

Unit 5 - Lists, Loops, and Traversals

Unit Overview

Students learn to build apps that use and process lists of information. Like the previous unit, students learn the core concepts of lists, loops, and traversals through a series of EIPM lesson sequences. Later in the unit, students are introduced to tools that allow them to import tables of real-world data to help further power the types of apps they can make. At the conclusion of the unit, students complete a week-long project in which they must design an app around a goal of their choosing that uses one of these data sets.

Course Master Vocabulary

Unit Philosophy and Pedagogy

- **Independent Creation and The Hackathon Project:**

Much like the project in Unit 4, the "Hackathon" project in this unit is designed as an opportunity for students to creatively and independently build something with their programming skills. While students are asked to include some technical requirements in their program to ensure they demonstrate mastery of new programming concepts, they have free rein to choose the goals, design, and implementation of their project. To avoid asking students to complete a major programming project right before the Create PT, this hackathon is the most "Create-PT-like" project of the course. It's the best chance for students to practice skills like budgeting time or scoping an open-ended project. In many classrooms, if you maintain the recommended pacing of the course, this project serves as an excellent end to the first semester.

- **Growing Comfort with EIPM:** By Unit 5, students (and teachers!) should be developing greater comfort with the flow of EIPM lessons. Students may begin to anticipate that sequences are building towards an independent Make lesson, or look forward to stepping away from computers to Explore. A nice feature of EIPM is that you'll find strategies and modifications to each lesson type that work best for your students. Keep an eye out for how you and your students are developing comfort with EIPM, and note strategies that help meet the needs in your classroom.
- **Programming with Real-world Data:** The Data Library is a new feature in App Lab for the 2020-21 school year and was designed to let students program with data from the real-world. The goal of this tool is to motivate students to build new kinds of data-powered apps that they find personally interesting. This tool also facilitates programming with lists of information, since students will need to manipulate lists of data in order to incorporate the different data sources. Encourage students to use datasets they find personally relevant as they draw on their creative ideas for bringing data to life.

Major Assessment and Projects

The unit project asks students to spend five days as part of a "Hackathon" project that they have nearly complete independence to scope and design. Students must choose one dataset from the Data Library in AppLab to be a component of their project to demonstrate what they have learned about lists and list processing; otherwise,

Teaching Tip

Modifications for Virtual and Socially-Distanced Classrooms



Are you teaching in a virtual setting or in a socially-distanced classroom? Check out **this document** for ideas and resources to help you tailor common practices like *Think Pair Share* or *Peer Feedback* to your learning environment.

For an overview of EIPM modifications, read through the **EIPM Modifications document** before reading the lesson-specific modifications within Lesson Plans.

Learn more about how to use these resources **here**.

scoping the project is completely up to them. Students submit their app, project guide, and written responses to reflection questions about how the app is designed and the development process they used to make it. Students will also complete an end-of-unit assessment aligned with CS Principles framework objectives covered in this unit.

AP Connections

This unit and unit project helps build towards the enduring understandings listed below. For a detailed mapping of units to Learning Objectives and EKs please see the "Standards" page for this unit.

- CRD-2: Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.
- AAP-1: To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.
- AAP-2: The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.
- AAP-3: Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.

This unit includes content from the following topics from the AP CS Principles Framework. For more detailed information on topic coverage in the course review **Code.org CSP Topic Coverage**.

- 3.2 Data Abstraction
- 3.4 Strings
- 3.8 Iteration
- 3.10 Lists
- 3.16 Simulations

The College Board has supplied formative Create PT questions to help prepare students to complete the Create Task. We recommend that students complete the following prompts with the unit project. More information can be found in Code.org CS Principles Topic Coverage.

- 3.b.i
- 3.b.ii
- 3.b.iii
- 3.b.iv
- 3.b.v

Week 1

Lesson 1: Lists Explore

Develop a mental model for how computers store and access lists of information.

Lesson 2: Lists Investigate

App Lab

Investigate and modify sample apps that use lists learn common programming patterns with lists.

Lesson 3: Lists Practice

App Lab

Practice programming with lists through a set of programming puzzles.

Lesson 4: Lists Make

App Lab

Make an app that uses lists and programming patterns with lists.

Lesson 5: Loops Explore

Develop a mental model for how computers repeat instructions over and over using loops.

Week 2

Lesson 6: Loops Investigate

App Lab

Investigate and modify sample apps that use loops and learn common programming patterns with loops.

Lesson 7: Loops Practice

App Lab

Practice programming with loops through a set of programming puzzles.

Lesson 8: Loops Make

App Lab

Practice making an app that uses loops and programming patterns with loops.

Lesson 9: Traversals Explore

App Lab

Develop a mental model for how computers use loops to traverse and process lists of information.

Lesson 10: Traversals Investigate

App Lab

Investigate and modify sample apps that use traversals.

Week 3

Lesson 11: Traversals Practice

App Lab

Practice programming with list traversals through a set of programming puzzles.

Lesson 12: Traversals Make

App Lab

Practice making an app that processes a list from a data set using traversal.

Lesson 13: Project - Hackathon Part 1

Project | App Lab

This is the first day of a five-day unit project. Students begin the project by choosing a partner, determining a dataset to design the app around, and creating a paper prototype.

Lesson 14: Project - Hackathon Part 2

Project | App Lab

This is the second day of a five-day unit project. Students continue to plan for the project by filling out tables of information on element IDs and programming constructs before each tackling a different role in the project as a designer or a programmer.

Lesson 15: Project - Hackathon Part 3

Project | App Lab

This is the third day of a five-day unit project. Students continue to build their apps.

Week 4

Lesson 16: Project - Hackathon Part 4

Project | App Lab

This is the fourth day of a five-day unit project. Students continue to build their apps.

Lesson 17: Project - Hackathon Part 5

Project | App Lab

This is the final day of a five-day unit project. Students complete a Written Response, individually answering prompts about the project. Students then share their apps during a gallery walk.

Lesson 18: Assessment Day

Project

Assessment day to conclude the unit.



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

English ▼

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

Lesson 1: Lists Explore

Overview

Students will learn the ways that lists are created, accessed, and changed through a teacher-guided activity using plastic baggies and pieces of paper. The lesson begins with a brief reflection on the value of lists. Students then complete the main activity which introduces the syntax to use lists and the ways they can be used. To wrap up students watch two short videos on lists and record the main concepts in their journals.

Purpose

In the warmup, students brainstorm different lists of information that they encounter on a daily basis. Then in the activity, students return to baggies and sticky notes from the previous unit to build a concrete model of a list before seeing how lists are programmed in Javascript. Students are exposed to different ways of interacting with a list like accessing, removing, appending, and inserting elements. The wrap up concludes the lesson with a summary video and a journal vocabulary exercise.

Agenda

Lesson Modifications

Warm Up (5 mins)

Preview Lists

Activity (30 mins)

Lists

Wrap Up (10 mins)

Video and Vocabulary

Assessment: Check For Understanding

View on Code Studio

Objectives

Students will be able to:

- Use appropriate vocabulary to describe lists.
- Use an index to reference specific elements in a list

Preparation

- ☐ 7+ sandwich baggies per pair of students
- ☐ 1 gallon-sized baggy per pair of students
- ☐ packs of red and orange stickies
- ☐ pens / pencils
- ☐ 1 dry erase marker per four students (pairs can share)
- ☐ Review the slides and click through all animations

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals - Presentation**

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (5 mins)

Preview Lists

Prompt: With a partner, brainstorm lists of information that you encounter on a daily basis. Why are lists useful?

Discuss: Have students discuss with a partner and then have a few students share out their responses.

Discussion Goal

Discussion Goal:

- Lists help us group together like information
- We can go through a list item by item to check off things we have completed
- Lists help us stay organized

Activity (30 mins)

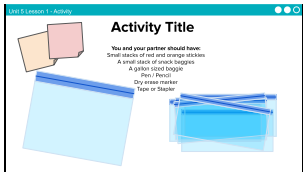
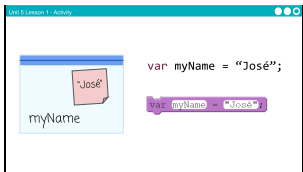
Lists

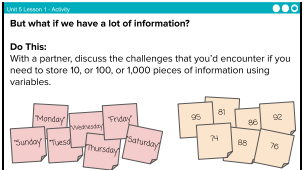
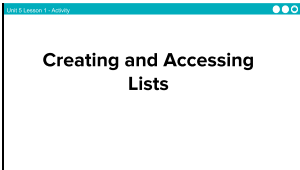
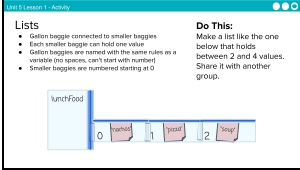
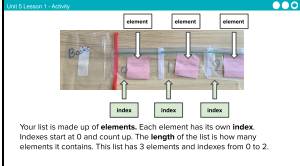
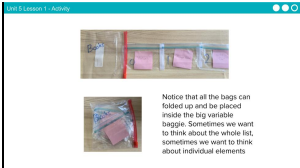
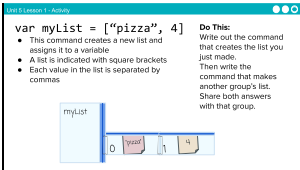
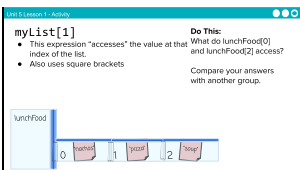
Group: Group students in pairs.

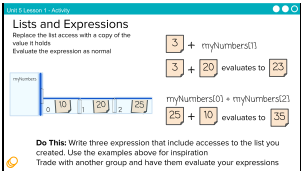


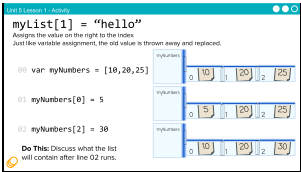

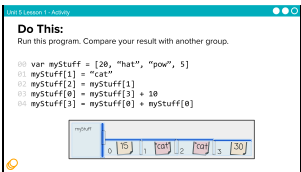

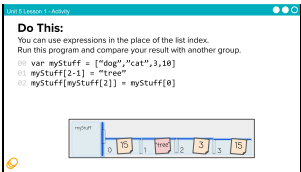

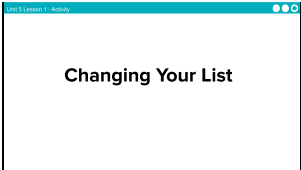
Distribute: Give each pair of students:

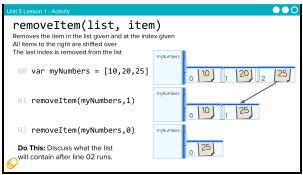

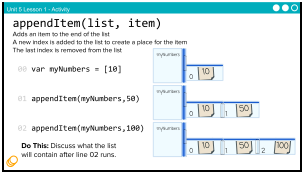

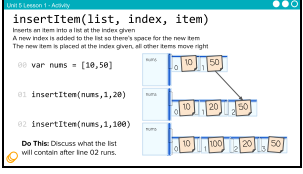

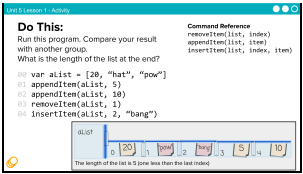

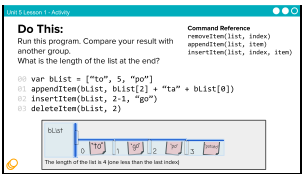

- A small stack of red and orange sticky notes
- A pen/pencil
- 7+ snack-sized plastic baggies
- One gallon-sized plastic baggy
- A dry erase marker to share with another group

Display: Use the activity slides for this lesson to guide the unplugged activity on Conditionals.

Slides	Speaker Notes
	<p>Say: Today, we are going to explore lists.</p> <p>Note: Supplies Substitutions</p> <ul style="list-style-type: none">• There's no need to use stickie notes if you have other scraps of colored paper. Also consider cutting stickies in 4 to make them go further. If you don't have dry erase markers handy consider using pieces of masking tape on the baggies.• The gallon-sized baggie can be replaced with any baggie that is large enough to hold the snack-sized baggies.
	<p>Say: We've already learned that variables can help us store one piece of information.</p>

Slides	Speaker Notes
	<p>Say: But what if we have lots of information?</p> <p>Do This: With a partner, discuss the challenges that you'd encounter if you need to store 10, or 100, or 1,000 pieces of information using variables.</p> <p>Discuss: Lead a short discussion on the challenges they'd encounter. The main challenges are that you'd need to make a variable for each piece of information and it could get difficult to name them all.</p>
	<p>Say: Let's look at how to use lists</p>
	<p>Say: Today we're going to practice using a new way to store information called a list. Follow the instructions here to make a list of your own.</p> <p>Do This: Have students create their own list. Circulate the room to check on how they're doing. Make sure they share with another group.</p>
	<p>Say: Your list is made up of elements. Each element has its own index. Indexes are just numbers that count up from zero. The length of the list is how many elements it contains. This list has 3 elements and indexes from 0 to 2.</p>
	<p>Say: Notice that all the bags can be folded up and be placed inside the big variable baggie. Sometimes we want to think about the whole list, sometimes we want to think about individual elements</p>
	<p>Say: We create a list with this command. It creates a new list and assigns it to a variable. The square brackets indicate we're making a list. Each value is separated by commas.</p> <p>Do This: Have students write the command to make their own list and the list of partner group. By sharing their answers with another group they can check that they're following along.</p>
	<p>Say: If we want to access the values in our list we use the square brackets next to the name of our list.</p> <p>Do This: Have students discuss with other groups. They should see the answers are "nachos" and "soup"</p>

Slides	Speaker Notes
 <p>Lists and Expressions Replace the list access with a copy of the value it holds. Evaluate the expression as normal.</p> <p>Do This: Write three expressions that include accesses to the list you created. Use the examples above for inspiration. Trade with another group and have them evaluate your expressions.</p>	<p>Say: We can use list accesses inside of expressions, just like variables.</p> <p> Click for animation</p> <p>Say: This first expression evaluates to 23 because myNumbers[1] contains 20.</p> <p> Click for animation</p> <p>Say: This second expression evaluates to 35 because myNumbers[0] and myNumbers[2] contain 25 and 10.</p> <p>Do This: Have students write three expressions using their own lists. Then have them trade with another group to practice evaluating. Circulate the room to make sure students are following the directions correctly.</p>
 <p>myList[1] = "hello" Assigns the value on the right to the index. Just like variable assignment, the old value is thrown away and replaced.</p> <pre> 01 var myNumbers = [10, 20, 25] 02 03 myNumbers[0] = 5 04 05 myNumbers[2] = 30 </pre> <p>Do This: Discuss what the list will contain after line 02 runs.</p>	<p>Say: We can assign the index of a list just like a variable.</p> <p>Do This: Have students discuss what the list will contain after line 02 runs.</p> <p> Click for animation</p> <p>Say: This code will assign a new value at index 2 of the list.</p>
 <p>Do This: Run this program. Compare your result with another group.</p> <pre> 01 var myStuff = [20, "hat", "paw", 5] 02 myStuff[1] = "cat" 03 myStuff[2] = myStuff[1] 04 myStuff[0] = myStuff[3] + 10 05 myStuff[5] = myStuff[0] + myStuff[0] </pre>	<p>Do This: Circulate the room while students run this program</p> <p> Click for animation</p> <p>Discuss: Discuss quickly with the class why the program ends with the list shown.</p>
 <p>Do This: You can use expressions in the place of the list index. Run this program and compare your result with another group.</p> <pre> 01 var myStuff = ["dog", "cat", 3, 10] 02 myStuff[2-1] = "tree" 03 myStuff[myStuff[2]] = myStuff[0] </pre>	<p>Do This: Circulate the room while students run this program</p> <p> Click for animation</p> <p>Discuss: Discuss quickly with the class why the program ends with the list shown.</p> <p>Note: Line 02 is tricky. You need to first evaluate myStuff[0] which is "dog" to determine the value being assigned. Then evaluate myStuff[2], which is 3. At this point, the command reads myStuff[3] = "dog". Now it is just a simple assignment.</p>
 <p>Changing Your List</p>	<p>Say: Now let's learn about three different commands that can change your list.</p>

Slides	Speaker Notes
 <p>removeItem(list, item) Removes the item in the list given and all the items given. All items to the right are shifted over. The last index is removed from the list.</p> <pre> var myNumbers = [10, 20, 25]; removeItem(myNumbers, 1); removeItem(myNumbers, 0); </pre> <p>Do This: Discuss what the list will contain after line 02 runs.</p>	<p>Say: The removeItem command will remove an item from a list. The item at the index given is removed, items to the right are shifted over, and the last index is removed.</p> <p>Do This: Have students discuss what the list will show after this command.</p> <p> Click for animation</p> <p>Discuss: Discuss with the class any questions.</p>
 <p>appendItem(list, item) Adds an item to the end of the list. A new index is added to the list to create a place for the item. The last index is removed from the list.</p> <pre> var myNumbers = [10]; appendItem(myNumbers, 50); appendItem(myNumbers, 100); </pre> <p>Do This: Discuss what the list will contain after line 02 runs.</p>	<p>Say: The appendItem command will add an item to a list. A new index is added to the end of the list and the new item is place in it.</p> <p>Do This: Have students discuss what the list will show after this command.</p> <p> Click for animation</p> <p>Discuss: Discuss with the class any questions.</p>
 <p>insertItem(list, index, item) Inserts an item into a list at the index given. A new index is added to the list so there's space for the new item. The new item is placed at the index given, all other items move right.</p> <pre> var nums = [10, 50]; insertItem(nums, 1, 20); insertItem(nums, 1, 100); </pre> <p>Do This: Discuss what the list will contain after line 02 runs.</p>	<p>Say: The insertItem command will add an item to the middle of a list. The item is placed in the index given. Then every item is moved over one space to make room. A new index is added to the end of the list to make room.</p> <p>Do This: Have students discuss what the list will show after this command.</p> <p> Click for animation</p> <p>Discuss: Discuss with the class any questions.</p>
 <p>Do This: Run this program. Compare your result with another group. What is the length of the list at the end?</p> <pre> var alist = [20, "hat", "pow"]; appendItem(alist, 5); appendItem(alist, 10); removeItem(alist, 1); insertItem(alist, 2, "bang"); </pre> <p>Command Reference removeItem(list, index) appendItem(list, item) insertItem(list, index, item)</p> <p>The length of the list is 4 (one less than the last index)</p>	<p>Do This: Have students run this program. Circulate the room to support them with any questions.</p> <p> Click for animation</p> <p>Discuss: Discuss with the class any questions.</p>
 <p>Do This: Run this program. Compare your result with another group. What is the length of the list at the end?</p> <pre> var blist = ["a", 0, "po"]; appendItem(blist, blist[2] + "ta" + blist[0]); insertItem(blist, 2-1, "go"); deleteItem(blist, 2); </pre> <p>Command Reference removeItem(list, index) appendItem(list, item) insertItem(list, index, item)</p> <p>The length of the list is 4 (one less than the last index)</p>	<p>Do This: Have students run this program. Circulate the room to support them with any questions.</p> <p> Click for animation</p> <p>Discuss: Discuss with the class any questions.</p>

Wrap Up (10 mins)

Video and Vocabulary

📺 **Video:** As a class watch both videos on lists.

📋 **Do This:** Review Key Takeaways and have students add to their journals: lists, element, and index.

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: Match the index with the value. `var myNumbers = [32, 64, 33, 0, 15, 26, 3]`

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

CSP2021

▶ **AAP-1** - To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways

▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result



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

English ▼

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

Lesson 2: Lists Investigate

Overview

In this lesson students work with partners to investigate three different apps that use lists. Students first explore all three apps without seeing the code to notice similarities and predict how they will work. Then they explore the code itself and make additions and modifications to the apps. To conclude the lesson, students review and discuss common programming patterns with conditionals.

Purpose

After building a conceptual model for list and list operations in the previous lesson, this lesson allows students to see how they are actually implemented in code. This lesson also introduces common programming patterns when using lists. Students will have some opportunities to modify working code in this lesson, but the most significant practice with lists will come in the following lesson.

Agenda

Lesson Modifications

Warm Up (5 mins)

Activity (35 mins)

Accessing Lists - 20 mins

Changing Lists

Modify Apps

Wrap Up (5 mins)

Patterns Review

Synthesis

View on Code Studio

Objectives

Students will be able to:

- Identify common programming patterns using lists
- Explain the purpose of programming patterns with lists both in terms of how they work and what they accomplish
- Modify apps that make use of common programming patterns with lists to adjust their functionality

Preparation

- Review the three apps that students will be investigating and the questions about them. Note that there are target responses to each of these questions on the levels.

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals - Presentation**


Teaching Guide

Lesson Modifications




Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (5 mins)


 **Prompt:** What are some similarities and differences between variables and lists? How does a list manage complexity in a program?


Activity (35 mins)

Accessing Lists - 20 mins

 **Level 2 - Video:** Show the video explaining how to determine the length of a list.

Group: Place students in pairs.

 **Levels 3 - 4: Band Namer and Outfit Picker** Assign half of the pairs to investigate the Band Namer app and half to investigate the Outfit Picker. Each pair should read the app carefully and prepare answers to the questions for their app.

 **Discuss:** Have pairs match up with another pair that investigated another app, forming a group of four. Each pair should spend ~5 minutes walking them through the way that the app works and what they learned specifically from answering each question. Afterwards discuss any open questions that they couldn't answer with the room.

Discussion Goal

Discussion Goal: Students may bring up the following points:


- Both are used to store information
- Lists store multiple items, but variables only store one
- When written in Javascript, both start with the keyword var
- Lists are written with square brackets around them
- Lists can store many different types of data

Managing complexity:

- You don't need to know how a list is created in order to use it - all you need is the name. There is a separation between the abstract properties of the data type (list) and the concrete details of its representation.
- Programs that use lists are easier to read and manage - separate variables are not needed for each individual element

Changing Lists

 **Level 5 - Video:** Show the video explaining how to determine the length of a list.

 **Levels 6 - Pair Maker:** Have all pairs investigate this app on their own, answering the questions listed there.

Discuss: Have pairs match up with another pair forming a group of four. The group should spend ~5 minutes discussing their responses to each of the questions. Bubble up confusion points or open questions to the room.

Modify Apps

If there is time remaining, have pairs return to one or more of the apps and make the suggested modifications.

Wrap Up (5 mins)


Patterns Review

Levels 7-8: Review the patterns in these levels as a class.

- Have students add any relevant notes about the patterns to their journals.

- Discuss which of the three apps you think were using which pattern.

Synthesis

 **Prompt:** What aspects of using lists do you feel you already understand? What questions do you want to dig into more tomorrow during the practice lesson?

Discuss: Have students write in their journals, then share with a neighbor, then finally discuss as a class.

Teaching Tip

Finding Target Responses: With a verified teacher account you should be able to see target responses to each question posed on each level.

Prepping for Investigate Lessons: The best way to prepare for this lesson is to go through the experience yourself. Check out the three apps in Code Studio to get a sense for how they work. Then watch the videos. Then move on to the Code Investigation and actually try to answer all the questions for each app. To help you out, however, answers are provided on the bottom of the instructions area for verified teachers.

Show Videos at the Front: Show videos at the front of the room rather than having students watch them individually. It will be a good opportunity to bring the room back together.

Display Code at the Front: If your room allows it, display the code during the Code Investigation at the front of the room. When students mentions specific lines of code actually scroll to that line and read through it together.

Save Modifications for the End: This lesson can be tight on time. Rather than have students modify the code all at once, you can save modifications for the end of the Code Investigation and have students pick a single app they wish to modify.

Discussion Goal

Goal: Use this discussion to identify any points of confusion that will need to be reviewed in coming lessons.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-1** - To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways



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

English ▼

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

Lesson 3: Lists Practice

Overview

Practice the basics of lists including creating lists and accessing, inserting, and removing elements from lists.

Purpose

This lesson is students primary opportunity to get hands on with lists in code prior to the Make activity in the following lesson. Give students as much class time as you can to work through these. For this lesson it's recommended that you place students in pairs as a support and to encourage discussion about the challenges or concepts they're seeing.

Agenda

Lesson Modifications

Warm Up (5 mins)

Activity (35 mins)

Practice Time

Wrap Up (5 mins)

Discussion

Assessment: Check For Understanding: AP Practice

View on Code Studio

Objectives

Students will be able to:

- Correctly set up a list in a program
- Debug programs with lists
- Accurately use list operations including accessing, inserting, and removing elements

Preparation

- ☐ Review the slide on string indexes which you'll cover at the beginning of the lesson
- ☐ Review the programming progression to understand what students will be asked to do.

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals - Presentation**

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (5 mins)

Do This: Run a quick vocabulary review with students. Click through the animation to see the vocabulary definitions.

Remarks

Today we're going to have a chance to practice programming with a lot of the concepts and patterns we've explored over the last two lessons. I encourage you to go through these with a partner, but pay close attention to what each other is doing. In our next lesson you're going to have to use a lot of these on an independent project, and these activities are good practice for what you'll find there! Alright, let's get to it!

Activity (35 mins)

Practice Time

Remarks

Display: Review the slides on the debugging process and remind students of important skills including the following:

- Try to zoom in on precisely where you're getting stuck.
- Talk to your partner! That's what they're there for!
- Hover over blocks to read the documentation about how they work.
- Read the resources in the Help & Tips tab
- Talk to the group next to you. If another group asks for help make sure to take some time to talk it through with them.

Display: Highlight important debugging skills specifically for lists including ways to use the Watch panel and the Debug Console with lists.

Display: Quickly introduce the fact that strings have indexes too. Students will see `string.length` and `string.substring()` during the lesson so this is just calling out a concept that students will need to recognize when they get to it.

Levels 2-3 Setting Up Lists: These levels involve setting up a list and printing the list to the console

- Level 2: a list of numbers
- Level 3: a list of strings

Levels 4-6 Accessing Elements in a List: Students practice accessing elements in a list using the index. These levels also practice the random list access and list scrolling pattern students saw in the Investigate lesson.

- Levels 4: Students practice printing specific elements to the console using bracket notation, as in `myList[1]`.
- Level 5: Students build a "Magic 8 Ball" app that uses the Random List access pattern.
- Level 6: Students build a "Class Schedule" app that uses the List Scrolling pattern.

Levels 7 Strings and Indexes: Students practice accessing elements in a list


- Levels 7: Students practice printing the length and first characters of strings
- Level 8: Students debug code that uses `string.substring()` to grab different parts of a date stored as a string.

Levels 9-12 List Operations: In these levels, students work with list operations: `appendItem()`, `removeItem()`, and `insertItem()`

- Level 9: Students practice using `appendItem` to increase the size of a list
- Level 10: Students modify a "Food Diary" app so that the user can add new elements to a list
- Level 11: Students practice using `removeItem` and `insertItem` to modify lists
- Level 12: Students modify a "Top Ten Songs" app so that the user can add new items into the middle of a list while keeping the total length of the list to 10.

Wrap Up (5 mins)

Discussion

 **Goal:** Use this opportunity to address any lingering questions or misconceptions in the room. You can also use this as a source of discussion topics to kick off the following lesson. As you lead the discussion, call out the many resources students have access to help when they're getting stuck.

Prompt: What aspects of working with lists do you feel like clicked today? What do you still feel like you have trouble with?

Discuss: Have students share with one another before sharing with the whole class.

Remarks

Working with lists can be tricky - especially working with the index. We will get more practice tomorrow by making an app that uses lists to store information.

Assessment: Check For Understanding: AP Practice

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: What will be displayed after this code segment is run?

```
gems ← ["ruby", "sapphire", "garnet"]
gems ← ["opal", "emerald"]
DISPLAY gems[2]
```

Question: What will be displayed after this code segment is run?

Teaching Tip

Providing Support: Circulate around the room through the lesson encouraging students to use the strategies introduced at the beginning of the lesson. Students have a number of supports at their fingertips, so a big part of your role is helping build their independence in using those resources.

Common Errors: The following are common errors students may encounter in this lesson:

- The last index in a list is at `list.length-1` because the first index is 0. One of the most common errors with lists, therefore, is trying to access indexes that are out of bounds, most notably `list.length`,
- Syntax errors are common with lists, as the bracket notation is new. Encourage students to use the blocks to help if they need.
- When using the patterns students may become overly reliant on simply copying and pasting without thinking through what the code from the patterns does. They may reference variables that don't exist in their program, for example, because it is in the example code of the pattern.

Teaching Tip

Extension Opportunities:

- Level 6: Ask students to add the ability to add new classes to the list in the Class Schedule app.
- Level 10: Ask students to add the ability to remove items in the Food Diary app
- Level 12: Ask students to add a button that will "Undo" adding the most recent song to the list. This is pretty tricky and requires keeping track of some new information.

```
luckyNumbers ← 15, 33, 25
INSERT luckyNumbers, 2, 13
APPEND luckyNumbers, 3
REMOVE luckyNumbers, 1
DISPLAY LENGTH luckyNumbers
```

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-1** - To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways
- ▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result



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

English ▼

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

Lesson 4: Lists Make

Overview

Using Programming Patterns and a step-by-step approach students make their own version of a Reminder app. At the beginning of the lesson students are able to explore a working version of the app. They are then given the design elements of the app but begin with a blank screen. Students use an Activity Guide to go through the high level steps they should use to develop their app but leaves it to them to decide how to write the code. At the end students submit their apps which can be assessed using a provided rubric.

Purpose

This lesson is an opportunity for students to take on the "blank screen" and build the code that runs an app entirely from scratch. Guidance provided throughout the lesson helps students break down the large task of "building an app" into more incremental steps that they can use on future projects, including this unit's final project and the Create PT.

Agenda

Lesson Modifications

Warm Up (5 mins)

Activity (35 mins)

Build the Reminder App

Wrap Up (5 mins)

Assessment: Make Project

View on Code Studio

Objectives

Students will be able to:

- Recognize the need for programming patterns with lists as part of developing a functioning app
- Implement programming patterns with lists to develop a functioning app
- Write comments to clearly explain both the purpose and function of different segments of code within an app
- Use debugging skills as part of developing an app

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals** - Presentation

For the Students

- **CSP Lists Make - Reminder App** - Activity Guide [Make a Copy](#)

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (5 mins)



Prompt: Imagine you want to build an app for reminders. What information do you think would be stored in a list?

Remarks

For the past few days, we've learned a lot about using lists to help apps keep track of information. In today's Make Project you'll be practicing using lists and programming patterns with lists to create a functioning Reminders app.

Discussion Goal

Discussion Goal: Students may struggle with this question. It's ok if they don't get it right away. The goal is to prime their thinking before they look at the sample app.

Sample Answer: The reminders could be stored in a list.

Activity (35 mins)

Build the Reminder App

Group: Make a determination as to whether this project will be completed in pairs or individually. You may even choose to let students decide.

Do This: Have students explore the working Reminder App in Level 2.



Prompt: If students are not working in pairs they should still discuss the prompts with a neighbor.

- How many lists do you think are needed to make this app work?
- What programming patterns with lists do you think you'll need to use?



Remarks

Now let's build this app. The screen has been set up for you - it's your job to add the code!

Do This: Direct students to level three where they complete the Reminder App. An optional Activity Guide is provided if students would like guidance in creating the app. The most relevant programming pattern is displayed on a slide. Review this pattern quickly with students, if needed.

Submit: Encourage students to check the rubric on the last page of the Activity Guide before submitting.

Discussion Goal

- How many lists do you think are needed to make this app work?
 - Only one list is needed. It stores all of the reminders.
- What programming patterns with lists do you think you'll need to use?
 - This app uses the List Scrolling pattern.

Wrap Up (5 mins)

Remarks

💡 📄 Awesome work today! Make sure to submit your project when you're done with it!

Assessment: Make Project

Use the rubric provided with the project to assess student projects.

💡 Teaching Tip

Supporting Students: While students are working on their apps, circulate the room and check in with students who need a little help. Encourage students to collaborate and discuss bugs with each other.

Debugging: Review with students steps they can use to debug if they get stuck:

- Run the code on turtle mode
- Add the variables to the watcher
- Explain the code to a friend

💡 Teaching Tip

Maximize Work Time: The wrap up is short to allow the maximum amount of time for students to complete the activity.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

► **AP** - Algorithms & Programming



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

English ▼

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

Lesson 5: Loops Explore

Overview

Students begin the lesson by discussing the purpose of loops before completing the unplugged activity. This activity involves moving a "robot" around a game board while practicing tracing blocks of code by hand. To conclude, the lesson is wrapped up with a vocabulary discussion and a video.

Purpose

Students are introduced to the concept of loops through a robot maze activity. This unplugged lessons provides students a physical mental model they will be able to use when they start programming with loops in the subsequent lessons.

Agenda

Lesson Modifications

Warm Up (5 mins)

Set-up

Activity (30 mins)

Loops

Wrap Up (10 mins)

Assessment: Check For Understanding

View on Code Studio

Objectives

Students will be able to:

- Use appropriate vocabulary to describe loops.
- Identify the exit point of a loop.
- Trace a simple program with a loop

Preparation

- ☐ 1 game board per pair of students
- ☐ 1 "robot" per pair of students
- ☐ Game pieces, markers, or tokens that can be used to represent barriers
- ☐ Review the Intro to Loops presentation and click through all animations

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals - Presentation**
- **Loops Game Board** [Make a Copy](#)

Teaching Guide


Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (5 mins)

Set-up

 **Distribute:** The activity today involves a lot of moving pieces. Take this time at the beginning of class to pass out supplies and pair students.

Per pair of students:

- 1 game board
- 1 "robot"
- Game pieces, markers, or tokens that can be used to represent barriers

Group: Put students in groups of two.


Teaching Tip

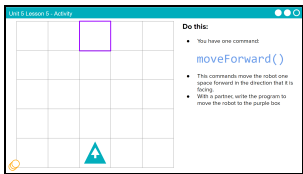

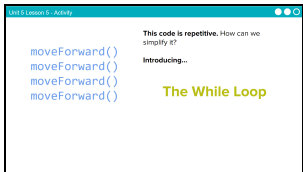
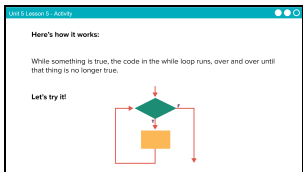
Supplies Substitutions:

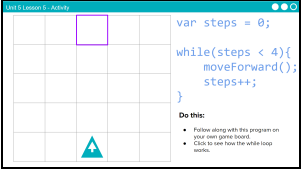
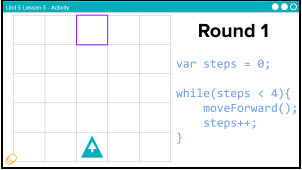
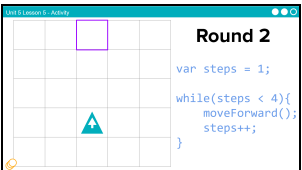
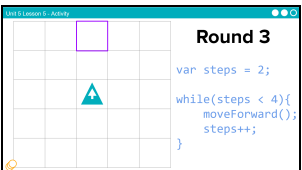
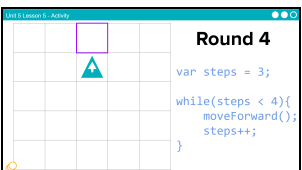
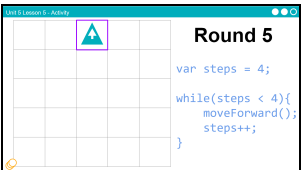
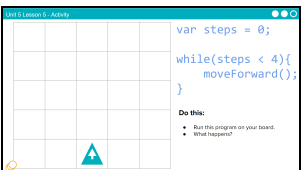
- The game pieces can be replaced with knick-knacks, tokens, markers, paperclips, or scraps of paper.
- The "robot" can be any small item that is clearly facing a direction. It could be as simple as a scrap of paper shaped like a triangle or a paperclip with googly eyes.

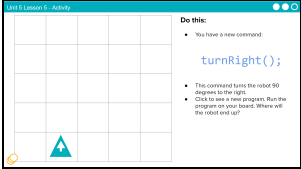
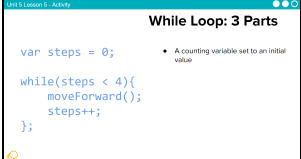
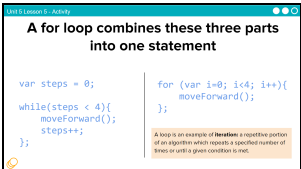
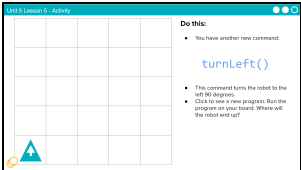
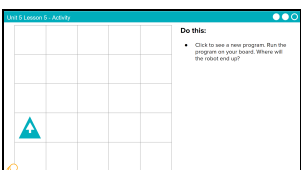
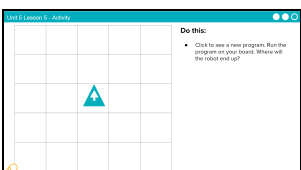
Activity (30 mins)

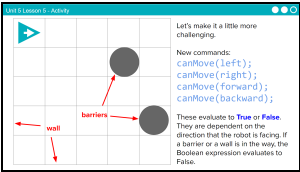
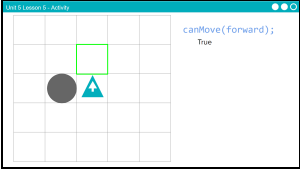
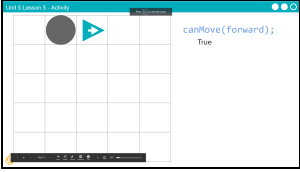
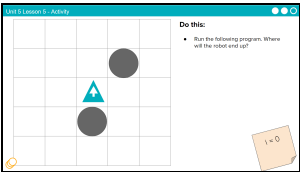
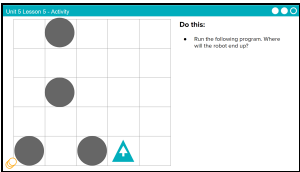
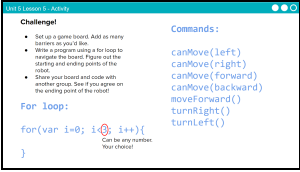
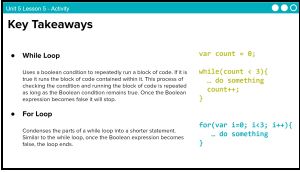
Loops

 **Display:** Use the activity slides for this lesson to guide the unplugged activity on Loops.

Slides	Speaker Notes
	<p>Say: Today we are going to explore Loops. You will move a robot around a game board following commands. Here's the first one: <code>moveForward()</code>.</p> <p>Do This: Use this command to write a program to move the robot to the purple box.</p> <p> Click through animation to reveal the answer.</p>
	<p>Say: This code is repetitive. How can we simplify it? By using a While Loop.</p>
	<p>Say: Here's how a While Loop works. While something is true, the code in the while loop runs, over and over until that thing is no longer true. Let's try it together!</p>


Slides	Speaker Notes
	<p>Say: Let's run this program together. Set up your board with the robot in the same place as the robot on the screen. We are going to run each step of the loop together. First you will try it on your board, and then we will look at the answers on the screen.</p>
	<p>Say: Read the code with your partner and complete the first round of the loop. You may want a sticky note or a scrap piece of paper out to keep track of the value in the variable.</p> <p>🕒 **Click through to view the answer after students have had a chance to follow the code on their own board. Discuss each part of the code as it is running.</p>
	<p>Say: Let's continue on with the second round. You do it first.</p> <p>🕒 **Click through to view the answer after students have had a chance to follow the code on their own board. Discuss each part of the code as it is running.</p>
	<p>Say: Now on to the third round. You do it first.</p> <p>🕒 **Click through to view the answer after students have had a chance to follow the code on their own board. Discuss each part of the code as it is running.</p>
	<p>Say: Time for the fourth round. You do it first.</p> <p>🕒 **Click through to view the answer after students have had a chance to follow the code on their own board. Discuss each part of the code as it is running.</p>
	<p>Say: What happens here? Discuss with your partner.</p> <p>**Click through to view the answer after students have had a chance to follow the code on their own board. Discuss each part of the code as it is running. The main thing to point out here is that the loop ends because the Boolean expression evaluates to false.</p>
	<p>Do This: With your partner, run this program on your board. What happens?</p> <p>🕒 Click for animation</p> <p>Discuss: Why does the robot run off the board?</p> <p>- this is an example of an infinite loop. The program never ends, because steps will always be less than 4.</p>

Slides	Speaker Notes
 <p>Slide 1: turnRight()</p> <p>Do this:</p> <ul style="list-style-type: none"> You have a new command <pre>turnRight();</pre> <ul style="list-style-type: none"> The command turns the robot 90 degrees to the right Click to see a new program. Run the program on your board. Where will the robot end up? 	<p>Say: You have a new command <code>turnRight()</code></p> <p>Do This: Run this program on your board. Where will the robot end up?</p> <p>🕒 Click through animation to see the answer</p>
 <p>Slide 2: While Loop: 3 Parts</p> <pre>var steps = 0; while(steps < 4){ moveForward(); steps++; };</pre> <ul style="list-style-type: none"> A counting variable set to an initial value 	<p>Say: A While Loop has three distinct parts.</p> <p>🕒 Click for animation: Read each step as the animation highlights the program code.</p>
 <p>Slide 3: A for loop combines these three parts into one statement</p> <pre>var steps = 0; while(steps < 4){ moveForward(); steps++; };</pre> <pre>for (var i=0; i<4; i++){ moveForward(); };</pre> <p>A loop is an example of iteration: a repetitive portion of an algorithm which repeats a specified number of times or until a given condition is met.</p>	<p>Say: A For Loop combines all three of these parts into one statement.</p> <p>Click for animation: Read each step as the animation highlights the program code.</p> <p>Say: A loop is an example of iteration: a repetitive portion of an algorithm which repeats a specified number of times or until a given condition is met.</p>
 <p>Slide 4: turnLeft()</p> <p>Do this:</p> <ul style="list-style-type: none"> You have another new command <pre>turnLeft();</pre> <ul style="list-style-type: none"> The command turns the robot to the left 90 degrees Click to see a new program. Run the program on your board. Where will the robot end up? 	<p>Say: You have another new command <code>turnLeft()</code></p> <p>🕒 Click to reveal the program code</p> <p>Do This: With your partner, follow the code with your robot.</p> <p>🕒 Click through the animation to see the answer</p>
 <p>Slide 5: Program on board</p> <p>Do this:</p> <ul style="list-style-type: none"> Click to see a new program. Run the program on your board. Where will the robot end up? 	<p>🕒 Click to reveal the program code</p> <p>Do This: Run the program on your board. Where will the robot end up?</p> <p>🕒 Click through the animation to see the answer</p>
 <p>Slide 6: Program on board</p> <p>Do this:</p> <ul style="list-style-type: none"> Click to see a new program. Run the program on your board. Where will the robot end up? 	<p>🕒 Click to reveal the program code</p> <p>Do This: Run the program on your board. Where will the robot end up?</p> <p>🕒 Click through the animation to see the answer</p>

Slides	Speaker Notes
	<p>Say: We're going to add some new commands to make things more challenging: <code>canMove(left)</code> <code>canMove(right)</code> <code>canMove(forward)</code> <code>canMove(backward)</code> . These evaluate to true or false and are dependent on the direction that the robot is facing. If a barrier or a wall is in the way, the Boolean expression evaluates to False.</p>
	<p>🕒 Click through the animation to see the answer</p>
	<p>🕒 Click through the animation to see the answer</p>
	<p>🕒 Click to reveal the program code</p> <p>Do This: Run the program on your board. Where will the robot end up?</p> <p>🕒 Click through the animation to see the answer</p>
	<p>🕒 Click to reveal the program code</p> <p>Do This: Run the program on your board. Where will the robot end up?</p> <p>🕒 Click through the animation to see the answer</p>
	<p>Say: Now it's your turn to write your own programs!</p> <p>Do This: With a partner, set up a game board and add as many barriers as you'd like. Write a program using a for loop to navigate the board. Figure out the starting and ending points of the robot. Share your board and code with another group. See if you agree on the ending point of the robot!</p>
	<p>🕒 Do This: Review the key takeaways with students.</p>

Wrap Up (10 mins)

📺 **Video:** As a class watch the video on Loops.

 **Journal:** Have students add to their journals: iteration and infinite loop.

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: When breaking a problem down, you often encounter elements that you want to use repeatedly in your code. Sometimes it's appropriate to write a new function; at other times it's appropriate to write a loop.

There is no hard-and-fast rule as to which is better, but what do you think? What kinds of circumstances would lead you to writing a function versus using a loop?

Teaching Tip

Running the Activity: This activity asks students to follow along as a number of core concepts for programming are introduced. The model is typically that a term or concept is introduced and modeled and then afterwards students are encouraged to try it out on their own. Trying it out typically means they are writing information on a sticky note and sharing it with another group before discussing the results with the whole class.

Slides with animations have an icon in the bottom left corner to let you know you need to click to reveal more of the slide's content.

To help you more easily prepare the activity and keep track of your instructions, detailed instructions have been included as speaker notes in the presentation. Here are some tips to help you throughout the presentation.

- There are opportunities throughout the presentation for students to actively engage. At these moments students should be making things with their manipulatives or using them to answer questions. Use these opportunities to check progress.
- There is a fair amount of new vocabulary introduced but it is introduced gradually and with intentional repetition. Make a point of actively modeling the use of new terms.
- The most important goal here is building a mental model. It is ok if students have some open questions that will get resolved over the subsequent conditional lessons.
- Both you and students can use the "Key Takeaways" to check your understanding at the end.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

CSP2021

- **AAP-2** - The way statements are sequenced and combined in a program determines the computed result



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

English ▼

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

Lesson 6: Loops Investigate

Overview

Students practice using the for loop in order to repeatedly run pieces of code. The lesson begins with a quick investigation of an app that flips coins. After that code investigation students complete another investigation with an app that uses loops to update screen elements.

Purpose

This lesson is students primary opportunity to get hands on with loops in code prior to the Practice activity in the following lesson. Give students as much class time as you can to work through these. For this lesson it's recommended that you place students in pairs as a support and to encourage discussion about the challenges or concepts they're seeing.

Agenda

Lesson Modifications

Warm Up (5 mins)

Activity (35 mins)

Wrap Up (5 mins)

Assessment: Check For Understanding

[View on Code Studio](#)

Objectives

Students will be able to:

- Read programs that use for loops
- Understand the parts of a for loop
- Update the Boolean expression in a for loop to change how many times the loop runs

Preparation

Review the two apps that students will be investigating and the questions about them. Note that there are target responses to each of these questions on the levels.

Links

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

For the Teachers

- [CSP Unit 5 - Lists Loops, and Traversals - Presentation](#)

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (5 mins)

Prompt: Imagine you are interested in finding out how much time it takes on average to walk from one end of your school to the next. You've decided to figure this out on your lunch break, and are able to complete the walk 20 times. What would your algorithm look like? Where could a loop show up?

- **Note:** You do not need to write your algorithm in a programming language. You can write it out in English or in pseudocode.

Discuss: Compare your algorithms with a partner and then discuss as a class.

Remarks

A loop helps simplify code. Let's investigate how this looks in two different apps.

Activity (35 mins)

Prompt: The first app we are going to investigate uses a loop in a simulation. What is a simulation? Why are they useful?

Remarks

Thanks for sharing your thoughts on simulations! Other examples of simulations are flight simulators which allows you to practice flying without damaging yourself or a real plane or a city building game where you can play with tweaking multiple values in the economy without real consequences.

These examples demonstrate the process of developing an abstract simulation where the specific details have been removed, or the functionality simplified.

While simulations are a useful way to model the real-world, we also need to be constantly on the lookout for bias in the real-world elements that are actually included in the simulation.

Today, we are going to look at a simple simulation.

Simulations can be helpful to run when it would take a long time or it would be too impractical to do something physically. For example, what if we wanted to see the results of a coin flip that we flipped a million times? That would take a long time to perform ourselves, but a computer can run that simulation much quicker. Let's look at an app that does just that.

Discussion Goal

Goal: The point of this activity is for students to see the need for a loop. Algorithms might look something like this:

- Repeat 20 times:
 - Stand at the entry doors to the school.
 - Start the timer.
 - Walk at a steady pace to the end of the hallway.
 - Turn off the timer.
 - Record the time in a notebook.
 - Walk back to the entry doors.
- Add all times together.
- Divide by 20 to find the average.

Teaching Tip

For this lesson, there are two separate investigations of two different apps. Lead the class through discussions, and circulate around the classroom as students work.

As time allows, encourage students to do the Modify tasks for each level. This will give students extra practice.

Discussion Goal

Simulations allow us to test solutions or run experiments in a way that is usually cheaper, safer, and often times faster. Simulations are models of the real world that allow the user to investigate hypotheses and draw conclusions. Oftentimes they involve inputting various sets of data or values to demonstrate the changing state of a phenomenon.

Group: Place students in pairs and have them navigate to the lesson on Code Studio.

Level 2 - Coin Flipper App: This code investigation includes a number of steps to help students get familiar with a new app. Students run the app and discuss the following:

- What information does the user input?
- How does the app process that information?
 - What is being repeated?
 - Where is there an opportunity to use a loop?
- What information does the app output?

Level 3 - Coin Flipper App: Students navigate to Level 3, run the program, watch the code run, and carefully read each individual part of the program before discussing the following:

Run the program for 10 coin flips, 100 coin flips, and 10,000 coin flips. When do you notice it taking longer? *On what lines of code is the program using a loop?* Which lines of code decide how many times to flip the coin? * What does the `++` command seem to do?

Prompt: How does a loop help when running similar simulations?

Remarks

Now let's investigate an app that uses loops in a different way. This time loops are used to update elements on the screen.

Level 4 - Font Tester App: Students navigate to Level 4 and run the app before discussing the following:

Prompt: Find the four different for loops in the program. What do they each do?

Prompt: Take a look at the names of the screen elements in Design Mode. Then navigate back to the code. How is `"text" + i` used? How does it evaluate with each round of the loop?

Do This:

- What happens if you change the text box variable names?
- Try changing them from `"text0"`, `"text1"`, etc. to `"textbox0"`, `"textbox1"`, etc.
- Navigate back to the code. What do you need to change in the code for the program to still work?

Do This:

- What happens if you change the Boolean expression `i < 5` in the for loops?
- Change all the Boolean expressions to one of the options below, run the program, and discuss. Then move to the next option and repeat:
 - `i < 4`
 - `i < 3`
 - `i <= 4`

Discussion Goal

A loop lets us repeat code quickly to determine an answer to a question.

Inside the loop we can use random number generation to get a variety of possible answers, similar to what would happen in the real world.

Wrap Up (5 mins)

Remarks

Let's review the parts of a for loop.

Prompt: What aspects of working with for loops do you feel like clicked today? What do you still feel like you have trouble with?

Discuss: Have students share with one another before sharing with the whole class.

Remarks

Working with for loops can be a little tricky. We'll have more opportunities to practice combining loops and lists together to make even more powerful apps.

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: Think back on the Font Tester App. Can you think of an example of another app or feature of an app which would use a loop to control different elements on a screen?

Discussion Goal

Goal: Use this opportunity to address any lingering questions or misconceptions in the room. You can also use this as a source of discussion topics to kick off the following lesson. As you lead the discussion, call out the many resources students have access to help when they're getting stuck.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming
- ▶ **DA** - Data & Analysis

CSP2021

- ▶ **AAP-3** - Programmers break down problems into smaller and more manageable pieces



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

English ▼

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

Lesson 7: Loops Practice

Overview

Students practice the basics of loops including using while loops, for loops, and updating multiple screen elements with a for loop. Along the way students develop debugging practices with loops.

Purpose

This lesson is students primary opportunity to get hands on with loops in code prior to the Make activity in the following lesson. Give students as much class time as you can to work through these. For this lesson it's recommended that you place students in pairs as a support and to encourage discussion about the challenges or concepts they're seeing.

Agenda

Lesson Modifications

Warm Up (5 mins)

Quick Warm Up

Activity (35 mins)

Practice Time

Wrap Up (5 mins)

Assessment: Check For Understanding: AP Practice

View on Code Studio

Objectives

Students will be able to:

- Write programs that use for loops with the support of sample code.
- Debug programs that use loops
- Use a for-loop to update multiple screen elements at once

Preparation

- ☐ Review the programming progression to understand what students will be asked to do

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals** - Presentation

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (5 mins)

Quick Warm Up

Remarks

Today we're going to have a chance to practice programming with a lot of the concepts and patterns we've explored over the last two lessons. I encourage you to go through these with a partner, but pay close attention to what each other is doing. In our next lesson you're going to have to use a lot of these on an independent project, and these activities are good practice for what you'll find there! Alright, let's get to it!

Teaching Tip

Move Quickly to the Activity: There's a lot in the main activity of today's lesson. You may optionally wish to do a quick vocabulary review or address any questions that came up in the last lesson. Otherwise, give students more time to get hands on with some code.

Activity (35 mins)

Practice Time

Remarks

Today you're mostly going to practice what we've learned about programming with loops. I'm here to help you when you need. However, I first want to remind you of the following:

- Use your debugging skills. Try to zoom in on precisely where you're getting stuck.
- Talk to your partner! That's what they're there for!
- Hover over blocks to read the documentation about how they work.
- Read the resources in the Help & Tips tab
- Talk to the group next to you. If another group asks for help make sure to take some time to talk it through with them.

We can debug loops by using the Watch Panel with our iterator variable, usually an `i`.

- Watcher Panel
 - Here you can see the length of a list in addition to all of the elements.

Do This: Direct students to Lesson 3, Level 2 on Code Studio.

Levels 2-4 While Loop Practice: Simple `console.log` levels in which students practice using while loops.

- Level 2: Students use a while loop to print the numbers 0-99
- Level 3: Students use a while loop to print a message 100 times
- Level 4: Students use a while loop to fill an array and print the results

Levels 5-7 For Loop Practice: Simple `console.log` levels to practice using for loops.

- Level 5: Students use a for loop to print the numbers 0-99

Teaching Tip

Providing Support: Circulate around the room through the lesson encouraging students to use the strategies introduced at the beginning of the lesson. Students have a number of supports at their fingertips, so a big part of your role is helping build their independence in using those resources.

- Level 6: Students use a for loop to print a message 100 times
- Level 7: Students use a for loop to fill an array and print the results

Levels 8-9 Loops and Screen Elements: In these levels students practice writing programs that modify screen elements with loops. Screen elements have already been designed to have repeated patterns (e.g. "dice0", "dice1", "dice2" ...) that make it possible to write code that modifies every screen element. This is important practice for the Make project in the next lesson.

- Level 8: This level is a simple introduction to using loops to generate a sentence with random font colors and font sizes.
- Level 9: Students must update a "Dice Roller" app that has some working functionality already. All of the different concepts they'll need to use are already in the starter code of this app, but they'll need to think through how to put them together to get the target code behavior.

Wrap Up (5 mins)

Prompt: What aspects of working with loops do you feel like clicked today? What do you still feel like you have trouble with?

Discuss: Have students share with one another before sharing with the whole class.

Remarks

Working with loops can be tricky - especially working with the iterator variable. We will get more practice tomorrow by making an app that uses loops in an interesting way.

Discussion Goal

Goal: Use this opportunity to address any lingering questions or misconceptions in the room. You can also use this as a source of discussion topics to kick off the following lesson. As you lead the discussion, call out the many resources students have access to help when they're getting stuck.

Assessment: Check For Understanding: AP Practice

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: What will be displayed after this code segment is run?

```
a ← 0
REPEAT 3 TIMES
  a ← a + 1
  DISPLAY a
  a ← a + 1
```

Question: What will be displayed after this code segment is run?

```
count ← 0
REPEAT UNTIL (count = 3)
  count ← count + 1
  DISPLAY "and"
  DISPLAY count
```

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming
- ▶ **DA** - Data & Analysis

CSP2021

- ▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result
- ▶ **CRD-2** - Developers create and innovate using an iterative design process



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

English ▼

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

Lesson 8: Loops Make

Overview

Using Programming Patterns and a step-by-step approach students make their own version of a Lock Screen Maker app. At the beginning of the lesson students are able to explore a working version of the app. They are then given the design elements of the app but begin with minimal starting code. A progression of levels guides students on the high level steps they should use to develop their app but leaves it to them to decide how to write the code. At the end students submit their apps which can be assessed using a provided rubric.

Purpose

This lesson is an opportunity for students to take on the "blank screen" and build the code that runs an app entirely from scratch. Guidance provided throughout the lesson helps students break down the large task of "building an app" into more incremental steps that they can use on future projects, including this unit's final project and the Create PT.

Agenda

Lesson Modifications

Warm Up (2 mins)

Intro the Project

Activity (38 mins)

Build the Lock Screen Maker App

Wrap Up (5 mins)

Assessment: Make Project

View on Code Studio

Objectives

Students will be able to:

- Recognize the need for programming patterns with loops as part of developing a functioning app
- Implement programming patterns with loops to develop a functioning app
- Write comments to clearly explain both the purpose and function of different segments of code within an app
- Use debugging skills as part of developing an app

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals - Presentation**

For the Students

- **CSP Loops Make - Lock Screen Maker - Activity Guide** [Make a Copy](#)

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (2 mins)

Intro the Project

Remarks

For the past few days, we've learned a lot about using lists to help apps keep track of information. In today's Make Project you'll be practicing using lists and programming patterns with loops to create a functioning Lock Screen app.

Teaching Tip

Short Intro: The Warm Up today is short and light. Students should spend the maximum amount of time working on their projects.

Activity (38 mins)

Build the Lock Screen Maker App

Group: Make a determination as to whether this project will be completed in pairs or individually. You may even choose to let students decide.

Distribute: Make sure students have access to scrap paper and pencils / paper for drawing flow charts.

Level 2 - Explore: Have students explore the working Lock Screen Maker app in Level 2. If students are not working in pairs they should still discuss the prompts with a neighbor.

Prompt:

- Where (if at all) do you think this app is using a loop?
- Where (if at all) do you think this app is using a list?

Remarks

Now let's build the this app. The screen has been set up for you - it's your job to add the code!

Distribute: Give students copies of **CSP Loops Make - Lock Screen Maker - Activity Guide** if you will be using it during the class.

Do This: Direct students to level three where they complete the Lock Screen Maker app. Based on the needs of your classroom decide whether you will collectively go through the activity guide or have students complete it individually. Afterwards give them time to work on their projects and circulate the room to offer support. Students who finish early can work on the extensions suggested in the activity guide.

Submit: Encourage students to check the rubric on the last page of the Activity Guide before submitting.

Discussion Goal

Discussion Goals:

Where (if at all) do you think this app is using a list?



- It's more important that students justify their answers. However, students may recognize that the different icons are likely being chosen from a list.

Where (if at all) do you think this app is using a loop?

- Students will hopefully recognize that with 20 different icons to be updated by each button there is almost certainly a loop being used by each of those button.

Wrap Up (5 mins)

Remarks

  Awesome work today! Make sure to submit your project when you're done with it!

Assessment: Make Project

Use the rubric provided with the project to assess student projects.

Teaching Tip

Supporting Students: While students are working on their apps, circulate the room and check in with students who need a little help. Encourage students to collaborate and discuss bugs with each other.

Debugging: Review with students steps they can use to debug if they get stuck:

- Build small parts of the program at a time and test them to make sure they work
- Run the code on turtle mode
- Add the variables to the watcher
- Explain the code to a friend and clarify the differences between what they expect to happen and what is actually happening in

Teaching Tip

Maximize Work Time: The wrap up is short to allow the maximum amount of time for students to complete the activity.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

► **AP** - Algorithms & Programming



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

English ▼

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

Lesson 9: Traversals Explore

Overview

The lesson begins with a quick review of lists and loops before moving into the main activity. Here students explore the concept with the Traversal Machine, a physical model of traversal using a for loop.

Purpose

Students are introduced to the concept of traversal using the Traversal Machine. This unplugged lessons provides students a physical mental model they will be able to use when they transition to programming traversals in App Lab.

Agenda

Lesson Modifications

Warm Up (5 mins)

Activity (30 mins)

Traversals

Wrap Up (10 mins)

Assessment: Check For Understanding

[View on Code Studio](#)

Objectives

Students will be able to:

- Use appropriate vocabulary to describe traversals.
- Understand how to use a loop to traverse a list
- Trace simple programs with loop traversals

Preparation

Per pair of students:

- ☐ 1 Traversal Machine
 - ☐ The index sheet contains enough sections to divide between six students
- ☐ Preview the Intro to Traversals slideshow. Click through all animations.

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals -** Presentation
- **Traversal Machine** [Make a Copy](#)

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click **here** to access the modifications.

Warm Up (5 mins)

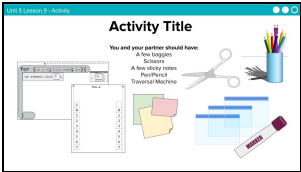
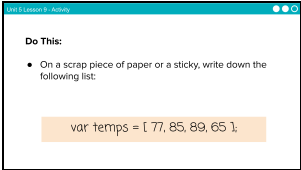
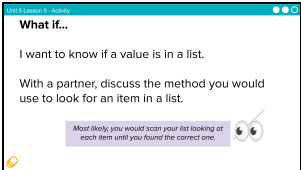

Discuss: Review key vocabulary for lists and loops with students:

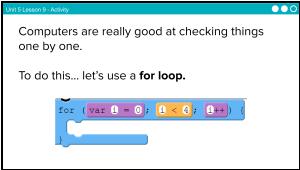
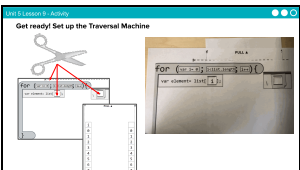
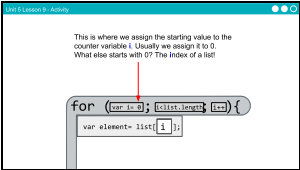
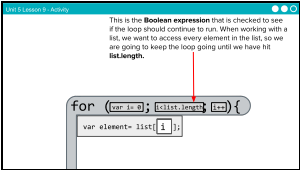
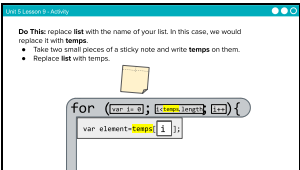
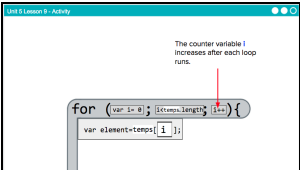
- List - an ordered sequence of elements.
- Element - an individual value in a list that is assigned a unique index.
- Index - a common method for referencing the elements in a list or string using natural numbers.
- Iteration - a repetitive portion of an algorithm. Iteration repeats until a given condition is met.

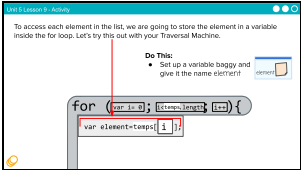

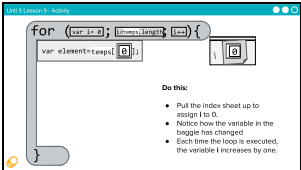

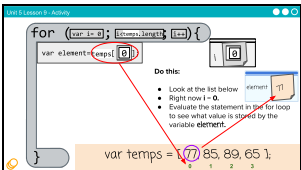

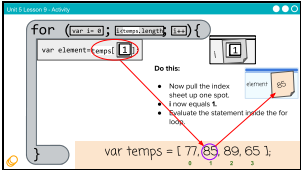

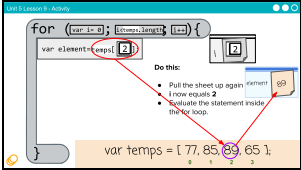

Activity (30 mins)

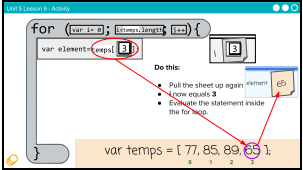

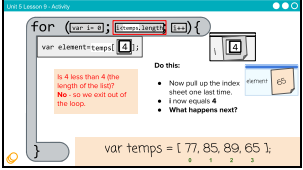

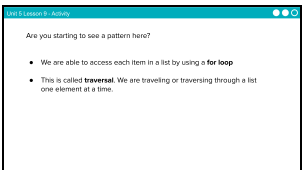
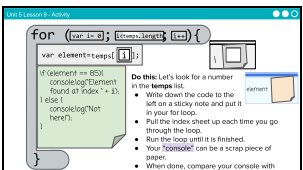
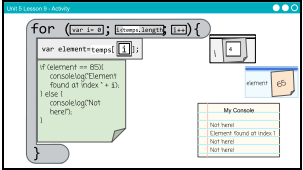
Traversals

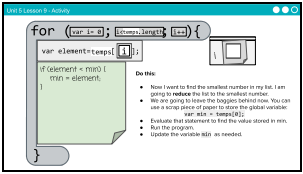
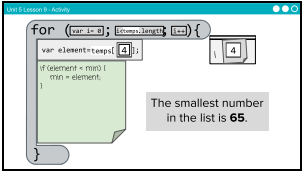
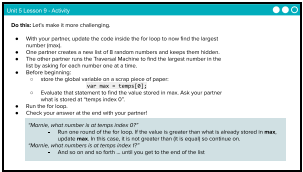
Display: Use the activity slides for this lesson to guide the unplugged activity on Traversals.

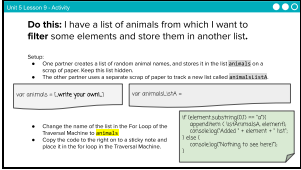
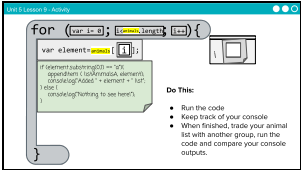
Slides	Speaker Notes
	<p>Say: Today we are going to learn about Traversals. Make sure you and your partner have all the supplies you see on the screen.</p> <p>Note to Teacher: If you are concerned about time or material management, the Traversal Machines can be prepared ahead of time. You only need one Traversal Machine per pair of students, and machines can be shared between classes.</p> <p>There are three places to make cuts:</p> <ul style="list-style-type: none">• The dotted line at the top of the sheet• The box with the list index• The box inside of the variable baggy
	<p>Do This: On a scrap piece of paper or a sticky note, write down the list on the screen.</p>
	<p>Prompt: What if I want to know if a value is in a list? With a partner, discuss the method you would use to look for an item in a list.</p> <p> Click for animation to reveal suggested answer</p>

Slides	Speaker Notes
	<p>Say: Computers are really good at checking things one by one. To do this... let's use a for loop.</p>
	<p>Do This: Set up the Traversal Machine.</p> <p>Note: Students use scissors make cuts as directed on the Traversal Machines. As noted before, this can all be done before class to save time.</p>
	<p>Do This: Direct students attention to their Traversal Machines as you describe the parts.</p> <p>Say: This is where we assign the starting value to the counter variable i. Usually we assign it to 0. What else starts with 0? The index of a list!</p> <p>Note: In Javascript, the index of a list starts at 0. You may want to remind students that on the AP Exam, pseudocode starts the index at 1.</p>
	<p>Say: This is the Boolean expression that is checked to see if the loop should continue to run. When working with a list, we want to access every element in the list, so we are going to keep the loop going until we have hit list.length.</p> <p>Note: You may want to remind students that list.length evaluates to the length of a list, which will be different than the last index number. For example, in a list of three, the length is 3 and the index numbers are 0, 1, 2</p>
	<p>Do This: replace list with the name of your list. In this case, we would replace it with temps.</p>
	<p>Say: The counter variable i increases after each loop runs.</p>

Slides	Speaker Notes
 <p>Unit Lesson 3: Array</p> <p>To access each element in the list, we are going to store the element in a variable inside the for loop. Let's try this out with your Traversal Machine.</p> <p>Do this:</p> <ul style="list-style-type: none"> Set up a variable baggy and give it the name element <pre>for (var i = 0; i < temps.length; i++) { var element = temps[i]; }</pre>	<p>Say: To access each element in the list, we are going to store the element in a variable inside the for loop. Let's try this out with your Traversal Machine.</p> <p> Click for animation</p> <p>Do This: Set up a variable baggy and give it the name element.</p> <p>Note: You may opt to not use variable baggies at all in this lesson and instead have students keep track of variables on a scrap piece of paper. In this case, the variable element is tracking the local variable that is inside of the for loop. With every round of the for loop, the variable will update. It's not necessary to build a baggy for the variable i - this will be represented on the Traversal Machine itself as a drawing of a baggy.</p>
 <p>Unit Lesson 3: Array</p> <pre>for (var i = 0; i < temps.length; i++) { var element = temps[i]; }</pre> <p>Do this:</p> <ul style="list-style-type: none"> Pull the index sheet up to assign i to 0 Notice how the variable in the baggy has changed Each time the loop is executed, the variable i increases by one. 	<p>Do This: Pull the index sheet up to assign i to 0.</p> <p> Click for animation</p> <p>Say: Notice how the variable in the baggy has changed. Each time the loop is executed, the variable i increases by one.</p> <p>Note: For the following six slides, students will follow along with what's happening on the screen with their own Traversal Machines. Give students a few seconds to pull up the index sheet at each step.</p>
 <p>Unit Lesson 3: Array</p> <pre>for (var i = 0; i < temps.length; i++) { var element = temps[i]; }</pre> <p>Do this:</p> <ul style="list-style-type: none"> Look at the list below Right now i = 0 Evaluate the statement in for for loop to see what value is stored by the variable element <p>var temps = [77, 85, 89, 65];</p>	<p>Say: Look at the list on the screen. Right now i=0.</p> <p>Do This: Evaluate the statement in the for loop to see what value is stored by the variable element.</p> <p> Click through animation: There are six steps to click through. temps[0] evaluates to 77 which is stored in the variable element</p>
 <p>Unit Lesson 3: Array</p> <pre>for (var i = 0; i < temps.length; i++) { var element = temps[i]; }</pre> <p>Do this:</p> <ul style="list-style-type: none"> Now pull the index sheet up one spot i now equals 1 Evaluate the statement inside the for loop <p>var temps = [77, 85, 89, 65];</p>	<p>Do This: Now pull the index sheet up one spot. i now equals 1. Evaluate the statement inside the for loop.</p> <p> Click through animation: There are seven steps to click through. temps[1] evaluates to 85 which is stored in the variable element</p>
 <p>Unit Lesson 3: Array</p> <pre>for (var i = 0; i < temps.length; i++) { var element = temps[i]; }</pre> <p>Do this:</p> <ul style="list-style-type: none"> Pull the sheet up again i now equals 2 Evaluate the statement inside the for loop <p>var temps = [77, 85, 89, 65];</p>	<p>Do This: Pull the index sheet up again. i now equals 2. Evaluate the statement inside the for loop.</p> <p> Click through animation: There are seven steps to click through. temps[2] evaluates to 89 which is stored in the variable element</p>

Slides	Speaker Notes
	<p>Do This: Pull the index sheet up again. i now equals 3. Evaluate the statement inside the for loop.</p> <p> Click through animation: There are seven steps to click through. temps[3] evaluates to 65 which is stored in the variable element</p>
	<p>Do This: Now pull up the index sheet one last time. i now equals 4.</p> <p>Prompt: What happens next?</p> <p> Click through animation There are three steps to click through.</p> <p>Say: Is 4 less than 4 (the length of the list)? No - so we exit out of the loop.</p>
	<p>Say: Are you starting to see a pattern here?</p> <ul style="list-style-type: none"> • We are able to access each item in a list by using a for loop. • This is called traversal. We are traveling or traversing through a list one element at a time.
	<p>Say: Let's look for a number in the temps list. It's easy for you to scan the list quickly with your own eyes, but for this exercise, let's pretend you are a computer and must use the Traversal Machine to look for the number.</p> <p>Do this:</p> <ul style="list-style-type: none"> • Write down the code to the left on a sticky note and put it in your for loop. • Pull the index sheet up each time you go through the loop. • Run the loop until it is finished. • Your "console" can be a scrap piece of paper. • When done, compare your console with another group.
	<p>Say: Check your console. Does it look like what you see on the screen? The number 85 was found at index 1, the second number in the list. Notice that i currently has the value 4. This is because we stopped after four rounds. Element is currently storing 65, the last element that we accessed, at index 3.</p>

Slides	Speaker Notes
	<p>Say: Now I want to find the smallest number in a list. I am going to reduce the list to the smallest number. We are going to leave the baggies behind now. You can use a scrap piece of paper to store the global variable <code>var min = temps[0]</code></p> <p>Do This:</p> <ul style="list-style-type: none">• Evaluate that statement to find the value stored in min.• Run the program• Update the variable min as needed
	<p>Say: The smallest number in the list is 65. Again, this is easy for you to quickly scan with your eyes when your list is short, but imagine if your list contained 1,000 or 1,000,000 elements? This is where a computer using a loop really shines.</p>
	<p>Say: Let's make it more challenging.</p> <p>Do This:</p> <ul style="list-style-type: none">• With your partner, update the code inside the for loop to now find the largest number (max)• One partner creates a new list of 8 random numbers and keeps them hidden.• The other partner runs the Traversal Machine to find the largest number in the list by asking for each number one at a time.• Before beginning:<ul style="list-style-type: none">◦ store the global variable on a scrap piece of paper: <code>var max = temps[0];</code>◦ Evaluate that statement to find the value stored in max. Ask your partner what is stored at "temps index 0"• Run the for loop.• Check your answer at the end with your partner! <p>Note: Read the example at the bottom of the screen out loud so students understand how they should interact while using the Traversal Machine. Allow students five or so minutes to fully complete the task on this slide. Here's a sample of what their updated code might look like:</p> <pre>if(element>max) {max = element; }</pre>

Slides	Speaker Notes
	<p>Say: Now I have a list of animals from which I want to filter some elements and store them in another list.</p> <p>Note: Guide the students through the setup listed on the screen.</p> <p>Do This:</p> <ul style="list-style-type: none"> One partner creates a list of eight random animals, and stores it in the list animals on a scrap of paper. Keep this list hidden. The other partner uses a separate scrap of paper to track a new list called animalsListA Change the name of the list in the for loop of the Traversal Machine to animals. Copy the code to the right on to a sticky note and place it in the for loop of the Traversal Machine.
	<p>Do This:</p> <ul style="list-style-type: none"> Run the code Keep track of your console When finished, trade your animal list with another group, run the code, and compare your console outputs <p>Note: Expect this activity to take around 5 minutes. When students have finished, move on to the Takeaways in the Wrap Up</p>

Wrap Up (10 mins)

Do This: Review key takeaways with the class.

Video: As a class watch the video on Traversals.

Journal: Have students add the definition for the word traversal to their journal.

Remarks

We can use traversals to modify data in several different ways that we will investigate tomorrow including transforming or changing each element in a list, filtering a dataset, and combining or comparing data.

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: Why is traversal with a for loop a good method for accessing and updating items in a lengthy list?

💡 Teaching Tip

Running the Activity: This activity asks students to follow along as a number of core concepts for programming are introduced. The model is typically that a term or concept is introduced and modeled and then afterwards students are encouraged to try it out on their own. Trying it out typically means they are writing information on a sticky note and sharing it with another group before discussing the results with the whole class.

Slides with animations have an icon in the bottom left corner to let you know you need to click to reveal more of the slide's content.

To help you more easily prepare the activity and keep track of your instructions, detailed instructions have been included as speaker notes in the presentation. Here are some tips to help you throughout the presentation.

- There are opportunities throughout the presentation for students to actively engage. At these moments students should be making things with their manipulatives or using them to answer questions. Use these opportunities to check progress.
- There is a fair amount of new vocabulary introduced but it is introduced gradually and with intentional repetition. Make a point of actively modeling the use of new terms.
- The most important goal here is building a mental model. It is ok if students have some open questions that will get resolved over the subsequent conditional lessons.
- Both you and students can use the "Key Takeaways" to check your understanding at the end.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result
- ▶ **DAT-2** - Programs can be used to process data



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

English ▼

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

Lesson 10: Traversals Investigate

Overview

In this lesson students work with partners to investigate three different apps that use traversal to access items in a lists. Students first explore all three apps without seeing the code to notice similarities and predict how they will work. Then they explore the code itself and make additions and modifications to the apps. To conclude the lesson, students review and discuss common programming patterns with traversals.

Purpose

After building a conceptual model using a for loop to traverse a list in the previous lesson, this lesson allows students to see how this is actually implemented in code. This lesson also introduces common programming patterns when using lists and traversals. Students will have some opportunities to modify working code in this lesson, but the most significant practice with lists and traversals will come in the following lesson.

Agenda

Lesson Modifications

Warm Up (0 mins)

Activity (35 mins)

Mile Tracker

Data Tab Investigate

Random Dog Picker

Wrap Up (5 mins)

Assessment: Check For Understanding

View on Code Studio

Objectives

Students will be able to:

- Identify common programming patterns using traversals
- Explain the purpose of programming patterns with traversals both in terms of how they work and what they accomplish
- Modify apps that make use of common programming patterns with traversals to adjust their functionality

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals - Presentation**


Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (0 mins)

 **Prompt:** Let's review the Traversal Machine. How did it work?


Discussion Goal

Goal: The Traversal Machine let us visualize how a for loop is used to traverse a list and interact with each element in the list one by one.


Activity (35 mins)


Mile Tracker

 **Group:** Group students into groups of four.

 **Level 2:** Students read the entire code, and then students focuses their attention on a single function. Student should re-read the code for their function.

- Student A - `average()` (lines 32-38)
- Student B - `slow()` (lines 40-48)
- Student C - `fast()` (lines 50-58)
- Student D - `numberedListDisplay()` (lines 60-66)

 **Discuss:** Students discuss in their groups how each of their functions work, specifically referencing the list it uses and how it is traversed using a for loop. Then as a class address any confusion over how the app works.

 **Modify:** Students work with a partner to create a function that adds together the total time of every element in the list. `console.log` the result. Call this function in the `updateScreen()` function.


Teaching Tip


Prepping for Investigate Lessons: The best way to prepare for this lesson is to go through the experience yourself. Check out the three apps in Code Studio to get a sense for how they work. Then move on to the Code Investigation and actually try to answer all the questions for each app. These questions are intended to help students read and make sense of the code. It is not about getting the “right answer” as much as it is about building skills in your students to read, test, and make sense of code that they haven’t seen before. Finally, you will notice that for the levels where the directions ask students to “Modify” the code, you will find “Example Solutions” in the “Teacher Panel” of that level.

Display Code at the Front: If your room allows it, display the code during the Code Investigation at the front of the room. When students mentions specific lines of code actually scroll to that line and read through it together.



Save Modifications for the End: This lesson can be tight on time. Rather than have students modify the code all at once, you can save modifications for the end of the Code Investigation and have students pick a single app they wish to modify.

Data Tab Investigate

 **Level 3 (5 mins):** Students read the code, and then investigate the data tab.

- **Discuss:** Encourage students to share out what they found in the data tab, and how they can interact with it with code.
-  **Modify:** Students choose a new dataset to import to the project. Then students modify the code to print out a column of data from the chosen dataset. Point out to students that when they pull out a column, they are creating a list.

Random Dog Picker

  **Level 4:** Instruct students to run the app and then carefully read it. There is a lot of information in the comments in this app. Students work through all of the questions on the screen.

Discuss: Go through the questions one by one and answer as a class. Afterwards, go line by line with the class explaining how the app works.

Levels 5-6: Review the patterns in these levels as a class.

- Have students add any relevant notes about the patterns to their journals.
- Discuss which patterns were used in the apps today.

Wrap Up (5 mins)

Prompt: What aspects of using traversals to process a list do you feel you already understand? What questions do you want to dig into more tomorrow during the practice lesson?

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: Explain how you would filter the dog dataset using traversal to have a filtered list of dogs who live long lives.

Discussion Goal

Discussion Goal: Make sure students arrive at the following answers.

- *What are the names of the five lists in this program?*
 - `dogNames`, `dogHeight`, `dogImages`, `filteredDogNames`, `filteredDogImages`
- *On what lines of code are the lists created?*
 - `dogNames` : 2
 - `dogHeight` : 3
 - `dogImages` : 4
 - `filteredDogNames` : 7
 - `filteredDogImages` : 8
- **On what line's of code are the lists filled?*
 - `dogNames` : 2
 - `dogHeight` : 3
 - `dogImages` : 4
 - `filteredDogNames` : 32-43
 - `filteredDogImages` : 32-43
- *How are these lists filled?*
 - `dogNames`, `dogHeight`, and `dogImages` are filled by pulling a column of data from the dataset
 - For `filteredDogNames` and `filteredDogImages`, the `dogHeight` list is traversed. If `dogHeight` and `dogSize` meet the given conditions the names and images at the same index position in the `dogNames` and `dogImages` lists are added to the filtered lists.
- *Open up the data tab and click to view the dogs table. What columns does this app use?*
 - "Name", "Maximum Height", "Image"
- *Look at the filter function. On what lines are the filtered lists reset to blank lists?*
 - Lines 23 & 24
- *What condition is being checked to determine if an element belongs in a filtered list?*
 - If a dog height is above or below a certain number

💡 Teaching Tip

Traversals can be tricky! In this case, we are filtering a list by information that is in another list.

There are three lists to start: `dogNames` , `dogHeights` , and `dogImages` . We want to sort out the dogs so that when the user picks a size from the dropdown, only dogs of that size show up in the display.

To do this, we traverse using a filter pattern. Using a loop to go through each item in the `dogHeights` list, if a dog's height matches up to the requirements, we pull the dog's image and name from those original lists (all the indexes will match) and put those in the new filtered lists.

Once this is done, we have filtered lists that we can pull from to display a random dog's name and image.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-3** - Programmers break down problems into smaller and more manageable pieces



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

English ▼

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

Lesson 11: Traversals Practice

Overview

Students practice traversing lists, filtering and reducing lists, and using the data import tools. Along the way students develop debugging practices with traversals.

Purpose

This lesson is students primary opportunity to get hands on with lists in code prior to the Make activity in the following lesson. Give students as much class time as you can to work through these. For this lesson it's recommended that you place students in pairs as a support and to encourage discussion about the challenges or concepts they're seeing.

Agenda

Lesson Modifications

Warm Up (0 mins)

Quick Warm Up

Activity (35 mins)

Practice Time

Wrap Up (5 mins)

Synthesizing Discussion

Assessment: Check For Understanding: AP Practice

View on Code Studio

Objectives

Students will be able to:

- Write programs that use list traversals, including the filter and reduce patterns, with the support of sample code
- Debug programs that use list traversals

Preparation

- ▣ Review the levels that students will be completing with an eye for how you will encourage them to use the debugging practices emphasized in today's lesson.

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals - Presentation**

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (0 mins)

Quick Warm Up

Remarks

Today we're going to have a chance to practice programming with a lot of the concepts and patterns we've explored over the last two lessons. I encourage you to go through these with a partner, but pay close attention to what each other is doing. In our next lesson you're going to have to use a lot of these on an independent project, and these activities are good practice for what you'll find there! Alright, let's get to it!

Teaching Tip


Move Quickly to the Activity: There's a lot in the main activity of today's lesson. You may optionally wish to do a quick vocabulary review or address any questions that came up in the last lesson. Otherwise, give students more time to get hands on with some code.

Activity (35 mins)


Practice Time


Group: It is recommended that students work in pairs for this lesson and a number of the activities feature discussion prompts. Consider using pair programming, having drivers and navigators switch every 3 minutes, not every level.

Remarks

 Today you're mostly going to practice what we've learned about programming with traversals. I'm here to help you when you need. However, I first want to remind you of the following:

- Use your debugging skills. Try to zoom in on precisely where you're getting stuck.
- Talk to your partner! That's what they're there for!
- Hover over blocks to read the documentation about how they work.
- Read the resources in the Help & Tips tab
- Talk to the group next to you. If another group asks for help make sure to take some time to talk it through with them.

 We can debug traversals by using many skills that helped us with lists like using the watch panel and console.log. Another important new debugging skill will be to actually go look at your data in the Data tab. Use console.log to make sure you're successfully getting the data you want.

 **Levels 2-4 Traversal Practice** These levels have students perform simple traversals over lists that are created inside their code (not with the data import tools). In each program sample code is provided which students can use to help writing the code of their own.

- Level 2: traverse over a list and console.log every element
- Level 3: traverse over a list and console.log every element and its position

Teaching Tip

Providing Support: Circulate around the room through the lesson encouraging students to use the strategies introduced at the beginning of the lesson. Students have a number of supports at their fingertips, so a big part of your role is helping build their independence in using those resources.

- Level 4: traverse over two parallel lists and print elements from both

Levels 5-7 Reduce and Filter Practice: Students practice the reduce and filter patterns introduced in the previous lesson. In each program sample code is provided which students can use to help writing the code of their own.


- Level 5: filter a list of students to keep only those with more than 6 letters in their names.
- Level 6: reduce a list to find the maximum price inside of it.
- Level 7: filter a list of countries to find those in the "Central America" region. This program makes use of the data import tools.

Levels 8 App Practice: In this level, students use the reduce pattern to finish building an app.

- Level 8: Write code for the "reduce" pattern to calculate a student's average grade

Wrap Up (5 mins)

Synthesizing Discussion

 **Prompt:** What aspects of working with traversals do you feel like clicked today? What do you still feel like you have trouble with?

Discuss: Have students share with one another before sharing with the whole class.

Remarks

Working with traversals can be tricky. We will get more practice tomorrow by making an app that uses traversals and the data tools.

Discussion Goal

Goal: Use this opportunity to address any lingering questions or misconceptions in the room. You can also use this as a source of discussion topics to kick off the following lesson. As you lead the discussion, call out the many resources students have access to help when they're getting stuck.

Assessment: Check For Understanding: AP Practice

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: What is stored in `studentScores` after running the program code?

```
studentScores ← [77, 32, 45, 92, 86]

FOR EACH item IN studentScores
{
  IF (item > 60)
  {
    item ← item + 5
  }
  ELSE
  {
    item ← 0
  }
}
```

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

CSP2021

- **AAP-2** - The way statements are sequenced and combined in a program determines the computed result



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

English ▼

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

Lesson 12: Traversals Make

Overview

Using Programming Patterns and a step-by-step approach students make their own version of a Random Forecaster app. At the beginning of the lesson students are able to explore a working version of the app. They are then given the design elements of the app but begin with a blank screen. Students use an Activity Guide to go through the high level steps they should use to develop their app but leaves it to them to decide how to write the code. At the end students submit their apps which can be assessed using a provided rubric.

Purpose

This lesson is an opportunity for students to take on the "blank screen" and build the code that runs an app entirely from scratch. Guidance provided throughout the lesson helps students break down the large task of "building an app" into more incremental steps that they can use on future projects, including this unit's final project and the Create PT.

Agenda

Lesson Modifications

Warm Up (2 mins)

Intro the Project

Activity (40 mins)

Build the Random Forecaster App

Wrap Up (5 mins)

Assessment: Make Project

[View on Code Studio](#)

Objectives

Students will be able to:

- Recognize the need for programming patterns with traversals as part of developing a functioning app
- Implement programming patterns with traversals to develop a functioning app
- Write comments to clearly explain both the purpose and function of different segments of code within an app
- Use debugging skills as part of developing an app

Preparation

- ☐ If time allows, try to build all or part of the Random Forecaster app yourself to understand the challenges involved
- ☐ Review the debugging practices you intend to reinforce and resources you'll direct students towards as they get stuck

Links

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

For the Teachers

- [CSP Unit 5 - Lists Loops, and Traversals - Presentation](#)

For the Students

- [CSP Traversals Make - Random Forecaster App](#) - Activity Guide [Make a Copy](#)

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (2 mins)



Intro the Project

Remarks

For the past few days, we've learned a lot about using traversals to make apps that manipulate large amounts of information. In today's Make Project you'll be practicing processing lists and using programming patterns with traversals to create a functioning Random Forecaster app.

Teaching Tip

Short Intro: The Warm Up today is short and light. Students should spend the maximum amount of time working on their projects.

Activity (40 mins)

Build the Random Forecaster App

Group: Make a determination as to whether this project will be completed in pairs or individually. You may even choose to let students decide.

Do This: Have students explore the working Random Forecaster app in Level 2.



Prompt: If students are not working in pairs they should still discuss the prompts with a neighbor.

- What information is needed to create this app?
- What list filtering patterns might be used?



Remarks

Now let's build the this app. The screen has been set up for you - it's your job to add the code!

Do This: Direct students to level three where they complete the Random Forecaster app. An optional Activity Guide is provided if students would like guidance in creating the app. The most relevant programming pattern is displayed on a slide. Review this pattern quickly with students, if needed.

Submit: Encourage students to check the rubric on the last page of the Activity Guide before submitting.

Discussion Goal

- What information is needed to create this app?
 - The weather forecast for tomorrow for random cities, including the weather condition, high and low temperatures, and a weather icon.
- What list filtering patterns might be used?
 - The List Filter Pattern: Filtering Multiple Lists is used.

Wrap Up (5 mins)

Remarks

Awesome work today! Make sure to submit your project when you're done with it!

Assessment: Make Project

Use the rubric provided with the project to assess student projects.

💡 Teaching Tip

Supporting Students: While students are working on their apps, circulate the room and check in with students who need a little help. Encourage students to collaborate and discuss bugs with each other.

Debugging: Review with students steps they can use to debug if they get stuck:

- Run the code on turtle mode
- Add the variables to the watcher
- Explain the code to a friend

💡 Teaching Tip

Maximize Work Time: The wrap up is short to allow the maximum amount of time for students to complete the activity.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

► **AP** - Algorithms & Programming



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

English ▼

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

Lesson 13: Project - Hackathon Part 1

Overview

This is the first day of a five-day unit project. Students begin the project by choosing a partner, determining a dataset to design the app around, and creating a paper prototype.

Purpose

Students will demonstrate their app design and programming skills throughout this five day project. In addition, students work with a dataset as this ensures students will be using the types of programming constructs required for the Create Performance Task, which this project is designed as a practice for. Students complete the project by individually filling out a Written Response, modeled after the Create PT.

This project can be used as a unit project, or as an end cap to the first semester of the course.

Agenda

Lesson Modifications

Warm Up (3 mins)

Activity (40 mins)

Hackathon Prep

Hackathon Project

Wrap Up (2 mins)

Assessment: Optional

View on Code Studio

Objectives

Students will be able to:

- Effectively plan a project using a paper prototype
- Determine a dataset for project usage

Preparation

- ☐ Read through the Planning Guide, paying particular focus to the Scoring Guidelines
- ☐ Go to the last lesson of the project to review the **Example Written Response Submission** and **Example Project App Submission** (in the blue teacher panel)

Links

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

For the Teachers

- CSP Unit 5 - Lists Loops, and Traversals - Presentation**

For the Students

- CSP U5 Hackathon Project Planning Guide** - Activity Guide [Make a Copy](#)
- CSP U5 Hackathon Project Written Response** - Written Response [Make a Copy](#)

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (3 mins)

Remarks

Today we are beginning the Hackathon Project, which is an opportunity for you to show off what you've learned so far.

What is a hackathon? Traditionally, hackathons are events where people come together to build something creative in a short amount of time. Sometimes hackathons are centered around a goal or a problem to solve. Usually team members take on different roles (programming, designing, etc.) to get the project done within the time limit.

You will work with a partner on the Hackathon Project, using a database to create an interesting app.

Activity (40 mins)

Hackathon Prep

Group: Place students in groups of two.

Distribute: Pass out the **CSP U5 Hackathon Project Planning Guide - Activity Guide** - one per group.

Read: As a class, read through the Project Description on the Planning Guide.

Hackathon Project

Step 1: Students navigate to a previous project and look at the dataset options. They choose a dataset that looks interesting to them.

Step 2: Now students select how they will traverse a list pulled from their chosen dataset. Using the Planning Guide, students select whether they will use the filter, map, or reduce pattern and explain the specifics.

- **Filter** (most common option): use the list from one column to determine information that will be filtered from a list created by another column
 - Example: dogHeight filters dogNames, so only the names of small dogs are added to the filtered list
- **Map:** Add or change each item in a list
 - Example: map a list of numbers pulled from a column using Math.round - now each number is rounded
- **Reduce:** Reduce the data in a list to a single number
 - Example: find the smallest number in a list

Step 3: The majority of the lesson should be spent on creating a detailed paper prototype. Students work together to draw out the screens of their app and design the flow from one screen to the next.

Wrap Up (2 mins)

Remarks

■

Teaching Tip

Forming Groups: You may opt to form the groups yourself, randomly place students in groups, or let students select their partners based on the dataset they want to work with.

Previewing the Written Responses: You may opt to share the **CSP U5 Hackathon Project Written Response - Written Response** with students early just so they know what they'll be expected to write at the end of the 5-day project.

In the next lesson, you will decide your project roles and we will begin to translate your paper prototype to the screen.

Assessment: Optional

Planning Guide: As an optional completion assessment, you can look at steps 1-3 in the Planning Guide.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

► **AP** - Algorithms & Programming



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

English ▼

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

Lesson 14: Project - Hackathon Part 2

Overview

This is the second day of a five-day unit project. Students continue to plan for the project by filling out tables of information on element IDs and programming constructs before each tackling a different role in the project as a designer or a programmer.

Purpose

Students will demonstrate their app design and programming skills throughout this five day project. In addition, students work with a dataset as this ensures students will be using the types of programming constructs required for the Create Performance Task, which this project is designed as a practice for. Students complete the project by individually filling out a Written Response, modeled after the Create PT.

This project can be used as a unit project, or as an end cap to the first semester of the course.

Agenda

Lesson Modifications

Warm Up (0 mins)

Activity (40 mins)

Hackathon Project

Wrap Up (5 mins)

Assessment: Optional

[View on Code Studio](#)

Objectives

Students will be able to:

- Translate a paper prototype to screens
- Begin programming an app which uses a database

Preparation

- ☐ Read through the Planning Guide, paying particular focus to the Scoring Guidelines

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals - Presentation**

For the Students

- **CSP U5 Hackathon Project Planning Guide - Activity Guide** [Make a Copy](#)

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (0 mins)

Do This: Move quickly to the activity portion of the lesson.

Activity (40 mins)

Hackathon Project

Step 4: Students work with their partners to fill out the tables on pages 4-6 in the Planning Guides. In these tables, students plan out all of the element IDs, onEvents, functions, variables, conditionals, and loops. This step, will help students smoothly transition into the programming phase.

Step 5 (3 mins): On one computer, partners quickly build their screens, to be used by the programmer to test their code. Emphasize that these are NOT the final screens. Students drag out elements to the screen and give them the agreed upon names from Step 4. The design does not matter here.

Do This: Partners select their primary roles.

- **Programmer:** Responsible for the majority of the programming. Needs to communicate decisions with the designer.
- **Designer:** Responsible for the design of the app. Pair programs with the programmer as needed.

Step 6: Students work on building their apps, guided by their roles.

- **Programmers:** Use the table to guide you in adding programming statements to your project.
- **Designers:** Use the chart to guide you in adding screen elements to your program. You can work on a separate computer from your partner.

Teaching Tip

If the designer finishes their screens early, they can start pair programming with the programmer.

Do This: When the designer is ready to share their screens with the programmer, there are specific steps that must be followed after the designer clicks share and copies the link to their app. The programmer:

1. Add a blank screen.
2. Delete the old project screens.
3. Click the screen dropdown, then click "Import screen"
4. Paste in the link from the designer.
5. Select to import all of the screens.
6. Delete the blank screen.
7. Set the home screen to be the default screen (Hint: Go to design mode and click on the screen)

Wrap Up (5 mins)

Do This: Take some time to answer student questions.

Assessment: Optional

Planning Guide: As an optional completion assessment, you can look at the table in Step 4 of the Planning Guide.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

► **AP** - Algorithms & Programming



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

English ▼

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

Lesson 15: Project - Hackathon Part 3

Overview

This is the third day of a five-day unit project. Students continue to build their apps.

Purpose

Students will demonstrate their app design and programming skills throughout this five day project. In addition, students work with a dataset as this ensures students will be using the types of programming constructs required for the Create Performance Task, which this project is designed as a practice for. Students complete the project by individually filling out a Written Response, modeled after the Create PT.

This project can be used as a unit project, or as an end cap to the first semester of the course.

Agenda

Lesson Modifications

Warm Up (0 mins)

Activity (40 mins)

Hackathon Project

Wrap Up (5 mins)

Assessment: Optional

[View on Code Studio](#)

Objectives

Students will be able to:

- Translate a paper prototype to screens
- Continue programming an app which uses a database

Preparation

- ☐ Read through the Planning Guide
 - ☐ Pay particular focus to the Scoring Guidelines

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals - Presentation**

For the Students

- **CSP U5 Hackathon Project Planning Guide**
- Activity Guide [Make a Copy](#)

Teaching Guide

Lesson Modifications




Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (0 mins)

Do This: Move quickly to the activity portion of the lesson.

Activity (40 mins)

Hackathon Project

 **Step 6 (continued):** Students work on building their apps, guided by their roles.

- **Programmers:** Use the table to guide you in adding programming statements to your project.
- **Designers:** Use the chart to guide you in adding screen elements to your program. You can work on a separate computer from your partner.

Do This: When the designer is ready to share their screens with the programmer, there are specific steps that must be followed after the designer clicks share and copies the link to their app. The programmer:

1. Add a blank screen.
2. Delete the old project screens.
3. Click the screen dropdown, then click "Import screen"
4. Paste in the link from the designer.
5. Select to import all of the screens.
6. Delete the blank screen.
7. Set the home screen to be the default screen (Hint: Go to design mode and click on the screen)

💡 Teaching Tip

Encourage students to check the rubric on the last page of the Planning Guide as they work through building their app. The Written Response portion will be completed on the last day of the project.

💡 Teaching Tip

If the designer finishes their screens early, they can start pair programming with the programmer.

Wrap Up (5 mins)

Do This: Take some time to answer student questions.

Assessment: Optional

Planning Guide: As an optional completion assessment, you can look at the table in Step 4 of the Planning Guide.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

► **AP** - Algorithms & Programming



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

English ▼

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

Lesson 16: Project - Hackathon Part 4

Overview

This is the fourth day of a five-day unit project. Students continue to build their apps.

Purpose

Students will demonstrate their app design and programming skills throughout this five day project. In addition, students work with a dataset as this ensures students will be using the types of programming constructs required for the Create Performance Task, which this project is designed as a practice for. Students complete the project by individually filling out a Written Response, modeled after the Create PT.

This project can be used as a unit project, or as an end cap to the first semester of the course.

Agenda

Lesson Modifications

Warm Up (0 mins)

Activity (40 mins)

Hackathon Project

Wrap Up (5 mins)

Assessment: Optional

View on Code Studio

Objectives

Students will be able to:

- Translate a paper prototype to screens
- Continue programming an app which uses a database

Preparation

- ☐ Read through the Planning Guide
 - ☐ Pay particular focus to the Scoring Guidelines

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals - Presentation**

For the Students

- **CSP U5 Hackathon Project Planning Guide** - Activity Guide [Make a Copy](#)

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (0 mins)

Do This: Move quickly to the activity portion of the lesson.

Activity (40 mins)

Hackathon Project

Step 6 (continued): Students work on building their apps, guided by their roles.

- Programmers: Use the table to guide you in adding programming statements to your project.
- 🧑 Designers: Use the chart to guide you in adding screen elements to your program. You can work on a separate computer from your partner.

💡 Teaching Tip

If the designer finishes their screens early, they can start pair programming with the programmer.

Do This: When the designer is ready to share their screens with the programmer, there are specific steps that must be followed after the designer clicks share and copies the link to their app. The programmer:

1. Add a blank screen.
2. Delete the old project screens.
3. Click the screen dropdown, then click "Import screen"
4. Paste in the link from the designer.
5. Select to import all of the screens.
6. Delete the blank screen.
7. Set the home screen to be the default screen (Hint: Go to design mode and click on the screen)

Do This: Take some time around the midpoint of the activity for students to check the Overall Project section of the Scoring Guidelines (final page of the Planning Guide).

Wrap Up (5 mins)

Do This: Remind students that they will be completing the Written Response portion of the project *individually* during the next lesson. Both students need to fully understand how the program works and have access to the final project.

Assessment: Optional

Planning Guide: As an optional completion assessment, you can look at the table in Step 4 of the Planning Guide.

Standards Alignment



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

English ▼

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

Lesson 17: Project - Hackathon Part 5

Overview

This is the final day of a five-day unit project. Students complete a Written Response, individually answering prompts about the project. Students then share their apps during a gallery walk.

Purpose

Students will demonstrate their app design and programming skills throughout this five day project. In addition, students work with a dataset as this ensures students will be using the types of programming constructs required for the Create Performance Task, which this project is designed as a practice for. Students complete the project by individually filling out a Written Response, modeled after the Create PT.

This project can be used as a unit project, or as an end cap to the first semester of the course.

Agenda

Lesson Modifications

Warm Up (0 mins)

Activity (40 mins)

Written Response

Wrap Up (5 mins)

Assessment: Grading the Project

View on Code Studio

Objectives

Students will be able to:

- Complete a Written Response modeled after the Create PT
- Complete the Hackathon Project app

Preparation

- ☐ Read through the Planning Guide paying particular focus to the Scoring Guidelines
- ☐ Review the **Example Written Response Submission** and **Example Project App Submission** (in the blue teacher panel)

Links

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

For the Teachers

- **CSP U5 Hackathon Submission on AP Classroom** - Resource [Make a Copy](#)
- **CSP Unit 5 - Lists Loops, and Traversals** - Presentation

For the Students

- **CSP U5 Hackathon Project Planning Guide** - Activity Guide [Make a Copy](#)
- **CSP U5 Hackathon Project Written Response** - Written Response [Make a Copy](#)

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Warm Up (0 mins)

Do This: Move quickly to the activity portion of the lesson.

Activity (40 mins)

Written Response

💡 **Note:** The Written Response is to be completed individually. From this point on, partners should not discuss anything about their project.

📖 **Do This:** All students individually navigate to their projects on one tab or browser window. In another tab, students open the Written Response.

🎤 **Remarks**

📖 💡 On your Written Response, you will need to insert screenshots of code segments. What's a code segment? It's a collection of program statements that are part of a program. Together, they demonstrate some working part of your program. You can use screenshots of text or blocks for your code segments.

📖 **Do This (30 mins):** Students work silently to complete the Written Response. Students will need to take screenshots of code and insert those screenshots into the appropriate boxes in the Written Response. Students can check the Scoring Guidelines in the Planning Guide to make sure they are meeting all the requirements.

📖 **Submit:** Students submit their project (one per group), Planning Guide (one per group), and Written Response (individual).

📖 **Share:** Now students display their projects on a computer and complete a gallery walk. If you decide to have a winner of the hackathon, students can vote on their favorite projects by writing down the name of the project on a sticky note and passing it in.

💡 **Teaching Tip**

The Written Response portion of this project is a modified, short version of the Create Performance Task students will turn in to the College Board for the AP Exam. To have your students practice submitting this project using a similar interface to the Create Performance Task, see the **CSP U5 Hackathon Submission on AP Classroom - Resource** for instructions on how to set up an assessment on AP Classroom.

💡 **Teaching Tip**

Programmers should share the project link with the designers. Designers open the project, then click "View Code", which is located on the top right of the screen.

💡 **Teaching Tip**

Students may struggle with the prompt on managing complexity. Encourage students to think how the program would be written differently if no lists could be used. In most cases, this would involve a large amount of variables. Students need to be specific and explain what these variables would store.

Wrap Up (5 mins)

🎤 **Remarks**

💡 Great work everyone on the Hackathon Project! It's amazing how much you were able to accomplish in a short amount of time.

And now, the moment that everyone has been waiting for. I have tallied up the votes and the winner is...

💡 Teaching Tip

Feel free to skip the announcement of a hackathon winner if you want to keep the project non-competitive.

Assessment: Grading the Project

Project: Use the Scoring Guidelines on the final page of the Planning Guide to assess student projects. The Written Response portion is individual, while the Overall Project portion is a group grade.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

► **AP** - Algorithms & Programming

CSP2021

► **CRD-2** - Developers create and innovate using an iterative design process



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

English ▼

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

Lesson 18: Assessment Day

Overview

Students complete a multiple choice assessment which covers the unit topics.

Agenda

Lesson Modifications

Assessment (25 mins)

Topic Coverage

Assessment Review (20 mins)

[View on Code Studio](#)

Preparation

- ☐ Preview the assessment questions

Links

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

For the Teachers

- **CSP Unit 5 - Lists Loops, and Traversals - Presentation**

Teaching Guide

Lesson Modifications



Attention, teachers! If you are teaching virtually or in a socially-distanced classroom, please read the full lesson plan below, then click [here](#) to access the modifications.

Assessment (25 mins)

📋🔗 Administer the Unit 5 Assessment, found on Code Studio. Make sure to unlock the assessment following instructions [here](#).

Assessment Review (20 mins)

Review the answers to the assessment with the class. Discuss any questions that come up and take note of topics where students may need extra review.

💡 Teaching Tip

Topic Coverage

The College Board has provided a bank of questions to help formatively assess student understanding of the content in the framework. These questions are mapped to topics with each topic having a handful of questions available.

The College Board has a few strict guidelines about how topic questions can be used. In particular, students may not receive a grade based on performance on topic questions nor can they be used for teacher evaluation. Beyond these requirements, however, they are primarily intended to formatively assess student progress and learning as they prepare for the end of course exam.

Within our own course we recommend that you use them in a variety of ways:

- Throughout the unit assign topic questions to students related to the topics students are learning about that day or that week
- Prior to the unit assessment assign topic questions to help students practice and prepare for the summative assessment
- After the unit assessment use these topic questions to help students track their progress towards preparation for the AP assessment

Unit 5: Lists, Loops, and Traversals	Topic 3.2 Data Abstraction
	Topic 3.4 Strings
	Topic 3.8 Iteration
	Topic 3.10 Lists
	Topic 3.16 Simulations

Click for more info: [Code.org CSP Topic Coverage](#)



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

English ▼

If you are interested in licensing Code.org materials for commercial purposes, [contact us](#).