Course E

Created with fourth grade students in mind, this course begins with a brief review of concepts previously taught in courses C and D. This introduction is intended to inspire beginners and remind the experts of the wonders of computer science. Students will practice coding with algorithms, loops, conditionals, and events before they are introduced to functions. At the end of the course, students will have the opportunity to create a capstone project that they can proudly share with peers and loved ones.

Journaling

The lessons in this course include journaling prompts. Journals are also useful as scratch paper for building, debugging, and strategizing. Journals can become a fantastic resource for referencing previous answers when struggling with more complex problems.

Think Spot Journal Student Handout

Debugging

From beginners to professionals, debugging is an essential yet often underrated practice. It is likely that your students will find most of their "coding" time is actually spent fixing bugs! To encourage students to take ownership of this practice, we provide this handy reference they can use while coding. Please consult the "Debugging" section of our CS Fundamentals Curriculum Guide for more information on this, as well as other debugging facilitation strategies for your classroom.

Debugging Guide Student Handout

Chapter 1: Ramp Up

Lesson 1: Sequencing in the Maze

Skill Building | Ramp Up

In this lesson, you will learn how to write your very own programs!

Lesson 2: Drawing with Loops

Skill Building | Ramp Up

In this lesson, loops make it easy to make even cooler images with Artist!

Lesson 3: Conditionals in Minecraft: Voyage Aquatic

Skill Building | Ramp Up

Here you will learn about conditionals in the world of Minecraft.

Lesson 4: Conditionals with the Farmer

Skill Building | Ramp Up

You will get to tell the computer what to do under certain conditions in this fun and challenging series.

Chapter Commentary

Ramp Up

Chapter 2: Sprites

Lesson 5: Simon Says

Unplugged | Behaviors

Play a game and think about what commands are needed to get the right result.

Lesson 6: Swimming Fish with Sprite Lab

Skill Building | Sprites

Learn how to create and edit sprites.

Lesson 7: Alien Dance Party with Sprite Lab

Skill Building | Sprites

Create an interactive project that can be shared with classmates.

Chapter Commentary

Sprites

Chapter 3: Digital Citizenship

Lesson 8: Private and Personal Information

Unplugged | Online Safety

The internet is fun and exciting, but it's important to stay safe too. This lesson teaches you the difference between information that is safe to share and information that is private.

Lesson 9: About Me with Sprite Lab

Application | Online Safety

By creating an interactive poster with SpriteLab, students will apply their understanding of sharing personal and private information on the web.

Chapter Commentary

Digital Citizenship

Chapter 4: Impacts of Computing

Lesson 10: Designing for Accessibility

Unplugged | Impacts of Computing

In this lesson, students will learn about accessibility and the value of empathy through brainstorming and designing accessible solutions for hypothetical apps.

Chapter Commentary

Impacts of Computing

Chapter 5: Nested Loops

Lesson 11: Nested Loops in Maze

Skill Building | Nested Loops

Loops inside loops inside loops. What does this mean? This lesson will teach you what happens when you place a loop inside another loop.

Lesson 12: Fancy Shapes using Nested Loops

Skill Building | Nested Loops

More nested loops! This time, you get to make some AMAZING drawing with nested loops.

Lesson 13: Nested Loops with Frozen

Application | Nested Loops

Anna and Elsa have excellent ice-skating skills, but need your help to create patterns in the ice. Use nested loops to create something super COOL.

Chapter Commentary

Nested Loops

Chapter 6: Functions

Lesson 14: Songwriting

Unplugged | Functions

Even rockstars need programming skills. This lesson will teach you about functions using lyrics from songs.

Lesson 15: Functions in Minecraft

Skill Building | Functions

Can you figure out how to use functions for the most efficient code?

Lesson 16: Functions with Harvester

Skill Building | Functions

Functions will save you lots of work as you help the farmer with her harvest!

Lesson 17: Functions with Artist

Skill Building | Functions

Make complex drawings more easily with functions!

Chapter Commentary

Functions

Chapter 7: End of Course Project

Lesson 18: End of Course Project

End of Course Project

Projects this big take time and plenty of planning. Find your inspiration, develop a plan, and unleash your creativity!

Chapter Commentary

End of Course Project

If you are interested in licensing Code.org materials for commercial purposes,**contact us**.

Lesson 1: Sequencing in the Maze

Overview

In this set of puzzles, students will begin with an introduction (or review depending on the experience of your class) of Code.org's online workspace. There will be videos pointing out the basic functionality of the workspace including the Run , Reset , and Step buttons. Also discussed in these videos: dragging Blockly blocks, deleting Blockly blocks, and connecting Blockly blocks. Next, students will practice their **sequencing** and **debugging** skills in the maze. Debugging is an essential element of learning to program. Students will encounter some puzzles that have been solved incorrectly. They will need to step through the existing code to identify errors, including incorrect loops, missing blocks, extra blocks, and blocks that are out of order.

Purpose

We recognize that every classroom has a spectrum of understanding for every subject. Some students in your class may be computer wizards, while others haven't had much experience at all. In order to create an equal playing (and learning) field, we have developed these ramp-up lessons. This can be used as either an introduction or a review of how to use Code.org and basic computer science concepts. Students in your class might become frustrated with this lesson because of the essence of debugging. **Debugging** is a concept that is very important to computer programming. Computer scientists have to get really good at facing the bugs in their own programs. Debugging forces the students to recognize problems and overcome them while building critical thinking and problem solving skills.

Agenda

Warm Up (15 min)
> **Introduction**
> **Vocabulary**

Main Activity (30 min)
> **Online Puzzles**

Wrap Up (5 - 10 min)
> **Journaling**

Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Order movement commands as sequential steps in a program.
- Modify an existing program to solve errors.
- Break down a long sequence of instructions into the largest repeatable sequence.
- Predict where a program will fail.
- Modify an existing program to solve errors.
- Reflect on the debugging process in an age-appropriate way.

Preparation

☐ Play through the puzzles yourself to find any potential problem areas for your class.
☐ (Optional) Pick a couple of puzzles to do as a group with your class.
☐ Make sure every student has a journal.
☐ Review **Debugging Recipe - Student Handout** with the class.

Links

> **Heads Up!** Please make a copy of any documents you plan to share with students.

For the Students

- **Unplugged Blocks (Courses C-F)** - Manipulatives Make a Copy ▾
- **Debugging Recipe** - Student Handout Make a Copy ▾

Vocabulary

- **Bug** - Part of a program that does not work correctly.
- **Debugging** - Finding and fixing problems in an algorithm or program.
- **Program** - An algorithm that has been coded into something that can be run by a machine.

- **Programming** - The art of creating a program.

Teaching Guide

Warm Up (15 min)

Introduction

Ask students to think about problems they have to solve in everyday life.

* How do you fix something that isn't working?
* Do you follow a specific series of steps?
* Some puzzles in this lesson have already been solved for you (yay!), but they don't seem to be working (boo!)
* We call the problems in these programs "bugs," and it will be your job to "debug" them.

Students will either be learning a lot of new concepts or reviewing a lot of basic concepts. Based on your class's experience, you can cover the following vocabulary. We recommend using the following words in sentences if the definitions aren't explicitly covered.

Vocabulary

This lesson has four new and important vocabulary words:

* **Program** - Say it with me: Pro - Gram An algorithm that has been coded into something that can be run by a machine.

* **Programming** - Say it with me: Pro - Gramm - ing The art of creating a program.

* **Bug** - Say it with me: Bug An error in a program that prevents the program from running as expected.

* **Debugging** - Say it with me: De - Bugg - ing Finding and fixing errors in programs.

Say:

Debugging is a process. First, you must recognize that there is an error in your program. You then work through the program step by step to find the error. Try the first step, did it work? Then the second, how about now? If you make sure that everything is working line by line, then when you get to the place that your code isn't doing what it's supposed to, you know that you've found a bug. Once you've discovered your bug, you can work to fix (or "debug") it!

If you think it will build excitement in the class you can introduce the character of today's puzzles, Scrat from Ice Age. If students aren't familiar with Scrat, **show some videos** of the quirky squirrel running into trouble.

Main Activity (30 min)

Online Puzzles

Teachers play a vital role in computer science education and supporting a collaborative and vibrant classroom environment. During online activities, the role of the teacher is primarily one of encouragement and support. Online lessons are meant to be student-centered, so teachers should avoid stepping in when students get stuck. Some ideas on how to do this are:

> 💡 Teacher Tip:
>
> Show the students the right way to help classmates:
> * Don't sit in the classmate's chair
> * Don't use the classmate's keyboard
> * Don't touch the classmate's mouse
> * Make sure the classmate can describe the solution to you out loud before you walk away

* Utilize **Pair Programming - Student Video** whenever possible during the activity.
* Encourage students with questions/challenges to start by asking their partner.
* Unanswered questions can be escalated to a nearby group, who might already know the solution.
* Remind students to use the debugging process before you approach.
* Have students describe the problem that they're seeing. What is it supposed to do? What does it do? What does that tell

you?
- Remind frustrated students that frustration is a step on the path to learning, and that persistence will pay off.
- If a student is still stuck after all of this, ask leading questions to get the student to spot an error on their own.

Before letting the students start on the computer, remind them of the advantages of**Pair Programming - Student Video** and asking their peers for help. Sit students in pairs and recommend they ask at least two peers for help before they come to a teacher.

🖥 Code Studio levels

Maze Intro: Programming with Blocks   ■◀ 1   *(click tabs to see student view)*

Practice   🖥 2   🖥 3   🖥 4   *(click tabs to see student view)*

Debugging with the Step Button   ■◀ 5   *(click tabs to see student view)*

Practice   🖥 6   🖥 7   🖥 8   🖥 9   *(click tabs to see student view)*

Challenge   🖥 10   *(click tabs to see student view)*

Practice   🖥 11   🖥 12   *(click tabs to see student view)*

As mentioned in the purpose of this lesson, make sure the students are aware that they will face frustrating puzzles. Tell them it is okay to feel frustrated, but it is important to work through the problem and ask for help. As the students work through the puzzles, walk around to make sure no student is feeling so stuck that they aren't willing to continue anymore.

Wrap Up (5 - 10 min)

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- What kind of bugs did you find today?
- Draw a bug you encountered in one of the puzzles today. What did you do to "debug" the program?

Extended Learning

Use these activities to enhance student learning. They can be used as outside of class activities or other enrichment.

Planting bugs

Have students go back through previous levels, purposefully adding bugs to their solutions. They can then ask other students to debug their work. This can also be done with paper puzzles.

When other students are debugging, make sure that the criticisms are constructive. If this could be a problem for your class, go over respectful debugging before this activity by role playing with another student.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Lesson 2: Drawing with Loops

Overview

Watch student faces light up as they make their own gorgeous designs using a small number of blocks and digital stickers! This lesson builds on the understanding of loops from previous lessons and gives students a chance to be truly creative. This activity is fantastic for producing artifacts for portfolios or parent/teacher conferences.

Purpose

This series highlights the power of loops with creative and personal designs.

Offered as a project-backed sequence, this progression will allow students to build on top of their own work and create amazing artifacts.

Agenda

Warm Up (15 min)
    Introduction
Main Activity (30 min)
    Online Puzzles
Wrap Up (15 min)
    Journaling

View on Code Studio

Objectives

Students will be able to:

- Identify the benefits of using a loop structure instead of manual repetition.
- Differentiate between commands that need to be repeated in loops and commands that should be used on their own.

Preparation

☐ Play through the puzzles to find any potential problem areas for your class.
☐ Make sure every student has a journal.

Vocabulary

- **Loop** - The action of doing something over and over again.
- **Repeat** - Do something again

Teaching Guide

Warm Up (15 min)

Introduction

Students should have had plenty of introduction to loops at this point. Based on what you think your class could benefit from, we recommend:

- Creating a new stack design with loops just like in "My Loopy Robotic Friends"
- Previewing a puzzle from this lesson

All of these options will either review loops or the artist, which will help prepare your class for fun with the online puzzles!

Main Activity (30 min)

Online Puzzles

🖥 Code Studio levels

Artist Intro with JR Hildebrande 🎥 1 *(click tabs to see student view)*

Practice 🖥 2 🖥 3 *(click tabs to see student view)*

Loops with the Artist 🎥 4 *(click tabs to see student view)*

Practice 🖥 5 🖥 6 *(click tabs to see student view)*

Challenge 🖥 7 *(click tabs to see student view)*

Practice 🖥 8 🖥 9 🖥 10 *(click tabs to see student view)*

Challenge 🖥 11 *(click tabs to see student view)*

Levels 🖥 Extra 🖥 Extra *(click tabs to see student view)*

Some students may discover where to add repeat loops by writing out the program without loops then circling sections of repetitions. If the students in your class seem like they could benefit from this, have them keep paper and pencils beside them at their machines. Students might also enjoy drawing some of the shapes and figures on paper before they program it online. (When drawing stamps, it can be easier to symbolize those with simple shapes like circles and squares.)

Wrap Up (15 min)

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- What was the coolest shape or figure you programmed today? Draw it out!
- What is another shape or figure you would like to program? Can you come up with the code to create it?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

If you are interested in licensing Code.org materials for commercial purposes,**contact us**.

Lesson 3: Conditionals in Minecraft: Voyage Aquatic

Overview

This lesson was originally created for the Hour of Code, alongside the Minecraft team. Students will get the chance to practice ideas that they have learned up to this point, as well as getting a sneak peek at conditionals!

Purpose

This set of puzzles will work to solidify and build on the knowledge of loops, and introduce conditionals. By pairing these two concepts together, students will be able to explore the potential for creating fun and innovative programs in a new and exciting environment.

Agenda

Warm Up (15 min)
> **Introduction**

Main Activity (30 min)
> **Online Puzzles**

Wrap Up (15 min)
> **Journaling**

Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Define circumstances when certain parts of a program should run and when they shouldn't.
- Determine whether a conditional is met based on criteria.

Preparation

☐ Play through the puzzles associated with this lesson to find any potential problem areas for your class.

☐ Make sure every student has a journal.

Vocabulary

- **Condition** - A statement that a program checks to see if it is true or false. If true, an action is taken. Otherwise, the action is ignored.
- **Conditionals** - Statements that only run under certain conditions.

Teaching Guide

Warm Up (15 min)

Introduction

Gather the class together and ask two volunteers to walk straight in some direction in the classroom. If they encounter a chair out of place, they must step over it. If they reach a wall, they must sit down.

Once all of the students are sitting down, ask how you would program a robot to respond to a wall or a chair. Remind students that you cannot simply say "Step over chair" unless you know there is a chair, and you will not always know there is a chair. It might be helpful to translate the task into instructions like:

- while there is a path ahead
    - walk forward
    - if there is a chair, step over it
- sit down

Tell students they will be using **conditionals** during this lesson. Give the definition of:

- **Condition**: A statement that a program checks to see if it is true or false. If true, an action is taken. Otherwise, the action is ignored.
- **Conditionals**: Statements that only run under certain conditions.

Open up a discussion of when you might use a conditional in your code.

Main Activity (30 min)

Online Puzzles

🖥 Code Studio levels

Practice	🖥 1	🖥 2	🖥 3	🖥 4	🖥 5	🖥 6	🖥 7	🖥 8	🖥 9	🖥 10

🖥 11 *(click tabs to see student view)*

Students are in for a real treat with this lesson. It's likely most of your students have heard of Minecraft, but give a brief introduction for those that may not know.

Minecraft is a game of cubes. You can play as Alex or Steve as you work through mazes. You'll need to apick up items, and explore in a world made up of cubes of things.

Demonstrate one of the puzzles to the class (we recommend puzzle 11.) Once all questions have been addressed, transition students to computers and let them start pair programming.

Wrap Up (15 min)

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- Draw a feeling face to show how you felt during today's lesson.
- Draw something else you could have built in this minecraft world.

- Can you draw a scene where someone is using a conditional?

Extended Learning

More Minecraft

If you find that your class really enjoys the Minecraft environment,**here are some links to other Minecraft games they can play online**. These games will also teach basic coding skills.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

If you are interested in licensing Code.org materials for commercial purposes,**contact us**.

Lesson 4: Conditionals with the Farmer

Overview

This lesson introduces students to while loops and if / else statements. **While loops** are loops that continue to repeat commands as long as a condition is true. While loops are used when the programmer doesn't know the exact number of times the commands need to be repeated, but the programmer does know what condition needs to be true in order for the loop to continue looping. If / Else statements offer flexibility in programming by running entire sections of code only if something is true, otherwise it runs something else.

Purpose

A basic understanding of conditionals is a recommended prerequisite for Course E. We created this introduction to give a review for the students already familiar to conditionals and allow practice for the students that are just learning. If you find that the understanding of conditionals varies widely in your classroom, we recommend a strategic pairing of students when completing this online lesson.

Agenda

Warm Up (15 min)
  Introduction
Main Activity (30 min)
  Online Puzzles
Wrap Up (15 min)
  Journaling
Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Define circumstances when certain parts of a program should run and when they shouldn't.
- Determine whether a conditional is met based on criteria.

Preparation

☐ Play through the puzzles to find any potential problem areas for your class.
☐ Make sure ever student has a journal.

Vocabulary

- **Condition** - A statement that a program checks to see if it is true or false. If true, an action is taken. Otherwise, the action is ignored.
- **Conditionals** - Statements that only run under certain conditions.
- **While Loop** - A loop that continues to repeat while a condition is true.

Teaching Guide

Warm Up (15 min)

Introduction

Gather the class together and ask two volunteers to walk straight in some direction in the classroom. If they encounter a chair out of place, they must step over it. If they reach a wall, they must sit down.

Once all of the students are sitting down, ask how you would program a robot to respond to a wall or a chair. Remind students that you cannot simply say "Step over chair" unless you know there is a chair, and you will not always know there is a chair. It might be helpful to translate the task into instructions like:

- while there is a path ahead
  - walk forward
  - if there is a chair, step over it
- sit down

Tell students they will be using **conditionals** to solve this problem on Code.org. Give the definition of:

- **Condition**: A statement that a program checks to see if it is true or false. If true, an action is taken. Otherwise, the action is ignored.
- **Conditionals**: Statements that only run under certain conditions.

Open up a discussion of when you might use a conditional in your code.

Main Activity (30 min)

Online Puzzles

🖥 Code Studio levels

While Loops with the Farmer | 🎥 1 | *(click tabs to see student view)*

Prediction | 🖥 2 | *(click tabs to see student view)*

Practice | 🖥 3 | | 🖥 4 | | 🖥 5 | | 🖥 6 | *(click tabs to see student view)*

Repeat Unit Statements | 🎥 7 | *(click tabs to see student view)*

Practice | 🖥 8 | | 🖥 9 | | 🖥 10 | *(click tabs to see student view)*

"If/Else" with the Bee | 🎥 11 | *(click tabs to see student view)*

Challenge | 🖥 12 | *(click tabs to see student view)*

The patterns in these puzzles may not be obvious to every student. We recommend that you play through these levels beforehand to best understand any problem areas for your class. Also, watching and using the techniques from **Pair Programming - Student Video** may be helpful for your class.

Wrap Up (15 min)

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- What was today's lesson about?
- How do you feel about today's lesson?
- What is a conditional? Why would you program a conditional?
- Give an example of using a conditional during your day? (ex.**If** I am hungry, I eat some food;**While** I am walking across the street, I keep an eye out for cars)

Extended Learning

While We Play

Gather the class for some fun outside or in a gym with a ball! This could be done in a circle or as a team in a court
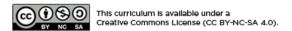
Rules:

- **While** the ball is in play, we must all be ready to hit it
- **If** the ball is hit to you, you must keep it in the air
- **If** you hit the ball once, you cannot hit it again (only one touch per person per turn, no double hitting)
- **If** the ball goes out of bound, every student must fall to the ground dramatically. The last kid to fall has to retrieve the ball.

At the end of the first round, ask the students if they can identify the conditionals in the game. Can they come up with others they might want in the game?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Lesson 5: Simon Says

Overview

In this lesson, students will play a game intended to get them thinking about the way commands need to be given to produce the right result. This will help them more easily carry over to Sprite Lab in the upcoming lessons.

Purpose

This lesson is designed to prepare students to think about one of the core programming concepts in Sprite Lab, behaviors.

Agenda

Warm Up (10 min)
> **Introduction**

Main Activity (20 min)
> **Simon Says**
> **Extension**

Wrap Up (15 min)
> **Reflection**

View on Code Studio

Preparation

☐ Play through the puzzles to find any potential problem areas for your class.
☐ Make sure ever student has a journal.

Teaching Guide

Warm Up (10 min)

Introduction

💬 **Think-Pair-Share:** Think of hobby, sport, or activity that you know how to do well. What does it take to do it well? Are there times when you need think about multiple things or perform different actions at the same time?

- What parts of your body are you using? Does this change at different points?
- Are there times when you need to stop one action before you can begin another? (For example, you must stop dribbling the basketball before you can shoot.)

🎤 *Remarks*

> Today we're going to play a game where you'll need to keep track of multiple tasks simultaneously. It's a little bit like Simon Says.

💬 Discussion

Goal: Help students think about times when they must keep of multiple tasks or instructions simultaneously. The main activity will get students performing silly actions, either in sequence or simultaneously. This will eventually lead to connections about a new way to write programs, but you don't need to the discussion there yet.

Main Activity (20 min)

Simon Says

🎤 *Remarks*

> In the game Simon Says, one player takes the roll of "Simon" or the leader who gives command to other players. Players must follow the leader's commands if and only if they are prefaced with the phrase "Simon Says". The point of the game is to think quickly and to distinguish between real and fake commands. In this version of the game, the rules are a little different.

Rules

All commands should be prefaced with either "begin" or "stop".

When players are told to "begin" a behavior, they should continue it until told to "stop". For example, players should not told be told to clap once, but might be told to "begin clapping". Their clapping should continue until they are told to "stop clapping" even if they are given other new behaviors in between.

The leader can also call for the players to "stop everything".

This should result in everyone just standing at rest regardless of all previous commands.

You as the teacher should take on the role of the leader. You can try make up your own sequences, but here are some you can try. Be sure to give a little space between commands. For each of these sections, consider running through the entire sequence without any discussion and later repeating it again after everyone has had a chance to debrief and process any confusion.

Basic:

- Begin marching in place.
- Stop marching in place.
- Begin clapping.
- Stop clapping.
- Begin marching in place.
- Begin clapping.
- Stop everything.

Debrief: What happened when you were told to clap but you were already marching in place? What happens if you are told to begin two different behaviors at once?

Intermediate:

- Begin waving your arms in the air.
- Begin bobbing your head.
- Stop waving your arms in the air.
- Stop bobbing your head.
- Begin shaking your knees.
- Begin flapping your arms like a bird.
- Stop shaking your knees.
- Begin bobbing your head.
- Begin marching in place.
- Stop flapping your arms like a bird.
- Stop everything.

Debrief:

What kinds of instructions caused people to make mistakes?

What strategies do you think are helpful for making sure you follow instructions correctly?

Why is it important to keep track of each behavior separately?

Challenging

- Begin crouching.
- Begin tapping your head.
- Stop crouching.
- Stop tapping your head.
- Begin jumping up and down.
- Begin tapping your head.
- Stop everything.
- Begin clapping.
- Begin flapping your arms like a bird.
- Stop everything.
- Begin crouching.
- Begin jumping up and down.
- Stop everything.
- Begin tapping your knees.
- Begin tapping your head.
- Stop everything.
- Begin spinning to the left.
- Begin spinning to the right.
- Stop spinning to the left.
- Stop spinning to the right.

Debrief: What happens if two behaviors seem to conflict with each other?

What should you do when told to clap your hands and flap your arms at the same time?

How can you jump up and down while crouching?

What happens if you need to tap your knees and your head at the same time?

When you were told to spin in two opposite directions what did you see people do? What would happen if you were told to spin left and right at the exact same time?

Extension

If you want to make things even more complicated, you can consider changing the rules so that only some players follow some commands. For example, you could try commands like "All girls begin spinning to the left", "All boys begin clapping your hands", or "Everyone stop everything."

Wrap Up (15 min)

Reflection

Journal: Think back to the activity or hobby you discussed at the beginning of class. Using "begin" and "stop" commands write down the instructions you could give someone if you wanted them to act like they were they were doing it. Be sure to remember when they might need to stop something before beginning something new.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

If you are interested in licensing Code.org materials for commercial purposes,**contact us**.

Lesson 6: Swimming Fish with Sprite Lab

Overview

In this lesson, students will learn about the two concepts at the heart of Sprite Lab: sprites and behaviors. Sprites are characters or objects on the screen that students can move, change, and manipulate. Behaviors are actions that sprites will take continuously until they are stopped.

Purpose

This lesson is designed to introduce students to the core vocabulary of Sprite Lab, and allow them to apply concepts they learned in other environments to this tool. By creating a fish tank, students will begin to form an understanding of the programming model of this tool, and explore ways they can use it to express themselves.

Agenda

Warm Up (10 min)
> **Introduction**

Bridging Activity
> **Swimming Fish Teacher Sandbox**

Main Activity (20 min)
> **Online Puzzles**

Wrap Up (15 min)
> **Journaling**

View on Code Studio

Objectives

Students will be able to:

- Define "sprite" as a character or object on the screen that can be moved and changed.
- Create new sprites and assign them costumes and behaviors.

Preparation

☐ Play through the puzzles to find any potential problem areas for your class.
☐ Make sure every student has a journal.

Vocabulary

- **Behavior** - An action that a sprite performs continuously until it's told to stop.
- **Sprite** - A graphic character on the screen with properties that describe its location, movement, and look.

Introduced Code

- set background color
- make new sprite
- set sprite property

Teaching Guide

Warm Up (10 min)

Introduction

Today students will learn how to work with sprites in Sprite Lab.

Display: Pull up a previous puzzle from Code.org, ideally one containing a "main character" like Scrat from Ice Age or one of the Angry Birds.

Discuss: Let the students know that this character on the screen is a "sprite." It is a graphic that is controlled by a program. In this lesson, students will have the opportunity to choose their own sprites to control.

Bridging Activity

This demostration and discussion can help students make the connection from the previous unplugged lesson "Simon Says" into the new Sprite Lab environment.

Swimming Fish Teacher Sandbox

☛Using a projector, show the sandbox level to your students. The goal is to make connections to the previous lesson and show them some of the unique ways that Sprite Lab works. Model writing a few programs and ask students to share their observations.

- What blocks would we need to connect to make the tumbleweed spin?
- What would happen if we told the sprite to begin two behaviors at once?
- Will the sprite ever stop these behaviors on its own?
- If we want the sprite to stop a behavior when we click it, how might we do that?

Before heading into the Main Activity, introduce or review today's lesson vocabulary.

> 🎓 Content Corner
>
> Sprite Lab works differently in some ways from the other online tools in the course. Most importantly, all code runs in order and immediately unless attached to an event block. Telling a sprite to begin and start the same behavior won't result in any observable effect because there is no time between each action.
>
> If students bring up time, such as making a behavior happen for a set duration, it can be helpful to reframe things by asking "**When** should the behavior stop?" Programs like this will rely on events to make things start and stop as desired.

Main Activity (20 min)

Online Puzzles

Goal: Today, students will be programming their own Fish Tank. They'll begin by learning how to put some sprites on the screen, then they will make them move. Finally, they'll customize their fish tank to add whatever creatures and objects they want.

💡 **Transition:** Move students to their machines. Encourage students to follow the instructions for each puzzle. Help them realize that this is a creative activity, intended to help them learn Sprite Lab. It is not an assessment activity of any sort.

🖥 Code Studio levels

Introducing Sprite Lab 📹 1 *(click tabs to see student view)*

Prediction 🖥 2 *(click tabs to see student view)*

Mini-Project: Swimming Fish 🖥 3 🖥 4 🖥 5 *(click tabs to see student view)*

How to Make a Sprite 📹 6 *(click tabs to see student view)*

Mini-Project: Swimming Fish (continued) 🖥 7 🖥 8

(click tabs to see student view)

Free Play 🖥 9 *(click tabs to see student view)*

Wrap Up (15 min)

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.
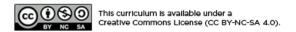
Journal Prompts:

- What was today's lesson about?
- How do you feel about today's lesson?
- How did it feel to make a scene that was more creative?
- Was it difficult to finish a lesson where there was no clear "right" and "wrong"?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Lesson 7: Alien Dance Party with Sprite Lab

Overview

This lesson features Sprite Lab, a platform where students can create their own interactive animations and games. In addition to behaviors, today students will incorporate user input as events to create an "alien dance party".

Purpose

Students will use events to make characters move around the screen, make noises, and change backgrounds based on user input. This lesson offers a great introduction to events in programming and even gives students a chance to show creativity! At the end of the puzzle sequence, students will be presented with the opportunity to share their projects.

Agenda

Warm Up (5 min)
    Introduction
Main Activity (30 min)
    Online Puzzles
Wrap Up (10 min)
    Journaling

View on Code Studio

Objectives

Students will be able to:

- Identify actions that correlate to input events.
- Create an interactive animation using sprites, behaviors, and events.

Preparation

☐ Play through the puzzles to find any potential problem areas for your class.
☐ Make sure every student has a journal.

Vocabulary

- **Event** - An action that causes something to happen.

Introduced Code

- set background
- random location
- location picker
- change color
- remove color
- sprite clicked
- sprite touches sprite

Teaching Guide

Warm Up (5 min)

Introduction

Today students will visit **events** in programming.

Demo: Ask the students to raise their hands in the air.

What you did was declare an event. When you say "raise your hands in the air" the students responded by raising their hands. In coding, you would declare this by saying something like "when I say 'raise your hands,' you raise your hands".

You can also think of cities as declaring events. There are laws that say "when there is a green light, cars move through the intersection".

Discuss: Ask the students why they think this is an event.

Today, students will play in Sprite Lab, but the events they will be working on will be more like the video games they are used to playing. Events will take the form of actions, such as clicking the screen or two characters running into each other.

Display: Begin by showing Puzzle 1 to your students.

Think/Pair: Ask them to predict what will happen when the code is run, and to discuss with their neighbors. Run the code, and discuss the outcome.

Main Activity (30 min)

Online Puzzles

- **Goal:** Today, students will be creating their own alien dance party! They'll begin by reviewing how to put sprites on the screen, then they will assign them behaviors and learn to change those behaviors when an event is initiated.

Transition: Move students to their machines. Encourage students to follow the instructions for each puzzle. Help them realize that this is a creative activity, intended to help them learn Sprite Lab. It is not an assessment activity of any sort.

> 💡 Teaching Tip
>
> Encourage students with questions/challenges to start by asking their partner. Unanswered questions can be escalated to a nearby group, who might already know the solution. Have students describe the problem that they're seeing:
>
> - What is it supposed to do?
> - What does it do?
> - What does that tell you?

🖥 Code Studio levels

Prediction   🖥 1   *(click tabs to see student view)*

Video   🎥 2   *(click tabs to see student view)*

Mini-Project: Alien Dance Moves   🖥 3   🖥 4   🖥 5   🖥 6

(click tabs to see student view)

Mini-Project: Alien Dance Party   🖥 7   🖥 8   🖥 9   *(click tabs to see student view)*

Wrap Up (10 min)

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- What was today's lesson about?
- How do you feel about today's lesson?
- How did it feel to have control over what your characters were able to do?
- Did you change the program in any way to make it feel more like your own?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

If you are interested in licensing Code.org materials for commercial purposes,**contact us**.

Lesson 8: Private and Personal Information

Overview

Developed by Common Sense Education, this lesson is about the difference between information that is safe to share online and information that is not.

As students visit sites that request information about their identities, they learn to adopt a critical inquiry process that empowers them to protect themselves and their families from identity theft. In this lesson, students learn to think critically about the user information that some websites request or require. They learn the difference between private information and personal information, as well as how to distinguish what is safe or unsafe to share online.

Purpose

Common Sense Education has created this lesson to teach kids the importance of security on the internet. By discussing the difference between personal and private information, students will be able to recognize what information should and shouldn't be shared. Students will also learn what signs you should look for to determine if a website is safe or not.

Agenda

Warm Up (5 min)
   Introduction
Main Activity (35 min)
   Log In
   Private and Personal
   What's Safe to Share Online?
Wrap Up (15 min)
   Flash Chat: What did we learn about today?
   Journaling
Assessment (10 min)
   Private and Personal Information

View on Code Studio

Objectives

Students will be able to:

- Learn about the benefits and risks of sharing information online.
- Understand what type of information can put them at risk for identity theft and other scams.

Preparation

☐ Copy the **Protect Yourself Student Handout** (7th page of the **Common Sense Education - Private and Personal Information - Teacher Prep Guide**), one for each student.

☐ Copy the **All About Me Student Handout** (6th page of the **Common Sense Education - Private and Personal Information - Teacher Prep Guide**), one for each student.

☐ Print out an assessment (8th page of the **Common Sense Education - Private and Personal Information - Teacher Prep Guide**) for each student. Teacher version is the page after the student assessment.

☐ Preview websites like **Neopets**, **Nickelodeon**, and **BookAdventure** and prepare to show them to the class.

☐ Review **CSF Digital Citizenship - Resource List** for more online safety content.

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **Common Sense Education - Private and Personal Information** - Teacher Prep Guide
- **Common Sense Education** - Website

- **CSF Digital Citizenship** - Resource List

Vocabulary

- **Identity Theft** - When a thief steals someone's private information in order to pretend to be that person.
- **Personal Information** - Information that can't be used to identify you.
- **Private Information** - Information that can be used to identify you.
- **Register (Online)** - To enter your information in order to sign up and get access to a website.

Teaching Guide

Warm Up (5 min)

Introduction

Ask:

- What types of information do you think are okay to share publicly online such as on an online profile that others will see?
  - Interests and favorite activities
  - Opinions about a movie
  - First name
- What are some examples of websites where you must register in order to participate?
  - Social networking sites
  - Video-sharing sites
  - Youth discussion sites
  - Ask-an-expert sites
  - Game sites

Write the names of the websites on the board. Explain that it's important to know that sharing some kinds of user information can put you and your family's privacy at risk.

Main Activity (35 min)

common sense education®

Log In

Project for the class, or have students go online to **Neopets**, **Nickelodeon**, or **BookAdventure**. **Do not ask the students to sign up for these sites!**

Discuss with the students the kinds of information that each website requires or requests before the users can participate.

Ask:

💡 Teacher Tip

As an offline alternative, print out and copy the website pages that ask for registration and log-in information. Distribute these to the students.

- What information is required? Why do you think it is required?
  - First name, username, password, password hint, gender, the state you live in, parent's permission, etc. This information is required because it helps distinguish one person from another. Or perhaps the website is keeping a record of who uses it.
- What information is optional? Why do you think it is optional?
  - Parent's email, birthday, state, country, gender, etc. This information is likely optional because the website does not require it for payment or to distinguish people. Or perhaps the website wants to keep track of this kind of information.
- Why do you think websites ask for this kind of information?
  - They want to get people to pay in order to use the site, they want to send messages to people who are signing up, or they want to try to sell things to those people.

Point Out that you do not have to fill out fields on websites if they are not required. Required fields are usually marked by an asterisk (*) or are highlighted in red.

Private and Personal

Explain to the students that some kinds of information are generally safe to share on the internet and some are not. However, the information that's considered safe should not be shared one-on-one with people the students don't already know offline.

Define:

- **Personal Information**: Information that can't be used to identify you.
- **Private Information**: Information that is about you and can be used to identify you.

Emphasize that personal information is usually safe to share online. Private information is usually unsafe to share online, meaning students should get permission from a parent or guardian before sharing this kind of information.

Share the following examples of information that is safe or unsafe to share:

SAFE - Personal Information	UNSAFE - Private Information
- Your favorite food - Your opinion (though it shoud be done respectfully) - First name (with permission)	- Mother's maiden name - Social Security number - Your date of birth - Parents' credit card information - Phone number

Ask:

- Why would someone want to steal someone else's identity on the internet?
  - To steal money
  - To do something bad or mean
  - To hide their real identity

Define:

- **Identity Theft**: When a thief steals someone's private information in order to pretend to be that person.

Explain that an identity thief uses private information to pretend to be the person whose identity he or she has stolen. Once the thief has taken someone's identity, he or she can use that person's name to get a driver's license or buy things, even if the person whose identity they stole isn't old enough to do these things! It's often not until much later that people realize their identity has been stolen. Identity thieves may also apply for credit cards in other people's names and run up big bills that they don't pay off. Let students know that identity thieves often target children and teens because they have a clean credit history and their parents are unlikely to be aware that someone is taking on their child's identity.

Emphasize the difference between private information (which can be used to steal your identity) and personal information (which cannot be used to steal your identity). Invite students to answer the following questions (write their answers on the board):

Ask:

- What kinds of private information could an identity thief use to find out and steal your identity?
  - First and last name, postal address, email address, phone numbers, passwords, credit card numbers, Social Security number, mother's maiden name.
- What kinds of personal information could you share about yourself without showing your identity?
  - Your age, gender, how many siblings you have, your favorite music, your favorite food, what pets you have, the name of your pet, your opinion about something.

Explain to students that on the internet, people you interact with could be your friends next door or strangers who live on

the other side of the world. Because it's hard to know the intentions of people who you've never met before, it is best to remain cautious when sharing your information. You wouldn't give strangers your private information in the real world, and you need to be just as careful when you're online.

Remind students how important it is each time they share information online to stop and think: "Am I giving out information that I should keep private?" Point out that it can sometimes be safe to give out some private information. For example, a website might ask for your birth date or email address. But students should always ask their parent or guardian before giving out private information.

Distribute the **Protect Yourself Student Handout** and have students complete the activity. Review the answers as a class.

What's Safe to Share Online?

Distribute the **All About Me Handout**. Have students write down all the personal information they would like to share on a public profile in an online community. Emphasize that even though personal information is safe to share online, it is okay to choose not to share it. Remind students that everything on the list should be safe to share; none of it should be private information that can put their identity at risk.

Encourage students to share their lists with the class.

Ask:

- Is there anything on the lists that could be used by an identity thief? Why?
  - Guide students to explain their answers and encourage them to use the vocabulary terms.

Wrap Up (15 min)

Flash Chat: What did we learn about today?

You can use these questions to assess your students' understanding of the lesson objectives. You may want to ask students to reflect in writing on one of the questions, using a journal or an online blog/wiki.

Ask:

- What is identity theft?
  - Using someone else's private information to pretend to be that person.
- How do personal information and private information differ?
  - Private information, such as a Social Security number, is unsafe to share. It should be kept private so that identity thieves cannot use it. Personal information, such as your favorite food, cannot be used by identity thieves and is safe to share. Even though personal information is usually safe to share online, you might choose not to share this information, and that's fine.
- What would be a good rule for kids about giving out private information?
  - They should not share it online without the permission of a teacher, parent, or guardian.

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- What did you learn in today's lesson?
- How do you feel about today's lesson?
- Give an example of personal information and private information.
- What's a website that you use often? How do you know it is a safe website to use?

Assessment (10 min)

Private and Personal Information

Hand out the assessment to students. Allow students time to complete the assessment. If there is time left over, go over the answers with the students.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **NI** - Networks & the Internet

If you are interested in licensing Code.org materials for commercial purposes,**contact us**.

Lesson 9: About Me with Sprite Lab

Overview

By creating an interactive poster with SpriteLab, students will apply their understanding of sharing personal and private information on the web.

Purpose

This lesson is meant to make the previous lesson on personal and private information personally relevant for students. With SAFE (personal) and UNSAFE (private) examples in mind, students practice safe self-expression on the web, using SpriteLab to fashion their own sprite costumes and generate text.

Agenda

Warm Up (10 min)

    Introduction
    Review of "Personal and Private Information"

Main Activity (30 min)

    Application - Interactive Poster

Wrap Up (15 min)

    Journaling

View on Code Studio

Objectives

Students will be able to:

- Choose what information about themselves is safe to share online.
- Create an interactive computer program that expresses who they are with text and custom images.

Preparation

☐ Play through the puzzles to find and potential problem areas for your class.
☐ Consider making an example project yourself to share with the class.
☐ Make sure every student has a journal.

Links

> **Heads Up!** Please make a copy of any documents you plan to share with students.

For the Teachers

- **Pause and Think Online** - Video

Introduced Code

- print

Teaching Guide

Warm Up (10 min)

Introduction

Today students will apply what they've learned about personal and private information to design an interactive poster about themselves with SpriteLab.

Review of "Personal and Private Information"

Remind students of information that is safe to share online and information that is strictly private.

SAFE - Personal Information	UNSAFE - Private Information
Your favorite food Your opinion *(though it shoud be done respectfully)* First name *(with permission)*	Mother's maiden name Social Security number Your date of birth Parents' credit card information Phone number

Discuss other examples of the two categories above.

Main Activity (30 min)

Application - Interactive Poster

💡 **Goal:** Today, students will be creating their own interactive posters with SpriteLab! They'll begin by working through a few examples centered on "Rikki", a fictional girl who needs their help deciding what information she should share on her poster. Then, after a quick introduction on creating custom sprite costumes, students will be free to create their posters to their liking.

> 💡 Teaching Tip
>
> Before beginning, be sure to review your classroom's policy on appropriate language and behavior. Students will be creating custom text and images for this lesson. This means they should be mindful of concepts covered in the personal and private information lesson, as well as general classroom etiquette. As students work, be sure to verify that their posters do not violate your classroom policies.

Model: Show students the first two or three levels from today's online puzzles. Demonstrate how the new print block works, along with how they can use the Costumes tab to create and edit costumes. If you have already gone through the entire lesson yourself, you can show them your own finished poster for inspiration!

Transition: Move students to their machines. Encourage students to follow the instructions for each puzzle.

Reminder: Remind the students to only share their work with their close friends or family. For more information watch or show the class Pause and Think Online - Video.

🖥 Code Studio levels

Free Play 🖥 1 *(click tabs to see student view)*

Wrap Up (15 min)

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- What was today's lesson about?
- How do you feel about today's lesson?
- What else would you like to add to your poster?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▸ **IC** - Impacts of Computing
- ▸ **NI** - Networks & the Internet

Lesson 10: Designing for Accessibility

Overview

In this lesson, students will learn about accessibility and the value of empathy through brainstorming and designing accessible solutions for hypothetical apps.

Purpose

Through learning about accessibility, students recognize the impacts of computing beyond their own lives. Accessibility might not seem like a relevant CS topic, but creating technology that is accessible for underserved users helps make tech better for everyone else as well.

Agenda

Warm Up (5 min)
    Background and Motivation
Main Activity (35 min)
    Designing for Accessibility Scenarios (10 min)
    App Redesign Activity (25 min)
Wrap Up (5 min)

View on Code Studio

Objectives

Students will be able to:

- In their own words, describe the impact of mobile apps on the modern world.
- Explain why accessibility is an important part of designing an app for users.
- Improve upon an existing app design by addressing the accessibility needs of users.

Preparation

☐ For context, **read about the types of disabilities here**
☐ Read through the speaker notes in the slide deck.
☐ Prepare enough sketching/drawing supplies for all students.

Links

> **Heads Up!** Please make a copy of any documents you plan to share with students.

For the Students
- **Designing for Accessibility** - Slide Deck

Teaching Guide

Warm Up (5 min)

Background and Motivation

Discuss: Ask students to define the term "app" in their own words. What is an app? What kinds of apps are there? Where do we use apps? While discussing, show screenshots or examples of apps on your smartphone or tablet if possible.

Say: Ensure that all students are on the same page regarding this topic by introducing some key terms and concepts.

- In this course we have been learning how to code. Code is used to create computer applications. Games, web browers, media players... the list of applications goes on and on!
- **App** is short for **application**. Usually, apps more specifically refer to applications running on small computers called **mobile devices**.
- Smartphones and tablets are examples of mobile devices.

Discuss: Apps are everywhere! Ask students why they think apps have become so popular. Here are some example explanations:

- Many people can more easily access or afford a mobile device (e.g., smartphone) than a larger computer with a keyboard, mouse, and so forth.
- Apps can address a variety of needs. For example, if I need a ride somewhere, I might try a ride sharing app and get there quickly. If I want to know what movies I should watch, there's probably an app that can help me decide. And of course, if I want to call or text someone on my phone, I use apps for those as well!

Say: Introduce accessibility with the remarks below.

- As apps become used by more and more people, app developers often come across issues that they had not considered, likely because they have never experienced those issues in their own lives.
- Many of these issues are related to **accessibility**. In the world of computer science, accessibility is about creating technology for people with **disabilities**.
- Disabilities include physical, auditory, visual, and many others.
- Disabilities are diverse and impact people in different ways. For example, someone who is slightly visually impaired might simply wear glasses. Someone who is severely visually impaired might be considered "blind" and will need more assistance.

Transition: Tell students you'd like to introduce them to a few kids who, even though they have disabilities, are just like them.

Main Activity (35 min)

Designing for Accessibility Scenarios (10 min)

Display: Display each scenario from the **Designing for Accessibility Slide Deck**. While displaying a scenario image, read its accompanying script to the class (see below). Discussion questions are included in relevant slides.

⧉ *Designing for Accessibility Scenarios*

> **Slide 1: Meet Wei**
>
> - Wei is a talented kid who LOVES music. She plays piano and sings very well!
> - Wei is visually impaired. While she is not completely blind, it is very difficult for her to see things, even if they are near to her eyes.
> - Wei has had to learn piano differently from kids who aren't visually impaired, but that hasn't stopped her from becoming a great musician!
> - Wei recently began creating her own music. She is very proud of it! She would like to record and share her music with

her friends.

Slide 2: Introducing Grammaphone!

- Grammaphone is a popular new app that allows people to upload and share their own music with friends.
- Here is a screenshot of the home screen. What do you think about it? How do you think this app works?

Slide 3: Wei's Problem

- Wei needs to use Grammaphone to share her music. After all, it's the most popular music sharing app!
- However, she's having a difficult time using the app. Why do you think that is?

Slide 4: Meet Roger

- Roger is an energetic kid who loves video games! He especially likes playing sports games with his friends.
- Things have been hard for Roger lately, though. Due to an accident a few years ago, Roger's hand had to be "amputated" (surgically removed).
- Thankfully, computer scientists, engineers, and doctors are working hard to provide Roger with a cool-looking robotic "prosthetic hand", which will be awesome!
- But for now, Roger wants to play sports video games with his friends!

Slide 5: Introducing Football Frenzy!

- Football Frenzy is a new mobile game that everyone loves!
- It has cool graphics, awesome sound effects, tight controls, and amazing action!
- Here's a screenshot of the game in action. Based on the interface, how do you think this game is played?

Slide 6: Roger's Problem

- All of Roger's friends play Football Frenzy. It's connected online so you can play it anywhere!
- Roger wants to play Football Frenzy with them, but he is having a difficult time. Why do you think that is?

Slide 7: I Wish...

- Neither of these apps were designed with people like Wei and Roger in mind.
- When it comes to these apps, what specific features do you think Wei and Roger wish for?
- Let's make their wishes come true by redesigning these apps!

App Redesign Activity (25 min)

The final scenario allows students to sketch a redesign of one app from the scenarios ("Grammaphone" for Wei or "Football Frenzy" for Roger).

Students should primarily focus their efforts on making their redesigned app more accessible for either of them, along with any other accessibility features they can think of. Students can design as many screens for the app as they wish, and they do not need to make their app look anything like the original examples.

If there is time, students may redesign apps for both characters.

Distribute: Pass out crayons/pencils/etc to each student as they open their**Thinkspot Journals**.

Share: After designing, students share their app design with a neighbor, or if there is time, some students can share with the whole class. Each student should answer how their app design addresses the need of the user in question.

Wrap Up (5 min)

Discuss: Discuss real-world examples of how apps and other technology can be made more accessible. Ask students how they think making something more accessible can also make it better for others. One example is wheelchair ramps near building entrances. These were originally made for people with wheelchairs, but they are also convenient for people with baby strollers, dollies, and other wheeled equipment.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming
- ▶ **CS** - Computing Systems
- ▶ **IC** - Impacts of Computing

Lesson 11: Nested Loops in Maze

Overview

In this online activity, students will have the opportunity to push their understanding of loops to a whole new level. Playing with the Bee and Plants vs Zombies, students will learn how to program a loop to be inside of another loop. They will also be encouraged to figure out how little changes in either loop will affect their program when they click Run .

Purpose

In this introduction to **nested loops**, students will go outside of their comfort zone to create more efficient solutions to puzzles.

In earlier puzzles, loops pushed students to recognize repetition. Here, students will learn to recognize patterns **within** repeated patterns to develop these **nested loops**. This stage starts off by encouraging students try to solve a puzzle where the code is irritating and complex to write out the long way. After a video introduces **nested loops**, students are shown an example and asked to predict what will happen when a loop is put inside of another loop. This progression leads into plenty of practice for students to solidify and build on their understanding of looping in programming.

Agenda

Warm Up (10 min)
   Introduction
Main Activity (30 min)
   Online Puzzles
Wrap Up (15 min)
   Journaling

View on Code Studio

Objectives

Students will be able to:

- Break complex tasks into smaller repeatable sections.
- Recognize large repeated patterns as made from smaller repeated patterns.
- Identify the benefits of using a loop structure instead of manual repetition.

Preparation

☐ Play through the puzzles to find any potential problem areas for your class.
☐ Make sure every student has a journal.

Vocabulary

- **Loop** - The action of doing something over and over again.
- **Repeat** - Do something again

Teaching Guide

Warm Up (10 min)

Introduction

Briefly review with the class what loops are and why we use them.

- What do loops do?
  - Loops repeat a set of commands. (see vocabulary on command if students don't recognize it)
- How do we use loops?
  - We use loops to create a pattern made of repeated actions.

Tell the class that they will now be doing something super cool: using loops inside loops. Ask the class to predict what kinds of things we would be using a loop inside of a loop for.

"If a loop repeats a pattern, then looping a loop would repeat a pattern of patterns!"

Students don't need to understand this right away, so feel free to move on to the online puzzles even if students still seem a little confused.

Main Activity (30 min)

Online Puzzles

We highly recommend **Pair Programming - Student Video** in this lesson. This may not be an easy topic for the majority of your students. Working with a partner and discussing potential solutions to the puzzles might ease the students' minds.

Also, have paper and pencils nearby for students to write out their plan before coding. Some puzzles have a limit on the number of certain blocks you can use, so if students like to write out the long answer to find the repeats, paper can be useful.

🖥 Code Studio levels

Practice | 🖥 1 | 🖥 2 | *(click tabs to see student view)*

Video | 🎥 3 | *(click tabs to see student view)*

Practice | 🖥 4 | 🖥 5 | 🖥 6 | 🖥 7 | 🖥 8 | 🖥 9 | *(click tabs to see student view)*

Challenge | 🖥 10 | *(click tabs to see student view)*

Practice | 🖥 11 | *(click tabs to see student view)*

Prediction | 🖥 12 | *(click tabs to see student view)*

Free Play | 🖥 13 | *(click tabs to see student view)*

Wrap Up (15 min)

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- What was today's lesson about?
- How did you feel about today's lesson?
- What is a nested loop?
- Can you draw a puzzle that would use a nested loop? Try coding the solution to your own puzzle.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Lesson 12: Fancy Shapes using Nested Loops

Overview

Students will create intricate designs using Artist in today's set of puzzles. By continuing to practice nested loops with new goals, students will see more uses of loops in general. This set of puzzles also offers a lot more potential for creativity with an opportunity for students to create their own design at the end of the stage.

Purpose

In this online activity, students will create designs in Artist that they can proudly share with their loved ones.

The purpose of this activity is to utilize nested loops as a way to inspire students with artistic minds to see coding as another creative outlet. This set of puzzles was built to develop critical thinking skills, an understanding of elementary geometry, and creativity -- all within the scope of nested loops!

Agenda

Warm Up (10 min)
> Introduction

Main Activity (30 min)
> Online Puzzles

Wrap Up (15 min)
> Flash Chat: What did you make today?
> Journaling

Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Combine simple shapes into complex designs with nested loops.
- Count the number of times an action should be repeated and represent it as a loop.
- Break complex tasks into smaller repeatable sections.

Preparation

☐ Play through the puzzles to find any potential problem areas for your class.
☐ Make sure every student has a journal.
☐ Consider what supports your students might need with turns and angles.

Links

> **Heads Up!** Please make a copy of any documents you plan to share with students.

For the Students

- **Turns & Angles** - Student Handout
  [Make a Copy ⌄]
- **Turns & Angles** - Student Video

Vocabulary

- **Loop** - The action of doing something over and over again.
- **Repeat** - Do something again

Teaching Guide

Warm Up (10 min)

Introduction

Review using nested loops in Maze.

Ask the students how they felt about nested loops.

- What did they like and dislike about them?
- What are some advantages of using nested loops?

Tell the students that they will be using nested loops again, but in Artist this time. They will be making amazing projects today!

Main Activity (30 min)

Online Puzzles

Students might benefit from having a puzzle done as a class. If you believe your class could benefit from that, we recommend puzzle 2 of stage 5.

We highly recommend **Pair Programming - Student Video** in this lesson. This may not be an easy topic for the majority of your students. Working with a partner and discussing potential solutions to the puzzles might ease the students' minds.

> ♀ Teacher Tip
>
> Students will have the opportunity to share their own work at the end of this stage. These pieces of artwork can be shared virtually or printed out. We recommend printing out the class's work and displaying it for the students' loved ones to see.

Be sure to have paper and pencils nearby for students to write out their plan before coding. Some puzzles have a limit on the number of certain blocks, so paper can be helpful if students like to write out the long answer before searching for repeating patterns.

🖥 Code Studio levels

Mini-Project: Snowflake #1   🖥 1   🖥 2   🖥 3   🖥 4   🖥 5

(click tabs to see student view)

Mini-Project: Snowflake #2   🖥 6   🖥 7   🖥 8   🖥 9   *(click tabs to see student view)*

Wrap Up (15 min)

Flash Chat: What did you make today?

Get the class together and allow time for students to show off their Artist drawings! Make sure everyone feels included by checking that every student is done with their Artist drawing before starting the presentations. Discuss how each drawing was made and what was in the student's nested loop.

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge

they obtained today and build a review sheet for them to look to in the future.
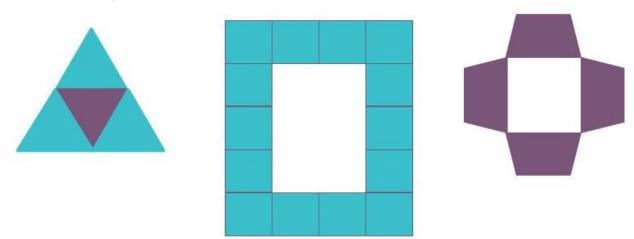
Journal Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- Draw something you used nested loops to make.
- How do nested loops help you code complex images?

Extended Learning

Together We Draw

Have the students pair up with two pieces of paper. Partners should individually draw a shape or simple pattern. Once the simple pattern has been drawn, have the partners switch papers. Now each partner must repeat that pattern how ever many times they want. For example, if one partner draws a square, the other partner can make a rectangle made up of squares! If one partner draws a staircase pattern, the other student can fill the page with staircases! Each pair will have a set of unique drawings. If there's time, have students discuss how they might code their drawings.
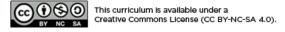
Here are some examples:

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

If you are interested in licensing Code.org materials for commercial purposes,**contact us**.

Lesson 13: Nested Loops with Frozen

Overview

Now that students know how to layer their loops, they can create so many beautiful things. This lesson will take students through a series of exercises to help them create their own portfolio-ready images using Anna and Elsa's excellent ice-skating skills!

Purpose

In this series, students will get practice nesting loops while creating images that they will be excited to share.

Beginning with a handful of instructions, students will make their own decisions when it comes to creating designs for repetition. They will then spin those around a variety of ways to end up with a work of art that is truly unique.

Agenda

Warm Up (15 min)
> **Introduction**

Main Activity (30 min)
> **Code Studio**

Wrap Up (15 min)
> **Journaling**

View on Code Studio

Objectives

Students will be able to:

- Describe when a loop, nested loop, or no loop is needed.
- Recognize the difference between using a loop and a nested loop.
- Break apart code into the largest repeatable sequences using both loops and nested loops.

Preparation

☐ Play through the puzzles to find and potential problem areas for your class.
☐ Make sure every student has a journal.

Vocabulary

- **Loop** - The action of doing something over and over again.
- **Repeat** - Do something again

Teaching Guide

Warm Up (15 min)

Introduction

Ask the class to discuss the last set of puzzles.

- What did they like/dislike?
- Which puzzles were hard? Why?
- Which puzzles were easy? Why?
- If you were to teach nested loops to a friend, what would you say to help them understand?

If there's time, give an introduction to the main characters of today's puzzles, Anna and Elsa from Frozen. Give the class the sister's back story if the class doesn't already know. To build excitement, tell the class they will be using nested loops to make some fantastic drawings with Anna and Elsa's ice skates!

Main Activity (30 min)

Code Studio

🖥 Code Studio levels

Mini-Project: Snowflake #1   🖥 1   🖥 2   🖥 3   🖥 4   🖥 5

(click tabs to see student view)

Mini-Project: Snowflake #2   🖥 6   🖥 7   🖥 8   🖥 9   *(click tabs to see student view)*

This set of puzzles is set up as a progression. This means every puzzle builds a foundation for the next puzzle. Students will enjoy making more and more interesting designs by making small and simple changes to code they have already written.

Wrap Up (15 min)

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- When do you use a loop? When do you use a nested loop?
- Thought exercise: Can you make everything a nested loop can with just a normal loop? Can you draw out an example?
  - Answer: Yes, you can, but it is a lot more difficult. Nested loops make programs simpler.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

Lesson 14: Songwriting

Overview

One of the most magnificent structures in the computer science world is the function. Functions (sometimes called procedures) are mini programs that you can use over and over inside of your bigger program. This lesson will help students intuitively understand why combining chunks of code into functions can be such a helpful practice.

Purpose

The use of functions helps simplify code and develop the student's ability to organize their program. Students will quickly recognize that writing functions can make their long programs easier to read and easier to debug if something goes wrong.

Agenda

Warm Up (20 min)
> **Vocabulary**
> **Sing a Song**

Main Activity (20 min)
> **Functions Unplugged: Songwriting - Worksheet**

Wrap Up (5 min)
> **Flash Chat: What did we learn?**
> **Journaling**

Assessment (5 min)
> **Functions Unplugged: Songwriting - Assessment**

Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Locate repeating phrases inside song lyrics.
- Identify sections of a song to pull into a function.
- Describe how functions can make programs easier to write.

Preparation

☐ (Optional) Watch the Lesson in Action Video.

☐ Print several worksheets for each group.

☐ Print one assessment for each student.

☐ Secure access to songs and lyrics for activity.

☐ Make sure every student has a journal.

Links

> **Heads Up!** Please make a copy of any documents you plan to share with students.

For the Teachers

- **Functions Unplugged: Songwriting** - Lesson in Action Video
- **Functions Unplugged: Songwriting** - Assessment Answer Key

For the Students

- **Songwriting with Functions** - Unplugged Video (**download**)
- **Functions Unplugged: Songwriting** - Worksheet Make a Copy ▾
- **Functions Unplugged: Songwriting** - Assessment Make a Copy ▾

Vocabulary

- **Function** - A named group of programming instructions. Functions are reusable abstractions that reduce the complexity of

writing and maintaining programs.

Teaching Guide

Warm Up (20 min)

Vocabulary

This lesson has one new and important word:

- **Function** - Say it with me: Func-shun

A piece of code that you can call over and over again.

Sing a Song

- Let the class know that today is song day!
- We're going to learn a song together.
    - Start with a simple song, either written out or projected on the screen.
    - Point to the chorus and be sure that the class knows how it goes before you begin on the song.
    - Blast through the song, singing it with them in the beginning, then see what happens when you get to the part where it calls the chorus.

💡 **Chorus:**

> 💡 Teaching Tip

Little bunny Foo Foo
Hopping through the forest
Scooping up the field mice
And bopping 'em on the head
Down came the Fairy
And she said
"Little bunny Foo Foo
I don't wanna see you
Scooping up the field mice
And bopping 'em on the head"*

Little Bunny Foo Foo is being used here as an example only. If your students know this song, feel free to use it. Otherwise, choose an appropriate song that they might be more familiar with (either from music class or the radio.)

Song:

Chorus

I'll give you 3 chances.
Then I'll turn you into a goon!
The next day. . .

Chorus

I'll give you 2 chances.
Then I'll turn you into a goon!
The next day. . .

Chorus

I'll give you 1 chance.
Then I'll turn you into a goon!
The next day. . .

Chorus

"I gave you two chances.
Now I'll turn you into a goon!"
(POOF!)

And the moral of the story is:
Hare today, goon tomorrow!

- It's quite likely that the majority of the class will sing the lyrics for the chorus when you point to that bit.
  - Stop the song once that happens, and explicitly highlight what just happened.
    - You defined the chorus.
    - You called the chorus.
    - They sang the chorus.
- Ask the class why they suppose you only wrote the chorus once at the top of the paper instead of writing it over and over in each place where it is supposed to be sung.
  - What are other benefits of only writing the chorus once when you sing it many times?

Now, imagine that this song is a computer program. Defining a title (like "chorus") for a little piece of code that you use over and over again is called creating a function.

This is helpful to computer scientists for some of the same reasons that it is helpful to songwriters.

- It saves time not having to write all the code over and over in the program.
- If you make a mistake, you only have to change it one place.
- The program feels less complicated with the repeating pieces defined just once at the top.

We are going to play with songs a little more, to try to really understand how often this technique is used!

Main Activity (20 min)

Functions Unplugged: Songwriting - Worksheet

A fantastic way to compare functions to something we see in our everyday lives is to look at songs. Songs often have certain groups of lyrics that repeat over and over. We call such a group a "chorus."

Directions:

- Divide into groups of 4, 5, or 6.
- Give each group several copies of the Songwriting Worksheet.
- Play a short song for the class that contains a clear chorus that does not change from verse to verse.
- Challenge the class to identify (and write down) the chorus.
- Compare results from each group.

Did everyone get the same thing? Sing your choruses together to find out! Play this game over and over until the class has little trouble identifying the choruses.

- It is often easier just to have the class listen to (or watch) the song, then vote on what the chorus is by singing it together, rather than writing the whole thing down. If you choose this method, consider having the class do a written chorus for the final song selection to be sure that the visual learners get proper reinforcement.

Wrap Up (5 min)

Flash Chat: What did we learn?

- Would you rather write lyrics over and over again or define a chorus?

- Do you think it's possible to make multiple choruses for the same song?
- Does it make sense to make a new chorus for every time it's needed in a song?

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- What was today's lesson about?
- How do you feel about today's lesson?
- What is a function and how do you use it?
- Can you think of another activity where you might want to call a special group of instructions several times?

Assessment (5 min)

Functions Unplugged: Songwriting - Assessment

Hand out the assessment worksheet and allow students to complete the activity independently after the instructions have been well explained. This should feel familiar, thanks to the previous activities.

Extended Learning

Use these activities to enhance student learning. They can be used as outside of class activities or other enrichment.

Functional Suncatchers Visit the **CS Fundamentals Unplugged Table** or click on the link for **Functional Suncatchers**. This activity does take a few supplies from the craft store, but it helps students to see the value of calling multiple functions.

Create Your Song

- Start by creating a chorus together, then repeat it between verses of a song that you develop around it.
- Make a change to the chorus, and ponder how much easier it is to change in just one place.
- Change the chorus again, making it much longer than it was originally.
- Add a second chorus and alternate between them in your verses.

Songwriting a Program

- What if we acted out songs instead of singing them? All of a sudden, our chorus would be a function of repeated actions, rather than words.
- Use the concepts of the arrows from the Graph Paper Programming lesson and create a program with lots of repeating instructions.
  - Circle those repeating actions so that the class can see where they are.
  - Define a function called "Chorus" above the program.
  - Cross out everywhere the repeating actions appear in the program and write "Chorus" instead.
- Repeat until the class can go through this process with little direction.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Lesson 15: Functions in Minecraft

Overview

Students will begin to understand how functions can be helpful in this fun and interactive Minecraft adventure!

Purpose

Students will discover the versatility of programming by practicing functions in different environments. Here, students will recognize reusable patterns and be able to incorporate named blocks to call pre-defined functions.

Agenda

Warm Up (10 min)

    Introduction

Bridging Activity - Functions (15 min)

    **Unplugged Activity Using Some Blockly
Preview of Online Puzzles**

Main Activity (30 min)

    Online Puzzles

Wrap Up (15 min)

    Journaling

View on Code Studio

Objectives

Students will be able to:

- Use functions to simplify complex programs.
- Use pre-determined functions to complete commonly repeated tasks.

Preparation

☐ Play through the puzzles to find any potential problem areas for your class.

☐ Make sure every student has a journal.

Vocabulary

- **Function** - A named group of programming instructions. Functions are reusable abstractions that reduce the complexity of writing and maintaining programs.

Teaching Guide

Warm Up (10 min)

Introduction

Help the class understand that functions are simply a chunk of code that has a name. Once defined, you can use that name over and over in your program to tell the computer to run the chunk of code that you assigned to it.

Bridging Activity - Functions (15 min)

This activity will help bring the unplugged concepts from "Functions Unplugged: Songwriting" into the online world that the students are moving into. Choose **one** of the following to do with your class:

Unplugged Activity Using Some Blockly

Pick a song to play that the students enjoy and print out the lyrics. You can use the same song from "Functions Unplugged: Songwriting." Break your class into groups or pairs. Pass out the printed out lyrics (including the repeated chorus) and the basic function blocks from **Unplugged Blocks (Courses C-F) - Manipulatives** to each group or pair of students. See lesson tip for details.

Ask the students to cross out any part of the song that can be made into a function (the chorus is a good

> **♀ Lesson Tip**
>
> Function blocks:
>
>
>
> The block to the left is a function declaration, a block that students will name and use to fill in the function. The block to the right is a function call, a block that makes the function code run. Students will need multiple of the function call blocks.

example) and put it into the function blocks provided. Students should fill in the function declaration with a function name and the words of the repeated lyrics. Once the function declaration is done, ask the students to fill in the function calls and place them on top of the crossed out lyrics.

Once every group or pair is done, ask the class where they put their functions and why. Did everyone make the same function? How often is the function repeated?
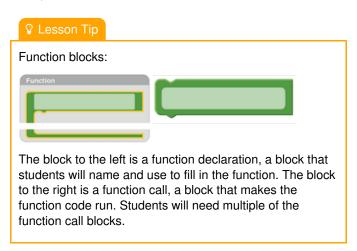
Preview of Online Puzzles

Pull up a puzzle from **Course E Online Puzzles 2018 - Website** . We recommend puzzle 9 of this lesson. As a class, work through the puzzle without using functions. Once you have gotten the solution, display it on a white board or overhead. Ask the class to point to the repeated code. Ask the class how they would simplify the program. Why can you not just use a loop?

On the white board or overhead, rewrite the program without the repeated code, but leaving one line space. In that/those line space(s), call a function. Off to the side, declare the function like the left example block in the lesson tip. Ask the class what they think the code will do now.

Open up a discussion with the class on why functions could be useful in programming. Invite students to discuss the difference between functions and loops.

Main Activity (30 min)

Online Puzzles

🖥 Code Studio levels

Practice 🖵 1 | 🖵 2 | 🖵 3 | 🖵 4 | 🖵 5 | 🖵 6 | 🖵 7 | 🖵 8 | 🖵 9 | 🖵 10

🖵 11   *(click tabs to see student view)*

Free Play   🖵 12   *(click tabs to see student view)*

We recommend providing paper and pencils for students to write (or draw) out ideas. Also, if students are having trouble recognizing patterns, have them work with a partner on the harder puzzles.

Wrap Up (15 min)

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

- What was today's lesson about?
- How do you feel about today's lesson?
- What did your functions do in the programs you wrote today? How did that help you?
- When should you use a function instead of a loop?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

If you are interested in licensing Code.org materials for commercial purposes,**contact us**.

Lesson 16: Functions with Harvester

Overview

Students have practiced creating impressive designs in Artist and navigating mazes in Bee, but today they will use functions to harvest crops in Harvester. This lesson will push students to use functions in the new ways by combining them with while loops and if / else statements.

Purpose

This lesson is meant to further push students to use functions in more creative ways. By also using conditionals and loops, students will learn there are many ways to approach a problem, but some are more efficient than others. These puzzles are intended to increase problem solving and critical thinking skills.

Agenda

Warm Up (10 min)
   Introduction
Main Activity (30 min)
   Online Puzzles
Wrap Up (15 min)
   Journaling

View on Code Studio

Objectives

Students will be able to:

- Recognize when a function could help to simplify a program.
- Use pre-determined functions to complete commonly repeated tasks.

Preparation

☐ Play through the puzzles to find any potential problem areas for your class.
☐ Make sure ever student has a journal.

Vocabulary

- **Function** - A named group of programming instructions. Functions are reusable abstractions that reduce the complexity of writing and maintaining programs.

Teaching Guide

Warm Up (10 min)

Introduction

At this point, your students should already be introduced to functions. Take this time to have them discuss the advantages and disadvantages of using functions in a program. Either have them pair share or discuss as a class. Try using examples of hard or easy puzzles in either Artist or Bee.

Ask the class:

- When would you use a function?
- Why does a function help to simplify your program?
- Do you think functions make programming easier or harder? Why?

Main Activity (30 min)

Online Puzzles

⌨ Code Studio levels

The Harvester ▐◀ 1 *(click tabs to see student view)*

Practice 🖥 2 🖥 3 🖥 4 🖥 5 🖥 6 🖥 7 *(click tabs to see student view)*

How to Create a Simple Function ▐◀ 8 *(click tabs to see student view)*

Practice 🖥 9 🖥 10 🖥 11 *(click tabs to see student view)*

Challenge 🖥 12 *(click tabs to see student view)*

Practice 🖥 13 *(click tabs to see student view)*

Prediction 🖥 14 *(click tabs to see student view)*

Some puzzles will have a function pre-declared for the students to fill in. It may be helpful for the students to write the entire program without a function first, then determine where a function would be useful in the program.

It's important to make sure that every student is completing each puzzle with a dark green dot. If some of your students are struggling to simplify code and use functions, set up teams of expert students within your class to go around and answer questions.

Don't forget to provide pencils and paper to help students sketch out possible solutions.

Wrap Up (15 min)

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- What was today's lesson about?
- How did you feel about today's lesson?
- What makes you realize a function could help your program?
- How do `while` loops and `if / else` statements help your program?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

If you are interested in licensing Code.org materials for commercial purposes,**contact us**.

Lesson 17: Functions with Artist

Overview

Students will be introduced to using functions with the Artist. Magnificent images will be created and modified. For more complicated patterns, students will learn about nesting functions by calling one function from inside another.

Purpose

One of the most important components to this lesson is providing students with a space to create something they are proud of. These puzzles progress to more and more complex images, but each new puzzle only builds off the previous puzzle. At the end of this lesson, students will feel confident with themselves and proud of their hard work.

Agenda

Warm Up (15 min)
   Introduction
Main Activity (30 min)
   Online Puzzles
Wrap Up (15 min)
   Journaling
Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Categorize and generalize code into useful functions.
- Recognize when a function could help to simplify a program.

Preparation

☐ Play through the puzzles to find any potential problem areas for your class.
☐ Make sure ever student has a journal.

Links

> **Heads Up!** Please make a copy of any documents you plan to share with students.

For the Students

- **Unplugged Blocks (Courses C-F)** - Manipulatives Make a Copy ▾

Vocabulary

- **Function** - A named group of programming instructions. Functions are reusable abstractions that reduce the complexity of writing and maintaining programs.

Teaching Guide

Warm Up (15 min)

Introduction

Ask the class to think back to "Functions Unplugged: Songwriting" and recall what a function is. Open a discussion about when to use a function when writing a song.

Tell the class that there are two main components to using functions.

1. **The Declaration**: Function declarations are what create a function. In a function declaration, you fill in the function with code and you give the function a name. You must declare a function before you can use it.

2. **The Call**: Function calls are what makes the program run the code in the function. To call a function, you place the name of the function in your program. Make sure your function is properly defined before calling it in your program.

The class can use songwriting as an example to understand these two components. In the unplugged activity, the function containing the lyrics to the chorus was named "chorus". When we first made this function, we circled the lyrics that would go in the function. Once we named the function, we could read through the lyrics and replace the repeated chorus lyrics with a function call to "chorus".

Continue the conversation until students have a basic understanding of functions being declared and called. If students don't get to this point, make sure to do one of the bridging activities before moving into the Code.org puzzles.

Main Activity (30 min)

Online Puzzles

🖥 Code Studio levels

Prediction   🖥 1   *(click tabs to see student view)*

Practice   🖥 2   🖥 3   🖥 4   🖥 5   🖥 6   🖥 7   🖥 8   🖥 9

(click tabs to see student view)

Challenge   🖥 10   *(click tabs to see student view)*

Practice   🖥 11   *(click tabs to see student view)*

Prediction   🖥 12   *(click tabs to see student view)*

Practice   🖥 13   *(click tabs to see student view)*

Levels   🖥 Extra   🖥 Extra   *(click tabs to see student view)*

Students may benefit from writing code without functions then creating functions from the repeated code. If students don't enjoy doing this in the Code.org workspace, we recommend providing paper and pencils for students to write (or draw) out their ideas.

Wrap Up (15 min)

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- What was today's lesson about?
- How do you feel about today's lesson?
- What are some differences between functions and loops?
- Sketch out a drawing you made today. Can you write the code needed to create this?
- Draw a picture you would like to create with code. Try writing or drafting the code that would make that drawing.

Extended Learning

Draw by Functions

Break the class into groups of 2-3 students. Have each group write a function that draws some kind of shape and a program that uses that function. Depending on the creativity or focus the groups, students might need to be assigned a shape to create. Once every group is done, have the groups switch programs. On a separate piece of paper, each group should draw what the program creates. The groups should then return the programs and drawings to the original group.

Did every group get the drawing they expected? If not, what went wrong? Have the class go through the debugging process and try again.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Lesson 18: End of Course Project

Overview

The next four lessons provide an opportunity for students to put their coding skills to use in a capstone project. This project will help individuals gain experience with coding and produce an exemplar to share with peers and loved ones. Intended to be a multi-lesson or multi-week experience, students will spend time exploring brainstorming, learning about the design process, building, and presenting their final work.

In the explore stage, students will play with pre-built examples of projects in both Artist and Sprite Lab for inspiration. Next, students will learn about the design process and how to implement it in their own projects. They will then be given the space to create their own project in Artist, Sprite Lab, or another interface that they have become familiar with (this is likely the longest stage of the project). Finally, students will be able to present their finished work to their peers.

Purpose

Students may be ready to jump straight into building their projects, but this lesson will help shape their ideas into plans. This structure will keep the dreamers grounded and illuminate a path for those feeling left in the dark. Provide students with ample time to build and revise their projects. The trial and error inevitably involved in this lesson will teach problem solving and persistence.

Agenda

  Day 1 - Explore Project Ideas (45 min)
    Example Projects
  Day 2 - The Design Process (45 min)
    Define and Prepare
  Day 3 - Build Your Project (45 min)
    Try
  Day 4 & 5 - Present Your Project (45 min each)
    Presentations
  Extension Activity
    Reflect and Try Again (45 min)
    Other

View on Code Studio

Objectives

Students will be able to:

- Learn to plan in advance for an ongoing assignment.
- Explain how system limitations can affect project design.
- Describe how compromise can help keep a project on track and inspire creativity.
- Draft and implement plans to resolve any issues in their code.
- Articulate the design process and how it helped shape the finished culminating project.

Preparation

☐ Spend time making your own project with both the Artist and Sprite Lab. Familiarize yourself with the capabilities and limitations of each tool.

☐ Modify the rubric to fit your class goals and print out a copy for each student.

☐ Modify the project design worksheet to fit your class and print one packet for each student.

Links

> **Heads Up!** Please make a copy of any documents you plan to share with students.

For the Teachers
- **CS Fundamentals Final Project** - Rubric
- **Design Process** - Teacher Prep Guide
  [Make a Copy ▾]

For the Students
- **End of Course Project Guide** - Worksheet
  [Make a Copy ▾]

Vocabulary

- **Define** - Figure out the details of the

problems that you are trying to solve
- **Prepare** - Research, plan, and acquire materials for the activity you are about to do
- **Reflect** - Carefully think back on something with the intention of improving the outcome in the future
- **Try** - Attempt to do something

Teaching Guide

Day 1 - Explore Project Ideas (45 min)

Example Projects

Goal: This part of the process is an exploration. Students will sit down with a stage full of example projects to remix and learn. Not only will this give students an idea of what is possible, it will also help them see the limitations of the tool.

Give students a day to play with and remix the example projects found in the lesson. Have them use their journals (or notebook paper) to keep track of thoughts and ideas as they go.

This activity should be done in the same pairs/groups that will be working on projects together over the next several lessons.

Make sure your class understands that they will be working with projects of their own, so they should pay close attention to how these programs were written, as well as the concepts that they use.

Day 2 - The Design Process (45 min)

Define and Prepare

Students will come up with a project and plan their strategy for programming that project in a single day. Students should have a project sketch and a description by the time the day is done.

Preparing Students for the Process:

The most important responsibility you have in kicking off this segment is to help your class understand the scope of this project. Students should be clear about the various expectations over the coming weeks so that they can prepare for their presentations appropriately.

To help your class manage this multi-stage undertaking, they should be given both the project guide and the **CS Fundamentals Final Project - Rubric** on the first day of planning. Students will then be able to follow the rubric each step of the way to predict what their project grade will be in the end.

The project guide will provide a place for students to capture relevant thoughts and processes as they go, so they are more prepared for their presentations in the end.

As the teacher, you should decide which elements of these documents are important to you and be sure to edit or remove anything that you do not intend to draw student focus.

Define and Prepare:

Now that the class has their project guide in hand, they should start filling out the questions under **Day 1**.

Students will likely need to refer back to their notes from playing with the example projects, especially if they don't have access to online Artist or Play Lab project levels while they plan.

> 💡 Lesson Tip
>
> Save 5 minutes or so at the end of the day to have students trade their project guide to look at each other's work. This will help make sure that nothing is omitted or overlooked.

Students should focus on defining and planning their project during Day 1, and not cross over into building until their ideas have been written up and/or drawn out.

If students get stuck, help them work through ideas by asking questions and recalling examples, rather than offering solutions.

Day 3 - Build Your Project (45 min)

Try

Students will use this day to build an initial version of their project.

Equipped with their **Final Project Design - Worksheet** , students should head to the computers to start bringing their projects to life.

This process will come complete with plenty of trial and error. Projects are likely to become truncated versions of the original scope (if not morphed altogether). Remind students that this kind of compromise is common in software design, but they need to be sure to document the reasons for the changes in their product.

Don't let the class forget to fill out their **Final Project Design - Worksheet** as they go. It might be helpful to suggest that pairs/groups take a worksheet break to begin discussing these questions about halfway through their lab time. Alternatively, the navigator can keep their eyes open for pertinent answers while the driver codes.

Be sure that each team member has their own Final Project Design Worksheet, as there are questions about each student's own individual thoughts and behaviors that need to get captured along the way.

Day 4 & 5 - Present Your Project (45 min each)

Presentations

Students will create and present their projects in an approved manner (written, oral, or using multimedia).

Create:

Ideally, you will have class time available to give students to work on their presentations. This will allow them to incorporate rich multimedia components, like **Google Slides**. For other presentation ideas, visit **72 Creative Ways for Your Students to Show What They Know - Website**.

Encourage students to include all of the information from Section J of the Final Project Design Worksheet into their presentation, as well as two or more questions from Section K.

> 💡 Lesson Tip:
>
> If you are looking for a section of this series to assign as homework, this is it! Projects do not have to be presented in electronic form, so this is a great offline option. Other ways to present projects (both online and offline) include:
>
> - Report
> - Blog post
> - Online
> - In front of the class with a poster

Present:

Students should showcase their apps first, then they can discuss the questions that they covered in their presentations.

It can be very helpful to have students sign up for a specific order in which to give their presentations, so that they are able to enjoy the demonstrations of their classmates without worrying about whether they will be called on next.

Extension Activity

Reflect and Try Again (45 min)

Students will work with another group to give and receive feedback in an effort to make each other's projects stronger.

Reflect:

For reflections, have each group pair up with another group to try each other's projects. After about 10 minutes, have the groups discuss the questions in the Final Project Design Worksheet.

Encourage students to ask the questions on the Final Project Design Worksheet and write down feedback provided by their reviewing teams so that they can refer back to it later. This portion should take approximately 15 more minutes.

Try Again:

With their new reflections in hand, students can head back to their machines to make a handful of edits. With just 10 minutes left, they will likely have to select only the most important feedback to incorporate.

Other

If your students are already comfortable with coding concepts, try having them create their projects in another platform, like **Scratch** or **Alice**.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

▶ **IC** - Impacts of Computing

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.