Course C

Course C was developed for students in and around the second grade. It uses a limited understanding of shapes and elementary math concepts.

Students will create programs with loops, events, and conditionals. They will translate their initials into binary, investigate different problem-solving techniques, and discuss how to respond to cyberbullying. By the end of the course, students will create interactive games that they can share. Each concept in Course C is taught from the beginning, graduating toward experiences that allow for growth and creativity to provide all students a rich and novel programming experience.

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

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

Chapter 1: Digital Citizenship

Lesson 1: Putting a STOP to Online Meanness

Unplugged | Cyberbullying | Nested Loop

In this lesson, you'll learn about meanness and what to do if you encounter it online.

Lesson 2: Password Power-Up

Unplugged | Online Safety

In this lesson, you'll learn about how passwords protect your information, and how to make a good password.

Chapter Commentary

Digital Citizenship

Chapter 2: Sequencing

Lesson 3: My Robotic Friends Jr.

Unplugged | Sequencing

In this lesson, you'll pretend your classmates are robots and program them to build patterns of stacked cups.

Lesson 4: Programming with Angry Birds

Skill Building | Sequencing

Learn about sequences and algorithms with Angry Birds.

Lesson 5: Debugging in Maze

Skill Building | Sequencing

Find problems in puzzles and practice your debugging skills.

Lesson 6: Collecting Treasure with Laurel

Skill Building | Sequencing

Write algorithms to help Laurel the Adventurer collect lots of gems!

Lesson 7: Creating Art with Code

Skill Building | Sequencing

Create beautiful images by programming the Artist.

Chapter Commentary

Sequencing

Chapter 3: Binary

Lesson 8: Binary Bracelets

Unplugged | Binary

Create your very own binary bracelet and learn how computers remember information!

Chapter Commentary

Binary

Chapter 4: Loops

Lesson 9: My Loopy Robotic Friends Jr.

Unplugged | Loops

In this lesson, you'll program your classmates again, but using loops you'll be able to solve bigger and more complicated problems.

Lesson 10: Loops with Rey and BB-8

Skill Building | Loops

Help BB-8 through mazes using loops!

Lesson 11: Harvesting Crops with Loops

Skill Building | Loops

Let's use loops to help the harvester collect some veggies!

Lesson 12: Looking Ahead with Minecraft

Skill Building | Loops

Avoid the lava! Here you will start to learn about conditionals in the world of Minecraft.

Lesson 13: Sticker Art with Loops

Application | Loops

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

Chapter Commentary

Loops

Chapter 5: Events

Lesson 14: The Big Event

Unplugged | Events

Play a fun game to learn about events.

Lesson 15: Build a Flappy Game

Skill Building | Events

Build you own Flappy Bird game however you like, then share it with your friends!

Lesson 16: Chase Game with Events

Skill Building | Events

It's time to get creative and make a game in Play Lab!

Chapter Commentary

Events

Chapter 6: Data

Lesson 17: Picturing Data

Unplugged | Data

Data can be used to help students understand their world and answer interesting questions. In this lesson, students will collect data from a Play Lab project and visualize it using different kinds of graphs.

Chapter Commentary

Data

Chapter 7: End of Course Project

Lesson 18: End of Course Project

End of Course Project

Get those hands ready for plenty of coding! It's time to start building your project.

Chapter Commentary

End of Course Project

This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 1: Putting a STOP to Online Meanness

Overview

common sense education

This lesson was originally created by **Common Sense Education**.

The internet is filled with all kinds of interesting people, but sometimes, some of them can be mean to each other. With this role play, help your students understand why it's often easier to be mean online than in person, and how to deal with online meanness when they see it.

Purpose

Common Sense Education created this lesson to teach students about meanness and what they should do if they encounter it online.

Agenda

Learn: What Is Meanness? (10 min)
  Key Vocabulary
Perform: Online vs. Face-to-Face (5 min)
Explore: STOP Meanness (15 min)
Wrap Up: Pause & Think Moment (5 min)
Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Understand what online meanness can look like and how it can make people feel.
- Identify ways to respond to mean words online, using S-T-O-P.

Preparation

- ☐ Review instructional materials.
- ☐ Print handout(s) for each student.
- ☐ Bring a stuffed animal or toy to class.

Links

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

For the Teachers

- **Putting a STOP to Online Meanness: Lesson Slides** - Slide Deck

For the Students

- **Putting a STOP to Online Meanness: STOP Online Meanness** - Student Handout
  Make a Copy ▾

Teaching Guide

Learn: What Is Meanness? (10 min)

Key Vocabulary

- **online**: using a computer, phone, or tablet to visit a website or app

Ask: *What does mean behavior look like?* (**Slide 3**)

Student answers will vary, but highlight the following: making fun of how someone looks, spreading lies about someone, and saying rude things to them.

Ask: *How does mean behavior make people feel?*

Invite students to respond. Answers will vary, but possible responses include feeling hurt, angry, upset, and scared.

Ask: *What should you do if you ever experience someone being mean to you or if you see someone being mean to another person?*

Invite students to respond, and emphasize that they should always tell a trusted adult if they experience or witness any form of meanness.

Say: *The internet is filled with fun and interesting things, but sometimes people can be mean to each other online.*

Clarify that **online** means *using a computer, phone, or tablet to visit a website or app.* (**Slide 4**)

Explain to students that some examples of online meanness include:

- Sending a mean message to someone
- Posting mean things about someone on a website
- Making fun of someone online
- Doing mean things to someone's character in an online game

If students are familiar with the term **cyberbullying**, *using digital devices, sites, and apps to intimidate, harm, and upset someone*, you can use that term instead.

Ask: *Have you ever seen someone make someone else feel bad online?*

Invite students to pair-share, but emphasize that they can explain what happened but not use real names.

Explain that they will be learning more about how online meanness can happen and what to do when it happens to them or to someone they know.

Perform: Online vs. Face-to-Face (5 min)

Ask: *Why do you think someone would be mean to someone else online?* (**Slide 5**)

Answers will vary, but possible reasons include: they were trying to joke around, or they feel badly about themselves and are taking it out on someone. Students might also note that people might say things online that they wouldn't face-to-face.

Say: *It can sometimes be easier for people to be mean online versus in person, because they don't have to actually see the person face-to-face and see the other person's emotions.*

Explain to students that they are going to pretend the stuffed animal is actually someone you know. While facing the stuffed animal, say (in a mean tone): *I can't believe you like superhero shows. That is something my little brother would watch!*

Then, project **Slide 6** and have students silently read the chat conversation.

Ask: *What was the difference between writing something mean versus saying it to the person?* Project **Slide 7** and gather student responses in the t-chart.

Emphasize that both were mean (saying it face-to-face and online), but that it was probably easier to type it out than to say it to the person, because you don't have to see the person's reaction and they can't hear the tone of your voice.

Say: *Remember, if you wouldn't say something to another person's face, it's definitely not OK to say it online.*

Explore: STOP Meanness (15 min)

Distribute the **STOP Online Meanness Student Handout**. Have a student read the scenario on **Slide 8** aloud. Allow pairs five minutes to complete the first two questions.

Invite students to share their responses with the class.

Say: *If you ever experience online meanness, remember to STOP!*

Project Slide 9 and review the four rules for dealing with online meanness. As you review each rule, ask the following follow-up questions to guide the discussion.

S: Step away

- *Why do you think you should stop using your device and step away?*

T: Tell a trusted adult

- *If someone makes you feel angry, sad, or scared online, which grown-ups can you tell and ask for help?*

O: OK sites only

- *Why is it important to go online only with an adult, or when an adult says it is OK?*

Emphasize that some sites are not made for kids, and allow kids to communicate with others. This opens up opportunities for online meanness to occur.

P: Pause and think

- *Would it be helpful to talk to the person who was being mean?*

Ask: *What advice would you give Jada to respond to this situation? Remember S-T-O-P!*

With a partner, have students complete question 3 of the **STOP Online Meanness Student Handout** and then invite students to share their responses with the class.

- **S**: Jada should step away from the computer.
- **T**: Jada should tell an adult she trusts what happened.
- **O**: Jada should not go back online or return to the pony website until an adult says it is OK.
- **P**: Jada should take a moment to pause and think about the situation. If Jada and Michael are good friends, Jada may want to tell Michael how his actions made her feel, after she gets advice from an adult. If Michael continues to be mean to her, she should immediately tell an adult.

Ask: *Which of the 4 rules in S-T-O-P do you think is the most important? Why?*

Give students a moment to think and then informally poll the class. Be sure to emphasize that telling a trusted adult (T) is the single most important thing they should do if they ever experience online meanness.

Wrap Up: Pause & Think Moment (5 min)

Say: *We go online to watch videos, send messages to people we know, play games, and do homework. Sometimes people say mean or scary things. If you ever experience someone being mean online, remember to S-T-O-P!*

Direct students to the Pause & Think Moment on page 2 of the **STOP Online Meanness Student Handout**. Read the directions aloud and allow students to complete the reflection independently. (**Slide 10**)

Invite students to share their reflections with the class. Collect handouts to assess student learning.

Extended Learning

Here are additional resources you can provide students to enhance their learning:

- **Family Activity**
- **Family Tips**

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **IC** - Impacts of Computing

© Common Sense Media 2020. Lessons are shareable under a Creative Commons BY-NC-ND license. No remixing permitted. View detailed license information at creativecommons.org. Common Sense and other associated names and logos are trademarks of Common Sense Media, a 501(c)(3) nonprofit organization (FEIN: 41-2024986).

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

Lesson 2: Password Power-Up

Overview

This lesson was originally created by **Common Sense Education**.

Stronger, more secure online passwords are a good idea for everyone. But how can we help kids create better passwords and actually remember them? Use the tips in this lesson to help kids make passwords that are both secure and memorable.

Purpose

Common Sense Education created this lesson to teach students about how strong passwords can help protect their privacy.

Agenda

Warm Up: Pssst ... What's a Password? (10 min)
- Key Vocabulary

Evaluate: Uh-Oh! If ... Then ... (20 min)
Create: Power Up Your Password (10 min)
Wrap Up: Password Tips Notes (5 min)
Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Define the term "password" and describe a password's purpose.
- Understand why a strong password is important.
- Practice creating a memorable and strong password.

Preparation

- ☐ Review instructional materials.
- ☐ Print handout(s) for each student.
- ☐ Prepare writing paper or notebooks for students.

Links

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

For the Teachers

- **Password Power-Up: Lesson Slides** - Slide Deck
- **Password Power-Up: If...Then...Scenarios Answer Key** - Student Handout Make a Copy
- **Password Power-Up: Power-Up Your Password Answer Key** - Student Handout Make a Copy
- **Password Power-Up: Lesson Quiz Answer Key** - Website

For the Students

- **Password Power-Up: If...Then...Scenarios** - Student Handout Make a Copy
- **Password Power-Up: Power-Up Your Password** - Student Handout Make a Copy
- **Password Power-Up: Lesson Quiz** - Form

Teaching Guide

Warm Up: Pssst ... What's a Password? (10 min)

Key Vocabulary

- **password**: a secret string of letters, symbols, and numbers that you can use to restrict who can access something digital
- **phrase**: a group of words that go together and are easy to remember
- **symbol**: a character other than a number or letter, such as #, !, or @.
- **username**: a name you create to sign into a website, app, or game

Before the lesson: As an optional activity before the lesson, have students play the **Password Protect** game in Digital Passport™ by Common Sense Education. This will help introduce key concepts of this lesson. To see more, check out the **Digital Passport Educator Guide**.

Ask: *What is something in your life that you take steps to protect? Why and how do you protect it?* (**Slide 4**)

Have students think-pair-share, and then call on students to share out. Examples might include *hiding a diary*, *using a bike lock*, *keeping a toy in a safe place*, etc. Clarify that one of the ways people protect something is by restricting who can access it.

Ask: *Let's think about when we use devices, like a phone or computer. How do you limit who can access something that you want to protect, and why would you limit access?*

Examples might include password-protection, device-lock, fingerprint scanner, not letting others use the device.

Say: *One of the most common ways to protect your devices and information online is by using a password. A **password** is a secret string of letters, symbols, and numbers that you can use to restrict who can access something digital. Some passwords, however, are stronger than others because they are harder for someone to figure out. Let's investigate why strong passwords are important and how you can make sure yours are strong.* (**Slide 5**)

Evaluate: Uh-Oh! If ... Then ... (20 min)

Distribute If ... Then Scenarios Student Handout. Tell students that they will be doing a jigsaw activity to understand WHY passwords are important.

Divide the class into five groups and tell students this is their "home" group. Call on a student to read the handout directions aloud. (**Slide 6**)

Assign each group one of the scenarios from the handout. Allow groups to work for five to seven minutes.

Re-divide the class into new groups so that each group includes at least one student from each scenario (1, 2, 3, 4, and 5). If necessary, there can be more than one person from a particular "home" group in an "expert" group. Allow five to seven minutes for each "expert" to present while the rest of the group takes notes. (**Slide 6**)

Reconvene and ask: *Based on these scenarios, why do you think it's important to have a strong password? Take turns sharing your idea with your partner.* Allow one minute to pair-share.

Invite students to share out their answers. If necessary, prompt students to refer directly to the scenarios and to the consequences that would happen if someone's password was compromised. Examples could include loss of money, people knowing your private information, identity theft, and other unknown future consequences.

Create: Power Up Your Password (10 min)

Say: *The consequences we just talked about might seem scary. But there is something you can do to make sure no one can ever guess your password. Here are some important steps to power up your password.*

Distribute and project the **Power Up Your Password Student Handout**. Say: *One way to make a strong password is to start with a memorable phrase. A **phrase** is a group of words that go together and are easy to remember. We're going to do a practice round together using the phrase "There's no way I'm kissing a frog." So we've already completed step one, which is to come up with a phrase.* (**Slide 7**)

Invite a student to read step two. Ask: *What word would our example phrase make?* Invite a student to answer and add "tnwikaf" on the projected handout.

Repeat steps three through five: reading the step aloud, calling on students to answer, and completing the "Practice Round" section of the handout. Student answers will vary as students will choose to capitalize different letters and insert different numbers. If necessary, clarify that students can insert numbers anywhere in the password.

Say: *Now you will follow these steps to come up with a password of your own. Work independently to complete the "Your Turn" section of the handout.* Allow students five minutes to complete the handout.

Wrap Up: Password Tips Notes (5 min)

Project the **Password Tips** and read each one aloud. Direct students to fill in the blanks on their handout as you read them. For the last tip, clarify that a **symbol** is *a character other than a number or letter, such as #, !, or @.* (**Slide 8**)

Say: *As you get older, having a strong password will become even more important. Passwords will help you protect your social networking profiles when you are in high school, keep your grades private when you are in college, and protect your bank accounts and online store accounts when you are an adult.*

Have students complete the **Lesson Quiz**.

Extended Learning

Here are additional resources you can provide students to enhance their learning:

- **Family Activity**
- **Family Tips**
- Facilitate a related BeakoutEDU game that was co-created with Common Sense Education, called **Lip Sync Revenge**! You will need to create a free account in order to access the game and related resources.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

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

© Common Sense Media 2020. Lessons are shareable under a Creative Commons BY-NC-ND license. No remixing permitted. View detailed license information at creativecommons.org. Common Sense and other associated names and logos are trademarks of Common Sense Media, a 501(c)(3) nonprofit organization (FEIN: 41-2024986).

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

Lesson 3: My Robotic Friends Jr.

Overview

Using a set of symbols in place of code, students will design algorithms to instruct a "robot" to stack cups in different patterns. Students will take turns participating as the robot, responding only to the algorithm defined by their peers. This segment teaches students the connection between symbols and actions, the difference between an algorithm and a program, and the valuable skill of debugging.

Purpose

This unplugged lesson brings the class together as a team with a simple task to complete: get a "robot" to stack cups in a specific design. This activity lays the groundwork for the programming that students will do throughout the course as they learn the importance of defining a clearly communicated algorithm.

Agenda

Warm Up (5 min)

- **Talking to Robots**

Activity (30 min)

- **Introduction and Modeling**
- **Handy Rules:**
- **Differentiation Options:**
- **Programming Your Robots**

Wrap Up (10 min)

- **Journaling**

View on Code Studio

Objectives

Students will be able to:

- Attend to precision when creating instructions
- Identify and address bugs or errors in sequenced instructions

Preparation

- ☐ Prepare a stack of 20 paper cups (or paper trapezoids) for each group of 2-3 students.
- ☐ Display the symbols from **My Robotic Friends - Symbol Key** where students can reference throughout the lesson.
- ☐ (Optional) Print out one **My Robotic Friends - Cup Stacking Ideas** per group of 2-3 students.
- ☐ Make sure each student has a journal.

Links

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

For the Students

- **My Robotic Friends** - Cup Spacing Make a Copy
- **Feeling Faces** - Emotion Images Make a Copy
- **My Robotic Friends** - Symbol Key Make a Copy
- **My Robotic Friends** - Unplugged Video (**download**)
- **My Robotic Friends** - Cup Stacking Ideas Make a Copy
- **My Robotic Friends** - Paper Trapezoid Template Make a Copy

Vocabulary

- **Algorithm** - A list of steps to finish a task.
- **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.

Teaching Guide

Warm Up (5 min)

Talking to Robots

Display: Watch one of the videos below to give students context for the types of things that robots can do:

- **Asimo by Honda** (3:58)
- **Egg Drawing Robot** (3:15)
- **Dancing Lego Robot** (1:35)

Discuss: Refer to the video that you chose and ask students how they think that the robot knew what to do. Does a robot really “understand” what you say? Is it worried about getting in trouble if it doesn't do what it's told?

Discussion Goal

The goal of this quick discussion is to call out that while robots may seem to behave like people, they're actually responding only to their programming. Students will likely refer to robots from movies and TV that behave more like humans. Push them to consider robots that they've seen or heard of in real life, like Roombas, or even digital assistants like Amazon Alexa.

Say: Robots can only do what they've been told to do, but we don't just tell them using words. In order to do something, a robot needs to have a list of steps that it can read. Today, we are going to learn what it takes to make that happen.

Activity (30 min)

Introduction and Modeling

Set Up: Have stacks of cups or cut paper trapezoids available for groups.

Display: Display **My Robotic Friends - Symbol Key** or write the allowed actions on the board - make sure these are in a place where they can be seen for the whole activity. Explain to the class that these will be the only four actions that they can use for this exercise. For this task, they will instruct their “robot” friend to build a specific cup stack using only the commands listed on the key.

Model: In order to explain how the instructions are intended to work, model for the class how to create and follow an algorithm for replicating a simple pattern. Place a single stack of cups in front of you to start.

Display: Hold up the pattern you plan to model. A simple three cup pattern is a great place to start.

U

Unplugged

My Robotic Friends

Symbol Key (Course B)

↑	↓
Pick Up Cup	**Put Down Cup**
→	←
Step Forward	**Step Backward**

Teaching Tip

Handy Rules:

- **Up** means that the cup automatically goes up as high as it needs to
- **Down** means that it automatically goes down until it lands on something
  - The hand automatically returns to cup stack after setting down a cup
- **Forward** means the robot moves one step (1/2 cup width) forward
- **Backward** means the robot moves one step (1/2 cup width) Backward
  - Note: Students may not use backward at this age unless they want to build the cup stacks in reverse (which is also okay)
- Programmers are not allowed to talk when the robot is working. This includes blurting out answers or pointing out when the robot has done something wrong
- Programmers should raise their hand if they see a bug

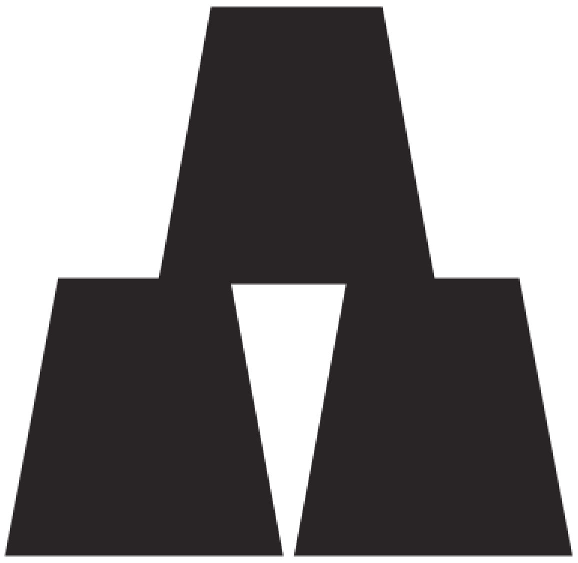
Differentiation Options:

Simplify: Does this all feel a little complicated for your students?

Don't forget to model this in front of the class until students understand all of the rules. If it's still confusing, try running this whole activity together as a classroom using volunteers as robots, instead of breaking up into groups!

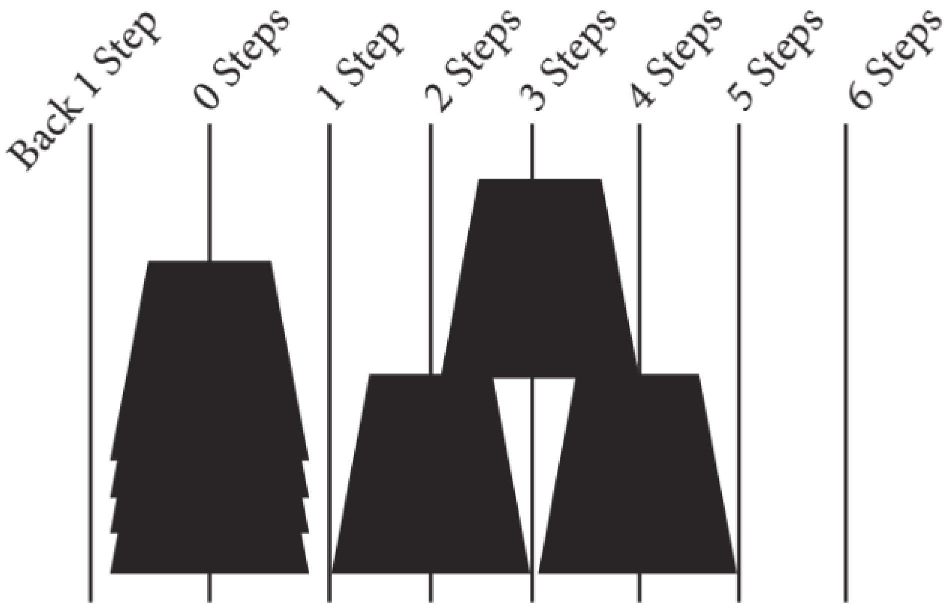
Intensify: Are your students more advanced? Do you want this lesson to relate more closely to the online puzzles? Here are some modifications that you can make:

- One arrow corresponds to one movement
  - When a cup is removed from the stack, it returns to table-level before moving
  - Students need to use multiple "up" arrows to lift the cup multiple levels
  - Students need to use multiple "down" arrows to lower the cups multiple levels
  - Students need to use the "back" arrows to get back to the cup stack



Prompt: Ask the class what the first instruction should be, using *only the four instructions allowed*. The first move should be to "pick up cup." If students suggest something else from the list, perform that action and allow them to see their error. If they suggest something not from the list, make a clear malfunction reaction and let them know that the command is not understood.

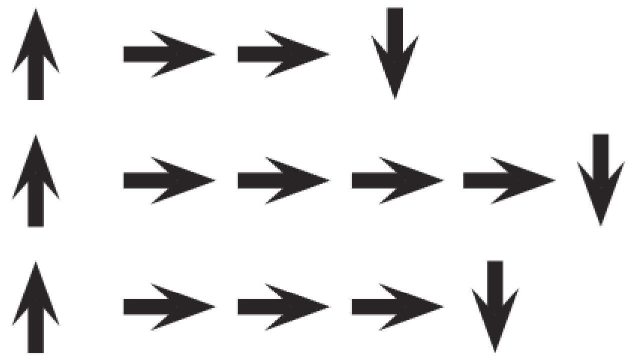
With cup in hand, ask the class to continue giving you instructions until the first cup is placed. This is a great place to clarify that "step forward" and "step backward" each imply moving half a cup width. See the image below for reference.



Continue asking for instructions from the classroom until you have completed the entire design.

Once your stack is complete, point out that they just gave you a list of steps for completing a task. That's an algorithm. Algorithms are great for sharing ideas, but spelling them out word by word can take a long time. That's what the symbols are for! When you change an algorithm into symbols that a robot (or computer) understands, that's called programming.

Ask the class to help you write the "program" for that first move by changing the text into an arrow. Then work with them to write down the rest of the moves necessary to complete the pattern. Depending on the confidence of your students, you might switch back and forth frequently between acting as the "robot" and writing down the code, or you might push them to write the whole program before you will implement it. One possible solution looks like this:



Volunteer: Once the class has completed the model program, ask one of the students to come up and act as the "robot" to ensure that the program really works. Encourage them to say the instructions out loud as they "run" the code.

Programming Your Robots

Group: Place students into groups of 4. Each group should then further break down into two pairs - each pair will develop their own program to be "run" by the other pair.

Distribute: Give each group one stack of cups or paper cutouts.

Display: Show **My Robotic Friends - Cup Stacking Ideas** to the class or hand out individual copies for groups to use. Have each pair (not group) choose which idea they would like their robots to do. Try to push for an easier idea for the first time, then have them choose a more complex design later on. Encourage pairs to keep their choice secret from the other half of their group.

Discuss: Give each pair time to discuss how the stack should be built, using only the provided symbols. Make sure each group writes down the "program" somewhere for the "robot" to read later.

Do: Once both of the group's pairs have completed their programs, they can take turns being "robots" for each other by following the instructions the other pair wrote. Encourage students to watch their "robot" closely to ensure that they are following instructions. If a student sees a bug and raises their hand, have the robot finish the instructions to the best of their ability. Afterward, have the students discuss the potential bug and come up with solutions. Continue repeating until the stack is built properly.

Circulate: Look for groups who are trying to take shortcuts by adding extra things (like numbers) to their code. Praise them for their ingenuity, but remind them that for this exercise, the robots do not understand *anything* but the provided symbols. If you like, you can hint that they should save their brilliant solution for the next time they play this game, since they might get the chance to use their invention soon!

Iterate: Depending on the time available, mix up the pairs and give them a chance to do a different pattern. Each time groups repeat the process, encourage them to choose a more challenging pattern.

Discuss: After everyone has had a chance to be the robot, bring the class back together to discuss their experience. In particular, discuss as a class:

- What was the most difficult part of coming up with the instructions?
- Did anyone find a bug in your instructions once your robot started following them?
    - What was the bug?
    - Why do you think you didn't notice it when writing the program?
- When you were the robot, what was the hardest part of following the instructions you were given?

Discussion Goal

Sense making: The goal of this discussion is to give students space to make sense of their experience both as robot and programmer. The questions are intentionally broad, but designed to get students thinking about the challenges of writing a clear program and the constraints of a robot or computer in interpreting your instructions.

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:

- Draw one of the **Feeling Faces - Emotion Images** that shows how you felt about today's lesson in the corner of your journal page.
- Draw your own stack of cups that you would like to see a robot build.
- Can you create a program for that cup stack?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **AP** - Algorithms & Programming

This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 4: Programming with Angry Birds

Overview

Using characters from the game Angry Birds, students will develop sequential algorithms to move a bird from one side of a maze to the pig at the other side. To do this they will stack code blocks together in a linear sequence, making them move straight, turn left, or turn right.

Purpose

In this lesson, students will develop programming and debugging skills on a computer platform. The block-based format of these puzzles help students learn about sequence and concepts, without having to worry about perfecting syntax.

Agenda

View on Code Studio

Objectives

Students will be able to:

- Translate movements into a series of commands.
- Identify and locate bugs in a program.

Preparation

☐ Play through the puzzles 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.

Links

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

For the Students

- **Course C, Lesson 2 Maze Bridging Page** - Puzzle Manipulative (PDF)
- **Unplugged Maze Blocks** - Manipulatives

Vocabulary

- **Algorithm** - A list of steps to finish a task.
- **Bug** - Part of a program that does not work correctly.
- **Debugging** - Finding and fixing problems in an algorithm or program.
- **Sequencing** - Putting commands in correct order so computers can read the commands.

Teaching Guide

Warm Up (4 min)

Review Unplugged Activity

This lesson relies on many of the unplugged ideas that students have learned in the weeks leading up to this first online activity. It is important that you bring those concepts (such as persistence, debugging, algorithms, and programs) around full-circle so that your class can benefit from them in their online work as well.

Display: Show students a cup stack from the "My Robotic Friends" exercise that they completed in the lessons prior to this one.

Discuss: Ask students to recall the symbols used in "My Robotic Friends."

- What happens when the robot reads the different arrows?

Encourage students to think about the debugging tips:

- Was everything right at the first step?
- How about the second?
- Where did it go wrong?

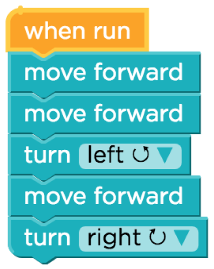
Transition: Once you are satisfied that your students remember "My Robotic Friends", you can move into the Bridging Activity.

Bridging Activity - Programming (10 min)

Transitioning from Unplugged to Online

This short activity will help students relate the ideas of persistence and debugging to the puzzles that they are about to complete online.

Display: Project a copy of the **Course C, Lesson 2 Maze Bridging Page - Puzzle Manipulative (PDF)** for the class to see. Make sure that you have pre-placed the movement blocks in the workspace using **Unplugged Maze Blocks - Manipulatives** in a configuration like the one below:



```
when run
move forward
move forward
turn left ↺ ▼
move forward
turn right ↻ ▼
```

Model: Tell students that you have this workspace on display that looks just like the area that they will see when they start to do the Code.org puzzles online. As the teacher, let them know that you are SO SMART that you already put all of the code in that you are going to need to solve this puzzle, then ask them to watch you "Run" it by moving your finger (or a penny, or some other indicator) along the path.

It won't be long before you run into a block of TNT. Feign frustration.

Discuss: - What am I feeling right now, do you think? - Should I quit? - Should I throw all of the code away and start over?

Think: How can I fix this program so that I don't run into the TNT?

Pair: Have students work on solutions to get the bird around the TNT. Depending on your classroom, you might want to either have them fix each mistake one at a time (with demos in between) or students might feel comfortable working together to fix the entire program.

Share: Have volunteers come up to help move the blocks into the right location. "Run" the program over and over as a class, fixing bugs, until the bird does what it is supposed to. Continue to point out experiences that relate to persistence, frustration, and debugging.

When your class reaches the pig, celebrate not only their achievements, but their persistence!

Previewing Online Puzzles as a Class (3 min)

Students should now be ready to see a real puzzle in action!

Model: Pull up Puzzle 5 to do in front of the class. This will be the same puzzle that they just saw in the bridging activity. While working through this puzzle with the class, remind students that making mistakes is okay and remind them that the only way to be successful is to be persistent.

Discuss: Does anyone remember how to solve this puzzle?

> Lesson Tip
>
> Some students may struggle with turning their bird in the correct direction, particularly when the bird isn't facing up. Remind students that when we say turn left or right, we're giving directions from the bird's point of view.

As the teacher, you should decide if you will have the students remind you how to solve it from their seats, or come to the computer to drag the actual blocks in one-by-one.

Transition: Now that students have seen an online puzzle in practice, they should be ready to start solving puzzles of their own. Continue to the lab or bring out their classroom machines.

Main Activity (30 min)

Online Puzzles

> 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

Code Studio levels

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

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

Challenge 8 *(click tabs to see student view)*

Practice 9 *(click tabs to see student view)*

Predict 10 *(click tabs to see student view)*

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

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

Circulate: 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:

- Utilize pair programming 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.

Discuss: After providing students with end-of-class warnings, grab everyone's attention and get them to reflect on the experiences that they just had.

- Did anyone feel frustrated during any of the puzzles?
- Did anyone notice the need to be persistent?

Transition: Have students grab their Thinkspot Journals and take a moment to leave lessons for themselves.

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?
- Draw an activity you like to do that you struggled with the first time. Draw or describe how you got better.

Extended Learning

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

Create Your Own

In small groups, let students design their own mazes and challenge each other to write programs to solve them. For added fun, make life-size mazes with students as the pig and bird.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **AP** - Algorithms & Programming

This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 5: Debugging in Maze

Overview

Debugging is an essential element of learning to program. In this lesson, students will encounter 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

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

View on Code Studio

Objectives

Students will be able to:

- 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 to find any potential problem areas for your class.
- ☐ (Optional) Pick a couple of puzzles to do as a group with your class.
- ☐ Review **CS Fundamentals Main Activity Tips - Lesson Recommendations**.
- ☐ Review the **Debugging Guide** from this course's curriculum overview page 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

- **CS Fundamentals Main Activity Tips** - Lesson Recommendations Make a Copy ▾

For the Students

- **Pair Programming** - Student Video

Vocabulary

- **Bug** - Part of a program that does not work correctly.
- **Debugging** - Finding and fixing problems in an algorithm or 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?
- The puzzles in this unit 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.

Vocabulary

This lesson has three new and important vocabulary words:

- ***Bug*** - Say it with me - Buhh-g. Something that is going wrong. An error.
- ***Debugging*** - Say it with me: Dee-bug-ing. To find and fix errors.
- ***Persistence*** - Say it with me: Purr-siss-tense. Not giving up. Persistence works best when you try things many different ways, many different times.

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

Before letting the students start on the computer, remind them of the advantages of pair programming 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

Debugging with the Step Button 1 ***(click tabs to see student view)***

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

Challenge 8 ***(click tabs to see student view)***

Predict 9 ***(click tabs to see student view)***

Practice 10 ***(click tabs to see student view)***

Levels Extra Extra *(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

This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 6: Collecting Treasure with Laurel

Overview

In this series of puzzles, students will continue to develop their understanding of algorithms and debugging. With a new character, Laurel the Adventurer, students will create sequential algorithms to get Laurel to pick up treasure as she walks along a path.

Purpose

In this lesson, students will be practicing their programming skills using a new character, Laurel the Adventurer. When someone starts *programming* they piece together instructions in a specific order using something that a machine can read. Through the use of programming, students will develop an understanding of how a computer navigates instructions and order. Using a new character with a different puzzle objective will help students widen their scope of experience with sequencing and algorithms in programming.

Agenda

View on Code Studio

Objectives

Students will be able to:

- Order movement commands as sequential steps in a program.
- Represent an algorithm as a computer program.
- Develop problem solving and critical thinking skills by reviewing debugging practices.

Preparation

☐ Play through the puzzles to find potential problem areas for your 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 Students

- **Unplugged Blocks (Courses C-F)** - Manipulatives

Vocabulary

- **Algorithm** - A list of steps to finish a task.
- **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 (5 min)

Introduction

This lesson uses most of the same blocks from the previous lessons and adds the ability to `collect`. Tell the students that this block will allow Laurel the Adventurer to pick up the treasure that she is standing over. This new block will be discussed more in the bridging activity.

Bridging Activity - Programming (10 min)

Previewing Online Puzzles as a Class

Pull a puzzle from the corresponding online stage. We recommend puzzle 7. Have students discuss a pattern that they think will get Laurel the Adventurer to collect all the treasure. Ask the students to share. See how many other students had the same answer!

Main Activity (30 min)

Laurel the Adventurer is looking to collect as much treasure as she can. Instruct the students to traverse the puzzle to collect whatever they can. Some levels will require you to only pick up one piece of treasure, but others will require you to pick up every piece of treasure. Pay attention to the instructions to know what to do!

Code Studio levels

The Collector 1 ***(click tabs to see student view)***

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

Challenge 8 ***(click tabs to see student view)***

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

Predict 12 ***(click tabs to see student view)***

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

Lesson Extras Extra Extra ***(click tabs to see student view)***

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?
- Draw a maze that you might solve with the blocks you used today.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 7: Creating Art with Code

Overview

In this lesson, students will take control of the Artist to complete drawings on the screen. This Artist stage will allow students to create images of increasing complexity using new blocks like `move forward by 100 pixels` and `turn right by 90 degrees`.

Purpose

Building off of the students' previous experience with sequencing, this lesson will work to inspire more creativity with coding. The purpose of this lesson is to solidify knowledge on sequencing by introducing new blocks and goals. In this case, students learn more about pixels and angles using the new blocks, while still practicing their sequencing skills. Also, students will be able to visualize new goals such as coding the Artist to draw a square.

Agenda

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

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

Wrap Up (10 - 15 min)
- **Journaling**

Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Create a program to complete an image using sequential steps.
- Break complex shapes into simple parts.

Preparation

☐ Play through the puzzles to find any potential problem areas for your class.
☐ (Optional) Obtain protractors for your class to visualize the angles they must use to complete the puzzles.
☐ Print one **Turns & Angles - Student Handout** for each student.

Links

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

For the Students

- **Artist Introduction** - Student Video
- **Turns & Angles** - Student Video
- **Turns & Angles** - Student Handout
  Make a Copy ▾

Teaching Guide

Warm Up (10 min)

Introduction

Show the students one or both of the following videos as an introduction to angles:

Artist Introduction - Student Video (1.5 minutes long)

Turns & Angles - Student Video (2 minutes long)

Use **Turns & Angles - Student Handout** to show the students interior versus exterior angles for different shapes. This document can be used as a hand out or you can choose to print it out as a poster for students to refer to.

Ask:

Discuss the square and triangle shapes from the document.

- How would you code a computer to draw that shape?
- What order do the instructions need to be in?

Tell the students that in these puzzles they will be moving a character who leaves a line everywhere he goes. The students will be writing code that gets the character to draw various shapes, including a square.

Main Activity (30 min)

Online Puzzles

In this set of puzzles, the artist will no longer be constrained to 90 degree angles. Having physical protractors available can be help students better visualize the angles they need. Otherwise, the stage provides images of the angles as the student selects which angle to use. (Please note: Angle choices are limited to two inside of the dropdown menu, reducing the number of options students have to work through.)

Before sending the students to the computers to work on the puzzles, it might be beneficial to give a brief presentation of how to use the tools in this level. We recommend puzzle 5 as a good puzzle to show how to use the protractor online.

Code Studio levels

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

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

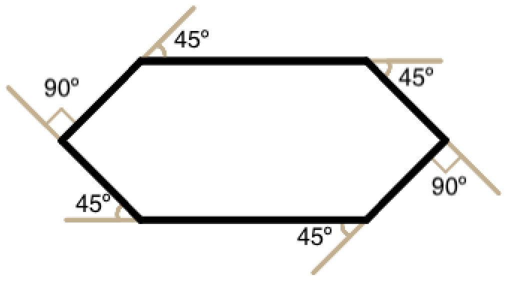
Challenge 8 ***(click tabs to see student view)***

Practice 9 ***(click tabs to see student view)***

Predict 10 ***(click tabs to see student view)***

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

The eighth puzzle asks the students to draw a 6 sided polygon. This might be challenging for some students. We recommend getting the students to try a few times, ask a peer, then ask the teacher for help. Below is an image that might be helpful for the students.



Wrap Up (10 - 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 are the interior angles that make up a square. What about for a triangle?
- Sketch a simple shape on your paper and imagine the code used to draw it. Can you write that code out next to the shape?

Extended Learning

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

The Copy Machine

- Give students two pieces of paper
- On one sheet draw a simple image, using straight lines only.
- On the second sheet draw instructions for recreating that image commands to move straight and turn at various angles.
- Trade instruction sheets and attempt to recreate the image using only the provided instructions.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **AP** - Algorithms & Programming

This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 8: Binary Bracelets

Overview

Binary is extremely important in the world of computers. The majority of computers today store all sorts of information in binary form. This lesson helps demonstrate how it is possible to take something from real life and translate it into a series of ons and offs.

Purpose

In this lesson students will learn how information is represented in a way such that a computer can interpret and store it. When learning *binary*, students will have the opportunity to write codes and share them with peers as secret messages. This can then be related back to how computers read a program, translate it to binary, use the information in some way, then reply back in a way humans can understand. For example, when we type a sentence into a document then press save, a computer translates the sentence into binary, stores the information, then posts a message indicating the document has been saved.

Agenda

Warm Up (15 min)
- **Vocabulary**
- **Off and On**

Main Activity (20 min)
- **Binary Bracelets - Worksheet**

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

Assessment (15 min)

Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Encode letters into binary.
- Decode binary back to letters.
- Relate the idea of storing letters on paper to the idea of storing information in a computer.

Preparation

☐(Optional) Watch the "Lesson in Action" video.
☐Gather markers for the bracelets. Other decorations like beads and pipecleaners are optional.
☐Print one **Binary Bracelets - Worksheet** and one **Binary Bracelets - Assessment** per student.
☐Make sure every student has a journal.
☐(Optional) Write a short message on the board in binary.

Links

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

For the Teachers

- **Binary Bracelets** - Lesson in Action Video
- **Binary Bracelets** - Assessment Answer Key
  Make a Copy ▾

For the Students

- **Bits Versus Bytes** - Student Video
- **Binary Bracelets** - Unplugged Video (**download**)
- **Binary Bracelets** - Worksheet
  Make a Copy ▾
- **Binary Bracelets** - Assessment
  Make a Copy ▾

Vocabulary

- **Binary** - A way of representing information using only two options.

Teaching Guide

Warm Up (15 min)

Vocabulary

This lesson has one new and important word:

Binary - Say it with me: Bye-nair-ee

A way of representing information using only two options

Off and On

- If you've written a short message on the board in binary, call the students' attention to it and ask if anyone knows what it is or what it means.
    - Put the message aside and move on to prepping for the activity.
- You can start by asking the class if they have ever seen inside a computer.
    - What's in there?
    - This is a good place to actually show them the inside of a computer (or pictures of the inside of a computer).

- Wires carry information through the machine in the form of electricity.
    - The two options that a computer uses with respect to this electrical information are "off" and "on." Just like the lights in this room!
        - When computers represent information using only two options, it's called "Binary."
    - That theme of two options doesn't stop when the information gets to its destination.

- Computers also *store* information using binary.
  - Binary isn't always off and on.
    - Hard Disk Drives store information using magnetic positive and magnetic negative.
    - DVDs store information as either reflective or non-reflective.
  - How do you suppose we can convert real-life things that we want to store in a computer into binary?
    - Let's start with letters.
    - Use the **Binary Bracelets - Worksheet** to show how a computer might represent capital letters.
      - This is a good time to mention that each spot where you have a binary option is called a "binary digit" or "bit" for short.
      - Ask if anyone knows what a grouping of eight bits is called (it's a byte.)
      - Fun fact: A grouping of four bits is called a nibble.
      - Watch the **Bits Versus Bytes - Student Video** (~1 minute)
    - Go over a few examples of converting letters into binary, then back.
    - Afterward, write an encoded letter and give the class a few seconds to figure out what it is.
    - When the class can figure out that encoded letter on their own, you can move on to the activity.

Main Activity (20 min)

Binary Bracelets - Worksheet

You do not need to cover the whole of binary, like counting and converting numbers back and forth from decimal. This lesson is intended to be a fun introduction to how computers store information, not a frustrating lesson in bases.

Lesson Tip

You know your classroom best. As the teacher, decide if students should do this individually or if students should work in pairs or small groups.

Directions:

- Find the first letter of your first name on the activity sheet.
- Fill in the squares of a bracelet to match the pattern of the squares next to the letter that you selected.
- Cut the bracelet out.
- Tape the bracelet around your wrist to wear it!
- Share your bracelet with your classmates to see if they can figure out your letter.

A	■□■■	■■■□	N	■□■■	□□□■
B	■□■■	■■□■	O	■□■■	□□□□
C	■□■■	■■□□	P	■□■□	■■■■
D	■□■■	■□■■	Q	■□■□	■■■□
E	■□■■	■□■□	R	■□■□	■■□■
F	■□■■	■□□■	S	■□■□	■■□□
G	■□■■	■□□□	T	■□■□	■□■■
H	■□■■	□■■■	U	■□■□	■□■□
I	■□■■	□■■□	V	■□■□	■□□■
J	■□■■	□■□■	W	■□■□	■□□□
K	■□■■	□■□□	X	■□■□	□■■■
L	■□■■	□□■■	Y	■□■□	□■■□
M	■□■■	□□■□	Z	■□■□	□■□■

After the activity, revisit the message that was on the board and see if your class can decypher it using what they've learned.

Wrap Up (5 min)

Flash Chat: What did we learn?

- What else do you think is represented as binary inside of a computer?
- How else might you represent binary instead of boxes that are filled or not filled?
- What was your favorite part about that activity?

Lesson Tip

If your class has extra budget for materials, try doing this exercise using thread (or pipe cleaners) and beads to create the binary bracelets instead of pen and paper. You can provide any combination of two colors in beads to the students, but black and white tend to be easiest, given the way that the key is done.

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?
- Use the activity worksheet to write out the rest of your name or your favorite word in binary.
- Imagine a world where we spoke in binary, saying "on" or "off", but nothing else. Draw two characters trying to talk to each other in binary.

Assessment (15 min)

- Hand out the **Binary Bracelets - Assessment** and allow students to complete it 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.

Binary Images

- There are several great resources on the web for taking this activity to the next level.
- If your students are interested in how images (or even music) can be represented as binary, you can find more details in Thinkersmith's **Binary Baubles**.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **AP** - Algorithms & Programming

This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 9: My Loopy Robotic Friends Jr.

Overview

Building on the initial "My Robotic Friends" activity, students tackle larger and more complicated designs. In order to program their "robots" to complete these bigger designs, students will need to identify repeated patterns in their instructions that could be replaced with a loop.

Purpose

This lesson serves as a reintroduction to loops, using the now familiar set of "robot" programming instructions. Students will develop critical thinking skills by looking for patterns of repetition in the movements of classmates and determining how to simplify those repeated patterns using loops.

Agenda

Warm Up (10 min)

My Robotic Friends Review

Activity (30 min)

Introduction and Modeling
Looping Your Robots

Wrap Up (5 min)
Extension Activities

View on Code Studio

Objectives

Students will be able to:

- Identify repeated patterns in code that could be replaced with a loop
- Write instructions that use loops to repeat patterns

Preparation

- [ ] Make sure each student has a journal.
- [ ] Prepare a stack of 20 paper cups (or paper trapezoids) for each group of 4 students.
- [ ] Display the symbols where students can reference throughout the lesson.
- [ ] (Optional) Print out one image pack per group of 4 students.

Links

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

For the Students

- **My Robotic Friends** - Cup Stacking Ideas Make a Copy
- **My Robotic Friends** - Cup Spacing Make a Copy
- **My Robotic Friends** - Symbol Key Make a Copy
- **My Robotic Friends** - Paper Trapezoid Template Make a Copy

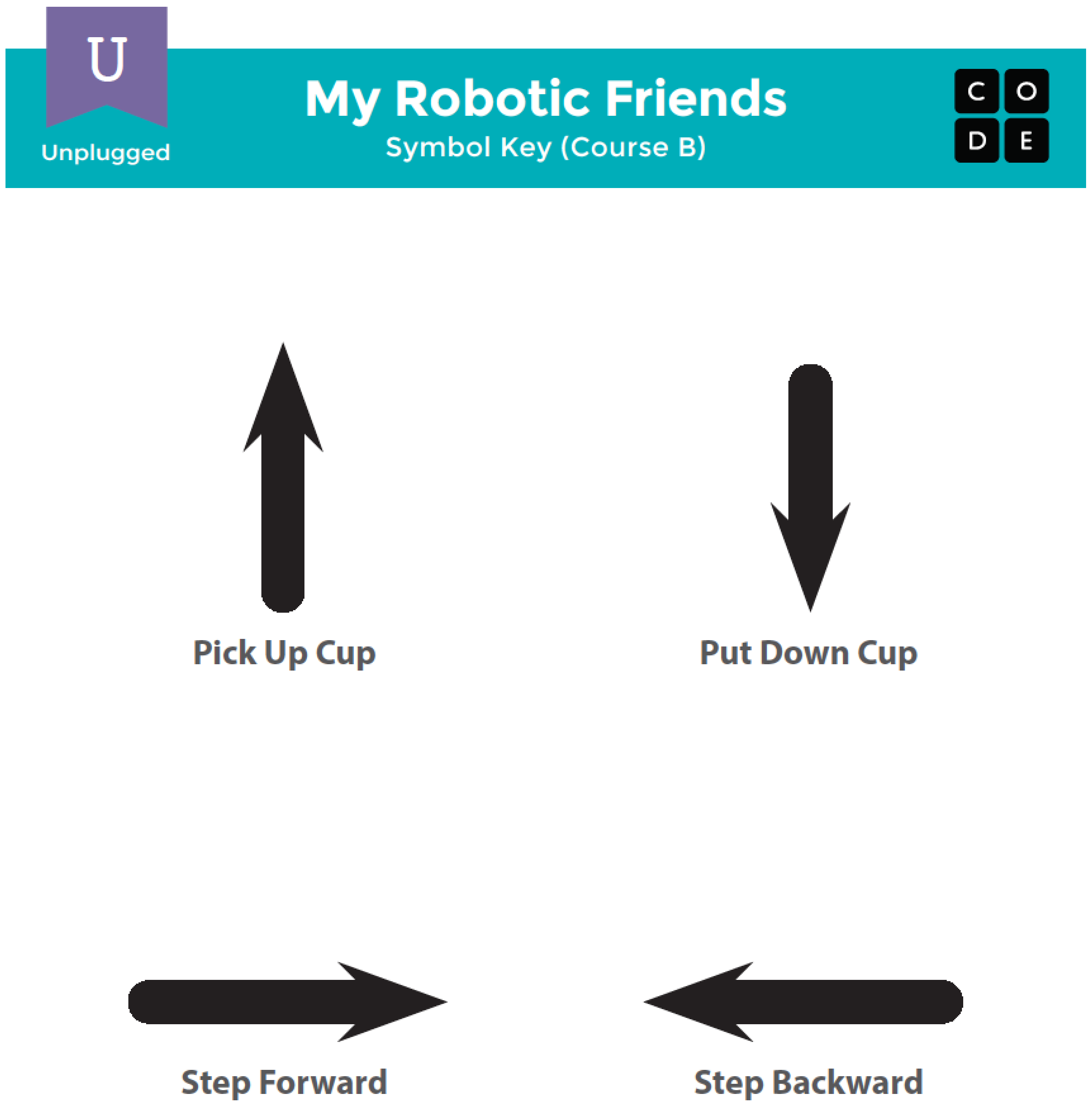
Vocabulary

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

Teaching Guide

Warm Up (10 min)

My Robotic Friends Review

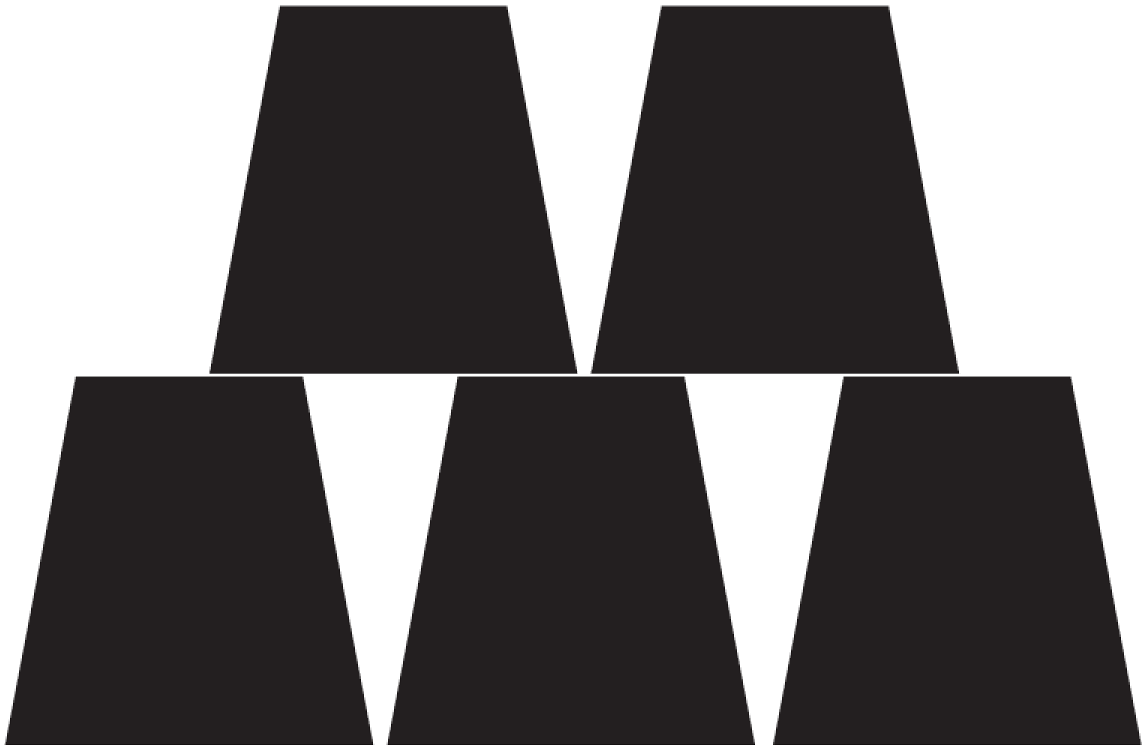


Goal: This review will refresh the students' minds about how quickly programs for the "My Robotic Friends" activity can get intense.

Display: Show the **My Robotic Friends - Symbol Key** that we used in My Robotic Friends. For each of the four symbols, ask students to show you what it looks like for a robot to follow that instruction.

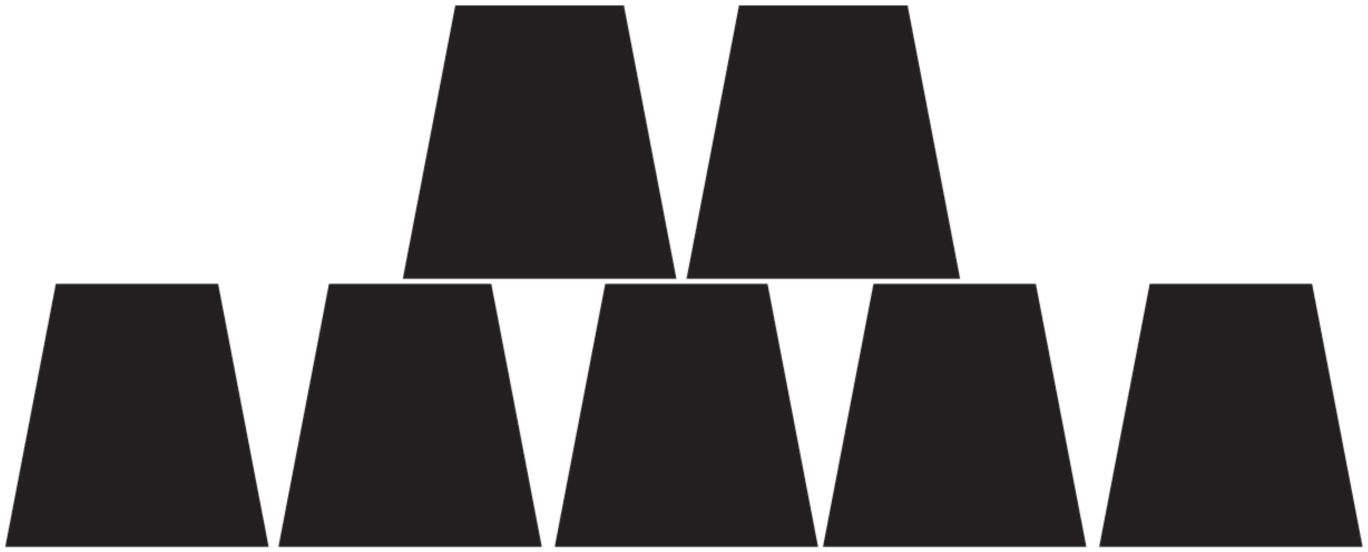
Model: With the class together as a group, pull an easy puzzle from the "My Robotic Friends" Cup Stack Pack and program with each other as a reminder of rules and terminology.

Next, pull a puzzle that's slightly harder, but also requires a lot of steps like the one below.



Volunteer: Ask a volunteer (or a group of volunteers) to come forward to help program this one on the board. If you make them stick strictly to the "no symbols other than those on the key" rule, it will probably take a while!

Display: Now, bring up this image:



What is the reaction of the class?

Prompt: Give students the opportunity to brainstorm shorter ways to relay the code that they're about to create. (This bit can be skipped over if your students start saying things like: "Move forward 6 times." Since that will open the discussion about how to show "six times" with symbols.)

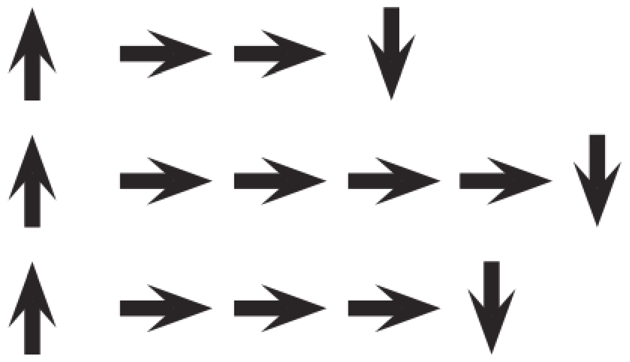
Once students have put together the idea of “repeating” code, give them the vocabulary around it. Make sure to share with them that often the terms “repeat something” and “loop something” are often used interchangeably.

Activity (30 min)

Introduction and Modeling

Set Up: Have stacks of cups or cut paper trapezoids available for groups.

Display: Take the program from one of your previous cup stacks and display it for the class, or use the one below.

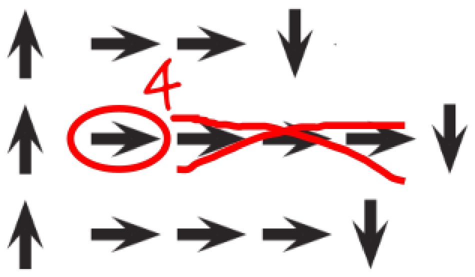


Think: Ask students to think quietly about where in this program they can find a pattern of instructions that repeat uninterrupted (one repetition after another).

Pair: Turn to a neighbor and share one of the repeating patterns you found.

Share: Ask a few students to share out the patterns they identified. Try to pull out different approaches to grouping patterns. For each pattern, ask students to identify how many times the pattern repeats.

Model: Using one of the repeating patterns that the class identified, model how Circle the instruction or pattern that repeats, write the number of loops near that circle, then cross out the rest of the arrows.



Repeat this until the entire program has been shortened, then re-write the program in a way where students can see how much more simple the resulting instructions are.

Looping Your Robots

Group: Place students into groups of 4. Each group should then further break down into two pairs - each pair will develop their own program "run" on the other pair.

Distribute: Give each group one stack of cups or paper cutouts.

Display: Show **My Robotic Friends - Cup Stacking Ideas** to the class or hand out individual copies for groups to use. Have each pair (not group) choose which stack they would like their robot to do. Encourage pairs to select a more complicated pattern this time around.

Discuss: Let each group discuss how the stack should be built, then instruct each group to translate the algorithm into symbols. Make sure each group writes down the symbol algorithm somewhere for the "robot" to read later. As students are working on their programs, remind them to be on the lookout for opportunities to replace a repeating pattern with a loop.

Do: When groups have finished their instructions, have each pair trade with another pair to run one another's code. Remind students to be on the lookout for bugs, but not to interrupt a robot until it's finished running the program.

Discuss: When all of the pairs have had a chance to run their programs, ask a few to share their solutions with the class. Use this opportunity to discuss how groups came up with different solutions to the same puzzle. In particular, you might ask of each program:

- How did they identify the loops?
- Are there other ways those loops could have been written?
- How much shorter is the program with loops than it would be without?
- Is the program easier to understand with loops, or written out longhand? Why?

Teaching Tip

Looking for Loops: Be sure to keep your eyes open for students using loops. Try to avoid correcting their overall algorithms or prescribing a solution, but feel free to direct students towards patterns that could be shortened by using a repeat circle.

Watch students as they run through the code. Are there any bugs? Use the debugging questions to help them find a solution.

- What does it do?
- What is it supposed to do?
- What does that tell you?
- Does it work at the first step?
- Does it work at the second step?
- Where does it stop working?

Wrap Up (5 min)

Journal Prompts:

- Draw one of the "Feeling FaceS" that shows how you felt about today's lesson in the corner of your journal page.
- Have the students write or draw something in their journal that will remind them later what loops are. This can come from a prompt like:
  - What does "repeat" mean to you?
  - Draw a picture of you repeating something.

Extension Activities

- Have students draw their own cup stacking creations for someone else to code.
- Provide students with algorithms that utilize repeats, then have them expand the program back out to a full step-by-step version.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **AP** - Algorithms & Programming

cc BY NC SA
This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 10: Loops with Rey and BB-8

Overview

Building on the concept of repeating instructions from "Getting Loopy," this stage will have students using loops to help BB-8 traverse a maze more efficiently than before.

Purpose

In this lesson, students will be learning more about *loops* and how to implement them in Blockly code. Using *loops* is an important skill in programming because manually repeating commands is tedious and inefficient. With the Code.org puzzles, students will learn to add instructions to existing loops, gather repeated code into loops, and recognize patterns that need to be looped. It should be noted that students will face puzzles with many different solutions. This will open up discussions on the various ways to solve puzzles with advantages and disadvantages to each approach.

Agenda

- **Warm Up (5 min)**
  - **Introduction**
- **Bridging Activity - Loops (10 min)**
  - **Unplugged Activity Using Paper Blocks**
  - **Previewing Online Puzzles as a Class**
- **Main Activity (30 min)**
  - **Online Puzzles**
- **Wrap Up (5 - 10 min)**
  - **Journaling**
- **Extended Learning**

View on Code Studio

Objectives

Students will be able to:

- Identify the benefits of using a loop structure instead of manual repetition.
- Break down a long sequence of instructions into the largest repeatable sequence.
- Employ a combination of sequential and looped commands to reach the end of a maze.

Preparation

- [ ] Play through the puzzles to determine if there will be any problem areas for your 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 Students

- **Pair Programming** - Student Video
- **Unplugged Blocks (Courses C-F)** - Manipulatives

Vocabulary

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

Teaching Guide

Warm Up (5 min)

Introduction

Review with students the "My Loopy Robotic Friends" activity:

- What are loops?
- Why do we use them?

Quickly show the students a program (with repeated steps) for the rest of the class to do. Ask the rest of the class to find the loops within the program and point them out.

If you're comfortable, give an introduction to BB-8 from Star Wars. Many children may be familiar with the lovable robot, but an introduction will surely build excitement.

Bridging Activity - Loops (10 min)

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

Unplugged Activity Using Paper Blocks

Revisit "My Loopy Robotic Friends". This time, work with the class to "code" it out using **Unplugged Blocks (Courses C-F) - Manipulatives** instead of writing the instructions on paper. Make sure the students know that the blocks need to go from top to bottom and they all need to touch!

Previewing Online Puzzles as a Class

Pull up the online puzzles and choose a puzzle to do in front of the class. We recommend puzzle 10 for its staircase pattern. Ask the students to write a program to solve the puzzle on paper. Have the students circle repeated chunks and label with the number of repeats, the same way they did in "My Loopy Robotic Friends."

Main Activity (30 min)

Online Puzzles

Code Studio levels

Programming with Rey and BB-8 1 ***(click tabs to see student view)***

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

Predict 3 ***(click tabs to see student view)***

Repeat Blocks with BB-8 4 ***(click tabs to see student view)***

Practice 5 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)*

Predict 13 *(click tabs to see student view)*

Practice 14 *(click tabs to see student view)*

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

As students work through the puzzles, see if they can figure out how many blocks they use with a loop vs. not using a loop. **Pair Programming - Student Video** works really well with this set of puzzles because there are a few ways to fill the loops. Push for friendly discussion between pairs in instances of disagreement on how to solve the puzzle. Have the students ask each other questions like:

- How did you come up with that solution?
- What are some benefits of solving the puzzle that way?

We also recommend having paper on hand for students to write out their code and find any repetition to use in loops.

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?
- How did loops make your program easier to write?
- Think of something that repeats over and over again. What might the program for that look like?

Extended Learning

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

So Moving

- Give the students pictures of actions or dance moves that they can do.
- Have students arrange moves and add loops to choreograph their own dance.
- Share the dances with the rest of the class.

Connect It Back

- Find some YouTube videos of popular dances that repeat themselves.
- Can your class find the loops?
- Try the same thing with songs!

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 11: Harvesting Crops with Loops

Overview

In the preceding stage, students used loops to create fantastic drawings. Now they're going to loop new actions in order to help the harvester collect multiple veggies growing in large bunches.

Purpose

It may seem unnecessarily repetitive to have two plugged stages introducing loops, but the practice of using loops for different reasons develops a student's understanding of what loops can do. In "Loops in Maze" students only used loops to repeat movements. In this lesson, students will use loops to repeat other actions like harvesting pumpkins. New patterns will emerge and students will use creativity and logical thinking to determine what code needs to be repeated and how many times.

Agenda

View on Code Studio

Objectives

Students will be able to:

- Write a program for a given task which loops a single command.
- Identify when a loop can be used to simplify a repetitive action.
- Employ a combination of sequential and looped commands to move and perform actions.

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** - To do something again.

Teaching Guide

Warm Up (5 - 10 min)

Introduction

At this point, students have only used loops in the Maze puzzles. Those puzzles focused on looping movement instructions. This stage will use loops to pick up multiple items in the same spot.

Ask:

- What are some other elements of our programs that could benefit from loops?
- Harvester puzzles can have lots of items in one spot. How can we use loops to pick up any number of them with just two blocks?

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 8 ***(click tabs to see student view)***

Challenge 9 ***(click tabs to see student view)***

Practice 10 11 12 ***(click tabs to see student view)***

Predict 13 ***(click tabs to see student view)***

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

When students are using loops to repeat an action (such as harvesting pumpkins), encourage them to think about the movements before and after that action. Could those actions be brought into the loop as well?

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 did you feel during today’s lesson?
- Give two examples of when you used loops in your code.
- What else could a farmer harvest? Draw the code block that you would need to harvest that item.

Extended Activity

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

Harvesting Supplies

- Let students create piles of school supplies at their desks (pencils, erasers, etc.)
- Have their partners figure out how a harvester would walk from pile to pile, collecting each group of items along the way.
- Share the programs with the rest of the class.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 12: Looking Ahead with Minecraft

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** - Something a program checks to see if it is true before allowing an action.
- **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

(click tabs to see student view)

Challenge 11 12 ***(click tabs to see student view)***

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

Free Play 14 ***(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 avoid lava, pick 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

This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 13: Sticker Art 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)
- **Application - Sticker Art**

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

- [ ] Practice making your own design. Make note of how these levels are different from everything that came before.
- [ ] Make sure every student has a journal.

Vocabulary

- **Loop** - The action of doing something over and over again.
- **Repeat** - To 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 dance with loops just like in "Getting Loopy"
- As a class, playing through a puzzle from the last lesson, "Loops with Rey and BB-8"
- Reviewing how to use Artist by playing through a puzzle from "Programming in Artist"
- 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)

Application - Sticker Art

Encourage creativity during this activity! The instructions give examples but it is okay for students to experiment with different stickers or types of designs. Even the example solutions we provide are not the *only* solutions!

Teaching Tip

This lesson may feel very different from what has come earlier in the course. These levels have some new characteristics you may want to explore before starting. The code your students write in one level will automatically transfer over to the others. This allows them to build gradually and iterate on their ideas as they learn. Note that these levels are not checked for correctness to allow for more open-ended creativity. Empower your students to determine for themselves when they have completed each task.

Code Studio levels

Mini-project: Sticker Design 1 2 3 4 5 6 7 8 ***(click tabs to see student view)***

Free Play 9 ***(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

This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 14: The Big Event

Overview

Students will soon learn that events are a great way to add flexibility to a pre-written algorithm. Sometimes you want your program to be able to respond to the user exactly when the user wants it to. Events can make your program more interesting and interactive.

Purpose

Today, students will learn to distinguish events and actions. The students will see activities interrupted by having a "button" pressed on a paper remote. When seeing this *event*, the class will react with a unique action. Events are widely used in programming and should be easily recognizable after this lesson.

Agenda

View on Code Studio

Objectives

Students will be able to:

- Repeat commands given by an instructor.
- Recognize movements of the teacher as signals to initiate commands.
- Practice differentiating pre-defined actions and event-driven ones.

Preparation

- ☐ Print or display the Event Controller.
- ☐ Print one assessment for each student.
- ☐ 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

- **The Big Event** - Assessment Answer Key
  Make a Copy ▾

For the Students

- **The Big Event** - Unplugged Video (**download**)
- **The Big Event (Course C)** - Event Controller
  Make a Copy ▾
- **The Big Event** - Assessment
  Make a Copy ▾

Vocabulary

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

Teaching Guide

Warm Up (15 min)

Vocabulary

This lesson has one new and important vocabulary word:

Event - Say it with me: E-vent

An event is an action that causes something to happen.

A Series of Events

- Prep your class to answer a question:
  - "I'm going to ask you a question. I want you to raise your hand if you want me to call on you for the answer."
  - Ask a simple question that most of your students should be able to answer, such as:
    - How many thumbs do I have?
    - What is bigger, a bird or a horse?
  - Call on a student who has their hand raised and let them give their answer.
  - Upon finishing that display, ask the class how you knew that the student wanted you to call on them.
    - Your class will likely mention the raising of the hand.
  - Explain to everyone that when students raise their hand, it is an "event" that causes you to know that they want to be called on.
- Ask the class if they can think of any other events that give signals.
  - You may need to remind them that you're not talking about an event like a birthday party or a field trip.
  - If they have trouble, you can remind them that an event is an action that causes something to happen.
    - What about an alarm clock going off? What does that make happen?
    - What about pressing "Start" on the microwave? What does that do?
    - What about pressing the power button on your tv remote?
- Today, we're going to practice changing programs by introducing events.

Main Activity (15 min)

- Do you remember guiding Red from Angry Birds to the pig in the Maze puzzles?
  - In that exercise, you knew in advance exactly where you wanted Red to go, so you could make a program that took Red from start to finish without any interruptions.
  - In most real programs, we can't do that because we want to have options, depending on what the user needs.
    - Say that I only want my character to move when my finger is on the screen of my phone. I would need to program the character to only move when I put my finger on the screen of my phone.
    - Putting my finger on the screen would then become an "event" that tells my character to move.

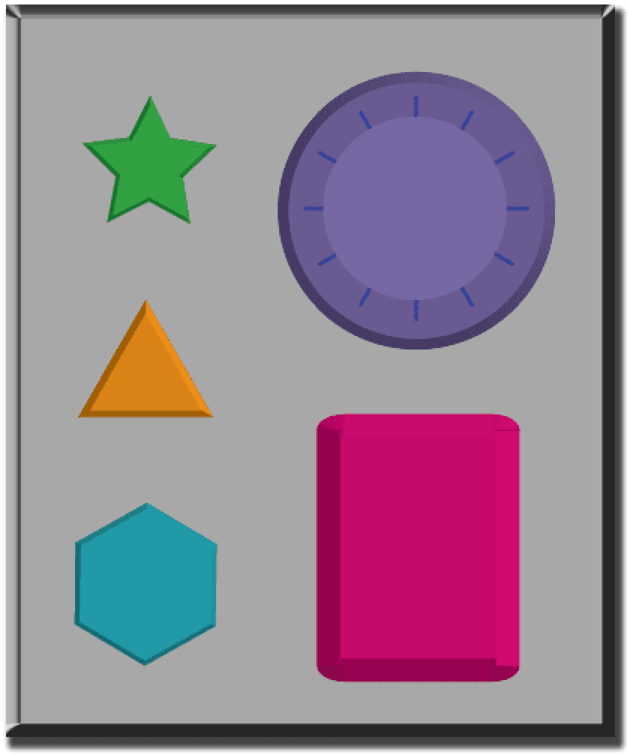
Lesson Tip

If your students seem confused, talk about their favorite games and all of the ways that they let the characters know what they're supposed to do. Point out how the game would be really boring if it ran from start to finish without any events required.

In earlier lessons, we created algorithms that allowed us to control a friend or other character for several steps at a time. It was fun and useful, but what happens when you don’t know everything that you want your friend to do in advance? This is where events come in!

Directions:

- Project the Event Controller onto your classroom screen.



- Decide with your class what each button does. We suggest:
  - Pink Button -> Say “Wooooo!”
  - Teal Button -> “Yeah!”
  - Purple Dial -> “Boom!”
  - Green Button -> Clap
  - Orange Dial -> Stomp
- Practice tapping the buttons on the overhead and having your class react.
- Add some button sequences into the mix and have the students try to keep up with their sounds.
- Let your class know that every time you push a button, it is an “event” that lets them know what they are expected to do next.
- Get the class started on a planned task before interrupting them again with the buttons. We suggest:
  - Counting to 10
  - Singing “Old MacDonald”
- Once their plan is underway, interject button presses sporadically.
- Continue the blend until they understand the difference between actions that are guided by a plan and those that are event driven.

Wrap Up (10 min)

Flash Chat: What did we learn?

- Why do we need to be able to handle events in a program?
- What are some other kinds of events that you can think of?

Assessment (10 min)

- Hand out **The Big Event - Assessment** 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.

One Person's Event is Another One's Reaction

- Assign each student an event to watch out for, and an appropriate reaction to that event. Chain the actions so that each child's reaction becomes an event that triggers the reaction of another student. Keep assigning until everyone has something to do and everyone makes someone react.

Eventopalooza

- Break the class up into groups. Using the Events Controller, assign each group a different reaction to the same button. Do this for all three buttons, then watch the chaos!

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 15: Build a Flappy Game

Overview

In this special stage, students get to build their own Flappy Bird game by using event handlers to detect mouse clicks and object collisions. At the end of the level, students will be able to customize their game by changing the visuals or rules.

Purpose

Events are very common in computer programs. In this lesson, students will further develop their understanding of events by making a Flappy Bird game. Students will learn to make their character move across the screen, make noises, and react to obstacles based on user-initiated events.

Agenda

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

Bridging Activity - Events (10 min)
- **Unplugged Activity Using Paper Blocks**
- **Preview of Online Puzzles as a Class**

Main Activity (30 min)
- **Online Puzzles and Free Play**

Wrap Up (10 - 15 min)
- **Journaling**

Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Match blocks with the appropriate event handler.
- Create a game using event handlers.
- Share a creative artifact with other students.

Preparation

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

- **The Big Event (Course C)** - Event Controller
  Make a Copy ▾
- **Unplugged Blocks (Courses C-F)** - Manipulatives

Vocabulary

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

Teaching Guide

Warm Up (10 min)

Introduction

- Review "The Big Event" activity with students:
  - What did we "program" the button click events to do?
- Now we're going to add events to our coding. Specifically, we're going to create an event for clicking the mouse and one for when the bird hits an object like the ground or an obstacle. When have you seen a character touch another object as an event in games?

Lesson Tip

Students will have the opportunity to share their final product with a link. This is a great opportunity to show your school community the great things your students are doing. Monitor and collect all of the links to the projects and keep them on your class website for all to see!

Bridging Activity - Events (10 min)

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

Unplugged Activity Using Paper Blocks

Using the remote from the **The Big Event (Course C) - Event Controller** and **Unplugged Blocks (Courses C-F) - Manipulatives**, gather your class to reprise the activity from the previous lesson. Ask the class "when the teal button is pushed, what do we do?" then fill in one of the `when` event blocks and one of the blue action blocks accordingly. Make sure that the students understand that the `when` blocks need to be on top of the blue block and they need to touch in order for the program to run.

Preview of Online Puzzles as a Class

Pull a lesson from the corresponding online stage, we recommend puzzle 2. Ask the students what should happen when the Flappy Bird runs into something like the ground or an obstacle. Explain that Flappy in this game will move forward with a click of the mouse and the game will end if Flappy runs into anything.

Complete the puzzle with the class and allow time for a quick discussion on what was and wasn't an event. For every event, ask the students what the action corresponding to this event is.

Main Activity (30 min)

Online Puzzles and Free Play

Code Studio levels

Code Your Own Flappy Game 1 ***(click tabs to see student view)***

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

Mini-project: Flappy Game 6 7 8 9 10 11

(click tabs to see student view)

This entire lesson builds towards students creating their own Flappy game, but the second half of the levels are all linked. This mean students will see their own code as they move between bubbles 6-10, allowing for more personal customization that will carry all the way to the final product.

Teacher Tip

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**.

In the final stage of this lesson students are able to tweak their game to make it unique - encourage them to see how different they can make each game within the constraints provided. If the class doesn't use pair programming, then tell students to go around and look at other student's games. Otherwise, have students discuss and try out different ways to set up their game with their partner.

Wrap Up (10 - 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 did you do to make your game unique?
- Draw out a game you want to make in the future.

Extended Learning

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

Look Under the Hood

When you share a link to your game, you also share all of the code that goes behind it. This is a great way for students to learn from each other.

- Post links to completed games online or on the board.
  - Make a game of your own to share as well!
- When students load up a link, have them click the "How it Works" button to see the code behind the game.
- Discuss as a group the different ways your classmates coded their games.
  - What suprised you?
  - What would you like to try?
- Choose someone else's game and build on it. (Don't worry; the original game will be safe.)

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **AP** - Algorithms & Programming

This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 16: Chase Game with Events

Overview

In this online activity, students will have the opportunity to learn how to use events in Play Lab and to apply all the coding skills they've learned to create an animated game. It's time to get creative and make a game in Play Lab!

Purpose

Here, students will further develop their understanding of events using Play Lab. Students will use events to make characters move around the screen, make noises, and change backgrounds based on user input. At the end of the puzzle sequence, students will be presented with the opportunity to share their projects.

Agenda

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

Main Activity (30 min)
- **Online Puzzles and Free Play**

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

Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Create an animated, interactive game using sequence and event-handlers.
- Identify actions that correlate to input events.

Preparation

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

- **Unplugged Blocks (Courses C-F)** - Manipulatives

Vocabulary

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

Teaching Guide

Warm Up (10 min)

Introduction

Briefly discuss the Flappy Bird game from the last lesson. Ask students to come up with the various events in the game. Events include:

- Flappy hitting the ground
- Flappy hitting an obstacle
- A player clicking the screen
- Flappy passing an obstacle

Now discuss the actions that correspond to these events. Flappy running into something ends the game, but Flappy passing an obstacle wins a point. A player clicking makes Flappy flap his wings.

Ask the students what other events and actions they'd like to see. What other kinds of games could be built around those events and actions?

Main Activity (30 min)

Online Puzzles and Free Play

This is the most free-form online activity of the course. At the final stage students have the freedom to create a game of their own. You may want to provide structured guidelines around what kind of game to make, particularly for students who are overwhelmed by too many options.

Lesson Tip

Students will have the opportunity to share their final product with a link. This is a great opportunity to show your school community the great things your students are doing. Collect all of the links and keep them on your class website for all to see!

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

Practice 1 2 3 4 5 ***(click tabs to see student view)***

Mini-project: Chase Game 6 7 8 9 10 11 ***(click tabs to see student view)***

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

Levels Extra ***(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 an event your program used today?
- Is there an event that would you like to have used in your game that you did not get to use in Play Lab?

Extended Learning

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

Look Under the Hood

When you share a link to your story, you also share all of the code that goes behind it. This is a great way for students to learn from each other.

- Post links to completed stories online.
  - Make a story of your own to share as well!
- When students load up a link, have them click the "How it Works" button to see the code behind the story.
- Discuss as a group the different ways your classmates coded their stories.
  - What surprised you?
  - What would you like to try?
- Choose someone else's story and click `Remix` to build on it. (Don't worry, the original story will be safe.)

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **AP** - Algorithms & Programming

This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 17: Picturing Data

Overview

Data can be used to help students understand their world and answer interesting questions. In this lesson, students will collect data from a Play Lab project and visualize it using different kinds of graphs.

Purpose

Computers were created to help process data. There is an increasing amount of data in the world, so being able to read and analyze it is important. This lesson is here to make sure students have the basic experience of collecting, visualizing, and analyzing a simple set of data.

Agenda

View on Code Studio

Objectives

Students will be able to:

- Collect and record data about quantities of real objects, or characters on a screen
- Create a bar graph and pie chart to represent simple data.
- Make comparisons between data visualizations made by others and use them to make a prediction.

Preparation

- [ ] Print out one **Graphing Data from Play Lab** worksheet for each student.
- [ ] Try today's lesson on Code Studio. This is meant to be used as a tool for today's activity. Be prepared to project it to the class, or otherwise allow students to visit it on their own computers.

Links

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

For the Teachers

- **Graphing Data from Play Lab** - Worksheet

Teaching Guide

Warm Up (5 - 10 min)

The Need for Visualization

Remarks

> In a moment I am going to ask each of you to tell me your favorite favorite season. On the count of 3, everyone should answer the question and then we will try to see if we can figure out which is the most popular season with the students in this class. Are you thinking of your favorite season? 1, 2, 3...

Students should all answer at once.

Discuss: Can we tell which season most of the class likes? How can we be sure? Encourage students to justify why they believe in their response.

Prompt: What could we do to be more certain that we know which season most people like? Take ideas from the class, but lead them towards writing the data down using tally marks on a board or chart paper.

Remarks

> Tally marks like these help us to keep track of everyone's answers, which is great. It still takes time to count them to know which option was chosen by the most or fewest students. Imagine if someone showed us tally marks for the entire school. It would take some time to count them!

Activity (35 min)

Graphing Data

Distribute: Pass out the Graphing Data worksheet along with pencils and coloring instruments of some kind. Direct students to Code Studio or project the level on the board.

This activity is broken into two parts. On the first page students use tally marks to keep track of how many times each pet appears in the program. On the second page they use a variety of graphs to visualize the data they collected.

Teaching Tip

This program generates a random assortment of pets each time it is run. If you are projecting the program for the whole class to use, consider running it multiple times and asking different groups of students to track the numbers each time so that there will be variety in their responses for later discussion.

Share: After finishing, encourage students can share their results with a neighbor and look for similarities and differences. When the whole class is ready, bring everyone together.

Discuss: Ask the class to discuss some of the following questions:

- Which graph shows you the most useful information? Why?
- Which pets did you see the most? Is that true for everyone?
- What do you think would happen if this program run for 100 animals? 1000?
- What questions could you answer with your data?
- What questions would you like to be able to answer?

Discussion Goal

The goal of this discussion is to get students thinking about the usefulness of data in identifying or answer questions, and to start them thinking about why they might want to use different kinds of visualizations (graphs in this case).

Wrap Up (5 min)

Journal: Collecting and graphing data can help us ask and answer interesting questions. In your journal, write down one question that you would like to answer and what data you might want to collect. For example, if you play soccer, you might want to know if goals are more often scored from the left, the right, or center - what data would you want to collect to answer that question? How would you show it?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **DA** - Data & Analysis

This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

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

Lesson 18: End of Course Project

Overview

This capstone lesson takes students through the process of designing, developing, and showcasing their own Play Lab projects! To ensure this process goes smoothly, we have provided a step-by-step structure for students to follow, from planning on paper to coding on our website. In addition, we offer ideas to help teachers facilitate a showcase finale!

Purpose

This lesson provides students with space to create a project of their own design, using a step-by-step process that requires planning but also allows for broad creativity.

Agenda

Warm Up (10 min)
- **Planning**

Main Activity (25 min)
- **Coding**

Wrap Up (10 min)
- **Showcase**

View on Code Studio

Objectives

Students will be able to:

- Use a planned design as a blueprint for creation.
- Overcome obstacles such as time constraints or bugs.

Preparation

- [ ] Play through the puzzles to find any potential problem areas for your class.
- [ ] Print out one **Play Lab Project Planning Guide** for each student (or pair).
- [ ] Review the **Play Lab Project Planning Guide (Exemplar)** document.
- [ ] (Optional) Complete your own planning guide and code your own project to show to students!

Links

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

For the Teachers

- **Play Lab Project Planning Guide (Exemplar)** - Lesson Resource Make a Copy

For the Students

- **Play Lab Project Planning Guide** - Worksheet Make a Copy

Teaching Guide

Warm Up (10 min)

Planning

Get students excited and ready for today's activity!

Remarks

> We have already had a chance to build a project in Play Lab. Today, this experience will be much more open-ended, so it will require planning beforehand! Planning is a very important part of coding a game or any other kind of software. So, before we jump onto computers, we will spend some time planning the projects we want to build.

Distribute: Distribute one **Play Lab Project Planning Guide** to each student or pair. With students, go over the steps listed on the guide, then allow them to complete it. Refer to the included exemplar if needed.

Teaching Tip

If students are pair programming for this assignment, this warm up is a great opportunity for them to practice sharing and respecting others' ideas. Ensure students are following group work norms you already have in place in your classroom. Otherwise, spend a brief moment going over your expectations.

Main Activity (25 min)

Coding

Equipped with their completed planning guides, students are now ready to bring their projects to life. These levels correspond to the structure of the planning guide, and help navigate students through the process of transforming their ideas into code.

Teaching Tip

Students will experience plenty of trial and error while coding. Their projects are likely to become truncated versions of their original scope. Remind students that this kind of compromise is common in software design. It's okay if they don't get to build in every feature they planned!

Code Studio levels

Project 1 2 3 4 5 6 7 ***(click tabs to see student view)***

Wrap Up (10 min)

Showcase

To celebrate students' work, spend the last 10 minutes or so allowing them to showcase their projects. This can be done in many ways, but here are a few:

- **Public Demo**: Select a few exemplary volunteers to briefly demo their projects in front of the class. As they do so, have them touch on what the planning-to-coding experience was like for them, including ideas they'd still like to implement.
- **Pair Playtesting**: Have students or groups pair up and playtest each other's projects. As they do, ask them to provide positive and constructive feedback to each other. The benefit here is that students will have the opportunity to provide and respond to feedback in a smaller setting.
- **Gallery Walk**: Ensure all students have their projects ready for testing. Have students move "musical chairs"-style to another computer and playtest the project there for a few minutes, until they receive a signal from you to move to another computer. Repeat this every few minutes. While there is less opportunity for structured communication here, this ensures students get to demo as many of their peers' projects as possible.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▸ **AP** - Algorithms & Programming

This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

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