

# Express Course (2022)

Learn computer science by trying the lessons below at your own pace! Learn to create computer programs, develop problem-solving skills, and work through fun challenges! Make games and creative projects to share with friends, family, and teachers.

▼ Teacher resources

▼  Printing Options



For your owned section:

Select a section ▼

## ▼ Sequencing

### ▼ Lesson 1: Programming with Angry Birds

In this **skill-building** lesson, students will develop sequential algorithms to move a bird from one side of a maze to the pig on the other side. To do this, they will stack code blocks together in a linear sequence, making them move straight, turn left, or turn right.

 1

Video: Maze Intro - Programming with Blocks

 2-7

Skill Building

2

3

4

5

6

7

 8

Challenge

 9

Practice

 10

Prediction

 11

Practice

 12-13

Challenge

12

13

### ▼ Lesson 2: Debugging in Maze

In this **skill-building** lesson, students will encounter pre-written code that contains mistakes. They will need to step through the existing code to identify errors.

 1

Video: Debugging with Scrat

 2-7

Skill Building

- 2 3 4 5 6 7

 8

Challenge

 9

Prediction

 10

Practice

### ▼ Lesson 3: Collecting Treasure with Laurel

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

 1

Video: The Collector

 2-7

Skill Building

- 2 3 4 5 6 7

 8

Challenge

 9-11

Practice

- 9 10 11

 12

Prediction

 13


Practice

### ▼ Lesson 4: Creating Art with Code

In this **skill-building** lesson, students will take control of the Artist to complete drawings on the screen.

 1

Video: Artist Intro with JR Hildebrand

 2-7

Skill Building

2

3

4

5

6

7

 8

Challenge

 9

Practice

 10

Prediction

## ▼ Sprites

### ▼ Lesson 5: Swimming Fish in Sprite Lab


Students will program a simple animated underwater scene in this **skill-building lesson**.

 1

Video: Introducing Sprite Lab

 2

Prediction

 3-5

Skill Building


3

4

5

 6

Video: How to Make a Sprite

 7-8

Practice

7

8

 9

Free Play


### ▼ Lesson 6: Making Sprites

In this **skill-building** lesson, students will work through a series of programming levels on the computer, finishing with an open-ended “free play” task where they can build whatever they like. Students will write programs and learn about the two concepts at the heart of Sprite Lab: sprites and behaviors.

 1-7

Exploration: Sample Apps

1 2 3 4 5 6 7

 8-12

Skill Building

8 9 10 11 12

 13-16

Practice

13 14 15 16

 17-22

Free Play: Make a Scene

17 18 19 20 21 22

## ▼ Events

### ▼ Lesson 7: Sprites in Action

In this **skill-building** lesson, students will work through a series of programming levels on the computer, finishing with an open-ended “free play” task where they can build whatever they like. Students will write programs that respond to timed events and user input.

 1

Prediction


 2

Video: Sprites in Action

 3-7

Skill Building

3 4 5 6 7

 8-13

Practice

8 9 10 11 12 13



 14-19

Free Play: Make an Interactive Scene

14 15 16 17 18 19

















## ▼ Lesson 8: Virtual Pet with Sprite Lab

Students will create an interactive Virtual Pet that looks and behaves how they wish in this **mini-project** lesson. Students will use Sprite Lab's "Costumes" tool to customize their pet's appearance. They will then use events, behaviors, and other concepts they have learned to bring their project to life.

-  1 Exploration
-  2-6 Mini-Project: Create a Virtual Pet
  - 
  - 
  - 
  - 
  - 
-  7 Free Play

## ▼ Lesson 9: Dance Party

In this **skill-building** lesson, students will program an interactive dance party.

-  1 Video: Dance Party Warm Up
-  2 Practice
-  3 Video: Dance Party Events
-  4-5 Events
  - 
  - 
-  6 Video: Dance Party Measures
-  7-9 Measures
  - 
  - 
  - 
-  10 Video: Dance Party Properties
-  11-13 Properties
  - 
  - 
  - 

 14

Video: Dance Party Party On!

 15

Free Play

## ▼ Loops

### ▼ Lesson 10: Loops with Rey and BB-8

This **skill-building** lesson has students using loops to help the *Star Wars* character *BB-8* efficiently traverse a maze.

 1

Video: Programming with Rey and BB-8

 2

Practice

 3

Prediction

 4

Video: Repeat Blocks with BB-8

 5-9

Skill Building

5

6

7

8

9

 10

Challenge

 11-12

Practice

11

12

 13

Prediction

 14

Practice

### ▼ Lesson 11: Mini-Project: Sticker Art

This **mini-project** lesson builds on the understanding of loops. Students will create unique artwork with the Artist.

 1-8

Mini-project: Sticker Design

1

2

3

4

5

6

7

8

 9

Free Play

## ▼ Lesson 12: Nested Loops in Maze

In this **skill-building** lesson, students will learn how to program a loop inside of another loop.

 1-2

Practice

1

2

 3

Video: Nested Loops with the Bee

 4

Prediction

 5-9

Practice

5

6

7

8

9

 10

Challenge

 11-12

Practice

11

12

 13

Prediction

## ▼ Lesson 13: Snowflakes with Anna and Elsa

This **mini-project** lesson takes students through a series of exercises to create snowflake images using characters from the *Frozen* movies.

 1-4

Skill Building

1

2

3

4

 5-6

Practice


5

6

## ▼ Conditionals

### ▼ Lesson 14: Looking Ahead with Minecraft

This **skill-building** lesson gives students the chance to practice concepts that they have learned up to this point and get their first experience with conditionals!

 1-10

Skill Building

1

2

3

4

5

6

7

8

9

10

 11-13

Challenge

11

12

13

 14

Free Play

### ▼ Lesson 15: If/Else with Bee


In this **skill-building** lesson, your class will continue to code with conditionals, allowing them to write code that functions differently depending on the specific conditions the program encounters.

 1

Video: Conditionals: If Statements

 2

Prediction

 3-7

Skill Building

3

4

5

6

7

 8

Video: Conditionals: If and If/Else Statements

 9

Skill Building



 10

Prediction

 11

Challenge

 12-13

Practice

12 13

## ▼ Lesson 16: While Loops in Farmer

In this **skill-building** lesson, students will be working to fill holes and dig dirt in Farmer, but they will not know the size of the holes or the height of the mounds of dirt. To solve these puzzles, students will use a new kind of loop.

 1-3

Skill Building

1 2 3

 4

Video: While Loops with the Farmer

 5

Prediction

 6-9

Skill Building

6 7 8 9

 10

Challenge

 11-12

Practice

11 12

 13


Prediction

## ▼ Lesson 17: Conditionals in Minecraft: Voyage Aquatic

In this **context-setting** lesson, students will get the chance to practice content that they have learned up to this point, as well as getting a sneak peek at conditionals!

 1

Video: Minecraft: Voyage Aquatic Introduction

 2-4

Skill Building

2 3 4

 5

Video: Minecraft: Voyage Aquatic Repeat Until

 6-8

Skill Building

6 7 8

 9

Video: Minecraft: Voyage Aquatic Conditionals

 10-14

Skill Building

10 11 12 13 14

 15

Video: Minecraft: Voyage Aquatic Congratulations

 16

Free Play

## ▼ Lesson 18: Until Loops in Maze

In this **skill-building** 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.

 1

Skill Building

 2

Video: Repeat Until Statements

 3

Prediction

 4-8

Skill Building

4 5 6 7 8

 9

Challenge

 10

Practice

 11

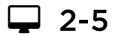
Prediction

## ▼ Lesson 19: Harvesting with Conditionals

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.



Video: Harvesting with Conditionals



Skill Building

2

3

4

5



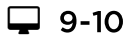
Practice

6

7



Challenge



Practice

9

10



Prediction

## ▼ Functions

### ▼ Lesson 20: Functions in Minecraft

In this **skill-building** lesson, students will begin to understand how functions can be helpful!



Video: Minecraft - The Agent



Skill Building

2

3



Video: Minecraft - Repeat Loops

 5-9

Skill Building

5 6 7 8 9

 10

Video: Minecraft - Functions

 11-14

Skill Building

11 12 13 14

 15

Video: Minecraft - Congratulations

 16

Free Play

## ▼ Lesson 21: Functions with Harvester

In this **skill-building** lesson, students will use conditionals with functions to harvest crops in Harvester.


 1-6

Practice

1 2 3 4 5 6

 7

Video: How to Create a Simple Function

 8-10

Practice

8 9 10

 11

Challenge

 12

Practice

 13

Prediction

## ▼ Lesson 22: Functions with Artist

In this **skill-building** lesson, students will use functions with the Artist.

 1

Prediction

 2-9

Practice

2

3

4

5

6

7

8

9

 10

Challenge

 11

Practice

 12

Prediction

 13

Free Play

## ▼ Variables

### ▼ Lesson 23: Text and Prompts

In this **skill-building** lesson, students will get practice with variables in Sprite Lab.

 1

Prediction

 2

Video: Text and Prompts

 3-8

Skill Building

3

4

5

6

7

8

 9-15

Practice

9

10

11

12

13

14

15

### ▼ Lesson 24: Counting with Variables

In this **skill-building** lesson, students will use variables to track a value that changes over time, like a counter. This lesson also includes a short **mini-project** in which students create a simple game.

 1-2

Prediction

1

2

 3-5

Skill Building


3

4

5

 6

Exploration: Clicker Game

 7-11

Mini-Project: Clicker Game

7

8

9

10

11

### ▼ Lesson 25: Using Variables with the Artist


In this **skill-building** lesson, students will explore the creation of repetitive designs using variables in the Artist environment. Students will learn how variables make code easier to write and easier to read. After guided puzzles, students will end in a free play level to show what they have learned and create new designs.

 1

Prediction

 2

Video: Variables in Artist

 3-6

Skill Building

3

4

5

6

 7

Free Play

### ▼ Lesson 26: Variables with the Bee

This **skill-building** lesson will help illustrate how variables can make programs more dynamic by allowing values to change while the code is running.

 1-7

Skill Building

1

2

3

4

5

6

7

 8-10

Practice

8


9

10

## ▼ For Loops

### ▼ Lesson 27: For Loops with Bee

This **skill-building** lesson focuses on for loops and using an incrementing variable to solve more complicated puzzles.

 1-2

Review

1

2

 3

Exploration: For Loops

 4

Video: For Loops

 5

Prediction

 6-8

Skill Building

6

7

8

 9-13

Practice

9

10

11

12

13

### ▼ Lesson 28: For Loops with Artist


In this **skill-building** lesson, students practice “for” loops with Artist. Students will complete puzzles to create complex designs and unique art.

 1

Video: For Loops

 2

Exploration

 3-7

Skill Building

3

4

5

6

7

 8

Free Play

## ▼ End of Course Project

### ▼ Lesson 29: End of Course Project

This **project** lesson takes students through the process of designing, developing, and showcasing new projects!

 1

Example Projects

 2

Create your project



# Lesson 1: Programming with Angry Birds

38 minutes

## Overview

In this **skill-building** lesson, students will develop sequential algorithms to move a bird from one side of a maze to the pig on the other side. To do this, they will stack code blocks together in a linear sequence, making them move straight, turn left, or turn right.

## Purpose

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (3 minutes)

Introduction

### Main Activity (30 minutes)

Programming with Angry Birds

### Wrap Up (5 minutes)

Reflection

### Extended Learning

### Cross-Curricular Opportunity

## Objectives

Students will be able to:

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

## Preparation

- Play through the puzzles to find any potential problem areas for your class.
- (Optional) Pick a couple of puzzles to do as a group with your class.
- Make sure every student has a reflection journal.

## Links

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

For the students

- **Unplugged Maze Blocks** - Resource

## Vocabulary

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

# Teaching Guide

## Warm Up (3 minutes)

### Introduction

**Model:** Pull up Puzzle 5 to do in front of the class. While working through this puzzle with the class, remind students that making mistakes is okay and remind them that the only way to be successful is to be persistent. Discuss what to do when a program doesn't work (debug it!) or how to get through the frustration that can come with working on a computer.

Next, you'll need to describe how the blocks in the workspace move the bird toward the pig. Show students how to drag blocks from the toolbox and connect them beneath the `when run` block, but don't solve the puzzle.

**Discuss:** Think about how we would get the bird to the pig using arrows. How do we use these blocks instead?

Have students use their fingers to point the direction that the bird should go next. Once you feel like you have a classroom consensus, try to get students to put into words which block will make that action happen. Roll your mouse over different options and have them shout "Yes" or "No".

Drag blocks into place one at a time, then click "Run" after each one. This will not only let them see how far the bird has gone, but set good habits for when they start working to solve their own puzzles.

Continue this pattern, fixing bugs as they arise, until the bird successfully gets to the pig.

### Additional Demonstration

We've included some multiple choice prediction levels. These could be used after finishing the Warm Up.

Prediction Levels:

- **Programming in Maze #1**
- **Programming in Maze #2**

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

#### 💡 Teaching Tip

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

## Main Activity (30 minutes)

### Programming with Angry Birds



1

Video: Maze Intro - Programming with Blocks

Show the students the right way to help classmates by: Don't sit in their chair Don't use their keyboard Don't touch their mouse Make sure the classmate can describe the solution before you walk away

2-7

Skill Building

2

3

4

5

6

7

8

Challenge

9

Practice

10

Prediction

11

Practice

12-13

Challenge

12

13

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

- Utilize Pair Programming whenever possible during the activity.
- Encourage students with questions/challenges to start by asking their partner.
- Unanswered questions can be escalated to a nearby group, who might already know the solution.
- Have students describe the problem that they're seeing. What is it supposed to do? What does it do? What does that tell you?
- Remind frustrated students that frustration is a step on the path to learning, and that persistence will pay off.
- If a student is still stuck after all of this, ask leading questions to get the student to spot an error on their own.

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

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

**Transition:** Have students grab their **\*Thinkspot Journals** and take a moment to reflect on the experience for themselves.

## Wrap Up (5 minutes)

### Reflection

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. When completed, this can be used as a review sheet for them to look at in the future.

#### Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- Draw an activity you like to do that you struggled with the first time. Draw or describe how you got better.

## Extended Learning

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

#### Create Your Own

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

## Cross-Curricular Opportunity

### Bugs, Bugs & More Bugs (45-60 minutes)

#### Computer Science + English Language Arts + Math

**Bugs, Bugs & More Bugs** is an optional activity aligned to Common Core ELA and Math standards, written by our teacher community. Students will practice debugging as they fix code, correct a math problem and proofread writing samples.

#### *Standards Addressed:*

- **CCSS.ELA-LITERACY.L.2.1.B:** Form and use frequently occurring irregular plural nouns (e.g., feet, children, teeth, mice, fish).
- **CCSS.ELA-LITERACY.L.2.2.A:** Capitalize holidays, product names, and geographic names.
- **CCSS.ELA-LITERACY.L.2.2.C:** Use an apostrophe to form contractions and frequently occurring possessives.
- **CCSS.MATH.CONTENT.2.OA.B.2:** Fluently add and subtract within 20 using mental strategies.
- **CCSS.MATH.CONTENT.2.NBT.A.4:** Compare two three-digit numbers based on meanings of the hundreds, tens, and ones digits, using  $>$ ,  $=$ , and  $<$  symbols to record the results of comparisons.



This work 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: Debugging in Maze

50 minutes

## Overview

In this **skill-building** lesson, students will encounter pre-written code that contains mistakes. They will need to step through the existing code to identify errors.

## Purpose

Students in your class might become frustrated with this lesson because of the essence of debugging. Debugging is a concept that is very important to computer programming. Computer scientists have to get really good at finding the bugs in their own programs. Debugging forces the students to recognize problems and overcome them while building critical thinking and problem solving skills.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction  
Vocabulary

### Main Activity (30 minutes)

Debugging with Scrat

### Wrap Up (5 minutes)

Reflection

### Extended Learning

## Objectives

Students will be able to:

- Modify an existing program to solve errors.
- Predict where a program will fail.
- Reflect on the debugging process in an age-appropriate way.

## Preparation

- Play through the lesson to find any potential problem areas for your class.
- (Optional) Pick a couple of puzzles to do as a group with your class.
- Make sure every student has a Reflection Journal.

## Links

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

For the students

- **Pair Programming** - Video

## Vocabulary

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

## Teaching Guide

# Warm Up (15 minutes)

## Introduction

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

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

## Vocabulary

This lesson has three new and important vocabulary words:

- **Bug** - Say it with me - Buhh-g.

Something that is going wrong. An error.

- **Debugging** - Say it with me: Dee-bug-ing.

To find and fix errors.

- **Persistence** - Say it with me: Purr-siss-tense.

Not giving up. Persistence works best when you try things many different ways, many different times.

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

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

# Main Activity (30 minutes)


## Debugging with Scrat

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

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



Video: Debugging with Scrat

 2-7

Skill Building

2

3

4

5

6

7

 8

Challenge

 9

Prediction

 10

Practice

 11-12

Lesson Extras



## Wrap Up (5 minutes)

### Reflection

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.

#### Prompts:

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

## Extended Learning

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

#### Planting bugs

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

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





This work 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: Collecting Treasure with Laurel

50 minutes

## Overview

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

## Purpose

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (5 minutes)

Introduction

### Bridging Activity (10 minutes)

Previewing Online Puzzles as a Class

### Main Activity (30 minutes)

Collecting Treasure with Laurel

### Wrap Up (5 minutes)

Reflection

## Objectives

Students will be able to:

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

## Preparation

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

## Vocabulary

- **Algorithm** - A list of steps to finish a task.
- **Program** - An algorithm that has been coded into something that can be run by a machine.
- **Programming** - The art of creating a program.

# Teaching Guide

## Warm Up (5 minutes)

### Introduction

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

## Bridging Activity (10 minutes)

### Previewing Online Puzzles as a Class

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

## Main Activity (30 minutes)

### Collecting Treasure with Laurel

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



1

Video: The Collector



2-7

Skill Building

2

3

4

5

6

7



8

Challenge



9-11

Practice

9

10

11

12

Prediction

13

Practice

14-15

Lesson Extras



## Wrap Up (5 minutes)

### Reflection

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.

#### Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- Draw a maze that you might solve with the blocks you used today.



This work 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: Creating Art with Code

50 minutes

## Overview

In this **skill-building** lesson, students will take control of the Artist to complete drawings on the screen.

## Purpose

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (10 minutes)

Introduction

### Main Activity (30 minutes)

Creating Art with Code

### Wrap Up (10 minutes)

Reflection

### Extended Learning

### Cross-Curricular Opportunity

## Objectives

Students will be able to:

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

## Preparation

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

## Links

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

For the students

- **Artist Introduction** - Video
- **Turns & Angles** - Video
- **Turns & Angles** - Handout

## Teaching Guide

### Warm Up (10 minutes)

Introduction

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

**\*Artist Introduction** (1.5 minutes long)

**\*Turns & Angles** (2 minutes long)

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

**Discuss** the square and triangle shapes from the document.

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

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

## Main Activity (30 minutes)

### Creating Art with Code

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

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



1

**Video: Artist Intro with JR Hildebrand**



2-7

**Skill Building**

2

3

4

5

6

7

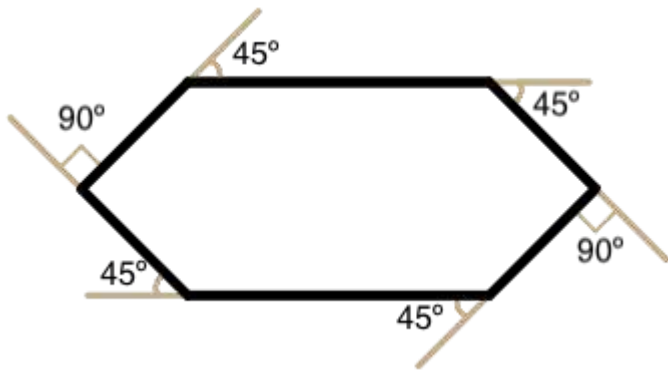


8

**Challenge**

💡 Teaching Tip

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



9

Practice

10

Prediction

11-12

Lesson Extras



## Wrap Up (10 minutes)

### Reflection

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.

#### Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- What are the interior angles that make up a square. What about for a triangle?
- Sketch a simple shape on your paper and imagine the code used to draw it. Can you write that code out next to the shape?

## Extended Learning

Use this activity to enhance student learning. It can be used as an outside of class activity or other enrichment.

### The Copy Machine

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

# Cross-Curricular Opportunity

## Shapes & Landscapes (45-60 minutes)

 **Computer Science + English Language Arts + Math + Science**

**Shapes & Landscapes** is an optional activity aligned to Common Core ELA, Common Core Math and Next Generation Science Standards, written by our teacher community. Students are asked to design a dam to prevent future flooding. Using code, you will create a blueprint to show the local town council how your dam will look.

*Standards Addressed:*

- **CCSS.ELA-LITERACY.W.2.8:** Recall information from experiences or gather information from provided sources to answer a question.
- **CCSS.MATH.CONTENT.2.MD.A.3:** Estimate lengths using units of inches, feet, centimeters, and meters.
- **NGSS.2-ESS2-1:** Compare multiple solutions designed to slow or prevent wind or water from changing the shape of the land.



This work 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: Swimming Fish in Sprite Lab

45 minutes

## Overview

Students will program a simple animated underwater scene in this **skill-building lesson**.

## Purpose

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (10 minutes)

Introduction

### Main Activity (20 minutes)

Swimming Fish with Sprite Lab

### Wrap Up (15 minutes)

Reflection

## Objectives

Students will be able to:

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

## Preparation

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

## Links

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

For the teachers

- **Sprite Lab Documentation** - Resource
- **Swimming Fish Teacher Sandbox**

## Vocabulary

- **Behavior** - An action that a sprite performs continuously until it's told to stop.
- **Sprite** - A graphic on the screen with a location, size, and appearance.

## Teaching Guide

# Warm Up (10 minutes)

## Introduction

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

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

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

This demonstration and discussion can help students move into the Sprite Lab environment.

## Swimming Fish Teacher Sandbox

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

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

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

### Content Corner

Sprite Lab works differently in some ways from the other online tools in the course. Most importantly, all code runs in order and immediately unless attached to an event block. Telling a sprite to begin and start the same behavior won't result in any observable effect because there is no time between each action.

# Main Activity (20 minutes)

## Swimming Fish with Sprite Lab

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

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

### Teaching Tip

Encourage students with questions/challenges to start by asking their partner. Unanswered questions can be escalated to a nearby group, who might already know the solution. Have students describe the problem that they're seeing:

- What is it supposed to do?
- What does it do?
- What does that tell you?



Video: Introducing Sprite Lab



Prediction



Skill Building

3

4

5



Video: How to Make a Sprite



Practice

7

8



Free Play

## Wrap Up (15 minutes)

### Reflection

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.

#### Prompts:

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



This work 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: Making Sprites

45 minutes

## Overview

In this **skill-building** lesson, students will work through a series of programming levels on the computer, finishing with an open-ended “free play” task where they can build whatever they like. Students will write programs and learn about the two concepts at the heart of Sprite Lab: sprites and behaviors.

## Purpose

This lesson is designed to introduce students to programming in Sprite Lab. Students will begin to form an understanding of the programming model of this tool, and explore ways they can use it to express themselves.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

**Warm Up (10 minutes)**

Exploring Apps

**Main Activity (30 minutes)**

**Wrap Up (5 minutes)**

Reflection

## Teaching Guide

## Objectives

Students will be able to:

- Create an animation using sprites, and behaviors.
- Create new sprites and assign them costumes and behaviors.

## Preparation

Play through the levels and review the **\*lesson slides**.

## Links

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

For the teachers

- **Sprite Lab Documentation** - Resource

## Vocabulary

- **Algorithm** - A list of steps to finish a task.
- **Behavior** - An action that a sprite performs continuously until it's told to stop.
- **Program** - An algorithm that has been coded into something that can be run by a machine.
- **Sprite** - A graphic on the screen with a location, size, and appearance.

## Warm Up (10 minutes)

### Exploring Apps (10 minutes)

**Do This:** Have students read the provided code and make a prediction about what will happen. After pressing “Run”, provide time for students to reflect on anything they found interesting or that surprised them.

 1-7

Exploration: Sample Apps

1

2

3

4

5

6

7

**Discuss:** What were the most interesting choices you were able to make with these apps? Was there ever a time you wished you could change something in the app but were not given the choice?

**Prompt:** Think about an app you have used, and come up with a way to make it better by giving the user more choices.

As time allows, have students consider the prompt above through one or more of the following methods:

- Have volunteers offer ideas in a whole-group discussion.
- Engage in a small discussion with pairs or small groups.

## Main Activity (30 minutes)

### Skill Building and Practice (20 mins)

These activities are suitable for independent learning or pair programming.

Students should complete the Skill Building levels and then spend any remaining time choosing from the various Practice activities.

 8-12

Skill Building

8

9

10

11

12

 13-16

Practice

13

14

15

16

### Free Play (10 mins)

In the final level in the lesson, students will be able to make their own project.

 17-22

Free Play: Make a Scene

## Wrap Up (5 minutes)

### Reflection

Today you learned how to write code to create programs in Sprite Lab.

**Program** - An algorithm that has been coded into something that can be run by a machine.

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

### 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.

**Prompts:** What advice would you have for a friend who is interested in creating their own programs in Sprite Lab?



This work 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: Sprites in Action

55 minutes

## Overview

In this **skill-building** lesson, students will work through a series of programming levels on the computer, finishing with an open-ended “free play” task where they can build whatever they like. Students will write programs that respond to timed events and user input.

## Purpose

This lesson is designed to introduce students to programming with events. Throughout this course, events will be the primary way students change how their programs run over time or in response to the user.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Follow the Algorithm

### Main Activity (35 minutes)

Sprites in Action

### Wrap Up (5 minutes)

Reflection

## Objectives

Students will be able to:

- Create an interactive animation using events.
- Develop programs that respond to timed events.
- Develop programs that respond to user input.

## Preparation

Play through the levels and review the lesson slides.

## Links

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

For the teachers

- **Sprites in Action** - Slides

## Vocabulary

- **Algorithm** - A list of steps to finish a task.
- **Event** - An action that causes something to happen.

## Teaching Guide

### Warm Up (15 minutes)

**Do this:** Review or introduce the word **algorithm**: *a list of steps to complete a task.*

Follow the Algorithm

## Remarks

Today we're going to start with a quick game. In this game, I will be giving you all commands to follow and your job is to follow the algorithm. The commands will always start with "begin" or "stop". If I tell you to begin doing something, you need to start that behavior and keep it going until I tell you to stop. I might also tell you to "stop everything" which means you can end all of the behaviors I've given you so far. We'll play 3 rounds and after each round I'll ask you all a couple questions.

### Teaching Tip

Be sure to give a little space between commands. For each of these sections, consider running through the entire sequence without any discussion and later repeating it again after everyone has had a chance to debrief and process any confusion.

## Round 1 (Basic):

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

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

Students should understand that multiple behaviors can happen simultaneously.

## Round 2 (Intermediate):

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

### **Discussion:**

*What kinds of instructions caused people to make mistakes? What strategies do you think are helpful for making sure you follow instructions correctly? Why is it important to keep track of each behavior separately?*

Students should understand that each behavior needs to be stopped individually. They need to keep track of each one separately.

## Challenging

- Begin crouching.
- Begin tapping your head.



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

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

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

*How can you jump up and down while crouching?*

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

*When you were told to spin in two opposite directions what did you see people do?*

*What would happen if you were told to spin left and right at the exact same time?*

Students should understand that some behaviors conflict with each other which can result in unexpected outcomes. It's possible for two opposite behaviors to effectively cancel each other out.

## Main Activity (35 minutes)

### Sprites in Action

#### Prediction (2 mins)

**Do This:** Have students read the provided code and make a prediction about what will happen. After pressing “Run”, provide time for students to reflect on anything they found interesting or that surprised them.



**Prediction**

#### Video: Sprites in Action (3 mins)

**Do This:** Play the video, “Sprites in Action”




**Video: Sprites in Action**

## Skill Building and Practice (20 mins)

**Transition:** Have students move to their computer and sign in. These activities are suitable for independent learning or pair programming.

Students should complete the Skill Building levels and then spend any remaining time choosing from the various Practice activities.

 3-7

**Skill Building**

3

4

5

6

7

 8-13

**Practice**

8

9

10

11

12

13

## Free Play (10 mins)

**Transition:** Students should now skip to the final level in the lesson where they will see the same free play choices as the previous lessons. Encourage students to add events to their projects from last time, or begin a new project.

 14-19

**Free Play: Make an Interactive Scene**

14

15

16

17

18

19

## Wrap Up (5 minutes)

### Reflection

#### *Remarks*

Today you learned to create programs that change over time using events.

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

### 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.

**Prompts:** Arrow keys are one way to make a program more interactive. What other ways could we give a user control over what happens in an app or game?



This work 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: Virtual Pet with Sprite Lab

60 minutes

## Overview

Students will create an interactive Virtual Pet that looks and behaves how they wish in this **mini-project** lesson. Students will use Sprite Lab's "Costumes" tool to customize their pet's appearance. They will then use events, behaviors, and other concepts they have learned to bring their project to life.

## Purpose

This lesson allows students to apply programming concepts from prior lessons in another creative context.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction

### Main Activity (30 minutes)

Virtual Pet with Sprite Lab

### Wrap Up (15 minutes)

Reflection

### Cross-Curricular Opportunity

## Objectives

Students will be able to:

- Create an interactive virtual pet using events, behaviors, variables, and custom art.
- Program solutions to problems that arise when designing a virtual pet, like feeding it or monitoring its happiness.

## Preparation

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

## Links

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

For the teachers

- **Sprite Lab Documentation** - Resource

For the students

- **Pause and Think Online** - Video

## Vocabulary

- **Behavior** - An action that a sprite performs continuously until it's told to stop.
- **Event** - An action that causes something to happen.

# Teaching Guide

## Warm Up (15 minutes)

### Introduction

Introduce the Sprite Lab "Costumes" tool that allows students to draw their own costumes.

**Review:** Ask students questions about events and behaviors.

- Do you remember what an event is?
- Do you remember what a behavior is?
- Can you remember some of the behaviors you have used ? What do they do?
  - patrolling
  - jittering
  - spinning right/left

**Display:** Begin by showing Level 1 of today's lesson to your students.

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

**Display:** Show Level 2. Briefly demonstrate how to do the following:

- Navigate between the *Code* and *Costumes* tabs.
- Draw a costume.
- Choose a costume from the costume library.
- Change the virtual pet's sprite's costume to a custom one.

## Main Activity (30 minutes)

### Virtual Pet with Sprite Lab

**Goal:** Today, students will be creating their own virtual pet! They will begin by drawing or selecting a new costume for a sprite. Then they will create events that cause actions and behaviors upon interaction.

### Online Puzzles

**Transition:** Move students to their computers. Encourage students to follow the instructions for each puzzle. Help them realize that this is a creative activity, intended to help them learn Sprite Lab.

#### 💡 Teaching Tip

Encourage students with questions/challenges to start by asking their partner. Unanswered questions can be escalated to a nearby group, who might already know the solution. Have students describe the problem that they're seeing:

- What is it supposed to do?
- What does it do?
- What does that tell you?

**Reminder:** If puzzles are sharable, 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**.

 1

Exploration

 2-6

Mini-Project: Create a Virtual Pet

2

3

4

5

6

 7

Free Play

## Wrap Up (15 minutes)

### Reflection

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.

#### Prompts:

- What was today's lesson about?
- How do you feel about today's lesson?
- What other options would you like to be able to have your pet do?

## Cross-Curricular Opportunity

### Providing Energy for your Virtual Pet (60-90 minutes)

#### Computer Science + English Language Arts + Science

**Providing Energy for your Virtual Pet** is an optional activity aligned to Common Core ELA and Next Generation Science Standards, written by our teacher community. Students will create a scientific model that shows how their virtual pet will receive the energy it needs to survive.

#### *Standards Addressed:*

- **CCSS.ELA.RI.5.7:** Draw on information from multiple print or digital sources, demonstrating the ability to locate an answer to a question quickly or to solve a problem efficiently.
- **CCSS.ELA.RI.5.9:** Integrate information from several texts on the same topic in order to write or speak about the subject knowledgeably.
- **5-PS3-1:** Use models to describe that energy in animals' food (used for body repair, growth, motion, and to maintain body warmth) was once energy from the sun.
- **5-LS2-1:** Develop a model to describe the movement of matter among plants, animals, decomposers, and the environment

- **SL.5.5:** Include multimedia components (e.g., graphics, sound) and visual displays in presentations when appropriate to enhance the development of main ideas or themes.



This work 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: Dance Party

12 minutes

## Overview

In this **skill-building** lesson, students will program an interactive dance party.

## Purpose

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

**Warm Up (5 minutes)**

**Activity (30-45 minutes)**

**Code Your Own Dance Party**

**Level by Level Support**

**Wrap Up (7 minutes)**

**Reflection**

**Cross-Curricular Opportunity**

## Objectives

Students will be able to:

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

## Preparation

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

- **Open-ended Programming Levels** - Answer Key
- **Spotify Playlist (all ages)**

For the students

- **Dance Party Project Guide** - Worksheet

## Vocabulary

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

- **Program** - An algorithm that has been coded into something that can be run by a machine.
- **code** - to write code, or to write instructions for a computer.

## Teaching Guide

### Warm Up (5 minutes)

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** to help kick things off. Capture your class's moves on video.



### 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. For students under 13, we limit the music to this filtered list. If you would like to use the filtered list with older students, you can share **\*Spotify Playlist** with your classroom.



# Level by Level Support

## Warm Up



### Video: Dance Party Warm Up

- 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.



### Practice

## Events



### Video: Dance Party Events

## Events Level 4

- Levels 4-5 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.

## Events Level 5

- 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.



### Events

4

5

## Measures



### Video: Dance Party Measures

## Measures Level 7

- Levels 7-8 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.

## Measures Level 8

- 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.

## Measures Level 9

- Level 9 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.

 7-9

Measures

7

8

9

## Properties

 10

Video: Dance Party Properties

## Properties Level 11

- Levels 11-13 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.

## Properties Level 12

- 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.

## Properties Level 13

- 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!

 11-13

Properties

11

12

13

 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.

## Party On!

 14

Video: Dance Party Party On!

## Party On! Level 15

- 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!

 15

Free Play

 16-17

Lesson Extras





## Wrap Up (7 minutes)

### Reflection (5 minutes)

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.

**Prompts:**

- What is something you enjoyed about today's activity?
- What is the connection between creativity and computer science?
- What is CS and why does it matter.

## Cross-Curricular Opportunity

### Survival of the Dancers (45-60 minutes)

#### **Computer Science + Science**

**Survival of the Dancers** is an optional activity aligned to Next Generation Science Standards, written by our teacher community. Using Dance Party, students will select at least one organism (dancer) which is well suited for the environment (world) they create. Students are encouraged to utilize the “properties” blocks of code to customize the characteristics of each organism (dancer).

*Standards Addressed:*

- **NGSS.3-LS4-2:** Use evidence to construct an explanation for how the variations in characteristics among individuals of the same species may provide advantages in surviving, finding mates, and reproducing.
- **NGSS.3-LS4-3:** Construct an argument with evidence that in a particular habitat some organisms can survive well, some survive less well, and some cannot survive at all.
- **OPTIONAL EXTENSION: NGSS.3-LS4-4:** Make a claim about the merit of a solution to a problem caused when the environment changes and the types of plants and animals that live there may change.



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

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

# Lesson 10: Loops with Rey and BB-8

45 minutes

## Overview

This **skill-building** lesson has students using loops to help the *Star Wars* character *BB-8* efficiently traverse a maze.

## Purpose

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (10 minutes)

Introduction

### Main Activity (30 minutes)

Loops with Rey and BB-8

### Wrap Up (5 minutes)

Reflection

### Extended Learning

## Objectives

Students will be able to:

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

## Preparation

- Play through the puzzles to determine if there will be any problem areas for your class.
- Make sure every student has a Reflection Journal.

## Links

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

For the students

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

## Vocabulary

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

## Teaching Guide

# Warm Up (10 minutes)

## Introduction

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

Pull up the online puzzles and choose a puzzle to do in front of the class. We recommend puzzle 10 for its staircase pattern. Ask the students to write a program to solve the puzzle on paper. Have the students circle repeated chunks and label with the number of repeats.

## Main Activity (30 minutes)

### Loops with Rey and BB-8



Video: Programming with Rey and BB-8



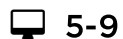
Practice



Prediction



Video: Repeat Blocks with BB-8



Skill Building

5

6

7

8

9



Challenge



Practice

11

12



Prediction



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

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

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

## Wrap Up (5 minutes)

### Reflection

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.

#### Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- How did loops make your program easier to write?
- Think of something that repeats over and over again. What might the program for that look like?

## Extended Learning

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

#### So Moving

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

#### Connect It Back

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



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



# Lesson 11: Mini-Project: Sticker Art

60 minutes

## Overview

This **mini-project** lesson builds on the understanding of loops. Students will create unique artwork with the Artist.

## Purpose

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

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction

### Main Activity (30 minutes)

Sticker Art with Loops

### Wrap Up (15 minutes)

Reflection

### Cross-Curricular Opportunity

## Objectives

Students will be able to:

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

## Preparation

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

## Vocabulary

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

## Teaching Guide

### Warm Up (15 minutes)

#### Introduction

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

- Create a dance, putting repeated steps in loops.
- As a class, play through a puzzle from the last lesson, "Loops with Rey and BB-8".

- Review how to use Artist by playing through a puzzle from "Creating Art with Code".
- Preview a puzzle from this lesson.

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

## Main Activity (30 minutes)

### Sticker Art with Loops

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

#### 💡 Teaching Tip

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

### Mini-project: Sticker Design

🖥️ 1-8

Mini-project: Sticker Design

1

2

3

4

5

6

7

8

🖥️ 9

Free Play

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 computers. Students might also enjoy drawing some of the shapes and figures on paper before they program it online. (When drawing stamps, it can be easier to symbolize those with simple shapes like circles and squares.)

## Wrap Up (15 minutes)

### Reflection

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.

#### Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?

- What was the coolest shape or figure you programmed today? Draw it out!
- What is another shape or figure you would like to program? Can you come up with the code to create it?

## Cross-Curricular Opportunity

### Loopy Forms & Their Functions (45-60 minutes)

 **Computer Science + English Language Arts + Math + Science**

**Loopy Forms & Their Functions** is an optional activity aligned to Common Core ELA, Common Core Math and Next Generation Science Standards, written by our teacher community. Artist needs to help his friends design their house by drawing blueprints. It is important that students understands basic shapes and how each shape serves different functions in the house blueprints. They will then write opinion pieces, supply reasons and use linking words to justify their design choices.

*Standards Addressed:*

- **CCSS.ELA-LITERACY.W.2.1:** Write opinion pieces in which they introduce the topic or book they are writing about, state an opinion, supply reasons that support the opinion, use linking words (e.g., because, and, also) to connect opinion and reasons, and provide a concluding statement or section.
- **CCSS.MATH.CONTENT.2.G.A.1:** Recognize and draw shapes having specified attributes, such as a given number of angles or a given number of equal faces.1 Identify triangles, quadrilaterals, pentagons, hexagons, and cubes.
- **NGSS.K-2-ETS1-2:** Develop a simple sketch, drawing, or physical model to illustrate how the shape of an object helps it function as needed to solve a given problem.



This work 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: Nested Loops in Maze

55 minutes

## Overview

In this **skill-building** lesson, students will learn how to program a loop inside of another loop.

## 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 to 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 to plenty of practice for students to solidify and build on their understanding of looping in programming.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (10 minutes)

Introduction

### Main Activity (30 minutes)

Nested Loops in Maze

### Wrap Up (15 minutes)

Reflection

## Teaching Guide

### Warm Up (10 minutes)

## Objectives

Students will be able to:

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

## Preparation

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

# Introduction

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

- *What do loops do?*
  - Loops repeat a set of commands.
- *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 minutes)

### Nested Loops in Maze

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.

 1-2

Practice

1

2

 3

Video: Nested Loops with the Bee

 4

Prediction

 5-9

Practice

5

6

7

8

9

 10

Challenge

 11-12

Practice

11

12

 13

Prediction

 14-15

Lesson Extras





## Wrap Up (15 minutes)

### Reflection

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.

#### 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.



This work 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: Snowflakes with Anna and Elsa

60 minutes

## Overview

This **mini-project** lesson takes students through a series of exercises to create snowflake images using characters from the *Frozen* movies.

## Purpose

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

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction

### Main Activity (30 minutes)

Snowflakes with Anna and Elsa

### Wrap Up (15 minutes)

Reflection

## Teaching Guide

### Warm Up (15 minutes)

#### Introduction

Ask the class to discuss the last set of puzzles.

## Objectives

Students will be able to:

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

## Preparation

- Play through the lesson to find and potential problem areas for your class.
- Make sure every student has a Reflection Journal.

## Vocabulary

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


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

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

## Main Activity (30 minutes)

### Snowflakes with Anna and Elsa

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

 1-4

**Skill Building**

1

2

3

4

 5-6

**Practice**

5

6

## Wrap Up (15 minutes)

### Reflection

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.

#### Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- When do you use a loop? When do you use a nested loop?
- How would the code for your snowflake look different if you only used one loop? No loops? Can you draw out an example?



This work 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: Looking Ahead with Minecraft

45 minutes

## Overview

This **skill-building** lesson gives students the chance to practice concepts that they have learned up to this point and get their first experience with conditionals!

## Purpose

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction

### Main Activity (30 minutes)

Looking Ahead with Minecraft

### Wrap Up

Reflection

### Extended Learning

## Objectives

Students will be able to:

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

## Preparation

- Play through the puzzles associated with this lesson to find any potential problem areas for your class.
- Make sure every student has a Reflection Journal.

## Vocabulary

- **Condition** - Something a program checks to see if it is true before allowing an action.
- **Conditionals** - Statements that only run under certain conditions.

## Teaching Guide

### Warm Up (15 minutes)

#### Introduction

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

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

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

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

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

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

## Main Activity (30 minutes)

### Looking Ahead with Minecraft

#### Online Puzzles

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

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

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



1-10

Skill Building

1

2

3

4

5

6

7

8

9

10



11-13

Challenge

11

12

13



14

Free Play

## Wrap Up

### Reflection

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.

**Prompts:**

- Draw a feeling face to show how you felt during today's lesson.
- Draw something else you could have built in this *minecraft* world.
- Can you draw a scene where someone is using a conditional?

## Extended Learning

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

### More Minecraft

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



This work 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: If/Else with Bee

55 minutes

## Overview

In this **skill-building** lesson, your class will continue to code with conditionals, allowing them to write code that functions differently depending on the specific conditions the program encounters.

## Purpose

Students will practice using conditionals 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.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (10 minutes)

Introduction

### Main Activity (30 minutes)

If/Else with Bee

### Wrap Up (15 minutes)

Reflection

### Extended Learning

## Objectives

Students will be able to:

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

## Preparation

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

## Vocabulary

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

## Teaching Guide

### Warm Up (10 minutes)

#### Introduction

We are 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.

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 minutes)

### If/Else with Bee

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.



1

**Video: Conditionals: If Statements**



2

**Prediction**



3-7

**Skill Building**

3

4

5

6

7



8

**Video: Conditionals: If and If/Else Statements**



9

**Skill Building**



10

**Prediction**



11

**Challenge**



12-13

**Practice**



## Wrap Up (15 minutes)

### Reflection

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.

#### 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?



This work 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: While Loops in Farmer

60 minutes

## Overview

In this **skill-building** lesson, students will be working to fill holes and dig dirt in Farmer, but they will not know the size of the holes or the height of the mounds of dirt. To solve these puzzles, students will use a new kind of loop.

## 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.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction

### Main Activity (30 minutes)

While Loops in Farmer

### Wrap Up (15 minutes)

Reflection

## 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 Reflection 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)**

## 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 (15 minutes)

### 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.

### 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 minutes)

### While Loops in Farmer

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 off each other before implementing the code. Pair programming works to increase confidence and understanding with topics like while loops.



1-3

Skill Building

1 2 3

 4

Video: While Loops with the Farmer

 5

Prediction

 6-9

Skill Building

6

7

8

9

 10

Challenge

 11-12

Practice

11

12

 13

Prediction

 14

Lesson Extras

## Wrap Up (15 minutes)

### Reflection

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.

#### 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?



This work 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: Conditionals in Minecraft: Voyage Aquatic

60 minutes

## Overview

In this **context-setting** lesson, students will get the chance to practice content that they have learned up to this point, as well as getting a sneak peek at conditionals!

## Purpose

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction

### Main Activity (30 minutes)

Conditionals in Minecraft: Voyage Aquatic

### Wrap Up (15 minutes)

Reflection

### Extended Learning

## Objectives

Students will be able to:

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

## Preparation

- Play through the puzzles associated with this lesson to find any potential problem areas for your class.
- Make sure every student has a Reflection Journal.

## Vocabulary

- **Condition** - Something a program checks to see if it is true before allowing an action.
- **Conditionals** - Statements that only run under certain conditions.

## Teaching Guide

### Warm Up (15 minutes)

#### Introduction

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

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

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

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

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

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

## Main Activity (30 minutes)

### Conditionals in Minecraft: Voyage Aquatic

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

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

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



1

**Video: Minecraft: Voyage Aquatic Introduction**



2-4

**Skill Building**

2

3

4



5

**Video: Minecraft: Voyage Aquatic Repeat Until**



6-8

**Skill Building**

6

7

8



9

**Video: Minecraft: Voyage Aquatic Conditionals**

 10-14

Skill Building

10

11

12

13

14

 15

Video: Minecraft: Voyage Aquatic Congratulations

 16

Free Play

## Wrap Up (15 minutes)

### Reflection

#### Prompts:

- Draw a feeling face to show how you felt during today's lesson.
- Draw something else you could have built in this Minecraft world.
- Can you draw a scene where someone is using a conditional?

## Extended Learning

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

### More Minecraft

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



This work 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: Until Loops in Maze

60 minutes

## Overview

In this **skill-building** 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.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction

### Main Activity (30 minutes)

Until Loops in Maze

### Wrap Up (15 minutes)

Reflection

## 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 Reflection 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 (15 minutes)

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 minutes)

### Until Loops in Maze

Bringing together concepts is not easy, but this set of levels 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.



**Skill Building**



**Video: Repeat Until Statements**



**Prediction**



**Skill Building**

4

5

6

7

8



**Challenge**



**Practice**



**Prediction**



## Wrap Up (15 minutes)

### Reflection

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.

#### 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?



This work 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 19: Harvesting with Conditionals

50 minutes

## 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.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (5 minutes)

Introduction

### Main Activity (30 minutes)

Harvesting with Conditionals

### Wrap Up (15 minutes)

Reflection

## Teaching Guide

### Warm Up (5 minutes)

#### Introduction

Students shouldn't need as much of an introduction to concepts today because they have had practice

## 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 the lesson to find any potential problem areas for your class.
- Make sure every student has a Reflection 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.

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 minutes)

### Harvesting with Conditionals

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.



1

Video: Harvesting with Conditionals



2-5

Skill Building

2

3

4

5



6-7

Practice

6

7



8

Challenge



9-10

Practice

9

10



11

Prediction



12-13

Lesson Extras



## Wrap Up (15 minutes)

## Reflection

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.

### 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?



This work 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 20: Functions in Minecraft

60 minutes

## Overview

In this **skill-building** lesson, students will begin to understand how functions can be helpful!

## Purpose

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction

### Main Activity (30 minutes)

Functions in Minecraft

### Wrap Up (15 minutes)

Reflection

## Objectives

Students will be able to:

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

## Preparation

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

## Vocabulary

- **Function** - A piece of code that you can easily call over and over again.

## Teaching Guide

### Warm Up (15 minutes)

#### Introduction

Help the class understand that functions are simply a chunk of code that has a name. Once defined, you

can use that name over and over in your program to tell the computer to run the chunk of code that you assigned to it.

Pull up a puzzle from the lesson. We recommend puzzle 9. As a class, work through the puzzle without using functions. Once you have gotten the solution, display it on a white board or overhead.

**Ask:** *Why can't you just use a loop?*

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

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

#### Teaching Tip

Function blocks:



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

## Main Activity (30 minutes)

### Functions in Minecraft

1

Video: Minecraft - The Agent

#### Teaching Tip

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

2-3

Skill Building

2

3

4

Video: Minecraft - Repeat Loops

 5-9

Skill Building

5

6

7

8

9

 10

Video: Minecraft - Functions

 11-14

Skill Building

11

12

13

14

 15

Video: Minecraft - Congratulations

 16

Free Play

## Wrap Up (15 minutes)

### Reflection

#### Prompts:

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



This work 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 21: Functions with Harvester

55 minutes

## Overview

In this **skill-building** lesson, students will use conditionals with functions to harvest crops in Harvester.

## Purpose

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (10 minutes)

Introduction

### Main Activity (30 minutes)

Functions with Harvester

### Wrap Up (15 minutes)

Reflection

## Objectives

Students will be able to:

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

## Preparation

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

## Vocabulary

- **Function** - A piece of code that you can easily call over and over again.

## Teaching Guide

### Warm Up (10 minutes)

#### Introduction

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

Ask the class:

- *When would you use a function?*



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

## Main Activity (30 minutes)


### Functions with Harvester

#### Online Puzzles

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

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

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

 1-6

Practice

1

2

3


4

5

6

 7

Video: How to Create a Simple Function

 8-10

Practice

8

9

10

 11

Challenge

 12

Practice

 13

Prediction

## Wrap Up (15 minutes)

### Reflection

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.

**Prompts:**

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



This work 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 22: Functions with Artist

60 minutes

## Overview

In this **skill-building** lesson, students will use functions with the Artist.

## Purpose

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

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction

### Main Activity (30 minutes)

Functions with Artist

### Wrap Up (15 minutes)

Reflection

### Extended Learning

## Objectives

Students will be able to:

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

## Preparation

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

## Vocabulary

- **Function** - A piece of code that you can easily call over and over again.

## Teaching Guide

### Warm Up (15 minutes)

#### Introduction

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

1. *The Declaration:* Function declarations are what create a function. In a function declaration, you fill in the function with code and you give the function a name. You must declare a function before you can use it.
2. *The Call:* Function calls are what makes the program run the code in the function. To call a function, you place the name of the function in your program. Make sure your function is properly defined before calling it in your program.

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

## Main Activity (30 minutes)

### Functions with Artist



Prediction



Practice



Challenge



Practice



Prediction



Free Play



Lesson Extras



## Wrap Up (15 minutes)

### Reflection

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.

**Prompts:**

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

## Extended Learning

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

### Draw by Functions

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

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



This work 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 23: Text and Prompts

45 minutes

## Overview

In this **skill-building** lesson, students will get practice with variables in Sprite Lab.

## Purpose

Variables will be used in this course to store and modify data. At this point, students will simply be storing and retrieving values without changing them. In later lessons, students will store numerical values and modify them over time to keep track of things like a player's score in a game.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (5 minutes)

Review

### Main Activity (30 minutes)

Text and Prompts

### Wrap Up (10 minutes)

Review

Reflection

## Objectives

Students will be able to:

- **Actions** Use variables to hold words and phrases.
- Use variables in conjunction with prompts.

## Preparation

Play through the levels and review the lesson slides.

## Links

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

For the teachers

- **Sprite Lab Documentation** - Resource
- **Text and Prompts** - Slides

## Vocabulary

- **Variable** - A label for a piece of information used in a program.
- **prompt** - A message on the computer screen that waits for input from the user.

## Teaching Guide

### Warm Up (5 minutes)

Review

**Discuss:** How do computer programs ask us for information?

**Discussion Goal:** Students should think about their own experiences as users and times when a computer asks them for information. There are lots of ways to input information into a computer, but focus on ideas where something is typed into a prompt for now.

## *Remarks*

When a message on the computer screen is waiting for your input we call that a prompt. When a user types into a prompt, it's like the computer is storing the information in a container. The variable's label tells us what kind of information to expect. Today we are going to learn the code to create a prompt in Sprite Lab.

## Main Activity (30 minutes)

### Text and Prompts

#### Prediction (5 mins)

**Do This:** Have students read the provided code and make a prediction about what will happen. After pressing "Run", provide time for students to reflect on anything they found interesting or that surprised them.



**Prediction**

#### Video (5 minutes)

Display the video to the whole class.



**Video: Text and Prompts**

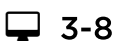
#### Teaching Tip

Just like when students are writing or communicating verbally, it's important to set expectations on using words appropriately in this activity. Students should understand that they are responsible for the code that they write, including any text that shows on the screen. The text should be respectful, as well as safe. Students can practice not sharing personal information in their programs, especially if they want to share with others. Note that any information typed into a Sprite Lab prompt isn't saved long-term. Words and messages typed by the users of these apps is gone once the app is reset.

#### Skill Building and Practice (20 mins)

**Transition:** Have students move to their computer and sign in. These activities are suitable for independent learning or pair programming.

Students should complete the Skill Building levels and then spend any remaining time choosing from the various Practice activities.



**Skill Building**

💡 Teaching Tip ▲

This lesson has more skill building and practice levels than previous lessons, because students are learning a few new skills all at once. As such, there is not a dedicated “free play” level. However, students who are feeling confident with their new skills can choose the last practice option to start from scratch with a blank project. In the next lesson, students will also be working exclusively on their own project that they plan and design ahead of time.



9-15

Practice

## Wrap Up (10 minutes)

### Review

**Do This:** Use the lesson slides to review the vocabulary for this lesson. Be sure that students can recall from the lesson where they saw each of these concepts.

### 🎤 Remarks

When a user enters information into a prompt, the computer stores it with a variable. The prompt can also trigger an event. If the code uses a variable, the computer will look for a matching label to find the stored information.

### Reflection

Pick one (or both) of the reflection prompts below to give to students. They can journal individually, or discuss the answers in groups or as a class.

**Prompts:**

- How is a variable like a box? How is it different?
- Why do programmers need variables?



This work 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 24: Counting with Variables

45 minutes

## Overview

In this **skill-building** lesson, students will use variables to track a value that changes over time, like a counter. This lesson also includes a short **mini-project** in which students create a simple game.

## Purpose

In previous lessons, students learn to use variables to store data that can be retrieved later. In this lesson, students will begin learning to modify the data stored in a variable. This key understanding of how variables work will also enable them to create more advanced projects.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (5 minutes)

Introduction  
Review

### Main Activity (30 minutes)

Counting with Variables

### Wrap Up (10 minutes)

Reflect

## Objectives

Students will be able to:

- Create a clicker game in Sprite Lab where sprites can be removed to score points
- Create a variable that stores information and changes over time

## Preparation

Play through the levels and review the lesson plan.

## Teaching Guide

### Warm Up (5 minutes)

#### Introduction

**Discuss:** Which of the following types of information change over time?

- Your first name
- Your birthday

- *Your height*
- *Your age*
- *The temperature outside*
- *The answer to 2+2*
- *The capital of our country*
- *Today's date*
- *The score in a game*

**Do This:** Have students discuss which information would change over time and which would not. Some items might generate a larger discussion. For example, it is possible to change your first name or use a nickname. Also, over time, your height will stop changing.

**Reflect:** *When is it useful to change what is stored in a variable?*

**Discussion goal:** Students should understand that they can create a variable that stores information that either stays the same or changes over time. In a game with points, a variable is responsible for keeping track of a user's score throughout the game. Typically, this variable starts out being set to 0 and increases by 1 for every point that the user scores. The final score is reported at the end by printing out the variable.

#### 💡 Teaching Tip

The goal of this discussion activity is to get students thinking about storing and modifying information using a variable. Students may recall that in previous lessons they would store information, like a word, by entering it into a prompt. In the next activities, students will learn a new way to store information as well as how to modify that information as a program runs. They should discuss when it would be necessary to change or update a variable over time.

For the discussion, decide beforehand what strategy will work best for your classroom. Some ideas include: using a T-chart, discussing in pairs or as a class.

#### 🎤 Remarks

The value stored in a variable can change over time.

When we start, our count variable is set to 0. After a few seconds, it increases by 1.

## Review

**Do this:** Use the lesson slides to review the vocabulary for this lesson. Be sure that students can recall from the lesson where they saw each of these concepts.

#### 🎤 Remarks

When a user enters information into a prompt, the computer stores it with a variable. The prompt can also trigger an event. If the code uses a variable, the computer will look for a matching label to find the stored information.

#### 💡 Teaching Tip

Students shouldn't be expected to know how variables will be used in this new context right away, so it's okay to frame these questions as things to consider while they work.

As students move through the levels in this lesson, they will learn how to set a variable to store a number, then change that number over time. Knowing how to modify a variable's value is the key to tracking something that changes, like the score in a game.

## Main Activity (30 minutes)

### Counting with Variables

#### Prediction (3 minutes)

**Do this:** Have students read the provided code and make a prediction about what will happen. After pressing “Run”, provide time for students to reflect on anything they found interesting or that surprised them.

**Do this:** Have students read the provided code and make a prediction about what will happen. After pressing “Run”, provide time for students to reflect on anything they found interesting or that surprised them.



1-2

Prediction

1

2

💡 Teaching Tip

This lesson includes two prediction puzzles: one to bridge what students have already learned and a second one to introduce the new blocks in the lesson.

#### Skill Building and Exploration (10 mins)

**Transition:** Move students to their computers. Encourage students to follow the instructions for each level.



3-5

Skill Building

3

4

5



6

Exploration: Clicker Game

#### Mini-Project (17 mins)



7-11

Mini-Project: Clicker Game

7

8

9

10

11

## Wrap Up (10 minutes)

## Reflect

**Reflect:** *How did you use variables in your game?*

**Discussion goal:** Students should understand that the variables in their game stored a value that changed over time. A variable's value can set directly in the code, modified as a user interacts with an app, or input directly such as with a prompt.



This work 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 25: Using Variables with the Artist

40 minutes

## Overview

In this **skill-building** lesson, students will explore the creation of repetitive designs using variables in the Artist environment. Students will learn how variables make code easier to write and easier to read. After guided puzzles, students will end in a free play level to show what they have learned and create new designs.

## Purpose

Variables are essentially placeholders for values that might be unknown at the time that you run your program or for values that can change during the execution of a program. These are vital to creating dynamic code because they allow your program to change and grow based on any number of potential modifications. This stage reinforces the use of variables, using the most basic capabilities of setting and using them.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

**Warm Up (5 minutes)**

Review

**Main Activity (20 minutes)**

Using Variables with Artist

**Wrap Up (15 minutes)**

Reflection

## Teaching Guide

### Warm Up (5 minutes)

## Objectives

Students will be able to:

- Assign values to existing variables.
- Use variables to change values inside of a loop.
- Utilize variables in place of repetitive values inside of a program.

## Preparation

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

## Links

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

For the students

- **Variables** - Student Video ([Download](#))

## Vocabulary

- **Variable** - A label for a piece of information used in a program.

## Review

It might be helpful to remind students of what they know about variables so far.

- How are variables like a box or container?
- Once information is stored in a variable, how do we use it later in a program?

## Main Activity (20 minutes)

### Using Variables with Artist

Consider reviewing the Prediction level and video with the full class before moving students on to the Skill Building section.



1

Prediction



2

Video: Variables in Artist



3-6

Skill Building

3

4

5

6



7

Free Play

## Wrap Up (15 minutes)

### Reflection

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.

#### Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- Have you tried mixing multiple variables into one program? What might that look like? When would it be helpful?



This work 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 26: Variables with the Bee

60 minutes

## Overview

This **skill-building** lesson will help illustrate how variables can make programs more dynamic by allowing values to change while the code is running.

## Purpose

This lesson will illustrate how code with changing values can be helpful and prepare students to understand how "for loops" work in upcoming lessons.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction

### Main Activity (30 minutes)

Variables with the Bee

### Wrap Up (15 minutes)

Reflection

## Objectives

Students will be able to:

- Examine code to find places where variables can be substituted for specific values.
- Identify areas where they can use variables to modify quantities during runtime.

## Preparation

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

## Vocabulary

- **Variable** - A label for a piece of information used in a program.

## Teaching Guide

### Warm Up (15 minutes)

#### Introduction

This series is a little different than what students have done in the past. Now, instead of simply assigning a value to a variable and running your code, you'll need to help students see how a variable can be modified during program runtime.

**Display** Show students the play area from one of the later puzzles.



There are several things to unpack here, so you might need to give your students a chance to look at it critically before you expect them to do anything with it.

**Think/Pair:** *What patterns do you notice in this puzzle? What are the differences between the flower honeycomb patches at the beginning, middle, and end of the bee's path?*

**Share:** Let students share their ideas.

**Discuss:** *What are the ways we could program a solution to this puzzle? How could we use a variable to have a loop do this for us?*

## Main Activity (30 minutes)

### Variables with the Bee

This set of puzzles takes some serious computational thinking skills. If you find that students are getting stuck, help them break down the puzzles into the individual pieces:

- What would it look like if the flowers/honeycomb all had the same amount of nectar/honey?
- Now how can you use a variable to get the quantities the way you want them?

1-7

Skill Building

1

2

3

4

5

6

7

8-10

Practice

8

9

10



## Wrap Up (15 minutes)

### Reflection

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.

#### Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- What are some ways you have used variables so far?
- What else do you think you can do with variables?



This work 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 27: For Loops with Bee

60 minutes

## Overview

This **skill-building** lesson focuses on for loops and using an incrementing variable to solve more complicated puzzles.

## Purpose

Today's concept, for loops are a very important topic in computer science. Not only are they widely used, the process of learning for loops enhances the learning of other important concepts (such as variables and parameters.) Students will have plenty of practice critically thinking through problems by determining the starting, ending, and stepping values for each for loop. This concept uses plenty of math as well, so feel free to pair it with a math lesson for an even deeper learning experience.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction

### Main Activity (30 minutes)

For Loops With Bee

### Wrap Up (15 minutes)

Reflection

## Teaching Guide

### Warm Up (15 minutes)

Introduction

## Objectives

Students will be able to:

- Determine starting value, stopping value, and stepping value for a for loop.
- Recognize when to use a for loop and when to use other loops such as repeat and while loops.

## Preparation

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

## Vocabulary

- **For Loop** - Loops that have a predetermined beginning, end, and increment (step interval).

Display a puzzle from the lesson. We recommend the prediction level because it displays a potential solution and asks the user to evaluate it.

## Main Activity (30 minutes)

### For Loops With Bee

#### Online Puzzles

Some students may have a hard time differentiating between repeat loops and for loops. We recommend having scratch paper out for students to make guesses on values like the start, stop, and step. Implementing pair programming amongst the class might also be helpful for your students.



1-2

Review

1

2



3

Exploration: For Loops



4

Video: For Loops



5

Prediction



6-8

Skill Building

6

7

8



9-13

Practice

9

10

11

12

13



14-15

Lesson Extras



## Wrap Up (15 minutes)

## Reflection

### Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- How is a for loop different from a repeat loop?
- Why do you think for loops could be useful?



This work 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 28: For Loops with Artist

60 minutes

## Overview

In this **skill-building** lesson, students practice “for” loops with Artist. Students will complete puzzles to create complex designs and unique art.

## Purpose

Creativity and critical thinking come together beautifully in this lesson. Students will continue their practice with for loops and variables while they create jaw-dropping images. This lesson inspires a creative mind while teaching core concepts to computer science.

## Standards

Full Course Alignment

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

- ▶ **AP** - Algorithms & Programming

## Agenda

### Warm Up (15 minutes)

Introduction

### Main Activity (30 minutes)

For Loops with Artist

### Wrap Up (15 minutes)

Reflection

## Objectives

Students will be able to:

- Recognize when to use a for loop and when to use other loops such as repeat and while loops.
- Use for loops to change loop several times with different values.

## Preparation

- Play through the lesson to find and potential problem areas for your class.
- Make sure every student has a Reflection Journal.

## Vocabulary

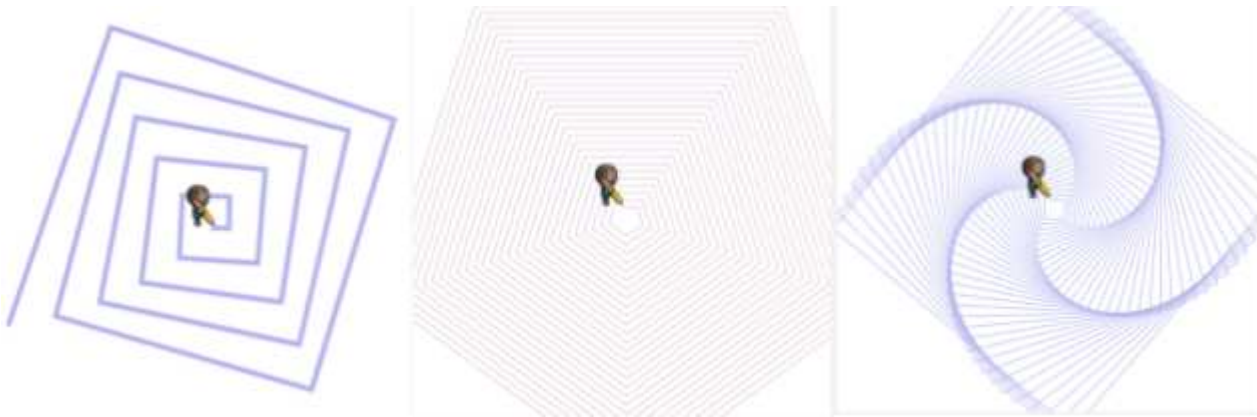
- **For Loop** - Loops that have a predetermined beginning, end, and increment (step interval).

## Teaching Guide

### Warm Up (15 minutes)

#### Introduction

On a board displayed to the entire class, draw (or display via projector) one of the final designs from the Code Studio puzzles associated with this lesson. We recommend one of the following:



Ask the class how a computer might draw the drawing you displayed.

After a few predictions have been said, reply *with for loops of course!*

Tell the students they will soon be learning how to create these fine drawings using for loops and variables.

## Main Activity (30 minutes)

### Flor Loops with Artist



1

Video: For Loops

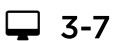
Teaching Tip

These puzzles are super fun, but it may be helpful for students to have protractors, pencils and scratch paper to see how these designs were made in the physical form. If that isn't an option in your class, try to get the students to trace on the computer screen with their fingers.



2

Exploration



3-7

Skill Building

3

4

5

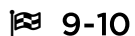
6

7



8

Free Play



9-10

Lesson Extras



## Wrap Up (15 minutes)

### Reflection

#### Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- Draw one of the designs you made today. What was the code needed to create it?
- What are some designs you would like to create? How do you think for loops or variables could help create those?



This work 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 29: End of Course Project

45 minutes

## Overview

This **project** lesson takes students through the process of designing, developing, and showcasing new 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 minutes)

Planning

### Main Activity (25 minutes)

Coding

Example Projects

Create your project

### Wrap Up (10 minutes)

Showcase

## Objectives

Students will be able to:

- Overcome obstacles such as time constraints or bugs.

## Preparation

- Spend time making your own project with both the project tools available to students. Familiarize yourself with the capabilities and limitations of each tool.
- Modify the **\*rubric** to fit your class goals and print one copy for each student or group.
- Modify the **\*project planning guide** to fit your class and print one for each student or group.

## Links

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

For the teachers

- **CS Fundamentals Final Project - Rubric**
- **Sprite Lab Documentation - Resource**

For the students

- **CSF Express End of Course Project Planning Guide - Project Guide**

## Vocabulary

- **Define** - Figure out the details of the problems that you are trying



to solve

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

## Teaching Guide

### Warm Up (10 minutes)

#### 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 **\*End of Course 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 minutes)

#### 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!

## Example Projects



1

Example Projects

## Create your project



2

Create your project

## Wrap Up (10 minutes)

### 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.



This work 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**.