Curriculum Guide
2020-2021

Computer Science Principles

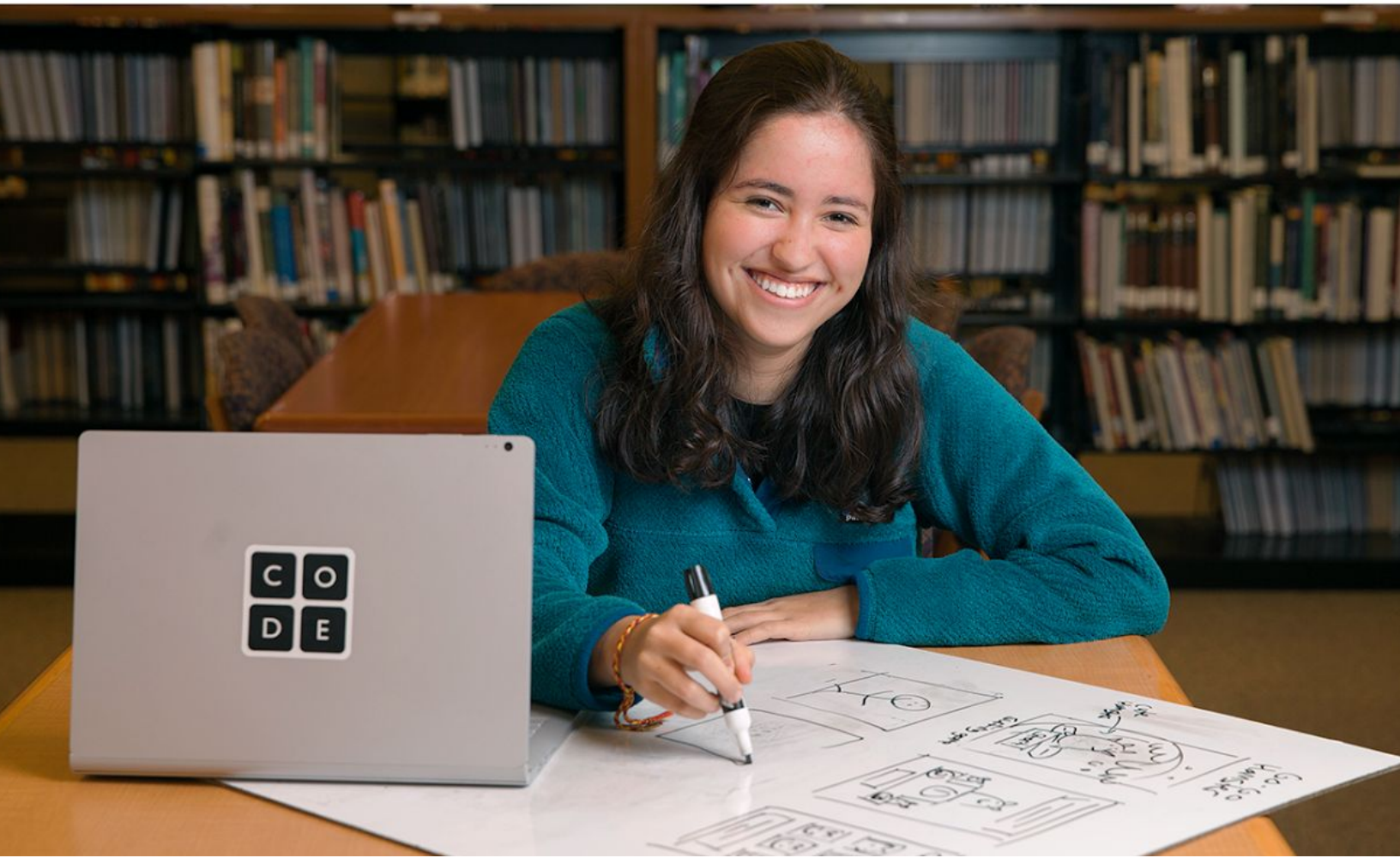


Table of Contents

Welcome to Computer Science Principles

Code.org's Computer Science Principles (CSP) curriculum is a **full-year, rigorous, entry-level course** that introduces high school students to the foundations of modern computing. The course covers a broad range of foundational topics such as programming, algorithms, the Internet, big data, digital privacy and security, and the societal impacts of computing. All teacher and student materials are provided for free online and can be accessed at code.org/csp.

Curriculum Overview and Goals

Computing affects almost all aspects of modern life, and all students deserve an education that prepares them to pursue the wide array of opportunities that computing has made possible. This course seeks to provide knowledge and skills to meaningfully participate in our increasingly digital society, economy, and culture.

Unit	Description
Unit 1 (14 lessons) Digital Information	Explore how computers store complex information like numbers, text, images and sound and debate the impacts of digitizing information.
Unit 2 (9 lessons) The Internet	Learn about how the Internet works and discuss its impacts on politics, culture, and the economy.
Unit 3 (11 lessons) Intro to App Design	Design your first app while learning both fundamental programming concepts and collaborative software development processes.
Unit 4 (15 lessons) Variables, Conditionals, and Functions	Expand the types of apps you can create by adding the ability to store information, make decisions, and better organize code.
Unit 5 (18 lessons) Lists, Loops, and Traversals	Build apps that use large amounts of information and pull in data from the web to create a wider variety of apps.
Unit 6 (6 lessons) Algorithms	Design and analyze algorithms to understand how they work and why some are considered better than others.
Unit 7 (11 lessons) Parameters, Return, and Libraries	Learn how to design clean and reusable code that you can share with a single classmate or the entire world.
Unit 8 (18 lessons) Create PT Prep	Practice and complete the Create Performance Task (PT).
Unit 9 (9 lessons) Data	Explore and visualize datasets from a wide variety of topics as you hunt for patterns and try to learn more about the world around you.
Unit 10 (14 lessons) Cybersecurity and Global Impacts	Research and debate current events at the intersection of data, public policy, law, ethics, and societal impact.

AP Endorsed

Code.org is recognized by the College Board as an endorsed provider of curriculum and professional development for AP® Computer Science Principles (AP CSP). This endorsement affirms that all components of Code.org CSP's offerings are aligned to the AP Curriculum Framework standards, the AP CSP assessment, and the AP framework for professional development. Using an endorsed provider affords schools access to resources including an AP CSP syllabus pre-approved by the College Board's AP Course Audit, and officially-recognized professional development that prepares teachers to teach AP CSP.

CS Principles Course At-A-Glance

Unit 1 - Digital Information

wk 1	Welcome to CSP Representing Information Circle Square Patterns Binary Numbers Overflow and Rounding
2	Representing Text Black and White Images Color Images Lossless Compression Lossy Compression
3	Intellectual Property Project - Digital Information Dilemmas - Parts 1-2 Assessment Day

Unit 2 - The Internet

wk 1	Welcome to the Internet Building a Network The Need for Addressing Routers and Redundancy Packets
2	HTTP and DNS Project - Internet Dilemmas - Parts 1-2 Assessment Day

Unit 3 - Intro to App Design

wk 1	Intro to Apps Introduction to Design Mode Project - Designing an App - Parts 1-2 The Need for Programming languages
2	Intro to Programming Debugging Project - Designing an App - Parts 3-5
3	Assessment Day

Unit 4 - Variables, Conditionals, and Functions

wk 1	Variables Explore Variables Investigate Variables Practice Variables Make Conditionals Explore
2	Conditionals Investigate Conditionals Practice Conditionals Make Functions Explore/Investigate Functions Practice
3	Functions Make Project - Decision Maker App - Parts 1 - 3 Assessment Day

Unit 5 - Lists, Loops, and Traversals

wk 1	Lists Explore Lists Investigate Lists Practice Lists Make Loops Explore

Unit 5 - Continued

2	Loops Investigate Loops Practice Loops Make Traversals Explore Traversals Investigate
3	Traversals Practice Traversals Make Project - Hackathon - Parts 1-3
4	Project - Hackathon - Parts 4-5 Assessment Day

Unit 6 - Algorithms

wk 1	Algorithms Solve Problems Algorithm Efficiency Unreasonable Time The Limits of Algorithms Parallel and Distributed Algorithms
2	Assessment Day

Unit 7 - Parameters, Return, and Libraries

wk 1	Parameters and Return Explore Parameters and Return Investigate Parameters and Return Practice Parameters and Return Make Libraries Explore
2	Libraries Investigate Libraries Practice Project - Make a Library - Parts 1-3
3	Assessment Day

Unit 8 - Create PT Prep

wk 1	Create PT - Review the Task Create PT - Make a Plan Create PT - Complete the Task (12 total class hours)
2	Create PT - Complete the Task (continued)
3	Create PT - Complete the Task (continued)

Unit 9 - Data

wk 1	Learning from Data Exploring One Column Filtering and Cleaning Data Exploring Two Columns Big, Open, and Crowdsourced Data
2	Machine Learning and Bias Project - Tell A Data Story - Parts 1-2 Assessment Day

Unit 10 - Cybersecurity and Global Impacts

wk 1	Project - Innovation Simulation - Parts 1-2 Data Policies and Privacy The Value of Privacy Project - Innovation Simulation - Part 3
2	Security Risks - Part 1-2 Project - Innovation Simulation - Part 4 Protecting Data - Parts 1-2
3	Project - Innovation Simulation - Parts 5-7 Assessment Day

Code.org Values and Philosophy

Curriculum Values

While Code.org offers a wide range of curricular materials across a wide range of ages, the following values permeate and drive the creation of every lesson we write.

Computer Science is Foundational for Every Student

We believe that computing is so fundamental to understanding and participating in society that it is valuable for *every* student to learn as part of a modern education. We see computer science as a liberal art, a subject that provides students with a critical lens for interpreting the world around them. Computer science prepares all students to be active and informed contributors to our increasingly technological society whether they pursue careers in technology or not. Computer science can be life-changing, not just skill training.

Teachers in Classrooms

We believe students learn best with the help of an empowered teacher. We design our materials for a classroom setting and provide teachers robust supports that enable them to understand and perform their critical role in supporting student learning. Because teachers know their students best, we empower them to make choices within the curriculum, even as we recommend and support a variety of pedagogical approaches. Knowing that many of our teachers are new to computer science themselves, our resources and strategies specifically target their needs.

Student Engagement and Learning

We believe that students learn best when they are intrinsically motivated. We prioritize learning experiences that are active, relevant to students' lives, and provide students authentic choice. We encourage students to be curious, solve personally relevant problems and to express themselves through creation. Learning is an inherently social activity, so we interweave lessons with discussions, presentations, peer feedback, and shared reflections. As students proceed through our pathway, we increasingly shift responsibility to students to formulate their own questions, develop their own solutions, and critique their own work.

Equity

We believe that acknowledging and shining a light on the historical inequities within the field of computer science is critical to reaching our goal of bringing computer science to all students. We provide tools and strategies to help teachers understand and address well-known equity gaps within the field. We recognize that some students and classrooms need more supports than others, and so those with the greatest needs should be prioritized. All students can succeed in computer science when given the right supports and opportunities, regardless of prior knowledge or privilege. We actively seek to eliminate and discredit stereotypes that plague computer science and lead to attrition of the very students we aim to reach.

Curriculum as a Service

We believe that curriculum is a service, not just a product. Along with producing high quality materials, we seek to build and nourish communities of teachers by providing support and channels for communication and feedback. Our products and materials are not static entities, but a living and breathing body of work that is responsive to feedback and changing conditions. To ensure ubiquitous access to our curriculum and tools, they are web-based and cross-platform, and will forever be free to use and openly licensed under a Creative Commons license.

Pedagogical Approach to our Values

When we design learning experiences, we draw from a variety of teaching and learning strategies all with the goal of constructing an equitable and engaging learning environment.

Role of the Teacher

We design curriculum with the idea that the instructor will act as the lead learner. As the lead learner, the role of the teacher shifts from being the source of knowledge to being a leader in seeking knowledge. The lead learner's mantra is: "I may not know the answer, but I know that together we can figure it out." A very practical residue of this is that we never ask a teacher to lecture or offer the first explanation of a CS concept. We want the class activity to do the work of exposing the concept to students allowing the teacher shape meaning from what they have experienced. We also expect teachers to act as the curator of materials. Finally, we include an abundance of materials and teaching strategies - too many to use at once - with the expectation that teachers have the professional expertise to determine how to best conduct an engaging and relevant class for their own students.

Discovery and Inquiry

We take great care to design learning experiences in which students have an active and equal stake in the proceedings. Students are given opportunities to explore concepts and build their own understandings through a variety of physical activities and online lessons. These activities form a set of common lived experiences that connect students (and the teacher) to the course content and to each other. The goal is to develop a common foundation upon which all students in the class can construct their understanding of computer science concepts, regardless of prior experience in the discipline.

Materials and Tools

Our materials and tools are specifically created for learners and learning experiences, and focus on foundational concepts that allow them to stand the test of time. They are designed to support exploration and discovery by those without computer science knowledge, so that students can develop an understanding of these concepts through "play" and experimentation. From our coding environments to other non-coding tools and videos, all our resources have been engineered to support the lessons in our curriculum, and thus our philosophy about student engagement and learning. In that vein, our videos can be a great tool for sensemaking about CS concepts and provide a resource for students to return to when they want to refresh their knowledge. They are usually packed with information and "star" a diverse cast of presenters and CS role models.

Creation and Personal Expression

Many of the projects, assignments, and activities in our curriculum ask students to be creative, to express themselves and then to share their creations with others. While certain lessons focus on learning and practicing new skills, our goal is always to enable students to transfer these skills to creations of their own. Everyone seeks to make their mark on society, including our students, and we want to give them the tools they need to do so. When computer science provides an outlet for personal expression and creativity, students are intrinsically motivated to deepen the understandings that will allow them to express their views and carve out their place in the world.

The Classroom Community

Our lessons almost always call for students to interact with other students in the class in some way. Whether learners are simply conferring with a partner during a warm up discussion, or engaging in a long-term group project, our belief is that a classroom where students are communicating, solving problems, and creating things is a classroom that not only leads to active and better learning for students, but also leads to a more inclusive classroom culture in which all students share ideas and listen to ideas of others. For example, classroom discussions usually follow a Think-Pair-Share pattern; we ask students to write computer code in pairs; and we strive to include projects for teams in which everyone must play a critical role.

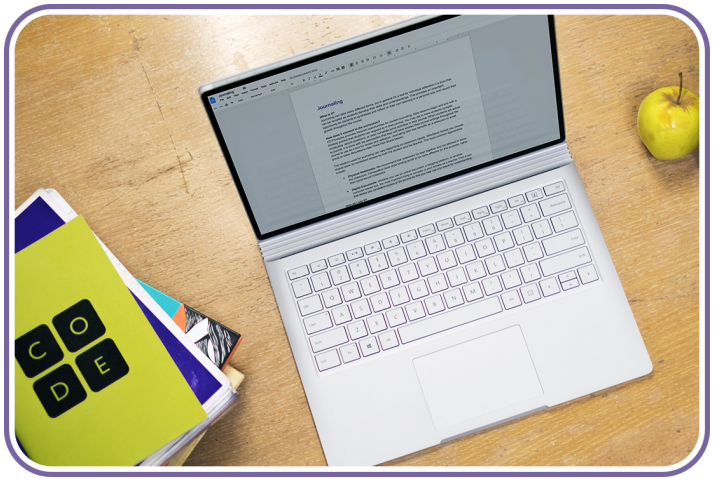
Instructional Strategies

The instructional strategies listed below are recommended for use throughout the curriculum. We believe these instructional strategies lead to positive classroom culture and ultimately student learning.

Journaling

What is it?

Journaling can take many different forms, but in general it's a tool for individual reflection in a form that can be revisited as students develop their skills and understandings. This provides an important opportunity for students to synthesize and reflect on their own learning in a personal way and record their growth throughout the course.



How does it connect to the curriculum?

CS Principles provides frequent opportunities for student journaling. Most lessons begin and end with a thinking prompt that students can respond to in their journal. You may opt to have students compile vocabulary, record questions, or even complete some activities in their journal. When students are asked to journal, it is done with the assumption that they will have access to their journal writings throughout the course to use as a tool for review and reflection. Journaling also has benefits as a precursor to small group or class discussions.

The medium used for journaling can vary depending on classroom needs. The format you choose should allow for consistent access by both the student and the teacher. The most common approaches include:

- **Physical Notebooks:** We recommend that notebooks be kept together and not allowed to leave the classroom. Composition book style binding tends to be more effective for this purpose, rather than spiral-bound notebooks.

- **Digital Documents:** Whether you use an online document, a blogging platform, or another computer-based tool, the most important thing to consider is your access as a teacher. Find a tool that allows you consistent access to the journal so that you may use it to check for understanding.

How do I use it?

- Provide students a journal at the beginning of the school year
- Prompt students to journal about specific challenges or bugs they encounter
- Give students time to revisit previous journal entries and reflect on their growth

Peer Feedback

What is it?
Peer feedback is the practice of students sharing their work with one another in order to prompt discussion, solicit suggestions, and iteratively improve their work. Peer feedback provides students opportunities to learn from each other, both by seeing ways others approach the same problem and by incorporating feedback to improve their own work.

How does it connect to the curriculum?
Throughout the CS Principles curriculum, there are many activities that have structured moments for students to give each other peer feedback. When lessons call for students to design a solution to a problem, for example, they typically will exchange early drafts with classmates to test ideas and identify potential improvements. Research and programming projects call on students to present their work to classmates in order to share what they have learned and collect ideas for approaching future projects. Many rubrics are written with the express intention that students use them to assess one another's work. These and other peer feedback opportunities integrated throughout the curriculum allow students to get quick, authentic, and personalized feedback about their work and thinking.

How do I use it?

- Create a structured peer feedback process.
- Decide who is giving who feedback.
- Allow students to share some areas that they would like feedback on.
- Give students time to provide feedback.
- Give students time to respond and incorporate feedback.
- Provide examples of constructive feedback.
- Have students use sentence starters for their feedback such as: I like… , I wish… , What if...
- Treat this as a skill that students develop throughout the course and which they will need to be taught.

Classroom Discussions

What is it?
Leading classroom discussions is the practice of bringing the whole classroom together so that students have an opportunity to share their own ideas and hear their peers' ideas. Classroom discussions typically start with a prompt provided by the teacher in the slides that is intended to achieve a specific discussion goal.

How does it connect to the curriculum?
Classroom discussions are how we provide students opportunities to draw upon previous memories or experiences, share their ideas with the room, or synthesize experiences and learning from the lesson. Sometimes the discussions are short conversations in a warm-up or wrap-up, and other times we ask teachers to orchestrate larger conversations within the activity itself. In the curriculum, you will typically find prompts for these discussions embedded in slides. Additionally, in the lesson plan you will often find a description of a "Discussion Goal" for these prompts. While many of the activities in the curriculum may be engaging for students, classroom discussions are where the learning is made more visible and concrete. The discussions also provide you as a teacher the opportunity to formatively assess your class at multiple points in the lesson and readjust instruction as needed.

How do I use it?
Most frequently, we use Think-Pair-Share as a classroom discussion strategy. This structure provides learners with time to individually collect their thoughts (during "Think") and an opportunity to engage in a low-risk discussion with a partner where they get a chance to share their ideas (during "Pair"). Finally, in this strategy allows the whole group to bubble up key ideas in the room so that everyone can hear and benefit from them (during "Share").

Sometimes using the same format frequently can get stale or portions of Think-Pair-Share might not work for every group of learners. You might have a class that are verbal processors and are eager to "think out loud" with a partner immediately, or a group of students who are reluctant to speak in a whole group situation. For these and other situations, see the variations on the next page when looking to add variety to the Think-Pair-Share routine in discussions.

Preparing for discussions.
In general, classroom discussions will be richer and more productive when you have prepared for them. To prepare for classroom discussions:

- Know what the discussion goals are so that you will know when to move on.
- Anticipate how students might respond to the prompts.
- Most importantly, during the discussion, listen. Students should do most of the talking, but your role is to listen to what they say so you can add follow-up questions or connect different student's ideas to each other and to the discussion goal. Once the discussion goal has been achieved, it is time to move on to the next part of the lesson.

Variations for Think-Pair-Share

Use this table to help you adapt classroom discussion needs to your group of students and their needs.

	Typical activity	Variations
Think	Students free-write in their own journals	• **Use sentence starters.** Provide students with sentence starters for their writing time. • **Skip the writing time.** Have students think to themselves, especially for shorter prompts. • **Go digital.** Provide a public digital forum for students to write in like Google Classroom or Socrative. • **Use whiteboards.** Have students write down their thoughts on personal whiteboards, if available.

Pair	Students share with a designated partner nearby.	• **Change partners.** Change the instructions to create variety in partnering up including partners behind, in front, diagonals, etc.
		• **Use clock or compass buddies.** To do this students select a partner for points of the compass or 4 positions on the clock. When you want students to work with a partner, call out a new time or compass direction.
		• **Use a more active method.** While this takes more time, getting more movement into the classroom can add new energy to the room, even if students seem reluctant about it at first. Some examples include:
		  ○ **Stand-up, pair-up.** Have participants stand up and pair up with a classmate. Variations include: someone wearing the same color clothes as you, your "sole-mate" (someone wearing the same shoes as you), someone at a different table as you, someone in a different grade than you, etc.
		  ○ **Speed "date".** If you want students to hear a variety of answers, you can have them "speed date" by standing up and finding a partner to share with for a set amount of time. After the time is up, they find a new partner. If speed "dating" makes the room uncomfortable, use the term "speed networking" instead.
		  ○ **Walk and talk.** If you have extra time and your building policies allow for it, have students walk in pairs and talk about the prompt in the building. Even if it is up and down the hallway, getting moving can help students generate thoughts.

Share	Students share the results of their partner conversation with the whole class	• **Random select.** Randomly choose names (e.g. from a popsicle sticks) or use a random number generator to call on students. • **Ask for a table or row to share.** Instead of calling on individuals, call from a table or row to share out. This invites others into the conversation without singling out an individual student. • **Make a circle.** Have students form a circle prior to sharing their ideas. • **Four corners.** Label each corner of the room with a common answer to the discussion prompt. Have groups move to the corner that best reflects their group's response and discuss with other classmates there. • **Share for your partner.** Ask that students only share ideas their partner came up with in the discussion

Pair Programming

What is it?

Pair programming is a technique in which two programmers work together at one computer. One, the driver, writes code while the other, the navigator, directs the driver on the design and setup of the code. The two programmers switch roles often. Pair programming has been shown to:

- improve computer science enrollment, retention, and students' performance
- increase students' confidence
- develop students' critical thinking skills
- introduce students to the “real world” working environment

How does it connect to the curriculum?

In CS Principles, there are many lessons on the computer during which students work with a partner to learn new programming skills or otherwise work with a digital tool. Whether or not the practice of Pair Programming is explicitly identified in the lesson, it is always an option. Pair programming can help to foster a sense of camaraderie and collaboration in your classroom. It has been shown to increase the enrollment, retention, and performance of students in computer science classes. It promotes diversity in the classroom by reducing the so-called “confidence gap” between female and male students, while increasing the programming confidence of all students.

How do I use it?

To get students pair programming:

1. Form pairs
2. Give each pair one computer to work on
3. Assign roles
4. Have students start working
5. Ensure that students switch roles at regular intervals (every 3 to 5 minutes)
6. Ensure that navigators remain active participants

It can be hard to introduce pair programming after students have worked individually for a while, so we recommend that teachers start with pair programming in the first few plugged lessons. Just like any other classroom technique, you may not want to use this all the time as different types of learners will respond differently to working in this context. Once you have established pair programming as a practice early on, it will be easier to come back to later.

Resources

Code.org also has a feature to help both students get “credit” on their accounts for the work they do together. Check out this article teacherblog.code.org/post/147349807334/try-pair-programmingtrack-the-progress-of on our blog about Pair Programming.

Code.org has made a video explaining pair programming to students: youtu.be/q7d_JtyCq1A

The National Center for Women & Information Technology (NCWIT) has a great resource about the benefits of pair programming. Check it out at: www.ncwit.org/resources/pair-programming-box-power-collaborative-learning

Debugging

What is it?
Debugging is the practice of finding and fixing problems. While debugging classically is used only in reference to fixing problems in program code, in this curriculum students will be asked to debug work in a variety of digital and pencil-and-paper contexts.

How does it connect to the curriculum?
Finding and fixing errors is an important skill in many computer science contexts that is emphasized repeatedly through the curriculum. For example, in programming units students will encounter explicit debugging activities where they are asked to identify and fix bugs in provided code. Students will also debug work in a number of contexts outside of programming. Lessons using the Internet Simulator or widgets (see the Tools section for details), for example, are typically framed around a bug or challenge that students must solve. While students are not writing program code, they are still creating structured solutions to a problem where careful attention to detail or unexpected behavior is critical.

It's important to build a culture of constant debugging, as this isn't an activity that is done in isolated moments. As with most things, people get better at debugging by doing it! That said, reflective strategies can help students learn more from debugging. Encourage students to talk about their bugs and how they were able to address them. Students may create bug logs in their journals, or a bug poster for the classroom. If students are too specific with the bugs that they have found, consider reframing what they say into something more generally useful (e.g. "The 'r' and the 'c' were switched." could become "My keyword was not spelled correctly.")

How do I use it?

- Emphasize debugging as a natural and expected component of creating in any CS context.
- Celebrate discovering (and fixing) new types of bugs to normalize the debugging process.
- When trying to find the source of a bug have students read their code aloud, line by line, explaining the purpose of each command.
- Ask questions about the code (and what changes were made when the bug was introduced), making sure that the students can clearly explain how the code is intended to work
- Avoid finding the bug for students, or being too specific with your questioning
- Encourage students to ask aloud the same questions that you ask them when helping them debug.
- Reinforce debugging strategies even in contexts where students aren't programming

Unplugged and Plugged Activities

Unplugged Activities

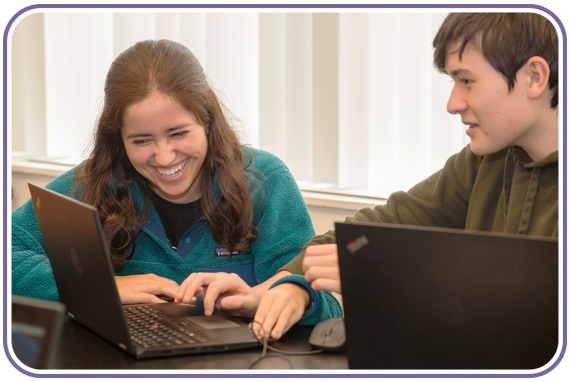
What are they?
We refer to activities where students are not working on a computer as “unplugged.” Students will often be working with pencil and paper, or physical manipulatives.

How are they used?
Unplugged activities are more than just an alternative for the days when the computer lab is full. They are intentionally placed, often kinesthetic, opportunities for students to digest concepts in approachable ways. Unplugged lessons are particularly good for building and maintaining a collaborative classroom environment, and they are useful touchstone experiences you can refer to when introducing more abstract concepts.

Tips for Effectively Teaching Unplugged Activities

- Don’t skip these activities!
- Teach units in the order they are written. The sequence is designed to scaffold student understanding.
- Help students identify the computer science concepts underlying these approachable activities.
- Refer back to unplugged activities to reinforce concepts in subsequent lessons.

Plugged Activities



What are they?
We refer to activities where students are working on a computer as “plugged.” Students may be conducting research, completing a programming assignment, or using an interactive “widget” (see the Tools section for details).

How are they used?
Plugged activities are designed to allow students to get hands-on with tools and concepts. Lessons will begin and end with discussions or activities that help motivate and synthesize learning. Students are encouraged and often even required to work with one another. Key moments for you to check in with your students are noted in lesson plans. Students will be using a computer, but the ways students interact with each other and your role as the teacher should remain largely unchanged.

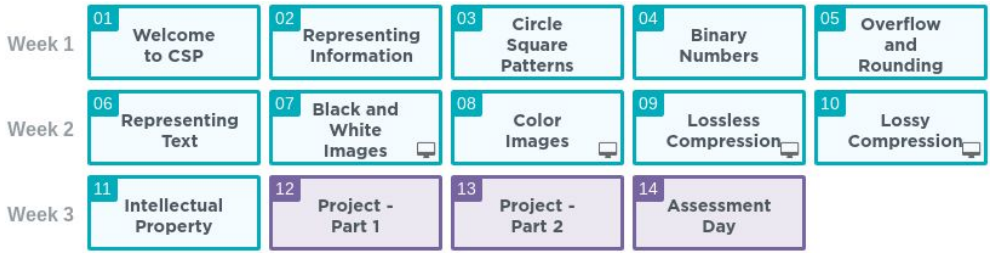
Tips for Effectively Teaching Plugged Activities

- Get to the widget quickly - do not do a lot of pre-teaching.
- Give students a chance to play with the widget before explaining it to them. Students frequently can discover how a widget works.
- After students explore the widget, have students share out so the class understands how the widget works.
- Have a strategy for knowing when students are ready to discuss what they learned from using the widget.
- Connect the computer science knowledge intended for the lesson back to the widget.
- Incorporate time for students on computers to collaborate during the lesson to keep engagement high.
- “Plugged” doesn’t mean the computer is the students’ teacher! If anything, you will need to take a more active role in checking student progress since it’s hard to know what’s happening when students are working on screens.

Code.org AP® Computer Science Principles Curriculum Overview

Unit 1 - Digital Information

Unit Overview



Students explore the way computers store and represent complex information like numbers, text, images, and sound. The unit begins with students investigating what it means to represent information, and challenges students to design their own representation systems. Students then learn the ideas behind real-world systems used to represent complex information. Later lessons focus on the challenges that arise from digitizing information, such as the need to compress it, or questions of intellectual property. The unit project emphasizes the profound impact digital information has on modern life.

Unit Philosophy and Pedagogy

Establishing a Strong Classroom Culture: This unit is designed to be hands-on, collaborative, and exploratory. A major focus of the unit is building a positive classroom culture in which students work together, explore problems, and communicate about their thinking. Most lessons either feature physical manipulatives or a digital widget, and the bulk of lesson time should be spent with students exploring these tools together to develop an understanding of the concepts they highlight. The course intentionally does not start with programming since, in many classrooms, some students have experience with programming and others do not. Choosing to begin with digital information and the internet lets you build community in the room while exploring a topic that is likely to be accessible to all students. The supportive and inclusive classroom environment built in this unit should help set a positive tone that can be carried through the school year.

Empowering "Deciders": An important goal of the course is not merely to teach students technical knowledge, but to put those skills to work in meaningful ways. This unit builds towards the unit project, which provides an opportunity for students to be "deciders" about the impacts of computing on modern life. Other units will emphasize empowering units as "creators."

Major Assessment and Projects

The unit project asks students to consider and debate issues that arise in modern society due to the digitizing of information. Students will analyze an article that addresses the intersection of digitizing information and current events. They will evaluate what data is being digitized and evaluate the benefits and harms caused by making this information digital. Students will also complete an end-of-unit assessment aligned with CS Principles framework objectives covered in this unit.

Unit 2 - The Internet

Unit Overview

Students learn how the Internet works and discuss its impacts on politics, culture, and the economy. This unit heavily features the Internet Simulator, a tool designed to let students see, use, and explore the way different layers of the internet work. Through a series of activities that build on one another, students investigate the problems the original designers of the internet had to solve and then "invent" their own solutions. At the conclusion of the unit, students research an "Internet Dilemma," both from the standpoint of its technical background and its impacts on different groups of people.

Unit Philosophy and Pedagogy

Inventing the Internet with the Internet Simulator: This unit features many different versions of the Internet Simulator, a digital widget that simulates the way different features or "layers" of the Internet work. As students move from lesson to lesson, the version of the Internet Simulator they use will have slightly more functionality than the last. Within a lesson, students will be presented with challenges that are modeled closely on those that the original inventors of the internet needed to solve. Students will collaboratively design and test solutions to those problems to develop an intuitive understanding of not just how the internet works, but why it was designed that way. By the end of the unit, students will have "invented the internet" themselves!

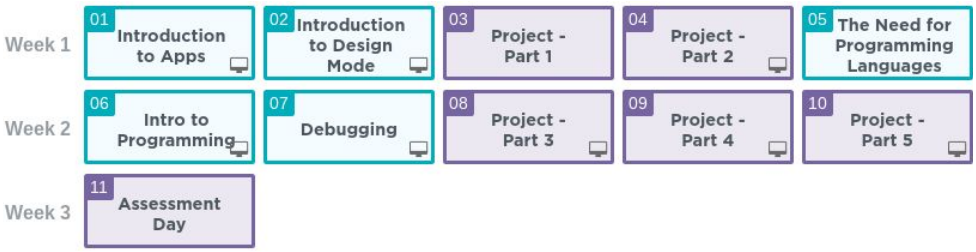
Continuing to Establish a Strong Classroom Culture: Much like the Digital Information unit that comes before it, this unit emphasizes collaborative problem solving and the development of a supportive and inclusive classroom culture.

Major Assessment and Projects

The unit project asks students to design a policy position for an imaginary political candidate related to an "Internet Dilemma." Students must analyze news stories about their topic to identify impacted groups, explain those groups interests, explain technical background about the dilemma, and then recommend a policy solution that the candidate should advocate for. Students will also complete an end-of-unit assessment aligned with CS Principles framework objectives covered in this unit.

Unit 3 - Intro to App Design

Unit Overview



Students design their first app while learning both fundamental programming concepts and collaborative software development processes. Students work with partners to develop a simple app that teaches classmates about a topic of personal interest. Throughout the unit, they learn how to use Code.org's programming environment, App Lab, to design user interfaces and write simple event-driven programs. Along the way, students learn practices like debugging, pair programming, and collecting and responding to feedback, which they will be able to use throughout the course as they build ever more complex projects. The unit concludes with students sharing the apps they develop with their classmates.

Unit Philosophy and Pedagogy

New Topics, Same Classroom Culture: This unit is students' first experience with programming but it is designed to maintain the collaborative and inclusive classroom environment developed in the previous two units. The collaborative project, fun unplugged activities, and the focus on experimenting should help keep your whole class working together and trying out ideas.

Emphasizing Skills: Since this is the first of many programming units, it emphasizes attitudes and skills that will serve your students well for the remainder of the year. The project that runs through this unit emphasizes that programming is a creative and collaborative endeavor that can be used to help others. Key practices like pair programming and debugging help normalize working with a partner, asking for help, and making mistakes. While students have a lot to learn about programming and App Lab, there is just as much emphasis on establishing these positive habits and mindsets.

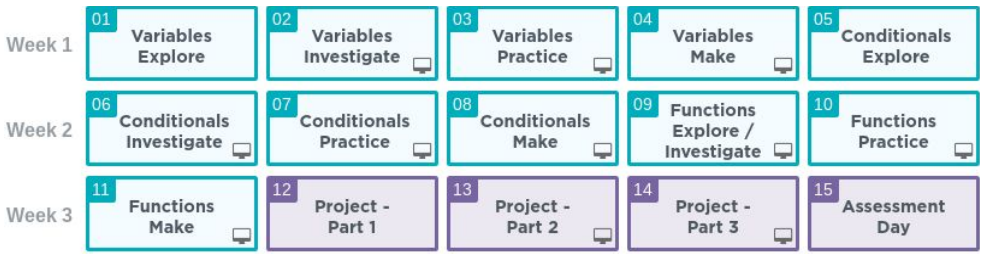
Empowering "Creators": This unit empowers students to be creators with a major emphasis on making projects that are personally meaningful. Students have a lot to learn about programming, but the goal is that they come away from this unit seeing programming as a powerful form of personal expression that allows them to draw on their innate talents and interests to help solve problems in their community.

Major Assessment and Projects

The unit project asks students to collaborate with a classmate to design an app that can teach others about a topic of shared interest. Students practice interviewing classmates to identify the goals of the project, mock up designs, collaboratively program the app, and run simple user tests. The app itself must include at least three screens and demonstrate what students have learned about user interface design and event-driven programming. Students submit their app, project guide, and written responses to reflection questions about how the app is designed and the development process used to make it. Students will also complete an end-of-unit assessment aligned with CS Principles framework objectives covered in this unit.

Unit 4 - Variables, Conditionals, and Functions

Unit Overview



Students expand the types of apps they can create as they learn how to store information (variables), make decisions (conditionals), and better organize code (functions). Each programming topic is covered in a specific sequence of lessons that ask students to 'Explore' ideas through hands-on activities, 'Investigate' these ideas through guided code reading, 'Practice' with sample problems, and apply their understanding as they 'Make' a one-day scoped project. The entire unit concludes with a three-day open-ended project in which students must build an app that makes a recommendation about any topic they wish.

Unit Philosophy and Pedagogy

Intro to EIPM: This unit is students' first experience with the Explore, Investigate, Practice, Make lesson sequence, or EIPM. This structured approach to teaching programming is covered in detail in the curriculum guide and we highly recommend that you watch the accompanying video series to better understand what EIPM should look like in the classroom. When used effectively, it supports deep learning of content and helps maintain a collaborative classroom culture, even as you move into more complex programming concepts.

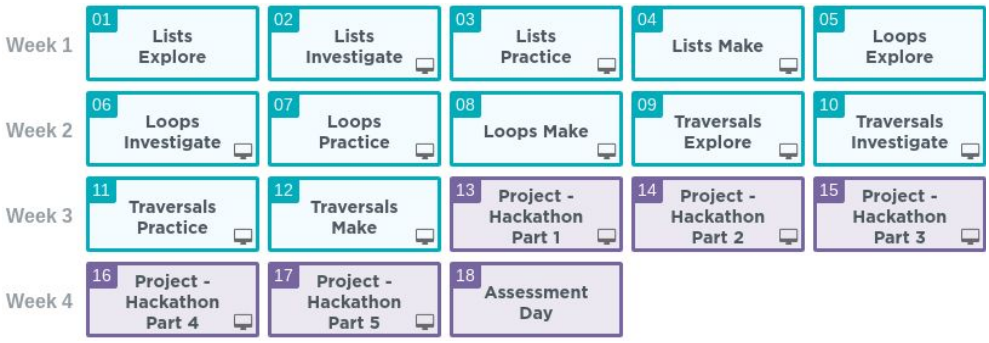
Scaffolding Towards Independent Projects: A major goal of this course is to empower students to design and build projects independently. The Create PT in Unit 8 offers students enormous freedoms to scope and build projects, and even this unit begins scaffolding towards that goal. Individual EIPM sequences of lessons gradually prepare students for scoped, independent Make projects. The unit project has a few requirements, but students are largely free to choose the design, topic, and implementation of their ideas. As you teach the unit, look for the opportunities to scaffold the skills and knowledge students will need to creatively and independently tackle the unit project.

Major Assessment and Projects

The unit project asks students to design an app that makes a recommendation based on input information frmo the user. Students are given a great deal of freedom to choose their topic, design their user interface, and decide how to actually program their app's behavior. Students submit their app, project guide, and written responses to reflection questions about how the app is designed and the development process they used to make it. Students will also complete an end-of-unit assessment aligned with CS Principles framework objectives covered in this unit.

Unit 5 - Lists, Loops, and Traversals

Unit Overview



Students learn to build apps that use and process lists of information. Like the previous unit, students learn the core concepts of lists, loops, and traversals through a series of EIPM lesson sequences. Later in the unit, students are introduced to tools that allow them to import tables of real-world data to help further power the types of apps they can make. At the conclusion of the unit, students complete a week-long project in which they must design an app around a goal of their choosing that uses one of these data sets.

Unit Philosophy and Pedagogy

Independent Creation and The Hackathon Project: Much like the project in Unit 4, the "Hackathon" project is this unit is designed as an opportunity for students to creatively and independently build something with their programming skills. While students are asked to include some technical requirements in their program to ensure they demonstrate mastery of new programming concepts, they have free rein to choose the goals, design, and implementation of their project. To avoid asking students to complete a major programming project right before the Create PT, this hackathon is the most "Create-PT-like" project of the course. It's the best chance for students to practice skills like budgeting time or scoping an open-ended project. In many classrooms, if you maintain the recommended pacing of the course, this project serves as an excellent end to the first semester.

Growing Comfort with EIPM: By Unit 5, students (and teachers!) should be developing greater comfort with the flow of EIPM lessons. Students may begin to anticipate that sequences are building towards an independent Make lesson, or look forward to stepping away from computers to Explore. A nice feature of EIPM is that you'll find strategies and modifications to each lesson type that work best for your students. Keep an eye out for how you and your students are developing comfort with EIPM, and note strategies that help meet the needs in your classroom.

Programming with Real-world Data: The Data Library is a new feature in App Lab for the 2020-21 school year and was designed to let students program with data from the real-world. The goal of this tool is to motivate students to build new kinds of data-powered apps that they find personally interesting. This tool also facilitates programming with lists of information, since students will need to manipulate lists of data in order to incorporate the different data sources. Encourage students to use datasets they find personally relevant as they draw on their creative ideas for bringing data to life.

Major Assessment and Projects

The unit project asks students to spend five days as part of a "Hackathon" project that they have nearly complete independence to scope and design. Students must choose one dataset from the Data Library in AppLab to be a component of their project to demonstrate what they have learned about lists and list processing; otherwise, scoping the project is completely up to them. Students submit their app, project guide, and written responses to reflection questions about how the app is designed and the development process they used to make it. Students will also complete an end-of-unit assessment aligned with CS Principles framework objectives covered in this unit.

Unit 6 - Algorithms

Unit Overview

Students learn to design and analyze algorithms to understand how they work and why some algorithms are considered more efficient than others. This short unit is entirely unplugged, and features hands-on activities that help students get an intuitive sense of how quickly different algorithms run and the pros and cons of different algorithms. Later in the unit, students explore concepts like undecidable problems and parallel and distributed computing.

Unit Philosophy and Pedagogy

A Break from Programming: This unit is intentionally designed as a short respite from programming. After three units and a major hackathon project, it's a great opportunity to get away from screens for a while before the final programming push that leads to the Create PT.

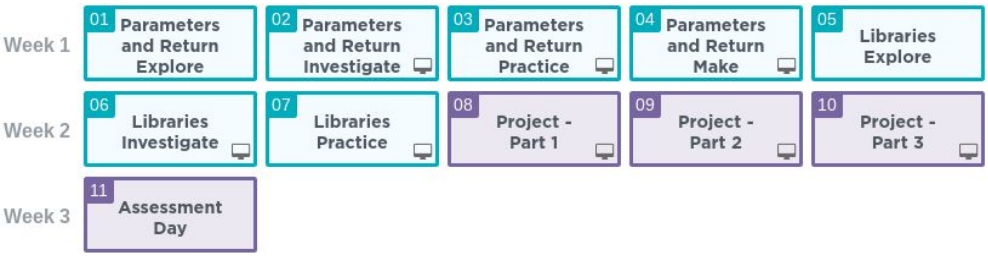
Just Enough Math: This unit includes some mathematical concepts which enrich students' understanding of how algorithms are analyzed, which might at first be a little intimidating to some students (and teachers!). The mathematical topics included in this unit are only those necessary to provide a solid foundation in algorithmic analysis to the depth described in the CS Principles framework. If you're a teacher with a strong mathematical background, check carefully that you don't needlessly add complexity to a unit which might already prove challenging for some students. All teachers should keep an eye out for the ways visuals, hands-on examples, and patterns in presentation style are used to ensure a consistent presentation of these mathematical topics.

Major Assessment and Projects

This unit does not conclude with a major project. Students will complete an end-of-unit assessment that is aligned with CS Principles framework objectives covered in this unit.

Unit 7 - Parameters, Return, and Libraries

Unit Overview



Students learn how to design clean and reusable code that can be shared with a single classmate or the entire world. In the beginning of the unit, students are introduced to the concepts of parameters and return, which allow for students to design functions that implement an algorithm. In the second half of the unit, students learn how to design libraries of functions that can be packaged up and shared with others. The unit concludes with students designing their own small library of functions that can be used by a classmate.

Unit Philosophy and Pedagogy

Learning by Building Libraries: In the second EIPM sequence of this unit, students learn to use the Student-Create Libraries tool in App Lab. This tool allows them to build and share libraries of functions that can be used in many different projects . This tool serves many purposes besides simply teaching students about libraries. By having to write functions that other students find useful, they'll need to think about common patterns or situations that they've seen in projects across the course. Students will also practice commenting their code so others can understand how it works, practice designing functions that use parameters and return.

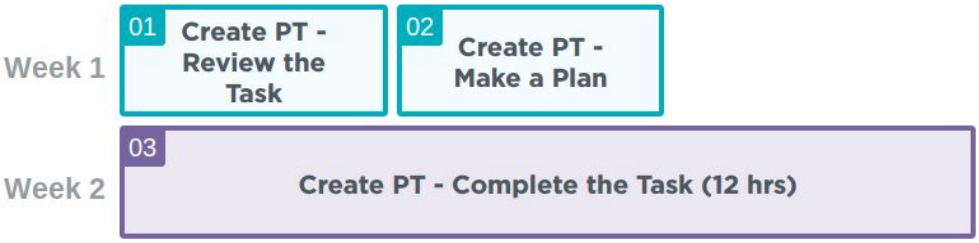
Final Preparation for the Create PT: Students learn very few new concepts in this unit; nevertheless, it can be challenging because students need to learn how to integrate the ideas of parameters and return with every other concept they've learned in this course so far. This unit presents a good opportunity to do a final review of every programming construct covered in the course as students prepare to demonstrate what they've learned on the Create PT.

Major Assessment and Projects

The unit project asks students to design a library of functions that they can share with classmates. Their library must contain at least two functions and at least one of those functions must include a parameter, return, a loop, and an if-statement. This requirement ensures students practice skills they'll use in the Create PT. Using a project guide, students choose a theme for their library, build it, test it, and exchange feedback with other students. Students submit their library code, project guide, and written responses to reflection questions about how the app is designed and the development process they used to make it. They will also complete an end-of-unit assessment aligned with CS Principles framework objectives covered in this unit.

Unit 8 - Create PT Prep

Unit Overview



In this unit, students practice and complete the Create Performance Task (PT), starting with a series of activities that ensure they understand the College Board requirements of the Create PT, which they have practiced throughout the year. Subsequently, students are given at least 12 class hours in which to complete the Create PT.

Unit Philosophy and Pedagogy

Understanding the Task: The Create PT has a number of components and requirements that can be tricky to understand. The lessons in this unit and, in particular, the Create PT Survival Guide introduced in Lesson 2 are designed to make sure students have a clear understanding of what is required of their projects. With a better understanding of what they need to do, students have more freedom to focus on what it is they want to create!

A True Performance: The Create PT is truly an opportunity for students to show off what they've learned about programming throughout the year. They have learned a great deal about how to turn their interests and creativity into working apps, and this project gives them plenty of space to bring their unique creations to life. This is the most significant programming project students will take on in the course.

Major Assessment and Projects

This unit builds towards the Create PT and has no other major assessments or projects.

Unit 9 - Data

Unit Overview

Students explore and visualize datasets from a wide variety of topics as they hunt for patterns and try to learn more about the world around them from the data. Once again, students work with datasets in App Lab, but are now asked to make use of a data visualizer tool that assists students in finding data patterns. They learn how different types of visualizations can be used to better understand the patterns contained in datasets and how to use visualizations when investigating hypotheses. At the conclusion of the unit, students learn about the impacts of data analysis on the world around them and complete a final project in which they must uncover and present a data investigation they've completed independently.

Unit Philosophy and Pedagogy

The Data Analysis Process: This unit is built around a data analysis process that helps students break down how data is turned into new information about the world. Some lessons are designed around different steps of this process, like cleaning data or building visualizations. Other lessons focus on ways this process is applied in the real world in contexts like citizen science or machine learning. The data analysis process helps provide a consistent reference point as students explore the importance of data analysis in computing.

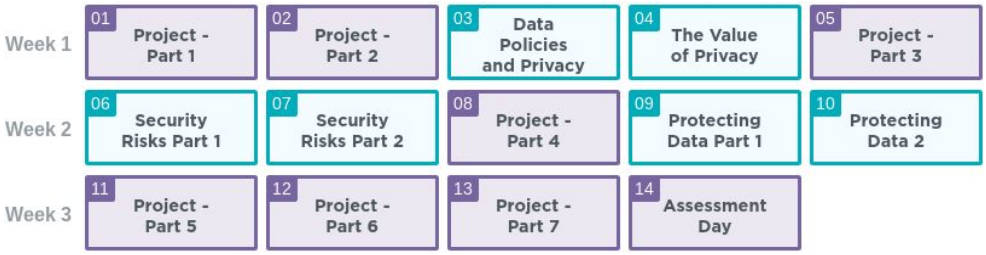
Exploring Data with the Data Visualizer: The Data Visualizer is a tool built into App Lab that allows students to quickly create visualizations of the data they've added to their projects. The set of possible visualizations is intentionally limited to a few ways to change or modify the chart. The goal of this tool is to encourage the exploration of the different kinds of questions that can be answered with data visualizations, with a greater emphasis on students' ability to quickly create a variety of visualizations.

Major Assessment and Projects

Students use the data visualizer to find and present a data story. Using what they've learned about the data analysis process, students either choose a dataset inside the data library, or upload one, of their own and create visualizations that find interesting patterns that possibly reveal new insights and knowledge. Students complete an activity guide describing their findings and the process they used in identifying them. Students will also complete an end-of-unit assessment aligned with CS Principles framework objectives covered in this unit.

Unit 10 - Cybersecurity and Global Impacts

Unit Overview



Students research and debate current events at the intersection of data, public policy, law, ethics, and societal impact in the final unit of the course. This unit is built around a simulated "future school" conference in which students must take on the persona of a stakeholder in a school setting and propose and debate technological innovations that could improve schools. Throughout the unit, students learn about the privacy and security risks of many computing innovations, and learn about the ways some of these risks can be mitigated. Students complete their Explore Curricular Requirement as part of this project as they investigate at least three computing innovations, then discuss and debate many others with their classmates. At the conclusion of the unit, the class holds a conference in which teams present their overall vision for a school of the future and the computing innovations that would power it.

Unit Philosophy and Pedagogy

Learning Through Full-class Simulation: The simulation project that runs through this unit serves a number of important goals. It helps contextualize what students are learning by moving from abstract ideas of privacy or security to concrete potential innovations. Since the simulation is based around the question of modernizing schools, students are able to consider the consequences of computing innovations in a familiar setting. By taking on an assigned role and interacting with a group of teammates who have done the same, students must consider a breadth of interests and goals beyond their own when it comes to innovating in schools.

Ending the Year as "Deciders": A major theme students engage with throughout this unit is the need to consider both sides of technological innovation. Computing technology has led to both benefits and harms to culture, economy, and society at large. Responding to important questions facing our world requires both an understanding of technology and an ability to identify and interpret the impacts it causes. This unit is not designed to advocate for any particular point of view on the impact of technology, but it should empower students to more adeptly see and weigh the consequences of the technology around them. While the Create PT may feel like the most significant project of this course, the Explore Curricular requirement and the questions faced in this unit are arguably more crucial. Many of the young people who take CS Principles may pursue studies or careers in which they are "creators" with technology, but all of them will need to be thoughtful "deciders" in a world that is profoundly shaped by computing.

Major Assessment and Projects

Students complete the "future school" simulation throughout this unit. Working in teams of roughly five people, students are assigned a role and a set of interests that they'll need to investigate. They research real-world innovations that could improve schools and align with the interests of their character. Throughout the unit, they are given opportunities to refine their proposals as a team, and debate the benefits and risks of different computing innovations. Eventually, their team submits an overall proposal for the "school of the future" and all students vote for the team and innovation they believe to be best. Students will also complete an end-of-unit assessment aligned with the CS Principles framework objectives covered in this unit.

Assessment

The course materials contain a number of assessment types and opportunities which can be used formatively (to check for understanding) or summatively (for evaluation).

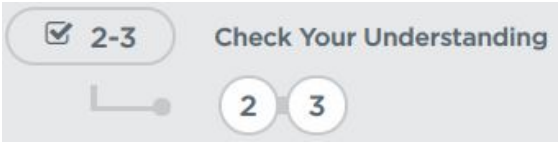
For students, the goal of the assessments is to prepare them for the AP exam and performance task. For teachers, the goal is to use assessments to help guide instruction, give feedback to students, and make choices about what to emphasize in lessons.

Code Studio includes features that assist the teacher in completing formative and summative assessments:

- Multiple choice or matching questions related to questions on the chapter summative assessment.
- Free-response text fields where students may input their answer.
- Access to student work within the App Lab programming environment and other digital tools and widgets used in the curriculum.
- The ability for students to submit final versions of App Lab projects.

Fixed Response Assessments

Lesson assessment items - You will find assessment items (multiple choice, free-response text) embedded in individual lessons, typically as the last few “bubbles” for a lesson, indicated with an assessment icon. These are intended to be used as *formative* assessment items. Students can always see them and change their responses at any time.



Unit assessments (lockable) - You will find a 15-question multiple choice assessment at the end of each Unit, with the exception of Units 8. These are intended to mimic AP-style questions. They look like their own lesson and have *lock settings* as well as the usual visibility settings. These can be used for formative or summative assessments. Our suggestion is that if you use these for summative assessment, you use them in tandem with the project based assessments for that unit to get a more complete picture of students’ skills and knowledge.

Lock settings enable or disable students from modifying their answers. For example, you may want to hide an assessment before students get to it, but after students take it, you might want it to remain visible but locked, while you review the answers.

Projects

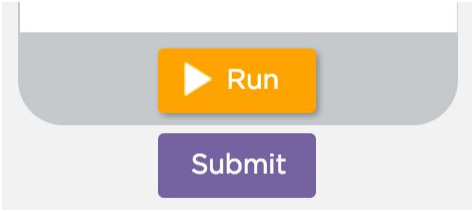
Each unit, with the exception of Units 6 and 8, contains one project that takes more than one class period. These projects may be used as summative assessments in tandem with the multiple choice assessment for the unit. Each project takes a different approach to assessing students' ability to apply what they have learned from the unit. Below you can find a summary of the project for each Unit.

Unit	Project Name	Length	Description
1 - Digital Information	Digital Information Dilemmas	2 lessons	Students research and debate dilemmas arising from the digitization of information.
2 - The Internet	Internet Dilemmas	2 lessons	Students help an imaginary politician design a political stance on a dilemma caused or impacted by the Internet.
3 - Inro to App Design	Designing an App	5 lessons	Students design an app with multiple screens that teaches their classmates about a topic of personal interest.
4 - Variables, Conditionals, and Functions	Decision Maker App	3 lessons	Students build an app that makes a decision or recommendation based on at least two pieces of user input.
5 - Lists, loops, and traversals	Hackathon	5 lessons	Students build an app with any purpose they wish that uses a dataset from App Lab's data library.
6 - Algorithms	--	--	No project
7 - Parameters, Return, and Libraries	Make a Library	3 lessons	Students design a library of functions to share with their classmates.
8 - Create PT Prep	--	--	Students complete Create PT.
9 - Data	Tell a Data Story	2 lessons	Students choose and analyze a dataset in order to find meaningful patterns and present their findings.
10 - Cybersecurity and Global Impact	Innovation Simulation	7 lessons	Students research and debate computing innovations as they work with a team to make a proposal for a "school of the future".

Unit 8 Note: No project exists for this unit because the entire unit is focused on preparing for and providing time to do the Create Performance Task. No new content knowledge is taught during this unit.

Submittable Programming Projects

When the curriculum calls for a programming project for assessment the App Lab environment will show a “submit” button below the typical “Run” button. When a student submits a project it is submitted with a timestamp and locked for teacher review. It also shows up in a special area of the teacher dashboard. The teacher can release the project back to the student as well.



Worksheets and Activity Guides

Worksheets and activity guides are good opportunities for assessment. Worksheets or activity guides often ask students to write, answer questions, and respond to prompts. When available, answer keys for worksheets and activities are provided through the “teacher only” panel on Code Studio.

It is up to the classroom teacher:

- To determine the appropriateness of the assessments for their classrooms.
- To decide how to use, or not to use, the assessments for grading purposes. The curriculum and online dashboards do not provide teachers with a gradebook, and we do not provide recommendations for how to assign grades based on performance on an assessment.

Tools

Many of the learning tools used in the CS Principles curriculum have been developed in-house at Code.org. The three major categories of tools are **widgets, the Internet Simulator,** and **App Lab.**

Widgets

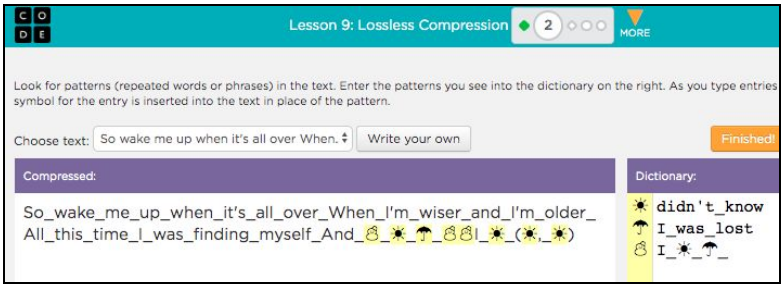
Widgets are small digital tools that act as a playground for exploring and experimenting with a CS concept. Widgets are meant to promote discovery and creativity within a fairly narrow scope where there are typically no right or wrong answers. These tools will enforce rules and automate processes, letting you experiment with different approaches to problems more quickly.

Below, we briefly describe each widget used in the course, the concepts they cover, and connections to other parts of the curriculum. For a link to stand-alone versions of all widgets head to code.org/widgets.

Text Compression Widget

Where in the curriculum: Unit 1 Lesson 9

Description



This widget lets students interactively experiment with compressing a piece of text by identifying patterns in text, storing those patterns in a "dictionary," and replacing the repeated pattern with a 1-byte symbol to create a compressed version of the text. The widget updates with every keystroke and also performs the compression calculations, so you can see if you're increasing or decreasing the total file size in real time. Since figuring out the optimal amount of compression is a computationally hard problem (i.e. there is no known algorithm to find or verify that the optimal compression has been found), students can experiment with many different approaches rather than focusing on finding the 'right' answer. The most intriguing idea to play with is figuring out how to best harness the power of representing patterns of patterns. Done effectively, it can dramatically improve compression; done incorrectly, the file will end up even bigger than when you started!

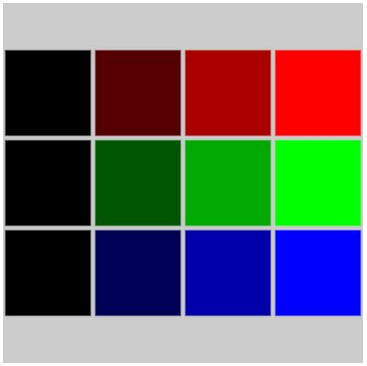
Concepts

The primary concept here is related to lossless compression, but the overarching concept is about abstraction in the representation of information. Compressing text this way can be viewed as a logical extension of text representation in binary, which is explored throughout Unit 1. The challenge is to think about how one can precisely represent the exact same information with fewer bits. The technique used in this widget - maintaining a dictionary of repeated patterns of text - is used in ZIP compression, and is also more or less the same technique used to compress images in GIF format.

Curriculum connections

This widget is a very fertile example to refer back to later in the course in two main areas The kinds of analysis and problem solving students do in finding repeated patterns and then expressing those patterns as a single reusable abstraction is very similar to the kinds of thinking students do throughout the course and particularly in programming units.

Pixelation Widget



Where in the curriculum: Unit 1 Lesson 7 - 8

Description

This widget lets students compose an image “in binary” by filling in binary information and the widget renders the image that the binary represents. It’s like having an instant binary interpreter that obeys the rules of the agreed upon image format. The widget has a few variants of increasing sophistication that are used over a few lessons. It starts with a very simple black-and-white image format, and ends with up to 24 bits of color information for each pixel.

Concepts

What students will grapple with the most is understanding the RGB color system and the tradeoffs between the number of bits needed to represent an image (file size) and how precise the color information is. A practical takeaway should be an understanding of why an RGB color is broken into red, green, and blue values that are each represented with a number between 0-255, and the factors that influence image file sizes and how they get so large when uncompressed.

Curriculum connections

You can refer back to this lesson when RGB colors come up in the programming unit. You can also return to this activity if the idea of image compression comes up in the next lesson about lossy compression and file formats.

Frequency Analysis Widget

Where in the curriculum: Unit 10 Lesson 9

Description

This widget lets you play with two classic substitution ciphers, one known as the Caesar Shift (encryption by shifting each letter of alphabet the same amount) and random substitution (encryption via a 1:1 substitution of one letter for another, but randomly assigned as opposed to a uniform shift).

Concepts

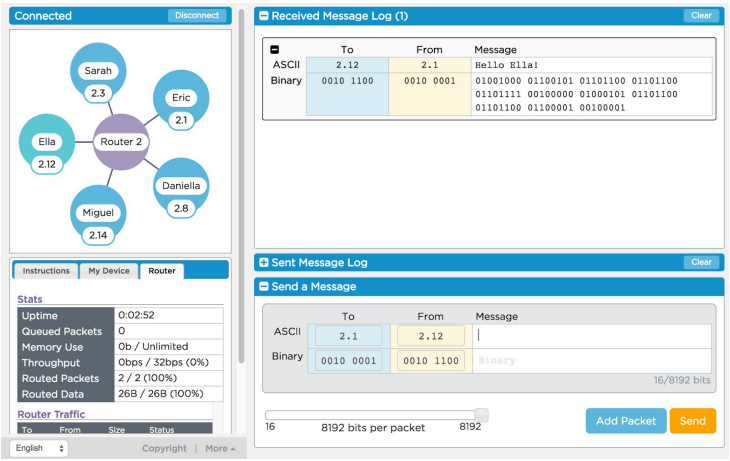
More than anything this tool is meant to expose how simple it is to crack a substitution cipher when armed with a tool that does a very simple frequency analysis. Students will also get a feel for the kinds of techniques that have been used historically to encrypt secret messages, which is a foundation for later discussions about current encryption techniques. The widget is also a concrete thing you can use to point out key terms that come up in encryption and security contexts: encryption, decryption, symmetric encryption, key, etc.

Curriculum connections

The distance between a protocol for encoding information (including image encoding, text compression etc.) and encryption is pretty short. You might refer back to the text compression widget in particular to ask whether or not text compression is a form of encryption.

Internet Simulator

Where in the curriculum: Unit 2



Description

Similar to a widget, but much larger in scope, the Internet Simulator is designed to let students visualize, experiment with, and solve different kinds of problems associated with networked computers in a hands-on way. Often these problems involve inventing a communication protocol, or inventing ways to encode information that makes transporting it over the Internet feasible.

It is essential to note that we use the Internet Simulator for much more than teaching Internet protocols. The Internet Simulator contextualizes exploration of deeper concepts in computer science, like the use of abstraction-to-solve problems and the binary representation of information. The goal of the Internet Simulator is not merely to present the functionality of the different layers of the internet, but to provide an opportunity for students to reason about why those structures exist and even develop their own solutions to the problems solved by the systems of the internet.

The simulator is configured differently in each lesson to enforce different rules or to expose different behaviors of the internet that students must creatively problem solve around. Specifically, each version of the Internet Simulator is configured to mirror a high level version of the layered Internet Protocol stack. With each lesson the Internet Simulator changes to incorporate the solution to the previous problem students solved. In this way we work from the bottom up, first solving physical coordination problems with sending bits back and forth, then addressing (IP), then packeting (TCP), then name-to-address mapping (DNS), and finally HTTP.

Internet Simulator Configurations

The Internet Simulator incrementally gains new features through Unit 2. In general the solution students come up with in one lesson will be similar to the new features added in the next lesson. The table below summarizes this process.

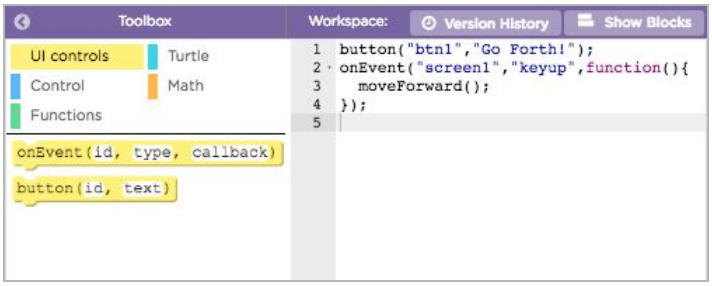
Lesson	Configuration	Problem
Lesson 1 Welcome to the Internet	**Sending ASCII on a Shared Wire (Binary/Decimal/ASCII text)** Simulates two computers connected by a wire which shows how two computers that can send and receive streams of text as represented by bits (0s and 1s). As messages are sent and received between the two computers, you can also view the decimal versions of the binary numbers representations of the ASCII text.	Students create rules for when to set and read the shared wire in order to communicate. Students explore the relationships between between binary, decimal, and ASCII representations.
Lesson 3 The Need for Addressing	**Broadcasting Messages ("Broadcast mode")** This is the first "networked" configuration. Multiple people (up to 6) join a small network, and every message sent is "broadcast" to all the others simultaneously.	Students create rules to ensure that messages get to and from their intended recipients. They must invent an addressing protocol similar to IP.
Lesson 4 Routers and Redundancy	**Routers and Addresses ("Router Mode")** The entire class joins one network. Each person is assigned an 8-bit "IP address." When you start, you "join" a router, which also has an address. You must enter the proper "IP address" for a message to get the intended recipient and each message "hops" across multiple routers to get to its final destination. A router log records each individual hop messages make through the network.	Students explore how traffic is routed through a network and some of the privacy / security concerns that arise as a result. They also learn how redundant paths between routers support the growth of the internet.
Lesson 5 Packets	**Packets and Reliability ("Router Mode with dropped packets")** Long messages are split into packets. Packet size is limited to force the use of multiple packets. Traffic is routed, but roughly 10-20% of packets get "dropped" or lost on the way to their destination.	Students create rules that can reliably get all the pieces of information to the intended destination. Their system will likely include numbering packets and rules for requesting and re-sending missing packets, similar to TCP.
Lesson 6 HTTP and DNS	**Automatic DNS ("DNS mode")** Users' IP addresses are hidden, but each router has a DNS server whose IP address is known. By sending a GET request to the the DNS server for example: "GET janesmith1" you can discover classmates' IP addresses.	In this configuration This mode is primarily used as an investigation into the rudiments of DNS more than solving a problem. In the lesson the problem is solved as an unplugged activity.

App Lab

Where in the curriculum: Units 3, 4, 5, 7, 8 and 9

Description

App Lab is an Integrated Development Environment (IDE) for building web applications in JavaScript. The tool is designed for new learners with the general idea that a student should be able to take an idea in their head and rapidly create a functional prototype in App Lab. A powerful feature of App Lab is that with it you can program either by drag-and-drop blocks or by typing text, and you can easily toggle back and forth between them. This allows for the best of both worlds: composing a piece of code with block structures, but getting into the text to tweak it.



App Lab is also chock full of supports for learners and users including:

- Integrated documentation with fully working examples
- Interactive debugging console
- A full debugger with breakpoints and line stepping
- A “design mode” that lets you compose an app screen with drag-and-drop HTML/CSS
- Integrated tools to import, explore, and access tables of data

Like any full-fledged programming environment, taking all of App Lab in at the beginning can be intimidating. Thus, when used in the curriculum, the programming “toolbox” is scoped to the lesson or problem at hand. This dramatically reduces cognitive load and allows the student to focus on solving the problem within the constraints of the environment. As the course proceeds, more and more of the commands and features of App Lab are exposed.

Planning for the Year

This section provides advice on how best to use the curriculum in your own classroom. The activities in Appendix A are also useful in formalizing your plan for teaching the course.

Pacing

Time will always be tight in an AP CS Principles course. The early date of the AP test, the 12 class-hours dedicated to the Create Task administration, and the wide breadth of topics covered all contribute to this fact. The curriculum is designed to help you successfully teach the course in a standard school year, but careful planning and attention to schedule are important to ensure you stay on track.

Pacing Consideratons

The following are important pieces of information to know as you plan the pacing of your course.

Lessons are Designed for 45 Minute Class Sessions

The lesson plans and unit calendars assume a class that meets for 45 minutes, five days a week, for the duration of the school year. For teachers on a block or other non-traditional schedule, you will need to plan on combining or modifying lessons to fit into your school schedule. Some teachers chose to do two lessons a day on the block schedule or combine “Wrap-up” and “Warm-up” activities to make lessons fit their time. Using a calendar to plan out your year will help you adjust your schedule if lessons go longer or shorter than expected.

Plan on Finishing Units 1 through 5 by the end of Semester 1

The course is structured so that students have enough time to learn the needed programming concepts and do the Create Task well before the AP deadline of April 30th. To stay on track, plan on finishing Units 1 - 5 by the end of the first semester, or the “halfway point” of your course if you are not on a semester system. There are 67 45-minute lessons in Units 1 - 5. This includes time for assessments.

Learning Objectives are Addressed in the Main Activity and Synthesized in the Wrap-up

The “Activity” portion of the lesson is typically core curriculum and the portion of the lesson that addresses the learning objectives found at the top of each lesson plan. You should aim not to significantly cut these portions of the lessons. Depending on your students’ needs, however, you likely can alter or cut warm ups and share outs while still hitting these objectives. The Wrap-up of many lessons contain key vocabulary and takeaways that help students process their learning, and while they may be shortened, skipping the Wrap-up is not advised. In all cases the learning objectives, lesson purpose, and teaching tips are designed to help you make these decisions. The teacher forums are also useful for understanding how other teachers are approaching each lesson.

Pacing Guides Assume No Homework

The curriculum does not assume that you can assign homework. This is done since many students do not have access to a computer or the internet at home which are requisite tools for completing the course. If you are certain your students do have access to technology outside of class you may optionally choose to assign some portions of lessons as homework.

Pacing Tips

Use the tips below to help you make adjustments to the curriculum in response to your classroom's pacing needs.

Minimize Frontloading, Get to the Activity

Warm ups are intended to be quick (usually ~5 minutes) for motivating discussions and often make no assumptions about students learning the content of the day. Avoid front-loading lectures to begin lessons, get to the main activity as quickly as possible, and save the synthesizing discussion for after.

Prioritize the Classroom Environment

Pacing considerations are obviously important, but "covering" every lesson in the curriculum is not the only goal. Collaborative and inquiry-based activities, especially those early in the curriculum, are designed to create a welcoming classroom environment that promotes CS Principles' top level goal of broadening participation in computing. When making cuts, aim to preserve creative, collaborative, and exploratory activities wherever possible.

End Activities Early If Students Understand the Concept

Activities, especially unplugged activities, usually have students solve a problem that highlights a concept. Often students do not need to solve the problem fully or complete every part of an activity to understand the learning objectives of the lesson. Carefully observing students during activities will help you determine if and when you can end an activity early and move on to synthesizing discussions.

Combine Wrap Ups and Warm Ups

Every lesson in CSP is written as though it needs to operate as a standalone entity that contains both a core activity as well as engaging warm up and wrap up activities, extensions for learning more, and so on. When teaching lessons in sequence, especially on a block schedule, it is often possible to combine the wrap up of one lesson with the warm up of the lesson following. This is one general purpose way you can save time during the school year.

Ask for Support on the Forum

Teaching a course for the first time can be intimidating to do alone. Luckily you have a large community of support at forum.code.org. When making pacing decisions don't hesitate to reach out on the forum where you'll be able to get advice from Code.org staff and other CSP teachers.

Planning for the AP Exam and Performance Task

The CS Principles AP assessment includes both a multiple choice exam and the Create Performance Task, a substantial independent programming project. Supports are integrated throughout the curriculum to help you prepare students for these assessments and plan the time necessary for students to complete the performance task. For detailed information about AP assessments go to https://apcentral.collegeboard.org/courses/ap-computer-science-principles/exam

Preparing for the Multiple Choice Exam

At the end of the year students will complete a 70-question multiple choice exam. Students should practice multiple choice problems in advance of the exam.

- **Unit Assessments:** Multiple choice assessments found at the end of each unit are written in a style that closely resembles the format of questions that students will find on the multiple choice exam. These assessments are excellent practice for the questions students will see at the end of the year.

- **College Board Practice Exams:** The College Board has released a full-length practice exam sometime in the spring. It is highly recommended that students complete this practice exam for the most authoritative and up-to-date representation of the types of questions students will see on the AP exam itself.

- **College Board Topic Questions:** The College Board has created a bank of questions mapped to different topics in the AP CS Principles framework. More information on these questions is on the following page.

- **Practice exams in the public domain:** AP CS Principles is relatively new, but there is an ever-growing set of resources available to help students prepare for the exam. We encourage you to seek out and share whatever you can find with CS Principles community and Code.org forum.

Preparing for the Create Performance Task

Students must complete the Create Performance Tasks during the school year and submit it at the end of April. The table below summarizes the requirements and supports for each task.

- **Unit Projects:** By far the most significant preparation for the Create PT are the unit projects in units 3, 4, 5, and 7 which include significant components of the task. Through completing these projects students will not only see the specific programming skills and prompts that appear on the Create PT, but they will also practice independently completing a major programming project.

- **Make Lessons:** EIPM sequences found in units 4, 5, and 7 conclude with projects that give students targeted practice with independently programming an app. These one-day activities give students a chance to practice independently using programming skills in between unit projects

- **The Create PT Prep Unit**: Unit 8 - Create PT Prep is designed to make sure students understand every aspect of the task. This unit includes sample submissions, activities, checklists, and a "Create PT Survival Guide" that walks students through every aspect of the task.

- **College Board Formative Create PT Prep:** The College Board has prepared a set of formative questions modeled after the Create PT which students can practice throughout the year. The curriculum includes indications of when it is best to assign these practice questions.

The AP Course Audit

What it is: In order to have an official "AP" course listed on your students' transcripts, the curriculum that you intend to teach must be "audited" by the College Board. The process is intended to ensure that (a) a teacher and school administration have confidence that the course meets the necessary guidelines and requirements for AP and (b) colleges and universities have confidence that AP courses that appear on students' transcripts meet the AP criteria across all high schools.

How it works: If you intend to teach CS Principles using Code.org's curriculum, the audit process is relatively brief and painless. Code.org's curriculum and syllabus have been pre-approved and endorsed by the College Board as meeting all of the necessary standards and criteria. If you use your own syllabus you will need to provide and submit this evidence yourself. If you are completely new to this process here are the broad strokes of what you need to do:

1. Create a teacher account on the College Board website
2. Log in to your teacher account, "Add a New Course" and choose AP CS Principles
3. Fill out the Audit form
4. "Adopt Sample Syllabus" and choose the Code.org Syllabus
5. Your school administrator will verify the submission

Detailed instructions for completing the audit can be found on Code.org's CS Principles home page (code.org/csp). Look for the "AP Endorsed" insignia on the page.

After that, since the Code.org syllabus is pre-approved, you should be done and ready to go. You and your school administrator will be notified that you have completed the audit process.

Using the College Board's Topic Questions

The College Board has provided a bank of questions in AP Classroom to help formatively assess student understanding of the content in the framework. These questions are mapped to topics with each topic having a handful of questions available. You can find a mapping of topic questions to each unit in Code.org's Computer Science Principles course in Appendix B of this document.

The College Board has a few strict guidelines about how topic questions can be used. In particular, students may not receive a grade based on performance on topic questions nor can they be used for teacher evaluation. Beyond these requirements, however, they are primarily intended to formatively assess student progress and learning as they prepare for the end of course exam.

Within our own course we recommend that you use them in a variety of ways:

- Throughout the unit assign topic questions to students related to the topics students are learning about that day or that week
- Prior to the unit assessment assign topic questions to help students practice and prepare for the summative assessment
- After the unit assessment use these topic questions to help students track their progress towards preparation for the AP assessment

Tech Requirements and Required Materials

Technical Requirements
The course requires and assumes a 1:1 computer lab or setup such that each student in the class has access to an internet-connected computer every day in class. Each computer must have a modern web browser installed. All of the course tools and resources (lesson plans, teacher dashboard, videos, student tools, programming environment, etc.) are online and accessible through a modern web browser. For more details on the technical requirements, please visit: code.org/educate/it

While the course features many "unplugged" activities designed to be completed away from the computer, daily access to a computer is essential for every student. It is not required that students have access to internet-connected computers at home to teach this course, but because almost all of the materials are online, it is certainly an advantage. PDFs of handouts, worksheets and readings are available on the course website.

Required Materials / Supplies
One potentially significant cost to consider is printing. Many lessons have handouts that are designed to guide students through activities. While it is not required that all of these handouts be printed, many were designed with print in mind and we highly recommend their use.

Beyond printing, some lessons call for typical classroom supplies and manipulatives such as a student journal, poster paper, markers, sticky notes, and graph paper. The following lessons require materials you may not typically have in a classroom environment

Lesson	Materials	Alternatives
Unit 1 Lesson 2	Assortment of craft materials for constructing physical devices. Recommendations: cups, string/yarn, construction paper, flashlights, slinkies, noise makers, markers, and glue, etc.	none
Unit 2 Lesson 2	String for table groups to build a network connecting them	Students draw their network but don't actually build it
Unit 3 Lesson 5	A handful of LEGO® blocks for every pair students	Sticky notes, construction paper
Unit 4 Lessons 1 & 5	Plastic bags, sticky notes, dry erase markers	Envelopes
Unit 5 Lesson 1	Plastic bags, gallon-sized plastic bags, sticky notes, dry erase markers, tape	Envelopes
Unit 6 Lessons 2 & 3	Sticky notes	Scraps of paper
Unit 6 Lesson 4	Decks of cards	Any item that could be combined into two categories (e.g. change with even / odd year)
Unit 7 Lessons 1 & 5	Sticky notes, envelopes, plastic bags, file folders	Scraps of paper, folders made of a folded sheet of paper, etc.

Appendix A: Planning Handouts

Assessment Approach Organizer

Part A: What do you personally value (or want to value) in your classroom?

Write down your thoughts here:

Part B: What would an assessment approach look like to match those values?

What types of summative assessments are used?

- Projects? Multiple choice tests? Free response questions?
- What would the role of collaboration be in assessments?

What is the role formative assessment?

What is the role of “re-dos,” “re-learning”, or “retakes”?

Part C: Creating your assessment approach for Units 1-3

	Unit 1 - Digital Information	Unit 2- The Internet	Unit 3 - Intro to App Design
1. What will you use as unit summative assessments for this unit? Will you create new or modify existing assessments or rubrics?			
2. If you are planning to have multiple components to your summative assessment, how will you weigh them?			
3. For the summative assessments you outlined above, what supports will students need to be successful? How will students prepare for the assessments?			
Pause to Regroup			
How will you use formative assessment so students can understand where they are in the learning process?			
If you plan to offer “re-learning” or “re-dos”, what will those look like?			

Planning Your Year

Part 1: Plans for further exploration. Use the space below to reflect on the week. You likely saw or heard things from other participants or facilitators that you want to explore further. Use this space to document that now while it is top of mind.

Things to explore further		
What *topics in the curriculum or course do you want to further explore before you start teaching the course?*	**How** *can you explore these topics?*	***When*** *do you plan to do this exploration?*

Part 2: Modifying lessons to fit your schedule. Use the table below below to document your plan for making the 45 minute lessons fit into the length of your class period.

How are your class periods structured?	
How do you plan to modify lessons to fit your class period structure?	

Part 3: Planning Instructional Units Use the table below below to document your pacing plan for the year. As reminder, if you are teaching the course as an AP class, you need to keep in mind the following dates:

- The date the Create Performance Task is due: _____
- The date of the Multiple Choice Exam: _____

What	*Duration*	When do you plan to start?	When do you plan to finish?	Notes or special considerations
Unit 1	*14 lessons*			
Unit 2	*9 lessons*			
Unit 3	*11 lessons*			
Unit 4	*15 lessons*			
Unit 5	*18 lessons*			
END OF FIRST SEMESTER				
Unit 6	*6 lessons*			
Unit 7	*11 lessons*			
Unit 8	*18 lessons*			
Unit 9	*9 lessons*			
Unit 10	*14 lessons*			
Multiple Choice Prep	_____ *amount of time*			

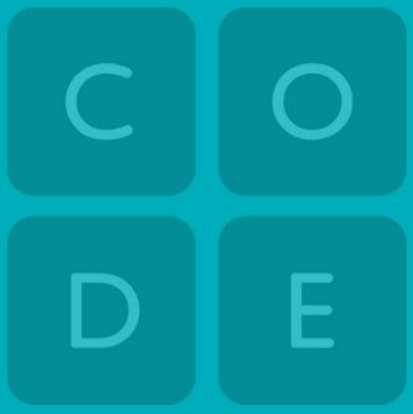
Appendix B: Code.org's AP Topic Question Coverage

Mapping Topics to Units in Code.org's CS Principles Curriculum

The list below indicates when we believe it would be best to use topic questions with your students. In a few instances topics are divided across multiple units in a course. In those instances the topic has been assigned to the unit in which it most makes sense for students to use the materials provided by the College Board, with indications of where you may wish to return to those materials elsewhere in the course.

Unit 1: Digital Information	**Topic 2.1** Binary Numbers **Topic 2.2** Data Compression **Topic 5.5** Legal and Ethical Concerns ● ***Note**: A few of the EKs in this topic are not covered until Unit 2 and 10 of the course. Since the majority of the EKs relate to Intellectual Property, we recommend you evaluate student learning of this topic in Unit 1 and return to this topic later in the course.*
Unit 2: The Internet	**Topic 4.1** The Internet **Topic 4.2** Fault Tolerance **Topic 5.2** Digital Divide.
Unit 3: Intro to App Design	**Topic 1.1** Collaboration ● **Note**: Students will continue to use collaborative practices throughout the course but many explicit skills and ideas will have been introduced in this unit. We recommend you use Topic-based resources in this unit and return to them throughout the course. **Topic 1.2** Program Function and Purpose ● **Note**: While the core ideas of this topic are covered in Unit 3, students continue to develop an understanding of these ideas throughout the programming units, the Create PT, and even into Unit 10. We recommend you initially cover these topics here and depending on student performance return to them throughout following programming units. **Topic 1.3** Program Design and Development ● **Note**: This topic is almost entirely covered in Unit 3 but students return to it throughout programming units.
Unit 4: Variables, Conditionals, and Functions	**Topic 1.4** Identifying and Correcting Errors ● **Note**: Students learn debugging practices in Unit 3 and continue to practice them throughout programming units. We recommend you initially use topic resources here and return to them later if you deem it necessary. **Topic 3.1** Variables and Assignment **Topic 3.3** Mathematical Expression **Topic 3.5** Boolean Expressions **Topic 3.6** Conditionals **Topic 3.7** Nested Conditionals

	Topic 3.15 Random Values • **Note**: While students are introduced to random values in Unit 3, we recommend you wait to use resources for this topic until Unit 4 when students have more experience programming expressions with random values
Unit 5: Lists, Loops, and Traversals	**Topic 3.2** Data Abstraction **Topic 3.4** Strings **Topic 3.8** Iteration **Topic 3.10** Lists **Topic 3.16** Simulations
Unit 6: Algorithms	**Topic 3.9** Developing Algorithms • **Note**: Some concepts will have been covered in previous units but we believe this to be the best moment to use these topic resources. **Topic 3.11** Binary Search **Topic 3.17** Algorithmic Efficiency **Topic 3.18** Undecidable Problems **Topic 4.3** Parallel and Distributed Computing
Unit 7: Parameters, Return, and Libraries	**Topic 3.12** Calling Procedures **Topic 3.13** Developing Procedures **Topic 3.14** Libraries
Unit 9: Data	**Topic 2.3** Extracting Information from Data **Topic 2.4** Using Programs with Data **Topic 5.3** Computing Bias **Topic 5.4** Crowdsourcing
Unit 10: Cybersecurity and Global Impacts	**Topic 5.1** Beneficial and Harmful Effects **Topic 5.6** Safe Computing



To view all lesson plans and the online activities associated with this course, please visit:

http://studio.code.org

It is thanks to these and other generous donors that we are able to develop and can offer this course at no cost to schools, teachers, or students:

Amazon, Facebook, Infosys Foundation USA, Microsoft, Bill and Melinda Gates Foundation, Google, Omidyar Network, PricewaterhouseCoopers, Vista Equity Partners, BlackRock, Chan Zuckerberg Initiative, Quadrivium, Salesforce, Verizon