in how data elements are organized and where data is stored.
 3B-AP-08 - Describe how artificial intelligence drives many software and physical systems.

Vision of A Learner Attributes: Students will be able to independently use their learning to... ("I can" statements to be demonstrated)
 TCC2(9-12): I can evaluate evidence from multiple perspectives, and recognize their limitations and implications, in order to justify new conclusions.
 TCC4(9-12): I can integrate my learning to adapt to experiences in the classroom, career and life.

Understandings: Students will understand that...

- We live in an era where data is so plentiful as to seem overwhelming and it has become increasingly important to know how to process and interpret it correctly.
- All forms of data visualization are only as good as the practices used to create them.

Essential Questions:

- When is it more appropriate to use a computer to analyze data than to complete the analysis by hand?

Students will know...

- Programs can help process data so we can understand it and learn from it.
- Correlation does not imply causation.
- Charts and other visualizations can help both find and communicate what we've learned from data.
- Bar charts and histograms are two common chart types for exploring one column of data in a table.
- We can develop insights and knowledge about our world from manipulating and visualizing data, in particular by finding patterns.
- When investigating two columns of data we can observe patterns as different values move together.

Students will be able to...

- Differentiate between what data shows and why that might be the case
- Explain the usefulness of metadata
- Create a bar chart, histogram, crosstab and scatter chart in App Lab's data visualizer
- Draw conclusions by reading bar charts, histograms, crosstab and scatter charts
- Explain the reasons that someone would create either a bar chart, histogram, crosstab and scatter chart in order to explore a single column of data
- Create filtered charts that answer specific questions
- Explain why data needs to be cleaned

<ul style="list-style-type: none"> Machine Learning occurs when we differentiate on some variable with large amounts of data. Artificial Intelligence is the result of computers using machine learning to make seemingly informed decisions. Algorithmic Bias occurs when data used in machine learning is not a fair representation of all possible appropriate data. 	<ul style="list-style-type: none"> Use the Data Visualizer to filter data Define and explain the impacts of crowdsourcing, crowdfunding, and citizen science Explain the impact of open data on scientific research and discovery Explain why in some contexts large amounts of data need to be analyzed in parallel and scalable systems Explain ways that designers and developers can consider the potential effects of their programs Reason about how human bias plays a role in machine learning. Reason about which types of tasks are should not be completed by an algorithm
<p>Key Vocabulary: Data, Correlation, Causation, Metadata, Bar Chart, Histogram, Cleaning Data, Filtering Data, Data Analysis, Crosstab Chart, Scatterplot, Open Data, Citizen Science, Crowdsourcing, Big Data, Machine Learning, AI, Algorithmic Bias</p>	
<p style="text-align: center;">Assessment Evidence</p>	
<p>Performance Tasks: Tell a Data Story Project:TCC2(9-12) and TCC4(9-12)</p> <p>This is a two-day project where students use the Data Analysis Process to tell a data story.</p> <p>Students will:</p> <ul style="list-style-type: none"> Create an effective visualization Describes new insights or decisions that can be made based on a visualization Explain information in a visualization Follow the Data Analysis Process to tell a data story Follow the Data Analysis Process to tell a data story Recognize and explain potential bias in a dataset or interpretation Write a short explanation of a data set referencing the metadata 	<p>Other Evidence:</p> <p>Unit 5 Quiz - This quiz consists of a series of released problems from AP CS Principles Exams, and requires students to assess what snippets of code do, debug code, or to evaluate different ways of accomplishing the same task.</p> <p>Code.org End of Lesson Checks for Understanding - Part of the code.org curriculum materials include 1-3 questions that summarize learning from the lesson in an open ended format. These questions can easily be shared out by the course instructor to create further discussion.</p> <p>Code.Org Unit 5 Assessment - This end-of-the-unit task assesses student understanding through a variety of multiple choice questions that cover topics from this unit.</p>

Learning Plan

In this unit learn how data analysis helps turn raw data into useful information about the world. Learn how to use data visualization to find patterns inside of data sets and learn how this data analysis process is being used in contexts like open data or machine learning to help make decisions or learn more about our world. In the unit project, you'll analyze a dataset of your choosing and present your findings.

- Learning from Data- In this lesson students explore the "What's Going on in this Graph?" site in order to tell a "data story" which explains both what the data shows and why that might be. Following this, students are introduced to the concept of metadata and look for the metadata of datasets on App Lab.**TCC4(9-12)**
- Exploring One Column- Students will practice making conclusions from charts and learn to use the Data Visualizer in App Lab to create two different kinds of charts: a bar chart, and a histogram. The lesson begins with a quick prompt to review the reasons charts are useful for looking at data. Students then practice reading a bar chart and review the kinds of questions it is and is not useful for answering. Afterwards they build different bar charts in the Data Visualizer and discuss why some are or are not useful. Afterwards they learn how to make histograms for building charts in instances where bar charts may not be useful. Students record their work on an activity guide. The lesson concludes with a brief review of what they learned and an introduction to the Data Analysis Process.**TCC2(9-12)**
- Filtering and Cleaning Data- In this lesson, students explore the challenges of working with a messy dataset. First students learn how to identify issues using the Data Visualizer, and then manually clean the data. Following this, students learn about the filtering tools in the Data Visualizer, and use a guided activity to answer data questions that require filtering a dataset. **TCC4(9-12)**
- Exploring Two Columns- Students will practice making conclusions from charts and learn to use the Data Visualizer in App Lab to create two different kinds of charts, a cross tab, and a scatterplot. Students will practice reading each type of chart before learning to make them in the Data Visualizer. Students will track their work using a provided activity guide. The lesson concludes with a review of key takeaways.**TCC2(9-12)**
- Big, Open, and Crowdsourced Data- Students will complete a jigsaw of three different topics at the intersection of data, computing, and global impacts. These are topics, big data, crowdsourcing, and open data. Students will watch videos or listen to audio recordings about the different topics. Groups will each complete an activity guide about their topic before having individuals from each group share out their findings. The lesson concludes with a review of key points.**TCC4(9-12)**
- Machine Learning- In this lesson, students are introduced to the concepts of Artificial Intelligence and Machine Learning using the AI for Oceans widget. First students classify objects as either "fish" or "not fish" to attempt to remove trash from the ocean. Then, students will need to expand their training data set to include other sea creatures that belong in the water. In the second part of the activity, students will choose their own labels to apply to images of randomly generated fish. This training data is used for a machine learning model that should then be able to label new images on its own.**TCC2(9-12)**
- Algorithmic Bias- In this lesson, students will practice cropping images to uncover the bias underlying the Twitter cropping algorithm. Then, students will read and watch a video about the discovery of this biased algorithm. Finally, students will discuss ways to recognize and reduce bias along with analyzing Twitter's response to the allegations of bias in their cropping algorithm.**TCC4(9-12)**

Teacher Resources:

Code.org online curriculum; <http://chortle.ccsu.edu/Java5/index.html#07>; Java Runtime Environment; Java Development Kit

Unit 6 - Lists, Loops, and Traversals

Desired Results - Goals, Transfer, Meaning, Acquisition

Established Goals:

- 3A-AP-14 - Use lists to simplify solutions, generalizing computational problems instead of repeated use of simple variables.
 3B-AP-23 - Evaluate key qualities of a program through a process such as a code review.
 3A-AP-15 - Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance and explain the benefits and drawbacks of choices made.
 3A-DA-12 - Create computational models that represent the relationships among different elements of data collected from a phenomenon or process.
 3B-AP-10 - Use and adapt classic algorithms to solve computational problems.

Vision of A Learner Attributes: Students will be able to independently use their learning to... ("I can" statements to be demonstrated)

TI1(9-12): I can implement a realistic plan and adapt when necessary to achieve my goals.

AA1(9-12): I can evaluate different approaches and justify the best pathway to success.

Understandings: Students will understand that...

- To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.

Essential Questions:

- What types of abstractions are available to programmers and how are they used?
- What common algorithms allow us to process data collections?

Students will know...

- A list is an ordered collection of elements.
- An element is an individual value in a list that is assigned a unique index.
- An index is a common method for referencing the elements in a list or string using numbers.
- The length of a list is how many elements it contains.
- Lists can grow or shrink as elements are added or removed.
- Lists are an example of data abstraction.
- Loops allow for repeated behavior.
- Loops can allow programs to traverse through all elements in a list.
- A while loop repeats until a specific condition is met.
- A for loop puts all three key elements of a loop into a single line

Students will be able to...

- Use an index to reference specific elements in a list.
- Use appropriate vocabulary to describe lists, loops, and traversals.
- Explain the purpose of programming patterns with lists both in terms of how they work and what they accomplish
- Identify common programming patterns using lists, loops, and traversals.
- Modify apps that make use of common programming patterns with lists, loops, and traversals to adjust their functionality
- Accurately use list operations including accessing, inserting, and removing elements
- Correctly set up a list in a program
- Debug programs with lists, loops, and traversals.

<p>of code.</p> <ul style="list-style-type: none"> • Bugs can cause loops to repeat infinitely. • We use a for loop to traverse a list, one element at a time. 	<ul style="list-style-type: none"> • Implement programming patterns with lists, loops, and traversals to develop a functioning app • Recognize the need for programming patterns with lists, loops, and traversals as part of developing a functioning app • Use debugging skills as part of developing an app • Write comments to clearly explain both the purpose and function of different segments of code within an app • Identify the exit point of a loop. • Trace a simple program with a loop • Read programs that use for loops • Understand the parts of a for loop • Update the Boolean expression in a for loop to change how many times the loop runs • Use a for-loop to update multiple screen elements at once • Understand how to use a loop to traverse a list • Write programs that use list traversals, including the filter and reduce patterns, with the support of sample code
<p>Key Vocabulary: List, Append, Insert, Remove, Element, Index, Loop, While Loop, For Loop, Iteration, Infinite Loop</p>	
<p>Assessment Evidence</p>	
<p>Performance Tasks: Hackathon Project: TI1(9-12) and AA1(9-12)</p> <ul style="list-style-type: none"> • Students determine a dataset to design the app around, and create a paper prototype. • Students fill out tables of information on element IDs and programming constructs before each tackling a different role in the project as a designer or a programmer. • Students build their apps and complete a Written Response, individually answering prompts about the project. • Students share their apps during a gallery walk. 	<p>Other Evidence:</p> <p>Unit 6 Quiz - This quiz consists of a series of released problems from AP CS Principles Exams, and requires students to assess what snippets of code do, debug code, or to evaluate different ways of accomplishing the same task.</p> <p>Code.org End of Lesson Checks for Understanding - Part of the code.org curriculum materials include 1-3 questions that summarize learning from the lesson in an open ended format. These questions can easily be shared out by the course instructor to create further discussion.</p> <p>Code.Org Unit 6 Assessment - This end-of-the-unit task assesses student understanding through a variety of multiple choice questions</p>

that cover topics from this unit.

Learning Plan

This unit introduces lists, loops, and traversals, and explores the way they can be used to build apps that store and process large amounts of information. Learn to program with the data library in App Lab and complete a 5-day hackathon project at the end of the unit where you can design a program about any topic of your choosing.

- Lists Explore- Students will learn the ways that lists are created, accessed, and changed through a teacher-guided activity using plastic baggies and pieces of paper. The lesson begins with a brief reflection on the value of lists. Students then complete the main activity which introduces the syntax to use lists and the ways they can be used. To wrap up, students watch two short videos on lists and record the main concepts in their journals. **AA1(9-12)**
- Lists Investigate- In this lesson students work with partners to investigate three different apps that use lists. Students first explore all three apps without seeing the code to notice similarities and predict how they will work. Then they explore the code itself and make additions and modifications to the apps. Students are also introduced to the getColumn command and how to use it to populate lists from the datasets in the AppLab Data Library. To conclude the lesson, students review and discuss common programming patterns with conditionals. **AA1(9-12)**
- Lists Practice- Practice the basics of lists including creating lists and accessing, inserting, and removing elements from lists. **TI1(9-12)**
- Lists Make- Using Programming Patterns and a step-by-step approach students make their own version of a Reminder app. At the beginning of the lesson students are able to explore a working version of the app. They are then given the design elements of the app but begin with a blank screen. Students use an Activity Guide to go through the high level steps they should use to develop their app but leaves it to them to decide how to write the code. At the end students submit their apps which can be assessed using a provided rubric. **TI1(9-12)**
- Loops Explore- Students begin the lesson by discussing the purpose of loops before completing the unplugged activity. This activity involves moving a "robot" around a game board while practicing tracing blocks of code by hand. To conclude, the lesson is wrapped up with a vocabulary discussion and a video. **AA1(9-12)**
- Loops Investigate- Students practice using the for loop in order to repeatedly run pieces of code. The lesson begins with a quick investigation of an app that flips coins. After that code investigation students complete another investigation with an app that uses loops to update screen elements. **AA1(9-12)**
- Loops Practice- Students practice the basics of loops including using while loops, for loops, and updating multiple screen elements with a for loop. Along the way students develop debugging practices with loops. **TI1(9-12)**
- Loops Make- Using Programming Patterns and a step-by-step approach students make their own version of a Lock Screen Maker app. At the beginning of the lesson students are able to explore a working version of the app. They are then given the design elements of the app but begin with minimal starting code. A progression of levels guides students on the high level steps they should use to develop their app but leaves it to them to decide how to write the code. At the end students submit their apps which can be assessed using a provided rubric. **TI1(9-12)**
- Traversals Explore- The lesson begins with a quick review of lists and loops before moving into the main activity. Here students explore the concept with the Traversal Machine, a physical model of traversal using a for loop. **AA1(9-12)**

- Traversals Investigate- In this lesson students work with partners to investigate three different apps that use traversals to access items in a list. Students first explore all three apps without seeing the code to notice similarities and predict how they will work. Then they explore the code itself and make additions and modifications to the apps. To conclude the lesson, students review and discuss common programming patterns with traversals. **AA1(9-12)**
- Traversals Practice- Students practice traversing lists, filtering and reducing lists, and using the data import tools. Along the way students develop debugging practices with traversals. **TI1(9-12)**
- Traversals Make- Using Programming Patterns and a step-by-step approach students make their own version of a Random Forecaster app. At the beginning of the lesson students are able to explore a working version of the app. They are then given the design elements of the app but begin with a blank screen. Students use an Activity Guide to go through the high level steps they should use to develop their app but leaves it to them to decide how to write the code. At the end students submit their apps which can be assessed using a provided rubric. **TI1(9-12)**

Teacher Resources:

Code.org online curriculum; <http://chortle.ccsu.edu/Java5/index.html#07>; Java Runtime Environment; Java Development Kit

Unit 7 - Parameters, Return, and Libraries

Desired Results - Goals, Transfer, Meaning, Acquisition

Established Goals:

2-AP-14 - Create procedures with parameters to organize code and make it easier to reuse.
 3B-AP-23 - Evaluate key qualities of a program through a process such as a code review.
 3B-AP-16 - Demonstrate code reuse by creating programming solutions using libraries and APIs.
 2-AP-17 - Systematically test and refine programs using a range of test cases.
 3A-AP-18 - Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.

Vision of A Learner Attributes: Students will be able to independently use their learning to... ("I can" statements to be demonstrated)

TI1(9-12): I can implement a realistic plan and adapt when necessary to achieve my goals.

AA1(9-12): I can evaluate different approaches and justify the best pathway to success.

Understandings: Students will understand that...

- Programmers break down problems into smaller and more manageable pieces.
- By creating procedures and leveraging parameters, programmers generalize processes that can be reused.
- Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.

Essential Questions:

- How do programmers utilize the work of one another to improve software?
- What information is needed by, and provided by, a function?

Students will know...

- A parameter is a variable in a function definition.
- An argument is the value passed by a parameter.
- A return statement is used to return the flow of control to the point where the function was called as well as to return a value of an expression.
- Functions with parameters and return values help us simplify code.
- Functions can only return one value at a time.
- A function can have:

Students will be able to...

- Remove specifics from a function so that it can be used in a variety of situations
- Use appropriate vocabulary to describe parameters, return values, and libraries.
- Explain the benefits of using a function with a parameter or return value in the context of a specific program
- Identify situations in which a function with a parameter or return value would be necessary
- Modify programs that use functions with parameters and return

<ul style="list-style-type: none"> ○ No parameters and return values ○ Parameters, but no return values ○ Return values, but no parameters ○ Parameters and return values ● Extracting shared features to generalize functionality is known as procedural abstraction. ● Using parameters allows a function to be generalized. ● Using procedural abstraction helps improve code readability. ● Procedural abstraction manages complexity by allowing code reuse. ● A library is a group of functions that may be used in creating new programs. ● An API specifies how functions in a library behave and can be used. ● Modularity is the subdivision of a computer program into separate subprograms. 	<ul style="list-style-type: none"> ● Use the modulus operator in a program ● Correctly set up a parameter in a function ● Correctly set up a return value in a function ● Write comments to explain the function purpose, parameters, and return values ● Debug programs that use functions with parameters and return ● Write functions with parameters and return values that meet a set of specified requirements ● Clearly write documentation for functions in a library ● Explain the process of preparing a function to be added to a library ● Explain the purpose of libraries as a way to simplify programs, allow for code reuse, and enable collaboration. ● Identify the use of a library within a program. ● Test the functions in libraries in order to understand their behavior ● Debug library code to remove any errors ● Read library code documentation in order to select the proper functions in the library to develop an app ● Test functions designed to be used in a library using different input values
Key Vocabulary: Argument, Parameter, Return, Procedural Abstraction, Library, Documentation, API, Modularity	
Assessment Evidence	
<p>Performance Tasks: Make a Library Project:TI1(9-12): and AA1(9-12)</p> <ul style="list-style-type: none"> ● Students design a library of functions. ● Students brainstorm common problems they've encountered while programming this year and begin to design functions that address those common problems. ● Students develop tests for their functions to make sure they return values as expected. ● Students exchange their library with a classmate for feedback. ● Students complete their library project, finalizing their code and 	<p>Other Evidence:</p> <p>Unit 7 Quiz - This quiz consists of a series of released problems from AP CS Principles Exams, and requires students to assess what snippets of code do, debug code, or to evaluate different ways of accomplishing the same task.</p> <p>Code.org End of Lesson Checks for Understanding - Part of the code.org curriculum materials include 1-3 questions that summarize learning from the lesson in an open ended format. These questions can easily be shared out by the course instructor to create further</p>

writing written responses explaining the way one of the functions in their library works.	discussion. Code.Org Unit 7 Assessment - This end-of-the-unit task assesses student understanding through a variety of multiple choice questions that cover topics from this unit.
Learning Plan	
<p>This unit introduces parameters, return, and libraries. Learn how to use these concepts to build new kinds of apps as well as libraries of code that you can share with your classmates. End the unit by designing a library of functions around any topic of your choosing.</p> <ul style="list-style-type: none"> Parameters and Return Explore- Students work with envelopes and paper to model functions with parameters and return values. Students create their own physical function envelope for drawing a house that takes in different parameters, and then build another function to calculate and return the cost of building that house. AA1(9-12) Parameters and Return Investigate- In this lesson, students work with partners to investigate two different apps that use parameters and return values. AA1(9-12) Parameters and Return Practice- Students practice writing programs with parameters and return values by creating and debugging functions that use them. TI1(9-12) Parameters and Return Make- Students use the knowledge and skills they've developed working with parameters and return values to develop a Rock Paper Scissors App. Unlike typical Make projects, students are given a significant portion of the starter code but are given three functions with parameters and return values that they'll need to develop themselves. At the end students submit their apps which can be assessed using a provided rubric. TI1(9-12) Libraries Explore- Students learn the basics of libraries while building upon the envelope model of a function with a folder to represent a library. AA1(9-12) Libraries Investigate- In this lesson students work with partners to investigate two apps that use libraries as well as the code used to make a library. Through a series of guided discussions they learn how libraries help programmers simplify and reuse their code. AA1(9-12) Libraries Practice- Students practice important skills for working with libraries, including testing and debugging libraries, and using libraries to help develop apps. After a brief introduction to these practices by the teacher, students spend the majority of their time programming in a level progression. TI1(9-12) 	
Teacher Resources: Code.org online curriculum; http://chortle.ccsu.edu/Java5/index.html#07 ; Java Runtime Environment; Java Development Kit	

Unit 8 - Cybersecurity and Global Impacts

Desired Results - Goals, Transfer, Meaning, Acquisition

Established Goals:

- 2-IC-23 - Describe tradeoffs between allowing information to be public and keeping information private and secure.
- 3A-IC-29 - Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.
- 3A-IC-30 - Evaluate the social and economic implications of privacy in the context of safety, law, or ethics.
- 3B-IC-28 - Debate laws and regulations that impact the development and use of software.
- 3A-NI-05 - Give examples to illustrate how sensitive data can be affected by malware and other attacks.
- 3A-NI-06 - Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.
- 3A-NI-07 - Compare various security measures, considering tradeoffs between the usability and security of a computer system.
- 3B-NI-04 - Compare ways software developers protect devices and information from unauthorized access.

Vision of A Learner Attributes: Students will be able to independently use their learning to... ("I can" statements to be demonstrated)

- TCC1(9-12): I can ask purposeful, insightful questions to find a variety of innovative solutions.
- CCE1(9-12): I can initiate discussions with my peers and teachers about a variety of topics, respecting differing viewpoints, actively listening to others, and responding thoughtfully with peer-reviewed evidence that is free of bias.
- AA4(9-12): I can create opportunities to extend my learning by remaining open-minded in any situation.

Understandings: Students will understand that...

- While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.
- The use of computing innovations may involve risks to personal safety and identity.

Essential Questions:

- Are innovators responsible for the harmful effects of their computing innovations even if those effects were unintentional? Why or why not?
- What data is generated by smartphones, and what is it being used for?

Students will know...

- Computing innovations can be physical or non-physical, and can be hardware, software, or concepts.
- Personally Identifiable Information is information that identifies, links, relates, or describes individuals.
- Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.

Students will be able to...

- Define Personally Identifiable Information as information about an individual that identifies, links, relates, or describes them.
- Describe the different types of data that are used and collected by modern computing innovations
- Explain how disparate pieces of personal information can be combined to identify individuals or deduce other private information.

<ul style="list-style-type: none"> • Geolocation, cookies, and browsing history can all be used to create knowledge about an individual. • Many services and websites collect information that can be used to advertise to you by creating detailed profiles of who you are and what you like. • Once data is made digital, and especially once it's shared online, it's much harder to control. • Our private data powers a lot of computing innovations in ways we like. It makes products that are convenient, interesting, personal, useful, and often "free" because we "pay" with our data. • Not every effect of a computing innovation is anticipated in advance. • The balance between innovation, privacy, and security is continually being debated. • Significant legal and ethical concerns are raised by computing innovations. • Phishing is a technique that attempts to trick a user into providing personal information. • Keylogging is the use of a program to record keystrokes made by a computer user in order to gain fraudulent access to passwords and other confidential information. • Malware is software intended to damage a computing system or take partial control over its operation. • Encryption is a process for encoding messages. Decryption is a process that reverses encryption. • Multi Factor Authentication requires a user to prove their identity in two different ways. 	<ul style="list-style-type: none"> • Assess a computing innovation to identify the specific privacy risks that could arise from the data it collects and stores. • Evaluate whether the benefits to society from a given computing innovation outweigh the privacy risks it poses. • Explain the risks to privacy that arise from using modern computing technology • Discuss the warning signals for these common security risks • Explain how these common security risks target people • Identify common security risks: phishing, keylogging, malware, rogue access points • Confidently explain security risks and their impact on society • Describe the role human error played in the Equifax breach • Explain how computing tools can be used for decryption • Explain the difference between asymmetrical and symmetrical encryption • Identify why Caesar Cipher and Random Substitution Ciphers are not adequate for most encryption needs • Discuss the benefits of computer virus scanning software and the need for regular updates • Explain the benefits of multi factor authentication • Thoughtfully answer a stimulus question
Key Vocabulary: Computing Innovation, Data Policy, Privacy, PII, Geolocation, Cookies, Browsing History, Keylogging, Phishing, Malware, Rogue Access Point, Cybersecurity, Encryption, Decryption, Cipher, Caesar's Cipher, Public Key Encryption, Multi Factor Authentication	
Assessment Evidence	
Performance Tasks: Innovation Simulation Project: TCC1(9-12), CCE1(9-12), and AA4(9-12)	Other Evidence: Unit 8 Quiz - This quiz consists of a series of released problems from

<ul style="list-style-type: none"> • In this simulation, students take on the roles of different stakeholders in school communities converging at a convention where they eventually will deliver a proposal on the best computing innovation for the Future School. • Students research three different computing innovations and discuss these innovations with team members. • Students consider the unintended consequences their proposed innovations may have. • Students collectively narrow down the proposals of their team members to a set that seems most beneficial and presents a coherent vision for the Future School. • Students develop a shared artifact or presentation that presents their collective vision for the Future School. • Students get feedback on their group innovation proposal from one other group. • Students present the group Innovation Proposal and participate in a gallery walk of their classmates' projects, asking questions and reviewing any materials. • Students vote for the unified project and overall innovation they found most compelling. 	<p>AP CS Principles Exams, and requires students to assess what snippets of code do, debug code, or to evaluate different ways of accomplishing the same task.</p> <p>Code.org End of Lesson Checks for Understanding - Part of the code.org curriculum materials include 1-3 questions that summarize learning from the lesson in an open ended format. These questions can easily be shared out by the course instructor to create further discussion.</p> <p>Code.Org Unit 8 Assessment - This end-of-the-unit task assesses student understanding through a variety of multiple choice questions that cover topics from this unit.</p>
--	--

Learning Plan

In this unit learn how computing innovations have impacted our world in beneficial and harmful ways. Learn how data can pose a threat to our privacy and security and the ways that encryption and other techniques are used to protect it. Throughout the unit participate in a "school of the future" conference in which you and a team make a proposal for how best to improve school life with computing innovations.

- Data Policies and Privacy- Students learn that the apps, websites, and other computing innovations they use every day require a lot of data to run, much of which they might consider to be private or personal. In the warm up students discuss which of a list of possible information types they consider private. Then students read the data policies from a website or service they use or know about. This investigation focuses on the kinds of data that are being collected, the way it's being used, and any potential privacy concerns that arise. A brief second activity reveals that even data that may not seem private, like a birthdate or zip code, can be combined to uniquely identify them. To conclude the lesson students prepare for a discussion in the following class about the pros and cons of sharing all this data by journaling about their current thoughts on whether the harms of giving up this privacy are outweighed by the benefits of the technology they power.**TCC1(9-12), CCE1(9-12), and AA4(9-12)**
- The Value of Privacy- Students develop their own opinions on the privacy tradeoffs inherent in many modern computing innovations. At the beginning of the lesson students watch a video on facial recognition technology that presents the tradeoffs between convenience and privacy and asks them to determine whether they think the tradeoff is worth it. Students respond to two videos that look at different tradeoffs between privacy, security, and convenience. Students then evaluate the website or app they investigated in the previous lesson

to determine if they think the benefits of the service outweigh the privacy risks. At the end of the class students discuss whether they generally think convenience outweighs privacy concerns. **TCC1(9-12), CCE1(9-12), and AA4(9-12)**

- Security Risks Part 1- Students investigate three different common security risks (phishing, keylogging, malware) in a jigsaw activity. In groups, students create Public Service Announcement slides warning of the dangers of their assigned security risk. Then students are grouped with students who investigate other security risks and are instructed to share their slide and give a voice over. The activity ends with the class coming together to discuss the security risks as a whole. **TCC1(9-12), CCE1(9-12), and AA4(9-12)**
- Security Risks Part 2- The lesson begins with a review of security risks by watching a video on Cybersecurity & Crime. Following this, the class does an investigation into the Equifax breach, and what went wrong. The class ends with a Kahoot quiz to review security risks. **TCC1(9-12), CCE1(9-12), and AA4(9-12)**
- Protecting Data Part 1- In this lesson students explore two different encryption widgets: The Caesar Cipher Widget and the Random Substitution Cipher. Afterwards, students watch a video that reviews these types of encryption and introduces a new concept: public key encryption. **TCC1(9-12), CCE1(9-12), and AA4(9-12)**
- Protecting Data Part 2- This lesson is a guided tour of multifactor authentication and software updates that students can use to protect their data. Following these discussions, students are introduced to a stimulus question where they will apply their knowledge gained throughout the unit to answer questions about an innovations data, benefits and harms, effects, and security or privacy concerns. **TCC1(9-12), CCE1(9-12), and AA4(9-12)**

Teacher Resources:

Code.org online curriculum; <http://chortle.ccsu.edu/Java5/index.html#07>; Java Runtime Environment; Java Development Kit

Unit 9 - Algorithms

Desired Results - Goals, Transfer, Meaning, Acquisition

Established Goals:

3B-AP-10 - Use and adapt classic algorithms to solve computational problems.

3B-AP-11 - Evaluate algorithms in terms of their efficiency, correctness, and clarity.

Vision of A Learner Attributes: Students will be able to independently use their learning to... (“I can” statements to be demonstrated)

TCC3(9-12): I can integrate relevant information to produce multiple valid solutions.

AA1(9-12): I can evaluate different approaches and justify the best pathway to success.

AA2(9-12): I can assess my past successes and mistakes to change my approach.

Understandings: Students will understand that...

- There exist real-world problems that computers cannot solve, and even when a computer can solve a problem, it may not be able to do so in a reasonable amount of time.
- Algorithms allow for the generalization and repetition of a set of directions to complete a task or meet an objective.

Essential Questions:

- What common real-world problems can be solved with computer processed algorithms?
- Are there problems that no algorithm can solve?

Students will know...

- Algorithmic efficiency is a measure of how many steps are needed to complete an algorithm.
- Linear Search is a search algorithm which checks each element of a list, in order, until the desired value is found or all elements in the list have been checked.
- Binary Search is a search algorithm that starts at the middle of a sorted set of numbers and removes half of the data, repeating the process until the desired value is found or all elements have been eliminated.
- Some algorithms can be solved in reasonable time, some cannot. This refers to how viable it is to perform an algorithm considering present day computing powers.
- The Travelling Salesman Problem is an example of a problem that can't be solved in reasonable time.
- Heuristics are good for problems that cannot be solved in reasonable time and when a “good enough” solution is sufficient.

Students will be able to...

- Explain that some algorithms may look or operate differently but still solve the same problem.
- Explain that some problems may look different but be similar or look similar but be different.
- Explain the formal definitions of a problem, an algorithm, sequencing, selection, and iteration.
- Compare the efficiency of Linear Search and Binary Search
- Use Binary Search to determine if a number is in a list
- Use Linear Search to determine if a number is in a list
- Explain how both formal mathematical reasoning and informal measurement can be used to determine an algorithms efficiency
- Explain the difference between problems that run in a reasonable time and those that do not
- Determine if an algorithm runs in unreasonable time.
- Develop a heuristic to solve a problem.
- Distinguish between decision problems and optimization problems.

<ul style="list-style-type: none"> • Sequential Computing is when programs run in order, one command at a time. • Parallel computing is when programs are broken into small pieces, some of which are run simultaneously. • Distributed computing means that a program is run by multiple devices. • Speedup is the time used to complete a task in parallel. 	<ul style="list-style-type: none"> • Explain the existence of undecidable problems • Calculate the speedup of a parallel solution to a problem • Describe the benefits and challenges of parallel and distributed computing. • Explain the difference between sequential, parallel, and distributed computing.
Key Vocabulary: Algorithm, Sequencing, Selection, Iteration, Efficiency, Linear Search, Binary Search, Unreasonable Time, Traveling Salesman Problem, Heuristic, Distributed Algorithm, Parallel Algorithm, Speedup,	
Assessment Evidence	
<p>Performance Tasks: Create Task: TCC3(9-12), AA1(9-12), and AA2(9-12)</p> <p>During the task, students will write a program about a topic entirely of their choosing and then write written responses explaining the purpose, functionality, data abstractions, and procedural abstractions they used. They will have at least 12 class hours to complete the task, at the end of which they will submit their work to College Board as part of their AP exam grade.</p>	<p>Other Evidence:</p> <p>Unit 9 Quiz - This quiz consists of a series of released problems from AP CS Principles Exams, and requires students to assess what snippets of code do, debug code, or to evaluate different ways of accomplishing the same task.</p> <p>Code.org End of Lesson Checks for Understanding - Part of the code.org curriculum materials include 1-3 questions that summarize learning from the lesson in an open ended format. These questions can easily be shared out by the course instructor to create further discussion.</p> <p>Code.Org Unit 9 Assessment - This end-of-the-unit task assesses student understanding through a variety of multiple choice questions that cover topics from this unit.</p>
Learning Plan	
<p>This unit is a quick exploration of how computer scientists design algorithms to solve problems and how they analyze the speed of different algorithms. Learn about the concept of algorithmic efficiency through a variety of hands-on activities and learn how it's being applied in modern computing.</p> <ul style="list-style-type: none"> • Algorithms Solve Problems- Students will complete two exploratory activities that introduce the concept of a problem and an algorithm. In the first students answer a series of questions about birthdates and names of their classmates. They then discuss the similarities and 	

differences between the problems. In the second activity students are given six different algorithms and must analyze them to determine which they think are the same or different. At the end of the lesson they are introduced to the formal definitions of a problem and an algorithm. **TCC3(9-12)**

- Algorithm Efficiency- In this lesson students follow a demonstration of Linear Search before writing their own search algorithms. Following this, students are introduced to Binary Search after which they compare graphs of the search algorithms to determine which is most efficient. **AA2(9-12)**
- Unreasonable Time- Students investigate two different types of raffles that highlight the way seemingly small problems can quickly grow large. The first raffle asks students to hunt for pairs of tickets that add to a target value. The second raffle asks students to hunt for any group of tickets that adds to a target value. After trying out each raffle live students will try to figure out the patterns for how many total combinations need to be checked in each. At the end they discuss the difference between reasonable and unreasonable algorithms based on their experiences. **AA1(9-12)**
- The Limits of Algorithms- Students explore the limits of what algorithms are able to compute. First they use a widget that exposes students to the Traveling Salesperson Problem. After recognizing this problem can only be solved using an unreasonable time algorithm the develop their own good enough solutions that run more efficiently. Later in the lesson students watch a video about undecidable problems for which no algorithm can ever be developed to solve them. **AA1(9-12)**
- Parallel and Distributed Algorithms- In this lesson students explore the benefits and limitations of parallel and distributed computing. First they discuss the way human problem solving changes when additional people lend a hand. Then they run a series of demonstrations that show how simple tasks like sorting cards get faster when more people help, but there is a limitation to the efficiency gains. Later in the lesson students watch a video about distributed computing that highlights the ways distributed computing can help tackle new kinds of problems. To conclude the lesson students record new vocabulary in their journals and discuss any open questions. **TCC3(9-12)**

Teacher Resources:

Code.org online curriculum; <http://chortle.ccsu.edu/Java5/index.html#07>; Java Runtime Environment; Java Development Kit