Curriculum Guide
2021-2022

Computer Science Discoveries

Welcome to Computer Science Discoveries 2

Code.org Values and Philosophy 4
- Curriculum Values 4
- Pedagogical Approach To Our Values 5

CS Discoveries Curriculum Overview 6
- Unit 1 - Problem Solving and Computing 6
- Unit 2 - Web Development 9
- Unit 3 - Interactive Animations and Games 13
- Unit 4 - The Design Process 17
- Unit 5 - Data and Society 20
- Unit 6 - Physical Computing 23
- Optional Unit - AI and Machine Learning 26

Learning Tools 30

Unplugged and Plugged Activities 32

Teaching and Learning Strategies 33
- Pair Programming 33
- Think-Pair-Share 34
- Peer Feedback 35
- Debugging 36
- Journaling 37

The Problem Solving Process 39
- The Problem Solving Process with Programming 40
- The Problem Solving Process with Design 41
- The Problem Solving Process with Data 42

Assessment and Feedback 43

Planning for the Year 45

Appendix A: Professional Learning Handouts 46
- Build your Action Plan: Make a commitment to support equity in CS education 47
- Build your Action Plan: Plan for the fall 48
- Pacing and Planning: Instructional units 49

Appendix B: CS Discoveries Getting Started and Implementation Options 50
- Getting Started 51
- Intended Implementation Options 53

Appendix C: Additional Guides and Resources 55
- Guide to Debugging - Student Facing 56
- Guide to Debugging - Teacher Facing 57
- Guide to Differentiation - Teacher Facing 60
- Guide to Resources - Student Facing 62
- Guide to Resources - Teacher Facing 63

Welcome to Computer Science Discoveries

CS Discoveries is an introductory computer science course that empowers students to create authentic artifacts and engage with computer science as a medium for creativity, communication, problem solving, and fun.

Semester 1 - Exploration and Expression

Unit 1: Problem Solving and Computing	Students learn the problem-solving process, the input-output-store-process model of a computer, and how computers help humans solve problems. Students end the unit by proposing their own app to solve a problem.
Unit 2: Web Development	Students learn to create websites using HTML and CSS inside Code.org's Web Lab environment. Throughout the unit, students consider questions of privacy and ownership on the internet as they develop their own personal websites.
Unit 3: Interactive Animations and Games	Students learn fundamental programming constructs and practices in the JavaScript programming language while developing animations and games in Code.org's Game Lab environment. Students end the unit by designing their own animations and games.

Semester 2 - Innovation and Impact

Unit 4: The Design Process	Students apply the problem solving process to the problems of others, learning to empathize with the needs of a user and design solutions to address those needs. During the second half of the unit, students form teams to prototype an app of their own design, first on paper and eventually in Code.org's App Lab environment.
Unit 5: Data and Society	Students explore different systems used to represent information in a computer and the challenges and tradeoffs posed by using them. In the second half of the unit, students learn how collections of data are used to solve problems and how computers help to automate the steps of this process.
Unit 6: Physical Computing	Students use Code.org's App Lab environment, in conjunction with the Adafruit Circuit Playground, to explore the relationship between hardware and software. Throughout the unit, students develop prototypes that mirror existing innovative computing platforms, before ultimately designing and prototyping one of their own.

Optional Units

AI and Machine Learning	Students learn how machine learning can be used to solve problems by preparing data, training a machine learning model, then testing and evaluating the model for accuracy and bias. Students use Code.org's AI Lab environment to train machine learning models, then import their models into App Lab to create apps that solve problems

Standards and Learning Framework

CS Discoveries was written using both the K-12 Framework for Computer Science and the CSTA standards as guidance. Lists of connected standards can be found within each lesson where they are addressed. An overview of all of the standards for the course can be viewed at https://studio.code.org/courses/csd-2021/standards

In addition to the CSTA standards and K-12 Framework, CS Discoveries was written with a learning framework, which outlines the expected student outcomes that are assessed throughout the unit and in that unit's major projects and post-project test. These outcomes are organized into concept clusters to help students and teachers understand the broad goals of each unit.

Tools Across the Course

CS Discoveries introduces students to tools and programming languages that are accessible for beginners while offering more advanced students opportunities to create sophisticated projects. All of the tools below are integrated directly into the Code.org website, allowing teachers to have visibility into all student work and progress.

Unit 1 *Problem Solving and Computing*	*No special learning tools*
Unit 2 *Web Development*	**Web Lab** — A browser-based tool for creating and publishing HTML and CSS web sites.
Unit 3 *Interactive Animations and Games*	**Game Lab** — A browser-based JavaScript programming environment designed to create sprite-based drawings, animations and games. Enables students to switch between programming in blocks or text.
Unit 4 *The Design Process*	**App Lab** — A browser-based JavaScript programming environment for creating interactive apps. Enables students to switch between programming in blocks or text.
Unit 5 *Data and Society*	*No special learning tools*
Unit 6 *Physical Computing*	**Circuit Playground** — Adafruit's low-cost microcontroller featuring multiple integrated sensors and output devices. **Maker App** — A downloadable program that connects App Lab to the Circuit Playground, allowing students to easily program the Circuit Playground directly from App Lab.
AI and Machine Learning	**AI Lab** — A browser-based tool for creating machine learning models from tabular data

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 support than others, and so those with the greatest needs should be prioritized. All students can succeed in computer science when given the right support 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 to shape meaning from what the students have experienced. We also expect teachers to act as the curator of materials. Finally, we include an abundance of materials and teaching strategies in our curricula - 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. They focus on foundational concepts that allow them to stand the test of time, and they are designed to support exploration and discovery by those without computer science knowledge. This allows students to develop an understanding of these concepts through "play" and experimentation. From our coding environments to our 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 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

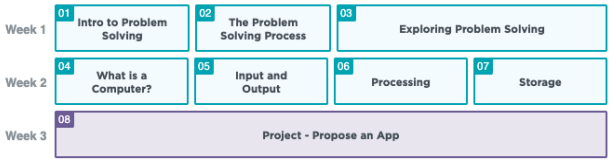
Our lessons almost always call for students to interact with other students in the class in some way. Whether learners are simply conferring with a partner during a warm up discussion, or engaging in a long-term group project, our belief is that a classroom where students are communicating, solving problems, 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.

CS Discoveries Curriculum Overview

The following pages provide an overview of each of the 6 units in the CS Discoveries curriculum. For each unit, there is a one page description of the unit, timeline, big questions answered throughout the unit, learning goals, major projects, and tools, and the 1 - 3 pages that follow the overview outline each lesson of the unit.

Unit 1 - Problem Solving and Computing

Overview and Timeline



The Problem Solving and Computing unit is a highly interactive and collaborative introduction to the field of computer science, as framed within the broader pursuit of solving problems. Through a series of puzzles, challenges, and real world scenarios, students are introduced to a problem solving process that they will return to repeatedly throughout the course. Students then learn how computers input, output, store, and process information to help humans solve problems within the context of apps. The unit concludes with students designing an app that helps solve a problem of their choosing.

Big Questions

Chapter 1 - The Problem Solving Process

- What strategies and processes can I use to become a more effective problem solver?

Chapter 2 - Computers and Problem Solving

- How do computers help people to solve problems?
- How do people and computers approach problems differently?
- What does a computer need from people in order to solve problems effectively?

Unit Goals

By the end of the unit, students should be able to identify the defined characteristics of a computer and how it is used to solve information problems. They should be able to use a structured problem solving process to address problems and design solutions that use computing technology. The unit also serves to build a collaborative classroom environment where students view computer science as relevant, fun, and empowering.

Alternate Lessons

Alternate versions of Lessons 1 and 3 are available, and they target the same learning goals. Teachers may choose which lesson to implement based on the availability of supplies or student interests. Chapter 1 of Unit 1 can be used as a review for students who have already taken a portion of the Computer Science Discoveries course in a different grade level or semester. In this case, the teacher can substitute in a different lesson from the one that students have already completed.

Major Projects

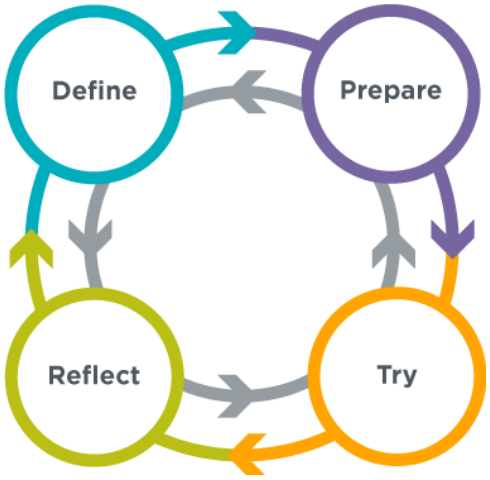
- **Lesson 8: Project - Propose an App**

To conclude their study of the problem solving process and the input/output/store/process model of a computer, students will propose an app designed to solve a problem of their choosing. To learn more about this project, check out the description in this unit's lesson progression.

Lesson Progression: Unit 1 - Problem Solving and Computing

Chapter 1 The Problem Solving Process	**Lesson 1 Alternate Lessons (Select One)**	**Lesson 1: Intro to Problem Solving - Aluminum Boats** The class works in groups to design aluminum foil boats that will support as many pennies as possible. At the end of the lesson, groups reflect on their experiences with the activity and make connections to the types of problem solving they will be doing for the rest of the course.
		Lesson 1: Exploring Problem Solving - Newspaper Table The class works in groups to design newspaper tables that will support as many books as possible. At the end of the lesson, groups reflect on their experiences with the activity and make connections to the types of problem solving they will be doing for the rest of the course.
		Lesson 1: Exploring Problem Solving - Spaghetti Bridge The class works in groups to design spaghetti bridges that will support as many books as possible. At the end of the lesson, groups reflect on their experiences with the activity and make connections to the types of problem solving they will be doing for the rest of the course.
	Lesson 2: The Problem Solving Process This lesson introduces the formal problem solving process that the class will use over the course of the year: Define - Prepare - Try - Reflect. The class relates these steps to the problem from the previous lesson, then to a problem they are good at solving, then to a problem they want to improve at solving. At the end of the lesson, the class collects a list of generally useful strategies for each step of the process to put on posters that will be used throughout the unit and year.	
	Lesson 3 Alternate Lessons (Select One)	**Lesson 3: Exploring Problem Solving** In this lesson, the class applies the problem solving process to three different problems: a word search, a seating arrangement for a birthday party, and planning a trip. The problems grow increasingly complex and poorly defined to highlight how the problem solving process is particularly helpful when tackling these types of problems.
		Lesson 3: Exploring Problem Solving - Animal Theme In this lesson, the class applies the problem solving process to three different problems: a tangram puzzle, choosing a pet according to criteria, and planning a pet adoption event. The problems grow increasingly complex and poorly defined to highlight how the problem solving process is particularly helpful when tackling these types of problems.
		Lesson 3: Exploring Problem Solving - Games Theme In this lesson, the class applies the problem solving process to three different problems: a maze, a logic puzzle, and planning field day activity. The problems grow increasingly complex and poorly defined to highlight how the problem solving process is particularly helpful when tackling these types of problems.
Chapter 2 Computers and Problem Solving	**Lesson 4: What is a Computer?** In this lesson, the class develops a preliminary definition of a computer. After brainstorming the possible definitions for a computer, the class works in groups to sort pictures into “is a computer” or “is not a computer” categories on poster paper, and explain their motivations for choosing some of the most difficult categorizations. The teacher then introduces a definition of the computer and allows groups to revise their posters according to the new definition.	

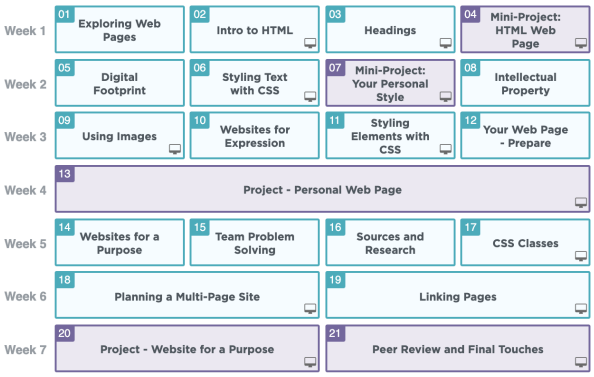
Chapter 2 Computers and Problem Solving (continued)	**Lesson 5: Input and Output** In this lesson, the class considers how computers get and give information to the user through inputs and outputs. After first considering what information they would need to solve a "thinking problem", the class identifies the inputs and outputs to that process. Afterwards, they explore a series of apps and determine the inputs and outputs for each one.
	Lesson 6: Processing This lesson introduces four types of processing that the class will use throughout the course. Through a series of apps, the class explores how processing is used to turn input into output. In the end, the class brainstorms more types of app processing that would be useful.
	Lesson 7: Storage This lesson covers the last part of the chapter's model of computing: storage. The class interacts with several different apps, determining which information should be stored for later and why. The input-output-storage-processing model of computing is then presented in full, and the class reflects on how various apps use each of the components, and how the model impacts whether or not something should be considered a computer.
	Lesson 8: Project - Propose an App To conclude the study of the problem solving process and the input/output/store/process model of a computer, the class proposes apps designed to solve real world problems. This project is completed across multiple days and culminates in a poster presentation where students highlight the features of their apps. The project is designed to be completed in pairs, though it can be completed individually.



Unit 2 - Web Development

Overview and Timeline

In Web Development, students are empowered to create and share content on their own web pages. They begin by thinking about the role of the web and how it can be used as a medium for creative expression. As students develop their pages and begin to see themselves as programmers, they are encouraged to think critically about the impact of sharing information online and how to be more critical consumers of content. They are also introduced to problem solving as it relates to programming while they learn valuable skills such as debugging, using resources, and teamwork. At the conclusion of the unit, students will have created a personal website they can publish and share.



Big Questions

Chapter 1 - Creating Web Pages

- Why do people create websites?
- How can text communicate content and structure on a web page?
- How do I safely and appropriately make use of the content published on the internet?
- What strategies can I use when coding to find and fix issues?

Chapter 2 - Multi-Page Websites

- How can websites be used to address problems in the world?
- What strategies can teams use to work better together?
- How do I know what information can be trusted online?

Unit Goals

By the end of the unit, students should be able to create a digital artifact that uses multiple computer languages to control the structure and style of their content, and view computer science as a tool for personal expression. They should understand that different programming languages allow them to solve different problems, and that these solutions can be generalized across similar problems. Lastly, they should understand their role and responsibilities as both creators and consumers of digital media.

Major Projects

- **Lesson 13: Project - Personal Web Page**
- **Lesson 20: Project - Website for a Purpose**

Throughout the unit, students use their developing skills to create a multi-page website, and have several opportunities to share out and engage in peer review at the end of each chapter. These projects emphasize many of the core practices of this course as students will need to tap into their creativity, problem solving skills, and persistence to complete their websites. To learn more about these projects, check out the descriptions in this unit's lesson progression.

Tools

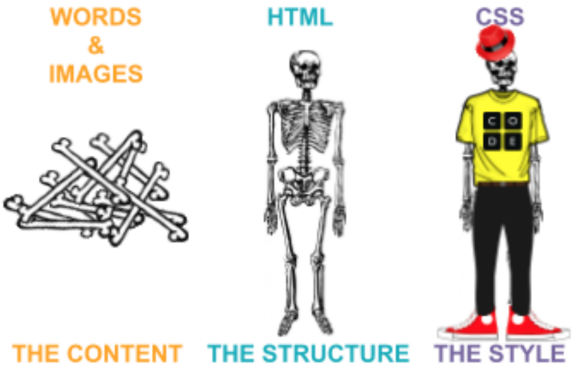
This unit uses **Web Lab**. For more detailed information, see the **Learning Tools** section of this curriculum guide

Lesson Progression: Unit 2 - Web Development

Chapter 1 Creating Web Pages	**Lesson 1: Exploring Websites** This lesson covers the purposes that a web page might serve, both for the users and the creators. The class explores a handful of sample web pages and describes how each of those pages is useful for users and how they might also serve their creators.
	Lesson 2: Intro to HTML This lesson introduces HTML as a solution to the problem of how to communicate both the content and structure of a website to a computer. The lesson begins with a brief unplugged activity that demonstrates the challenges of effectively communicating the structure of a web page. Then, the class looks at an HTML page in Web Lab and discusses how HTML tags help solve this problem, before using HTML to write their first web pages of the unit.
	Lesson 3: Headings This lesson continues the introduction to HTML tags, this time with headings. The class practices using heading tags to create page and section titles and learns how the different heading elements are displayed by default.
	Lesson 4: Mini-Project: HTML Web Page In this lesson, the class creates personal web pages on a topic of their choice. The lesson starts with a review of HTML tags. Next, the class designs web pages, first identifying the tags needed to implement them, and then creating the pages in Web Lab.
	Lesson 5: Digital Footprint This lesson takes a step back from creating the personal website to talk about the personal information that people choose to share digitally. The class begins by discussing what types of information they have shared on various websites, then they look at several sample social media pages to see what types of personal information could be shared intentionally or unintentionally. Finally, the class comes up with a set of guidelines to follow when putting information online.
	Lesson 6: Styling Text with CSS This lesson introduces CSS as a way to style elements on the page. The class learns the basic syntax for CSS rule-sets and then explores properties that impact HTML text elements. Finally, they discuss the differences between content, structure, and style when making a personal web page.
	Lesson 7: IMini-Project: Your Personal Style In this lesson, students create their own styled web pages. The lesson starts with a review of the CSS. They then design the web page, identify which CSS properties they will need, and create their web pages in Web Lab.
	Lesson 8: Intellectual Property Starting with a discussion of their personal opinions on how others should be allowed to use their work, the class explores the purpose and role of copyright for both creators and users of creative content. They then move on to an activity exploring the various Creative Commons licenses as a solution to the difficulties of dealing with copyright.
	Lesson 9: Using Images The class starts by considering the ethical implications of using images on websites, specifically in terms of intellectual property. They then learn how to add images to their web pages using the <img> tag and how to cite the image sources appropriately.

Chapter 1 Creating Web Pages (continued)	**Lesson 10: Websites for Expression** This lesson introduces websites as a means of personal expression. Students first discuss the different ways that people express and share their interests and ideas, then they look at a few exemplar websites made by students from a previous course. Finally, everyone brainstorms and shares a list of topics and interests to include in a personal website, creating a resource for developing a personal website in the rest of the unit.
	Lesson 11: Styling Elements with CSS This lesson continues the introduction to CSS style properties, this time focusing more on non-text elements. The class begins by investigating and modifying the new CSS styles on a Desserts of the World page. Afterwards, everyone applies this new knowledge to their personal websites.
	Lesson 12: Your Web Page - Prepare In this lesson, students engage in the "prepare" stage of the problem solving process by deciding what elements and style their web pages will have. They review the different HTML, CSS, and digital citizenship guidelines, then design and plan their pages, as well as download and document the images they will need. Afterwards, they reflect on how their plan will ensure that the website does what it is designed to do.
	Lesson 13: Project - Personal Web Page After quickly reviewing the debugging process, the class goes online to create the pages that they have planned out in previous lessons, with the project guides as a reference. Afterwards, they engage in a structured reflection and feedback process before making any final updates.
Chapter 2 Multi-page Websites	**Lesson 14: Websites for a Purpose** In this lesson, students explore the different reasons people make websites. After brainstorming various reasons that they visit websites, they investigate sample web sites that have been created to address a particular problem and decide what different purposes those websites might serve for the creators. The class then thinks of problems they might want to solve with their own websites.
	Lesson 15: Team Problem Solving Teams work together to set group norms and brainstorm what features they would like their websites to have. The class starts by reflecting on what makes teams successful. Teams then make plans for how they will interact and achieve success in their own projects before brainstorming ideas for their website projects.
	Lesson 16: Sources and Research This lesson covers how to find relevant and trustworthy information online. After viewing and discussing a video about how search engines work, students search for information relevant to their sites, then analyze the sites for credibility to decide which are appropriate to use on their own website.
	Lesson 17: CSS Classes This lesson introduces CSS classes, which allow web developers to treat groups of elements they want styled differently than other elements of the same type. Students first investigate and modify classes on various pages, then create their own classes and use them to better control the appearance of their pages. Teams then reflect on how they could use this skill to improve their websites.
	Lesson 18: Planning a Multi-Page Site The class works in teams to plan out the final web sites, including a sketch of each page. They then download the media that they will need for their sites. At the end of the activity, they decide how the work will be distributed among them and report whether the entire team agreed to the plan.

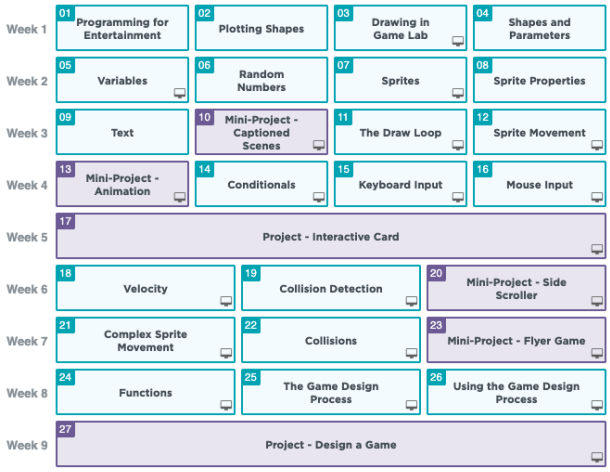
Chapter 2 Multi-page Websites (continued)	**Lesson 19: Linking Pages** The class begins this lesson by looking online for the internet's first web page and discussing how its use of links was what started the web. They then transition to Web Lab where they learn how to make their own links, as well as good conventions that make it easier for users to navigate on a page. Last, they reflect on their team project and what their personal goals are for the final stretch.
	Lesson 20: Project - Website for a Purpose In this lesson, teams are finally able to code the pages that they have been planning. Using the project guide, the team works together and individually to code all of their pages, then puts all of the work together into a single site.
	Lesson 21: Peer Review and Final Touches This lesson focuses on the value of peer feedback. The class first reflects on what they are proud of, and what they would like feedback on. They then give and get that feedback through a structured process that includes the project rubric criteria. Afterwards, everyone puts the finishing touches on their sites and reflects on the process before a final showcase.



Unit 3 - Interactive Animations and Games

Overview and Timeline

In the Interactive Animations and Games unit, students build on their coding experience as they create programmatic images, animations, interactive art, and games. Starting off with simple, primitive shapes and building up to more sophisticated sprite-based games, students become familiar with the programming concepts and the design process computer scientists use daily. They then learn how these simpler constructs can be combined to create more complex programs. In the final project, students develop a personalized, interactive program. Along the way, they practice design, testing, and iteration, as they come to see that failure and debugging are an expected and valuable part of the programming process.



Big Questions

Chapter 1 - Images and Animations

- What is a computer program?
- What are the core features of most programming languages?
- How does programming enable creativity and individual expression?
- What practices and strategies will help me as I write programs?

Chapter 2 - Building Games

- How do software developers manage complexity and scale?
- How can programs be organized so that common problems only need to be solved once?
- How can I build on previous solutions to create even more complex behavior?

Unit Goals

By the end of the unit, students should be able to create an interactive animation or game that includes basic programming concepts such as control structures, variables, user input, and randomness. They should manage this task by working with others to break it down using objects (sprites) and functions. Throughout the process, they should give and respond constructively to peer feedback, and work with their teammates to complete a project. Students should leave this unit viewing themselves as computer programmers, and see programming as a fun and creative form of expression.

Major Projects

- **Lesson 17: Project - Interactive Card**
- **Lesson 27: Project - Design a Game**

There are two major projects in this unit, which are at the end of each chapter. Both offer students an opportunity to demonstrate what they've learned while leveraging creativity and peer feedback. To learn more about these projects, check out the descriptions in this unit's lesson progression.

Tools

This unit uses **Game Lab**. For more detailed information, see the **Learning Tools** section of this curriculum guide.

Lesson Progression: Unit 3 - Interactive Animations and Games

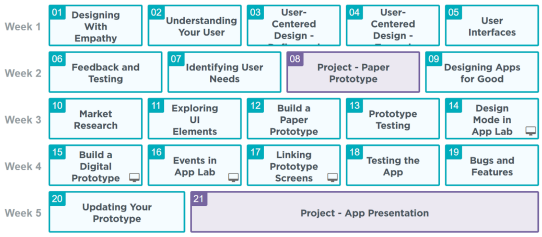
Chapter 1 Images and Animations	**Lesson 1: Programming for Entertainment** The class is asked to consider the "problems" of boredom and self expression, and to reflect on how they approach those problems in their own lives. From there, they will explore how Computer Science in general, and programming specifically, plays a role in either a specific form of entertainment or as a vehicle for self expression.
	Lesson 2: Plotting Shapes This lesson explores the challenges of communicating how to draw with shapes and use a tool that introduces how this problem is approached in Game Lab.The class uses a Game Lab tool to interactively place shapes on Game Lab's 400 by 400 grid. Partners then take turns instructing each other how to draw a hidden image using this tool, which accounts for many of the challenges of programming in Game Lab.
	Lesson 3: Drawing in Game Lab The class is introduced to Game Lab, the programming environment for this unit, and begins to use it to position shapes on the screen. The lesson covers the basics of sequencing and debugging, as well as a few simple commands. At the end of the lesson, students will be able to program images like the ones they made with the drawing tool in the previous lesson.
	Lesson 4: Shapes and Parameters In this lesson, students continue to develop a familiarity with Game Lab by manipulating the width and height of the shapes they use to draw. The lesson kicks off with a discussion that connects expanded block functionality (e.g. different sized shapes) with the need for more block inputs, or "parameters." Finally, the class learns to draw with versions of ellipse() and rect() that include width and height parameters and to use the background() block.
	Lesson 5: Variables This lesson introduces variables as a way to label a number in a program or save a randomly generated value. The class begins the lesson with a very basic description of the purpose of a variable and practices using the new blocks, then completes a level progression that reinforces the model of a variable as a way to label or name a number.
	Lesson 6: Random Numbers Students are introduced to the randomNumber() block and how it can be used to create new behaviors in their programs. They then learn how to update variables during a program and use those skills to draw randomized images.
	Lesson 7: Sprites In order to create more interesting and detailed images, the class is introduced to the sprite object. The lesson starts with a discussion of the various information that programs must keep track of, then presents sprites as a way to keep track of that information. Students then learn how to assign each sprite an image, which greatly increases the complexity of what can be drawn on the screen.
	Lesson 8: Sprite Properties Students extend their understanding of sprites by interacting with sprite properties. The lesson starts with a review of what a sprite is, then moves on to Game Lab for more practice with sprites, using their properties to change their appearance. The class then reflects on the connections between properties and variables.
	Lesson 9: Text This lesson introduces Game Lab's text commands, giving students more practice using the coordinate plane and parameters. At the beginning of the lesson, they are asked to caption a cartoon created in Game Lab. They then move onto Code Studio where they practice placing text on the screen and controlling other text properties, such as size.

Chapter 1 Images and Animations (continued)	**Lesson 10: Mini-Project - Captioned Scenes** After a quick review of the code learned so far, the class is introduced to the first creative project of the unit. Using the problem solving process as a model, students define the scene that they want to create, prepare by thinking of the different code they will need, try their plan in Game Lab, then reflect on what they have created. In the end, they also have a chance to share their creations with their peers.
	Lesson 11: The Draw Loop This lesson introduces the draw loop, one of the core programming paradigms in Game Lab. Students learn how to combine the draw loop with random numbers to manipulate some simple animations first with dots and then with sprites.
	Lesson 12: Sprite Movement In this lesson, the class learns how to control sprite movement using a construct called the counter pattern, which incrementally changes a sprite's properties. After brainstorming different ways that they could animate sprites by controlling their properties, students explore the counter pattern in Code Studio, using the counter pattern to create various types of sprite movements.
	Lesson 13: Mini-Project - Animation In this lesson, the class is asked to combine different methods from previous lessons to create an animated scene. Students first review the types of movement and animation that they have learned, and brainstorm what types of scenes might need that movement. They then begin to plan out their own animated scenes, which they create in Game Lab.
	Lesson 14: Conditionals This lesson introduces students to booleans and conditionals, which allow a program to run differently depending on whether a condition is true. The class starts by playing a game in which they respond according to whether particular conditions are met. They then move to Code Studio where they learn how the computer evaluates boolean expressions, and how they can be used to structure a program.
	Lesson 15: Keyboard Input Following the introduction to booleans and if statements in the previous lesson, students are introduced to a new block called keyDown(), which returns a boolean and can be used in conditionals statements to move sprites around the screen. By the end of this lesson they will have written programs that take keyboard input from the user to control sprites on the screen.
	Lesson 16: Mouse Input The class continues to explore ways to use conditional statements to take user input. In addition to the keyboard commands learned yesterday, they learn about several ways to take mouse input. They also expand their understanding of conditional to include `else`, which allows for the computer to run a certain section of code when a condition is true, and a different section of code when it is not.
	Lesson 17: Project - Interactive Card In this cumulative project for Chapter 1, students plan for and develop an interactive greeting card using all of the programming techniques they've learned to this point.
Chapter 2 Building Games	**Lesson 18: Velocity** After a brief review of how the counter pattern is used to move sprites, the class is introduced to the idea of hiding those patterns in a single block, in order to help manage the complexity of programs. They then head to Code Studio to try out new blocks that set a sprite's velocity directly, and look at the various ways that they are able to code more complex behaviors in their sprites.
	Lesson 19: Collision Detection In this lesson, the class learns about collision detection on the computer. Working in pairs, they explore how a computer could use math, along with the sprite location and size properties,h to detect whether two sprites are touching. They then use the isTouching() block to create different effects when sprites collide and practice using the block to model various interactions.

Chapter 2 Building Games (continued)	**Lesson 20: Mini-Project - Side Scroller** Students use what they have learned about collision detection and setting velocity to create simple side scroller games. After looking at a sample side scroller game, they brainstorm what sort of side scroller they would like, then use a structured process to program the game in Code Studio.
	Lesson 21: Complex Sprite Movement The class learns to combine the velocity properties of sprites with the counter pattern to create more complex sprite movement. After reviewing the two concepts, they explore various scenarios in which velocity is used in the counter pattern, and observe the different types of movement that result. In particular, students learn how to simulate gravity. They then reflect on how they were able to get new behaviors by combining blocks and patterns that they already knew.
	Lesson 22: Collisions In this lesson, the class programs their sprites to interact in new ways. After a brief review of how they used the isTouching block, students brainstorm other ways that two sprites could interact. They then use isTouching to make one sprite push another across the screen before practicing with the four collision blocks (collide, displace, bounce, and bounceOff).
	Lesson 23: Mini-Project - Flyer Game Students use what they have learned about simulating gravity and the different types of collisions to create simple flyer games. After looking at a sample flyer game, they brainstorm what sort of flyer they would like, then use a structured process to program the game in Code Studio.
	Lesson 24: Functions This lesson covers functions as a way for students to organize their code, make it more readable, and remove repeated blocks of code. The class learns that higher level or more abstract steps make it easier to understand and reason about steps, then begins to create functions in Game Lab.
	Lesson 25: The Game Design Process This lesson introduces the process that students will use to design games for the remainder of the unit. This process is centered around a project guide that asks students to define their sprites, variables, and functions before they begin programming their game. They walk through this process in a series of levels. At the end of the lesson, students have an opportunity to make improvements to the game to make it their own.
	Lesson 26: Using the Game Design Process In this multi-day lesson, the class uses the problem solving process from Unit 1 to create a platform jumper game. After looking at a sample game, they define what their games will look like and use a structured process to build them. Finally, the class reflects on how the games could be improved, and implements those changes.
	Lesson 27: Project - Design a Game Students plan and build original games using the project guide from the previous two lessons. Working individually or in pairs, they plan, develop, and give feedback on the games. After incorporating the peer feedback, students share out their completed games.

Unit 4 - The Design Process

Overview and Timeline



The Design Process unit transitions students from thinking about computer science as a tool to solve their own problems towards considering the broader social impacts of computing. Through a series of design challenges, students are asked to consider and understand the needs of others while developing a solution to a problem. The second half of the unit consists of an iterative team project, during which students have the opportunity to identify a need that they care about, prototype solutions both on paper and in App Lab, and test their solutions with real users to get feedback and drive further iteration.

Big Questions

Chapter 1: User Centered Design

- How do computer scientists identify the needs of their users?
- How can we ensure that a user's needs are met by our designs?
- What processes will best allow us to efficiently create, test, and iterate upon our design?

Chapter 2: App Prototyping

- How do teams effectively work together to develop software?
- What roles beyond programming are necessary to design and develop software?
- How do designers incorporate feedback into multiple iterations of a product?

Unit Goals

By the end of the unit, students should see the design process as a form of problem solving that prioritizes the needs of a user. They should be able to identify user needs and assess how well different designs address them. In particular, they know how to develop paper and digital prototypes, gather and respond to feedback about a prototype, and consider ways different user interfaces do or do not affect the usability of their apps. Students should leave the unit with a basic understanding of other roles in software development, such as product management, marketing, design, and testing, and how to use what they have learned about computer science as a tool for social impact.

Major Projects

- **Lesson 8: Project - Paper Prototype**
- **Lesson 21: Project - App Presentation**

Students are encouraged to focus on the design process in the two major projects in this unit, which are at the end of each chapter. To learn more about these projects, check out the descriptions in this unit's lesson progression.

Tools

This unit uses the **App Lab**. For more detailed information, see the **Learning Tools** section in this curriculum guide.

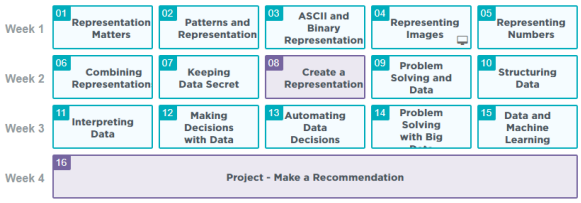
Lesson Progression: Unit 4 - The Design Process

Chapter 1 User Centered Design	**Lesson 1: Designing with Empathy** The class explores a variety of different shoe designs to consider design choices. Building on this, students explore the relationship between users, their needs, and the design of objects they use.
	Lesson 2: Understanding Your User Using user profiles, students explore how different users might react to a variety of products. Role playing as a different person, each member of the class will get to experience designs through someone else's eyes.
	Lesson 3: User-Centered Design - Define and Prepare In small groups, students use the design process to come up with ideas for smart clothing. Today's lesson focuses on brainstorming users and ideas that will meet their needs. Over the course of both lessons, students will brainstorm ideas, identify users, and finally propose a design. This activity serves as the first of several opportunities for students to practice designing a solution for the needs of others.
	Lesson 4: User-Centered Design - Try and Reflect In small groups, students will use the design process to come up with ideas for smart clothing. Today's lesson focuses on creating a design and reflecting on how well it meets the needs of users. Over the course of both lessons, students will brainstorm ideas, identify users, and finally propose a design. This activity serves as the first of several opportunities for students to practice designing a solution for the needs of others.
	Lesson 5: User Interfaces In this lesson, students get to see how a paper prototype can be used to test and get feedback before writing any code. To help out a developer with their idea, the class tests and provides feedback on an app prototype made of paper.
	Lesson 6: Feedback and Testing Users have been testing an app, and they have lots of feedback for the developer. The class needs to sort through all of this feedback, identify the common themes and needs, and start revising the prototype to make it better meet the users' needs.
	Lesson 7: Identifying User Needs In this lesson, the class begins thinking about designing their own paper prototype for an app that can solve a problem in our community. Using interviews from different users, students identify needs and interests that they can use to design an app for these people in their community.
	Lesson 8: Project - Paper Prototype Using the interview information from the previous lesson, students come up with app ideas to address the needs of their users. To express those ideas, and test out their effectiveness, students create and test paper prototypes.
Chapter 2 App Prototyping	**Lesson 9: Designing Apps for Good** To kick off the app design project, the class organizes into teams and starts exploring app topics. Several examples of socially impactful apps serve as inspiration for the project.
	Lesson 10: Market Research In this lesson, the class dives into app development by exploring existing apps that may serve similar users. In groups, students will identify a handful of apps that address the same topic they are working on, and use those apps to help refine the app idea they will pursue.

Chapter 2 App Prototyping *(continued)*	**Lesson 11: Exploring UI Elements** Paper prototypes allow developers to quickly test ideas before investing a lot of time writing code. In this lesson, teams explore some example apps created in App Lab and use these examples to help inform the first paper prototypes of their apps.
	Lesson 12: Build a Paper Prototype In teams, students will create a paper prototype for the app they've been developing. Each team member will create a different screen and design how the user will navigate between each screen.
	Lesson 13: Prototype Testing In this lesson, teams test out their paper prototypes with other members of the class. As one student role plays as the computer, one narrates, and the rest observe, teams will get immediate feedback on their app designs, which will inform the next version of their app prototypes.
	Lesson 14: Design Mode in App Lab Teams now move to App Lab to build the next iteration of their apps. This lesson focuses on how to use Design Mode in App Lab to create digital prototypes for their apps.
	Lesson 15: Build a Digital Prototype Using the drag-and-drop Design Mode, each team member builds out at least one page of their team's app, responding to the feedback received in the previous round of testing.
	Lesson 16: Events in App Lab Building on the previous lesson, we learn how to import new screens into our apps and link them together using buttons and events to complete the Recycle Finder app we started in an earlier lesson.
	Lesson 17: Linking Prototype Screens Building on the screens that they designed in the previous lesson, teams combine screens into a single app. Simple code can then be added to make button clicks change to the appropriate screen.
	Lesson 18: Testing the App In this lesson, teams run another round of user testing with their interactive prototype. Feedback gathered from this round of testing will inform the final iteration of the digital prototype.
	Lesson 19: Bugs and Features Teams analyze the feedback they received from the last round of testing and make a plan for how they would like to address it. Students categorize feedback as either a bug or a feature and decide which items are most important for improving their app.
	Lesson 20: Updating Your Prototype Using the feedback from the last round of testing, teams implement changes that address the needs of their users. Each team tracks and prioritizes the features they want to add and the bugs they need to fix.
	Lesson 21: Project - App Presentation Each team prepares a presentation to "pitch" the app they've developed. This is the time they can share the struggles, triumphs, and plans for the future.

Unit 5 - Data and Society

Overview and Timeline



The Data and Society unit is about the importance of using data to solve problems and it highlights how computers can help in this process. The first chapter explores different systems used to represent information in a computer and the challenges and tradeoffs posed by using them. In the second chapter, students learn how collections of data are used to solve problems, and how computers help to automate the steps of this process. In the final project, students gather their own data and use it to develop an automated solution to a problem.

Big Questions

Chapter 1: Representing Information

- Why is representation important in problem solving?
- What features does a representation system need to be useful?
- What is necessary to create usable binary representation systems?
- How can we combine systems together to get more complex information?

Chapter 2: Solving Data Problems

- How does data help us to solve problems?
- How do computers and humans use data differently?
- What parts of the data problem solving process can be automated?
- What kinds of real world problems do computers solve by using data?

Unit Goals

By the end of the unit, students should have a broad understanding of the role of data and data representation in solving information problems. They should be able to explain the necessary components of any data representation scheme, as well as the particulars of binary and the common ways that various types of simple and complex data are represented in binary code. Students should also be able to design and implement a data-based solution to a given problem and determine how the different aspects of this problem solving process could be automated.

Major Projects

- **Lesson 8: Project - Create a Representation**
- **Lesson 16: Project - Make a Recommendation**

There are two major projects in this unit, which are at the end of each chapter. Each project asks students to take a different approach to looking data — the first focuses on building a system to represent complex data, and the second focuses on processing data to make a recommendation. To learn more about these projects, check out the descriptions in this unit's lesson progression.

Lesson Progression: Unit 5 - Data and Society

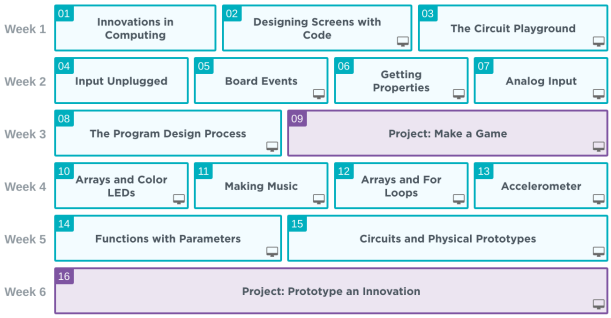
Chapter 1 Representing Information	**Lesson 1: Representation Matters** This first lesson provides an overview of what data is and how it is used to solve problems. Groups use a data set to make a series of meal recommendations for people with various criteria. Afterward, groups compare their responses and discuss how the different representations of the meal data affected how they were able to solve the different problems.
	Lesson 2: Patterns and Representation This lesson looks closer at what is needed to create a system of representation. Groups create systems that can represent any letter in the alphabet using only a single stack of cards. They then create messages with their systems and exchange with other groups to ensure the system worked as intended. Finally, the class discusses commonalities between working systems while recognizing that there are many possible working solutions.
	Lesson 3: ASCII and Binary Representation This lesson introduces students to a formal binary system for encoding information: the ASCII system for representing letters and other characters. At the beginning of the lesson, the teacher introduces the fact that computers must represent information using either "on" or "off." The class then learns about the ASCII system for representing text using binary symbols and practices using this system. Finally, they encode their own messages using ASCII.
	Lesson 4: Representing Images This lesson continues the study of binary representation systems, this time with images. The class is introduced to the concept of splitting images into squares or "pixels," which can then be turned on or off individually to make an entire image. After doing a short set of challenges using the Pixelation Widget, students make connections between the system for representing images and the ASCII system for representing text that they learned about in the previous lesson.
	Lesson 5: Representing Numbers This lesson introduces students to the binary number system. With a set of cards that represent the place values in a binary (base-2) number system, the class turns bits "on" or "off" by turning cards face up and face down, then observes the numbers that result from these different patterns. Eventually, the pattern is extended to a generic 4-bit system.
	Lesson 6: Combining Representations This lesson combines all three types of binary representation systems (ASCII characters, binary numbers, and images) to explore ways to encode more complex types of information in a record. After seeing a series of bits and being asked to decode them, students are introduced to the idea that understanding binary information requires an understanding of both the system that is being used, and the meaning of the information encoded.
	Lesson 7: Keeping Data Secret Students continue to explore how data is represented in a punchcard, and begin considering whether some data should be protected from public view because it is too personal or sensitive. Once students understand the reasons for protecting data, they learn a binary encryption system that lets them encrypt and decrypt data in their punchcards.
	Lesson 8: Project - Create a Representation The class designs structures to represent their perfect day using the binary representation systems they've learned in this chapter. After deciding which pieces of information the record should capture, students decide how a punch card of bytes of information will be interpreted to represent those pieces of information. Afterwards, they use the ASCII, binary number, and image formats they have learned to represent their perfect days and try to decipher what a partner's perfect day is like.

Chapter 2 Solving Data Problems	**Lesson 9: Problem Solving and Data** This lesson covers how the problem solving process can be tailored to deal with data problems. The class is tasked with deciding what a city most needs to spend resources on. They must find and use data from the internet to support their decision.
	Lesson 10: Structuring Data This lesson goes further into the interpretation of data, including how to clean and visualize raw data sets. The class first looks at how presenting data in different ways can help people to understand it better. After seeing how cleaning and visualization can help people make better decisions, students look at which parts of this process can be automated, and which parts need a human.
	Lesson 11: Interpreting Data Students look at a cake preference survey and discuss how knowing the relationship between cake and icing preference helps them better decide which combination to recommend. Students are then introduced to cross tabulation, which allows them to graph relationships to different preferences. They use this technique to find relationships in a preference survey, then brainstorm the different types of problems that this process could help solve.
	Lesson 12: Making Decisions with Data This lesson gives students a chance to practice the data problem solving process introduced in the last lesson. Not all questions have right answers, and in some cases the class can and should decide that they need to collect more data. The lesson concludes with a discussion about how different people could draw different conclusions from the same data, and how collecting different data might have affected the decisions they made.
	Lesson 13: Automating Data Decisions In this lesson, students create an algorithm that suggests a vacation spot. They then create rules that a computer could use to make this decision automatically. Students share their rules and test their rules with the class data. Next, they use data from their classmates to test whether their rules would make the same decision that a person would. The lesson concludes with a discussion about the benefits and drawbacks of using computers to automate the data problem solving process.
	Lesson 14: Problem Solving with Big Data This lesson covers how data is collected and used to solve problems in the real world. Students look at three scenarios that could be solved using data and brainstorm the types of data they would want to use to solve each problem, as well as strategies they could use to collect the data. Each scenario also includes a video about a real-world service that has solved a similar problem with data.
	Lesson 15: Data and Machine Learning Students explore how machine learning can be used to make decisions about data. They help an AI Bot learn how to identify fish, which will help it clean up trash from the ocean floor. Along the way, they see how machine learning and big data can be used to solve problems in society.
	Lesson 16: Project - Make a Recommendation To conclude this unit, the class designs ways to use data to make recommendations or predictions to help solve a problem. In the first several steps, students brainstorm problems, perform simple research, and define a problem of their choosing. They then decide what kind of data they want to collect, how it could be collected, and how it could be used, before exchanging feedback and giving a final presentation.

Unit 6 - Physical Computing

Overview and Timeline

In the Physical Computing unit, students further develop their programming skills, while exploring more deeply the role of hardware platforms in computing. Harkening back to the Input/Storage/Processing/Output model for a computer, students look towards modern “smart” devices to understand the ways in which non-traditional computing platforms take input and provide output in ways that couldn't be done with the traditional keyboard, mouse, and monitor.



Using App Lab and Adafruit’s Circuit Playground, students develop programs that utilize the same hardware inputs and outputs that we see in many modern smart devices, and they get to see how a simple rough prototype can lead to a finished product. The unit concludes with a design challenge that asks students to use the Circuit Playground as the basis for an innovation of their own design.

Big Questions

Chapter 1: Programming with Hardware

- How does software interact with hardware?
- How can computers sense and respond to their environment?
- What kind of information can be communicated with hardware outputs?

Chapter 2: Building Physical Prototypes

- How do programmers work with larger amounts of similar values?
- How can complex real-world information be represented in code?
- How can simple hardware be used to develop innovative new products?

Unit Goals

By the end of the unit, students should be able to design and build a physical computing device that integrates hardware inputs and outputs with software. This unit builds on the skills and understandings from the Interactive Animations and Games unit with more sophisticated programming constructs, such as arrays, for-loops, and parameters, as well as deepens students’ understanding of the types of input and output that can be used in computing. Students should leave the unit feeling equipped to use physical computing to solve problems in fun and innovative ways.

Major Projects

- **Lesson 9: Project - Make a Game**
- **Lesson 16: Project - Prototype an Innovation**

There are two major projects in this unit, which are at the end of each chapter. Each project is cumulative for the chapter completed, and each takes a different approach to hardware — the first focusing on building a game that uses the inputs and outputs of Circuit Playground and the second focusing on developing and testing an physical prototype of an innovative computing device. To learn more about these projects, check out the descriptions in this unit’s lesson progression.

Tools

This unit uses **Maker App.** For more detailed information, see the **Learning Tools** section of this curriculum guide.

Lesson Progression: Unit 6 - Physical Computing

Chapter 1 Programming with Hardware	**Lesson 1: Innovations in Computing** In this lesson, students explore a wide variety of new and innovative computing platforms while expanding their understanding of what a computer can be.
	Lesson 2: Designing Screens with Code By reading and changing the content on the screen of an app, the class starts to build apps that only need a single screen. Even with just one screen, students can begin to see that these techniques allow for lots of user interaction and functionality.
	Lesson 3: The Circuit Playground In this lesson, students get to know the Circuit Playground, the circuit board that will be used throughout the rest of this unit. Using App Lab, they develop programs that use the Circuit Playground for output.
	Lesson 4: Input Unplugged Students experience two different ways that an app can collect input from a user, while learning more about the event-driven programming model used in App Lab.
	Lesson 5: Board Events Using the hardware buttons and switch, students develop programs that use the Circuit Playground as an input.
	Lesson 6: Getting Properties This lesson introduces students to the getProperty block, which allows them to access the properties of different elements with code. They first practice using the block to determine what the user has input in various user interface elements. Students later use getProperty and setProperty together with the counter pattern to make elements move across the screen. A new screen element, the slider, and a new event trigger, onChange, are also introduced.
	Lesson 7: Analog Input Students get to explore the analog inputs on the Circuit Playground, writing programs that respond to the environment through sensors.
	Lesson 8: The Program Design Process This lesson introduces students to the process they will use to design programs of their own throughout this unit. This process is centered around a project guide that asks students to sketch out their screens, identify elements of the Circuit Playground to be used, define variables, and describe events before they begin programming (a process very similar to the Game Design Process that students used in Unit 3). Students begin by playing a tug o' war style game where the code is hidden. They discuss what they think the board components, events, and variables would need to be to make the program. Then, they are then given a completed project guide that shows one way to implement the project, and are walked through this process in a series of levels. At the end of the lesson, students have an opportunity to make improvements to the program to make it their own.
	Lesson 9: Project: Make a Game For this project, students design and create a game that leverages the new inputs and outputs that are available to them. This project is purposefully left very open-ended to empower students to think broadly about how physical output might be useful in an app, while still giving them a chance to review the program development process and try out the new features available through the Circuit Playground.

Chapter 2 Building Physical Prototypes

Lesson 10: Arrays and Color LEDs
In this lesson, students are introduced to the ring of color LEDs, which are exposed as an array called colorLeds. Students learn how to access and control each LED in an array individually, preparing them to access multiple LEDs through iteration later in the chapter.

Lesson 11: Making Music
In this lesson, students will use the Circuit Playground's buzzer feature to its full extent by producing sounds, notes, and songs. Students start with a short review of the buzzer's frequency and duration parameters, then move on to the concept of musical notes. Notes allow students to constrain themselves to frequencies that are used in Western music and provide a layer of abstraction that helps them to understand which frequencies might sound good together. Once students are able to play notes on the buzzer, they use arrays to hold and play sequences of notes, and compose simple songs.

Lesson 12: Arrays and For Loops
Students learn to combine lists and for-loops in this lesson, which allows them to write code that impacts every element of a list, regardless of how long it is. The class uses this structure to write programs that process all of the elements in lists, including the list of color LEDs.

Lesson 13: Accelerometer
In this lesson, students explore the Circuit Playground's accelerometer feature and its capabilities. They become familiar with the accelerometer's events and properties as they create multiple programs with the feature, similar to those they've likely come across in real world applications.

Lesson 14: Functions with Parameters
This lesson starts with a quick review of parameters in the context of the App Lab blocks that students have seen recently. Students then look at examples of parameters within user-created functions in App Lab, and create and call functions with parameters to control multiple elements on a screen. Afterward, they use for loops to iterate over an array, passing each element into a function. Last, students use what they have learned to create a star catching game.

Lesson 15: Circuits and Physical Prototypes
In this lesson, students wire simple circuits to create a physical prototype using low-cost and easily-found materials.

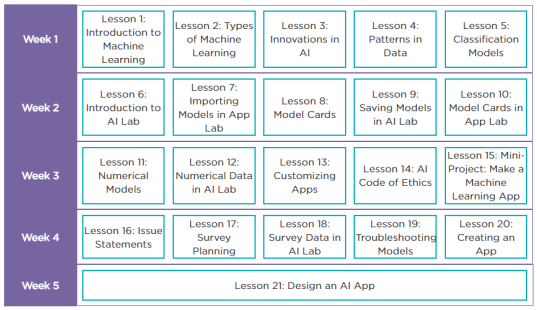
Lesson 16: Project: Prototype an Innovation
This final project challenges students to develop and test a prototype for an innovative computing device that interacts with the physical world through various types of input and output, whichallow for interesting and unique user interactions. This project is an opportunity for students to showcase their technical skills, but they will also need to demonstrate collaboration, constructive peer feedback, and iterative problem solving as they encounter obstacles along the way. This project should be student-directed whenever possible, and provide an empowering and memorable conclusion to the final unit of CS Discoveries.

Students using the Circuit Playground

Optional Unit - AI and Machine Learning

Overview and Timeline

In this optional AI and Machine Learning unit, students learn how computers can find patterns in data to make decisions. Students use the Problem Solving Process for machine learning to define a problem, prepare their data, train a model, then test and evaluate their model for accuracy and potential bias. Students explore a variety of scenarios and datasets that lend themselves to machine learning. They also explore some of the modern problems with machine learning, especially around bias and impact.



Big Questions

Chapter 1: Understanding Machine Learning

- How does machine learning find patterns in data to make decisions?
- How can we avoid bias when training a machine learning model?

Chapter 2: Design a Machine Learning App

- How can machine learning be used to solve problems in our community?

Unit Goals

Students will use AI Lab to create a machine learning model to solve a problem, and use App Lab to create an app that uses their model. Students should leave the unit with an understanding of how machine learning models make decisions from data, and with the ability to create machine learning models from their own data to solve problems in their community. Students use AI Lab to train their machine learning models, then import their models into App Lab to further customize their apps. This unit assumes students have prior App Lab experience, so we recommend completing this unit after Unit 4, which focuses on designing for users and introduces App Lab.

Major Projects

- **Lesson 15: Make a Machine Learning App**
- **Lesson 21: Design an AI App**

Tools

This unit uses **AI Lab** and **App Lab.** For more detailed information, see the **Learning Tools** section of this curriculum guide.

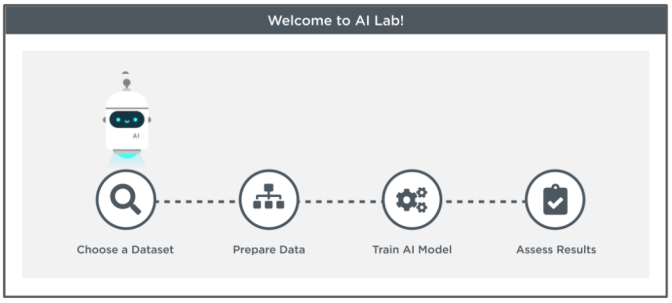
Lesson Progression: Optional Unit - AI and Machine Learning

Click on the title of any lesson to view the lesson plan

Chapter 1: Understanding Machine Learning	**Lesson 1: Introduction to Machine Learning** In this lesson students are introduced to a form of artificial intelligence called machine learning and how they can use the Problem Solving Process to help train a robot to solve problems. They participate in three machine learning activities where a robot - A.I. Bot - is learning how to detect patterns in fish.
	Lesson 2: Types of Machine Learning In this lesson students will consider how they create “mental” models when learning new concepts, and how those can be similar to a “machine learning” model. They participate in a color pattern activity to simulate building a machine learning model without help, then they play a game called "Green Glass Door" as an example of supervised learning, and finally, they will sort several scenarios into “supervised” or “unsupervised” learning.
	Lesson 3: Innovations in AI In this lesson, students explore an application of AI called Seeing AI and examine how it is supporting people with visual impairments. Then, students research other examples of how AI is impacting society, focusing on users who are impacted by the examples they find. Finally, students share their findings with each other.
	Lesson 4: Patterns in Data In this lesson students will examine several apps that make decisions about what shoes to wear, ultimately building up to an understanding of how machine learning can help make this decision. Students are guided to the conclusion that surveying their users can help them make the best decision by looking for patterns in the data and basing their decisions on these patterns.
	Lesson 5: Classification Models In this lesson students will participate in an unplugged activity simulating one of the machine learning algorithms computers use to separate data into groups to help make decisions. Students will be tasked with helping a computer learn to classify food as fruits or vegetables, graph 20 different fruits on two axes comparing “sweetness” to “easy to eat”, and then try to separate the data into groups - a fruit area, and a veggie area.
	Lesson 6: Introduction to AI Lab In this lesson students will dive into the AI Lab tool for the first time, where they select features to train a model that predicts a given label. They start by exploring AI Lab and training a model to recognize shapes. Then they pretend they have been hired by several restaurants who would like to make recommendations to new customers based on survey data they’re collected, go through each dataset, and use data visualization tools to identify features with high relationships in the data.
	Lesson 7: Importing Models in App Lab In this lesson students are introduced to importing their models into App Lab and linking their model to their screens. They help create a book recommendation app and learn how to add a welcome screen and events to their code. This lesson assumes students are already familiar with App Lab - for classrooms that have not seen App Lab before, consider extending this lesson and including additional videos or activities that are recommended in the lesson plan.
	Lesson 8: Model Cards In this lesson, students will investigate a model for bias and be introduced to a Model Card, which is a way of representing important information about a trained model that could help uncover bias. They will be investigating a Medical Priority app, which helps a hospital decide how soon to view patients based on their symptoms. As students go through the activity, they realize that the app is biased based on personal information and examine how this could happen.

Chapter 1: Understanding Machine Learning	**Lesson 9: Saving Models in AI Lab** Students complete the full process of training and saving a model, then importing into App Lab. For the first time, students are able to choose the label they would like to predict and spend time deciding the features they will use to help predict their label of choice. Students also create a model card for their models in order to save them and import it into App Lab
	Lesson 10: Model Cards in App Lab In this lesson, students practice importing their models into App Lab, this time including models that have numerical data and using model cards to help improve the user experience of filling out their form. They will then learn how to view the model card within App Lab and use this to add more descriptive elements to an app. Next, they focus on improving the user experience by adding informational text to help guide users through completing the form and adding a style to their app to improve the user experience.
	Lesson 11: Numerical Models In this lesson, students participate in an unplugged activity simulating a zombie outbreak. Students must predict which parts of town have the least amount of zombies using data from a neighboring town. Students will use degrees of similarity and averages to make predictions about the number of zombies at a particular location. Then, students are rescued and get to compare their predictions to the actual numbers as a way to discuss how accuracy is different for numerical data compared to categorical data.
	Lesson 12: Numerical Data in AI Lab In this lesson, students will be introduced to numerical data which represents a range of values. Students are presented with a scenario where every feature and label is represented with numerical data, and they learn to use the new data visualization tools within AI Lab to help find patterns.
	Lesson 13: Customizing Apps In this lesson, students will explore how to customize the code of their app to make additional changes to the design of their app. They will start by exploring a single-screen app and then practice expanding the app to two-screens and updating the code to use the new design mode elements. After this, students help create a Driver Alert app that requires changes to the code using new design mode elements. Using the skills from this lesson, students will be able to create multi-screen apps where questions can appear on multiple screens instead of a single screen.
	Lesson 14: AI Code of Ethics In small groups, students conduct research using articles and videos that expose ethical pitfalls in an Artificial Intelligence (AI) area of their choice. Afterward, each group develops at least one solution-oriented principle that addresses their chosen area. These principles are then assembled into a class-wide “Our AI Code of Ethics” resource (e.g. a slide presentation, document, or webpage) for AI creators and legislators everywhere.
	Lesson 15: Mini Project: Make a Machine Learning App In this one or two day mini-project, students apply their skills from the unit so far and create a machine learning app using real-world data. Students are provided with several real-world datasets from a variety of contexts, and they choose which dataset they would like to investigate. They train and save their model, then make a simple App Lab app that uses the model. This mini-project is an opportunity to assess how well students can use features to create accurate machine learning models, and how well they can create apps that use machine learning.
Chapter 2: Design a Machine Learning App	**Lesson 16: Issue Statements** This is the first of a five-day sequence of lessons that prepare students for the final project. In this lesson, students meet a team of fictional students who want to use machine learning to address an issue in their community. Students participate in an issue brainstorm using the 5 Why's strategy, then they help evaluate the ideas that the other student team came up with. The steps students take in this lesson are identical to the steps students will take in their final project.

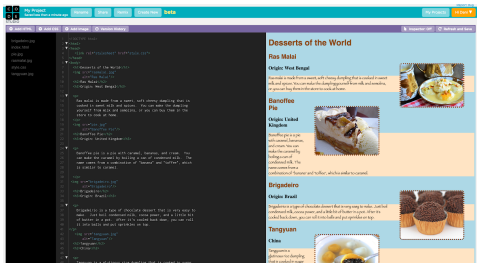
Chapter 2: Design a Machine Learning App	**Lesson 17: Survey Planning** This is the second in a five-day sequence of lessons that prepare students for the final project. In this lesson, students learn that the other team of students would like to create a club recommender app based on the clubs at their school. Students imagine what questions would be most useful to help make this recommendation, then they learn how to use a Google Form template to create a survey. The steps students take in this lesson are identical to the steps students will take in their final project.
	Lesson 18: Survey Data in AI Lab This is the third in a five-day sequence of lessons that prepare students for the final project. In this lesson, students learn how to view survey data in Google Sheets and save the data to their computer as a csv file. Then, they upload the saved data to AI Lab and examine the survey results from one of the students to train a model using their data. Then, students use Google Sheets to examine data from another student where the data has errors and then try to fix the errors. The steps students take in this lesson are identical to the steps students will take in their final project, and the problem-solving strategies they develop will help them overcome challenges in their own final project.
	Lesson 19: Troubleshooting Models This is the fourth of a five-day sequence of lessons that prepare students for the final project. In this lesson, students examine survey data from other members of the student team and analyze why their models are not working correctly. In examining the data, students develop strategies for avoiding these issues in the future and strategies for coping with these issues should they happen again. These are skills students will use in the final project as they develop their own surveys and collect data.
	Lesson 20: Creating an App This is the fifth of a five-day sequence of lessons that prepare students for the final project. In this lesson, students import the club recommender app into App Lab and begin customizing the app. Students add a welcome screen and update the descriptions of each feature, then they can decide how they would like to further customize the app. The steps students take in this lesson are identical to the steps students will take in their final project.
	Lesson 21: Design an AI App To conclude this unit, students develop an AI app that addresses the social issue they have returned to throughout the unit. After looking at a sample app, students follow a project guide to complete this multi-day activity. In the first step, students prepare the data they will use to train their model in AI Lab. After training, testing, and generating a model card, they export their model into App Lab for development. Here they use their model to create a user-friendly app based on their mockup from the previous lesson, “Planning Your App”. Students perform a peer review and make any necessary updates to their projects before preparing a presentation to the class.



Learning Tools

Web Lab

Where it's located in the curriculum: Unit 2 - Web Development



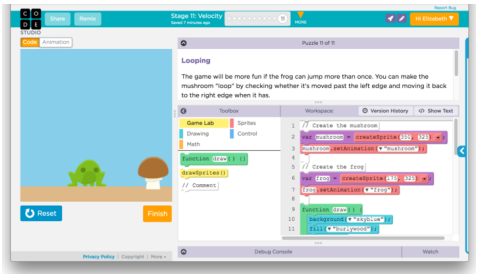
Description

Web Lab is a browser-based text editor for building web pages in HTML and CSS. It features a text editor with many helpful tools for creating and debugging HTML and CSS code, including a live preview of the web page that updates in real time and the ability to publish the completed web page to its own unique URL. Try the tool at code.org/weblab.

Game Lab

Where it's located in the curriculum: Unit 3 - Interactive Animations and Games

Description

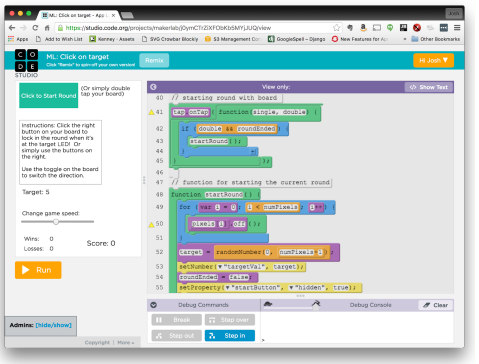


Game Lab is a programming environment for developing animations and games using JavaScript. Students start by creating sprites - characters whose appearance, movement, and interactions can be controlled through code. A preloaded library of images and sounds and a full-featured pixel editor allow students to customize the look of their sprites. Then, using Game Lab, students learn fundamental programming constructs while being given the freedom to create their own virtual worlds. Students can program using either blocks or text and instantly switch between either mode. Game Lab allows for activities with a scoped toolbox of commands that focuses attention on the specific blocks and concepts being introduced in that lesson. In addition, embedded support tools help students track down errors in their code. Try the tool at code.org/gamelab.

App Lab

Where it's located in the curriculum: Unit 4 - The Design Process

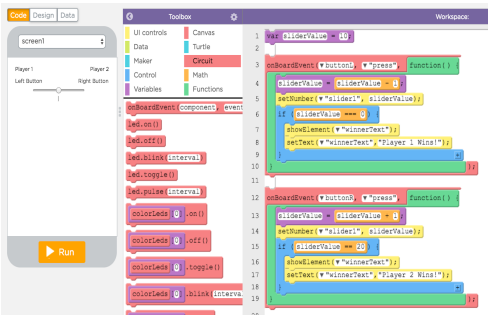
Description



App Lab is a programming environment for developing applications in JavaScript. A drag-and-drop editor allows students to add and edit page elements without having to write the associated HTML and CSS. Working in either blocks or text, students define the behavior of these page elements using code. The scoped toolbox of commands focuses attention on the specific blocks or concepts being introduced in that lesson. The embedded support tools help students track down errors in their code. Thanks to these features, App Lab is particularly well-suited for quickly prototyping apps. Try the tool at code.org/applab.

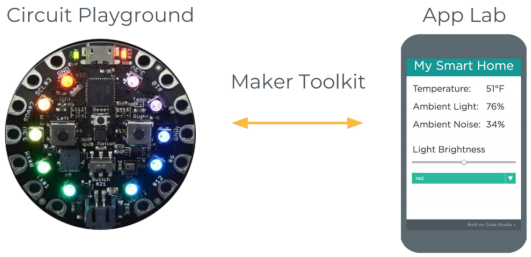
Maker App

Where it's located in the curriculum: Unit 6 - Physical Computing



Description

The Maker App is an environment that allows you to communicate with the Circuit Playground using a set of additional commands available in App Lab. Using the same drag-and-drop editor that students have become comfortable with in App Lab and Game Lab, students can turn on LEDs, read sensors, and write programs that use physical hardware for user input and output. By integrating these commands, App Lab allows you to quickly prototype apps that combine hardware and software without the additional challenge of transitioning to a new language or tool or worrying about wiring and electronics. Try the tool at studio.code.org/maker/setup.



AI Lab

Where it's located in the curriculum: AI and Machine Learning

Description

AI Lab is an environment that lets you interact with tabular data to train a machine learning model. You can select from several pre-generated datasets, or upload your own data. You decide what you would like to predict and which features to base your prediction off of. AI Lab will analyze the data and create a machine learning model which you can import into App Lab. Before saving your machine learning model, you also need to save a Model Card which acts like the documentation for your model.

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.

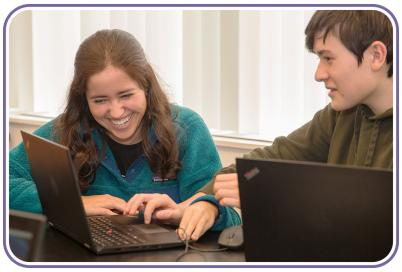
Tips for Effectively Teaching Unplugged Activities

- 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 plugged 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".



How are they used?

Plugged activities are designed to allow students to get hands-on with tools and concepts. That said, plugged lessons typically have many of the same features of their unplugged counterparts. 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

- Use warm ups, wrap ups, and suggested check-ins to ensure students are synthesizing concepts.
- Encourage students to work with one another to maintain the collaborative classroom culture more easily established during unplugged activities.

"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.

Teaching and Learning Strategies

The teaching and learning strategies listed below are generally useful across different lessons and units. We believe these strategies lead to positive classroom culture and, ultimately, student learning.

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 Discoveries, there are many lessons on the computer (plugged lessons) during which students develop programming skills through online progressions. Pair programming can help to foster a sense of camaraderie and collaboration in your classroom during sets of plugged lessons. 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 our support article on pair programming: bit.ly/pair-programming-support

Videos:

- For Teachers: bit.ly/pair-programming- teacher (Created for CS Fundamentals, but still applicable)
- For Students: bit.ly/pair-programming- student

Other Resources:

- The ETR Pair Programming Toolkit provides additional strategies for implementing Pair Programming in your classroom. Check it out at: http://bit.ly/pair-programming-toolkit
- The National Center for Women & Information Technology (NCWIT) has a great resource about the benefits of pair programming. Check it out at: http://bit.ly/pair-programming-ncwit

Think-Pair-Share

What is it?

Think-Pair-Share is a three part activity in which students are presented with a problem or task to work on.

Think: First, students work individually. Working individually gives students the opportunity to collect their thoughts before communicating them with others. They should write down their thoughts in a journal for later sharing.

Pair: Once students have had time to work individually, they then enter the "Pair" stage in which they work with a small group. These groups can consist of two or three students. The group discusses the thoughts each member collected during the "Think" stage. The goal is for students to engage in a low-risk discussion where they get a chance to share their ideas with others. This activity is especially useful in the early stages of developing collaborative skills such as attentive listening to a partner.

Share: Finally, the groups share out some of the ideas they discussed to the whole class, and the discussion will continue as needed in the whole group setting. This allows major ideas to bubble up to the whole group, where everyone can hear and benefit from them.

How does it connect to the curriculum?

Almost every lesson in the CS Discoveries curriculum involves some kind of discussion that uses a version of Think-Pair-Share. It is one of the most common practices used for warm ups and wrap ups. Think-Pair-Share is used for these discussions as it gives students time to think on their own and engage with the content before talking to someone else. When students talk to their partner, it should be a low risk environment to try out an idea. It also allows everyone to play a part in the discussion, even if they don't like talking in the whole class environment.

How do I use it?

- Whenever you are given a prompt, consider giving students time to work individually and then with a partner before bringing the discussion or creation to the whole class.
- View Think-Pair-Share as a way for the class to learn from each other as much as possible. As the lead learner, you should direct the conversation without giving away answers or cutting off the conversation too early.

Peer Feedback

What is it?
Peer feedback is the practice where students give each other feedback on work they have done. The feedback is meant to provide opportunities for students to learn from each other, both by seeing ways others approached the same problem and by incorporating feedback to improve their own work.

How does it connect to the curriculum?
Throughout the CS Discoveries curriculum, there are many activities that have structured moments for students to give each other peer feedback. We support these activities with structured guides for the peer feedback process. Many of the guides follow a similar format where students are first given the opportunity to express what they would like feedback on. Then the peer reviewer gives feedback on some standard questions, often related to the goals of the work, as well as leaving some free response feedback using the sentence starters “I like,” “I wish,” and “What if.” Finally, students are encouraged to reflect on the feedback they received and think about ways to incorporate it in the future.

How do I use it?
- Create a structured peer feedback process.
- Decide who is giving feedback to whom.
- 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.

Debugging

What is it?

Debugging is the act of finding and fixing problems in code. It's a major part of programming and a critical skill for students to develop. Your role as the teacher is to avoid directly debugging for your students, but to help guide them in the development of their own debugging skills.

How does it connect to the curriculum?

Occasionally students will encounter a debugging level where they are explicitly asked to identify and fix bugs in provided code. While these are great opportunities to highlight specific kinds of errors and misconceptions, 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?

- 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.
- Have students read their code aloud, line by line, explaining the purpose of each command.
- Encourage students to ask aloud the same questions that you have been asking them.
- Avoid finding the bug for students or being too specific with your questioning.
- Celebrate discovering (and fixing) new types of bugs to normalize the debugging process.

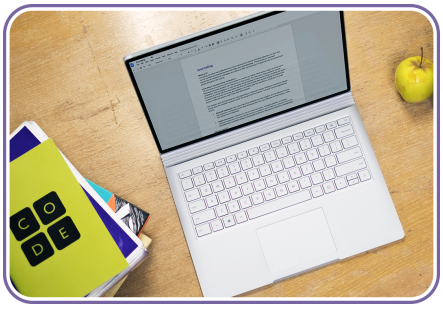
Resources

See Appendix C for a more detailed teacher facing guide to debugging along with a student facing debugging resource.

Journaling

What is it?

Journaling can take many different forms, but in general it is 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 reflect on their own learning in a personal way and record their growth throughout the course.



How does it connect to the curriculum?

CS Discoveries frequently provides opportunities for student journaling, often as a tool for individual sensemaking after having completed an activity as a class. When students are asked to journal, it is done with the assumption that they will have access to their journal writings throughout the course as a tool for review and reflection. Occasionally students are also asked to revisit specific journal prompts. The medium used for journaling can vary, depending on classroom needs. Whichever 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 Google Docs, 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?

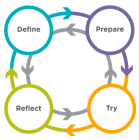
- Provide students a journal at the beginning of the school year.
- Prompt students to journal about specific challenges or bugs they encounter.
- Give students time to revisit previous journal entries and reflect on their growth.

Student Practices

The work students do in CS Discoveries is connected by a core set of practices developed over time. These student practices provide coherence and represent the high-level skills and dispositions students develop throughout the course. In the curriculum you will find reminders of moments when students can reflect on this development, and all major projects include an opportunity for student reflection on their growth in each practice.

Problem Solving	Use a structured problem solving process to help address new problems View challenges as solvable problems Break down larger problems into smaller components
Persistence	Expect and value mistakes as a natural and productive part of problem solving Continue working towards solutions in spite of setbacks Iterate and continue to improve partial solutions
Creativity	Incorporate personal interests and ideas into activities and projects Experiment with new ideas and consider multiple possible approaches Extend or build upon the ideas and projects of others
Collaboration	Work with others to develop solutions that incorporate all contributors Mediate disagreements and help teammates agree on a common solution Actively contribute to the success of group projects
Communication	Structure work so that it can be easily understood by others Consider the perspective and background of your audience when presenting your work Provide and accept constructive feedback in order to improve your work

Problem Solving Process



The Problem Solving Process is a tool for structured problem solving that is woven throughout the entire course to promote student growth and development. Having a strategy for approaching problems can help you develop new insights and come up with new and better solutions. This is an iterative process that is broadly useful for solving all kinds of problems:

Define

- Determine the problem you are trying to solve
- Identify your constraints
- Describe what success will look

Prepare

- Brainstorm / research possible solutions
- Compare pros and cons
- Make a plan

Try

- Put your plan into action

Reflect

- Compare your results to the goals you set while defining the problem
- Decide what you can learn from this or do better next time
- Identify any new problems you have discovered

Each unit leverages the problem solving process in a different way.

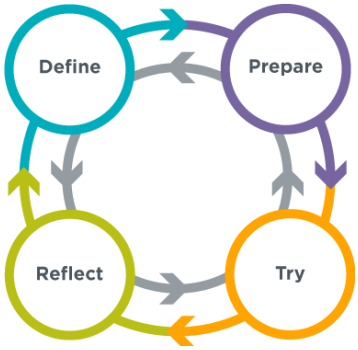
- **The Problem Solving Process** is used in the Problem Solving and Computing unit and is the template for each of the other processes below
- **The Problem Solving Process with Programming** is used in the Web Development, Animations and Games, and Physical Computing units
- **The Problem Solving Process for Design** is used in The Design Process unit
- **The Problem Solving Process with Data** is used in the Data and Society and AI and Machine Learning units

More information about each process is on the following pages

The Problem Solving Process

This process is used in the Problem Solving and Computing unit

Having a strategy for approaching problems can help you develop new insights and come up with new and better solutions. This is an iterative process that is broadly useful for solving all kinds of problems.



Define

- Determine the problem are you trying to solve
- Identify your constraints
- Describe what success will look like

Prepare

- Brainstorm / research possible solutions
- Compare pros and cons
- Make a plan

Try

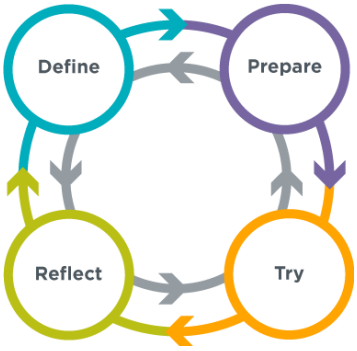
- Put your plan into action

Reflect

- Compare your results to the goals you set while defining the problem
- Decide what you can learn from this or do better next time
- Identify any new problems you have discovered

The Problem Solving Process with Programming

This process is used in the Web Development, Animations and Games, and Physical Computing units



Define

- Read the instructions carefully to ensure you understand the goals
- Rephrase the problem in your own words
- Identify any new skills you are being asked to apply
- Look for other problems you've solved that are similar to this one
- If there is starter code, read it to understand what it does

Prepare

- Write out an idea in plain English or pseudocode
- Sketch out your idea on paper
- List what you already know how to do and what you don't yet
- Describe your idea to a classmate
- Review similar programs that you've written in the past

Try

- Write one small piece at a time
- Test your program often
- Use comments to document what your code does
- Apply appropriate debugging strategies
- Go back to previous steps if you get stuck or don't know whether you've solved the problem

Reflect

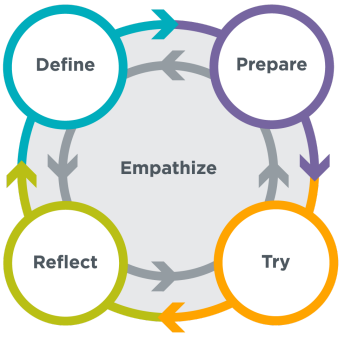
- Compare your finished program to the defined problem to make sure you've solved all aspects of the problem
- Ask a classmate to try your program and note places where they struggle or exhibit confusion
- Ask a classmate to read your code to make sure that your documentation is clear and accurate
- Try to "break" your program to find types of interactions or input that you could handle better
- Identify a few incremental changes that you could make in the next iteration

The Problem Solving Process with Design

This process is used in The Design Process unit

Define

- Identify potential users
- Interview users
- Read user profiles
- Identify needs and wants



Prepare

- Connect needs and wants to specific problems
- Research how others have addressed these issues
- Brainstorm potential solutions
- Discuss pros and cons
- Identify the minimum work need to test your assumptions

Try

- Draw your product on paper
- Develop a low fidelity prototype to communicate your design
- Share prototypes with potential end users for feedback

Reflect

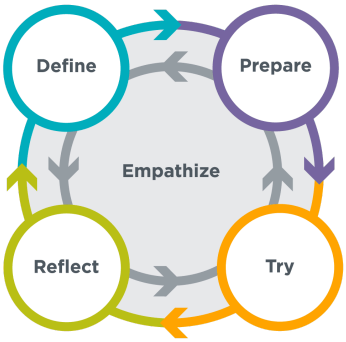
- Present to stakeholders
- Review user feedback

The Problem Solving Process with Data

This process is used in the Data and Society and AI and Machine Learning units.

Define

- Decide what problem you are trying to solve or what question you are trying to answer
- Make sure you understand your target audience (it could be you!) and what specifically it needs
- Identify the parts of your problem you could address with data, and how more information could help



Prepare

- Decide what kinds of data you will collect
- Decide how you will collect the data and in which format you will collect it
- Anticipate possible challenges in data collection and change your plan to account for them
- Develop a plan for how you will analyze your data and make sure your data will be useful for that kind of analysis

Try

- Collect your data using the plan you created
- Clean your data by removing errors, unexpected values, and inconsistencies
- Visualize the data by creating tables, graphs, or charts that help you see broad trends in your data
- Interpret the trends and patterns in your visualizations based on your knowledge of the problem

Reflect

- Review what you've learned about your question or problem
- Decide if what you've learned has solved your problem and allows you to make a decision, or if you'll need to go back to one of the previous steps

Assessment and Feedback

Frequent assessment and feedback are critical to ensuring that students are actively involved in their own learning and that teachers have evidence that their class is making progress. We have incorporated opportunities for both formal and informal evaluation throughout the CS Discoveries curriculum to support you in measuring student growth and informing the pace of your instruction. However, since schools have diverse grading systems, it is up to you to decide how to use the assessment resources for grading purposes.

Each unit includes a learning framework document that outlines the expected student outcomes assessed throughout the unit and within that unit's major projects and post-project test. These outcomes are organized into concept clusters to help students and teachers understand the broad goals of each unit, and, when taken together, address the Big Questions included in the unit's description. Learning objectives and assessment opportunities listed in individual lesson plans connect back to the unit's overall framework.

Specific student outcomes within these learning frameworks may be mapped to external standards such as the CSTA K-12 Computer Science Standards or other similar state-level standards for computer science.

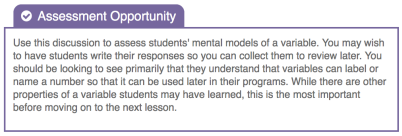
To allow students and teachers flexibility in reaching these learning goals, the course does not assess completion of specific activities.

Opportunities for assessment and feedback

We believe that teachers in classrooms are in the best position to assess and give feedback on student performance. In most cases, we expect that teachers evaluate student work using the criteria and rubrics provided in the lesson plans.

Lesson-level Assessment

Opportunities for assessment within each lesson are listed out in the lesson plans themselves. These opportunities may include sections of the activity guides, discussion and reflection questions, or online programming and quick check levels. Guidance for how to use these opportunities to assess student progress is included in each lesson plan.



Teachers have access to student activity throughout the lesson, including input on prediction levels, the code for all programming levels, and reflective activities students may engage in. While teachers might choose to assess and give feedback on student performance in any of these situations, some places have specifically been marked as useful for assessing student progress. Focusing on these areas can free teachers from feeling the need to look at every level a student has completed.

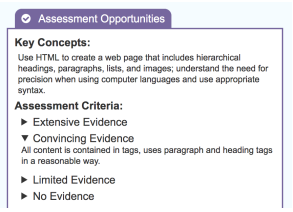
Class discussions provide an opportunity for group sensemaking and for teachers to informally assess student understanding. These discussions may begin with students writing down their individual thoughts before sharing with a partner or group. The goals of each discussion and how teachers might use them to evaluate learning are included in the lesson plan as callout instructions.

Journal questions allow students to reflect on what they have learned, and what they hope to learn more about. Journal prompts often accompany discussion questions, and guidance for assessment is also included in the lesson plan.

Quick-check levels include multiple choice or short answer questions. These are usually given after students have had a chance to explore a concept. They check for common misunderstandings before students move on to the next lesson or

task. Students are able to get feedback from the system immediately, and revise their answers before moving on to the next task. Each quick-check level includes teacher notes detailing the learning objective being assessed.

Programming levels challenge students to complete a small programming task. Teachers have access to all student work in these levels, can read and run the code that students have produced, and can leave feedback for students about their work. These levels also include exemplar solutions for teachers to reference and a mini-rubric that provides assessment criteria and the learning objective being assessed.

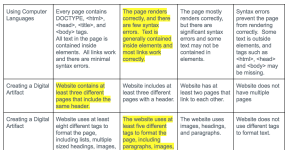


Activity Guides accompany unplugged lessons in the curriculum. They include prompts and questions that teachers can use to follow students' progress through the lesson and reflection questions that can give insight into what students have learned from the activity. Guidance for assessment is included in the lesson plan and any related exemplars.

Chapter- and Unit-level Assessment

End of chapter projects incorporate the skills and understandings students have developed while progressing through that chapter's lessons. These projects are designed to assess unit-specific skills and the five student practices that thread through the entire course. There is broad guidance for the activity, but project implementation leaves room for students to put their personal stamp on the creation.

Student-facing rubrics give guidance on the skills they must demonstrate, while allowing for plenty of choice in how to show what they have learned. These rubrics are directly related to the expected student outcomes listed in the learning framework and give criteria for evaluation of the various objectives of the unit.



Using Computer Languages	Every page contains DOCTYPE, <html>, <head>, <title>, and <body> tags. All text in the page is contained inside elements. All links work and there are minimal syntax errors.	The page renders correctly, and there are few syntax errors. Text is generally contained inside elements and most links work correctly.	The page mostly renders correctly, but there are significant syntax errors and some text may not be contained in elements.	Syntax errors prevent the page from rendering correctly. Some text is outside elements, and tags such as <html>, <head> and <body> may be missing.
Creating a Digital Artifact	Website contains at least three different pages that include the same header.	Website includes at least three different pages with a header.	Website has at least two pages that link to each other.	Website does not have multiple pages
Creating a Digital Artifact	Website uses at least eight different tags to format the page, including lists, multiple sized headings, images,	The website uses all basic semantic-oriented tags to format the page, including paragraphs, images,	The website uses images, headings, and paragraphs.	Website does not use different tags to format text.

Teacher-facing sample projects and associated marked rubrics provide guidance in how student work can be assessed. The sample projects demonstrate varied levels of mastery of the different skills assessed by the project, to allow teachers to compare their own student work to the marked exemplars.

Practice reflections are more open ended, with space for students and teachers to reflect on how the five student practices played into the student experience with the project. We've intentionally made these reflections the same throughout all projects in the course, so that students and teachers can track progress over time.

Post-project tests are included at the end of every unit. These include several multiple choice and matching questions as well as open ended reflections on the final project of the unit. These tests are aligned to the learning framework of each unit and are designed to assess parts of the framework that may not have been covered by the project rubrics.

Individual assessment of group activities

Many CS Discoveries activities are designed to be completed in pairs or small groups. While group work is essential to help students develop the collaborative skills that are a key focus of the course, teachers may find it difficult to assess each individual student's learning. Consider using reflection questions that dig into both the student's role in the project and in the student's understandings and skills for more insight into individual learning outcomes.