

# Course D

Course D was created for students who read at roughly a third grade level. Angles and mathematical concepts are introduced with helpful videos and hints.

The course begins with a review of the concepts found in Courses A, B, and C. This review helps introduce or refresh basic ideas such as repeat loops and events. Students will develop their understanding of algorithms, nested loops, while loops, conditionals, and events. Lessons on digital citizenship are also included. This course is crafted to build a strong foundation of basic concepts before opening up to a wide range of new and exciting topics.

## Journaling

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

*Think Spot Journal* **Student Handout**

## Debugging

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

*Debugging Guide* **Student Handout**

## Chapter 1: Sequencing

### Lesson 1: Graph Paper Programming

Unplugged | Sequencing

In this lesson, you will program your classmate to draw pictures!

### Lesson 2: Introduction to Online Puzzles

Skill Building | Sequencing

This lesson will give you practice in the skills you will need for this course.

### Lesson 3: Relay Programming

Unplugged | Sequencing

Remember at the beginning of the course when you made drawings with code? In this lesson, you will be working with a team to do something very similar!

### Lesson 4: Debugging with Laurel

Skill Building | Sequencing

Have you ever run into problems while coding? In this lesson, you will learn about the secrets of debugging. Debugging is the process of finding and fixing problems in your code.

# Chapter Commentary

Sequencing

## Chapter 2: Events

### Lesson 5: Events in Bounce

Skill Building | Events

Ever wish you could play video games in school? In this lesson, you will get to make your own!

### Lesson 6: Build a Star Wars Game

Skill Building | Events

Feel the force as you build your own Star Wars game in this lesson.

### Lesson 7: Dance Party

End of Course Project

Time to celebrate! In this lesson, you will program your own interactive dance party.

# Chapter Commentary

Events

## Chapter 3: Loops

### Lesson 8: Loops in Ice Age

Skill Building | Loops

In this lesson you'll use the repeat block to help Scrat reach the acorn as efficiently as possible.

### Lesson 9: Drawing Shapes with Loops

Skill Building | Loops

In this lesson, loops make it easy to make cool images with the Artist!

### Lesson 10: Nested Loops in Maze

Skill Building | Loops

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

# Chapter Commentary

## Chapter 4: Conditionals

### Lesson 11: Conditionals with Cards

Unplugged | Conditionals

It's time to play a game where you earn points only under certain conditions!

### Lesson 12: If/Else with Bee

Skill Building | Conditionals

Now that you understand conditionals, it's time to program Bee to use them when collecting honey and nectar.

### Lesson 13: While Loops in Farmer

Skill Building | Conditionals

Loops are so useful in coding. This lesson will teach you about a new kind of loop: while loops!

### Lesson 14: Until Loops in Maze

Skill Building | Conditionals

You can do some amazing things when you use `until` loops!

### Lesson 15: Harvesting with Conditionals

Skill Building | Conditionals

It's not always clear when to use each conditional. This lesson will help you get practice deciding what to do.

## Chapter Commentary

Conditionals

---

## Chapter 5: Binary

### Lesson 16: Binary Images

Unplugged | Binary

Learn how computers store pictures using a language with only two options.

### Lesson 17: Binary Images with Artist

Skill Building | Binary

In this lesson, you will learn how to make images using only 0s and 1s.

## Chapter Commentary

## **Chapter 6: Digital Citizenship**

### **Lesson 18: Be A Super Digital Citizen**

Unplugged | Online Safety

Learn how you can be an upstander when you see cyberbullying.

## **Chapter Commentary**

Digital Citizenship

---

## **Chapter 7: End of Course Project**

### **Lesson 19: 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: Graph Paper Programming

## Overview

By "programming" one another to draw pictures, students get an opportunity to experience some of the core concepts of programming in a fun and accessible way. The class will start by having students use symbols to instruct each other to color squares on graph paper in an effort to reproduce an existing picture. If there's time, the lesson can conclude with images that the students create themselves.

## Purpose

The goal of this activity is to build critical thinking skills and excitement for the course, while introducing some of the fundamental programming concepts that will be used throughout the course. By introducing basic concepts like sequencing and algorithms to the class in an unplugged activity, students who are intimidated by computers can still build a foundation of understanding on these topics. In this lesson, students will learn how to develop an algorithm and encode it into a program.

## Agenda

### Warm Up (10 min)

Introduction to Graph Paper Programming

### Main Activity (30 min)

Practice Together

The Students' Turn

### Wrap Up (15 min)

Journaling / Flash Chat

### Optional Assessment (10 min)

### Extended Learning

### View on Code Studio

## Objectives

Students will be able to:

- Reframe a sequence of steps as an encoded program
- Explain constraints of translating problems from human language to machine language

## Preparation

- (Optional) Watch the Lesson in Action video.
- Print out one worksheet and 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

- **Graph Paper Programming** - Lesson in Action Video
- **Graph Paper Programming** - Worksheet Answer Key [Make a Copy](#)
- **Graph Paper Programming** - Assessment Answer Key [Make a Copy](#)

### For the Students

- **Graph Paper Programming** - Activity Worksheet [Make a Copy](#)
- **Graph Paper Programming** - Unplugged Video ([download](#))
- **Graph Paper Programming** - Assessment [Make a Copy](#)

## 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.

# Teaching Guide

## Warm Up (10 min)

### Introduction to Graph Paper Programming

In this activity, students will encode instructions to guide each other toward making drawings without letting the rest of their group see the original image. This warm-up frames the activity for the class.

**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:** How do you suppose that robots know how to do the things that they do? Do they have brains that work the same way that ours do?

Work this into a discussion on how people have to program robots to do specific things, using specific commands.

#### 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.

## Main Activity (30 min)

### Practice Together

In this activity, students will act as both programmers and robots, coloring in squares according to programs that they have written for one another.

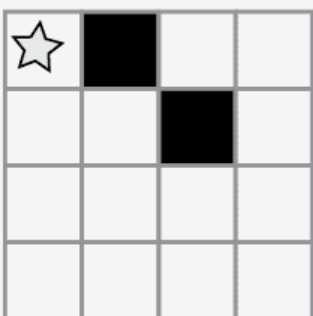
**Distribute:** Students will use 4x4 grids (or sheets of graph paper with 4x4 boxes sectioned off). They will also need the image worksheet.

**Display:** Project these commands, or write them on the board. They won't persist long, but they will help students make the transition from Algorithm to Program.

```
Move one square right
Move one square left
Move one square up
Move one square down
Fill in square with color
```

**Say:** Today, we all get to program robots...and they're already here in the room! It's you! We're going to write programs using symbols with special meanings to help each other recreate a picture. First, we'll practice together as if I am the robot and you are the programmers, then we can break up into groups so that everyone can get a turn.

**Display:** Display both the image that you are going to have the students walk you through, and a blank grid that you will fill-in with your ARM. Make sure that the instructions, grid, and image remain visible at the same time.



## 🎤 **Remarks**

Here is an image. Pretend that I am the robot with an Automatic Realization Machine (ARM). These are the only instructions that I understand.

Starting at the upper left-hand corner, guide my ARM out loud with your words.

**Model:** The class might give you instructions like these below. As you hear an instruction that you intend to follow, make sure to repeat it out loud so that the students can keep track of what you are doing.

```
Move One Square Right
Fill In Square with Color
Move One Square Right
Move One Square Down
Fill In Square with Color
```

Continue with the activity until you have completed your sample square.

**Capture:** Write each of the commands down so that students can see all of the steps that went into the one image.

```
Move One Square Right
Fill In Square with Color
Move One Square Right
Move One Square Down
Fill In Square with Color
```

**Say:** You just gave me a list of steps to finish a task. In programming, they call that an algorithm. Algorithms are great, because they are easy for you to understand as the programmer. BUT, what happens when we want to write down the algorithm for a drawing like this?

**Display:** Show the students a more complicated image, like the one below.



Next, begin writing down some of the instructions that it would take to replicate that image. Hopefully, students will see that writing everything out longhand would quickly become a bit of a nightmare.

```
Move One Square Right
Fill In Square with Color
Move One Square Right
Move One Square Right
Fill In Square with Color
Move One Square Down
Move One Square Left
Fill In Square with Color
Move One Square Left
Move One Square Left
Fill In Square with Color
PLUS 12 MORE INSTRUCTIONS!
```

**Display:** Show the students this list of symbols.



🗨️ **Discussion:** How could we use these symbols to make our instructions easier?



Draw out ideas that relate to transitioning from the verbal instructions to the symbols. Once the students get to that place, point out that this text:

- “Move one square right, Move one square right, Fill-in square with color”

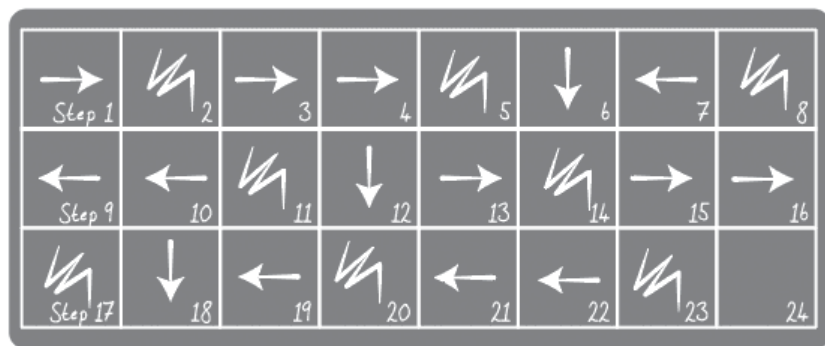
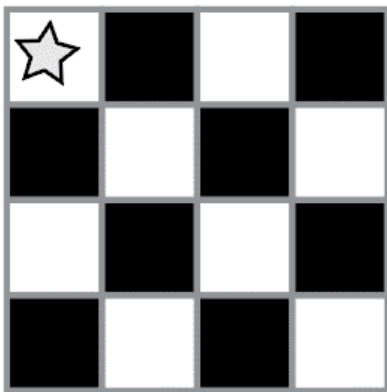
would now correspond to the program:



🔗 **Model:** Now, have the class help you draw the larger image using only symbols. Do not worry about unnecessary steps for now. If their final program works to create the image, consider it a win.

The classroom may be buzzing with suggestions by this point. If the class gets the gist of the exercise, this is a good place to discuss alternate ways of filling out the same grid. If there is still confusion, save that piece for another day and work with another example.

See a sample solution below:



### Discussion Goal

The goal for this discussion is to get at the idea that students can use symbols to stand for entire phrases. Once they understand that, share with them that making the switch from listing steps in detail to encoding them is called "programming."

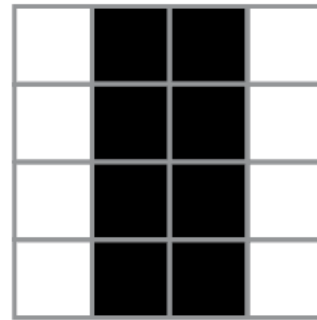
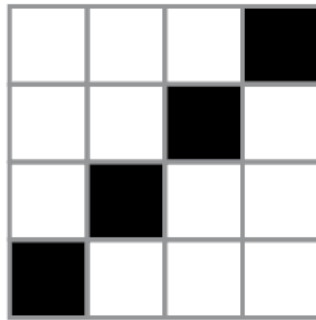
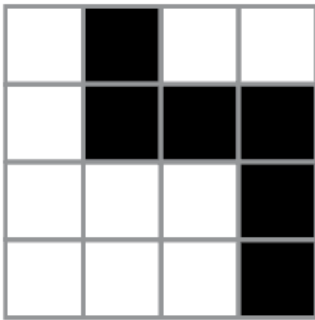
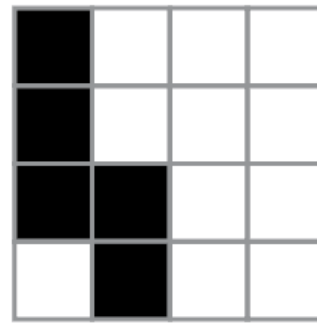
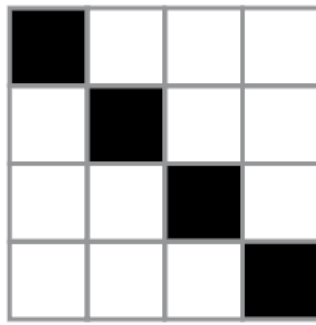
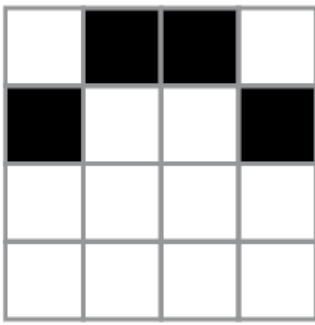
### Teaching Tip

Notice that we have written our program from left to right like you would read a book in English. Some students prefer this method. Others like to start each line of the grid on a new line of paper. The way they write their program doesn't matter as much as whether the other people in their groups can follow along!

## The Students' Turn

**Group:** Divide students into pairs or small groups.

- Have each pair/group choose an image from the worksheet.
- Discuss the algorithm to draw chosen image with partner(s).
- Convert algorithm into a program using symbols.
- Trade programs with another pair/group and draw one another's image.
- Choose another image and go again!



## Wrap Up (15 min)

### Journaling / Flash Chat

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 another image that you could code. Can you write the program that goes with this drawing?
- What other types of robots could we program if we changed what the arrows meant?

## Optional Assessment (10 min)

- Hand out **Graph Paper Programming - 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.

### Better and Better

- Have your class try making up their own images.
- Can they figure out how to program the images that they create?

### Class Challenge

- As the teacher, draw an image on a 5x5 grid.

- Can the class code that up along with you?

## 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 2: Introduction to Online Puzzles

## Overview

In this set of puzzles, students will begin with an introduction (or review depending on the experience of your class) of Code.org's online workspace. There will be videos pointing out the basic functionality of the workspace including the Run, Reset, and Step buttons. Also discussed in these videos: dragging Blockly blocks, deleting Blockly blocks, and connecting Blockly blocks. Next, students will practice their *sequencing* and *debugging* skills in maze. From there, students will see new types of puzzles like Collector, Artist, and Harvester when they learn the very basics of *loops*.

## Purpose

We recognize that every classroom has a spectrum of understanding for every subject. Some students in your class may be computer wizards, while others haven't had much experience at all. In order to create an equal playing (and learning) field, we have developed this "Ramp Up Stage" for Course D. This can be used as either an introduction or a review of how to use Code.org and basic computer science concepts. This stage covers all prerequisites needed to start Course D.

## Agenda

### Warm Up (10 min)

Introduction  
Vocabulary

### Bridging Activities - Programming (10 min)

Preview of Online Puzzles as a Class

### Main Activity (30 min)

Online Puzzles

### Wrap Up (10 min)

Journaling

[View on Code Studio](#)

## Objectives

Students will be able to:

- Order movement commands as sequential steps in a program.
- Modify an existing program to solve errors.
- Break down a long sequence of instructions into the largest repeatable sequence.

## 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

- **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.
- **Loop** - The action of doing something over and over again.
- **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 (10 min)

### Introduction

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

### Vocabulary

This lesson has four new and important vocabulary words:

- **Program** - Say it with me: Pro - Gram An algorithm that has been coded into something that can be run by a machine.
- **Programming** - Say it with me: Pro - Gramm - ing The art of creating a program.
- **Bug** - Say it with me: Bug An error in a program that prevents the program from running as expected.
- **Debugging** - Say it with me: De - Bugg - ing Finding and fixing errors in programs.
- **Loop** - Say it with me: Loo-p The action of doing something over and over again

## Bridging Activities - Programming (10 min)

This activity will help bring the unplugged concepts from "Graph Paper Programming" into the online world that the students are moving into.

### Preview of Online Puzzles as a Class

Pull up a puzzle from the lesson. We recommend puzzle 6 for this activity. Break up the students into groups of three or four. Have them "program" Red, the angry bird, to get to the pig using arrows from "Graph Paper Programming."



The class will not need to use the last arrow.

Once all the groups have an answer, discuss the path as a class.

## Main Activity (30 min)

### Online Puzzles

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

- Utilize **Pair Programming - Student Video** whenever possible during the activity.
- Encourage students with questions/challenges to start by asking their partner.
- Unanswered questions can be escalated to a nearby group, who might already know the solution.

- 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.

#### 🔔 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

🖥️ 10

*(click tabs to see student view)*

## Wrap Up (10 min)

### Journaling

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

#### Journal Prompts:

- What was today's lesson about?
- How did you feel about today's lesson?
- List some of the bugs you found in your programs today.
- What was your favorite puzzle to complete? Draw your favorite character completing a puzzle.

## 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 3: Relay Programming

## Overview

This activity will begin with a short lesson on debugging and persistence, then will quickly move to a race against the clock as students break into teams and work together to write a program one instruction at a time.

## Purpose

Teamwork is very important in computer science. Teams write and debug code with each other, instead of working as individuals. In this lesson, students will learn to work together while being as efficient as possible.

This activity also provides a sense of urgency that will teach students to balance their time carefully and avoid mistakes without falling too far behind. This experience can be stressful (which is expected!) Make sure you provide students with the tools to deal with potential frustration.

## Agenda

### Warm Up (15 min)

Where did I go wrong?

### Main Activity (20 min)

Relay Programming

### Wrap Up (15 min)

Journaling

### Extended Learning

### View on Code Studio

## Objectives

Students will be able to:

- Define ideas using code and symbols.
- Verify work done by teammates.
- Identify signs of frustration

## Preparation

- ☐ Watch the **Relay Programming - Teacher Video**.
- ☐ Locate a wide open space for this activity, such as the gym or outdoor field.
- ☐ Print out one **Relay Programming - Activity Packet** for each group.
- ☐ Supply each group with plenty of paper and pens/pencils.
- ☐ Print one **Relay Programming - Worksheet** 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

- **Relay Programming** - Worksheet Answer Key [Make a Copy](#)
- **Relay Programming** - Teacher Debugging Image [Make a Copy](#)
- **Relay Programming** - Teacher Video

### For the Students

- **Relay Programming** - Worksheet [Make a Copy](#)
- **Relay Programming** - Unplugged Video ([download](#))
- **Relay Programming** - Activity Packet [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.
- **Frustrated** - Feeling annoyed or angry because something is not the way you want it.
- **Persistence** - Trying again and again, even when something is very hard.
- **Program** - An algorithm that has been coded into something that can be run by a machine.



# Teaching Guide

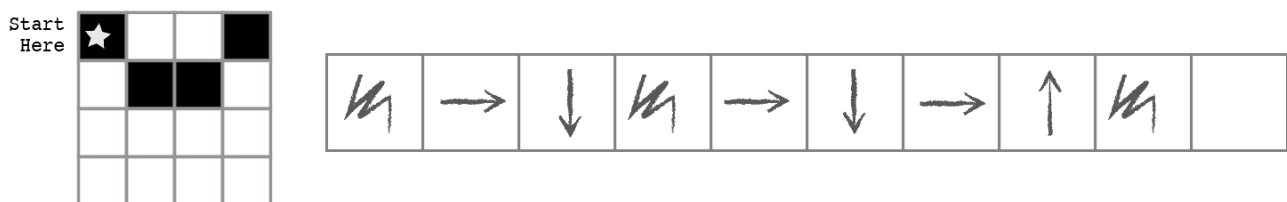
## Warm Up (15 min)

Recall that in "Graph Paper Programming" we guided our teammate's Automatic Realization Machine (ARM) using arrows. This warm up will bring back these ideas, which will be needed in the main activity.

Where did I go wrong?

**Goal:** In this lesson, we want to help students learn to identify and fix bugs in their own programs. The easiest way to do that is to first present students with a program that contains bugs that are not their fault. Once they've helped you fix "your" program, share with them how frustrating it can be to make mistakes, and help them see that those feelings are completely normal and they shouldn't feel embarrassed by them.

**Display:** Show your students the provided **Relay Programming - Teacher Debugging Image**.



**Discuss:** Get the attention of the class and let them know that you are stuck! You have this challenge, and you thought you had solved it, but it doesn't seem to be working. Your program has a bug, can they help you fix it?

Take a moment to walk them through the rules: - Start at the star - Follow the instructions step-by-step - End when all of the right squares are filled in

*Optional:* Follow along by filling in a blank grid. Express frustration when the picture doesn't turn out the way that you wanted it to.

**Think:** Can you figure out why my program doesn't work?

**Pair:** Let students work together to see if they can figure out what the program is supposed to say.

**Share:** Ask students if anyone was able to figure out a way to solve the problem. When you get a correct answer, let the students know that they are great at "debugging"!

**Discuss:** Ask the students if they could tell how you were feeling when you couldn't figure out the answer. They might suggest that you were "mad" or "sad". Instead of telling them "no", describe that you were feeling a little bit mad, a little bit sad, and a little bit confused. When you put all of those emotions together, it makes a feeling called "frustration". When you are "frustrated" you might think you are mad, sad, or confused -- and you might be tempted to give up -- but frustration is a natural feeling and it's a big hint that you are about to learn something! Instead of quitting, practice persistence. Keep trying over and over again. After a few times, you will start to understand how to debug your problems!

**Distribute:** To make sure that students understand the idea of finding and fixing errors (debugging) pass out the **Relay Programming - Worksheet** and have students complete the task in pairs.

*Optional:* If you want to move the activity along more quickly, feel free to complete these as a class, instead.

**Transition:** Now it's time to play the game!

## Main Activity (20 min)

# Relay Programming

With Graph Paper Programming in mind, it's time to split up into teams and prepare to run the activity as a relay!

**Set-Up:** Prepare the **Relay Programming - Activity Packet** by printing out one copy for each team of 4-5 students. Cut or fold each page along the center dotted line.

Go over the rules of the game with your class:

- Divide students into groups of 3-5.
- Have each group queue up relay-style.
- Place an identical image at the other side of the room/gym/field from each team.
- Have the first student in line dash over to the image, review it, and write down the first symbol in the program to reproduce that image.
- The first student then runs back and tags the next person in line, then goes to the back of the queue.
- The next person in line dashes to the image, reviews the image, reviews the program that has already been written, then either debugs the program by crossing out an incorrect symbol, or adds a new one. That student then dashes back to tag the next person, and the process continues until one group has finished their program.

First group to finish with a program that matches the image is the winner! Play through this several times, with images of increasing difficulty.

Go through the game as many times as you can before time runs out or your students begin feeling exhausted.

**Transition:** Once the game is over, circle everyone up to share lessons learned.

**Discuss:** What did we learn today?

- What if each person on a team were allowed to do five arrows at a time?
  - How important would it be to debug our own work and the work of the programmer before us?
  - How about with 10 arrows?
  - 10,000? Would it be more or less important?
- Do you think a program is better or worse when more than one person has worked on it?
- Do you think people make more or fewer mistakes when they're in a hurry?
- If you find a mistake, do you have to throw out the entire program and start over?

## 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?

### Content Corner

For more on persistence and frustration, try reading **Stevie and the Big Project** to your students. It will help them spot moments of frustration. It will also help give them the tools to deal with it.

If you do not read the book, take a moment to cover tips on frustration and persistence as a class:

Tips to Help With Frustration

- Count to 10
- Take deep breaths
- Journal about them
- Talk to a partner about them
- Ask for help

Tips for Being Persistent

- Keep track of what you have already tried
- Describe what is happening
- Describe what is supposed to happen
- What does that tell you?
- Make a change and try again

### Clarifications

Here are some clarifications that need to be shared from time to time:

- Only one person from each group can be at the image at one time.
- It is okay to discuss algorithms with the rest of the group in line, even up to the point of planning who is going to write what when they get to the image.
- When a student debugs a program by crossing out an incorrect instruction (or a grouping of incorrect instructions) this counts as their entire turn. The next player will need to figure out how to correct the removed item.

- How did you feel during today's lesson?
- How did teamwork play a role in the success of writing today's program?
- Did you start to get frustrated at any point? What did you do about it?

## Extended Learning

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

### Pass the paper

- If you don't have the time or room for a relay, you can have students pass the paper around their desk grouping, each writing one arrow before they move the paper along.

### Fill It, Move It

- As the teacher, draw an image with as many filled squares as children in each group.
- Have the students write as many arrows in the program as it takes to get to a filled-in square (including actually filling that square in) before passing to the next person.

### Debugging Together

Draw an image on the board. Have each student create a program for the image. Ask students to trade with their elbow partner and debug each other's code.

- Circle the first incorrect step, then pass it back.
- Give the students another chance to review and debug their own work.
- Ask for a volunteer to share their program.

Ask the class:

- How many students had the same program?
- Anyone have something different?

## 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: Debugging with Laurel

## Overview

In this online activity, students will practice debugging in the "collector" environment. Students will get to practice reading and editing code to fix puzzles with simple algorithms, loops and nested loops.

## Purpose

The purpose of this lesson is to teach students that failure is normal when learning a new skill. Students will be given pre-written programs that do NOT work. They will be asked to fix these programs. This process, called "debugging", teaches students essential problem solving and critical thinking skills. These skills transfer over as students proceed to harder and harder programming projects.

## Agenda

### Warm Up (15 min)

Introduction

### Bridging Activities - Debugging (15 min)

Unplugged Activity with Paper Blocks

Preview of Online Puzzles as a Class

### Main Activity (30 min)

Online Puzzles

### Wrap Up (15 min)

Journaling

### View on Code Studio

## Objectives

Students will be able to:

- Read and comprehend given code.
- Identify a bug and the problems it causes in a program.
- Describe and implement a plan to debug a program.

## 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

- **Relay Programming** - Activity Packet
- **Unplugged Blocks (Courses C-F)** - Manipulatives

## 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

One of the most important parts of learning to program is learning to debug. Ask the class if they have ever learned a new skill and faced failure.

For example:

- Learning to ride a bike and falling down
- Learning to bake and burning the food
- Learning to play a sport and not winning a game

Facing failure is very common when learning new things. Have students discuss past failures and how they overcame them.

In programming, computer scientists often run into "bugs" in their code.

- **Bug:** Part of a program that does not work correctly.

A bug can really mess up the program, so it's important to learn to "debug" your code.

- **Debug:** Finding and fixing problems in your algorithm or program.

Continue the conversation if you think your class needs more of an introduction, but leave time for one of the bridging activities.

## Bridging Activities - Debugging (15 min)

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

### Unplugged Activity with Paper Blocks

Break your class up into teams of 3-5 and go to a large space. This space could be in a gym or outside. Queue up these teams like in "Relay Programming". Pick a semi-difficult design from **Relay Programming - Activity Packet**. Display this design at the end of a long distance between each team. Along with the display, provide each team with enough paper blocks from **Unplugged Blocks (Courses C-F) - Manipulatives**. Each team will need plenty of `fill 1` and `move ___` blocks. The `move ___` blocks can either be filled in ahead of time, or filled in during the game. Either way, make sure these blocks are properly defined when the game is playing out.

Once all the teams are lined up, display or read out the following rules:

- The first student in line gets to run over to the image, review it, and place the first code block in the program to reproduce that image.
- The first student then runs back and tags the next person in line, then goes to the back of the queue.
- The next person in line dashes to the image, reviews the image, reviews the program that has already been written, then either debugs the program by taking out an incorrect code block, or adds a new block in. That student then dashes back to tag the next person, and the process continues until one group has finished their program.

Make sure the students are only using the `fill 1` or `move ___` blocks, and only placing one down per turn. The first team to *correctly* write out the code for their image wins.

### Preview of Online Puzzles as a Class

Group the students into teams of 3. Choose a puzzle from the lesson. We recommend the fifth puzzle. Have the students in each team sit at a computer with the puzzle displayed. Each team only gets one computer and only one student can be looking at the screen. Display or read the following rules:

- Only one student in each team can look at the screen.
- This person can only delete or add one block at a time. Once that person has added or removed a block, they can tap the shoulder of the next person.
- The next person can proceed to play out their turn.
- No turns can be skipped or repeated, everyone must play an equal amount.

The first team to finish the puzzle *correctly* wins!

## Main Activity (30 min)

### Online Puzzles

It might be helpful for students to sit with their teams from the bridging activities. Every student should work on these puzzles individually or in pairs, but having a closely knit group to ask and answer questions with can help develop confidence and understanding with the subject matter.

#### Code Studio levels

##### De-bugging with the Step Button

 1

*(click tabs to see student view)*

##### Practice

 2

 3

 4

 5

 6

*(click tabs to see student view)*

##### Challenge

 7

*(click tabs to see student view)*

##### Practice

 8

 9

*(click tabs to see student view)*

## Wrap Up (15 min)

### Journaling

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

Journal Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- What is a bug? How do you know there is a bug in your program?
- What does it mean to debug code? How do you debug a program?

## 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: Events in Bounce

## Overview

In this online activity, students will learn what events are, and how computers use them in programs like video games. Students will work through puzzles making the program react to events (like arrow buttons being pressed.) At the end of the puzzle, students will have the opportunity to customize their game with different speeds and sounds.

## Purpose

Events are very common in computer programs, especially in video games.

In this lesson, students will develop their understanding of events by making a sports-based game. Students will learn to make their paddle move according to arrow keys, and make noises when objects collide. At the very end, they will get to customize their game to make it more unique!

## Agenda

### Warm Up (10 min)

Introduction

### Main Activity (30 min)

Online Puzzles

### Wrap Up (10 min)

Assessment

Journaling

### Extended Learning

### View on Code Studio

## Objectives

Students will be able to:

- Identify actions that correlate to input events.
- Create an interactive game using sequence and event-handlers.
- Share a creative artifact with other students.

## Preparation

- Make sure every student has a journal.
- Play through the puzzles to find any potential problem areas for your class.
- Read the "Events in Bounce - How Do They Happen?" slide deck (to be presented to students at end of class).

## Links

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

For the Teachers

- **Events in Bounce - How Do They Happen? - Slide Deck**

## Vocabulary

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



# Teaching Guide

## Warm Up (10 min)

### Introduction

Ask the students to come sit down near you. Now tell them to all stand up!

Tell the students what you just did was declare an event and an action. When you say to sit down, it is an event. The action responding to this event is the class sitting down. This is the same when you ask the class to stand up. Events and actions are easily identifiable in our lives.

Some other events and actions include:

- Feeling hungry and eating food
- Stubbing your toe and yelling "Ouch!"
- Getting the basketball in the basket and scoring a point for your team!

Ask the class to come up with a couple of more events. Tell them that they will be making a game where the program will have actions associated to events that they code!

## Main Activity (30 min)

### Online Puzzles

#### Code Studio levels

#### Practice

 1

 2

 3

 4

 5

 6

 7

*(click tabs to see student view)*

#### Free Play

 8

*(click tabs to see student view)*

🔗 At the end of the set of puzzles, students will have the opportunity to make their game unique. Have the students try new ways to make the game more challenging. For example, try playing with many balls at once, or each time the ball bounces off a wall, launch more balls.

#### 💡 Teaching 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**.

## Wrap Up (10 min)

### Assessment

🔗 Present the **Events in Bounce - How Do They Happen?** slide deck to students. Allow them to record the correct order of events on the second slide in their journals first. Call on a few students to share their answers before revealing the third slide. Discuss the correct sequence with the class.

**Correct Order: D, B, A, C**

#### 💡 Teaching Tip

Although introduced in this lesson, the slide deck "Events in Bounce - How Do They Happen?" can be applied more generally to express the relationship between hardware and software. Particularly, the final slide simplifies the input-to-output sequence and can be made into a poster for your classroom.

### 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 did you do to make your game super cool?
- What kind of game do you want to code in the future?

## Extended Learning

### Take Me Out to the Ball Game

Take the students outside to play some sort of ball game. Keep track of events and actions. For example, not dribbling in basketball results in a traveling foul and the other team gets the ball. In soccer, kicking the ball out of bounds results in the other team kicking the ball in. Getting the ball to the goal results in a point! Make up more events if your students are into it. Have the all of the students yell "Yippee" when the captain of one team scores a point. Have everyone fall to the ground and roll around if a student makes two goals in a row!

## Standards Alignment

### CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming
- ▶ **CS** - Computing Systems



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: Build a Star Wars Game

## Overview

In this lesson, students will practice using events to build a game that they can share online. Featuring R2-D2 and other Star Wars characters, students will be guided through events, then given space to create their own game.

## Purpose

CS Fundamentals is not simply about teaching computer science, it is about making computer science fun and exciting. In this series, students will learn about events using popular characters from Star Wars. These puzzles blur the lines between "learning" and "fun". Also, students will learn to recognize regular programming practices in games so that when they play games at home, they can see common computer science principles being used.

## Agenda

### Warm Up (15 min)

Introduction

### Main Activity (30 min)

Online Puzzles

### Wrap Up (15 min)

Journaling

[View on Code Studio](#)

## Objectives

Students will be able to:

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

## Preparation

- Play through the puzzles and find any potential problem areas.
- Make sure every student has a journal.

## Vocabulary

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

# Teaching Guide

## Warm Up (15 min)

### Introduction

In a class discussion, ask the students what their favorite video game is (you might need to remind the students to only use games that are classroom appropriate). Ask the students what their favorite part of the game is.

Most of the time, students will respond with some kind of event. When you recognize a student response that describes an event, ask the student to describe it further.

Once the student is done describing their fun, take a minute to relate it back to the definition of an event.

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

Ask the students to try and relate some of their favorite parts of video games and how they can be described as events. Have them pair share and discuss the differences between their events and their partner's.

#### Teacher Tip

If you're not quite sure if a student's response describes an event, try to break down the response. Is there an action and a response?

For example:

- Crossing the finish line and having on screen characters congratulate you
- Finding a big pot of treasure (or other item) and watching your inventory grow
- Buying new items from the game's store and having the item to use
- Pressing the buttons on a game controller and having your character do something cool

## Main Activity (30 min)

### Online Puzzles

#### Code Studio levels

##### Practice

1

2

3

4

5

6

7

8

9

(click tabs to see student view)

##### Free Play

10

(click tabs to see student view)

Students will likely be very excited to make their own Star Wars game at the end of this set of puzzles. If there's time, ask them to plan out what they want the game to do. The planning and preparation will help the students better recognize the key concepts this lesson is trying to teach. Encourage the students to share and remix each other's games at the end of this lesson.

#### 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.**

## Wrap Up (15 min)

### Journaling

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

**Journal Prompts:**

- What was today's lesson about?
- How do you feel about today's lesson?
- Give an example of an event you used in your program today?
- Why is it important not to share private information online? How do you know if information is private?

## 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: Dance Party

## Overview

In this lesson, students will program their own interactive dance party. This activity requires sound as the tool was built to respond to music.

## Purpose

This lesson introduces the core CS concepts of coding and event programming (using blocks).

## Agenda

### Getting Started (5 minutes)

#### Setting the Stage

### Activity (30-45 minutes)

#### Code Your Own Dance Party

#### Level by Level Support

### Wrap Up (5 minutes)

#### Debrief

### Assessment (2 minutes)

### View on Code Studio

## Objectives

Students will be able to:

- Develop programs that respond to timed events
- Develop programs that respond to user input
- Create dance animations with code

## Preparation

- Play through the puzzles to find any potential problem areas for your class.
- Make sure every student has a journal.
- Consider the need for headphones. This activity relies on sound.

## Links

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

### For the Teachers

- [Spotify Playlist \(all ages\)](#)

### For the Students

- [Dance Party Project Guide - Worksheet](#)

[Make a Copy](#)

## Vocabulary

- **code** - (v) to write code, or to write instructions for a computer.
- **Event** - An action that causes something to happen.
- **Program** - An algorithm that has been coded into something that can be run by a machine.

# Teaching Guide

## Getting Started (5 minutes)

### Setting the Stage

Welcome students to class and very briefly introduce the day's activity.

#### **Remarks**

Today we're going to do something really creative. What's your favorite way to be creative?

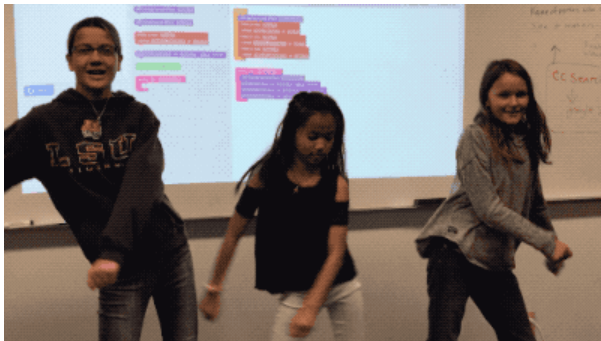
Encourage students to share the ways they express creativity, such as with art, dance, music, writing.

Explain that today we're going to be creative with our code. Just like choosing which type of colors of paint to use, or what kinds of words you write with can be express creativity, choosing what code you write and how people interact with it can be an opportunity to express your creativity too!

**Get up and dance:** Announce to the class that today we're going to see how we can combine coding with dancing in a creative way. Ask your kids to floss, dab, or do a creative dance move of their own for 10 seconds to get them in the mood. You can play a song from this **Spotify Playlist (all ages)** to help kick things off. Capture your class's moves on video.

#### Teaching Tip

If you have time and would like to prepare your students with an unplugged activity, consider delivering "**Dance Party: Unplugged**" before this lesson. This brief lesson introduces students to the idea of events triggering different dance moves.



## Activity (30-45 minutes)

### Code Your Own Dance Party

#### Music Filtering

This tutorial features songs from popular artists. To get a preview of the song list in this tutorial, check out this **Spotify Playlist**. We are using radio-safe versions of all songs and for students under 13, we limit the music to this filtered list **Spotify Playlist (all ages)**. If you would like to use the filtered list with older students, you can share **this link** with your classroom.

#### Code Studio levels

##### **Warm Up**

 1

*(click tabs to see student view)*

##### **Practice**

 2

*(click tabs to see student view)*

##### **Events**

 3

 4

 5

*(click tabs to see student view)*

## Measures

[6](#)[7](#)[8](#)[9](#)

(click tabs to see student view)

## Properties

[10](#)[11](#)[12](#)[13](#)

(click tabs to see student view)

## Wrap-up

[14](#)

(click tabs to see student view)

## Free Play

[15](#)

(click tabs to see student view)

## Levels

[Extra](#)[Extra](#)

(click tabs to see student view)

## Level by Level Support

### Level 1

- Drag the red `make a new block` from the toolbox on the left to the workspace on the right. Connect it inside the `setup` block.
- You have now written your first program. Make sure to press Run to see what happens. You should hear music and see a character start to move in the display area.

### Level 2

- Levels 2-3 are about making the dance interactive.
- The green blocks are event blocks. These blocks start a new sequence of code and do not need to be connected inside the `setup` block.
- Connecting the purple block under the green event block allows you to make the character perform a dance by pressing the orange arrow buttons or keys on your keyboard.
- Make sure to press the arrow buttons after pressing Run or the dancer(s) won't move.

### Level 3

- Make sure to bring out a second purple `do once` block. You should have a `bears do once` block and a `cats do once` block in your workspace. Both should be connected to a green `when pressed` event block.
- Make sure to press the arrow buttons after pressing Run or the dancers won't move.

### Level 4

- Levels 4-5 are about synchronizing the dance to the music.
- The `after measures` event blocks also start a new sequence of code and should not be connected inside the `setup` block.
- Connecting the purple `do forever` block under the green `after measures` event block should make the character perform a dance move after the number of measures you indicate.
- The `do forever` block works differently from the `do once` blocks seen in the previous levels.

### Level 5

- Make sure to bring out a second green event block. You should have a `after 4 measures` block and a `after 6 measures` block in your workspace. Both should have purple block connected underneath.

### Level 6

- Level 6 is about creating groups of dancers quickly.
- Use the new block provided into the toolbox to create a set of smaller dancers. \*You should also use the normal `make a new` block to create a larger "lead" dancer.
- Many students will be familiar with the idea that you can make something seem to be further away by drawing it a smaller scale. In the next level you'll be able to fine tune this effect.

### Level 7

- Levels 7-9 are about adjusting the properties (e.g. size, color) of the dancers.



- It is important to make sure that the teal `set` block is placed somewhere in the program *after* the dancers have been created. To solve this puzzle, place a `set size` block anywhere in your program and use it to change the size of some of your dancers.
- Dancers created as a group have a default size of 30. Other dancers have a default size of 100.

#### Level 8

- As with the previous level, make sure to only use the `set tint to (color)` block after you have made the dancers in your program. For example, placing it as the first step in the `setup` area of your program will have no effect.



#### Level 9

- With the right code, you should see the dancer cycle through different colors, sizes, or dance moves.
- Make sure there is a teal `change`, a teal `randomize` block, or a purple `do forever` block connected inside the `every 2 measures` block.
- Make sure `do forever` blocks are set to either (Next), (Previous), or (Random). Otherwise, the dancer will just perform the `set` move repeatedly.
- Note that the **code students write in this level is not checked for correctness**. This means they will always pass the level, even if they do not change the program. Students should feel free to experiment with their code in ways that are interesting to them. Click the “Finish” button to move on.
- Making new dancers inside the `every 2 measures` block will cause your program to create multiple identical dancers at the same location(s) and may lead to unintended consequences!

#### Teaching Tip

By this point in the lesson you may notice that the instructions are less prescriptive. Encourage students to be creative and explore the new blocks that are introduced. From this point on student code is **not checked for correctness** in order to encourage experimentation instead of solving a specific task.

#### Level 10

- This last level is very open-ended. The tutorial itself is designed to give students ample time to keep working on their own dance.
- **Encourage Sharing:** If students have cell phones with a data plan they can quickly text a link to their projects to their own phone or a friend's. If your school policy allows it, encourage them to do so here.
- **Encourage Creativity:** Creativity is important throughout this lesson, but this is true here more than anywhere else!

## Wrap Up (5 minutes)

### Debrief

- Pose a prompt that has multiple answers such as “What is something you enjoyed about today's activity?” or “What is the connection between creativity and computer science?”

## Assessment (2 minutes)

Ask students to add their “Whip Around” sticky notes or note cards to your “Computer Science” mind map on their way out the door. Try to populate the board with lots of great ideas about what CS is and why it matters.



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: Loops in Ice Age

## Overview

As a quick update (or introduction) to using loops, this stage will have students using the `repeat` block to get Scrat to the acorn more efficiently.

## 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 these Code.org puzzles, students will learn to add instructions to existing loops, gather repeated code into loops, and recognize patterns that need to be repeated.

## Agenda

### Warm Up - The Unplugged Foundation (10 min)

Repeat After Me

### Online Foundation: Preview Loops in Ice Age

### Main Activity (30 min)

Online Puzzles

### Wrap Up (5 - 10 min)

Journaling

### Extended Learning

### View on Code Studio

## Objectives

Students will be able to:

- Construct a program using structures that repeat areas of code
- Improve existing code by finding areas of repetition and moving them into looping structures

## Preparation

- (Optional) Pick a couple of puzzles to do as a group with your class.
- 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

- **Pair Programming** - Student Video
- **Feeling Faces** - Emotion Images

[Make a Copy](#)

## Vocabulary

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

# Teaching Guide

## Warm Up - The Unplugged Foundation (10 min)

### Repeat After Me

**Model:** Ask for a volunteer and have them stand.

- Instruct your volunteer to walk around the table (or their chair, or a friend).
- When they finish, instruct them to do it again, using the exact same words you did before.
- When they finish, instruct again.
- Then again.

**Prompt:** Would it have been easier for me to just ask you to go around the table four times?

**Think:** What if I wanted you to do it ten times? How would you reword my instructions so that they were more efficient and I didn't have to repeat myself so much? Feel free to write your instructions down on a piece of scrap paper.

**Share:** Ask a few students to share their instructions with the class, pointing out how each approach has simplified the overall approach to giving instructions.

**Say:** Today we're going to work on finding ways to make giving lots of instructions easier, especially when those instructions repeat themselves a lot.

## Online Foundation: Preview Loops in Ice Age

To finish the connection, preview an online puzzle (or two) as a class.

**Model:** Reveal an entire online puzzle from the progression to come. We recommend Puzzle 5. Point out the "Play Area" with Scrat and the acorn, as well as the "Work Space" with the Blockly code. Explain that this Blockly code is now the language that the class will be using to help Scrat get to the acorn. Do students see any similarities to the exercise that they just did? What are the big differences?

Work with your class to drag code into the workspace in such a way that Scrat (eventually) gets to the acorn.

**Transition:** Students should now be ready to transition to computers to complete online puzzles on their own.

## Main Activity (30 min)

### Online Puzzles

As students work through the puzzles, see if they can figure out how many blocks they use with a loop vs. without a loop.

### Code Studio levels

#### Practice

 1

 2

*(click tabs to see student view)*

#### Using the Repeat Block

 3

*(click tabs to see student view)*

#### Practice

 4

 5

 6

 7

 8

 9

*(click tabs to see student view)*

## Challenge

10

(click tabs to see student view)

## Practice

11

12

(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 - Student Video** whenever possible
- 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.

### Teacher Tip:

Show the students the *right* way to help classmates by:

- 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

## 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?
- 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 maze with Scrat trying to get to an Acorn. Will loops help you solve it?
- Draw yourself using a loop to do an everyday activity, like brushing your teeth.

## Extended Learning

### 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



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: Drawing Shapes with Loops

## Overview

This lesson builds on the understanding of loops from previous lessons and gives students a chance to be truly creative. This activity doubles as a debugging exercise for extra problem-solving practice.

## Purpose

This series highlights the power of loops with an array of puzzles meant to get students thinking about why `repeat` loops are superior to `longhand`.

## Agenda

### Warm Up (15 min)

Introduction

### Main Activity (30 min)

Online Puzzles

### Wrap Up (15 min)

Journaling

[View on Code Studio](#)

## Objectives

Students will be able to:

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

## Preparation

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

## Vocabulary

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

# Teaching Guide

## Warm Up (15 min)

### Introduction

Students should have had an 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"**
- Reviewing a puzzle from the last lesson
- Previewing a puzzle from this lesson

Any of these should help prepare your class for fun with the online puzzles!

## Main Activity (30 min)

### Online Puzzles

#### Code Studio levels

#### Artist Intro with JR Hildebrand

 1

*(click tabs to see student view)*

#### Practice

 2

 3

*(click tabs to see student view)*

#### Loops with the Artist

 4

*(click tabs to see student view)*

#### Practice

 5

 6

 7

 8

 9

*(click tabs to see student view)*

#### Challenge

 10

*(click tabs to see student view)*

#### Practice

 11

*(click tabs to see student view)*

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 as they program online.

## 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 could program using loops? 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 10: Nested Loops in Maze

## Overview

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

## Purpose

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

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

## Agenda

### Warm Up (10 min)

Introduction

### Main Activity (30 min)

Online Puzzles

### Wrap Up (15 min)

Journaling

[View on Code Studio](#)

## Objectives

Students will be able to:

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

## Preparation

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

## Vocabulary

- **Command** - An instruction for the computer. Many commands put together make up algorithms and computer programs.
- **Loop** - The action of doing something over and over again.
- **Repeat** - To do something again.

# Teaching Guide

## Warm Up (10 min)

### Introduction

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

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

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

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

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

## Main Activity (30 min)

### Online Puzzles

We highly recommend pair programming for this lesson. This may not be an easy topic for the majority of your students. Working with a partner and discussing potential solutions to the puzzles might ease the students' minds.

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

### Code Studio levels

#### Practice

[1](#)[2](#)

*(click tabs to see student view)*

#### Nested Loops with the Bee

[3](#)

*(click tabs to see student view)*

#### Prediction

[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)*

#### Prediction

[13](#)

*(click tabs to see student view)*

#### Levels

[Extra](#)[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 a nested loop?
- Can you draw a puzzle that would use a nested loop? Try coding the solution to your own puzzle.

## Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ AP - Algorithms & Programming



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: Conditionals with Cards

## Overview

This lesson demonstrates how conditionals can be used to tailor a program to specific information. We don't always have all of the information we need when writing a program. Sometimes you will want to do something different in one situation than in another, even if you don't know what situation will be true when your code runs. That is where conditionals come in. Conditionals allow a computer to make a decision, based on the information that is true any time your code is run.

## Purpose

One of the best parts of teaching *conditionals* is that students already understand the concept from their everyday lives.

This lesson merges computer science into the real world by building off of their ability to tell if a condition is true or false. Students will learn to use `if` statements to declare when a certain command should be run, as well as `if / else` statements to declare when a command should be run and what to run otherwise. Students may not recognize the word *conditionals*, but most students will understand the idea of using "if" to make sure that some action only occurs when it is supposed to.

## Agenda

### Warm Up (20 min)

Vocabulary

Introduction

### Main Activity (20 min)

Conditionals with Cards Sample Program - Teacher Prep Guide

### Wrap Up (15 min)

Flash Chat: What did we learn?

Journaling

### Assessment (5 min)

### 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.
- Traverse a program and predict the outcome, given a set of input.

## Preparation

- (Optional) Watch the Lesson in Action Video.
- Gather decks of cards or something similar.
- Print or display the sample programs.
- 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

- **Conditionals with Cards** - Lesson in Action Video
- **Conditionals with Cards Sample Program** - Teacher Prep Guide [Make a Copy](#)
- **Conditionals with Cards** - Assessment Video
- **Conditionals with Cards** - Assessment Answer Key [Make a Copy](#)

### For the Students

- **Conditionals with Cards** - Unplugged Video ([download](#))
- **Conditionals with Cards** - Assessment [Make a Copy](#)

## Vocabulary

- **Conditionals** - Statements that only run under certain conditions.

# Teaching Guide

## Warm Up (20 min)

### Vocabulary

This lesson has one new and important word:

**Conditionals** - Say it with me: Con-di-shun-uls

Statements that only run under certain conditions.

### Introduction

- We can start this lesson off right away
  - Let the class know that if they can be completely quiet for thirty seconds, you will do something like:
    - Sing an opera song
    - Give five more minutes of recess
    - Do a handstand
  - Start counting right away.
  - If the students succeed, point out that they succeeded, so they get the reward.
  - Otherwise, point out that they were not completely quiet for a full thirty seconds, so they do not get the reward.
- Ask the class "What was the condition of the reward?"
  - The condition was IF you were quiet for 30 seconds
    - If you were, the condition would be true, and you would get the reward.
    - If you weren't, the condition would be false, so the reward would not apply.
  - Can we come up with another conditional?
    - If you can guess my age correctly, the class can give you applause.
    - If I know an answer, I can raise my hand.
    - What examples can you come up with?
- Sometimes, we want to have an extra condition, in case the "IF" statement is not true.
  - This extra condition is called an "ELSE" statement
  - When the "IF" condition isn't met, we can look at the "ELSE" for what to do
    - Example: IF I draw a king from this deck of cards, everybody claps. Or ELSE, everyone says "Awwwwwwwwe."
    - Let's try it. (Draw a card and see if your class reacts appropriately.)
  - Ask the class to analyze what just happened.
    - What was the IF?
    - What was the ELSE?
    - Which condition was met?
  - Believe it or not, we have even one more option.
    - What if I wanted you to clap if I draw a 7, or else if I draw something less than seven you say "YAY," or else you say "Awwwwwwwwe"?.
      - This is why we have the terms If, Else-If, and Else.
      - If is the first condition
      - Else-If gets looked at only if the "If" isn't true.
      - Else gets looked at only if nothing before it is true.

Now let's play a game.

## Main Activity (20 min)

Conditionals with Cards Sample Program - Teacher Prep Guide

**Directions:**

- Create a few programs with your class that depend on things like a card's suit, color, or value to award or subtract points. You can write the program as an algorithm, pseudocode, or actual code.

Here is a sample algorithm:

```
if (CARD is RED)
  Award YOUR team 1 point

Else
  Award OTHER team 1 point
```

Here is a sample of the same program in pseudocode:

```
If (card.color == RED){
  points.yours = points.yours + 1;
}

Else {
  points.other = points.other + 1;
}
```

- Decide how you want to split your class into teams.
- Each team should have a pile of cards (at least as many cards as team members) nearby.
- Put one of your “Programs” up on the board for all to see.
- Have the teams take turns drawing cards and following the program to see how many points they score in each round.
- Play several times with several different programs to help the students really understand conditionals.

Once the class has had some practice, you can encourage students to nest conditionals inside one another. Make sure they understand that if the card is red, YOUR team is awarded 1 point, and then *nothing else happens*, since the condition was met:

```
If (CARD is RED)
  Award YOUR team 1 point

Else
  If (CARD is higher than 9)
    Award OTHER team 1 point
  Else
    Award YOUR team the same number of points on the card
```

Here is the same program in pseudocode:

```
If (card.color == RED ){
  points.yours = points.yours + 1;
}
Else {
  if (card.value > 9){
    points.other = points.other + 1;
  }
  Else {
    points.yours = points.yours + card.value;
  }
}
```

## Wrap Up (15 min)

Flash Chat: What did we learn?

- If you were going to code this up in Blockly, what would you need to add around your conditionals to let the code run more than one time? (A loop)
- What other things do you do during the day under certain conditions?
- If you are supposed to do something when the value of a card is more than 5, and you draw a 5, do you meet that condition?

- Notice that conditions are either "True" or "False." There is no assessment of a condition that evaluates to "Banana."
- When you need to meet several combinations of conditions, we can use something called "nested conditionals."
  - What do you think that means?
  - Can you give an example of where we saw that during the game?
- What part of that game did you like the best?

#### 💡 Lesson Tip

Flash Chat questions are intended to spark big-picture thinking about how the lesson relates to the greater world and the students' greater future. Use your knowledge of your classroom to decide if you want to discuss these as a class, in groups, or with an elbow partner.

## Journaling

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

### Journal Prompts:

- What was today's lesson about?
- How do you feel about today's lesson?
- What is a conditional? How did you use a conditional today?
- What are some of the conditionals you used today? Can you come up with some more that you would use with a deck of cards?

## Assessment (5 min)

Hand out **Conditionals with Cards - 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. Here's a **Conditionals with Cards - Assessment Video** to watch as a guide.

## Extended Learning

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

### True/False Tag

- Line students up as if to play **Red Light / Green Light**.
- Select one person to stand in front as the Caller.
- The Caller chooses a condition and asks everyone who meets that condition to take a step forward.
  - If you have a red belt, step forward.
  - If you are wearing sandals, take a step forward.
- Try switching it up by saying things like "If you are *not* blonde, step forward."

### Nesting

- Break students up into pairs or small groups.
- Have them write if statements for playing cards on strips of paper, such as:
  - the suit is clubs
  - the color is red
- Have students create similar strips for outcomes.
  - Add one point
  - Subtract one point
- Once that's done, have students choose three of each type of strip and three playing cards, paying attention to the order selected.
- Using three pieces of paper, have students write three different programs using only the sets of strips that they selected, in any order.
  - Encourage students to put some if statements inside other if statements.

- Now, students should run through all three programs using the cards that they drew, in the same order for each program.
  - Did any two programs return the same answer?
  - Did any return something different?

## 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: If/Else with Bee

## Overview

Up until this point students have been writing code that executes exactly the same way each time it is run - reliable, but not very flexible. In this lesson, your class will begin to code with conditionals, allowing them to write code that functions differently depending on the specific conditions the program encounters.

## Purpose

After being introduced to conditionals in "Conditionals with Cards," students will now practice using them in their programs. The `if / else` blocks will allow for a more flexible program. The bee will only collect nectar **if** there is a flower or make honey **if** there is a honeycomb. Students will also practice and recognize a connection between `if / else` blocks and `while` loops in this set of puzzles.

## Agenda

### Warm Up (10 min)

Introduction

### Bridging Activity - Conditionals (15 min)

Unplugged Activity Using Paper Blocks

Preview of Online Puzzles

### Main Activity (30 min)

Online Puzzles

### Wrap Up (15 min)

Journaling

### Extended Learning

### View on Code Studio

## Objectives

Students will be able to:

- Translate spoken language conditional statements into a program.
- Solve puzzles using a combination of looped sequences and conditionals.

## 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

- **Conditionals** - Statements that only run under certain conditions.

# Teaching Guide

## Warm Up (10 min)

### Introduction

Review the "Conditionals with Cards" activity with your students.

- What is a conditional statement?
- When is a conditional useful?
- What are some of the conditions you used in the last activity?

Now we're going to use conditionals with the Code.org bee to help him deal with some mysterious clouds. We don't know if his flowers have nectar or not, so we'll need to use conditionals to make sure that we collect nectar if it's there, but that we don't try to collect nectar from a flower that doesn't have any.

## Bridging Activity - Conditionals (15 min)

This activity will help bring the unplugged concepts from "Conditionals With Cards" 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

Print and cut out 2-3 `if / else` and blank action blocks from **Unplugged Blocks (Courses C-F) - Manipulatives** and pull out a deck of cards. Ask the class to come up with a couple of conditionals to use with the deck of cards like they did in "Conditionals with Cards." When the conditionals have been decided on as a class, fill in the blank part of the `if` block with the various card values that the kids came up with. Examples include "King of Hearts", "Even Numbered", or "Diamonds". Fill in the action blocks with the actions the students came up with. Make sure the students know the action blocks need to be directly under the `if` or `else` block. Below is an example.



Now shuffle the deck of cards and play "Conditionals with Cards" again. Flip through the deck card-by-card, reacting to cards if a conditional has been made for it.

### Preview of Online Puzzles

Pull up a puzzle from this lesson, we recommend puzzle 9.

- Ask the class what the bee should do when it gets to the cloud.
  - The bee should use a conditional to check for a flower or a honeycomb.
- Use the `if` at `flower / else` block. Ask the class what the bee should do if there's a flower. If there's not a flower, there will be a honeycomb. What should the bee do then?
  - The bee should `get nectar` if there is a flower and `make honey` if there is a honeycomb.

Fill in the rest of the code and press `Run`. Discuss with the class why this worked.

## Main Activity (30 min)

### Online Puzzles

These puzzles might sprout some questions, so have the students work in pairs or implement the "Ask three before you ask me" rule (have the students ask three other peers for help before they go to the teacher.) This will spark discussions that will develop each student's understanding.

## Code Studio levels

### Conditionals: If Statements

1

(click tabs to see student view)

### Prediction

2

(click tabs to see student view)

### Practice

3

4

5

6

7

(click tabs to see student view)

### Conditionals: If and If/Else Statements

8

(click tabs to see student view)

### Practice

9

(click tabs to see student view)

### Prediction

10

(click tabs to see student view)

### Challenge

11

(click tabs to see student view)

### Practice

12

13

(click tabs to see student view)

### Levels

Extra

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 today's lesson make you feel?
- What conditionals did you use in your code today?
- What are some other conditionals a bee might use? Examples include:
  - if there is a tree in front of me, buzz out of the way
  - if my wing is hurt, rest on the ground
  - if I see another bee, say "Hello!"

## Extended Learning

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

### True/False Tag

- Line students up as if to play **Red Light / Green Light**.
- Select one person to stand in front as the Caller.
- The Caller chooses a condition and asks everyone who meets that condition to take a step forward.
  - If you have a red belt, step forward.
  - If you are wearing sandals, take a step forward.
- Try switching it up by saying things like "If you are not blonde, step forward."

### **Nesting**

- Break students up into pairs or small groups.
- Have them write if statements for playing cards on strips of paper, such as:
  - If the suit is clubs
  - If the color is red
- Have students create similar strips for outcomes.
  - Add one point
  - Subtract one point
- Once that's done, have students choose three of each type of strip and three playing cards, paying attention to the order selected.
- Using three pieces of paper, have students write three different programs using only the sets of strips that they selected, in any order.
  - Encourage students to put some if statements inside other if statements.
- Now, students should run through all three programs using the cards that they drew, in the same order for each program.
  - Did any two programs return the same answer?
  - Did any return something different?

## **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: While Loops in Farmer

## Overview

By the time students reach this lesson, they should already have plenty of practice using `repeat` loops, so now it's time to mix things up.

*While loops* are loops that continue to repeat commands while a condition is met. `while` loops are used when the programmer doesn't know the exact number of times commands need to be repeated, but does know what condition needs to be true in order for the loop to continue repeating. For example, students will be working to fill holes and dig dirt in Farmer. They will not know the size of the holes or the height of the mountains of dirt, but the students will know they need to keep filling the holes and digging the dirt as long as the ground is not flat.

## Purpose

As your students continue to deepen their knowledge of loops, they will come across problems where a command needs to be repeated, but it is unknown how many times it needs to be repeated. This is where `while` loops come in. In today's lesson, students will develop a beginner's understanding of condition-based loops and also expand their knowledge of loops in general.

## Agenda

### Warm Up (10 min)

Introduction

### Bridging Activity (15 min)

Unplugged Activity Using Paper Blocks

Preview of Online Puzzles

### Main Activity (30 min)

Online Puzzles

### Wrap Up (15 min)

Journaling

[View on Code Studio](#)

## Objectives

Students will be able to:

- Distinguish between loops that repeat a fixed number of times and loops that repeat as long as a condition is true.
- Use a `while` loop to create programs that can solve problems with unknown values.

## 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 Teachers

- [Conditionals with Cards Sample Program - Teacher Prep Guide](#) [Make a Copy](#)

For the Students

- [Unplugged Blocks \(Courses C-F\) - Manipulatives](#)

## Vocabulary

- **Condition** - Something a program checks to see if it is true before allowing an action.
- **Loop** - The action of doing something over and over again.
- **Repeat** - To do something again.
- **While Loop** - A loop that continues to repeat while a condition is true.

# Teaching Guide

## Warm Up (10 min)

### Introduction

Use "while" in a sentence in front of the students. Ask the students what the word "while" means. If you were to say "while there is a hole, fill it with dirt" what would they do? How long would they do that?

When you use a word like "while", you are relying on a condition to tell the computer how long the loop should run. A condition is a statement that is tested and found to be true or false. In the case above, the condition is if there is a hole. It's only possible for there to be a hole or for there not to be a hole, thus the statement is only ever true or false.

Tell the students they will be learning about a new kind of loop. Previously, students only used loops to repeat a command a certain number of times. Here, they won't always know how many times to repeat the command, however, they will know when to stop or when to keep going. `while` loops allow the programmer to repeat a command as long as a condition is still true. In the previous example, the condition is the existence of a hole.

If there's time, have the students discuss other times using a `while` loop would be useful. Examples include:

- Running toward a ball **while** it is in front of you.
- Filling a glass **while** it has space for more liquid.
- Walk forward **while** there is a path ahead.

## Bridging Activity (15 min)

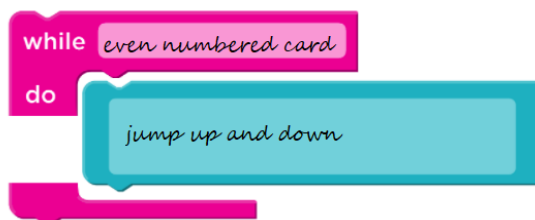
This activity will help bring the unplugged concepts from "Conditionals With Cards" into the online world that the students are moving into. It will also help bridge the concept of conditionals to a new type of loop- `while` loops. Choose one of the following to do with your class:

### Unplugged Activity Using Paper Blocks

Print and cut out 2-3 `while` / `do` blocks and blank action blocks from **Unplugged Blocks (Courses C-F) - Manipulatives** and pull out a deck of cards. Ask the class to come up with a couple of conditionals to use with the deck of cards like they did in "Conditionals with Cards." This time, instead of using the conditional in an `if` / `else` statement, they will be using it in a `while` loop.

When the conditionals have been decided on as a class, fill in the blank part of the `while` block with the various card groups that the kids came up with. Examples include "Even Numbered", "Red card", or "Diamonds". Fill in the action blocks with the actions the students came up with. Make sure the students know the action blocks need to be directly inside the `while` block. Line the blocks up sequentially so students can see how it runs as a program.

Below is an example.



Now shuffle the deck of cards and play "Conditionals with Cards" again. Flip through the deck card-by-card, reacting to cards if a conditional has been made for it. To maintain authenticity for this `while` loop experience, make sure you hold up the selected card for a good long while to give students time to react over several beats. Feel free to

jump back to previous cards to get the point across that once you have left a `while` loop, you continue moving forward and do not return to the previous loop just because the conditions match again.

## Preview of Online Puzzles

Pull up a puzzle from today's online Code Studio puzzles. We recommend Puzzle 6.

- Ask the class what the farmer should do when she gets to the pile of dirt.
  - She should use a `while` loop to start removing the dirt.
- Use the `while there is a pile / do` block. Ask the class what the farmer should do within the `while` loop.
  - The farmer should `remove 1`. The farmer will keep "removing 1 dirt" while there is dirt. In other words, when there is no dirt, `remove 1` will no longer execute!

Fill in the rest of the code and press `Run`. Discuss with the class why this worked.

## Main Activity (30 min)

### Online Puzzles

`While` loops are not always a difficult concept for students to understand, but if you think your class might struggle with these puzzles, we recommend pair programming. This will allow students to bounce ideas of each other before implementing the code. Pair programming works to increase confidence and understanding with topics like `while` loops.

### Code Studio levels

#### Practice

 1

 2

 3

*(click tabs to see student view)*

#### While Loops with the Farmer

 4

*(click tabs to see student view)*

#### Prediction

 5

*(click tabs to see student view)*

#### Practice

 6

 7

 8

 9

*(click tabs to see student view)*

#### Challenge

 10

*(click tabs to see student view)*

#### Practice

 11

 12

*(click tabs to see student view)*

#### Prediction

 13

*(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 do you feel about today's lesson?
- What is the difference between a `while` loop and a normal `repeat` loop?
- Give an example of a puzzle where you would use a `while` loop, but not use a `repeat` loop. Can you give an example of a puzzle where you would use a `repeat` loop, but not a `while` loop?

## 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: Until Loops in Maze

## Overview

In this lesson, students will learn about `until` loops. Students will build programs that have the main character repeat actions `until` they reach their desired stopping point.

## Purpose

This set of puzzles will work to solidify and build on the knowledge of loops by adding the `until` conditional. By pairing these concepts together, students will be able to explore the potential for creating complex and innovative programs.

## Agenda

### Warm Up (10 min)

Introduction

Preview of Online Puzzles

### Main Activity (30 min)

Online Puzzles

### Wrap Up (15 min)

Journaling

[View on Code Studio](#)

## Objectives

Students will be able to:

- Build programs with the understanding of multiple strategies to implement conditionals.
- Translate spoken language conditional statements and loops into a program.

## Preparation

- Play through the puzzles 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.
- **Loop** - The action of doing something over and over again.
- **Repeat** - To do something again.
- **Until** - A command that tells you to do something only up to the point that something becomes true.

# Teaching Guide

## Warm Up (10 min)

### Introduction

In this lesson, students will be creating loops that only run `until` a condition is true. Help the students understand how this works by leading them in group activities and having them do an action `until` some condition is true. *For example: Have students touch their nose until you tell them to stop.*

### Preview of Online Puzzles

Pull up a puzzle from today's Code Studio puzzles. We recommend Puzzle 4.

- Ask the class what the bird should repeat to get to the pig.
  - The bird should repeat `move forward, turn right, move forward, and then turn left`.
- Ask the class what they can use to repeat this code.
  - The bird should repeat this pattern *until* it reaches the pig.

Fill in the rest of the code using the `repeat until` loop and press `Run`. Discuss with the class why this worked.

## Main Activity (30 min)

### Online Puzzles

#### Code Studio levels

**Practice**  1 *(click tabs to see student view)*

**Repeat Until Statements**  2 *(click tabs to see student view)*

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

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

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

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

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

Bringing together concepts is not easy, but this set of lessons is meant to help students see the endless possibilities of coding when using conditions. If students struggle at all with understanding the similarities or differences between `while` loops and `until` loops, have them try to think of how they would use similar statements in their real lives.

## Wrap Up (15 min)

## Journaling

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

Journal Prompts:

- What was today's lesson about?
- How do you feel about today's lesson?
- What's the difference between an `until` loop and a `while` loop?

## 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: Harvesting with Conditionals

## Overview

Students will practice while loops, until loops, and if / else statements. All of these blocks use conditionals. By practicing all three, students will learn to write complex and flexible code.

## Purpose

Practicing the use of conditionals in different scenarios helps to develop a student's understanding of what conditionals can do. In the previous lesson, students only used conditionals to move around a maze. In this lesson, students will use conditionals to help the farmer know when to harvest crops. New patterns will emerge and students will use creativity and logical thinking to determine the conditions where code should be run and repeated.

## Agenda

### Warm Up (5 min)

Introduction

### Main Activity (30 min)

Online Puzzles

### Wrap Up (15 min)

Journaling

[View on Code Studio](#)

## Objectives

Students will be able to:

- Nest conditionals to analyze multiple value conditions using if, else if, else logic.
- Pair a loop and conditional statement together.

## Preparation

- Play through online puzzles 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.
- **Loop** - The action of doing something over and over again.
- **Repeat** - To do something again.
- **While Loop** - A loop that continues to repeat while a condition is true.

# Teaching Guide

## Warm Up (5 min)

### Introduction

Students shouldn't need as much of an introduction to concepts today because they have had practice with them in the previous lesson. Instead, you can share the story of the harvester.

The harvester is trying to pick crops like pumpkins, lettuce, and corn. However, the farmer has forgotten where she planted these crops, so she needs to check each plant before harvesting.

## Main Activity (30 min)

### Online Puzzles

Students will continue to work with `if / else` statements, `while` loops, and `until` loops. These puzzles are a bit more challenging, though, so encourage students to stick with them until they can describe what needs to happen for each program.

### Code Studio levels

#### The Harvester

 1

*(click tabs to see student view)*

#### Practice

 2

 3

 4

 5

 6

 7

*(click tabs to see student view)*

#### Challenge

 8

*(click tabs to see student view)*

#### Practice

 9

 10

*(click tabs to see student view)*

#### Prediction

 11

*(click tabs to see student view)*

#### Levels

 Extra

 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 do you feel about today's lesson?
- How can you see conditionals being useful in programs?
- What if people only spoke in `if/else` statements? What would be some advantages and disadvantages of this?

# 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: Binary Images

## Overview

Though many people think of binary as strictly zeros and ones, students will be introduced to the idea that information can be represented in a variety of binary options. This lesson takes that concept one step further as it illustrates how a computer can store even more complex information (such as images and colors) in binary, as well.

## 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 code and share it with peers to view as images. 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 stored.

## Agenda

### Warm Up (10 min)

Vocabulary  
Introduction to Binary

### Main Activity (20 min)

Binary Images - Worksheet

### Wrap Up (10 min)

Flash Chat: What did we learn?  
Journaling

### Assessment (10 min)

Binary Images - Assessment

### Extended Learning

### View on Code Studio

## Objectives

Students will be able to:

- Identify methods for encoding images into binary.
- Relate images to a peer using binary encoding.
- Reproduce an image, based on binary code.

## Preparation

- Print one worksheet and assessment per student.
- Make sure every student has a journal.
- (Optional) Gather groupings of items that can show opposites for students to use when coming up with their own binary encodings.

## Links

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

### For the Teachers

- **Binary Images** - Assessment Answer Key [Make a Copy](#)

### For the Students

- **Binary Images** - Unplugged Video ([download](#))
- **Binary Images** - Worksheet [Make a Copy](#)
- **Binary Images** - Assessment [Make a Copy](#)

## Vocabulary

- **Binary** - A way of representing information using only two options.
- **Binary Alphabet** - The two options used in your binary code.

# Teaching Guide

## Warm Up (10 min)

### Vocabulary

This lesson has two new terms:

- **Binary** - Say it with me: Bi-nare-ee

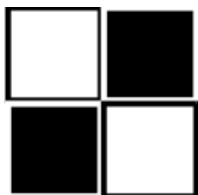
A way of representing information using only two options.

- **Binary Alphabet** - Say it with me: Bi-nare-ee Al-fa-bet

The two options used in your binary code.

### Introduction to Binary

What if we had a picture like this, where there's only two color options for each square, black or white.



How might we encode this so that someone else could recreate the picture without seeing it?

- Some students might think back to the Graph Paper Programming lesson. While there could be a lot of similarities, let them know that this is different enough that they should not use that lesson to guide this one

You may hear suggestions like: "Say 'white, black, white, black'."

- "That's a great suggestion! Now I'm going to break you up into pairs. Work with your teammate to decide on a binary alphabet."

Decide whether you want your pairs to share their encodings with the other groups ahead of time, and tell them if they will be creating a key, or keeping their methods secret.

- "Now, let's encode some images, just like a computer would!"

## Main Activity (20 min)

### Binary Images - Worksheet

Now it's the students' turn!

Activity Directions:

1. Divide students into pairs.
2. Have them choose an image with their partner.
3. Encourage them to figure out what their binary alphabet is going to be.
4. Have them encode their image using their new binary alphabet.
5. Instruct students to trade encodings with another team and see if they can figure out which picture the other worked on.
6. Choose a Level
  - Easy: Let the other team know what your encoding method was
  - Tough: Have the other team guess your encoding method.

## Wrap Up (10 min)



## Flash Chat: What did we learn?

- What did we learn today?
- What kind of binary alphabet did you create?
- Can you think of how you could encode an image using only your fingers?
- Do you think you could create a binary alphabet out of sounds?

## Journaling

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

### Journal Prompts:

- What was today's lesson about?
- How do you feel about today's lesson?
- What is a binary alphabet?
- What kind of information can you share using binary?

## Assessment (10 min)

### Binary Images - Assessment

Pass out this assessment for students to do individually. Try to save time at the end to go over answers.

## Extended Learning

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

### Storing Color Images

- If your class really gets the idea behind storing binary images, they may want to know how to do color images.
  - First, you'll need to discuss how color works using binary (as in **Binary Baubles - Thinkersmith Lesson**, page 21).
  - Then, introduce some images that use combinations of those colors
- Encourage your students to come up with ways to code these color images.



### Hexadecimal

- Take the idea of color one step further to introduce **hexadecimal color codes**.

## 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: Binary Images with Artist

## Overview

This series of online lessons will have students learning to make images using on and off.

## Purpose

This will help reinforce the fact that computers can do a multitude of things with 0s and 1s.

## Agenda

**Warm Up (15)**

**Main Activity (30)**

Online Puzzles

**Wrap Up (15)**

Journaling

[View on Code Studio](#)

## Objectives

Students will be able to:

- Create pictures using unique combinations of on and off
- Identify repeated sequences and break long codes up into smaller chunks that can be looped
- Utilize loops and binary code to recreate provided images

## Preparation

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

## Vocabulary

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

# Teaching Guide

## Warm Up (15)

To begin, it can be helpful to review the previous lesson, specifically different ways of using binary to indicate how to create an image on a grid. This stage will translate the unplugged activity into a simple, independent online lesson.

## Main Activity (30)

Online Puzzles

 Code Studio levels

**Practice**

 1

 2

 3

 4

 5

 6

 7

 8

 9

*(click tabs to see student view)*

**Prediction**

 10

*(click tabs to see student view)*

**Free Play**

 11

*(click tabs to see student view)*

**Levels**

 Extra

 Extra

*(click tabs to see student view)*

Not all of these images are intuitive. Encourage students to click "Run" to see what happens, even if their code isn't "finished" yet.

## Wrap Up (15)

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?
- Did you like drawing on the 8x8 grid or the 16x16 grid better? Why?
- Computers also store sounds using binary. Use your imagination to write down a possible way for that to work.

## 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 18: Be A Super Digital Citizen

## Overview



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

Online tools are empowering for kids, and they also come with big responsibilities. But do kids always know what to do when they encounter cyberbullying? Show your students appropriate ways to take action and resolve conflicts, from being upstanders to helping others in need.

## Purpose

Common Sense Education created this lesson to teach students how they can be upstanders when they see cyberbullying.

## Agenda

**Warm Up: Secret Superhero (5 min)**

Key Vocabulary

**Learn: Being an Upstander (10 min)**

**Create: Your Digital Citizen Superhero (15 min)**

**Wrap Up: Save the Day! Comic Strip (15 min)**

**Extended Learning**

**View on Code Studio**

## Objectives

Students will be able to:

- Reflect on the characteristics that make someone an upstanding digital citizen.
- Recognize what cyberbullying is.
- Show ways to be an upstander by creating a digital citizenship superhero comic strip.

## Preparation

- Review instructional materials.
- Print handout(s) for each student.
- Prepare colored pencils or markers for students.

## Links

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

For the Teachers

- **Be A Super Digital Citizen: Lesson Slides - Slide Deck**
- **Be A Super Digital Citizen: What Would A Super Digital Citizen Do? Answer Key - Student Handout** [Make a Copy](#)
- **Be A Super Digital Citizen: Lesson Quiz Answer Key - Website**

For the Students

- **Be A Super Digital Citizen: Super Digital Citizen - Student Video (download)**
- **Be A Super Digital Citizen: Digital Citizen Superhero - Student Handout** [Make a Copy](#)
- **Be A Super Digital Citizen: What Would A Super Digital Citizen Do? - Student Handout** [Make a Copy](#)
- **Be A Super Digital Citizen: Lesson Quiz - Form**

# Teaching Guide

## Warm Up: Secret Superhero (5 min)

### Key Vocabulary

- **cyberbullying:** using digital devices, sites, and apps to intimidate, harm, and upset someone
- **digital citizen:** someone who uses technology responsibly to learn, create, and participate
- **upstander:** a person who supports and stands up for someone else

**Before the lesson:** As an optional activity before the lesson, have students play the **E-volve** 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:** *Do you have a favorite superhero? If so, who is it and why? If not, why not? Take turns sharing with your partner. (Slide 4)*

Invite students to share out. Follow up by asking students to name specific super powers from their favorite superhero. Point out that one thing that all superheroes have in common is that they use their powers to help other people.

**Say:** *Today we're going to talk about how we can all be superheroes and help others. We're going to watch a video about being super digital citizens. As we watch, think about the question, "What does a super digital citizen do?"*

**Show** the video **Super Digital Citizen** on **Slide 5**.

**Invite** students to share out their responses. Their answers should be based on what the Super Digital Citizen helps Guts do:

- *Changes his password to be secure.*
- *Protects his tablet with a case and carries it in his backpack so that it won't break.*
- *Asks permission from Heart first before sharing a photo of her online.*

**Ask:** *You might have noticed that Guts became a superhero too. How did he do that?*

Invite students to respond. Explain that Guts became a superhero because he took steps to help himself and others be responsible online. Define **digital citizen** as *someone who uses technology responsibly to learn, create, and participate. (Slide 6)*

## Learn: Being an Upstander (10 min)

**Say:** *One situation a super digital citizen might see online is cyberbullying. **Cyberbullying** is when someone uses digital devices, sites, or apps to intimidate, harm, or upset someone. (Slide 7)*

Explain that cyberbullying can take many forms, including:

- Someone making fun of or pressuring someone else repeatedly
- Comments, memes, private messages, or chatting
- The person being bullied not knowing everyone who's doing the bullying (as people can hide their identity online)
- A group of people ganging up on someone

**Ask:** *So if you saw someone being cyberbullied, what's something you could do to stop it? Take turns sharing with your partner.*

Invite students to respond. Point out three ways that cyberbullying can be addressed:

- Defending or supporting the person being bullied
- Telling a trusted adult
- Addressing it directly with the bully

**Explain** to students that doing any of these things makes you an upstander. An **upstander** is a *person who supports and stands up for someone else*. (Slide 8)

**Distribute** the **What Would a Super Digital Citizen Do? Student Handout** to each student and have a student read the directions for Part 1 aloud. Allow students five minutes to complete the activity. (Slide 9)

## Create: Your Digital Citizen Superhero (15 min)

**Distribute** colored pencils or markers and tell students: *You're going to create your own digital citizen superhero, who will help people like Guts to become super digital citizens.*

**Distribute** the **Digital Citizen Superhero Student Handout** and read the directions aloud. (Slide 10)

**Optional:** Have students use **Marvel's Create Your Own Superhero**, save the image, and paste the image into the handout.

**Allow** students 10 minutes to work on their superheroes. If time permits, have students share out their superheroes with the class, or have them on display for a gallery walk.

## Wrap Up: Save the Day! Comic Strip (15 min)

**Have** students go back to their **What Would a Super Digital Citizen Do? Student Handout** and go to Part 2. Have a student read the directions aloud before students create their comic strip stories. (Slide 11)

**Optional:** Have students use a digital comic creation tool. See our recommendations: **Classroom-Friendly Websites and Apps for Making Comics**.

**Have** students complete the **Lesson Quiz**.

## Extended Learning

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

- **Family Activity**
- **Family Tips**
- Have students brainstorm and write an opinion piece on what the quote "with great power comes great responsibility" means in the digital world. How is having access to the internet a "great power"? What responsibilities do we have to ourselves and others? Students can post their work to a blog or your class page.

## Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **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](https://creativecommons.org/licenses/by-nc-nd/4.0/). 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 19: End of Course Project

## Overview

This capstone lesson takes students through the process of designing, developing, and showcasing their own projects!

## 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:

- 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 **Project Planning Guide** for each student (or pair).
- (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

- **Course D Project Planning Guide - Project Guide** [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 variety of projects. 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 **Course D 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

**Create your project** *(click tabs to see student view)*

**End of Course Project** 

 1

 1a

 1b

 1c

 1d

## 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**.