

LESSON: Traversing List Program		Time: 50 minutes
<p>Overview:</p> <p>An important concept when working with lists is the ability to traverse the list. In the previous lesson, students worked on constructing for loops to traverse a list. In this lesson, students will write a program that traverses a list. They will use a specialized for loop, and then change it to a for loop. They will also practice using a parameter in a if statement.</p>		<p>Objectives:</p> <ul style="list-style-type: none"> • I can define “traverse” • I can use a specialized for loop to traverse a list • I can use a for loop to traverse a list • I can use a parameter in an if statement
<p>Standards:</p> <p>2-CS-03 Systematically identify and fix problems with computing devices and their components.</p> <p>2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.</p> <p>3A-AP-17 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.</p>	<p>CSP Framework:</p> <p>Computational Thinking Practices:</p> <p>2.B Implement and apply an algorithm.</p> <p>3.B Use abstraction to manage complexity in a program.</p> <p>3.C Explain how abstraction manages complexity.</p> <p>4.C Identify and correct errors in algorithms and programs, including error discovery through testing.</p>	<p>Key Concepts:</p> <ul style="list-style-type: none"> • Traversing a list is a common activity in coding that uses a list. • Traversing a list uses iteration – either a while loop or a for loop. • Traversing a list means each element or item in the list will be accessed.
<p>Preparation:</p> <p>Make a copy of the assignment or put it in the LMS</p> <p>Prepare any formative assessments you want to use in the wrap-up</p>	<p>Links:</p> <ul style="list-style-type: none"> • Assignment • Instructions slide deck • Solution - part 1 • Solution - part 2 • Solution - part 3 w/ challenge <p>Resources:</p> <ul style="list-style-type: none"> • Spicy Challenge solution 	<p>Agenda:</p> <ul style="list-style-type: none"> • Warm-up / list review (5 minutes) • Programming parts 1-4 (40 minutes) • Challenges (15 minutes – probably won’t have time but they are included just in case) • Wrap-up & Assessment (5 minutes)
<p>Vocabulary: (review from previous lesson)</p> <ul style="list-style-type: none"> • Traverse: traveling or traversing through a list one element at a time, in order, starting with index 0 (first element) and going through to the last element (index len-1) • Review sequential, selection, and iteration from previous lessons (Mission 3, 4, and 6) 		
<p>Assessment:</p> <ul style="list-style-type: none"> • Daily reflection journal or Google form • Assignment completion • Programming Journal • Write a program that works correctly without errors 		

Teaching Guide

Warm-up / Traversing a List (5 minutes)

This short warm-up is to review traversing a list. Warm-up questions are included on the assignment and slides, but they can be modified to any warm up questions you feel are important to review. You can have students work individually or in groups. You may want to have a class discussion before starting the program.

💡 Teaching tip – warm-up

- Go through slides #2-5
- Ask the students what they remember about traversing a list. You can have students do a share-out or think-pair-share.
- Ask the students how to traverse a list (questions on the slide and assignment) You can have students do a share-out or think-pair-share.
- A sample of each type of loop is included on slide 5

Traversing a list Program (30-40 minutes)

🖥️ Students will work in pairs, doing pair programming.

💡 Teaching tip - Part 1: Slides 7-12 (10 minutes)

Students will use code from Practice #1 so it won't take a lot of time to get started. They will use the first list from this program. Instructions are included to copy and paste from Practice #1 to their new Traversals program.

After students have copied and pasted code from Practice #1 to Traversals, they will make changes. They will change the introduction and make it shorter.

Then they will add a function for `slideshow()` (similar to `display_info()`) that will traverse the list. Sample code is given, but students must put in their own code.

They need to fill in the parts circled in red:

Run the code and make sure it works before going to Part 2.

```
def slideshow():  
    for {.....}:  
        display.clear()  
        display.print({...})  
        sleep(1)  
    display.print("End of list")
```

💡 Teaching tip - Part 2: Slides 13-16 (10 minutes)

Students will add another list to their code (from Practice #1). Therefore, they will need to change the specialized for loop to a for loop and use the index to reference items from both lists. They practiced this in the previous lesson. Refer to it if needed.

The code for this is not given to them. However, if they need a hint, the code is shown on slide 20.

Run the code and make sure it works before going to Part 3.


💡 Teaching tip - Part 3: Slides 17-24 (10-15 minutes)

Similar to Practice #1, students will add two more lists and incorporate them in the program. They will add a parameter to `slideshow()` and use it in an if statement to assign values (lists) to two local variables. Code is not given for this, but students are referred to Practice #1, since it is very similar.

 **Teaching tip - Part 3 challenge: Slides 25 (5 minutes)**

This is the same challenge that was given in Practice #1. Students create another local variable for the team (or whatever their topic is) and display it with the data from the list. Some sample code is given to help them out, but they can also refer to Practice #1. The only thing left to do is to add another print statement in the for loop that displays the new local variable. If they didn't do this for Practice #1, they will get a chance to do it in the next lesson.

Challenges (10-15 minutes) Probably won't have time in a regular 50-minute class period


 The challenges are meant for students who work extremely quickly, or if you have a block schedule and more time, or as something you can return to later when time permits as a great review or extension.


 **Teaching tip:**

Two challenges are given, with varying levels of difficulty. Students can pick the challenge they want. Also, depending on time, students can complete more than one challenge. (slides 25-26)

Very little direction is given for the challenges. Students are expected to work on the code without tips.

- **Medium Challenge:** Open a program they did earlier that used lists and add or modify code to traverse the list
 - This will be similar to the activities in the “traversing a list” assignment.
 - Students can add code to another button to run sequentially through the list without pressing a button (slideshow)
 - Mission 6, Mission 7, or Remix #2 work well for this challenge
- **Spicy Challenge:** Students will modify the Game Spinner (Mission 9) to traverse the ARROWS list. This can be a little tricky because the list is traversed inside another loop. Solution is included in the folder.
 - **Suggestions:**
 - Use your own arrows list
 - Adjust loop count to complete traversals instead of overall arrows
 - Comments about this are included in the solution

 You decide what you want students to turn in for a grade, and how they turn it in. You can require work from all three missions, or just the last one. You can look at the code on student computers, or have them submit code. The assignment document is a review and can be turned in.

 If students used the CodeX, they should clear the CodeX at the end of the class period.

Wrap-Up (5 minutes)

The wrap-up can be very short for this lesson (slides 29-30). This is the class's chance to review traversing a list and write meaningful notes. They are basically the same questions as the warm up to today's assignment and the wrap-up from the previous lesson.

Formative Assessment:

- Daily reflection journal or Google form



- Class discussion on what they learned about traversing a list
- Assignment completion
- Programming journal
- Wrap-up questions
- Exit ticket

SUCCESS CRITERIA:

- Define “traversing a list”
- Traverse a list using a simplified for loop
- Traverse a list using a for loop
- Use a parameter and if statement to assign lists to local variables
- Use if statements to set a state, or topic, variable and use it as an argument in a function call