Unit 5: Lists, Loops, and Traversals

Unit 5 - Lesson 1
Lists Explore

Warm Up

● ○ ○

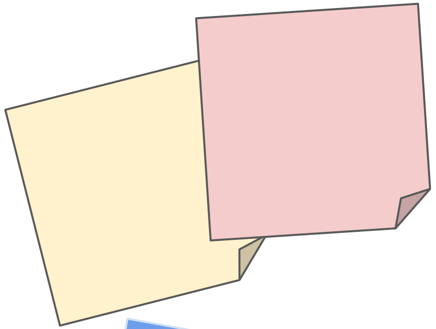
Prompt:

With a partner, brainstorm lists of information that you encounter on a daily basis.

Why are these lists useful?

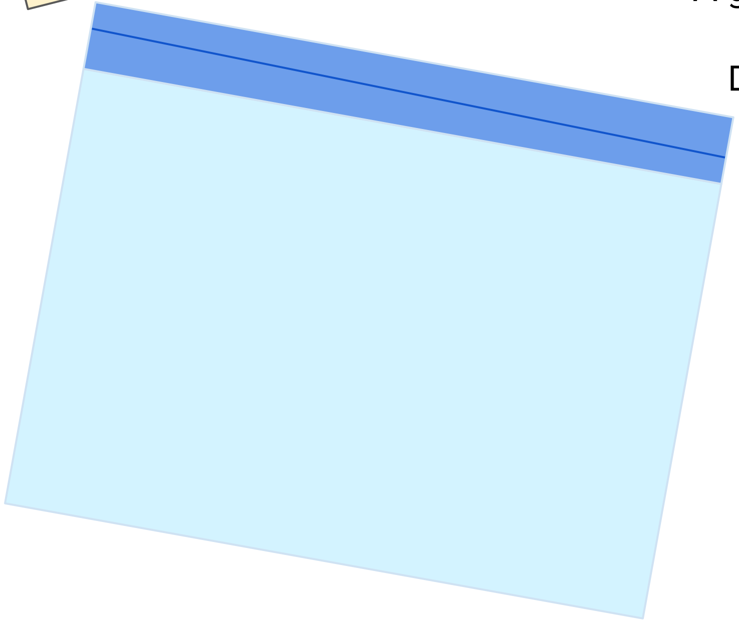
Activity

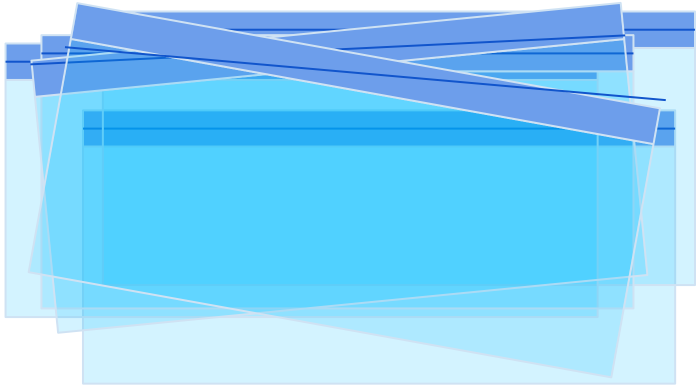
●●○

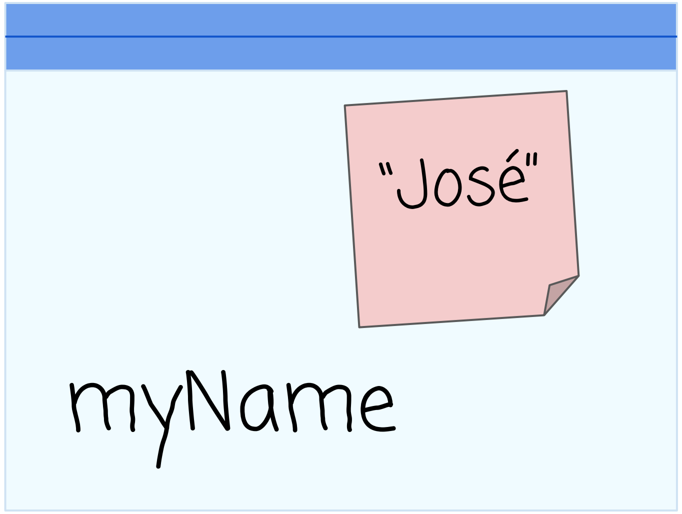


Activity Title

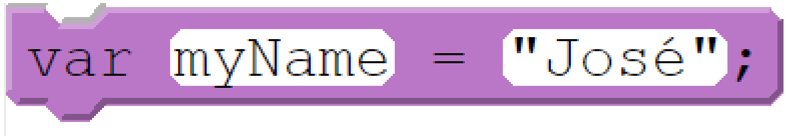
You and your partner should have:
Small stacks of red and yellow stickies
A small stack of snack baggies
A gallon sized baggie
Pen / Pencil
Dry erase marker
Tape or Stapler







```
var myName = “José”;
```



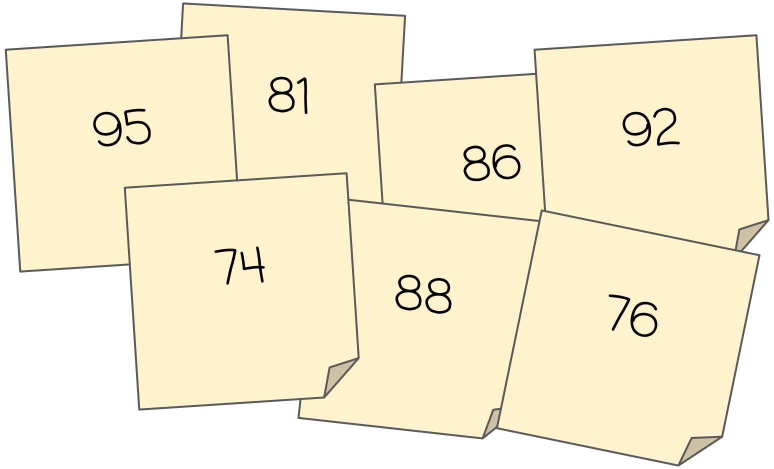
```
var myName = "José";
```

But what if we have a lot of information?

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.

"Monday"
"Wednesday"
"Friday"
"Sunday"
"Tuesd
"Thursday"
"Saturday"



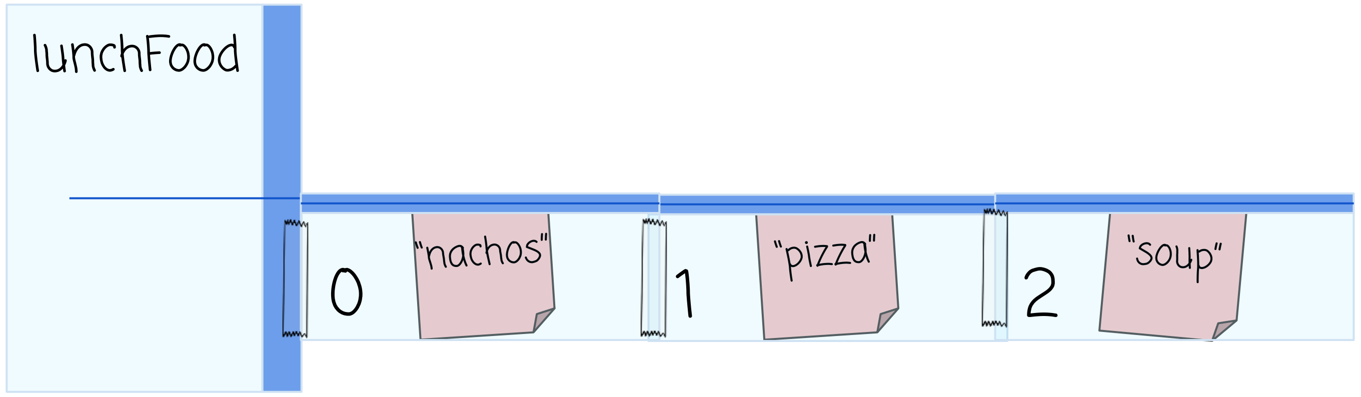
Creating and Accessing Lists

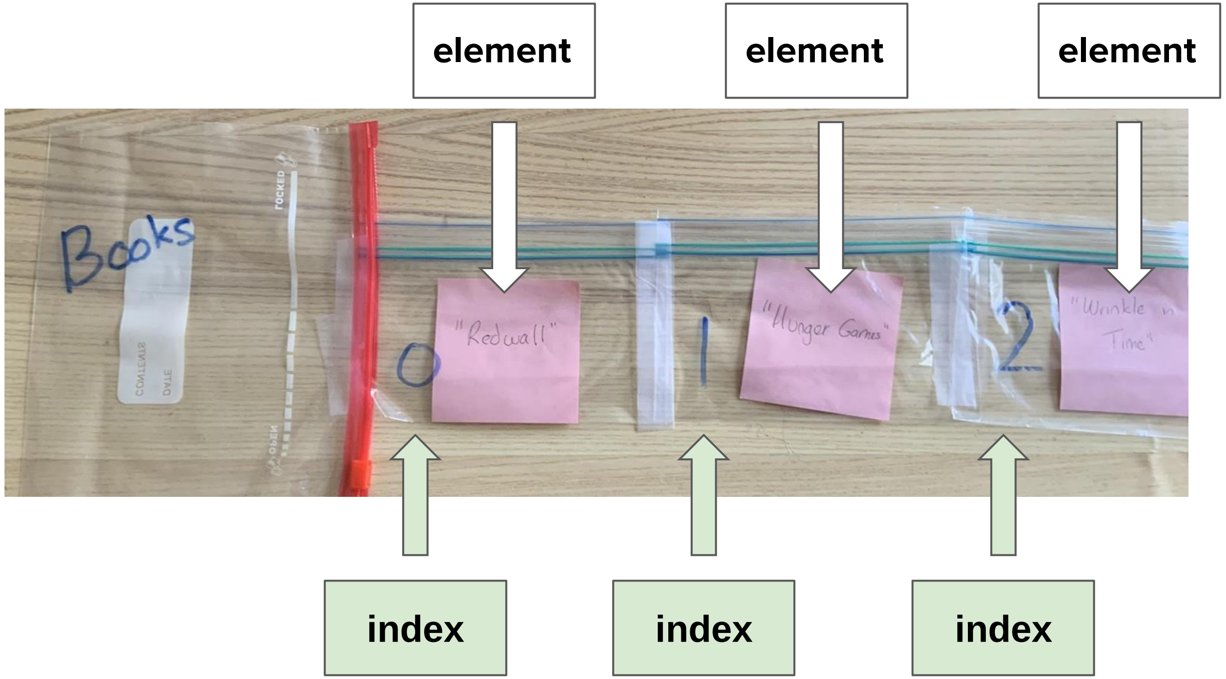
Lists

- Gallon baggie connected to smaller baggies
- Each smaller baggie can hold one value
- Gallon baggies are named with the same rules as a variable (no spaces, can't start with number)
- Smaller baggies are numbered starting at 0

Do This:

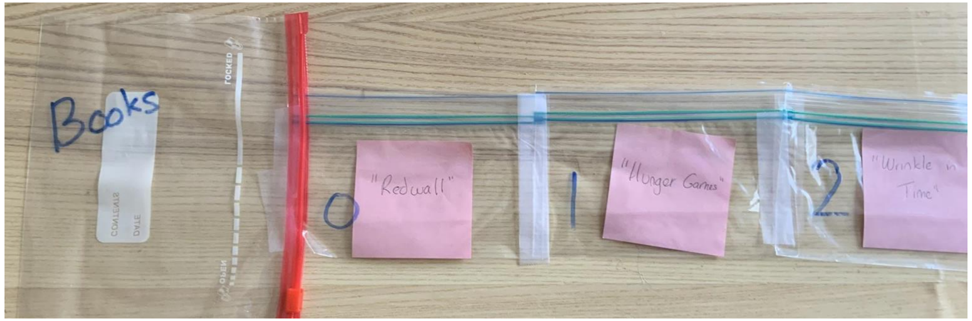
Make a list like the one below that holds between 2 and 4 values. Share it with another group.

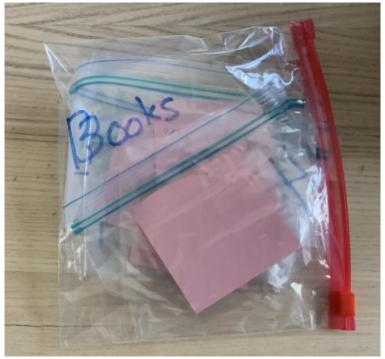




Your list is made up of **elements.** Each element has its own **index**. Indexes start at 0 and count up. The **length** of the list is how many elements it contains. This list has 3 elements and indexes from 0 to 2.

Unit 5 Lesson 1 - Activity



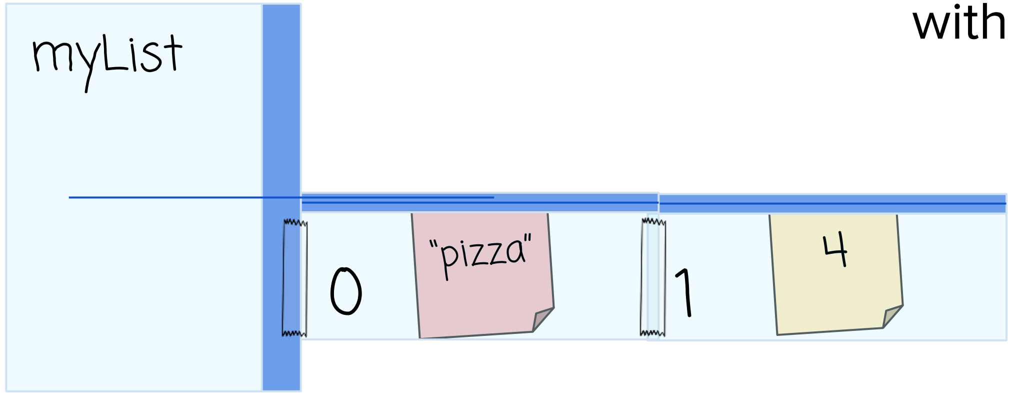


Notice that all the bags can be folded up and be placed inside the big variable baggie. Sometimes we want to think about the whole list, sometimes we want to think about individual elements

```
var myList = ["pizza", 4]
```

- This command creates a new list and assigns it to a variable
- A list is indicated with square brackets
- Each value in the list is separated by commas

Do This:
Write out the command that creates the list you just made.
Then write the command that makes another group's list.
Share both answers with that group.



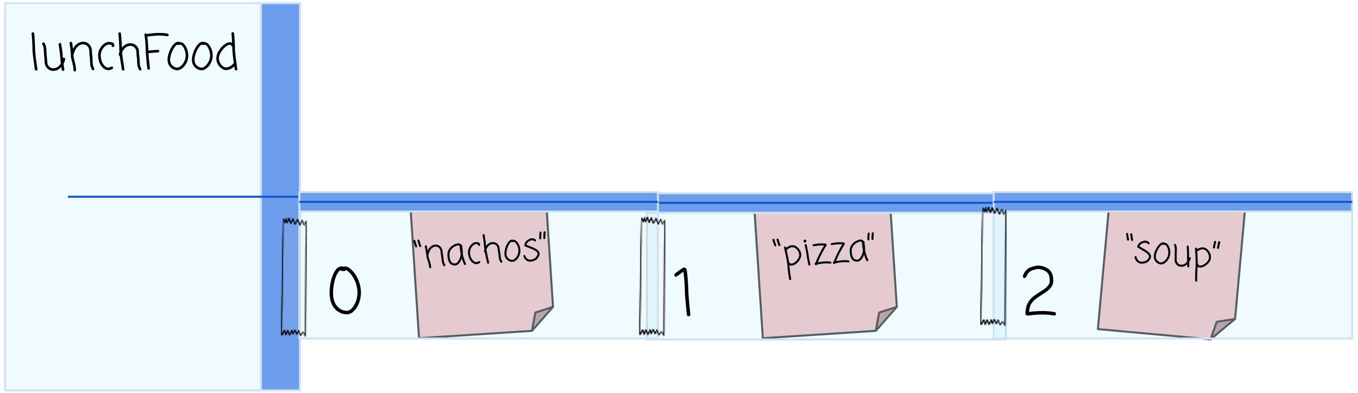
`myList[1]`

- This expression “accesses” the value at that index of the list.
- Also uses square brackets

Do This:

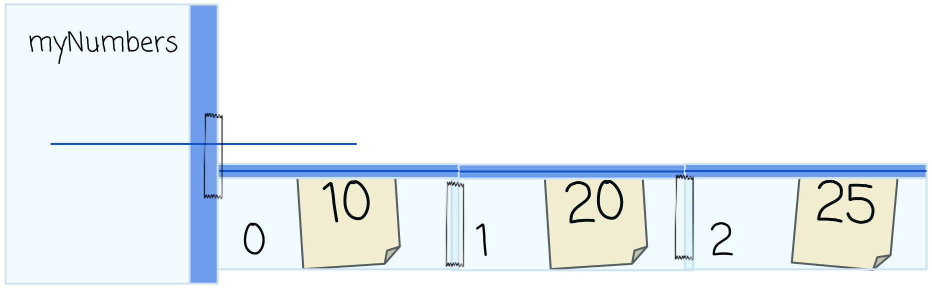
What do lunchFood[0] and lunchFood[2] access?

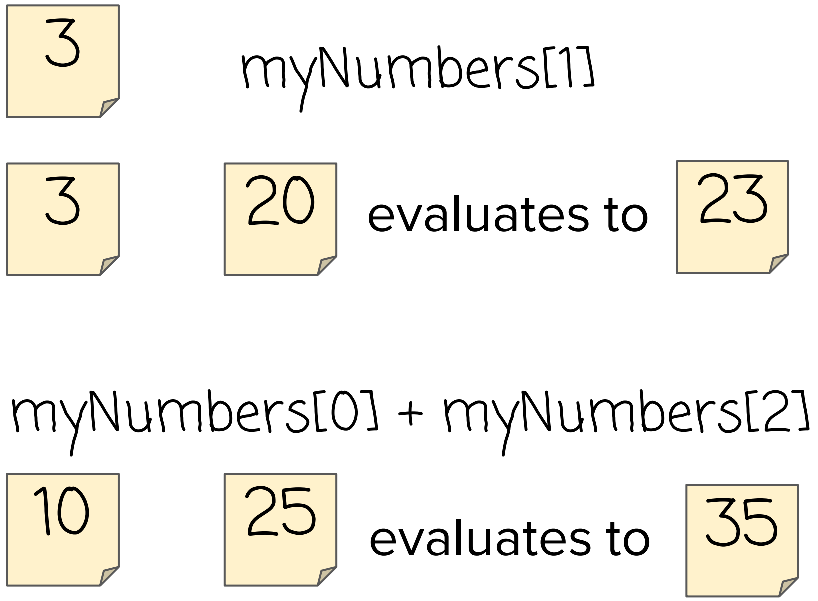
Compare your answers with another group.



Lists and Expressions

Replace the list access with a copy of the value it holds
Evaluate the expression as normal



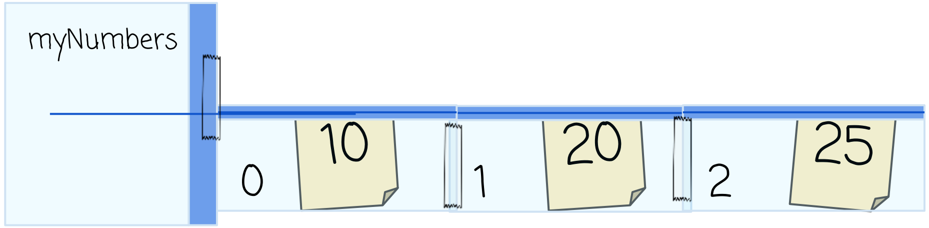


Do This: Write three expression that include accesses to the list you created. Use the examples above for inspiration
Trade with another group and have them evaluate your expressions

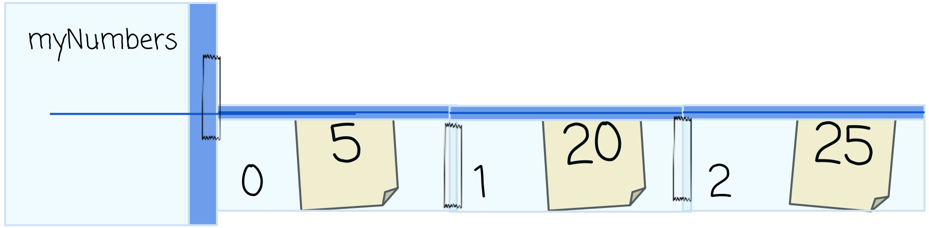
myList[1] = “hello”

Assigns the value on the right to the index
Just like variable assignment, the old value is thrown away and replaced.

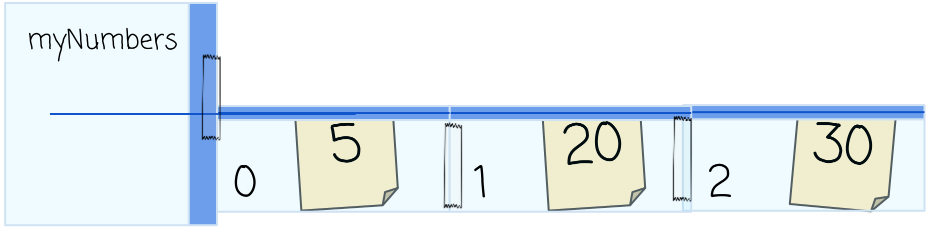
```
00 var myNumbers = [10,20,25];
```



```
01 myNumbers[0] = 5;
```



```
02 myNumbers[2] = 30;
```

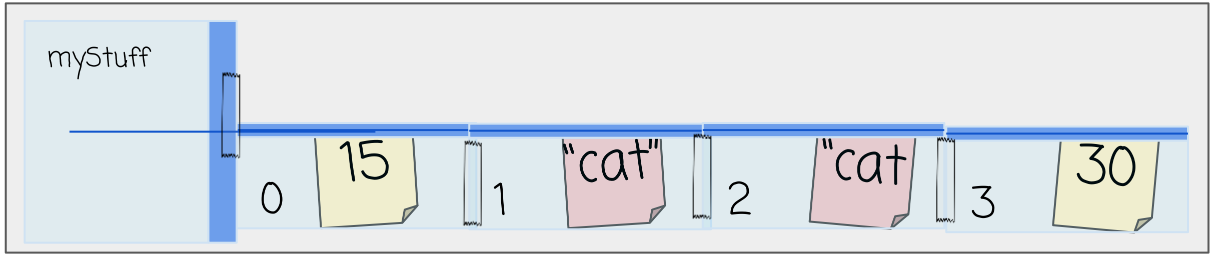


Do This: Discuss what the list will contain after line 02 runs.

Do This:

Run this program. Compare your result with another group.

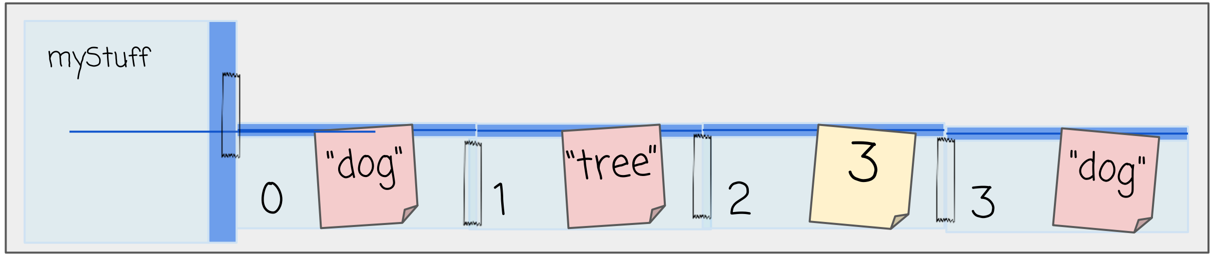
```
var myStuff = [20, "hat", "pow", 5];
myStuff[1] = "cat";
myStuff[2] = myStuff[1];
myStuff[0] = myStuff[3] + 10;
myStuff[3] = myStuff[0] + myStuff[0];
```



Do This:

You can use expressions in the place of the list index.
Run this program and compare your result with another group.

```
var myStuff = [“dog”,”cat”,3,10];
myStuff[2-1] = “tree”;
myStuff[myStuff[2]] = myStuff[0];
```

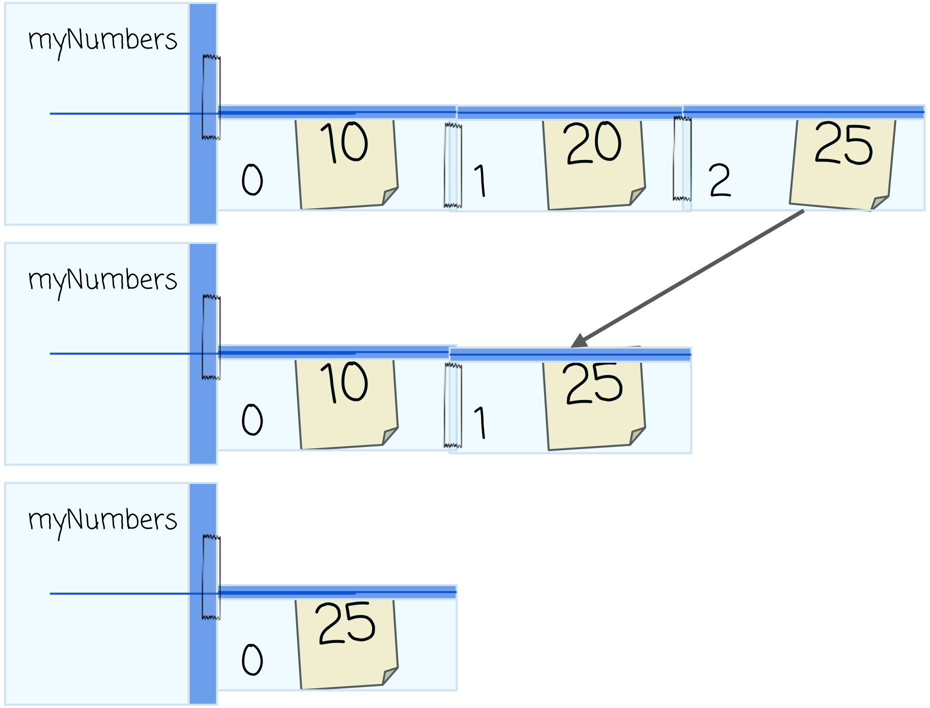


Changing Your List

`removeItem(list, index)`

Removes the element in the given list at the given index
All items to the right are shifted over
The last index is removed from the list

```
00 var myNumbers = [10,20,25];

01 removeItem(myNumbers,1);

02 removeItem(myNumbers,0);
```

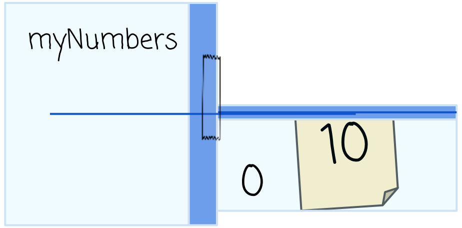


Do This: Discuss what the list will contain after line 02 runs.

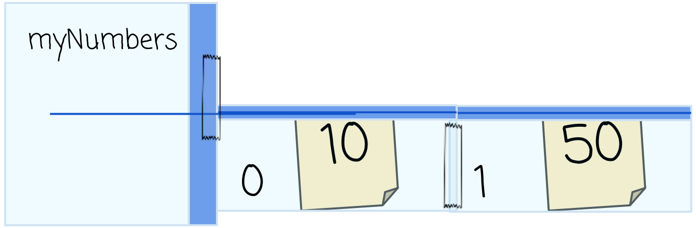
`appendItem(list, item)`

Adds an element to the end of the list
A new index is added to the list to create a place for the element
The new item is placed in this new index

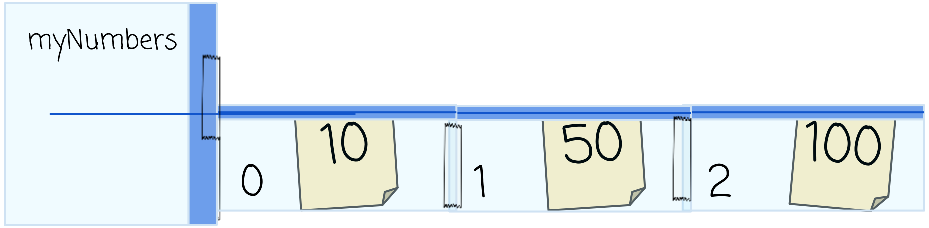
```
00 var myNumbers = [10];
```



```
01 appendItem(myNumbers,50);
```



```
02 appendItem(myNumbers,100);
```

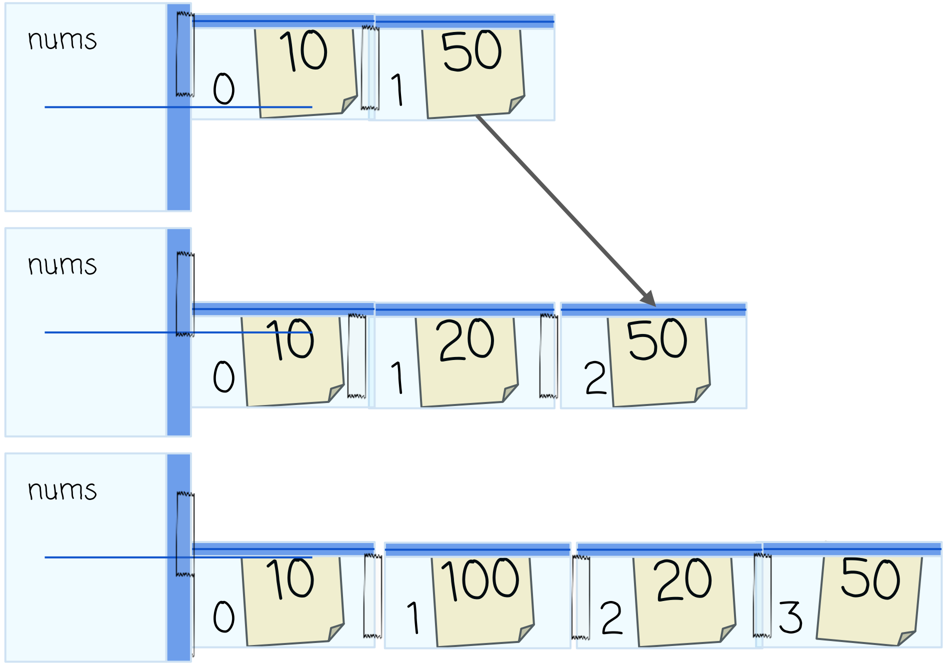


Do This: Discuss what the list will contain after line 02 runs.

`insertItem(list, index, item)`

Inserts an element into a list at the index given
A new index is added to the list so there's space for the new element
The new element is placed at the index given, all other items move right

```
00 var nums = [10,50];

01 insertItem(nums,1,20);

02 insertItem(nums,1,100);
```



Do This: Discuss what the list will contain after line 02 runs.

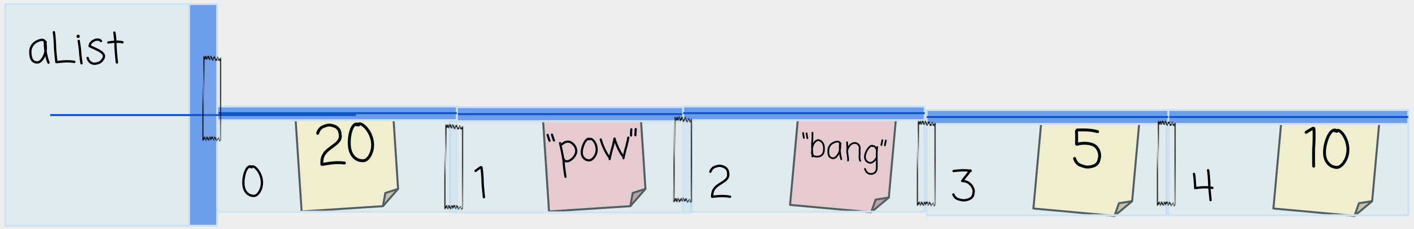
Do This:

Run this program. Compare your result with another group.
What is the length of the list at the end?

Command Reference

```
removeItem(list, index)
appendItem(list, item)
insertItem(list, index, item)
```

```
var aList = [20, “hat”, “pow”];
appendItem(aList, 5);
appendItem(aList, 10);
removeItem(aList, 1);
insertItem(aList, 2, “bang”);
```



The length of the list is 5 (one greater than the last index)

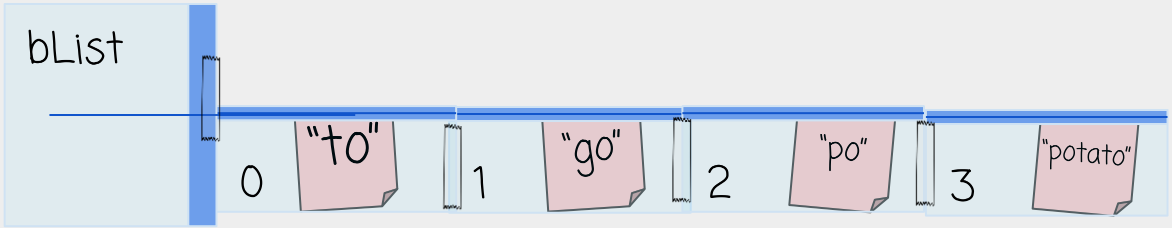
Do This:

Run this program. Compare your result with another group.
What is the length of the list at the end?

Command Reference

```
removeItem(list, index)
appendItem(list, item)
insertItem(list, index, item)
```

```
var bList = [“to”, 5, “po”];
appendItem(bList, bList[2] + “ta” + bList[0]);
insertItem(bList, 2-1, “go”);
removeItem(bList, 2);
```



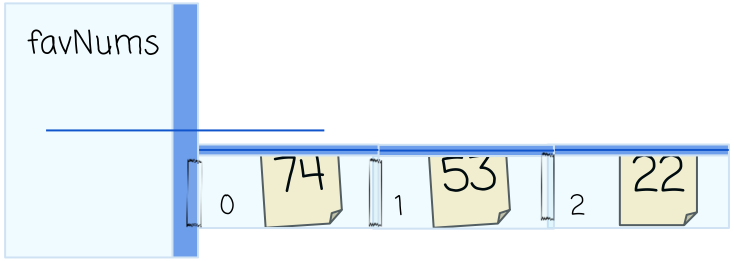
The length of the list is 4 (one greater than the last index)

Wrap Up

Unit 5 Lesson 1 - Wrap Up

Unit 5 Lesson 1 - Wrap Up

Key Takeaways



```
var favNums = [74, 53, 22];
```

- A **List** is an ordered collection of elements
- An **Element** is an individual value in a list that is assigned a unique index
- An **index** a common method for referencing the elements in a list or string using numbers
- The length of a list is how many elements it contains. Lists can grow or shrink as elements are added or removed.
- Lists are an example of data abstraction. They allow us to name and program with large collections of information while ignoring the low level details of how the data is stored, organized, etc. These programs are easier to develop and maintain.

Unit 5 - Lesson 2
Lists Investigate

Warm Up

● ○ ○

Prompts:

What are some similarities and differences between lists and variables?

How does a list manage complexity in a program?

Activity

●●○

Unit 5 Lesson 2 - Activity

Do This:

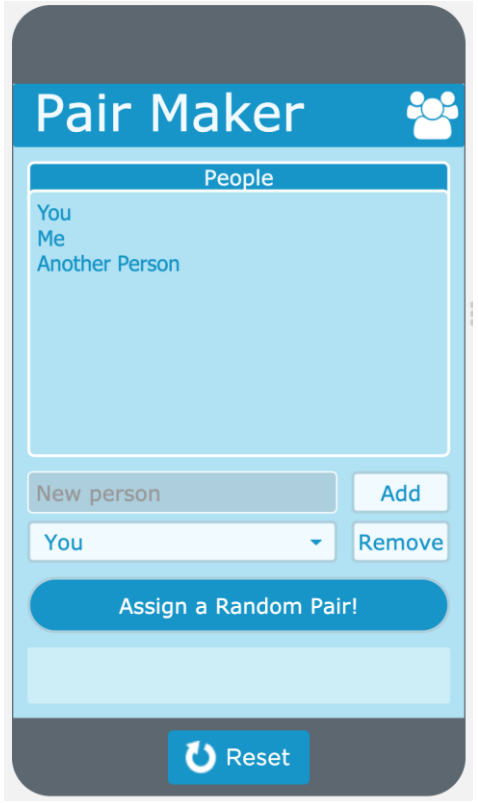
- Find a partner
- Choose either the Outfit Picker or the Band Namer
- Read the program with your partner
- Respond to all the questions for your app
- Be ready to share your responses and what you learned with another group.

Discuss: Spend 3-4 minutes explaining your app to another group. Then switch.

Unit 5 Lesson 2 - Activity

Do This:

- All groups read the Pair Maker app and respond to the questions.
- Be ready to discuss your responses with another group.



Wrap Up

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?

Unit 5 - Lesson 3
Lists Practice

Warm Up

● ○ ○

Vocabulary Review

List -	an ordered collection 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 numbers

Activity

● ● ○

Unit 5 Lesson 3 - Activity

Debugging: the process of finding and fixing problems in code

Describe
The Problem

What do you expect it to do?
What does it actually do?
Does it always happen?

Hunt
For Bugs

Are there warnings or errors?
What did you change most recently?
Explain your code to someone else
Look for code related to the problem

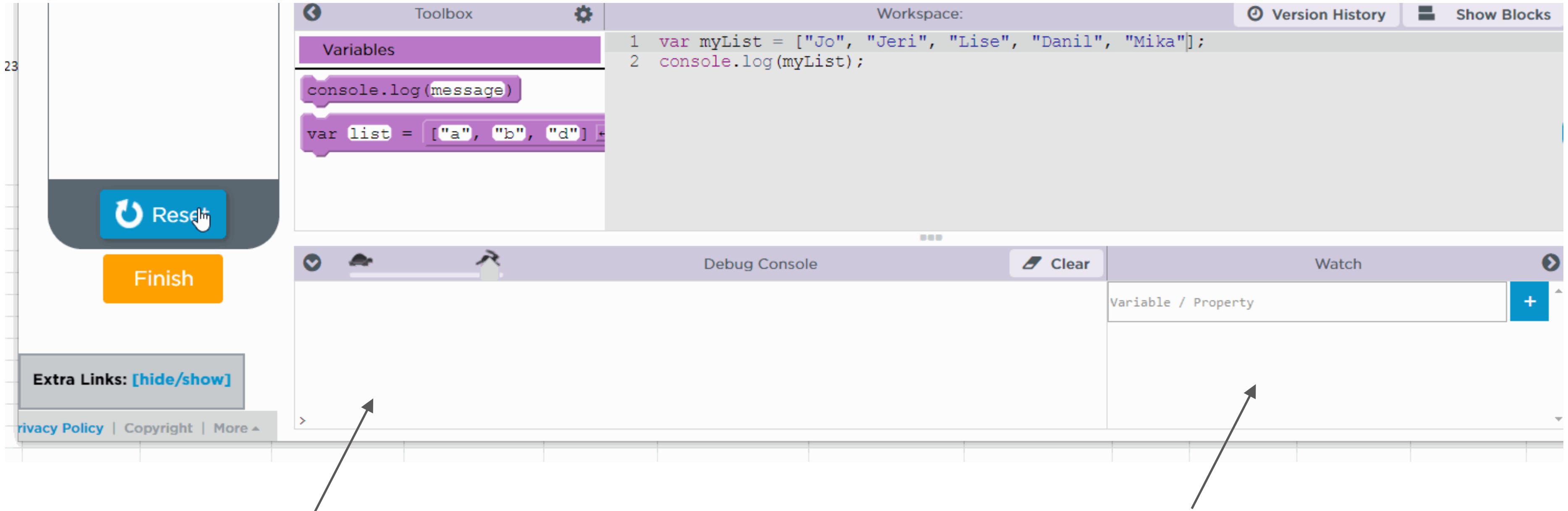
Try
Solutions

Make a small change

Document
As You Go

What have you learned?
What strategies did you use?
What questions do you have?

Debugging Lists



```
var myList = ["Jo", "Jeri", "Lise", "Danil", "Mika"];
console.log(myList);
```

Use `console.log` to track lists in the Debug Console. Click the arrow to see elements listed by index.

Strings have indexes too!

`var myString = “This or that”`

T	h	i	s		o	r		t	h	a	t
0	1	2	3	4	5	6	7	8	9	10	11

`myString.length` will give the length of the string, which is 12

`myString.substring(start,end)` returns a substring starting at `start`, and going up to but not including end

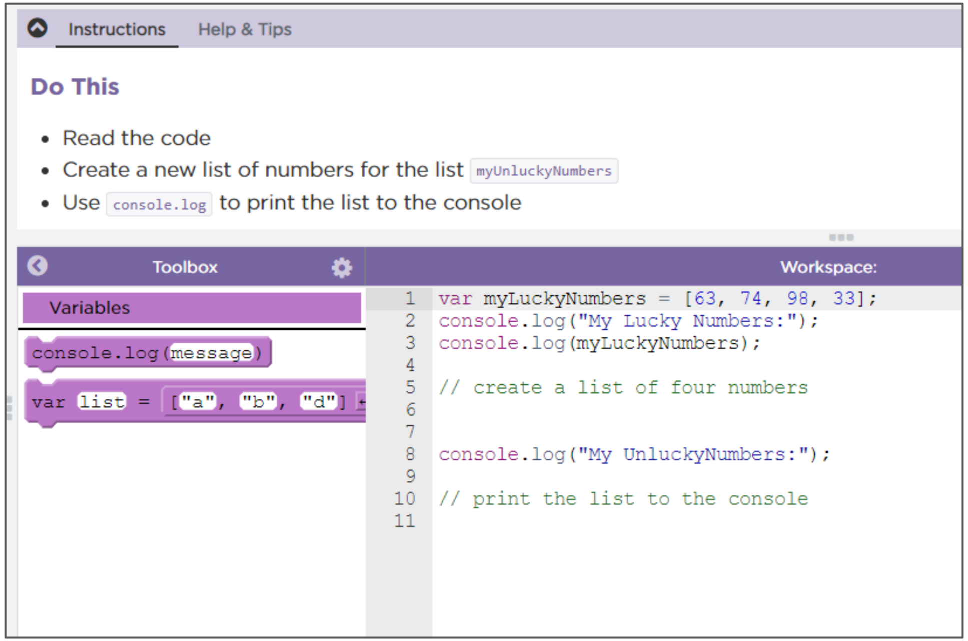
`myString.substring(0,4)` is “`This`”

T	h	i	s		o	r		t	h	a	t
0	1	2	3	4	5	6	7	8	9	10	11

`myString.substring(5,7)` is “`or`”

T	h	i	s		o	r		t	h	a	t
0	1	2	3	4	5	6	7	8	9	10	11

Lists Practice



Do This

- Read the code
- Create a new list of numbers for the list `myUnluckyNumbers`
- Use `console.log` to print the list to the console

```
console.log(message)
var list = ["a", "b", "d"]
```

```
var myLuckyNumbers = [63, 74, 98, 33];
console.log("My Lucky Numbers:");
console.log(myLuckyNumbers);

// create a list of four numbers


console.log("My UnluckyNumbers:");

// print the list to the console

```

Do This:

- Navigate to Lesson 3, Level 2 on Code Studio

Wrap Up

Prompt:

What aspects of working with lists do you feel clicked today?

What do you still have trouble with?

Unit 5 - Lesson 4
Lists Make

Warm Up

● ○ ○

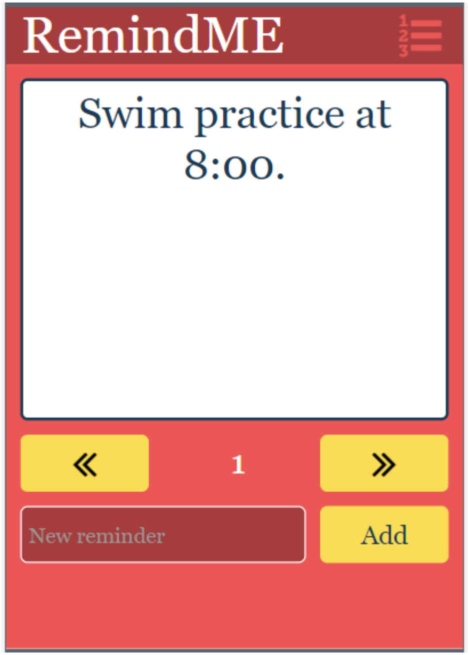
Prompt:

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

Activity

●●○

Lists Make: Reminder App



Do This:

- Navigate to Lesson 4, Level 2 on Code Studio

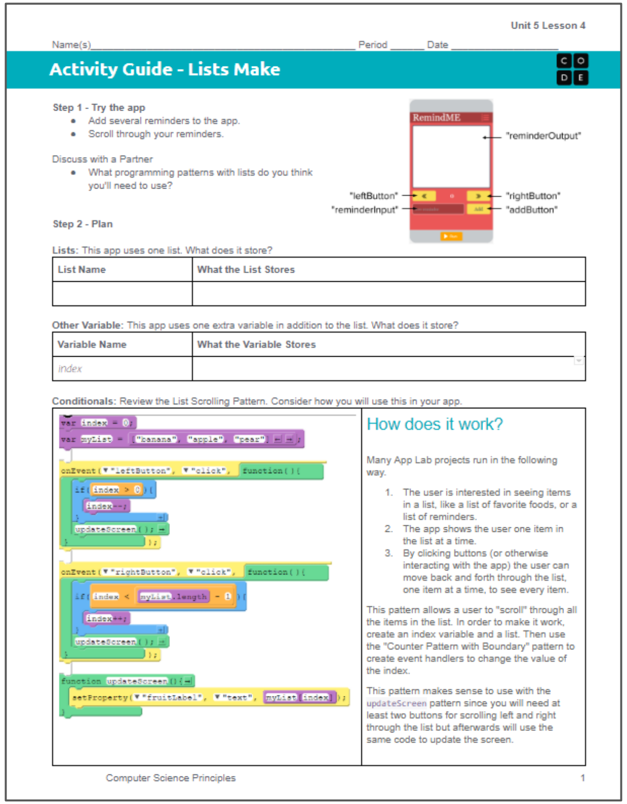
Do This:

- Try using the app.
- Add several reminders to the app.
- Scroll through your reminders.

Prompts:

- 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?

Do This: Make the Reminder App!



Unit 5 Lesson 4

Name(s)_______________________________________ Period ______ Date ________________

Activity Guide - Lists Make

Step 1 - Try the app
- Add several reminders to the app.
- Scroll through your reminders.

Discuss with a Partner
- What programming patterns with lists do you think you'll need to use?

RemindME
"reminderOutput"
"leftButton"
"rightButton"
"reminderInput"
"addButton"

Step 2 - Plan

Lists: This app uses one list. What does it store?

List Name	What the List Stores

Other Variable: This app uses one extra variable in addition to the list. What does it store?

Variable Name	What the Variable Stores
index	

Conditionals: Review the List Scrolling Pattern. Consider how you will use this in your app.

How does it work?

Many App Lab projects run in the following way.

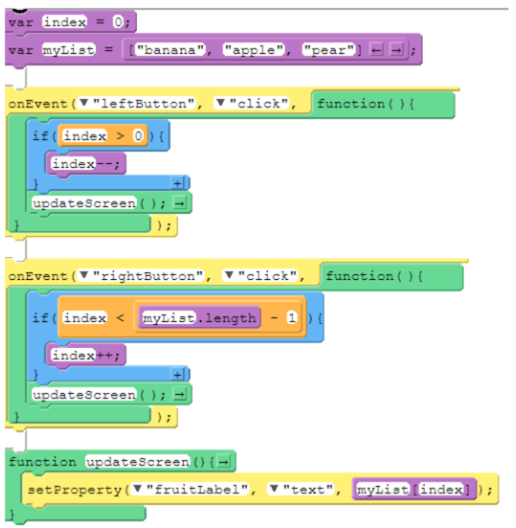
1. The user is interested in seeing items in a list, like a list of favorite foods, or a list of reminders.
2. The app shows the user one item in the list at a time.
3. By clicking buttons (or otherwise interacting with the app) the user can move back and forth through the list, one item at a time, to see every item.

This pattern allows a user to "scroll" through all the items in the list. In order to make it work, create an index variable and a list. Then use the "Counter Pattern with Boundary" pattern to create event handlers to change the value of the index.

This pattern makes sense to use with the `updateScreen` pattern since you will need at least two buttons for scrolling left and right through the list but afterwards will use the same code to update the screen.

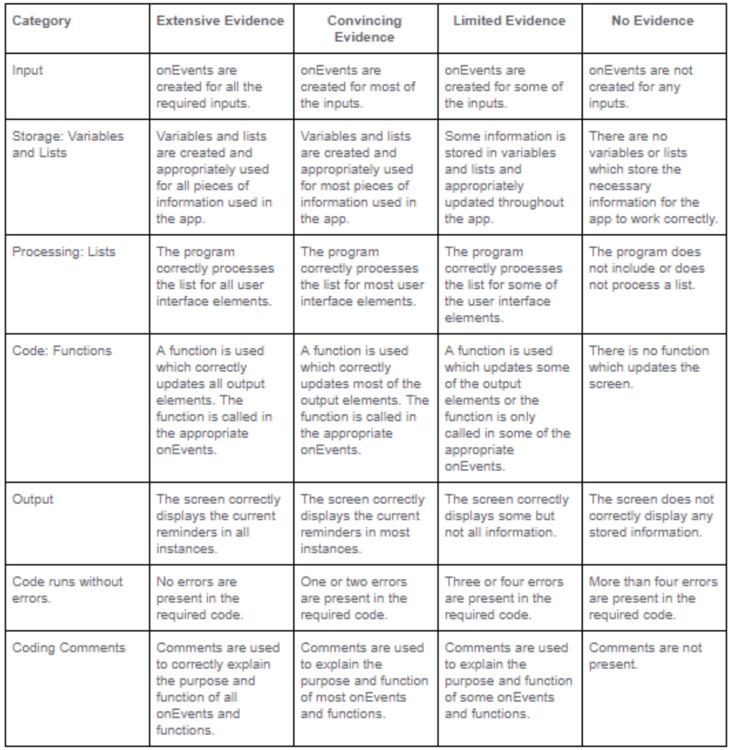
Computer Science Principles 1

The List Scrolling Pattern



```
var index = 0;
var myList = ["banana", "apple", "pear"];

onEvent("leftButton", "click", function(){
  if(index > 0){
    index--;
  }
  updateScreen();
});

onEvent("rightButton", "click", function(){
  if(index < myList.length - 1){
    index++;
  }
  updateScreen();
});

function updateScreen(){
  setProperty("fruitLabel", "text", myList[index]);
}
```

Don't forget to check the rubric before hitting submit!



Category	Extensive Evidence	Convincing Evidence	Limited Evidence	No Evidence
Input	onEvents are created for all the required inputs.	onEvents are created for most of the inputs.	onEvents are created for some of the inputs.	onEvents are not created for any inputs.
Storage: Variables and Lists	Variables and lists are created and appropriately used for all pieces of information used in the app.	Variables and lists are created and appropriately used for most pieces of information used in the app.	Some information is stored in variables and lists and appropriately updated throughout the app.	There are no variables or lists which store the necessary information for the app to work correctly.
Processing: Lists	The program correctly processes the list for all user interface elements.	The program correctly processes the list for most user interface elements.	The program correctly processes the list for some of the user interface elements.	The program does not include or does not process a list.
Code: Functions	A function is used which correctly updates all output elements. The function is called in the appropriate onEvents.	A function is used which correctly updates most of the output elements. The function is called in the appropriate onEvents.	A function is used which updates some of the output elements or the function is only called in some of the appropriate onEvents.	There is no function which updates the screen.
Output	The screen correctly displays the current reminders in all instances.	The screen correctly displays the current reminders in most instances.	The screen correctly displays some but not all information.	The screen does not correctly display any stored information.
Code runs without errors.	No errors are present in the required code.	One or two errors are present in the required code.	Three or four errors are present in the required code.	More than four errors are present in the required code.
Coding Comments	Comments are used to correctly explain the purpose and function of all onEvents and functions.	Comments are used to explain the purpose and function of most onEvents and functions.	Comments are used to explain the purpose and function of some onEvents and functions.	Comments are not present.

Submit

Wrap Up

Unit 5 Lesson 4 - Wrap Up

Great job today!

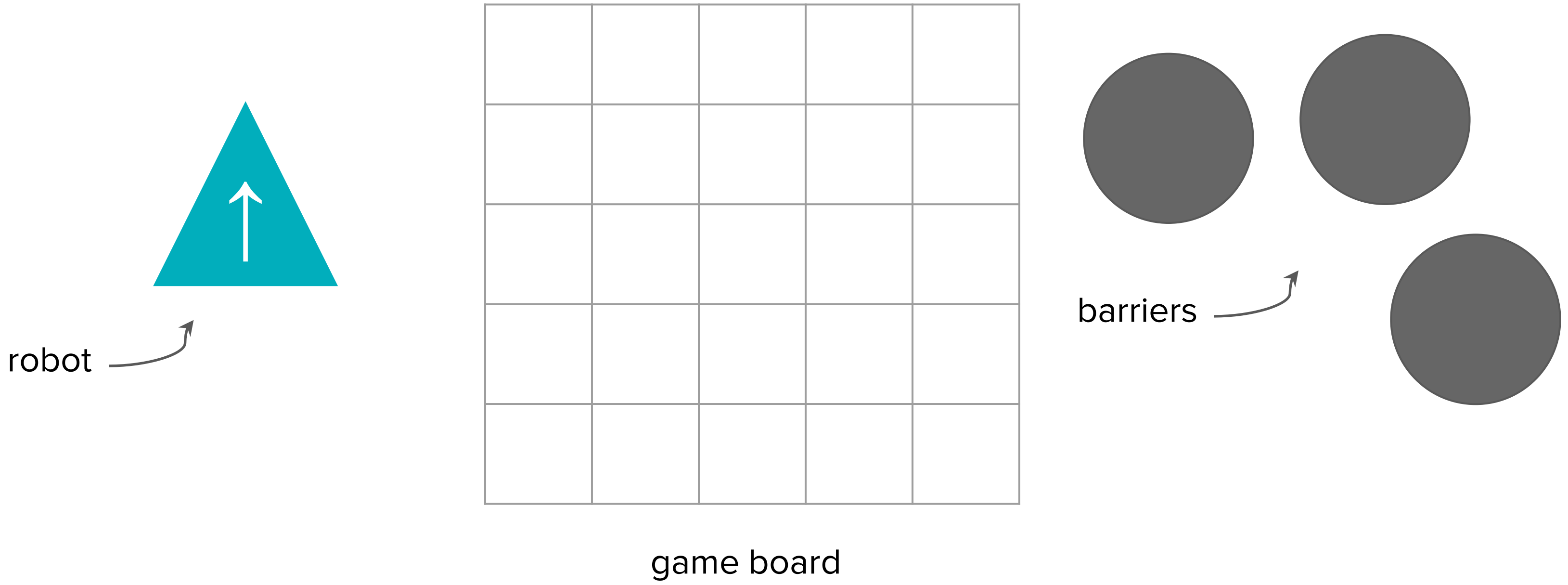
Unit 5 - Lesson 5
Loops Explore

Warm Up

● ○ ○

You and your partner should have:

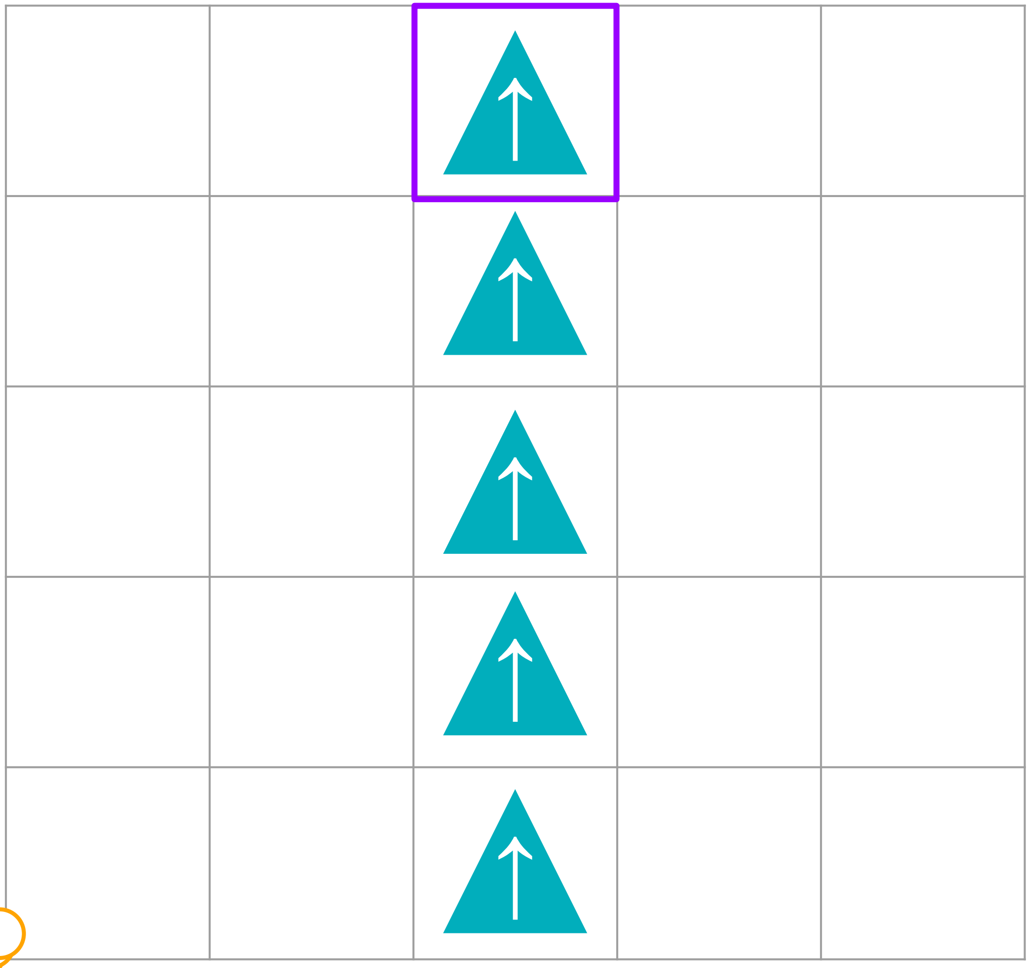
A game board
Game pieces (tokens, markers, etc.) to use as barriers
A “robot”



Activity

● ● ○

Unit 5 Lesson 5 - Activity



Do this:

- You have one command:
- This command moves the robot one space forward in the direction that it is facing
- With a partner, write the program to move the robot to the purple box

```
moveForward()
moveForward()
moveForward()
moveForward()
moveForward()
moveForward()
moveForward()
moveForward()
moveForward()
```

This code is repetitive. How can we simplify it?

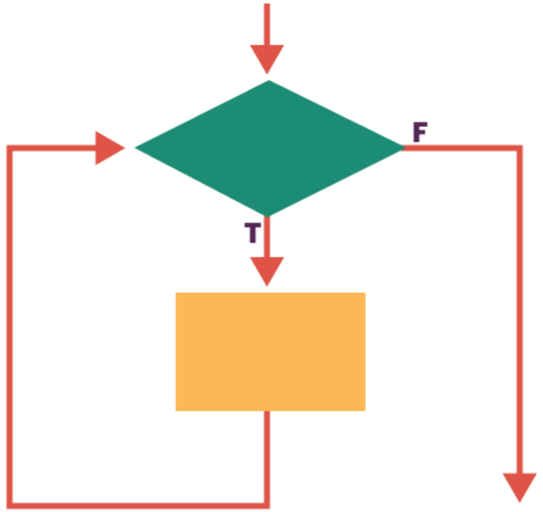
Introducing...

The While Loop

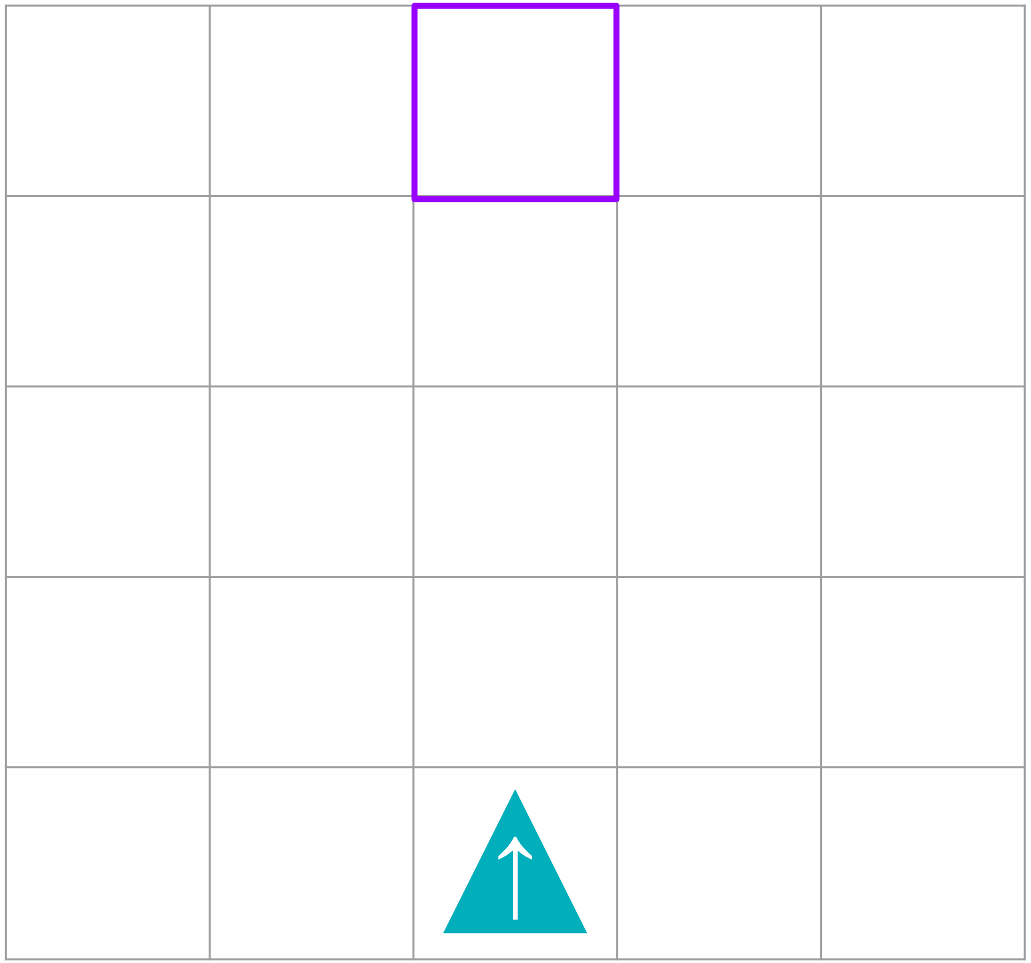
Here’s how it works:

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!



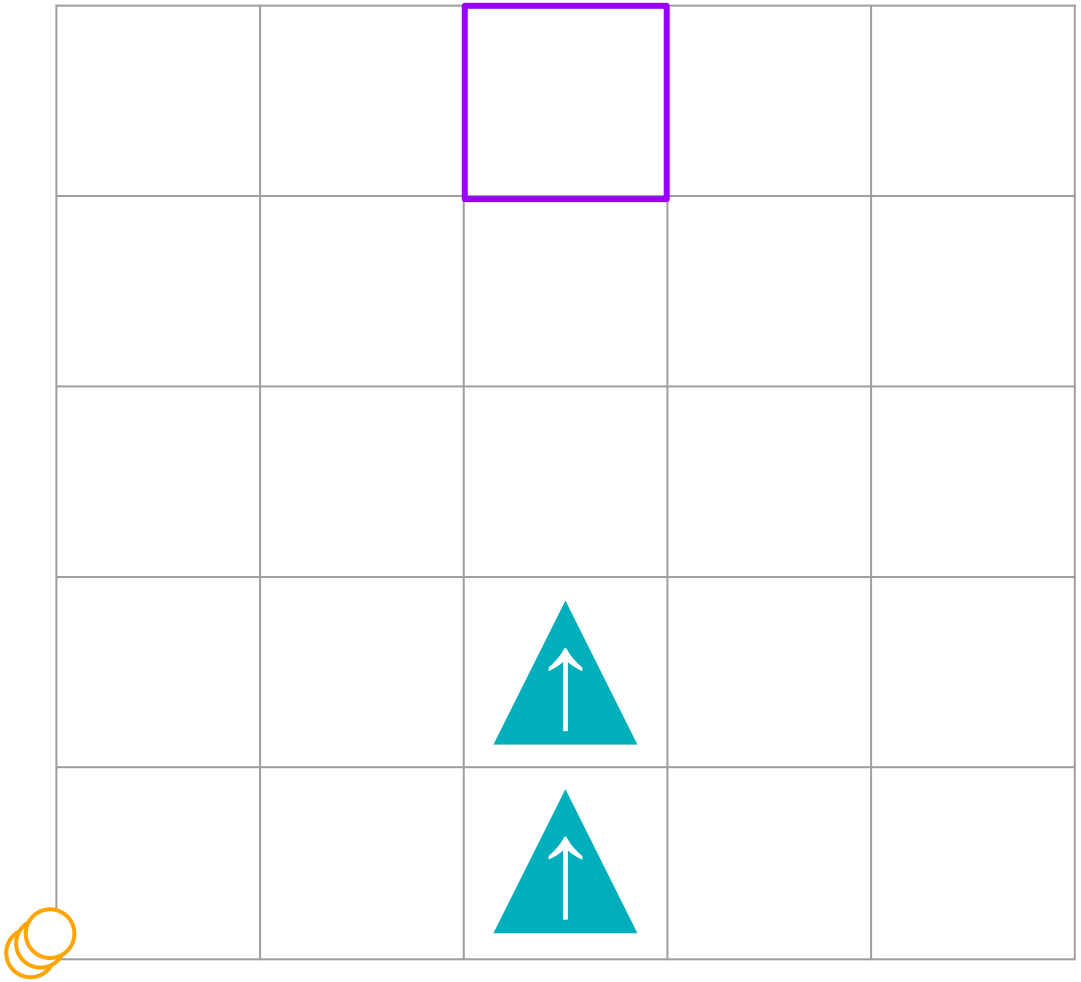
Unit 5 Lesson 5 - Activity



```
var steps = 0;

while(steps < 4){
    moveForward();
    steps++;
}
```

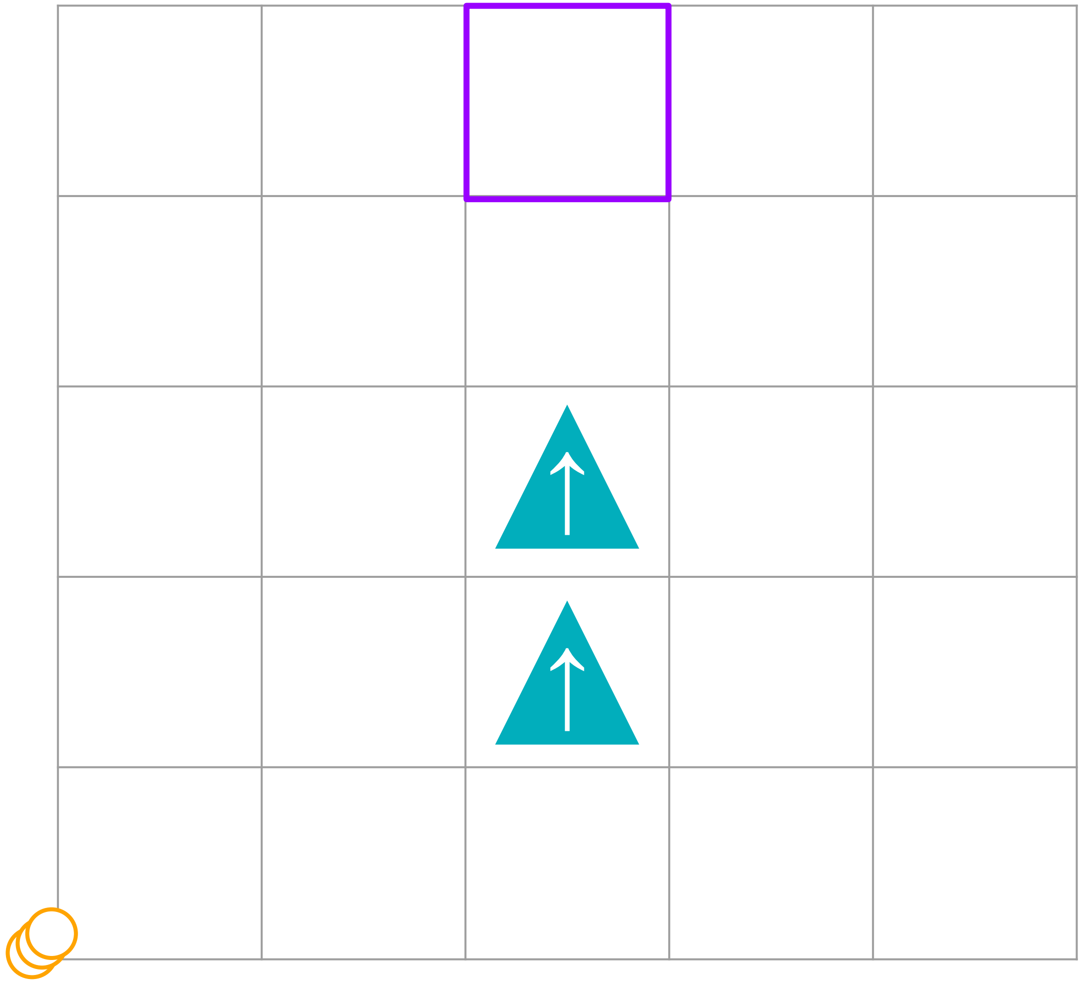
Do this:

- Follow along with this program on your own game board.
- Click to see how the while loop works.



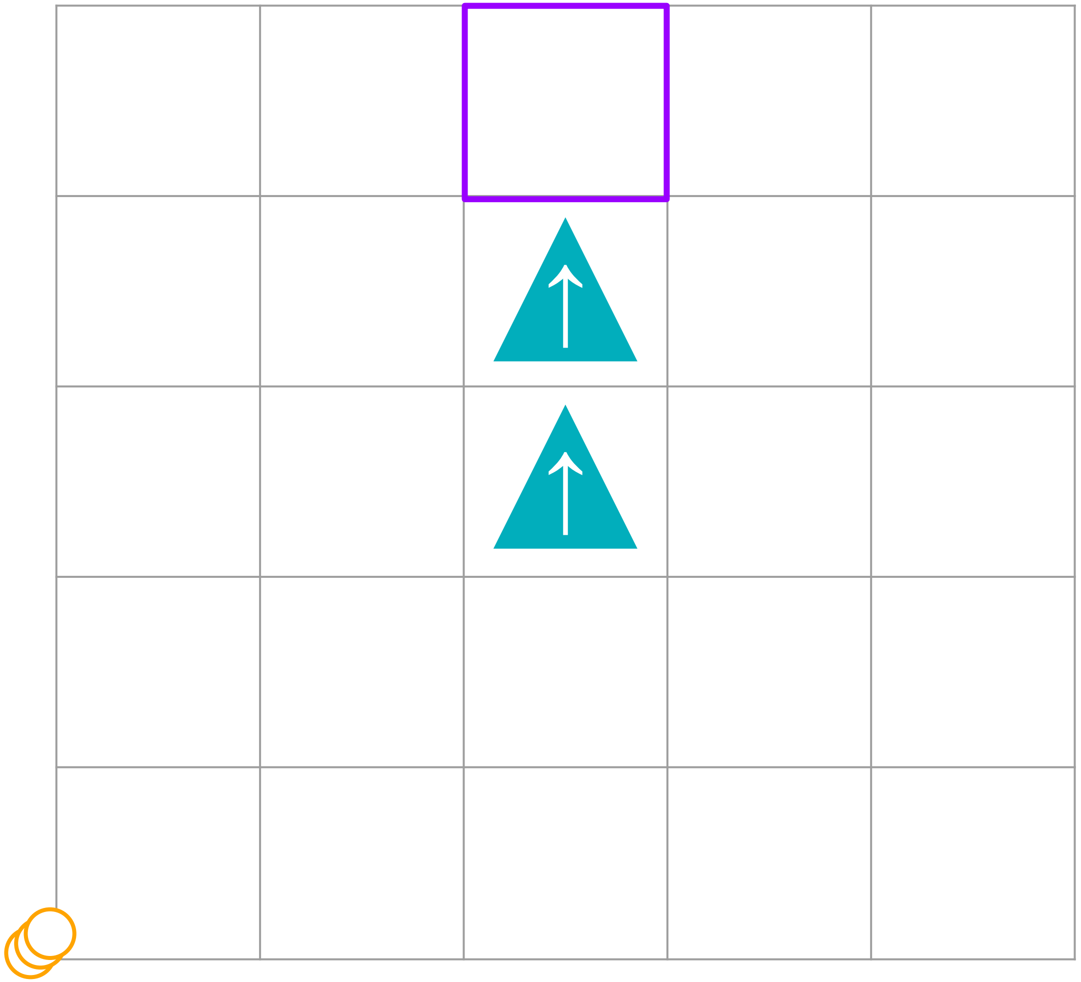
Round 1

```
               1
var steps = ~~0~~;

               true
while(steps < 4){
    moveForward();
    steps++;
}
```



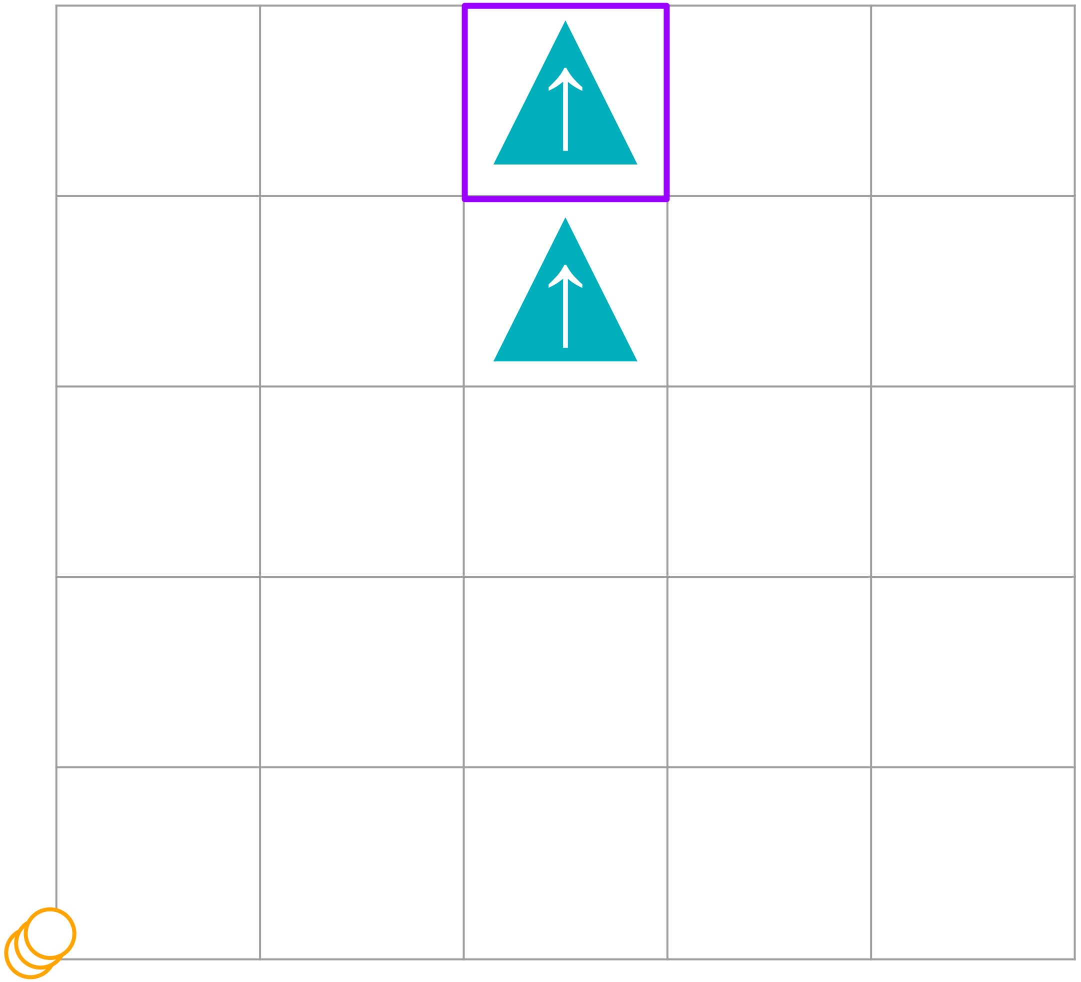
Round 2

```
var steps = 2 ~~1~~;

while(steps < 4){   true
    moveForward();
    steps++;
}
```



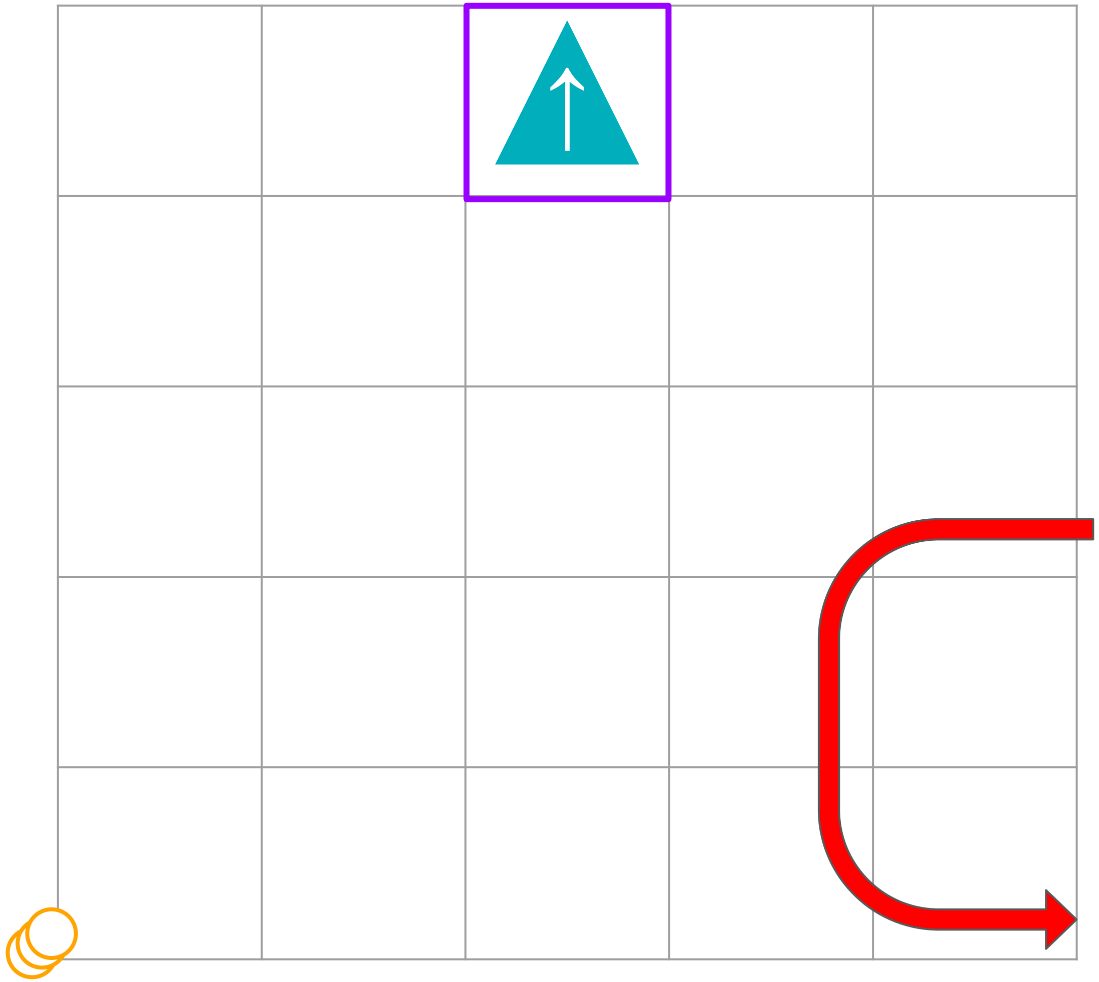
Round 3

```
var steps = ~~2~~ 3;

          true
while(steps < 4){
    moveForward();
    steps++;
}
```



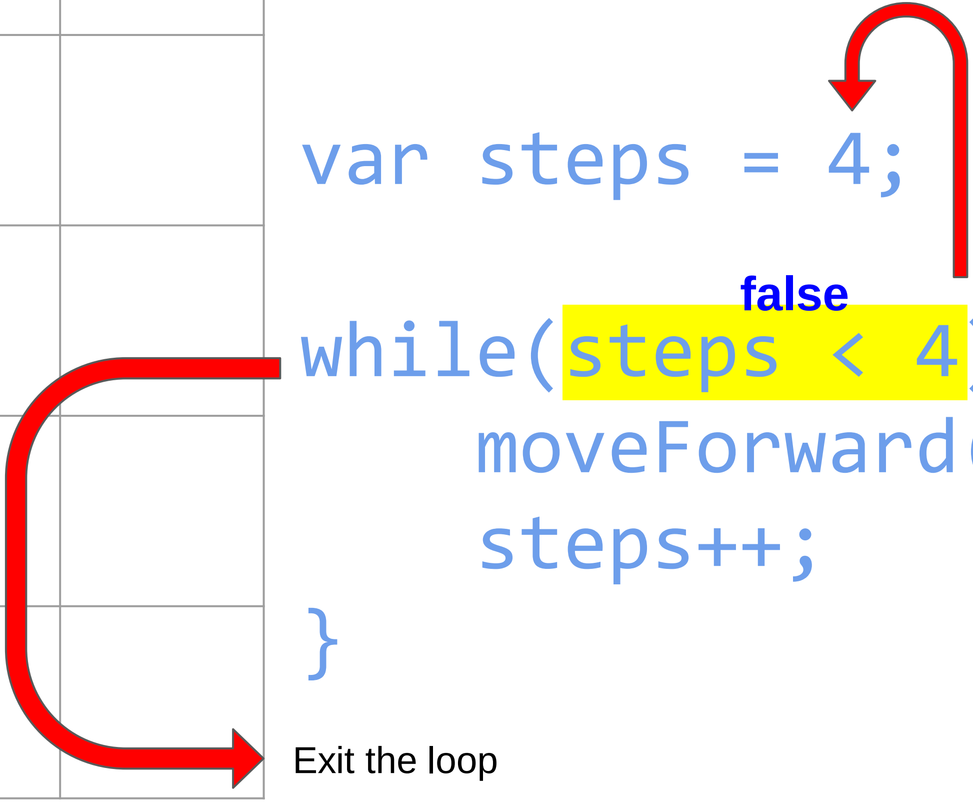
Round 4

```
var steps = ~~3~~ 4;

while(steps < 4){        true
    moveForward();
    steps++;
}
```

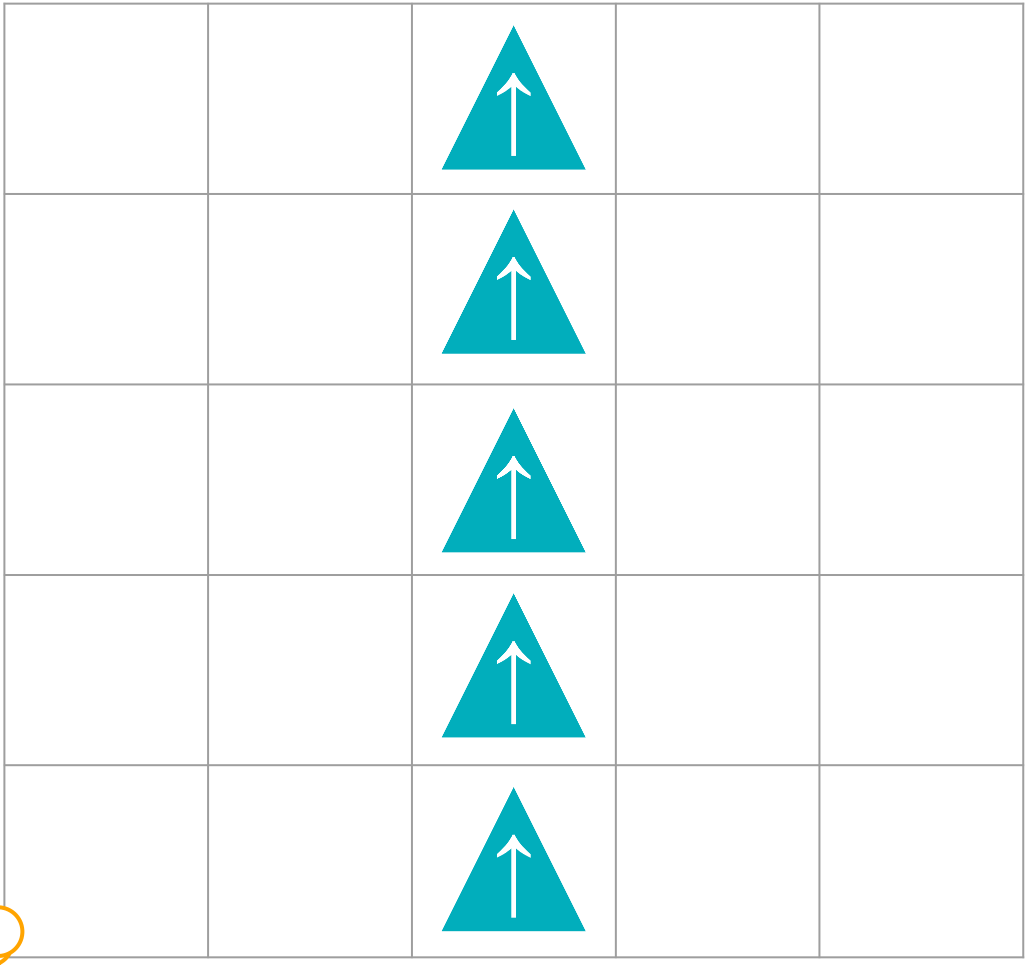


Round 5

```
var steps = 4;

while(steps < 4){
    moveForward();
    steps++;
}
```



Unit 5 Lesson 5 - Activity



```
var steps = 0;

while(steps < 4){
    moveForward();
}
```

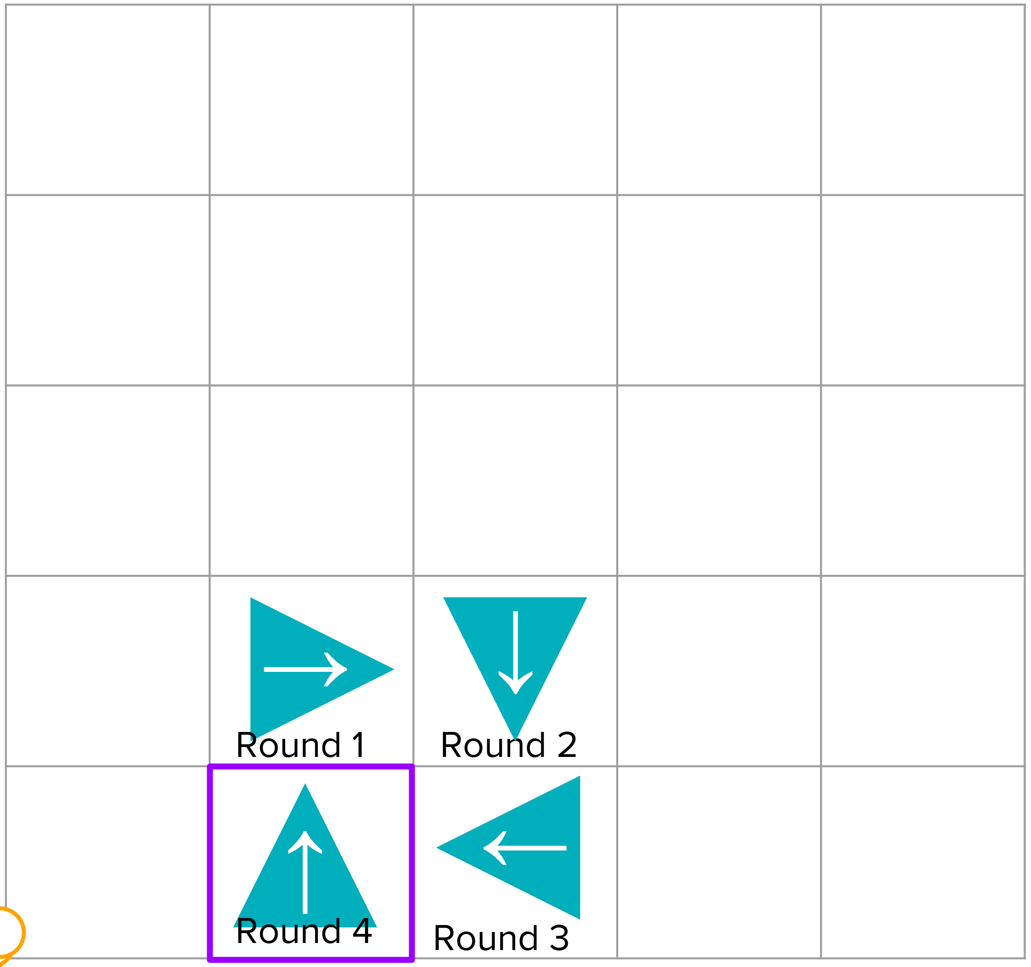
Do this:

- Run this program on your board.
- What happens?

Discuss:

- Why does the robot run off the board?

Unit 5 Lesson 5 - Activity



Do this:

- You have a new command:

```
var steps = 0;
    turnRight();
while(steps < 4){
    moveForward();
    turnRight();
    steps++;
}
```

- This 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?

While Loop: 3 Parts

```
var steps = 0;

while(steps < 4){
    moveForward();
    steps++;
};
```

- A counting variable set to an initial value
- A Boolean expression which checks the condition of that variable
- A statement which increases or decreases the variable that is being checked.
    - Note: if this piece is missing, you may create an **infinite loop** that never stops running, or crashes your browser!

A for loop combines these three parts into one statement

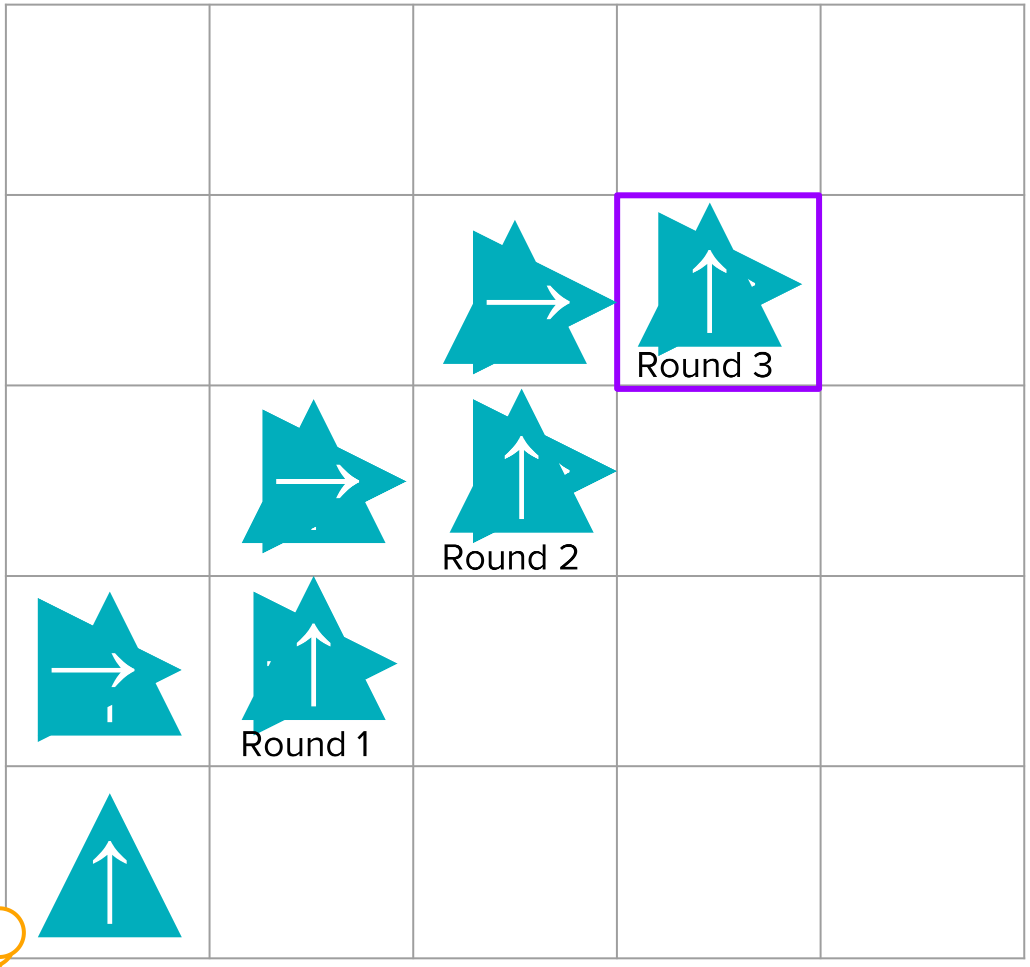
```
var steps = 0;

while(steps < 4){
      moveForward();
      steps++;
};
```

Any variable name can be used here. It's most common to use i.

```
for (var i=0; i<4; i++){
      moveForward();
};
```

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.

Unit 5 Lesson 5 - Activity



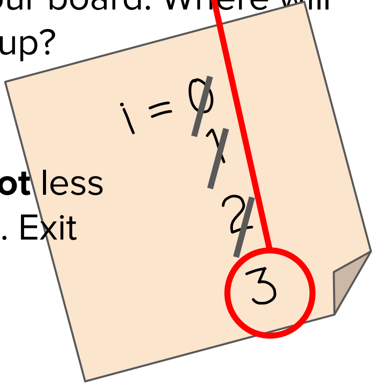
Do this:

Hint: Keep track of the variable **i** on a scrap piece of paper.

- You have another new command:

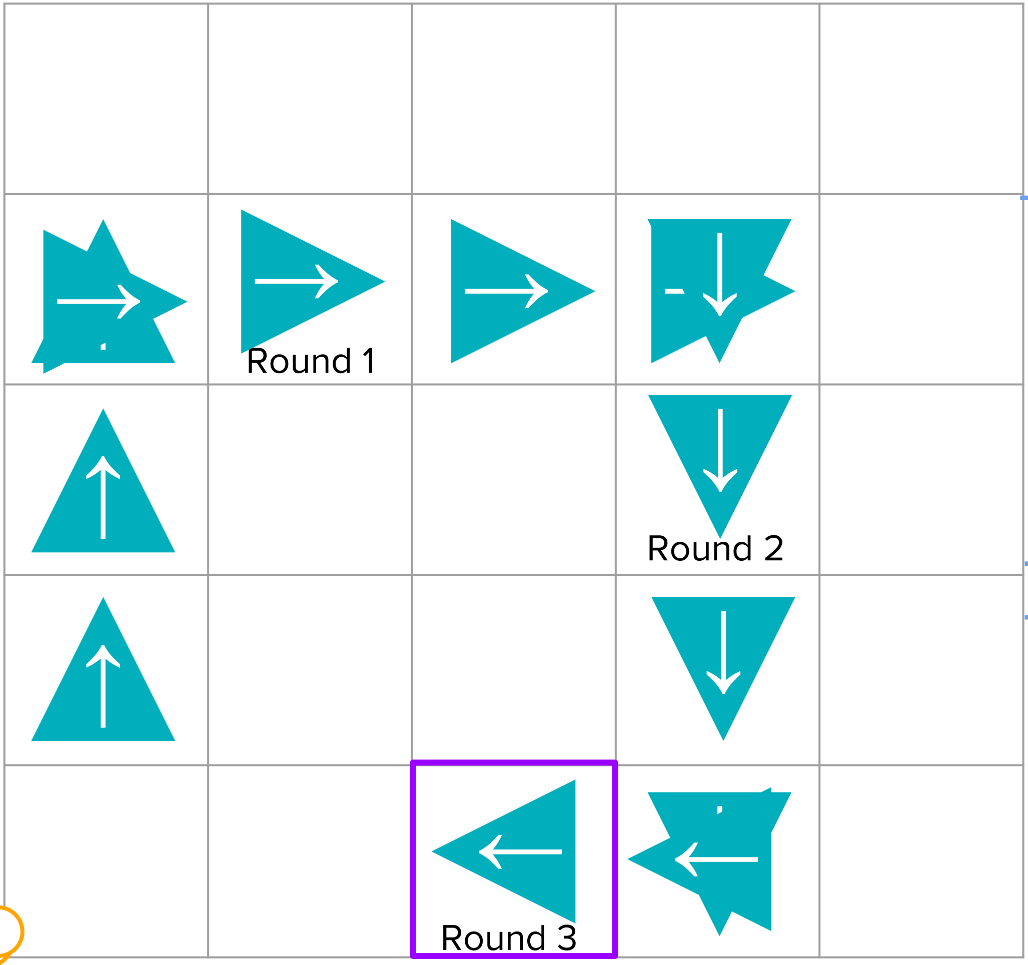
```
for(var i=0; i<3; i++){
  turnLeft();
  moveForward();
  turnRight();
  moveForward();
  turnLeft();
}
```

- This 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?



Three **is not** less than three. Exit the loop.

Unit 5 Lesson 5 - Activity

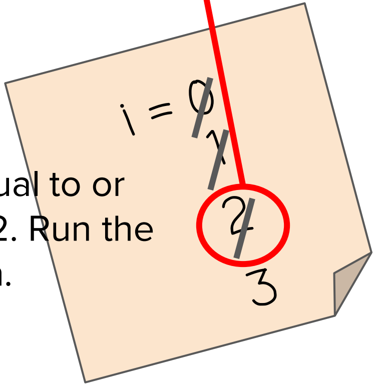


Do this:

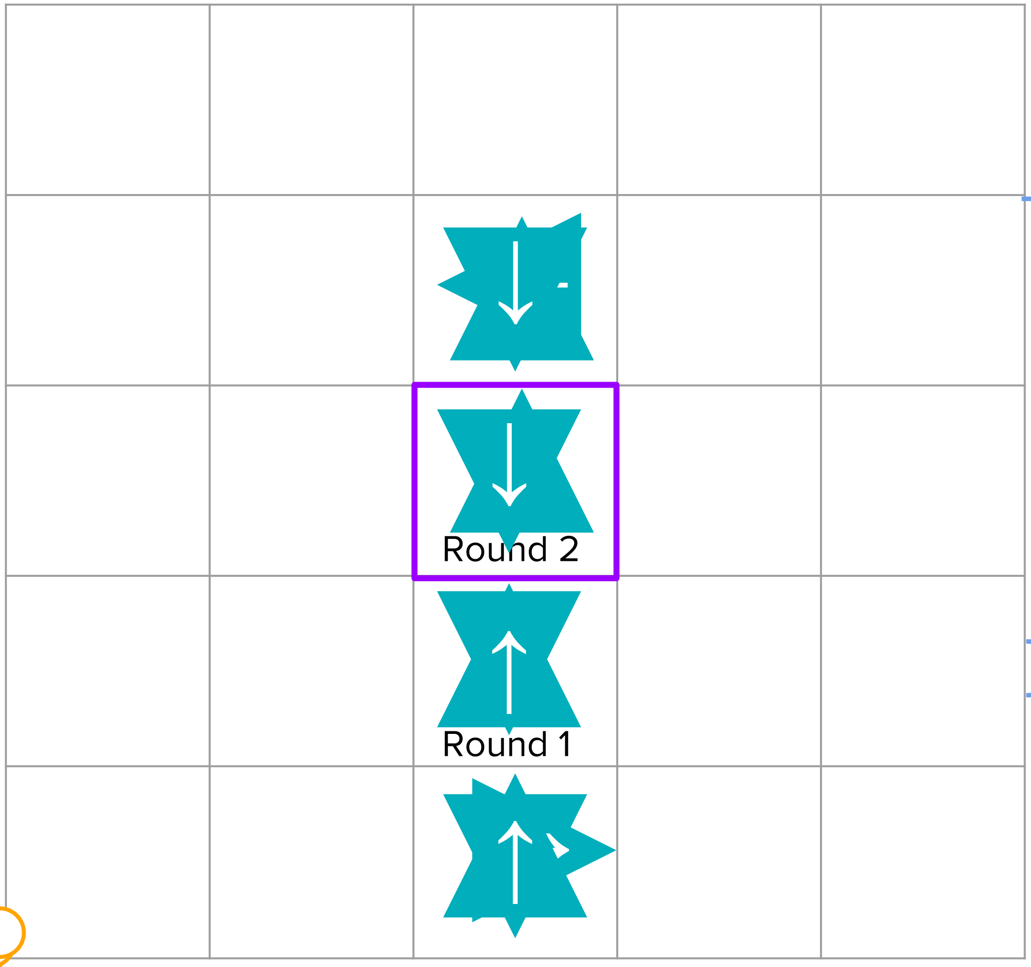
Hint: Keep track of the variable **i** on a scrap piece of paper.

- Click to see a new program. Run the program on your board. Where will the robot end up?

```
for(var i=0; i<=2; i++){
    moveForward();
    moveForward();
    turnRight();
    moveForward();
}
```



Two **is** equal to or less than 2. Run the loop again.

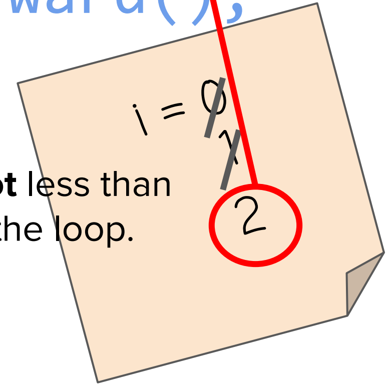


Hint: Keep track of the variable **i** on a scrap piece of paper.

Do this:

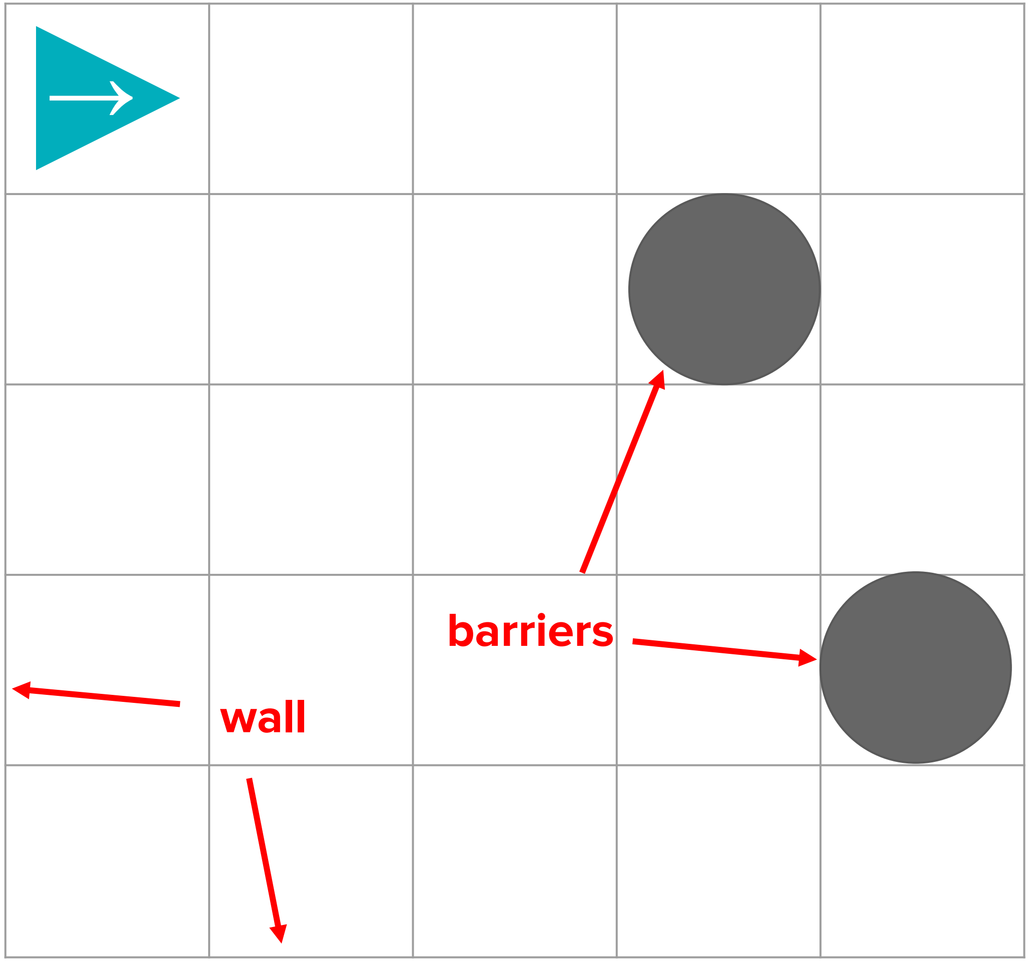
- Click to see a new program. Run the program on your board. Where will the robot end up?

```
for(var i=0; i<2; i++){
    moveForward();
    turnLeft();
    turnLeft();
    moveForward();
    moveForward();
}
```



Two **is not** less than two. Exit the loop.

Unit 5 Lesson 5 - Activity



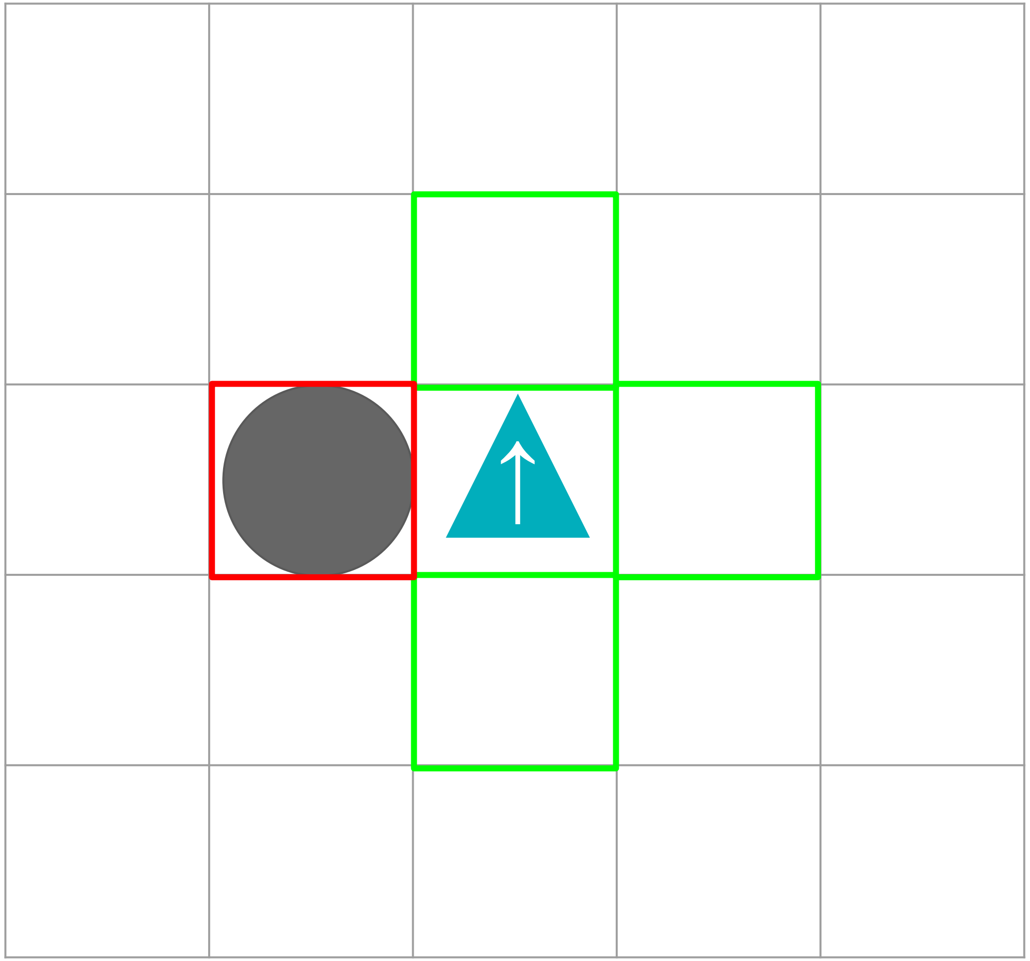
Let’s make it a little more challenging.

New commands:

```
canMove(left);
canMove(right);
canMove(forward);
canMove(backward);
```

These evaluate to **True** or **False**. They 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.

Unit 5 Lesson 5 - Activity



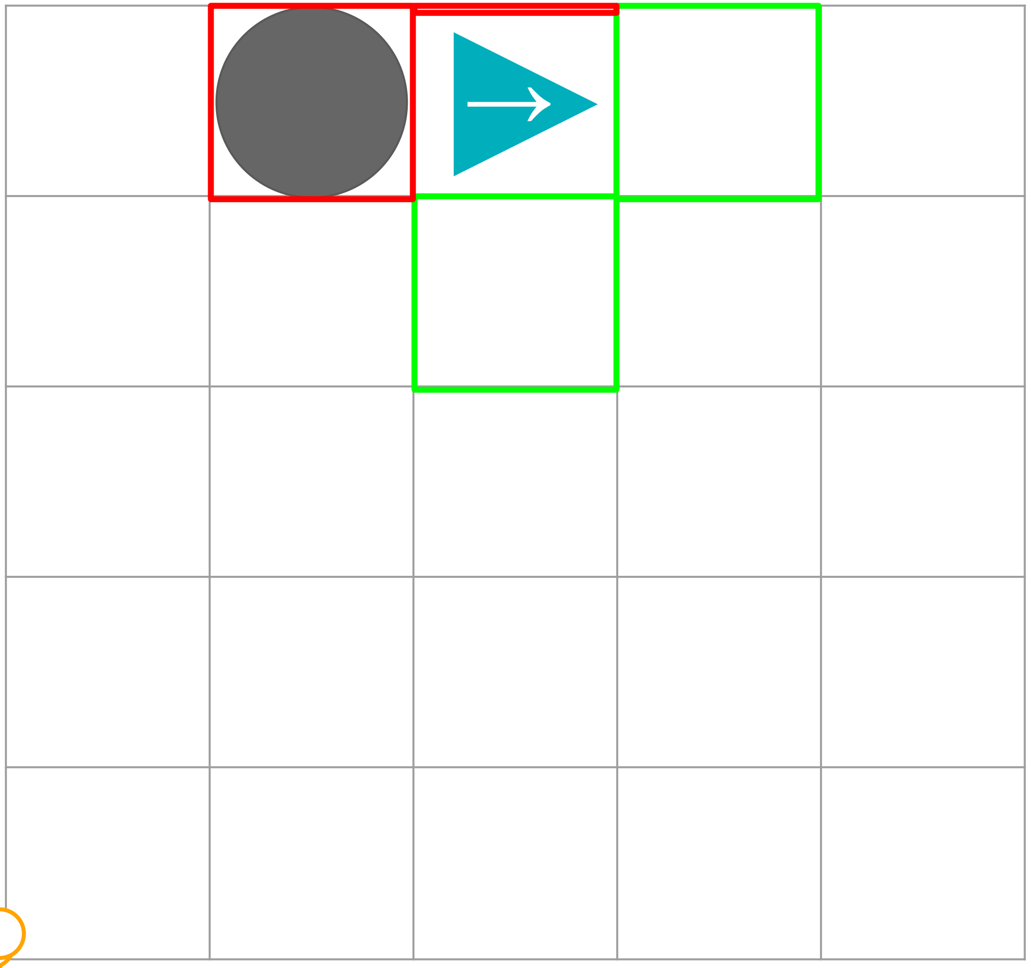
```
canMove(forward);
```
True

```
canMove(right);
```
True

```
canMove(left);
```
False

```
canMove(backward);
```
True

Unit 5 Lesson 5 - Activity



```
canMove(forward);
```

True

```
canMove(right);
```

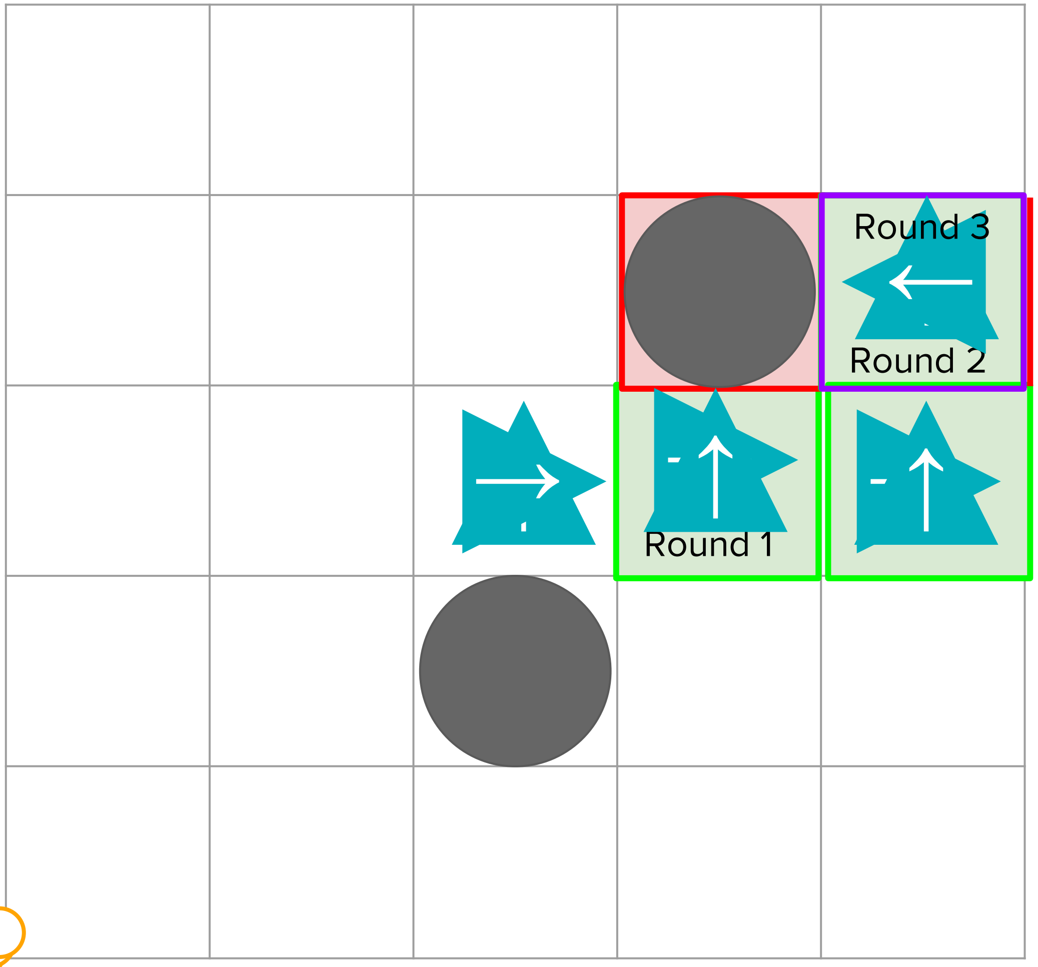
True

```
canMove(left);
```

False

```
canMove(backward);
```

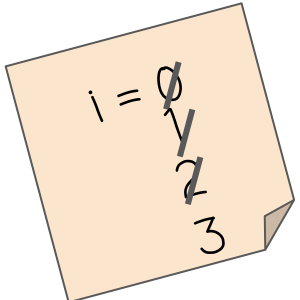
False



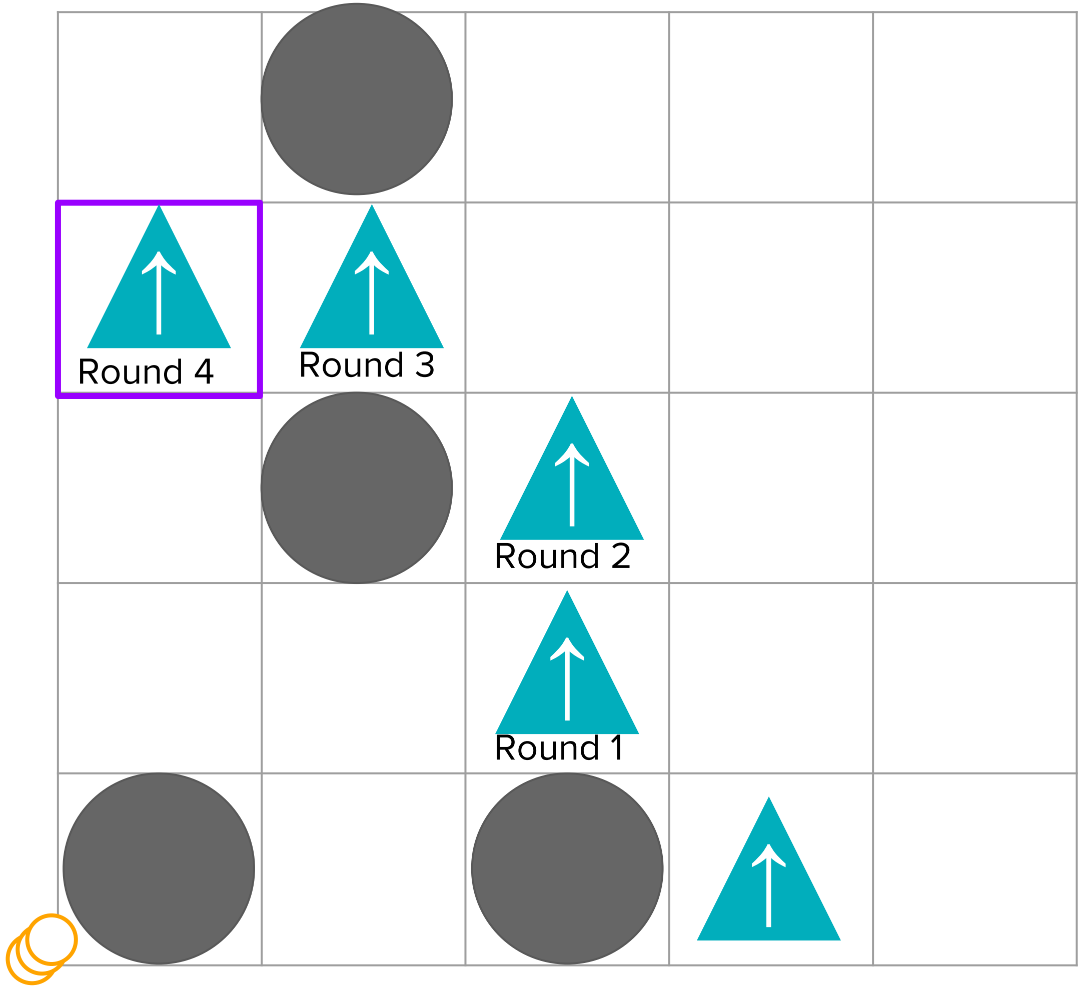
Do this:

- Run the following program. Where will the robot end up?

```
for(var i=0; i<3; i++){
    if(canMove(right)){
        turnRight();
        moveForward();
    }
    turnLeft();
    if(canMove(forward)){
        moveForward();
    }
}
```



Unit 5 Lesson 5 - Activity



Do this:

- Run the following program. Where will the robot end up?

```
for(var i=0; i<4; i++){
    if(canMove(forward)){
        moveForward();
    }
    turnLeft();
    if(canMove(forward)){
        moveForward();
    }
    turnRight();
}
```

Challenge!

- Set up a game board. 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!

For loop:

```
for(var i=0; i<3; i++){

}
```

Can be any number.
Your choice!

Commands:

```
canMove(left)
canMove(right)
canMove(forward)
canMove(backward)
moveForward()
turnRight()
turnLeft()
```

Key Takeaways

- **While Loop**

  Uses a boolean condition to repeatedly run a block of code. If it is true it runs the block of code contained within it. This process of checking the condition and running the block of code is repeated as long as the Boolean condition remains true. Once the Boolean expression becomes false it will stop.

```
var count = 0;

while(count < 3){
  … do something
  count++;
}
```

- **For Loop**

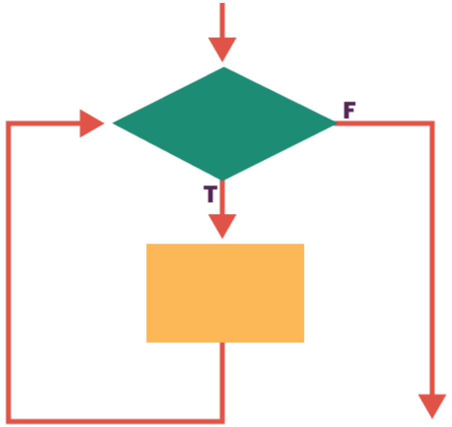
  Condenses the parts of a while loop into a shorter statement. Similar to the while loop, once the Boolean expression becomes false, the loop ends.

```
for(var i=0; i<3; i++){
   … do something
}
```

Wrap Up

Unit 5 Lesson 5 - Wrap Up

- **iteration:** a repetitive portion of an algorithm which repeats a specified number of times or until a given condition is met.



- **infinite loop:** occurs when the ending condition will never evaluate to true.

Unit 5 - Lesson 6
Loops Investigate

Warm Up

● ○ ○

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.

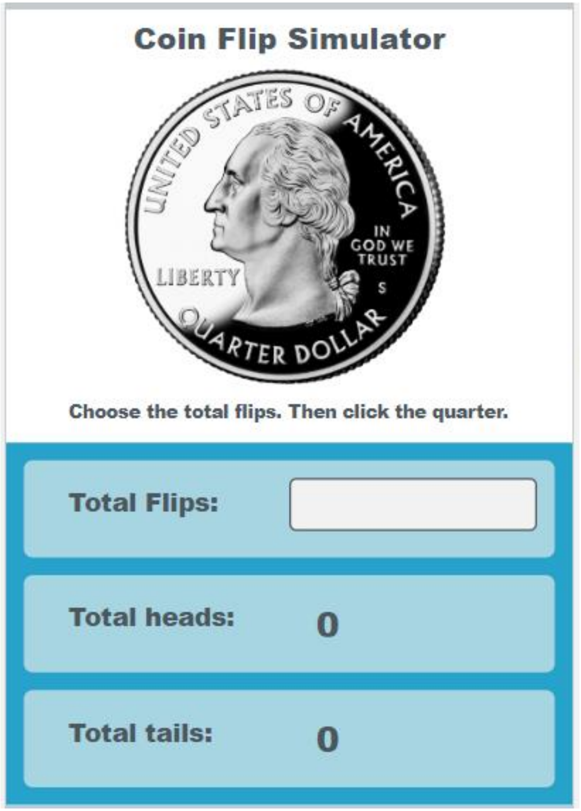
Activity

● ● ○

Prompt:

What is a simulation?

Why are they useful?



Choose the total flips. Then click the quarter.

Total Flips:

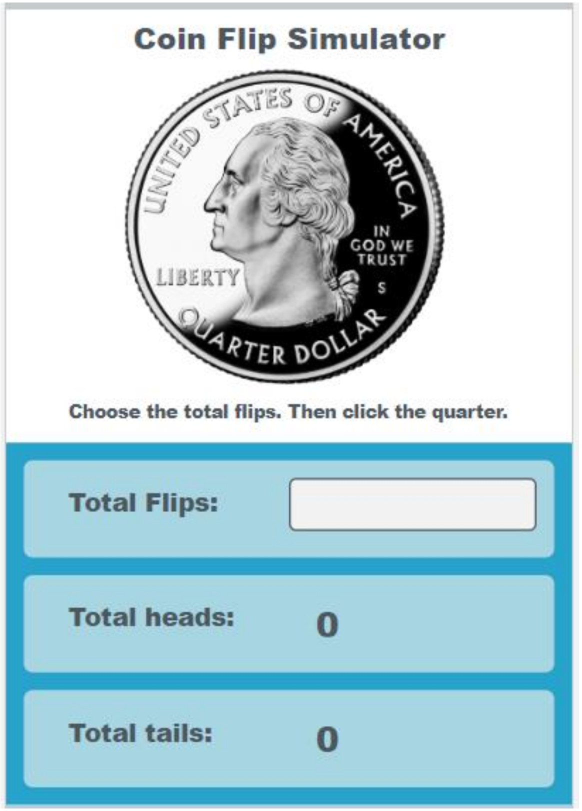
Total heads: 0

Total tails: 0

Do This:

With a partner

- Navigate to Lesson 6, Level 2
- Run the app



Coin Flip Simulator

Choose the total flips. Then click the quarter.

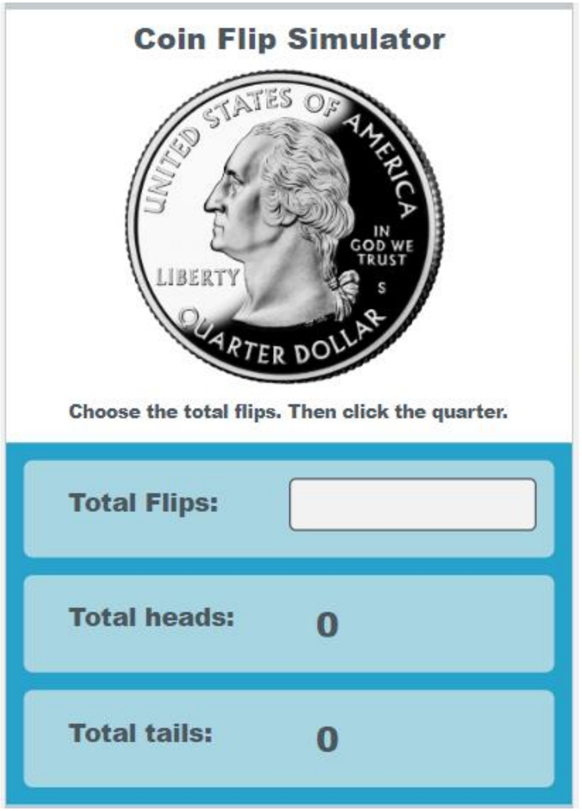
Total Flips:

Total heads: 0

Total tails: 0

Discuss:

- 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?


Coin Flip Simulator

Choose the total flips. Then click the quarter.

Total Flips:

Total heads: 0

Total tails: 0

Do This:

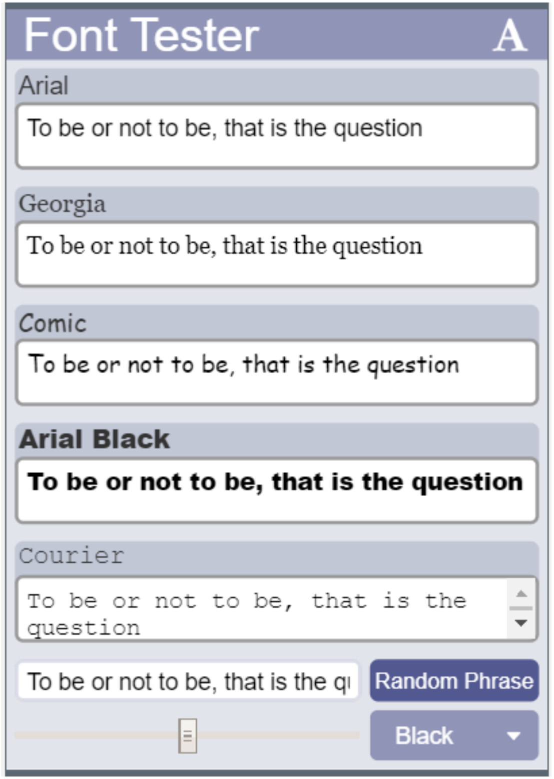
- Navigate to Level 3
- Run the program, watch the code run, and carefully read each individual part of the program

Discuss:

- 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?
    - For example: `tails++`

Prompt:

How does a loop help when running similar simulations?


Font Tester

Arial

To be or not to be, that is the question

Georgia

To be or not to be, that is the question

Comic

To be or not to be, that is the question

Arial Black

To be or not to be, that is the question

Courier

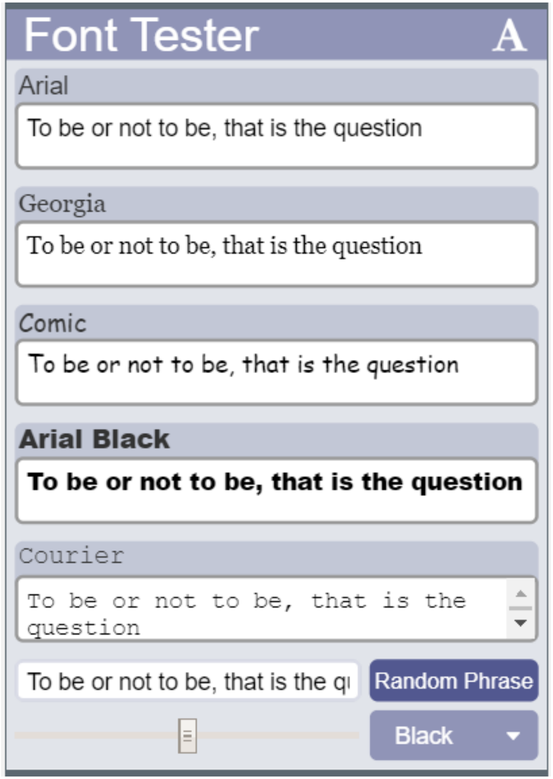
To be or not to be, that is the question

To be or not to be, that is the qu

Do This:

With a partner

- Navigate to Level 4
- Run the app



Font Tester A

Arial

To be or not to be, that is the question

Georgia

To be or not to be, that is the question

Comic

To be or not to be, that is the question

Arial Black

To be or not to be, that is the question

Courier

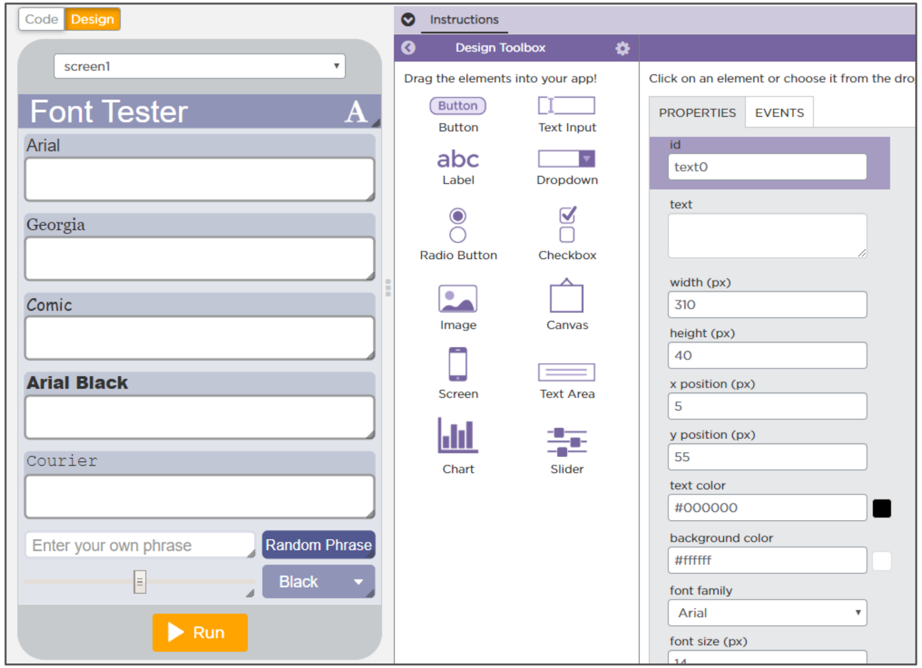
To be or not to be, that is the question

To be or not to be, that is the qu

Discuss:

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

Unit 5 Lesson 6 - Activity


Code Design

screen1

Font Tester

Arial

Georgia

Comic

Arial Black

Courier

Enter your own phrase

Random Phrase

Black

Run

Instructions

Design Toolbox

Drag the elements into your app!

Click on an element or choose it from the dro

PROPERTIES EVENTS

id: text0

text:

width (px): 310

height (px): 40

x position (px): 5

y position (px): 55

text color: #000000

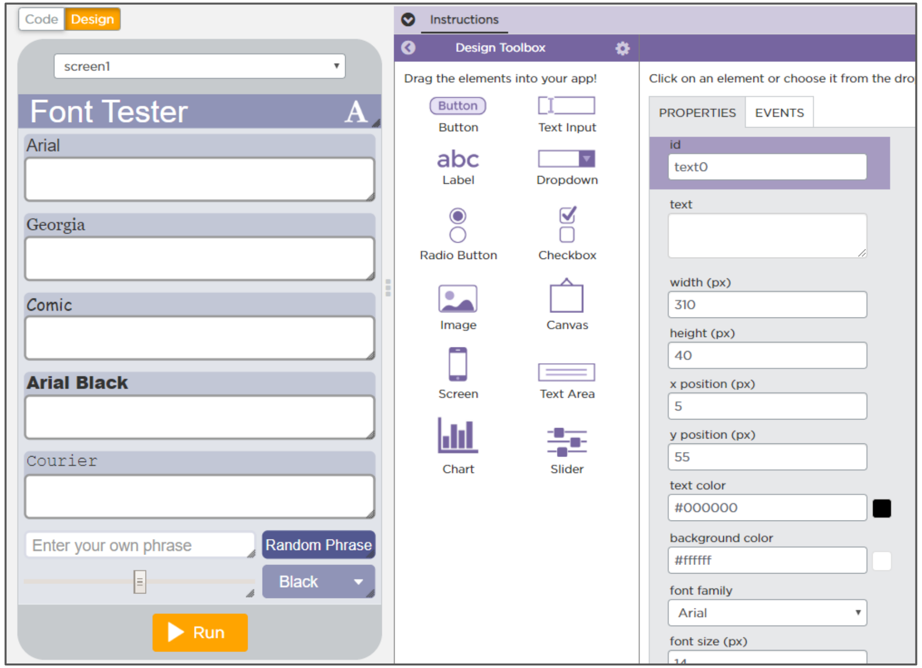
background color: #ffffff

font family: Arial

font size (px): 14

Do This:

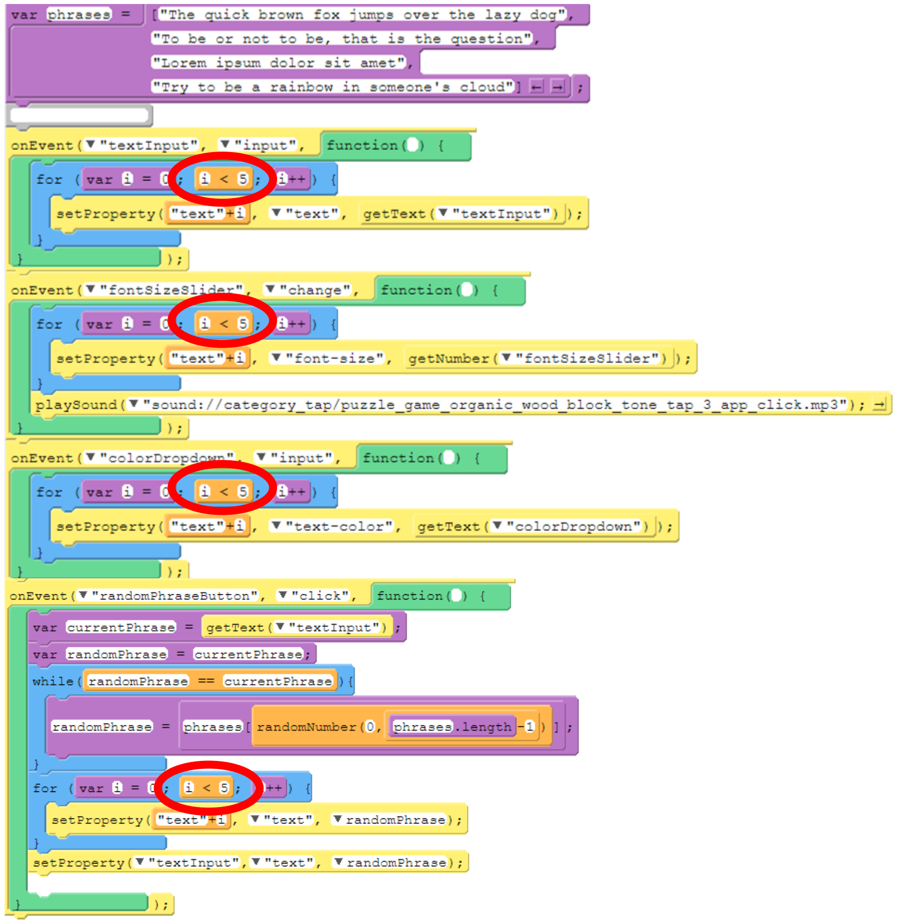
- Take a look at the names of the screen elements in Design Mode.
- Navigate back to the code. How is `"text" + i` used?
- How does it evaluate with each round of the loop?



id: text0
text:
width (px): 310
height (px): 40
x position (px): 5
y position (px): 55
text color: #000000
background color: #ffffff
font family: Arial
font size (px): 14

Do This:

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



```
var phrases = ["The quick brown fox jumps over the lazy dog",
               "To be or not to be, that is the question",
               "Lorem ipsum dolor sit amet",
               "Try to be a rainbow in someone's cloud"];

onEvent("textInput", "input", function( ) {
  for (var i = 0; i < 5; i++) {
    setProperty("text"+i, "text", getText("textInput"));
  }
});
onEvent("fontSizeSlider", "change", function( ) {
  for (var i = 0; i < 5; i++) {
    setProperty("text"+i, "font-size", getNumber("fontSizeSlider"));
  }
  playSound("sound://category_tap/puzzle_game_organic_wood_block_tone_tap_3_app_click.mp3");
});
onEvent("colorDropdown", "input", function( ) {
  for (var i = 0; i < 5; i++) {
    setProperty("text"+i, "text-color", getText("colorDropdown"));
  }
});
onEvent("randomPhraseButton", "click", function( ) {
  var currentPhrase = getText("textInput");
  var randomPhrase = currentPhrase;
  while (randomPhrase == currentPhrase) {
    randomPhrase = phrases[randomNumber(0, phrases.length-1)];
  }
  for (var i = 0; i < 5; i++) {
    setProperty("text"+i, "text", randomPhrase);
  }
  setProperty("textInput", "text", randomPhrase);
});
```

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 on to the next option and repeat:
  - `i < 4`
  - `i < 3`
  - `i <= 4`

Wrap Up

A for loop combines these three parts into one statement

```
var steps = 0;

while(steps < 4){
    moveForward();
    steps++;
};
```

Any variable name can be used here. It's most common to use i.

```
for (var i=0; i<4; i++){
    moveForward();
};
```

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.

Prompt:

What aspects of working with for loops do you feel clicked today?

What do you still feel like you have trouble with?

Unit 5 - Lesson 7
Loops Practice

Warm Up

● ○ ○

Activity

●●○

Debugging: the process of finding and fixing problems in code

Describe

The Problem

What do you expect it to do?
What does it actually do?
Does it always happen?

Hunt

For Bugs

Are there warnings or errors?
What did you change most recently?
Explain your code to someone else
Look for code related to the problem

Try

Solutions

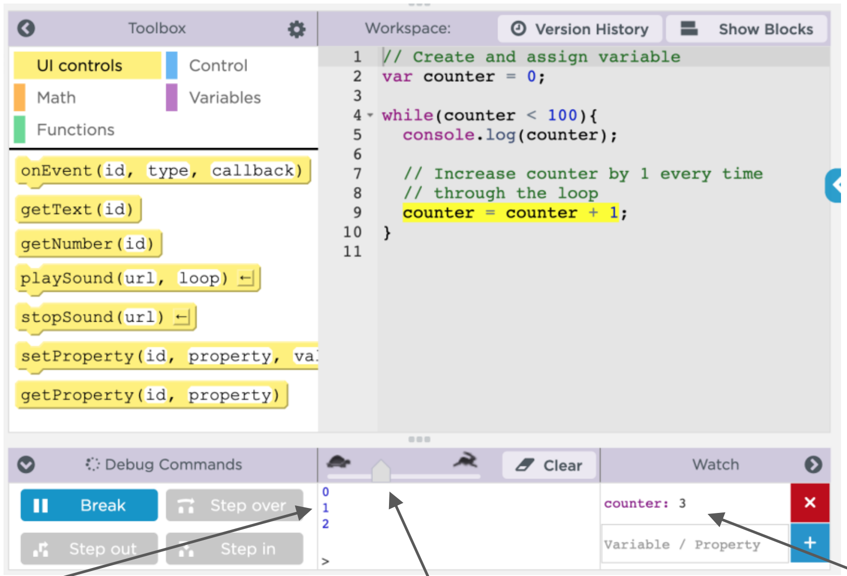
Make a small change

Document

As You Go

What have you learned?
What strategies did you use?
What questions do you have?

Debugging Loops



```
// Create and assign variable
var counter = 0;

while(counter < 100){
  console.log(counter);

  // Increase counter by 1 every time
  // through the loop
  counter = counter + 1;
}
```

```
0
1
2
>
```

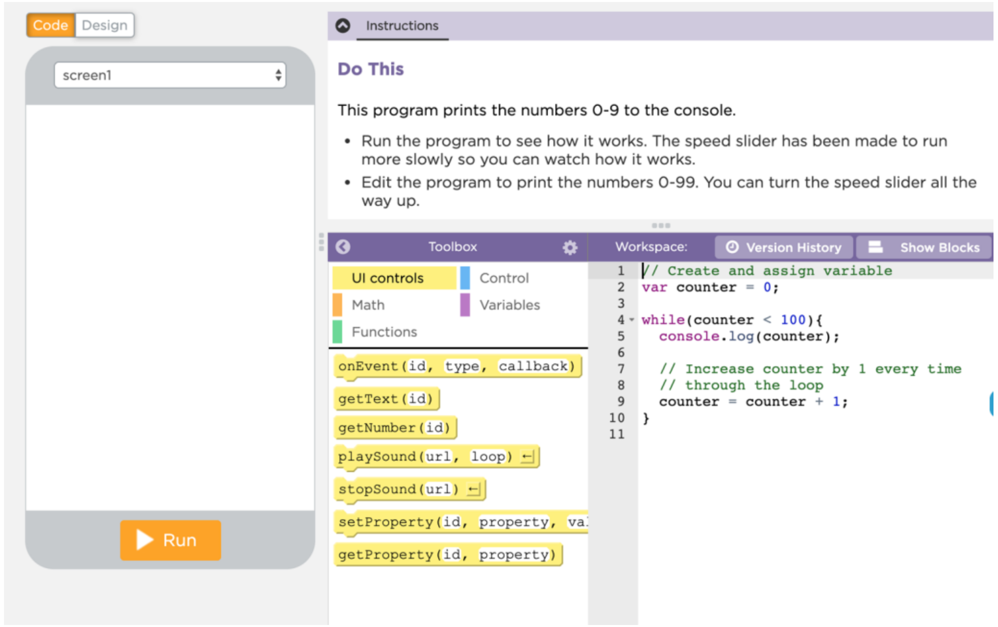
```
counter: 3
```

Use `console.log` to print out important information.

Run your code slowly so you can see how your loop is actually working.

Track your iterator variable in the Watch panel.

Loops Practice

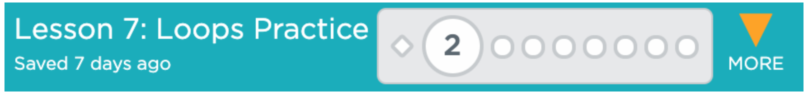


Do This

This program prints the numbers 0-9 to the console.

- Run the program to see how it works. The speed slider has been made to run more slowly so you can watch how it works.
- Edit the program to print the numbers 0-99. You can turn the speed slider all the way up.

```
// Create and assign variable
var counter = 0;

while(counter < 100){
  console.log(counter);

  // Increase counter by 1 every time
  // through the loop
  counter = counter + 1;
}
```



Do This:

- Navigate to Lesson 7, Level 2 on Code Studio

Wrap Up

Prompt:

What aspects of working with loops do you feel clicked today?

What do you still have trouble with?

Unit 5 - Lesson 8
Loops Make

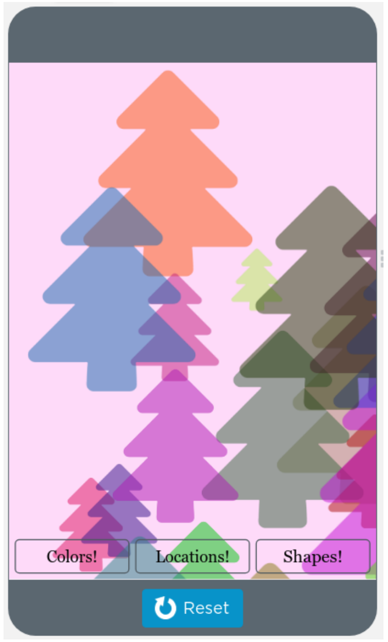
Warm Up

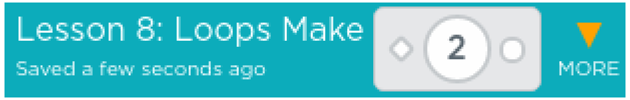
● ○ ○

Activity

●●○

Lists Make:
Lock Screen Maker





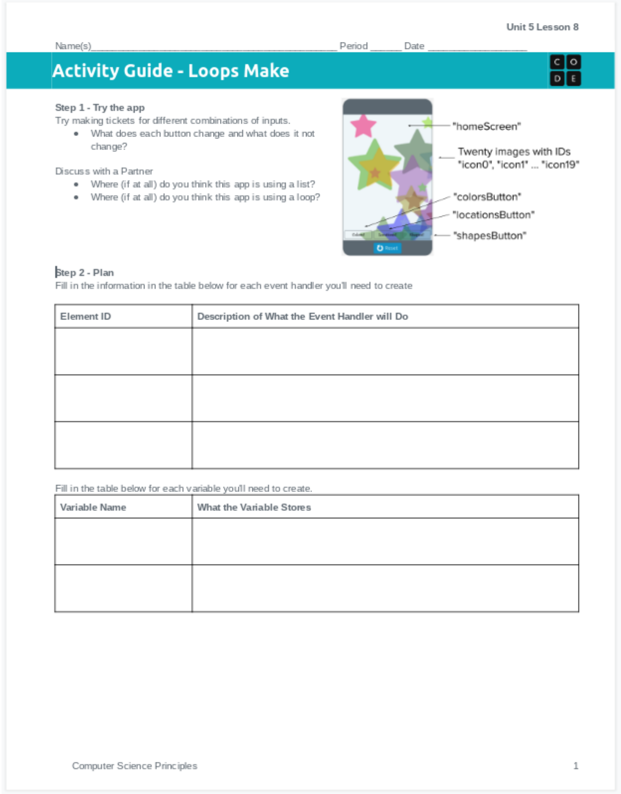
Do This:

- Navigate to Lesson 8, Level 2 on Code Studio

Prompt:

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

Do This: Make the Lock Screen Maker!



Unit 5 Lesson 8

Name(s)_______________________________________ Period ______ Date ________________

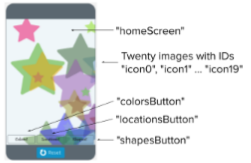
Activity Guide - Loops Make

Step 1 - Try the app
Try making tickets for different combinations of inputs.

- What does each button change and what does it not change?

Discuss with a Partner

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



Step 2 - Plan
Fill in the information in the table below for each event handler you'll need to create

Element ID	Description of What the Event Handler will Do

Fill in the table below for each variable you'll need to create.

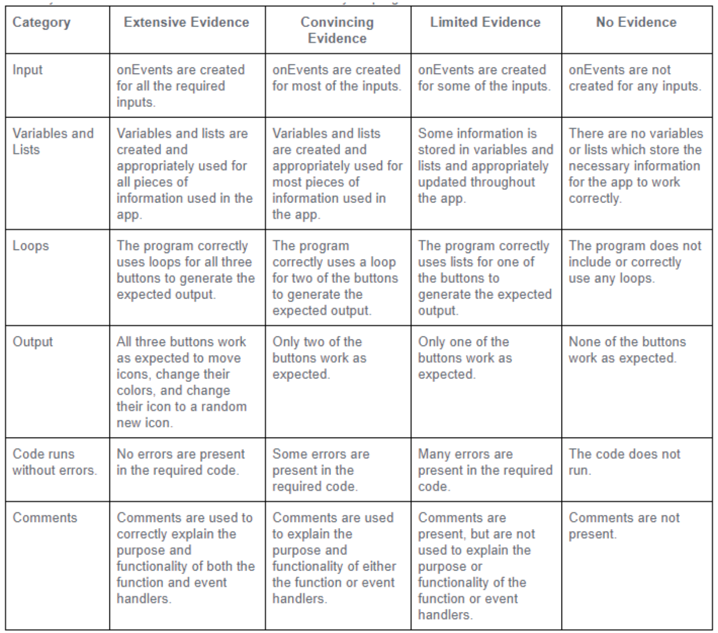
Variable Name	What the Variable Stores

Computer Science Principles 1

Use the activity guide to plan out your code, including the variables you'll create and the event handlers.

Step 3 includes steps you can follow to build the app, or you can use your own process.

Don't forget: check the rubric before submitting



Category	Extensive Evidence	Convincing Evidence	Limited Evidence	No Evidence
Input	onEvents are created for all the required inputs.	onEvents are created for most of the inputs.	onEvents are created for some of the inputs.	onEvents are not created for any inputs.
Variables and Lists	Variables and lists are created and appropriately used for all pieces of information used in the app.	Variables and lists are created and appropriately used for most pieces of information used in the app.	Some information is stored in variables and lists and appropriately updated throughout the app.	There are no variables or lists which store the necessary information for the app to work correctly.
Loops	The program correctly uses loops for all three buttons to generate the expected output.	The program correctly uses a loop for two of the buttons to generate the expected output.	The program correctly uses lists for one of the buttons to generate the expected output.	The program does not include or correctly use any loops.
Output	All three buttons work as expected to move icons, change their colors, and change their icon to a random new icon.	Only two of the buttons work as expected.	Only one of the buttons work as expected.	None of the buttons work as expected.
Code runs without errors.	No errors are present in the required code.	Some errors are present in the required code.	Many errors are present in the required code.	The code does not run.
Comments	Comments are used to correctly explain the purpose and functionality of both the function and event handlers.	Comments are used to explain the purpose and functionality of either the function or event handlers.	Comments are present, but are not used to explain the purpose or functionality of the function or event handlers.	Comments are not present.

Submit

Wrap Up

Great job today!

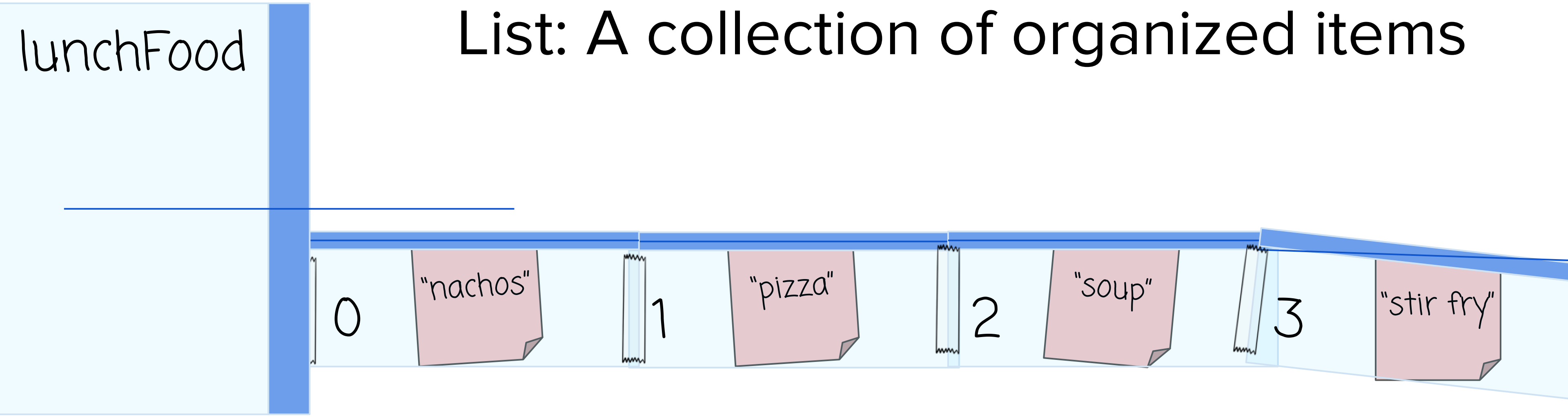
Unit 5 - Lesson 9
Traversals Explore

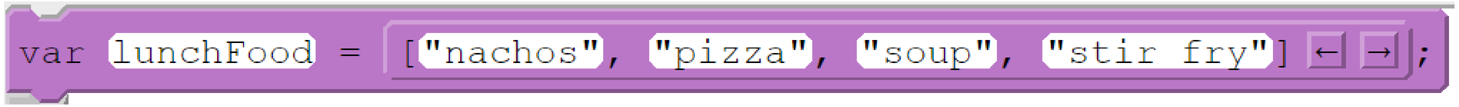
Warm Up

● ○ ○

Review

List: A collection of organized items

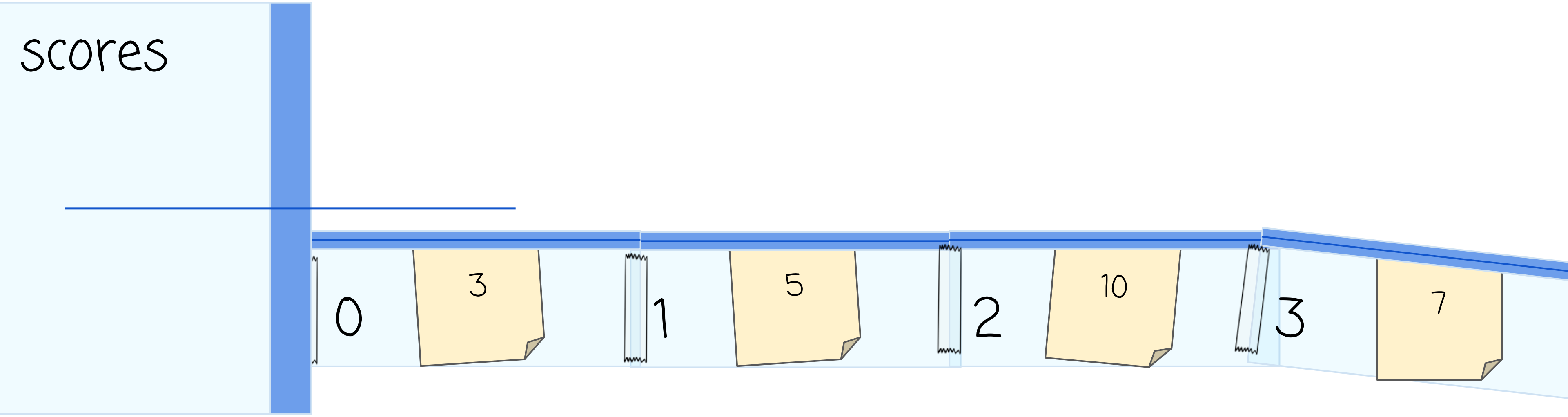


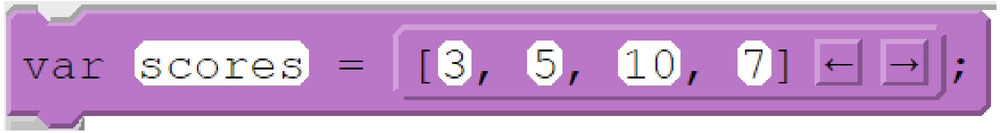


```
var lunchFood = ["nachos", "pizza", "soup", "stir fry"] ← → ;
```

```
var lunchFood = [“nachos”, “pizza”, “soup”, “stir fry”];
```

Lists can contain different types of data, including numbers and strings.





```
var scores = [3, 5, 10, 7];
```

```
var scores= [3, 5, 10, 7];
```

Activity

● ● ○

Traversals Explore

You and your partner should have:

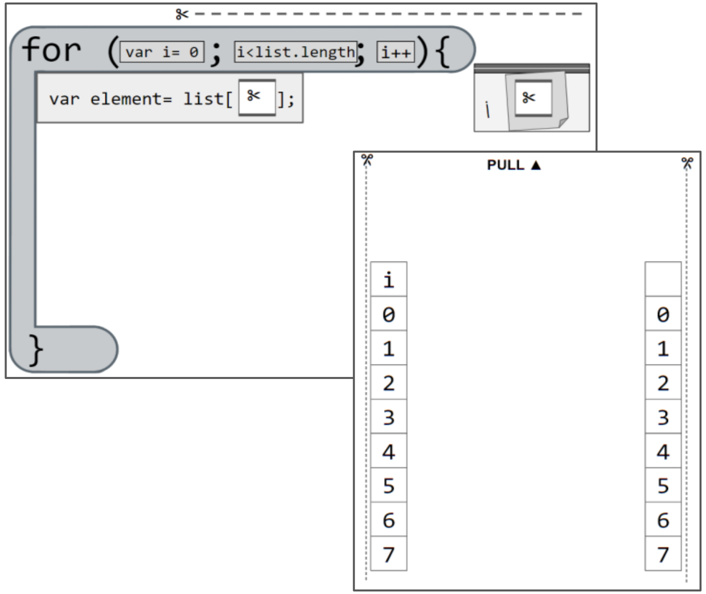
A few baggies

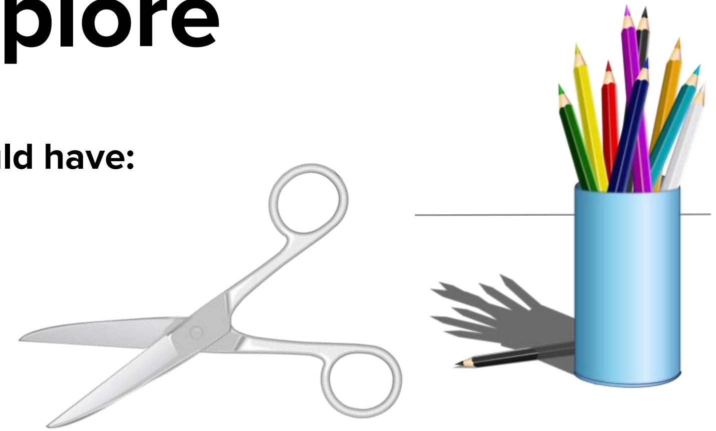
Scissors

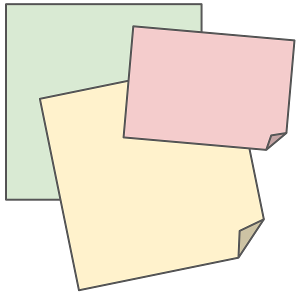
A few sticky notes

Pen/Pencil

Traversal Machine







Do This:

- On a scrap piece of paper or a sticky note, write down the following list:

```
var temps = [ 77, 85, 89, 65 ];
```

What if...

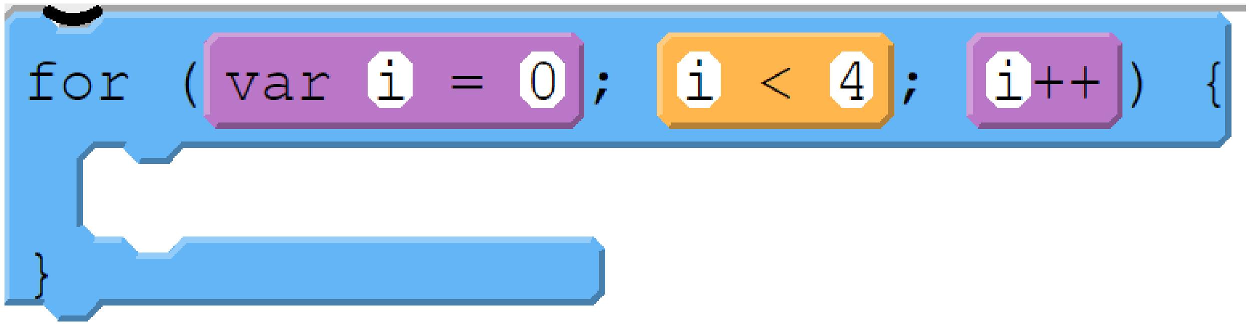
I want to know if a value is in a list.

With a partner, discuss the method you would use to look for an item in a list.

Most likely, you would scan your list looking at each item until you found the correct one.

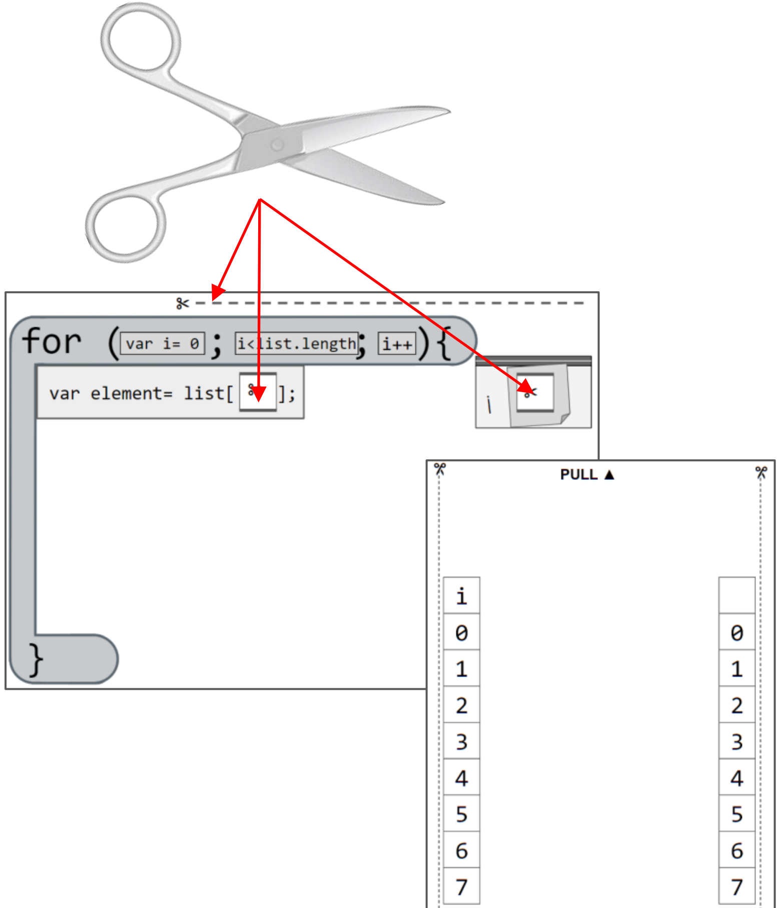
Computers are really good at checking things one by one.

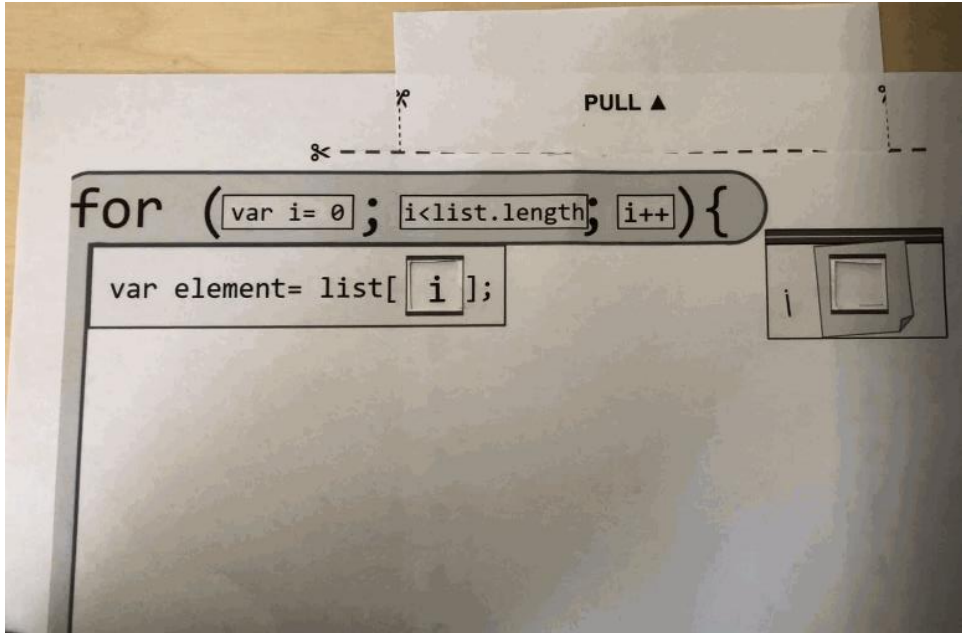
To do this... let’s use a **for loop.**



```
for ( var i = 0; i < 4; i++) {
}
```

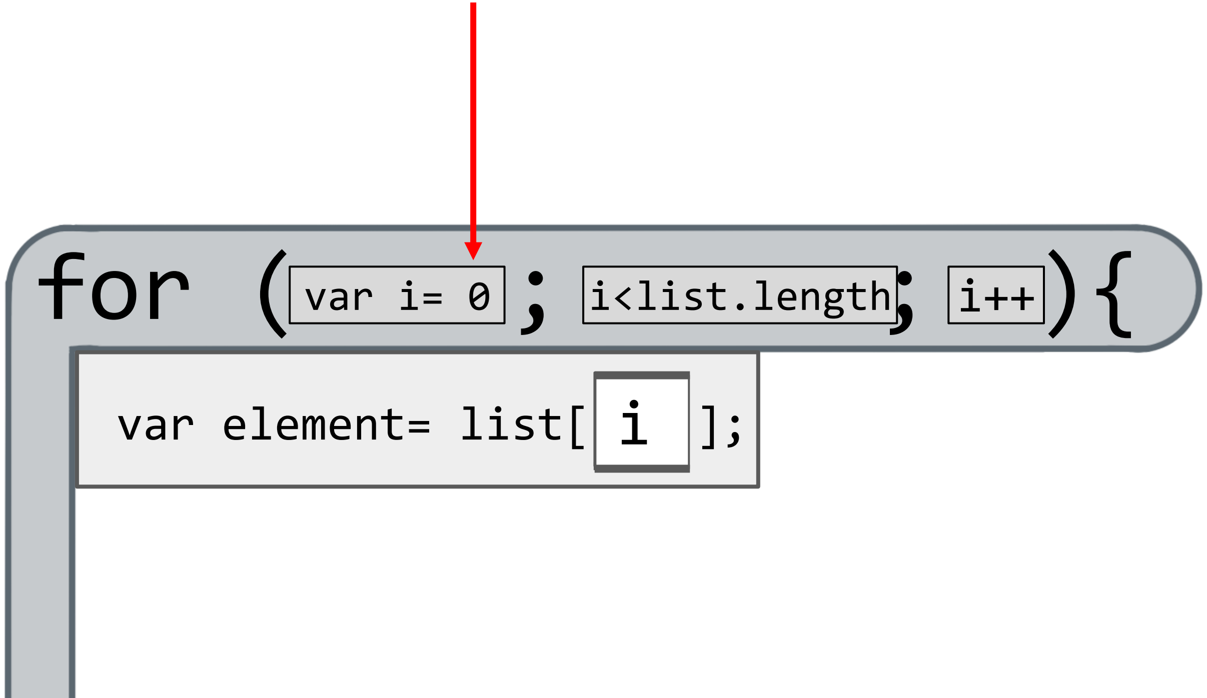
Get ready! Set up the Traversal Machine





Unit 5 Lesson 9 - Activity

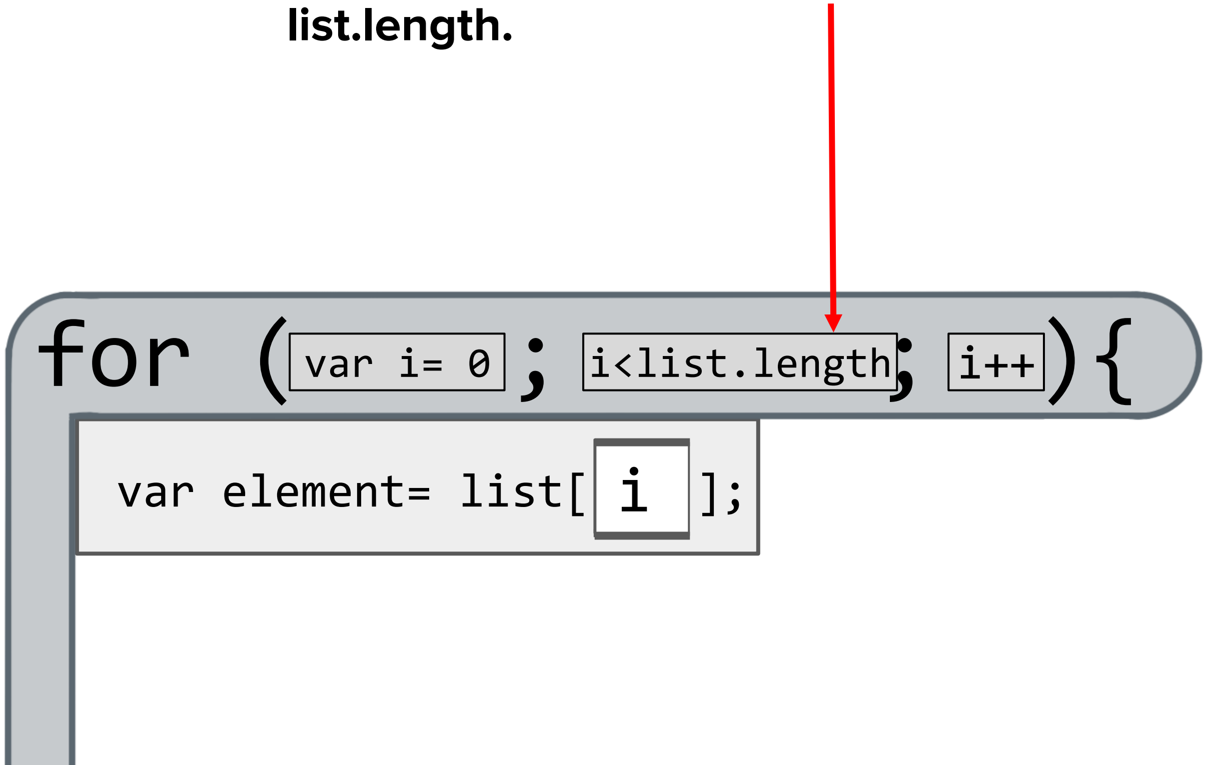
This is where we assign the starting value to the counter variable i. Usually we assign it to 0.
What else starts with 0? The index of a list!



```
for (var i= 0; i<list.length; i++){
    var element= list[ i ];
```

Unit 5 Lesson 9 - Activity

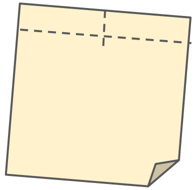
This is the **Boolean expression** that is checked to see if the loop should continue to run. When working with a list, we want to access every element in the list, so we are going to keep the loop going until we have hit **list.length.**

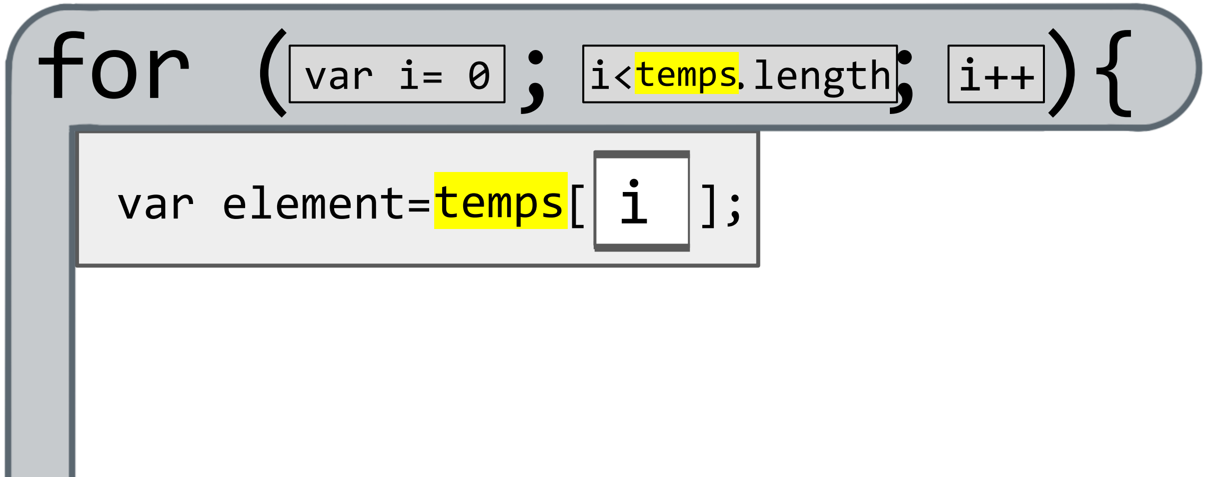


```
for (var i= 0; i<list.length; i++){
    var element= list[ i ];
```

Do This: replace **list** with the name of your list. In this case, we would replace it with **temps**.

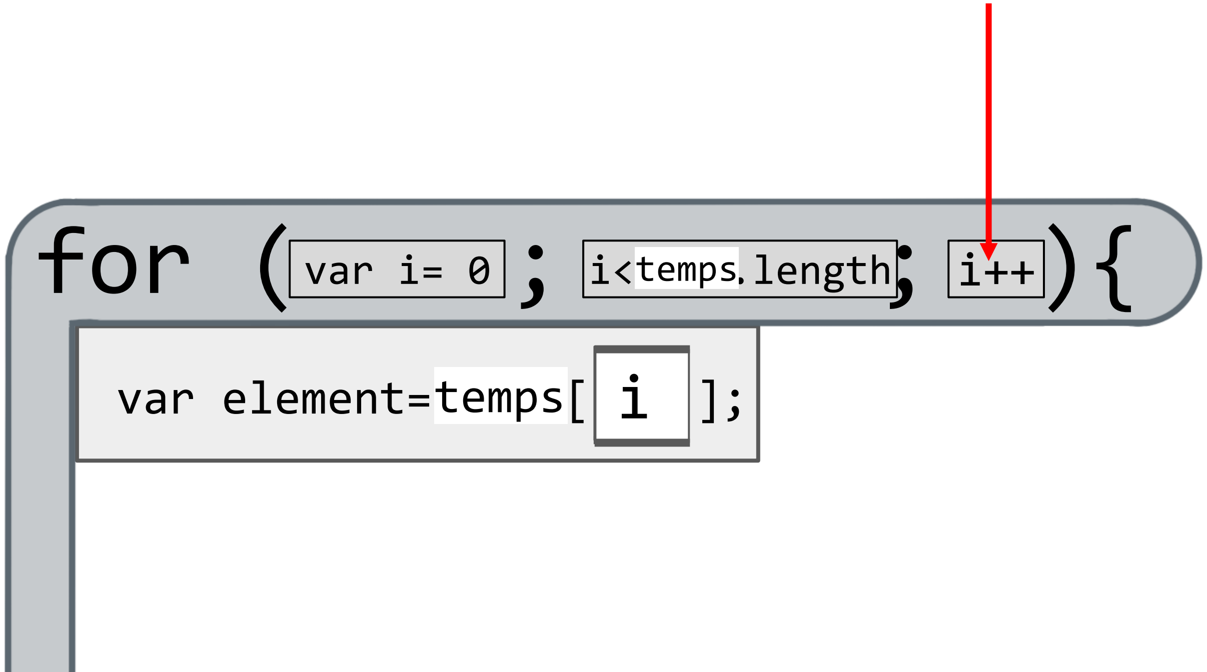
- Take two small pieces of a sticky note and write **temps** on them.
- Replace **list** with temps.





```
for (var i= 0; i<temps.length; i++){
    var element=temps[ i ];
```

The counter variable **i** increases after each loop runs.

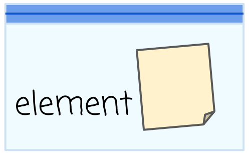


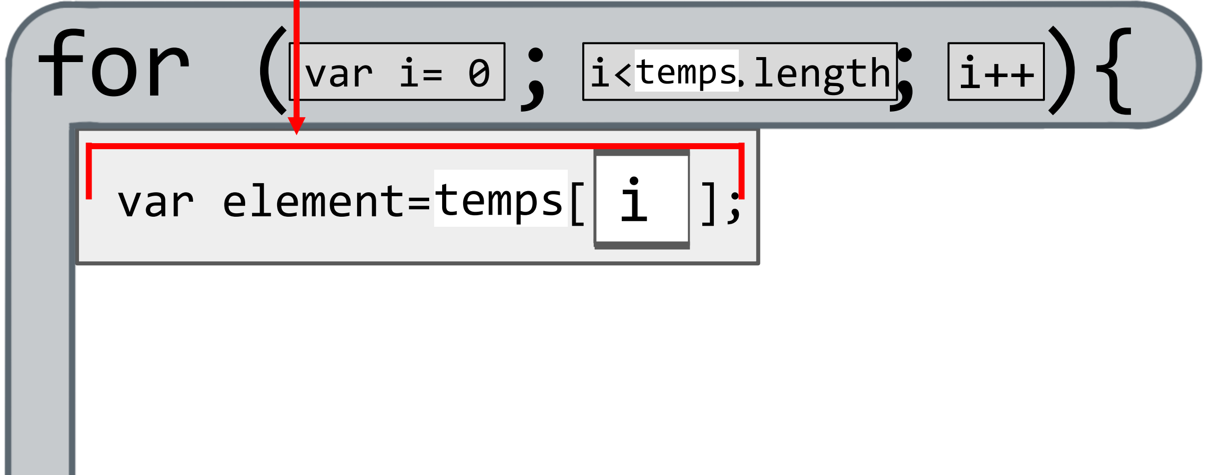
```
for (var i= 0; i<temps.length; i++){
    var element=temps[ i ];
```

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.

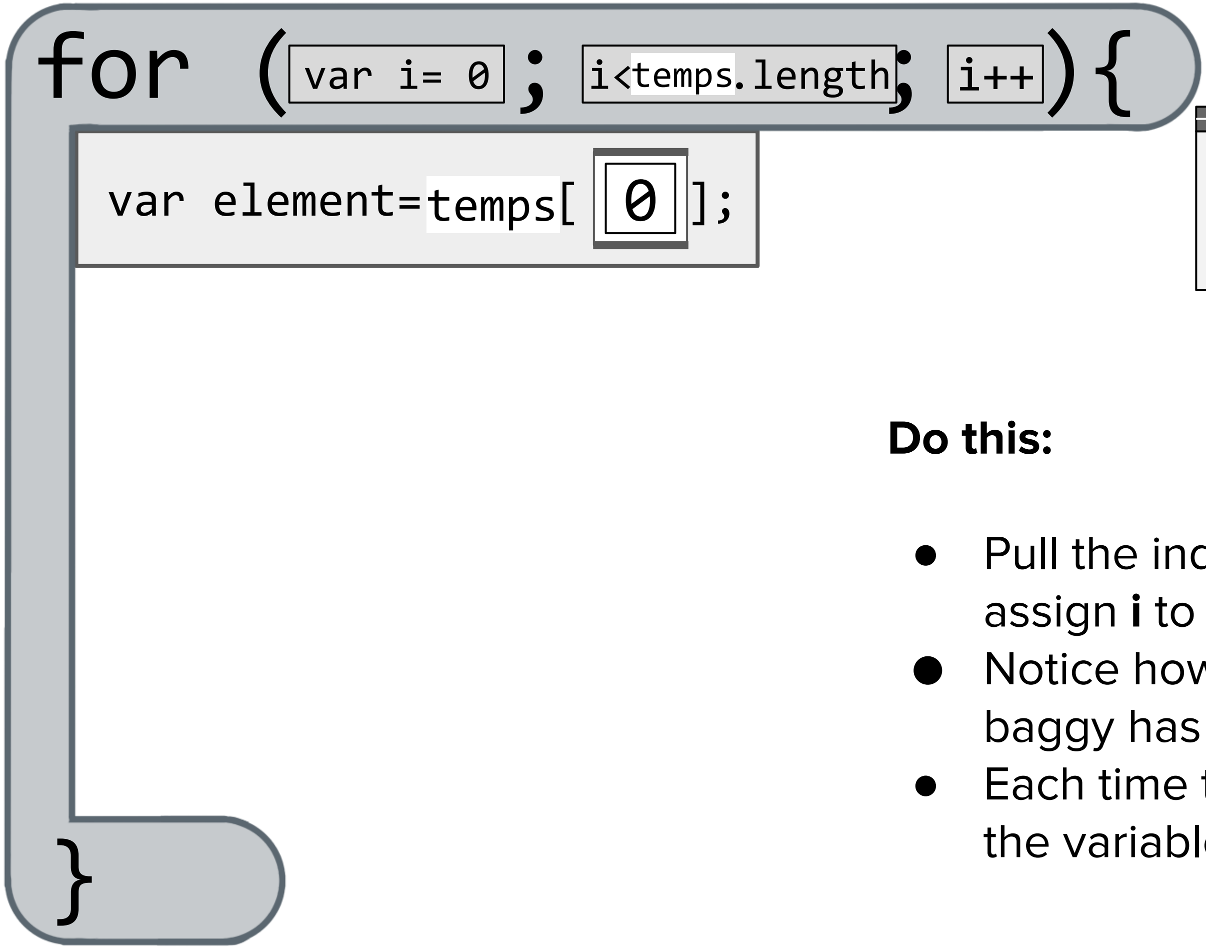
Do This:

- Set up a variable baggy and give it the name element

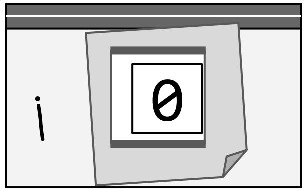




```
for (var i= 0; i<temps.length; i++){
    var element=temps[ i ];
```

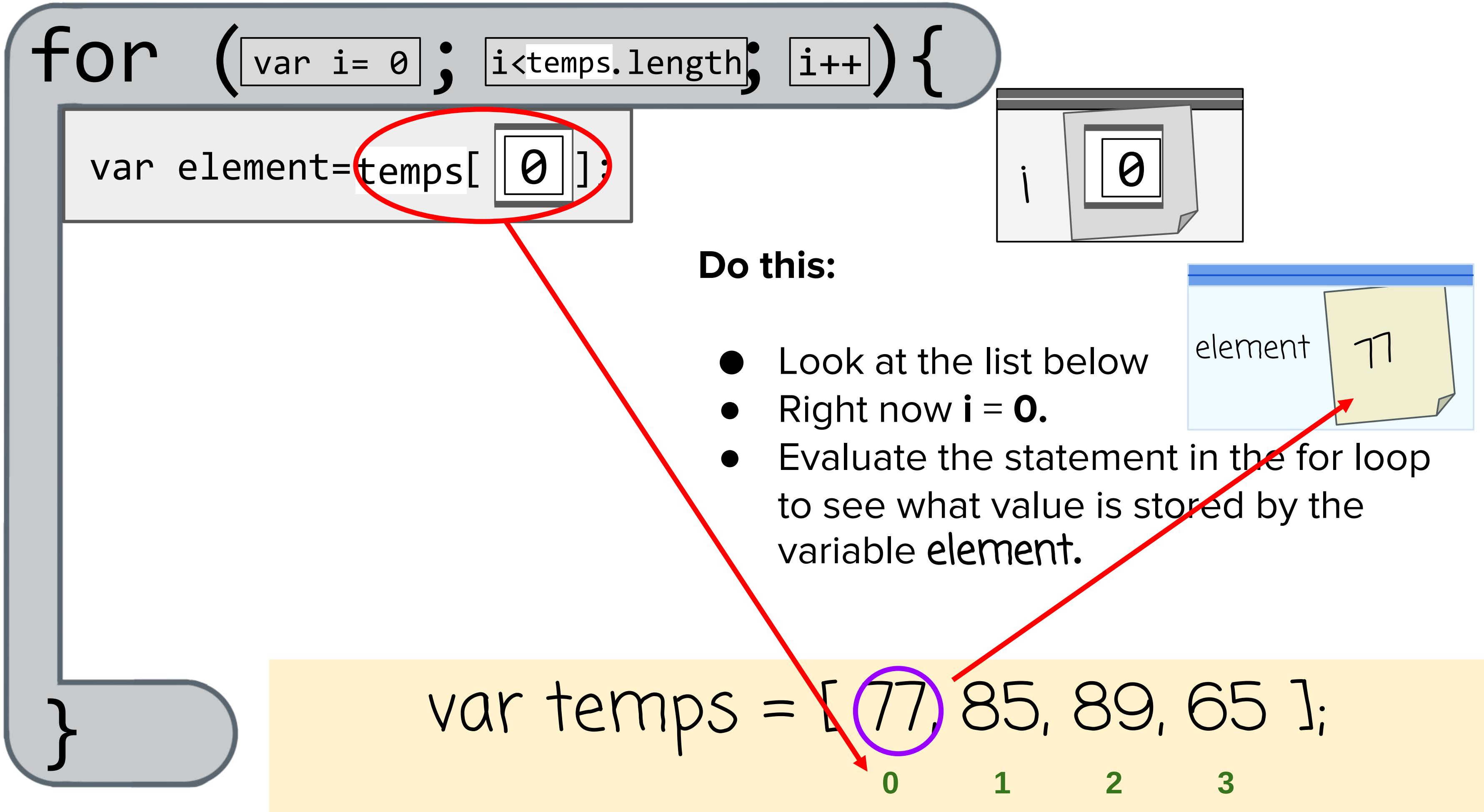


```
for (var i= 0; i<temps.length; i++){
    var element=temps[ 0 ];
}
```



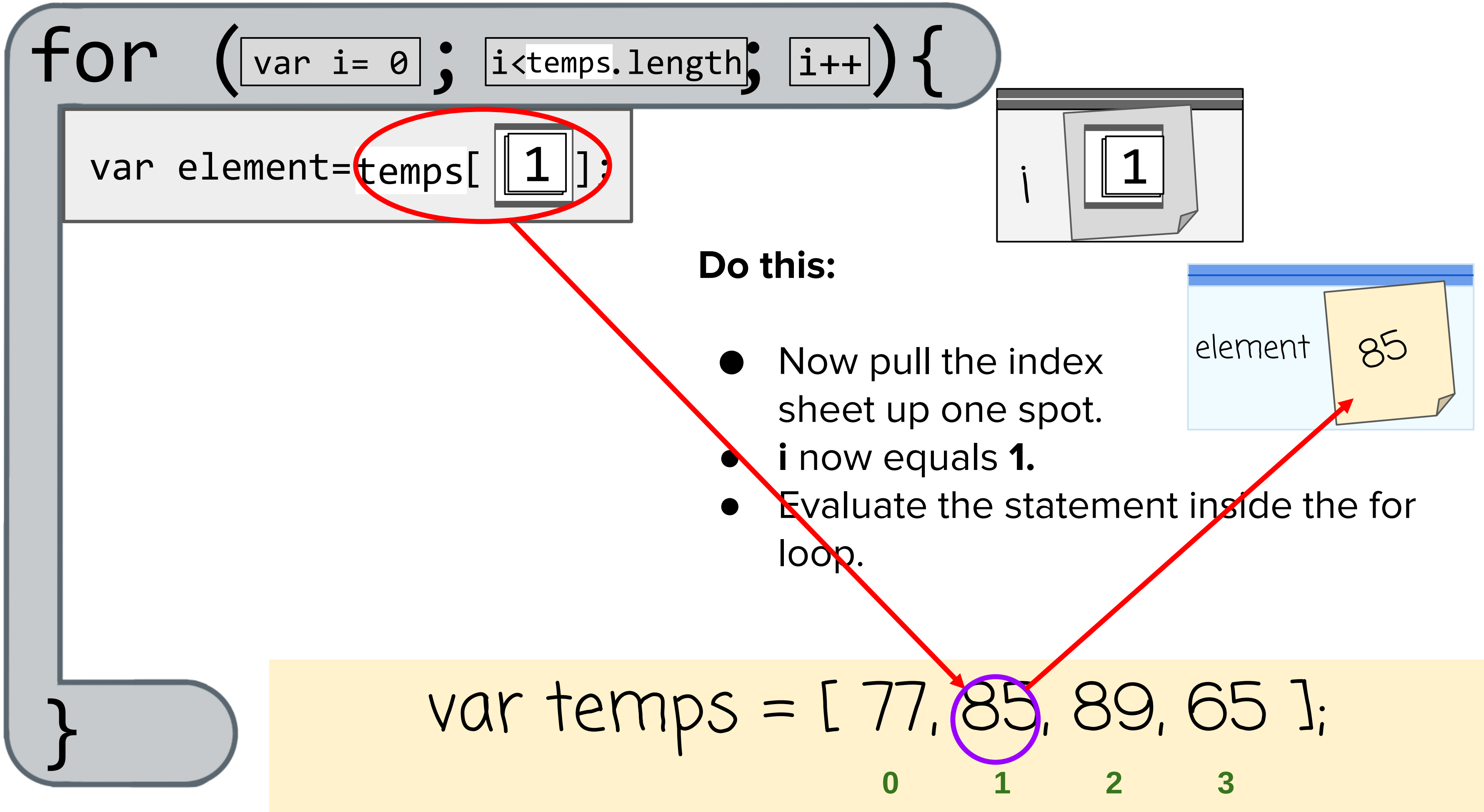
Do this:

- 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.


for (var i= 0 ; i<temps.length; i++){
var element=temps[0];
}
i
0
element
77
var temps = [77, 85, 89, 65];
0
1
2
3

Do this:

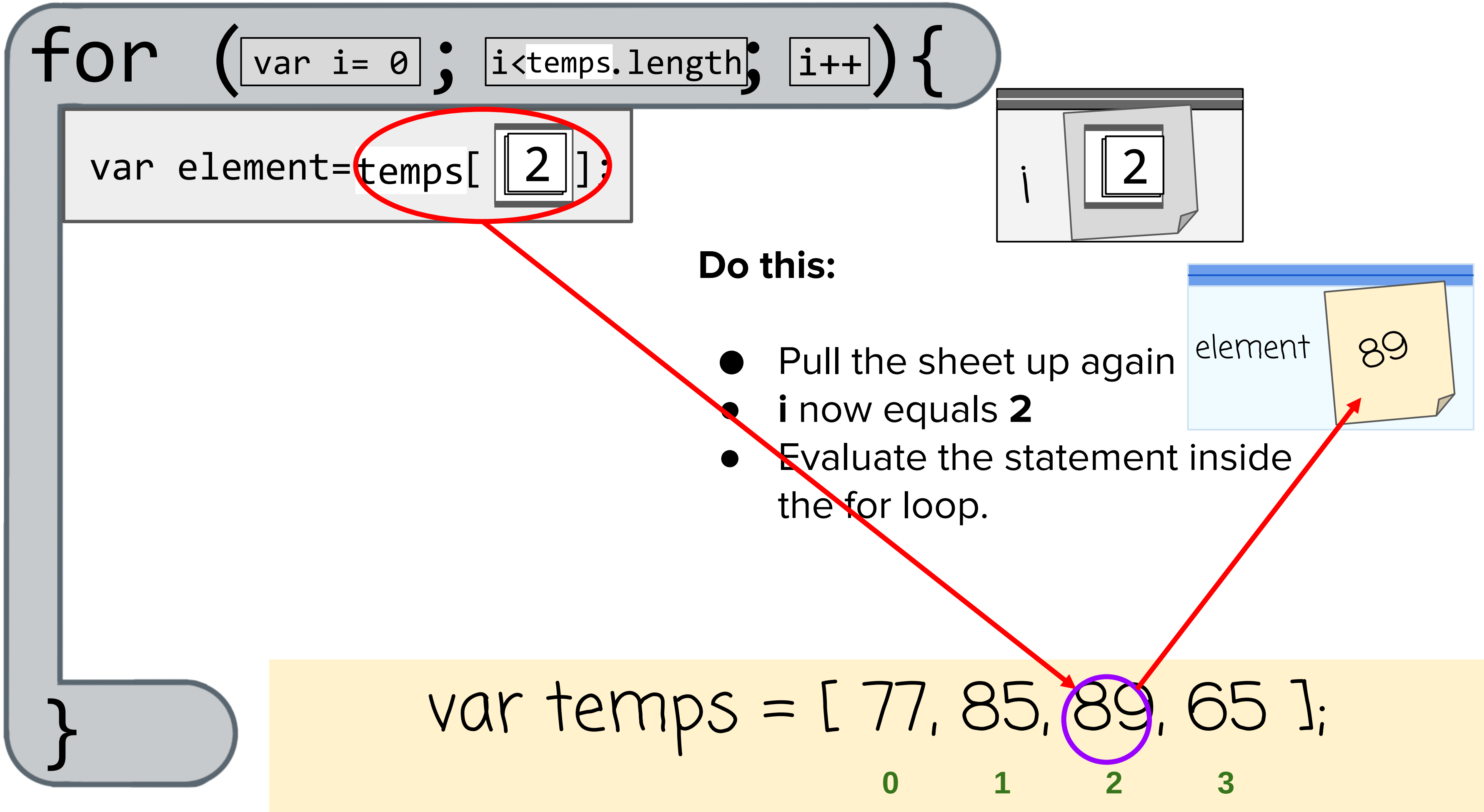
- Look at the list below
- Right now **i** = **0.**
- Evaluate the statement in the for loop to see what value is stored by the variable element.


for (var i= 0 ; i<temps.length; i++){
var element=temps[1];
}
i
1
element
85
var temps = [77, 85, 89, 65];
0
1
2
3

Do this:

- Now pull the index sheet up one spot.
- **i** now equals **1.**
- Evaluate the statement inside the for loop.

Unit 5 Lesson 9 - Activity



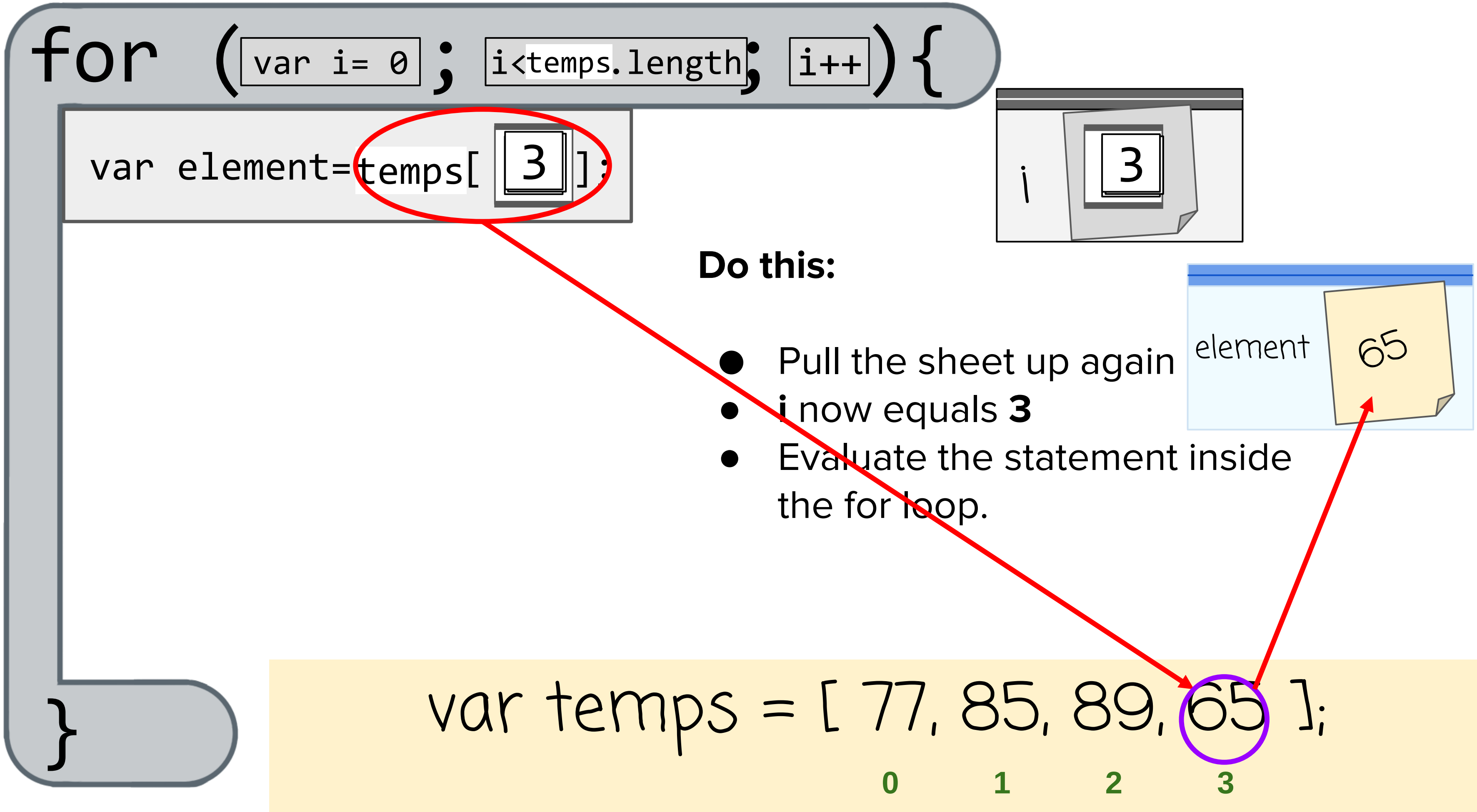
```
for (var i= 0; i<temps.length; i++){
    var element=temps[ 2 ];
}
```

```
var temps = [ 77, 85, 89, 65 ];
```

Do this:

- Pull the sheet up again
- **i** now equals **2**
- Evaluate the statement inside the for loop.

Unit 5 Lesson 9 - Activity

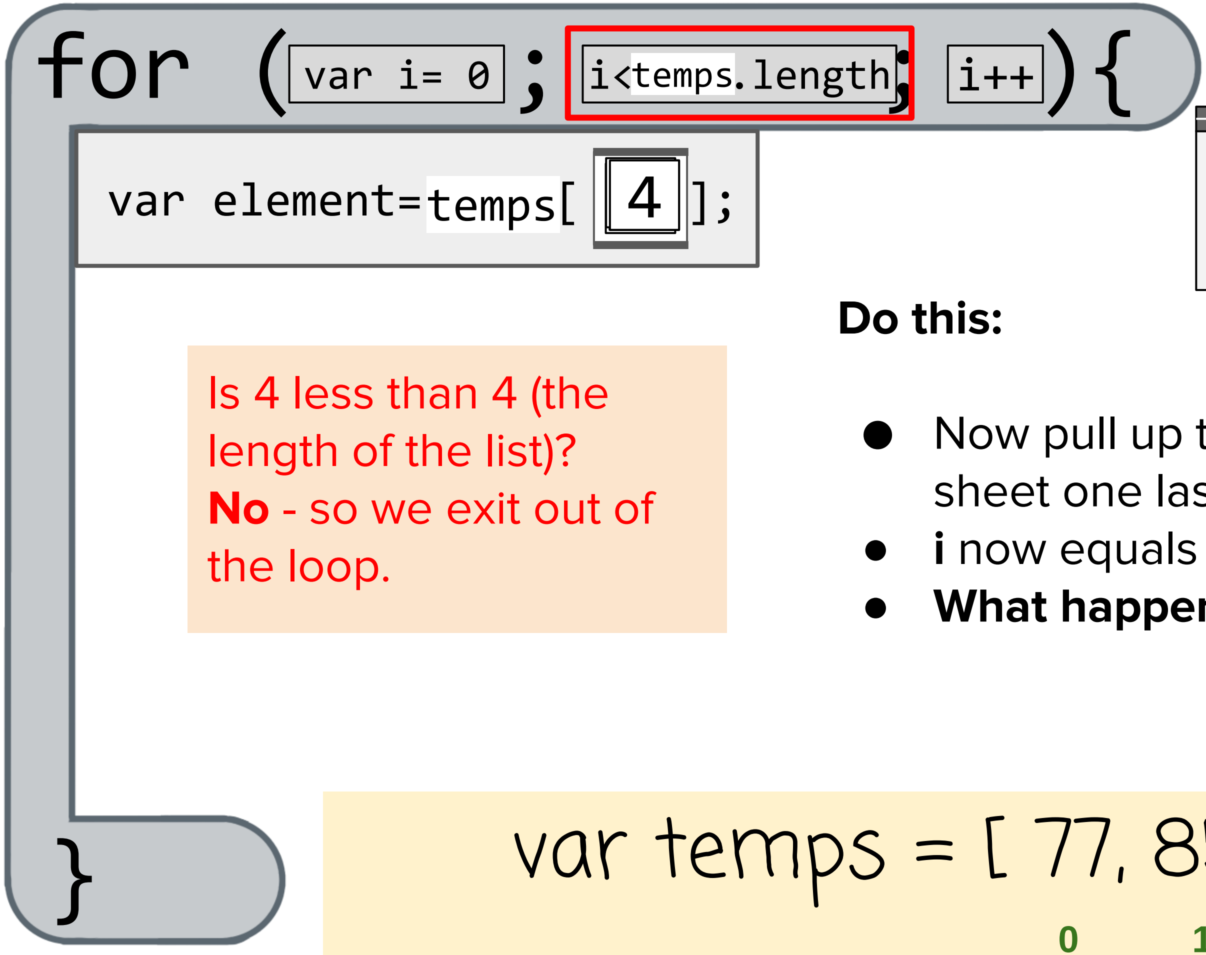


```
for (var i= 0; i<temps.length; i++){
    var element=temps[ 3 ];
}
```

```
var temps = [ 77, 85, 89, 65 ];
```

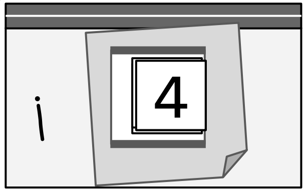
Do this:

- Pull the sheet up again
- **i** now equals **3**
- Evaluate the statement inside the for loop.



```
for (var i= 0; i<temps.length; i++){
    var element=temps[ 4 ];
}
```

Is 4 less than 4 (the length of the list)?
No - so we exit out of the loop.



Do this:

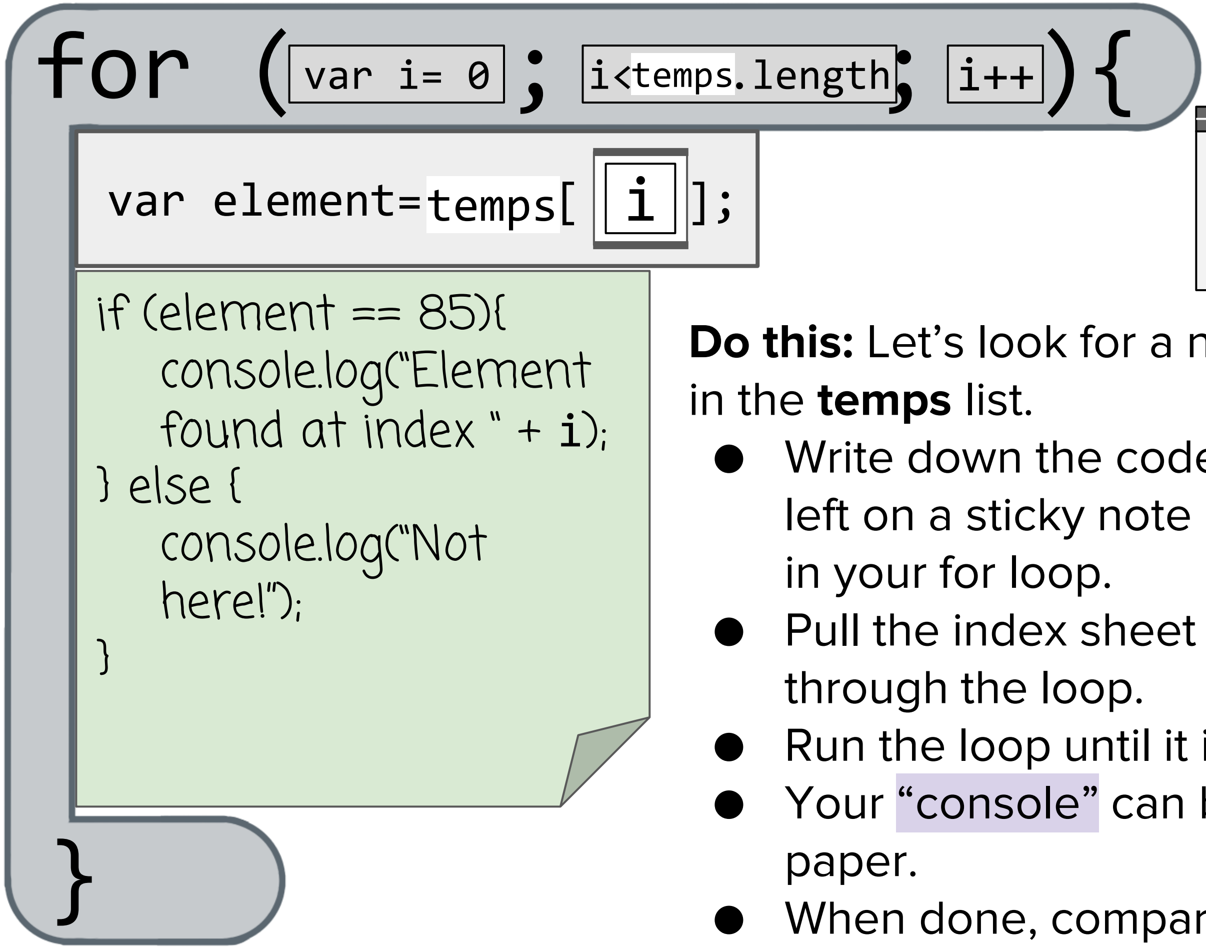
- Now pull up the index sheet one last time.
- **i** now equals **4**
- **What happens next?**

```
var temps = [ 77, 85, 89, 65 ];
```

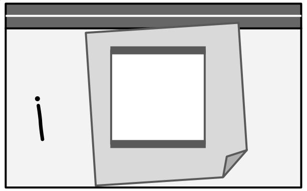
0	1	2	3

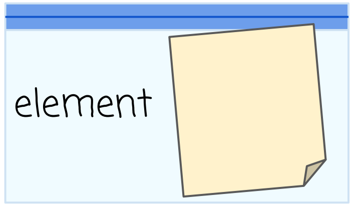
Are you starting to see a pattern here?

- 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.



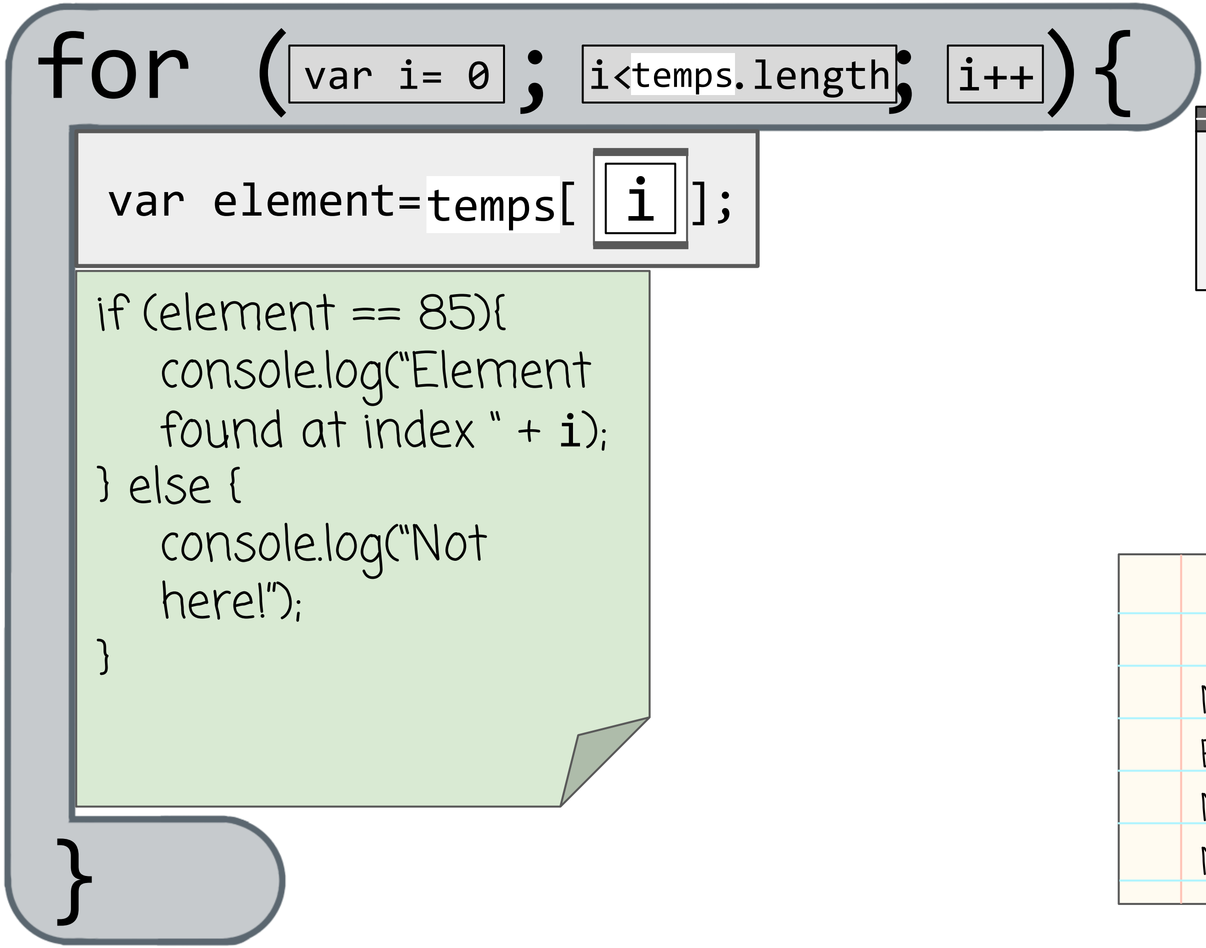
```
for (var i= 0; i<temps.length; i++){
  var element=temps[ i ];
  if (element == 85){
      console.log("Element
      found at index " + i);
  } else {
      console.log("Not
      here!");
  }
}
```



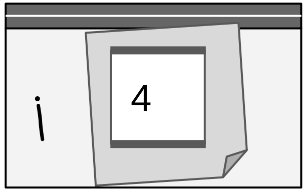


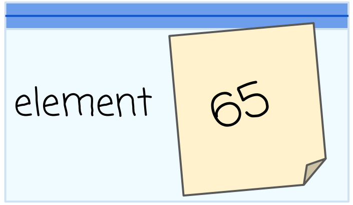
Do this: Let's look for a number in the **temps** list.

- 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



```
for (var i= 0; i<temps.length; i++){
    var element=temps[ i ];
    if (element == 85){
        console.log("Element found at index " + i);
    } else {
        console.log("Not here!");
    }
}
```

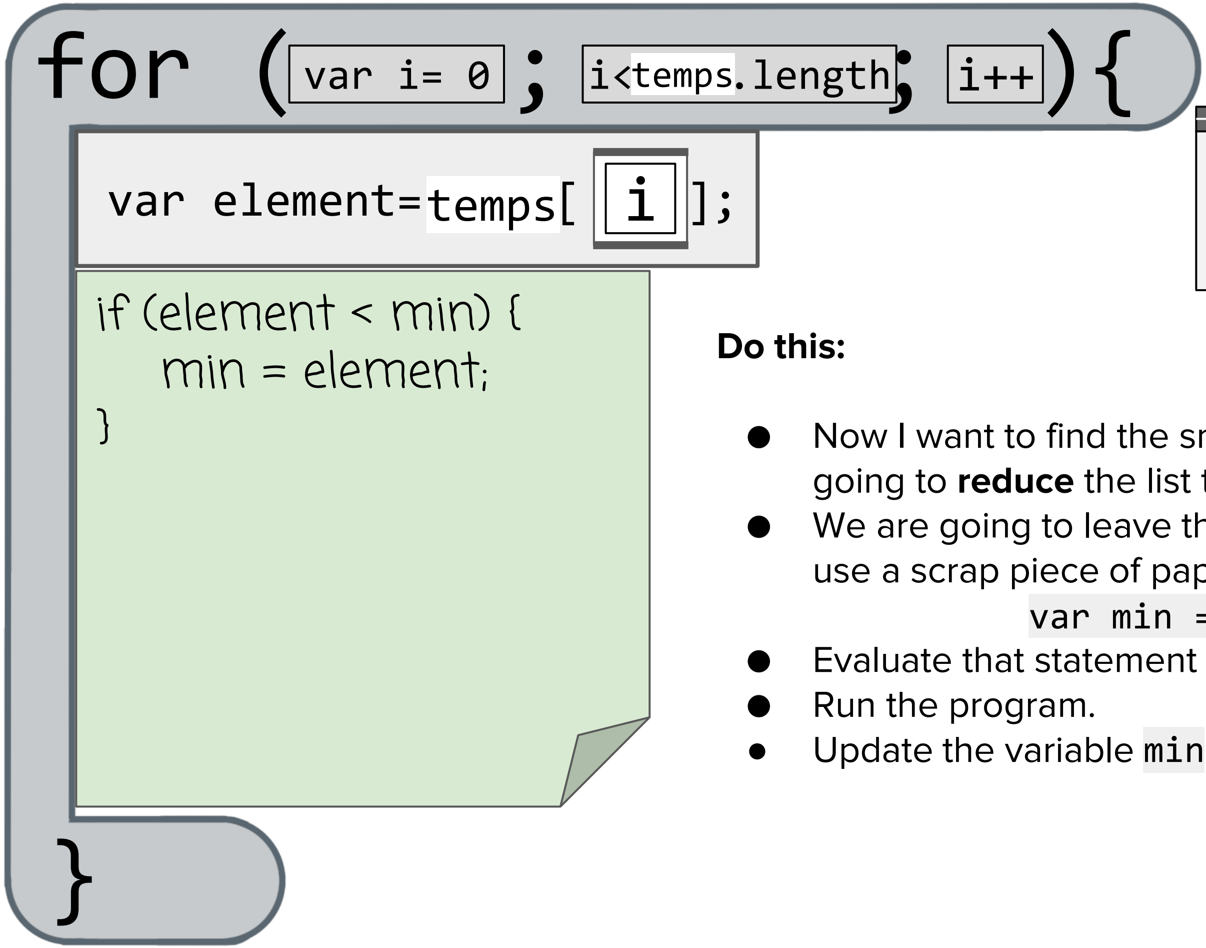




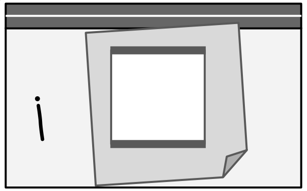
My Console

```
Not here!
Element found at index 1
Not here!
Not here!
```

Unit 5 Lesson 9 - Activity



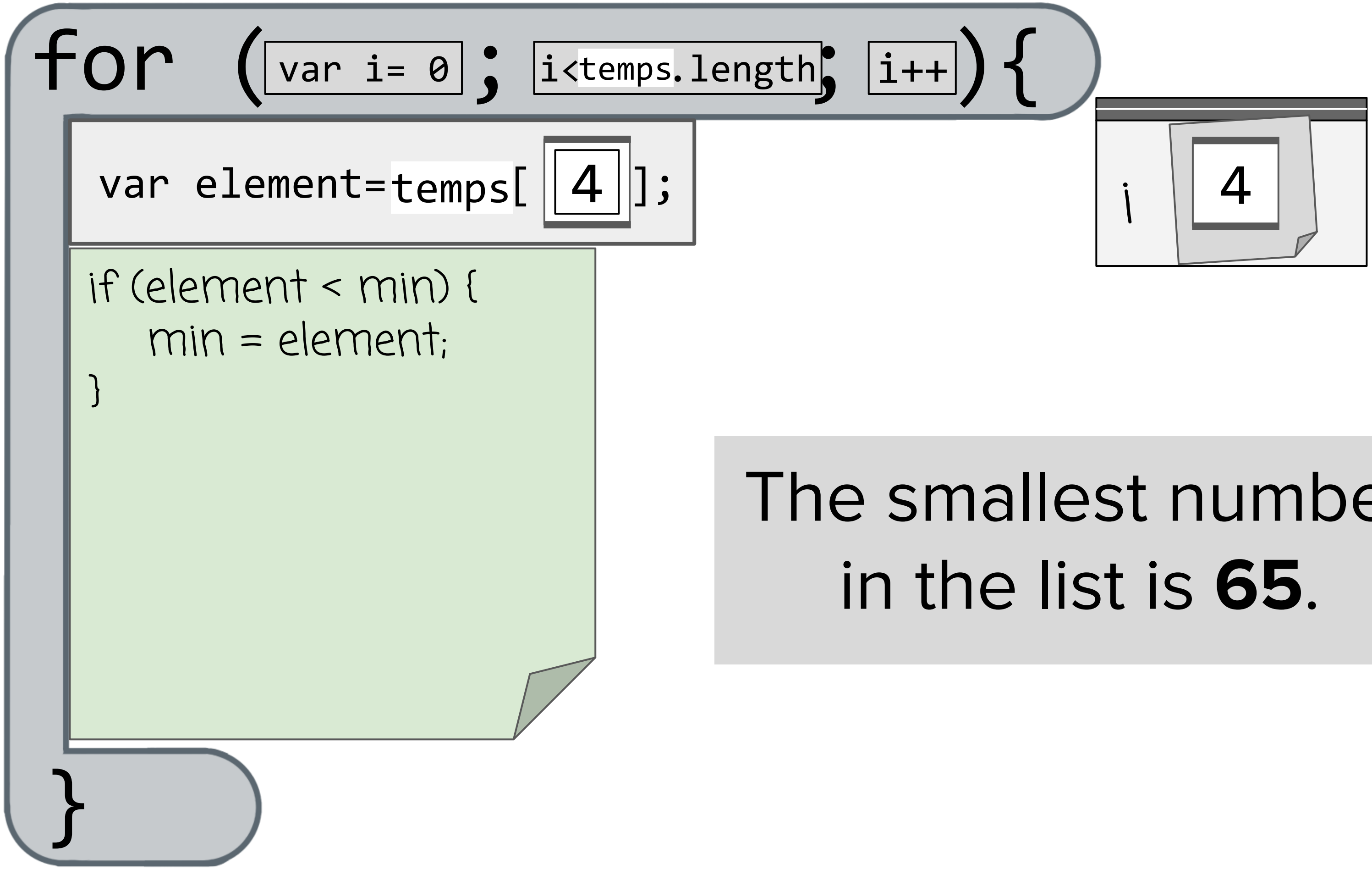
```
for (var i= 0; i<temps.length; i++){
    var element=temps[ i ];
    if (element < min) {
        min = element;
    }
}
```



Do this:

- Now I want to find the smallest number in my 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:

  `var min = temps[0];`
- Evaluate that statement to find the value stored in min.
- Run the program.
- Update the variable `min` as needed.



```
for (var i= 0; i<temps.length; i++){
    var element=temps[ 4 ];
    if (element < min) {
        min = element;
    }
}
```

The smallest number in the list is **65**.

Unit 5 Lesson 9 - Activity

Do this: Let's make it more challenging.

- 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:
  - store the global variable on a scrap piece of paper:

    ```
    var max = temps[0];
    ```
  - 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!

"Marnie, what number is at temps index 0?"

- Run one round of the for loop. If the value is greater than what is already stored in **max**, update **max**. In this case, it is not greater than (it is equal) so continue on.

"Marnie, what numbers is at temps index 1?"

- And so on and so forth ... until you get to the end of the list

Do this: I have a list of animals from which I want to **filter** some elements and store them in another list.

Setup:

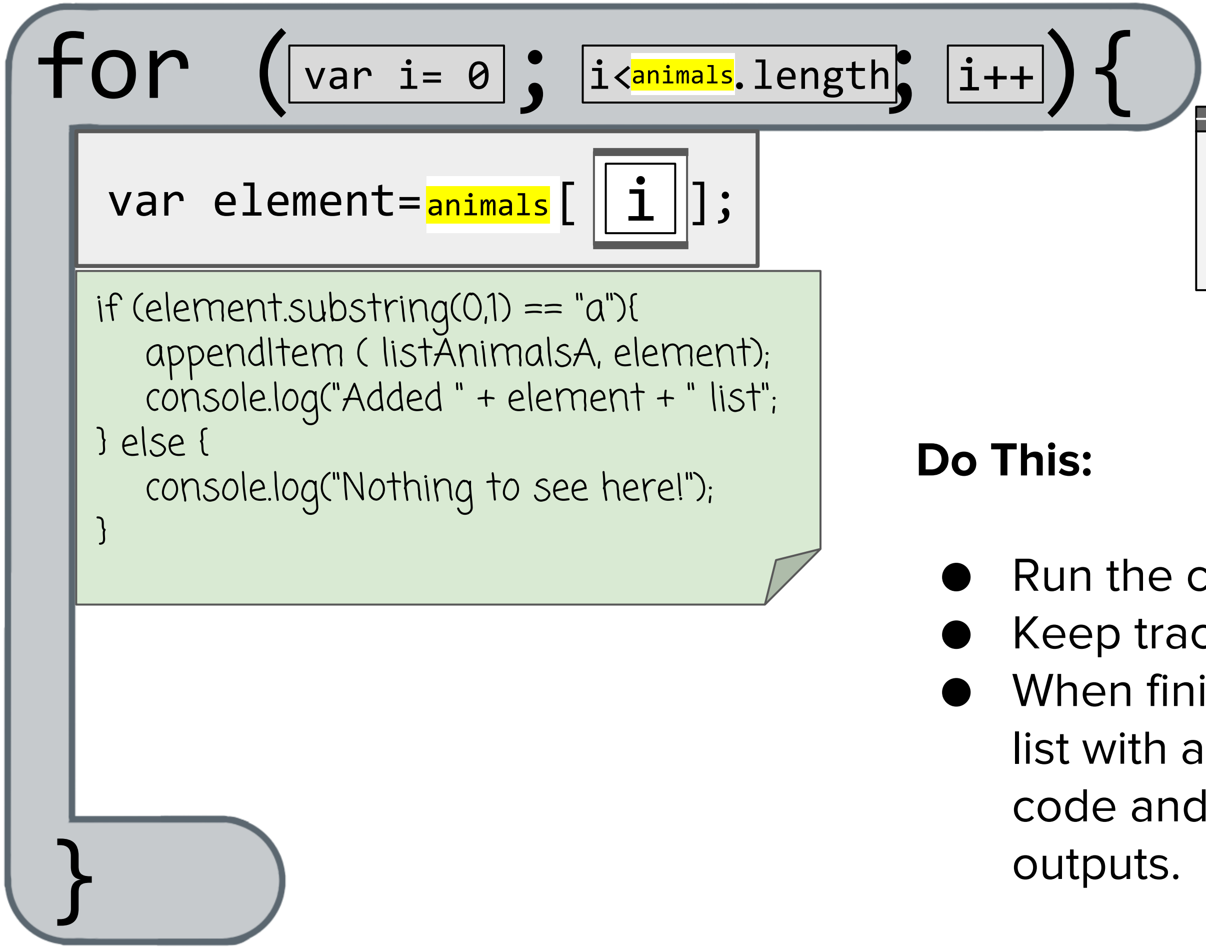
- One partner creates a list of eight random animal names, and stores it in the list `animals` on a scrap of paper. Keep this list hidden.
- The other partner uses a separate scrap of paper to track a new list called `animalsListA`.

```
var animals = [..write your own!..]
```

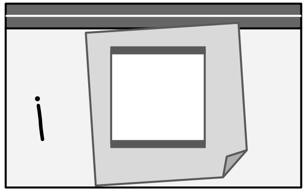
```
var animalsListA =
```

- 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 in the Traversal Machine.

```
if (element.substring(0,1) == "a"){
    appendItem ( listAnimalsA, element);
    console.log("Added " + element + " list";
} else {
    console.log("Nothing to see here!");
}
```



```
for (var i= 0; i<animals.length; i++){
    var element=animals[ i ];
    if (element.substring(0,1) == "a"){
        appendItem ( listAnimalsA, element);
        console.log("Added " + element + " list";
    } else {
        console.log("Nothing to see here!");
    }
}
```



Do This:

- 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.

Wrap Up

Key Takeaways

- We use a **for loop** to traverse a list, visiting each element one at a time.
- Each pass through the loop, the counting variable changes - usually by one. We can access each element by evaluating `list[i]` in the for loop.
- Once we know the element at each spot in a list, we can do different things with it:
  - **Filter** (create a subset of elements from the original list)
  - **Reduce** (reduce the list down to a single element, for example: the smallest number in the list)

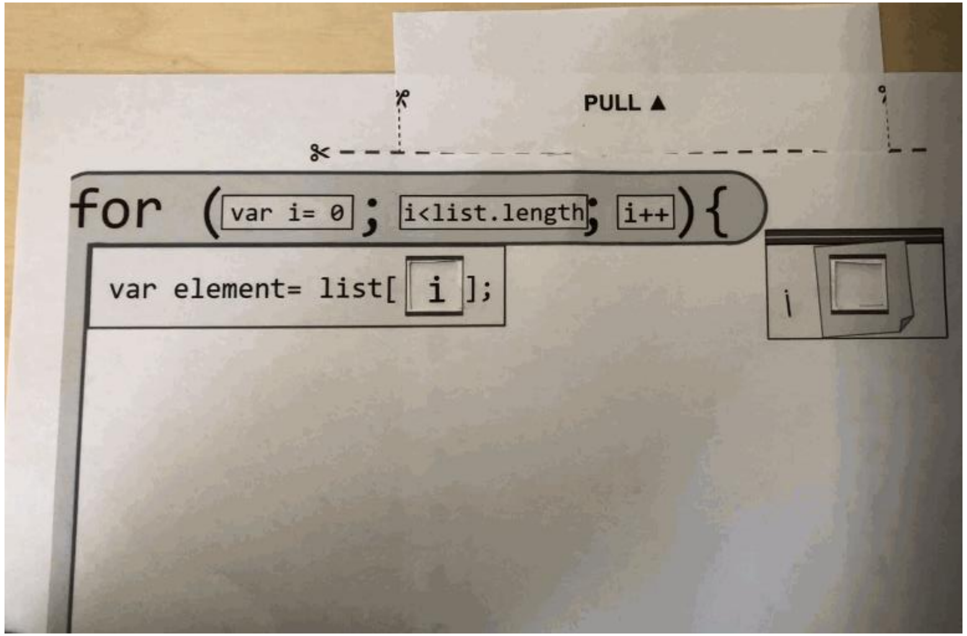
Unit 5 Lesson 9 - Wrap Up

Traversal: the process of accessing each item in a list one at a time

Unit 5 - Lesson 10
Traversals Investigate

Warm Up

● ○ ○



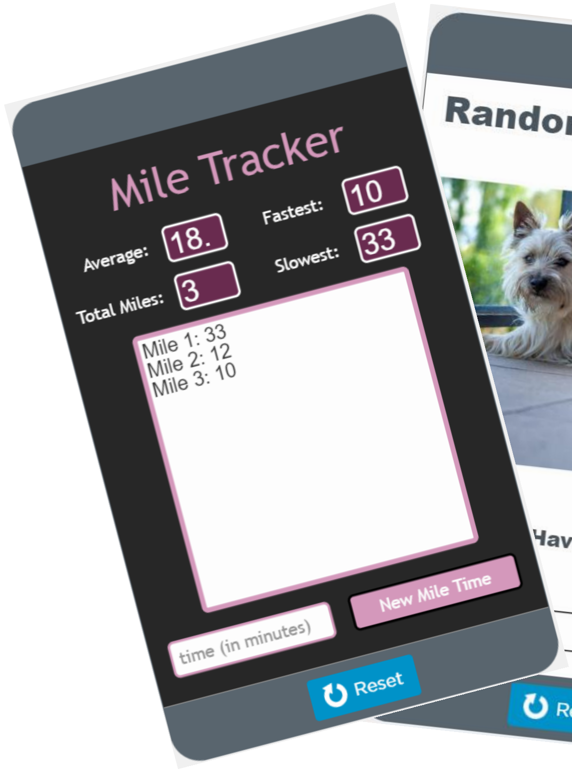
Review:

The Traversal Machine - How did it work?

Activity

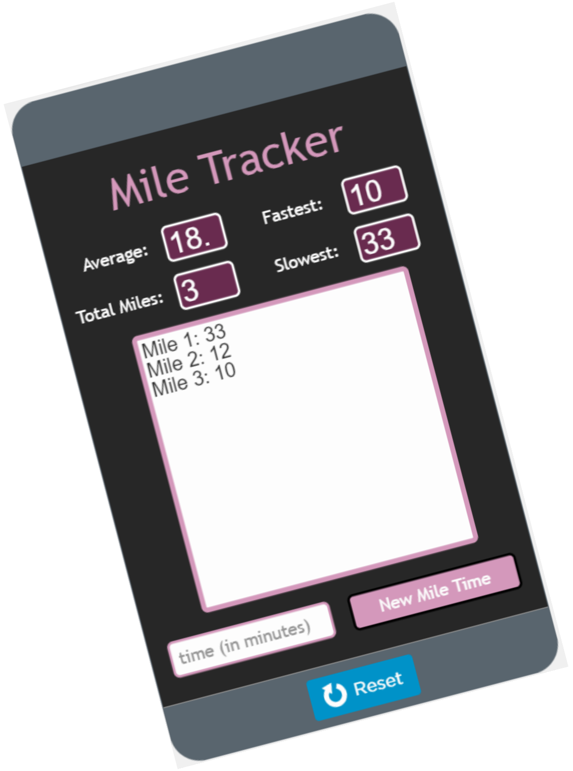
●●○

Traversals Investigate



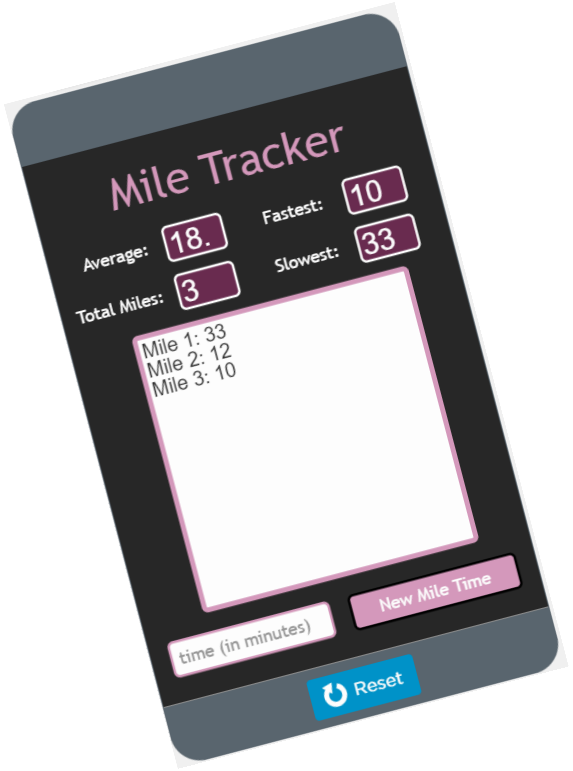
Do This:

- Get into groups of four. Assign each student in a group a different letter (A, B, C, D)
- Navigate to Level 2



Level 2:

- Read through the entire code
- Each student focuses their attention on a single function. Re-read the code for that 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)



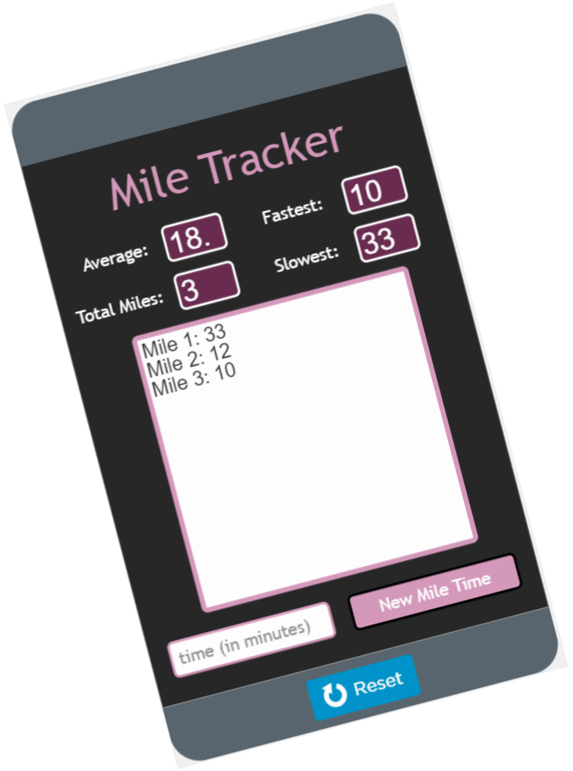
Mile 1: 33
Mile 2: 12
Mile 3: 10

Discuss in your group:

- How does your function work?
  - What list does it use?
  - How is it traversed using a for loop?

Discuss as a class:

- How does the app work?



```
Mile 1: 33
Mile 2: 12
Mile 3: 10
```

Modify with a partner:

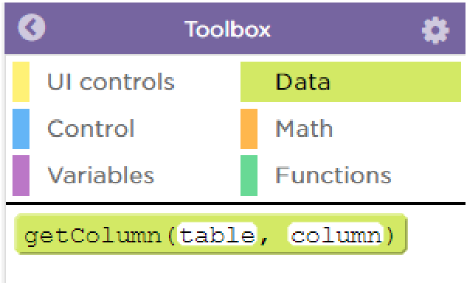
- Create a function that adds together the total time of every element in the list.
- `console.log` the result.
- Call this function in the `updateScreen()` function.

Do This:

- Navigate to Level 3 with a partner
- Read through the code and run the app. What looks new?
- Open the data tab. Investigate!
  - How does it work?
  - What information is available?
  - What can you change? What can't you change?
  - Investigate a few different datasets.

Do This:

- Modify the code!
- Choose a dataset and import it to your project.
- Look at the code. Check out the "data" drawer in the toolbox.
- Modify to print out a column of data from your chosen dataset.

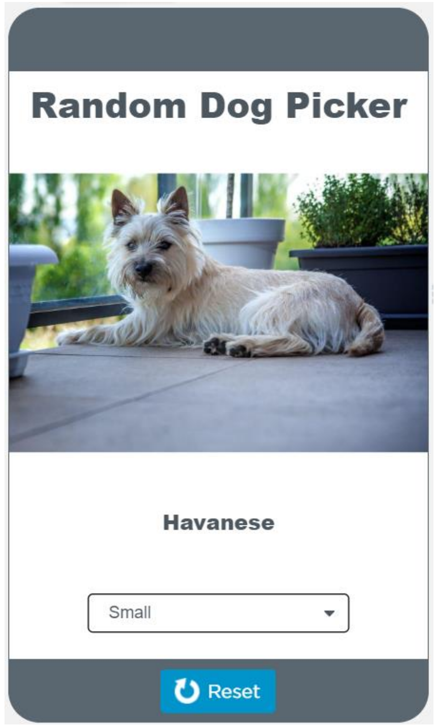


```
getColumn(table, column)
```

Do This:

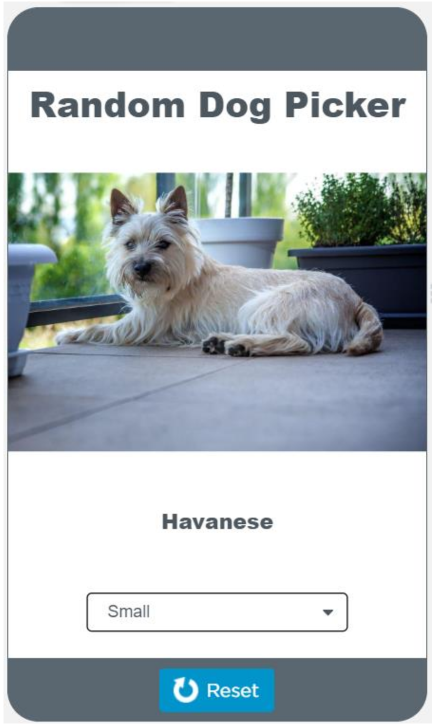
- Navigate to Level 4 with a partner
- Follow the directions on the screen

Note: Read the code very carefully!



Discuss as a Class:

- What are the names of the five lists in this program?
- On what lines of code are the lists created?
- On what lines of code are the lists filled?
- How are these lists filled?
- Open up the data tab and click to view the dogs table. What columns does this app use?
- Look at the filter function. On what lines are the filtered lists reset to blank lists?
- What condition is being checked to determine if an element belongs in a filtered list?



Wrap Up

Prompt:

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

Unit 5 - Lesson 11
Traversals Practice

Warm Up

● ○ ○

Activity

● ● ○

Debugging: the process of finding and fixing problems in code

Describe
The Problem

What do you expect it to do?
What does it actually do?
Does it always happen?

Hunt
For Bugs

Are there warnings or errors?
What did you change most recently?
Explain your code to someone else
Look for code related to the problem

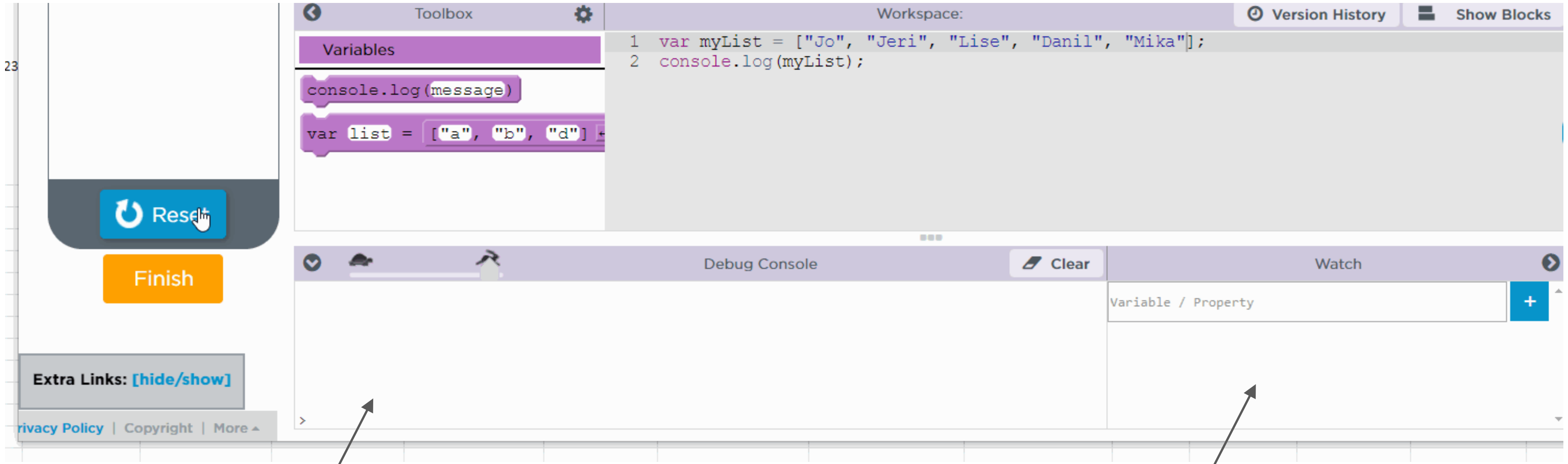
Try
Solutions

Make a small change

Document
As You Go

What have you learned?
What strategies did you use?
What questions do you have?

Debugging Lists



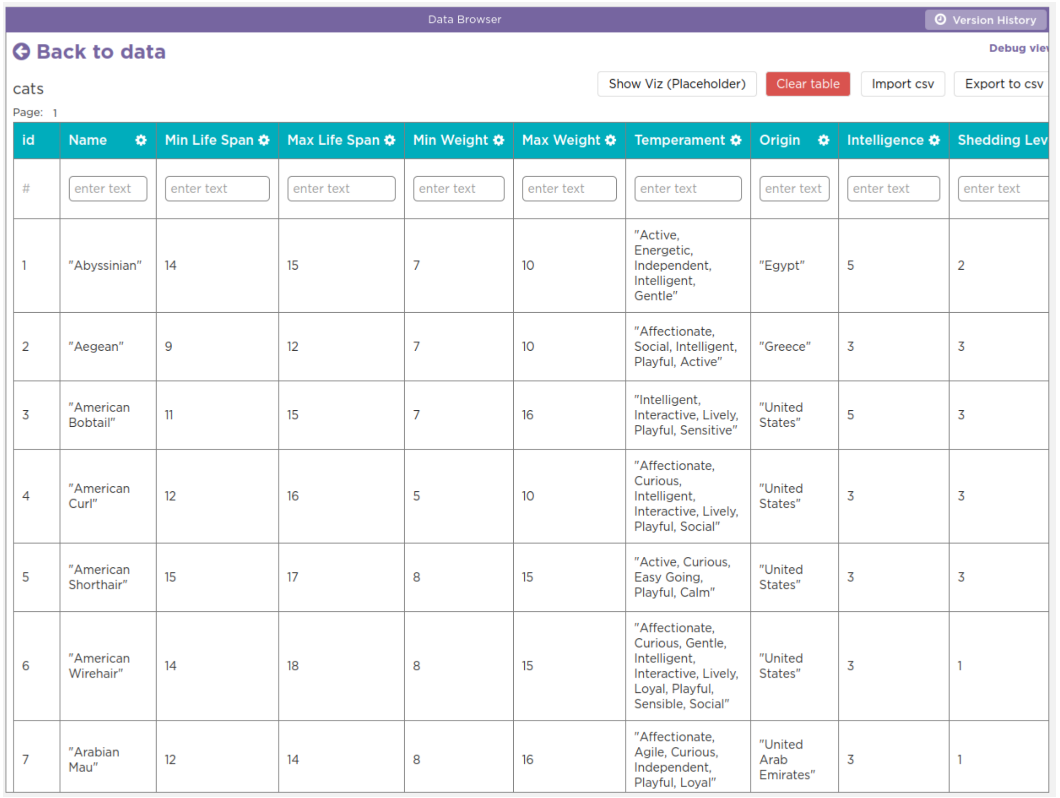
```
var myList = ["Jo", "Jeri", "Lise", "Danil", "Mika"];
console.log(myList);
```

```
console.log(message)
var list = ["a", "b", "d"]
```

Use `console.log` to track lists in the Debug Console. Click the arrow to see elements listed by index. **Don't forget the last index is `length-1`**

Add the list to the Watcher panel to track changes to the list as the app runs. The Watcher also shows the length of the list.

Debugging and the Data Tab



cats

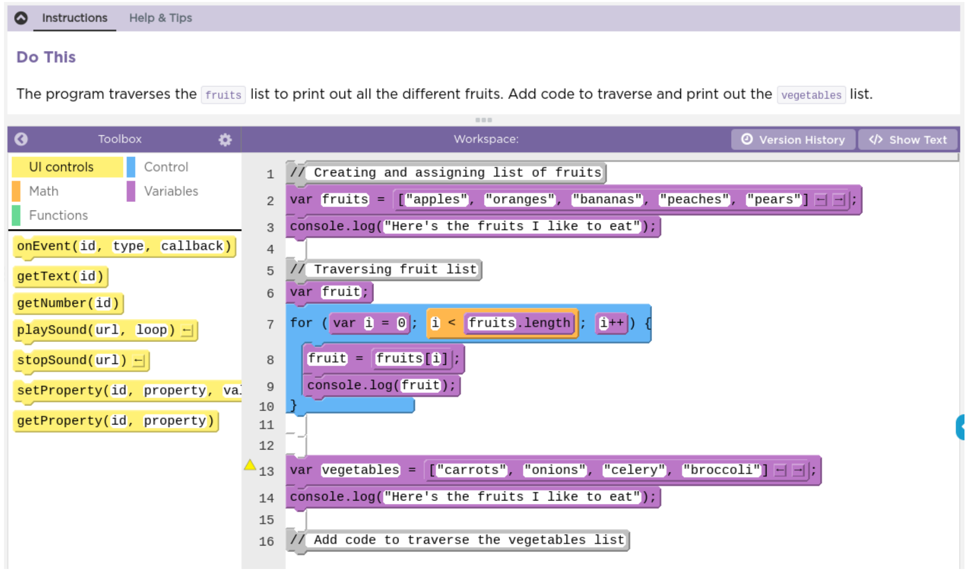
Page: 1

id	Name	Min Life Span	Max Life Span	Min Weight	Max Weight	Temperament	Origin	Intelligence	Shedding Lev
#	enter text	enter text	enter text	enter text	enter text	enter text	enter text	enter text	enter text
1	"Abyssinian"	14	15	7	10	"Active, Energetic, Independent, Intelligent, Gentle"	"Egypt"	5	2
2	"Aegean"	9	12	7	10	"Affectionate, Social, Intelligent, Playful, Active"	"Greece"	3	3
3	"American Bobtail"	11	15	7	16	"Intelligent, Interactive, Lively, Playful, Sensitive"	"United States"	5	3
4	"American Curl"	12	16	5	10	"Affectionate, Curious, Intelligent, Interactive, Lively, Playful, Social"	"United States"	3	3
5	"American Shorthair"	15	17	8	15	"Active, Curious, Easy Going, Playful, Calm"	"United States"	3	3
6	"American Wirehair"	14	18	8	15	"Affectionate, Curious, Gentle, Intelligent, Interactive, Lively, Loyal, Playful, Sensible, Social"	"United States"	3	1
7	"Arabian Mau"	12	14	8	16	"Affectionate, Agile, Curious, Independent, Playful, Loyal"	"United Arab Emirates"	3	1

Go actually look at your data!

Use `console.log` to make sure you're actually getting the information you want from your tables

Traversals Practice



Instructions Help & Tips

Do This

The program traverses the `fruits` list to print out all the different fruits. Add code to traverse and print out the `vegetables` list.

Toolbox: UI controls, Control, Math, Variables, Functions

```
onEvent(id, type, callback)
getText(id)
getNumber(id)
playSound(url, loop)
stopSound(url)
setProperty(id, property, va
getProperty(id, property)
```

Workspace: Version History, Show Text

```
// Creating and assigning list of fruits
var fruits = ["apples", "oranges", "bananas", "peaches", "pears"];
console.log("Here's the fruits I like to eat");

// Traversing fruit list
var fruit;
for (var i = 0; i < fruits.length; i++) {
  fruit = fruits[i];
  console.log(fruit);
}


var vegetables = ["carrots", "onions", "celery", "broccoli"];
console.log("Here's the fruits I like to eat");

// Add code to traverse the vegetables list
```

Lesson 11: Traversals Practice
Saved 6 days ago
2
MORE

Do This:

- Navigate to Lesson 11, Level 2 on Code Studio

Wrap Up

Prompt:

What aspects of working with traversals do you feel clicked today?

What do you still have trouble with?

Unit 5 - Lesson 12
Traversals Make

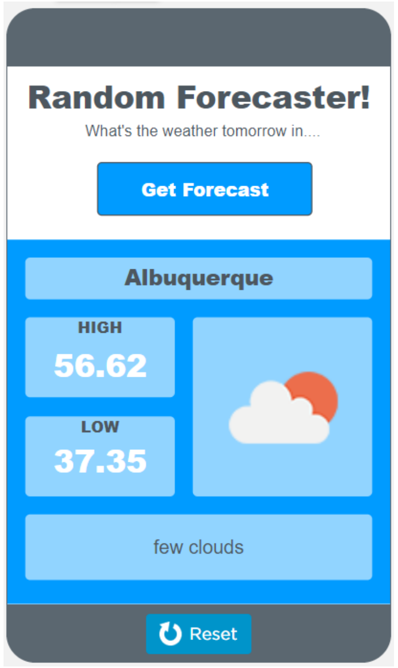
Warm Up

● ○ ○

Activity

●●○

Traversals Make: Random Forecaster



Do This:

- Navigate to Lesson 12, Level 2 on Code Studio

Do This:

- Click on "Get Forecast".
- Click the button several times to see how the display changes.

Prompts:

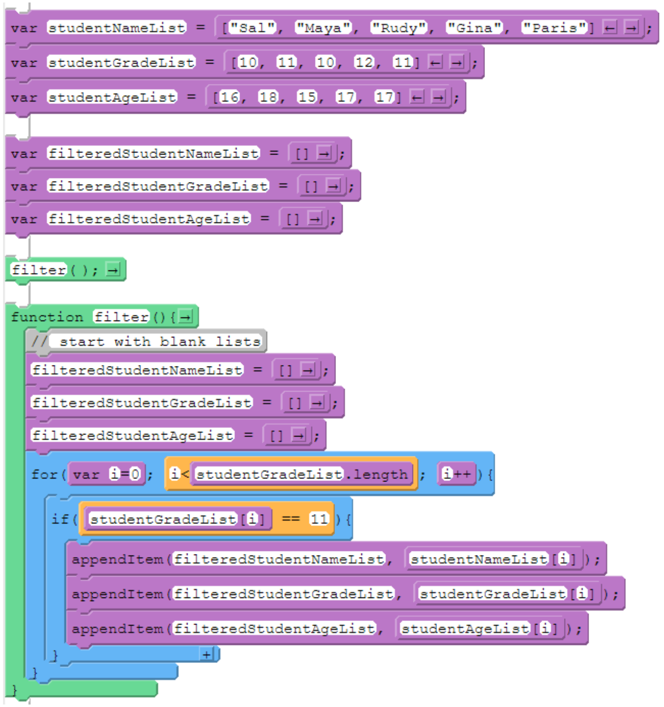
- What information is needed to create this app?
- What list filtering patterns might be used?

Do This: Make the Random Forecaster App!

Unit 5 Lesson 12

Activity Guide - Traversals Make

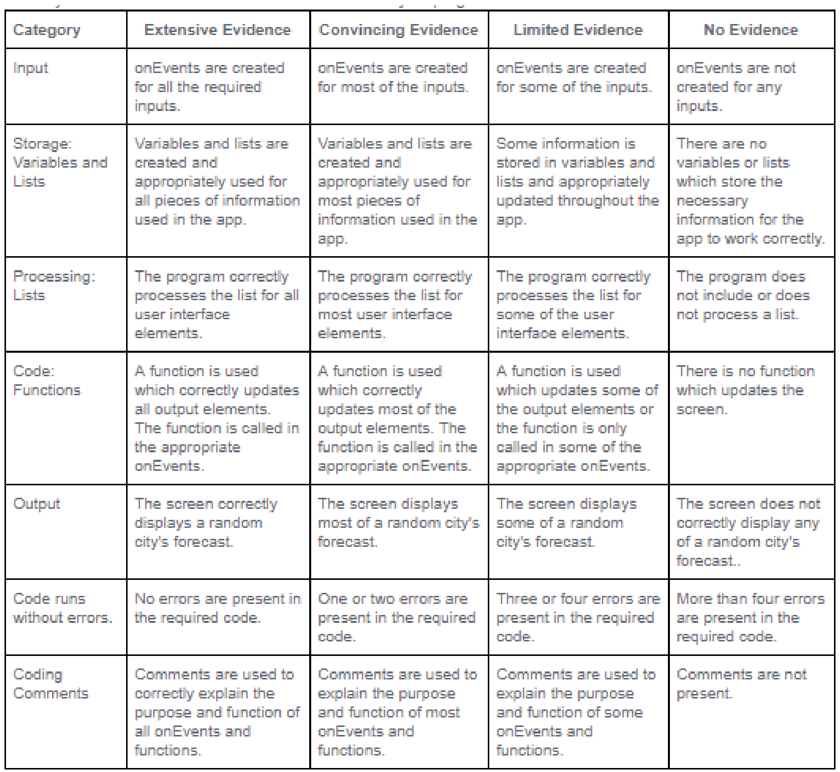
The List Filter Pattern: Filtering Multiple Lists


```
var studentNameList = ["Sal", "Maya", "Rudy", "Gina", "Paris"];
var studentGradeList = [10, 11, 10, 12, 11];
var studentAgeList = [16, 18, 15, 17, 17];

var filteredStudentNameList = [];
var filteredStudentGradeList = [];
var filteredStudentAgeList = [];

filter();

function filter(){
  // start with blank lists
  filteredStudentNameList = [];
  filteredStudentGradeList = [];
  filteredStudentAgeList = [];
  for(var i=0; i<studentGradeList.length; i++){
    if(studentGradeList[i] == 11){
      appendItem(filteredStudentNameList, studentNameList[i]);
      appendItem(filteredStudentGradeList, studentGradeList[i]);
      appendItem(filteredStudentAgeList, studentAgeList[i]);
    }
  }
}
```

```
var studentNameList = ["Sal", "Maya", "Rudy", "Gina", "Paris"];
var studentGradeList = [10, 11, 10, 12, 11];
var studentAgeList = [16, 18, 15, 17, 17];

var filteredStudentNameList = [];
var filteredStudentGradeList = [];
var filteredStudentAgeList = [];

filter();

function filter(){
  // start with blank lists
filteredStudentNameList = [];
filteredStudentGradeList = [];
filteredStudentAgeList = [];
for(var i=0; i<studentGradeList.length; i++){
    if(studentGradeList[i] == 11){
      appendItem(filteredStudentNameList, studentNameList[i]);
      appendItem(filteredStudentGradeList, studentGradeList[i]);
      appendItem(filteredStudentAgeList, studentAgeList[i]);
    }
  }
}
```

Unit 5 Lesson 12 - Activity

Don't forget to check the rubric before hitting submit!



Category	Extensive Evidence	Convincing Evidence	Limited Evidence	No Evidence
Input	onEvents are created for all the required inputs.	onEvents are created for most of the inputs.	onEvents are created for some of the inputs.	onEvents are not created for any inputs.
Storage: Variables and Lists	Variables and lists are created and appropriately used for all pieces of information used in the app.	Variables and lists are created and appropriately used for most pieces of information used in the app.	Some information is stored in variables and lists and appropriately updated throughout the app.	There are no variables or lists which store the necessary information for the app to work correctly.
Processing: Lists	The program correctly processes the list for all user interface elements.	The program correctly processes the list for most user interface elements.	The program correctly processes the list for some of the user interface elements.	The program does not include or does not process a list.
Code: Functions	A function is used which correctly updates all output elements. The function is called in the appropriate onEvents.	A function is used which correctly updates most of the output elements. The function is called in the appropriate onEvents.	A function is used which updates some of the output elements or the function is only called in some of the appropriate onEvents.	There is no function which updates the screen.
Output	The screen correctly displays a random city's forecast.	The screen displays most of a random city's forecast.	The screen displays some of a random city's forecast.	The screen does not correctly display any of a random city's forecast..
Code runs without errors.	No errors are present in the required code.	One or two errors are present in the required code.	Three or four errors are present in the required code.	More than four errors are present in the required code.
Coding Comments	Comments are used to correctly explain the purpose and function of all onEvents and functions.	Comments are used to explain the purpose and function of most onEvents and functions.	Comments are used to explain the purpose and function of some onEvents and functions.	Comments are not present.

Submit

Wrap Up

Great job today!

Unit 5 - Lesson 13
Project - Hackathon Part 1

Warm Up

● ○ ○

Activity

● ● ○

Group: Find a partner!

Distribute: Project Planning Guides

Project Description

For this project you will work with a partner. Together you will create an app that uses a dataset. There are two roles in this project: designer and programmer. You are both responsible for the overall project, but you will focus on different parts of the project at any given time. On the last day of the project, you will individually complete a written response that will help you practice for the Create Performance Task.

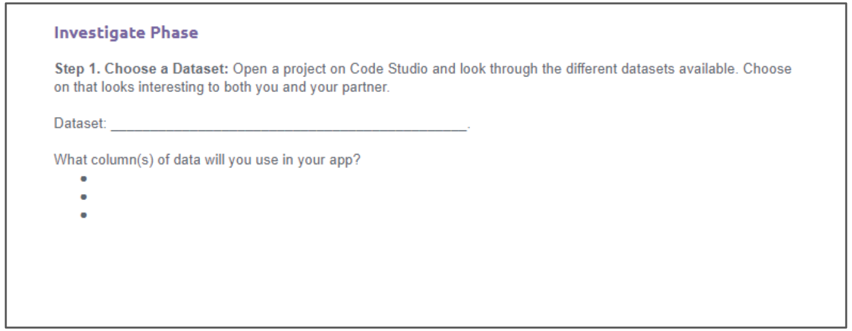
You will submit

- Your final app
- This completed project-planning guide
- A written response

App Requirements

- At least three screens
  - All screens can be easily navigated to through the user interface
- A dataset used in a meaningful way towards the programs purpose
- At least one list is traversed using: map, reduce, or filter (indicate which in a comment) in a meaningful way towards the program's purpose
- Programming constructs: variable(s), function(s), conditional(s), list(s), loop(s)
- All functions include comments that explain their purpose and how they work
- All element IDs have meaningful names
- No errors in the code

Hackathon Project



Investigate Phase

Step 1. Choose a Dataset: Open a project on Code Studio and look through the different datasets available. Choose on that looks interesting to both you and your partner.

Dataset: ____.

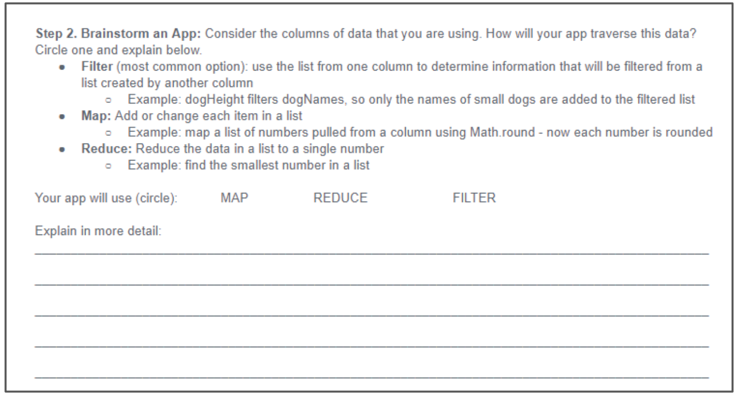
What column(s) of data will you use in your app?

-
-
-

Step 1: Choose a Dataset: With a partner...

- Go to the Make Project you just finished. Open up the data tab and look a the datasets.
- Choose a dataset that looks interesting to you
- Fill out the bottom of Page 1 in the Planning Guide.

Hackathon Project



Step 2. Brainstorm an App: Consider the columns of data that you are using. How will your app traverse this data? Circle one and explain below.

- **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

Your app will use (circle): MAP REDUCE FILTER

Explain in more detail:

____

____

____

____

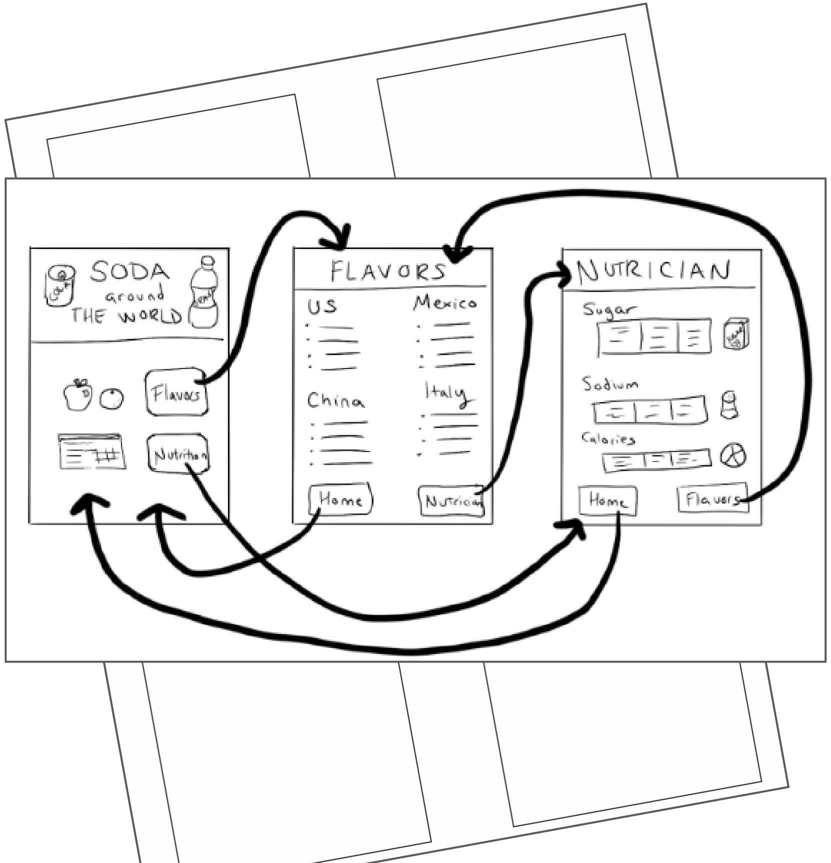
____

Step 2: Brainstorm an App: With a partner...

Consider the columns of data that you are using. How will your app traverse this data?

- **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
- Complete in Planning Guide.

Hackathon Project



Step 3: Create a paper prototype: With a partner...

- Draw a prototype which shows how your app will actually run.
- Include all the buttons, text, and images that the user will be able to use and see.
- Write notes or draw arrows showing how different user interface elements should work. For example, if clicking a button takes me to another screen, I should draw an arrow from that button to the drawing of the screen.

Wrap Up

Unit 5 - Lesson 14
Project - Hackathon Part 2

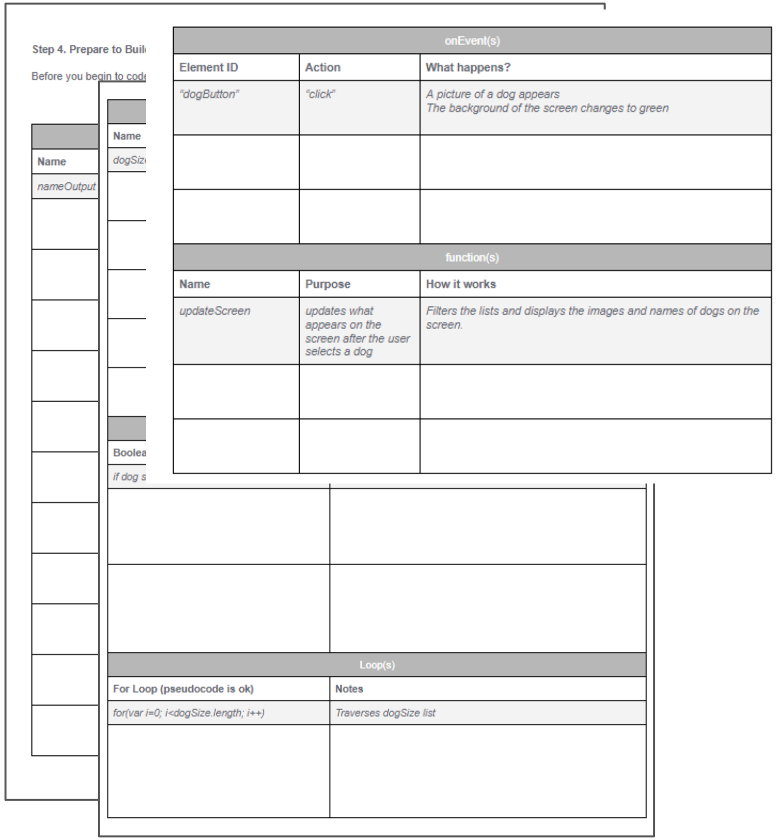
Warm Up

● ○ ○

Activity

●●○

Hackathon Project



Step 4. Prepare to Buil

Before you begin to code

Name
nameOutput

Name
dogSiz

Boolea
if dog s

onEvent(s)		
Element ID	Action	What happens?
"dogButton"	"click"	A picture of a dog appears The background of the screen changes to green

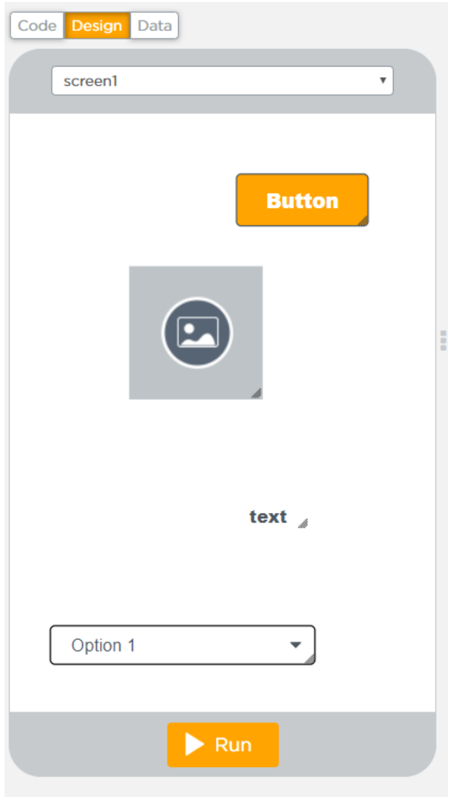
function(s)		
Name	Purpose	How it works
updateScreen	updates what appears on the screen after the user selects a dog	Filters the lists and displays the images and names of dogs on the screen.

Loop(s)	
For Loop (pseudocode is ok)	Notes
for(var i=0; i<dogSize.length; i++)	Traverses dogSize list

Step 4: Prepare to Build Your App: With a partner...

- Fill out the tables on pages 4-6 of the Planning Guide

Hackathon Project



Step 5: Test Screens: With a partner...

- Build a quick version of the screens which includes all elements with their proper element IDs. Do not worry about design at all. This is purely to allow the programmer the ability to test their code as they go along.

Hackathon Project

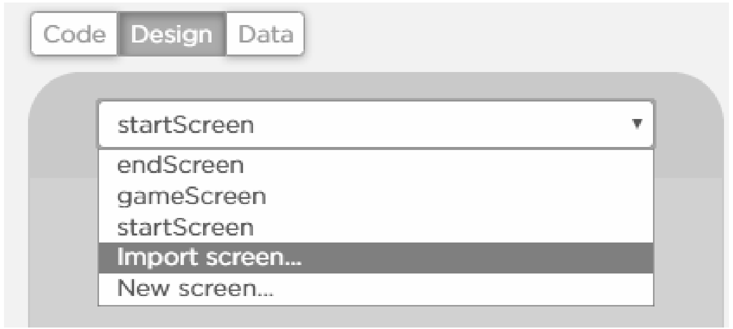
Choose Your Roles

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.

Hackathon Project

Step 6: Build:



Programmers: Use the table to guide you in adding programming statements to your program.

Designers: Use the chart to guide you in adding screen elements to your program. You can work on a separate computer from your partner.

- When you have finished screens, your partner should delete the test screens and then import the finished screens (see the next slide for instructions).
- **Note**: If screens are not deleted before importing the project you will get an error message because element IDs will be the same.

Designers - if you finish early, start pair programming with the programmer.

Unit 5 Lesson 14 - Activity

Designer, ready to share your screens with the programmer? Follow the steps below.

Why is this step necessary?
It's a little complicated. You cannot delete all of the screens from a project. In order to get rid of the old project screens, you'll need a new temporary screen. Once you've imported the Designer's screens, you can get rid of this temporary screen.

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

Why do I need to delete before importing?
The Programmer and Designer will have used the same element IDs for the placeholder screens and the imported screens. This may lead to problems when importing if the placeholder screens are not deleted first.

Wrap Up

Unit 5 - Lesson 15
Project - Hackathon Part 3

Warm Up

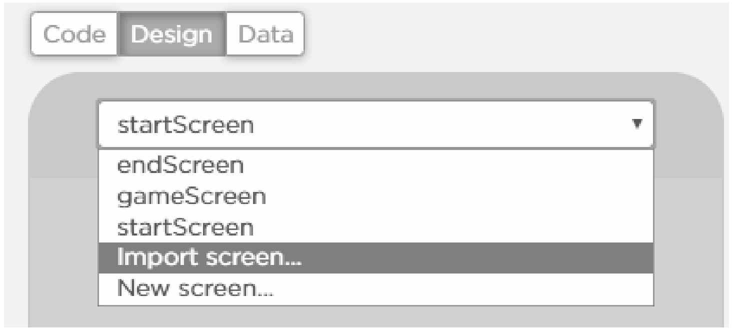
● ○ ○

Activity

● ● ○

Hackathon Project

Step 6: Build:



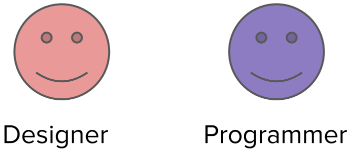
Programmers: Use the table to guide you in adding programming statements to your program.

Designers: Use the chart to guide you in adding screen elements to your program. You can work on a separate computer from your partner.

- When you have finished screens, your partner should delete the test screens and then import the finished screens (see the next slide for instructions).
- **Note**: If screens are not deleted before importing the project you will get an error message because element IDs will be the same.

Designers - if you finish early, start pair programming with the programmer.

Designer, ready to share your screens with the programmer? Follow the steps below.



Why is this step necessary?
It's a little complicated. You cannot delete all of the screens from a project. In order to get rid of the old project screens, you'll need a new temporary screen. Once you've imported the Designer's screens, you can get rid of this temporary screen.

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

Why do I need to delete before importing?
The Programmer and Designer will have used the same element IDs for the placeholder screens and the imported screens. This may lead to problems when importing if the placeholder screens are not deleted first.

Wrap Up

Unit 5 - Lesson 16
Project - Hackathon Part 4

Warm Up

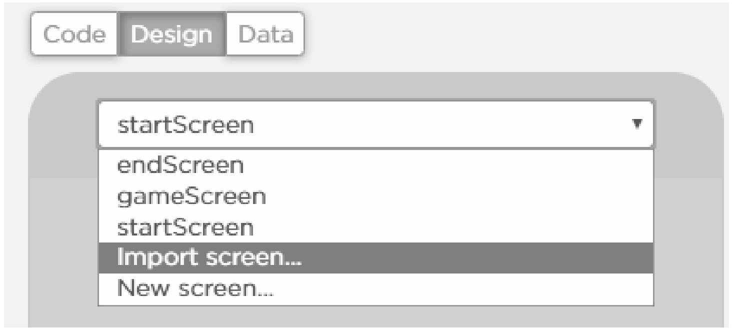
● ○ ○

Activity

● ● ○

Hackathon Project

Step 6: Build:



Programmers: Use the chart to guide you in adding programming statements to your program.

Designers: Use the chart to guide you in adding screen elements to your program. You can work on a separate computer from your partner.

- When you have finished screens, your partner should delete the test screens and then import the finished screens.
- **Note**: If screens are not deleted before importing the project you will get an error message because element IDs will be the same. To avoid this, add a blank screen, delete the old project screens, import the finished screens, and delete the blank screen. Make sure to set your home screen to be the default.

Designers - if you finish early, start pair programming with the programmer.

NOTE: When designers finish the screens, they should start pair programming with their partner.

Hackathon Project

Do This:

- Check the Scoring Guidelines on the final page of the Planning Guide.
- Update your code as needed to meet the requirements in the "Overall Project" section.

Wrap Up

Share Code:

Both partners should have access to the final project.

Unit 5 - Lesson 17
Project - Hackathon Part 5

Warm Up

● ○ ○

Activity

●●○

Written Response

On your individual computer:

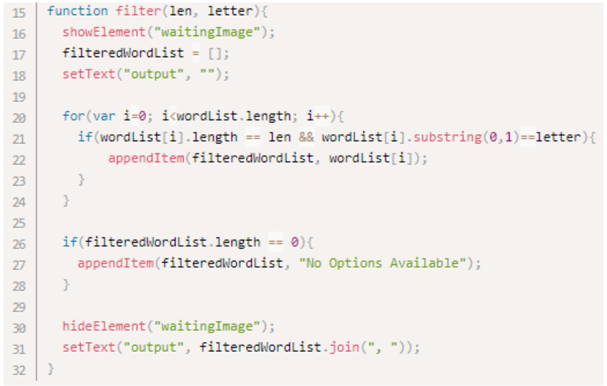
- One tab with the project
- One tab with the Written Response

Unit 5 Lesson 17 - Activity

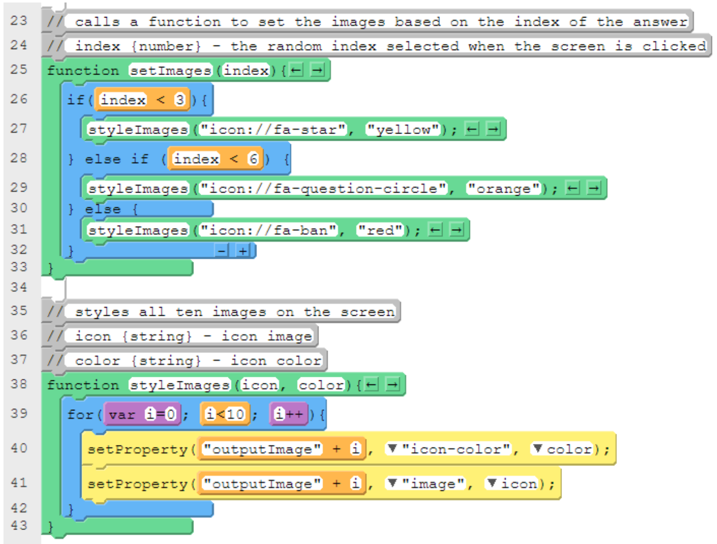
On your Written Response, you will need to take screenshots of **code segments.**

What's a code segment?

- a collection of program statements that are part of a program



```
function filter(len, letter){
  showElement("waitingImage");
  filteredWordList = [];
  setText("output", "");

  for(var i=0; i<wordList.length; i++){
    if(wordList[i].length == len && wordList[i].substring(0,1)==letter){
        appendItem(filteredWordList, wordList[i]);
    }
  }

  if(filteredWordList.length == 0){
    appendItem(filteredWordList, "No Options Available");
  }

  hideElement("waitingImage");
  setText("output", filteredWordList.join(", "));
}
```



```
// calls a function to set the images based on the index of the answer
// index {number} - the random index selected when the screen is clicked
function setImages(index){
  if(index < 3){
    styleImages("icon://fa-star", "yellow");
  } else if (index < 6) {
    styleImages("icon://fa-question-circle", "orange");
  } else {
    styleImages("icon://fa-ban", "red");
  }
}

// styles all ten images on the screen
// icon {string} - icon image
// color {string} - icon color
function styleImages(icon, color){
  for(var i=0; i<10; i++){
    setProperty("outputImage" + i, "icon-color", color);
    setProperty("outputImage" + i, "image", icon);
  }
}
```

Do This:

Complete the Written Response:

- Copy/paste code from your project to the boxes in the Written Response.
- Write out answers to the prompts.

Submit:

- Your Project
- Planning Guide
- Written Response

Share Your Projects!

Do This:

- Gallery Walk
- Pick your favorite project and write it down on a sticky note
- Your teacher will collect the sticky notes

Wrap Up

And the winner of the hackathon is...

Unit 5 - Lesson 18
Unit Assessment

Activity

● ● ○

Unit Assessment