Unit 7: Parameters, Return, and Libraries

Unit 7 - Lesson 1
Parameters and Return Explore

Warm Up

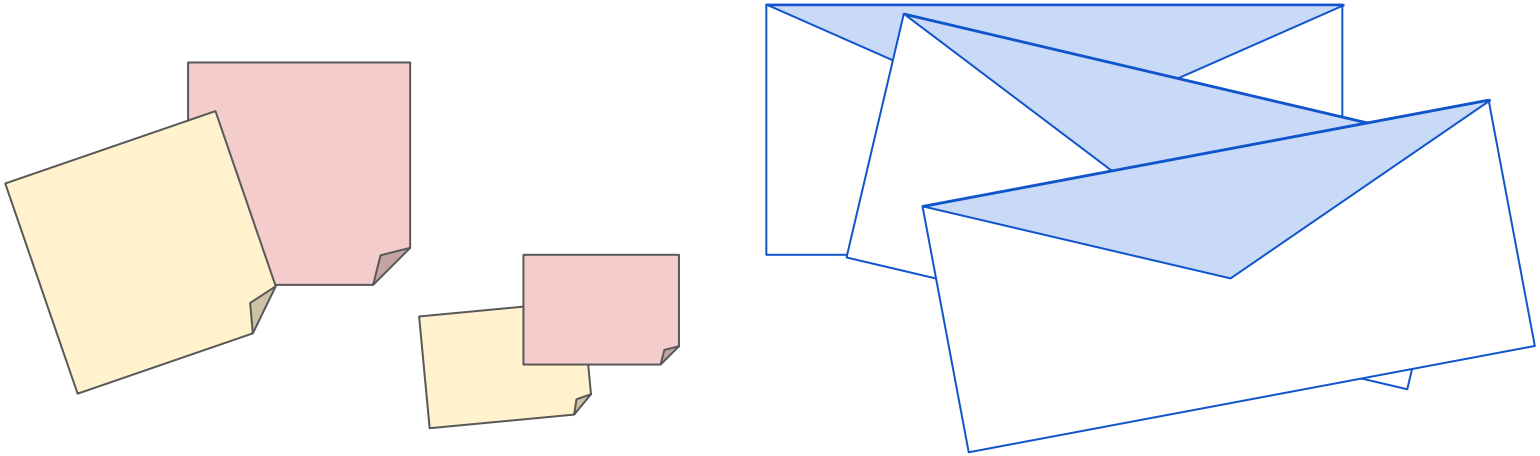
Prompt:

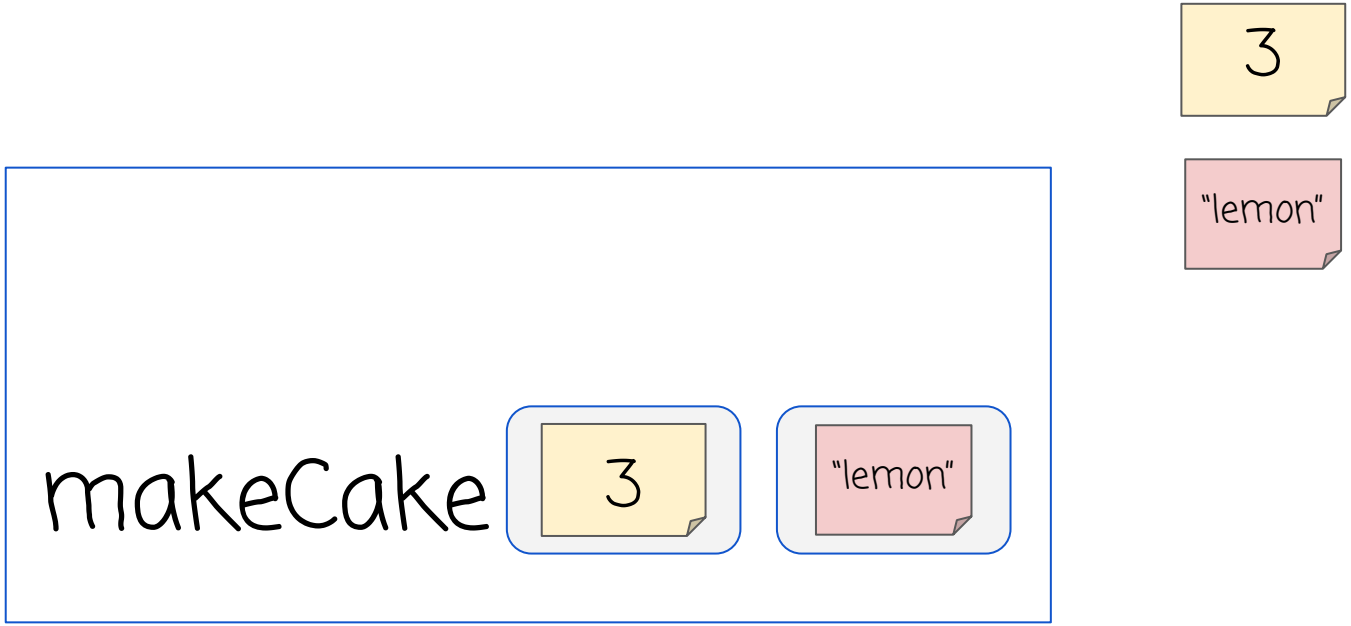
Why would you want to make your code easier to work with or read?

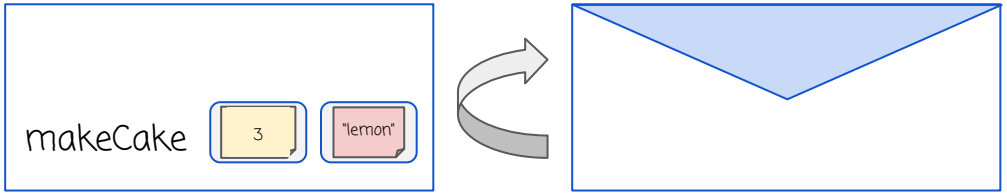
Activity

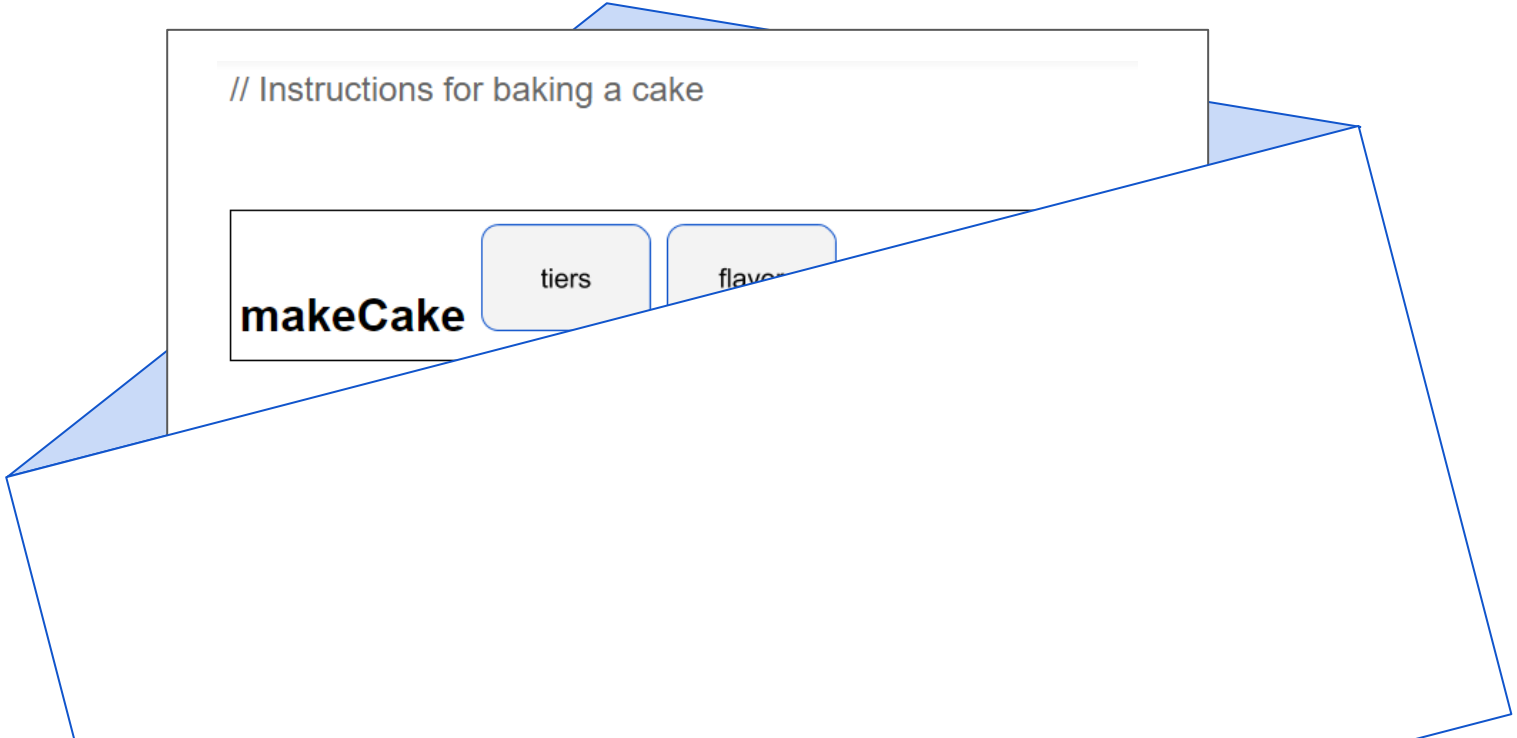
You and your partner should have

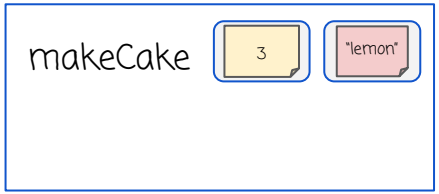
Small sticky notes
Regular sized sticky notes
Pen / Pencil
Envelopes




3
"lemon"
makeCake
3
"lemon"


makeCake
3
"lemon"


// Instructions for baking a cake
makeCake
tiers



// Instructions for baking a cake

makeCake 3 "lemon"

repeat 3 times:

Bake a "lemon" cake.
Ice the cake.

Assemble 3 layers into one cake.

Put in a box.

// Instructions for baking a cake

makeCake 3 "lemon"

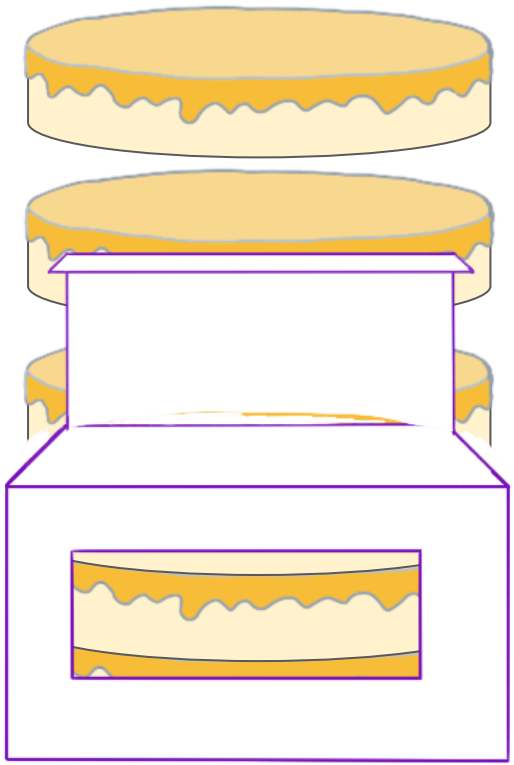
repeat 3 times:

Bake a "lemon" cake.

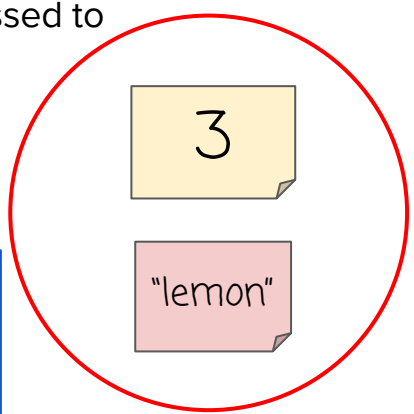
Ice the cake.

Assemble 3 layers into one cake.

Put in a box.

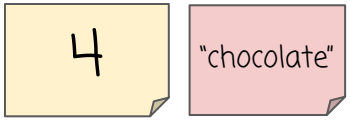


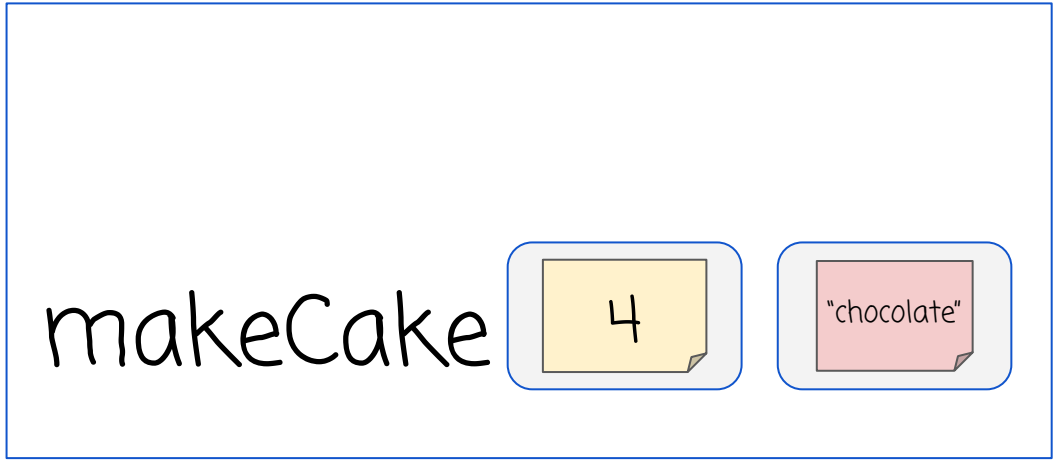
Argument - the value passed to the parameter

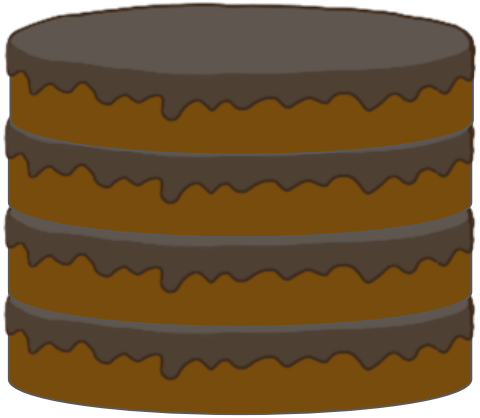


Parameter - a variable in a function definition. Used as a placeholder for values that will be passed through the function.

What if I wanted a four layer chocolate cake? What would that look like?

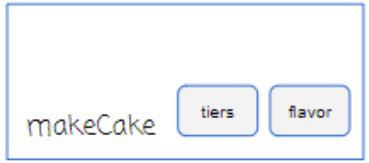


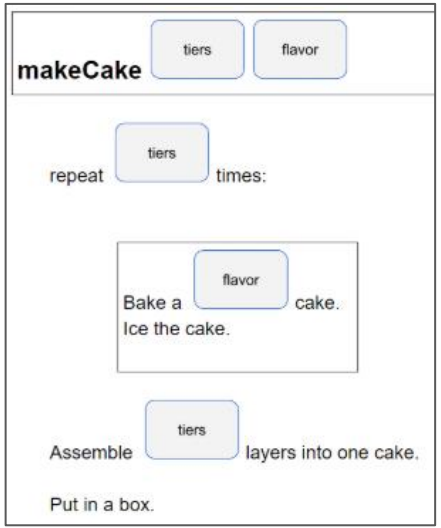




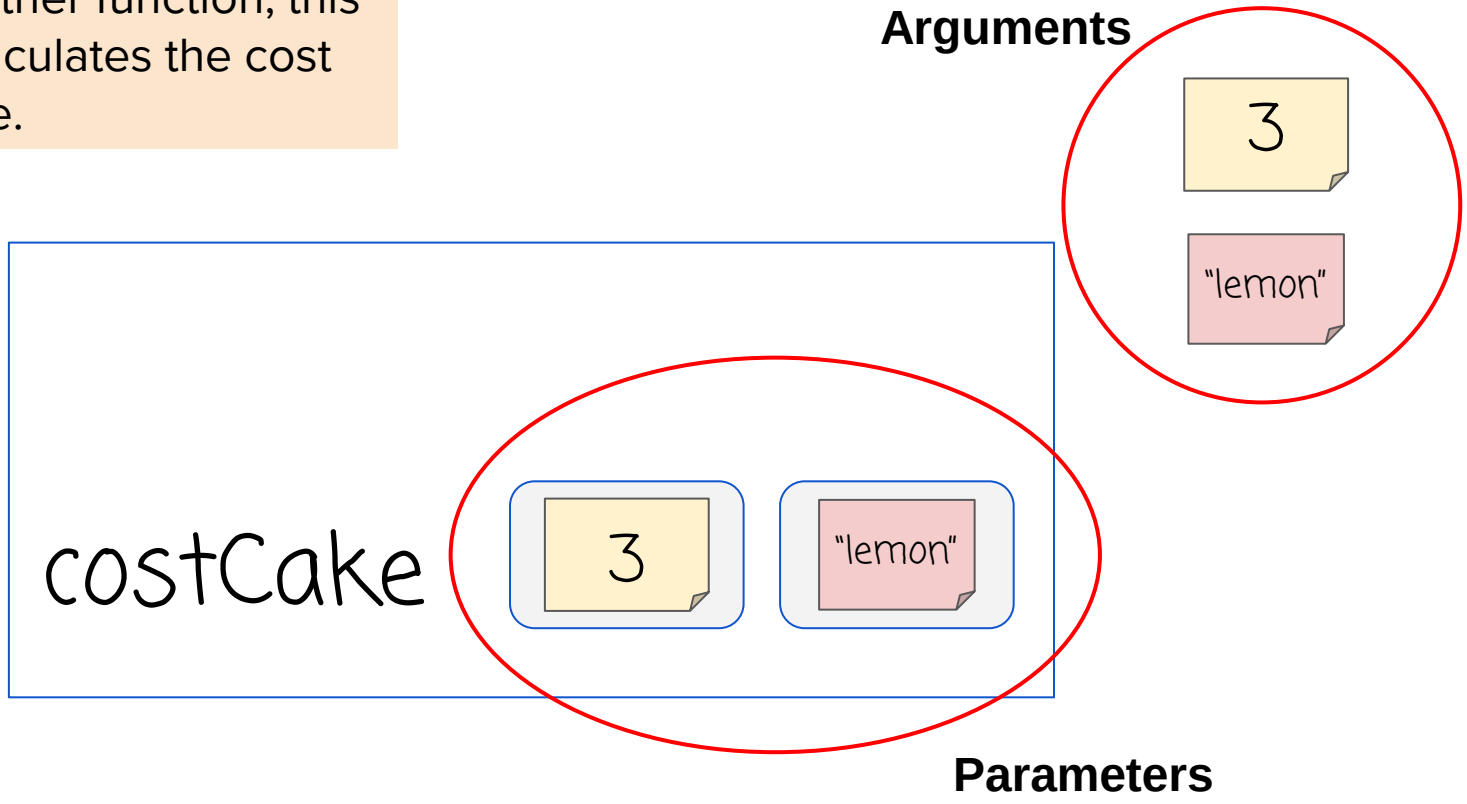
Do This: Time to create your own function with parameters!

- Complete Challenge #1 in your Activity Guide

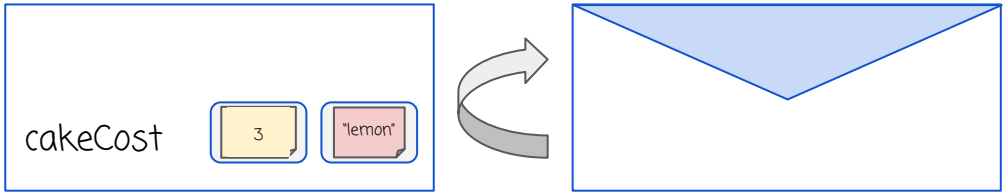


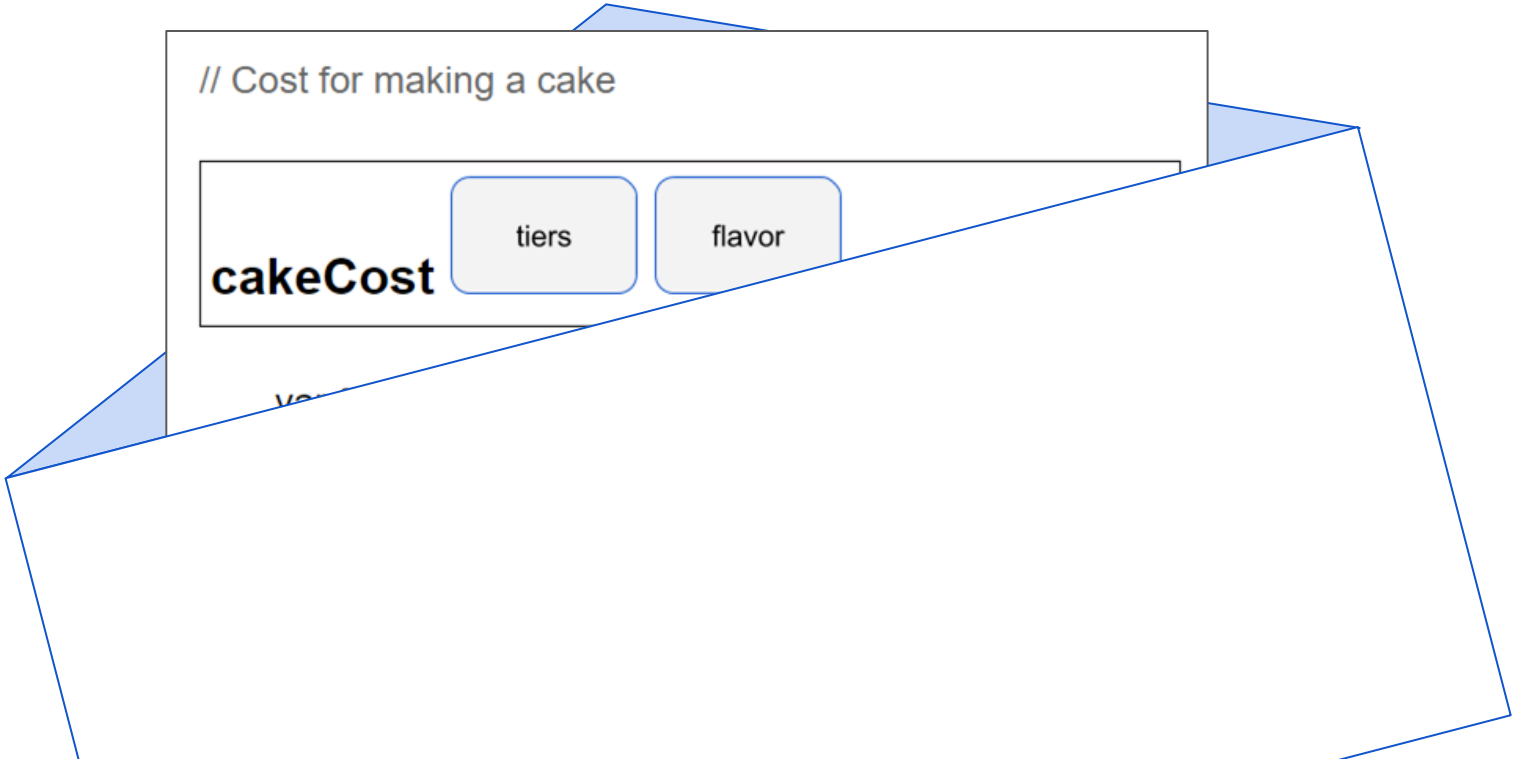


Let's look at another function, this time one that calculates the cost of making a cake.



Unit 7 Lesson 1 - Activity





```
// Cost for making a cake

cakeCost (tiers, flavor)

    var flavorCost
    var total

    If (flavor) equals "chocolate":
        flavorCost = 5

    If (flavor) equals "lemon":
        flavorCost = 4

    If (flavor) equals "vanilla":
        flavorCost = 3

    total = flavorCost multiplied by (tiers)

    return total
```

Create two new local variables, **flavorCost** and **total.**

Determine the value of **flavorCost** based on the <u>argument</u> passed through the flavor parameter

Calculate **total** using **flavorCost** and the <u>argument</u> passed through the tiers parameter

???????

```
// Cost for making a cake

cakeCost (3, "lemon")

var flavorCost
var total

If [flavor] equals "chocolate":
    flavorCost = 5

If [flavor] equals "lemon":
    flavorCost = 4

If [flavor] equals "vanilla":
    flavorCost = 3

total = flavorCost multiplied by [tiers]

return total
```

After running this, what does **flavorCost** equal? **4**

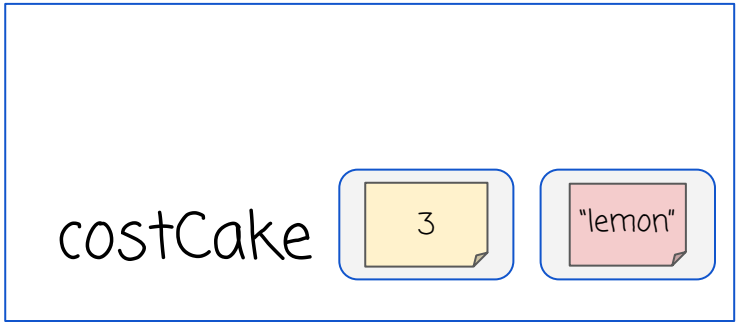
What does **total** equal? **12**

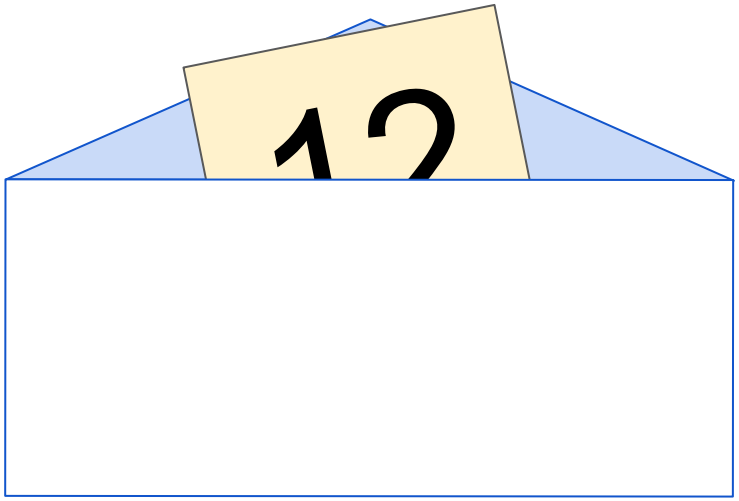
```
// Cost for making a cake

cakeCost [ 3 ] [ "lemon" ]

var flavorCost
var total

If [ flavor ] equals "chocolate":
    flavorCost = 5

If [ flavor ] equals "lemon":
    flavorCost = 4

If [ flavor ] equals "vanilla":
    flavorCost = 3

total = flavorCost multiplied by [ tiers ]

return total
```

What does it mean to **return total**?

A return does two things:

- It stops the flow of the function. If a return is inside of a conditional, if that condition is met the function ends there.
- It **returns** a value to the place where the function was called.

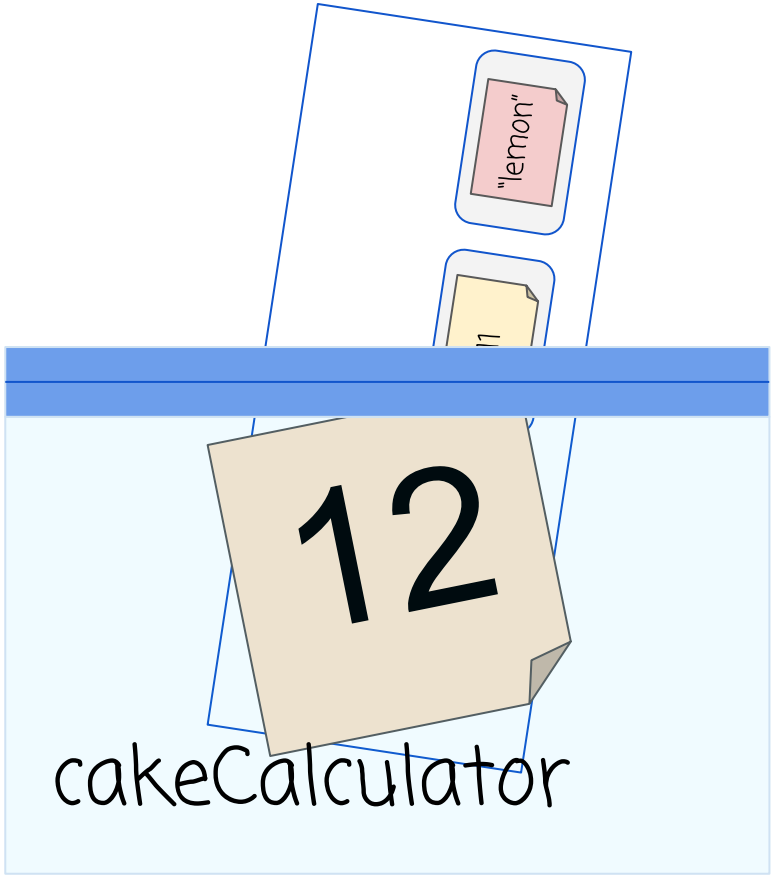




We've called the costCake function. It has returned the value **12.**

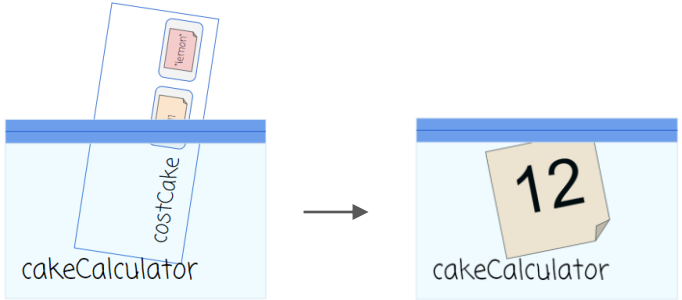
But what happens to that value?

How is it stored?



Let's return to variable baggies!

A function **return** value can be stored in a variable.



Here's how this looks in Javascript:

```
var cakeCalculator = cakeCost(3, "lemon");
```

After the expression is evaluated,
cakeCaculator stores the value **12.**

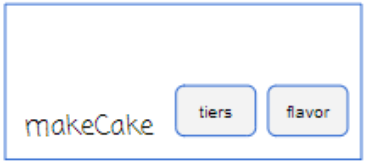
We can also print to the console like so:

```
console.log(“Cake cost: ” + cakeCost(3,
“lemon”));
```

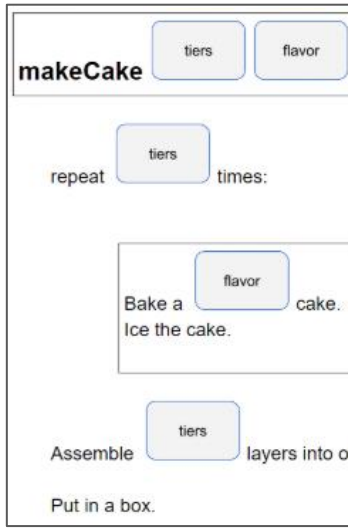
Console

```
Cake cost: 12
```

Do This: Create a cost calculator function for building the house you created a function for earlier.



- Complete Challenge #2 on your Activity Guide



Wrap Up

Takeaways

- Functions with parameters and return values help us simplify our code
- Functions can only return one value at a time
- A function can have:
    - No parameters and no return values
    - Parameters, but no return values
    - Return values, but no parameters
    - Parameters and return values

Vocabulary

- **Parameter -** a variable in a function definition. Used as a placeholder for values that will be passed through the function.
- **Argument -** the value passed to the parameter
- **Return -** used to return the flow of control to the point where the procedure (also known as a function) was called and to return the value of expression.

Unit 7 - Lesson 2
Parameters and Return Investigate

Warm Up

Prompt:

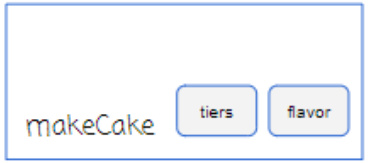
Are clean and organized programs more useful for computers or people?

Why?

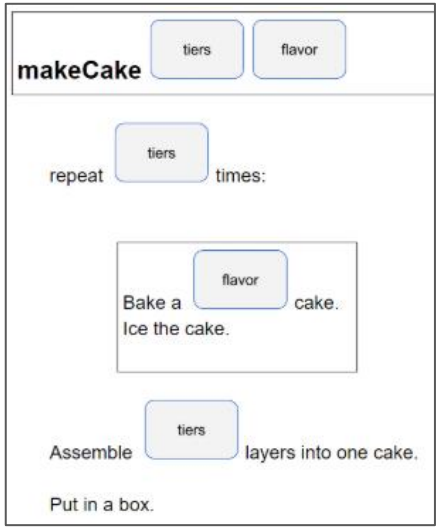
Try to give examples from programs you've written or seen in this class.

Activity

Review

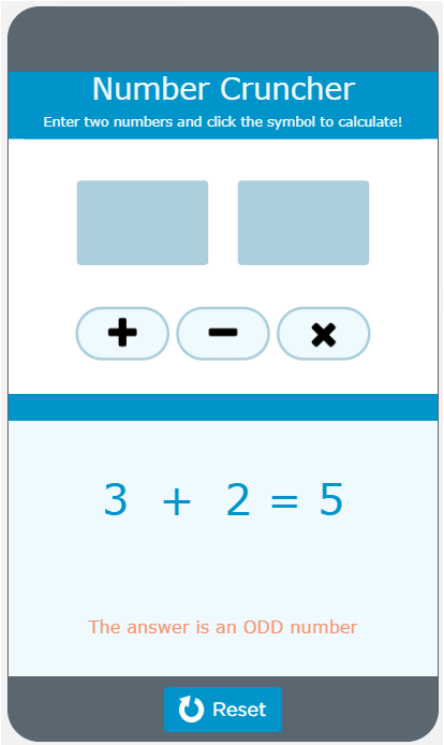


- Functions with parameters and return values help us simplify our code
- Functions can only return one value at a time
- A function can have:
    - No parameters and no return values
    - Parameters, but no return values
    - Return values, but no parameters
    - Parameters and return values



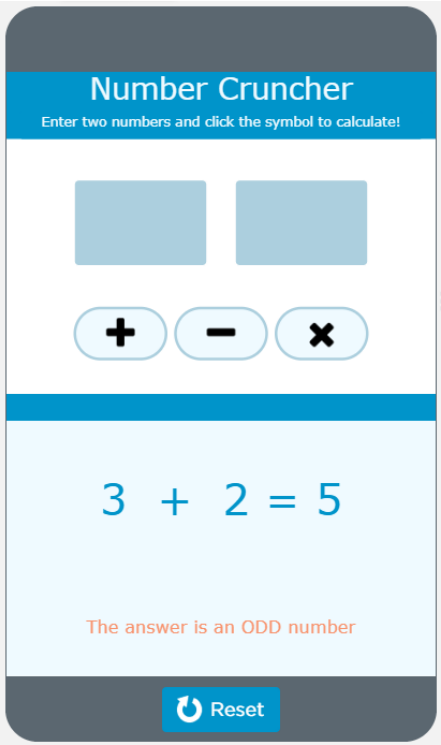
Do This:

- Find a partner
- Navigate to Level 2
- 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 the class.



Discuss as a class:

- How does `calculate(symbol)` work?
- What is the parameter?
- What is the argument?
- What is returned?



Do This:

- Look at lines 30-36. Discuss with a partner how the MOD operator % works.

```
// the MOD operator "%" - divides two numbers and returns the remainder
// in this case - if a number divide by two has a remainder of zero, it's an even number
if( answer%2 ==0 ){
  setText( "evenOddLabel", "The answer is an EVEN number.");
} else {
  setText( "evenOddLabel", "The answer is an ODD number");
}
```

MOD

Add	3 + 2	5
Subtract	3 - 2	1
Multiply	3 * 2	6
Divide	3 / 2	1.5
MOD	3 % 2	1

MOD is the remainder that is left after a number is divided by another number

Let's Practice

the answer

17 % 5

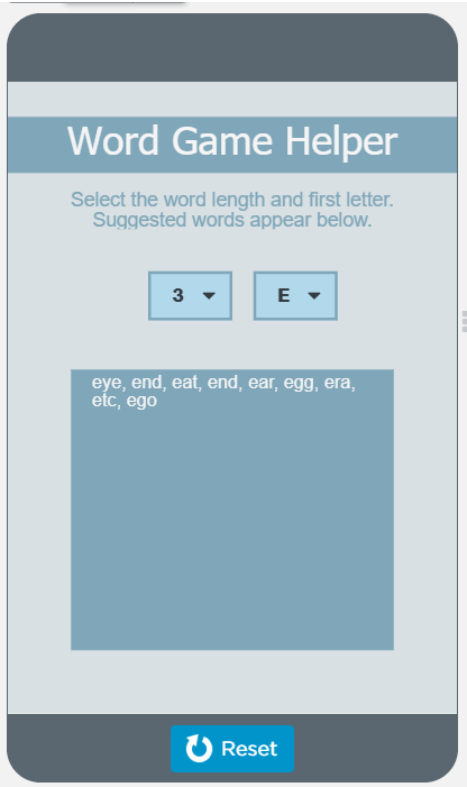
```
      3   remainder 2
5 | 17
   -15
     2
```

When is this useful?

- A common usage is to determine if a number is even or odd. If you divide any number by two and there is no remainder, the number is even!
- You can use MOD to determine if a number is divisible by another number.

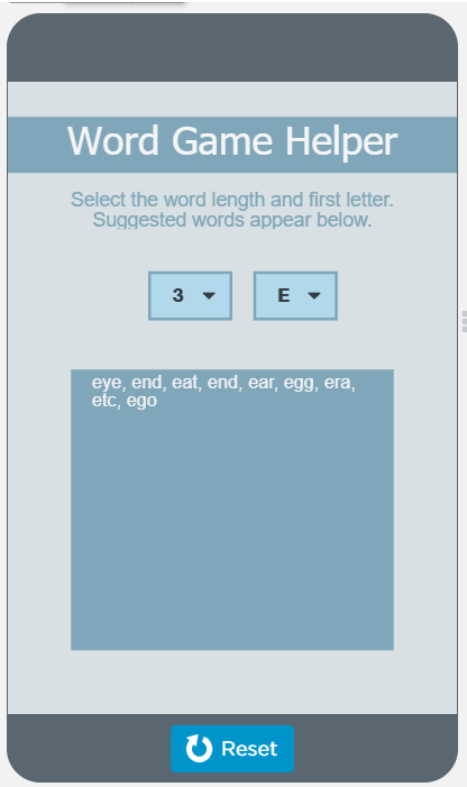
Do This:

- With your partner:
- Navigate to Level 3
- Read lines 1-14
- What is happening here?
- Be prepared to discuss as a class.



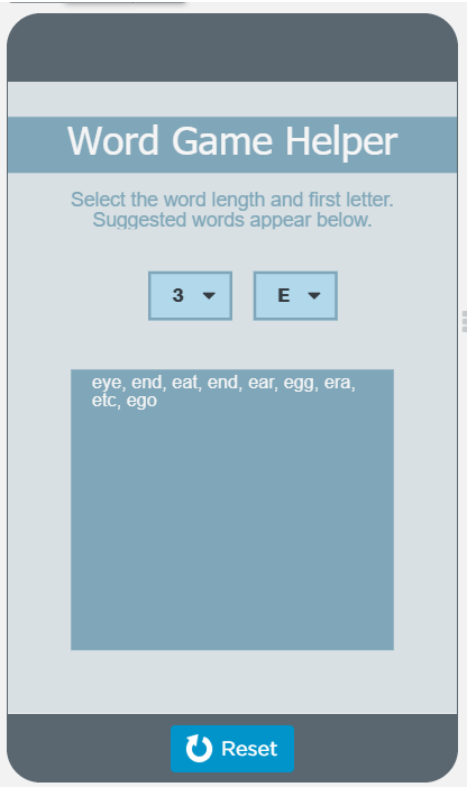
Do This:

- Now, read the function on lines 15-34 carefully. After you are done, explain to your partner how the function works, what parameters it takes and what is returned.
- Extra time? Complete the modify.



Discuss as a class:

How does the Word Game Helper work?



Wrap Up

Takeaways:

Extracting shared features to generalize functionality is known as **procedural abstraction**.

Using parameters allows the functions (also called procedures) to be generalized.

Using procedural abstraction helps improve code readability.

Procedural abstraction manages complexity by allowing for code reuse.

- For example: the function `move(id, direction)` could be used to move an element in any direction, rather than writing separate functions for each direction.

Unit 7 - Lesson 3
Parameters and Return Practice

Warm Up

Prompt: What is one reason why parameters and return values are useful?

What is one way you think programming with parameters and return values may make programming or debugging more challenging?

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

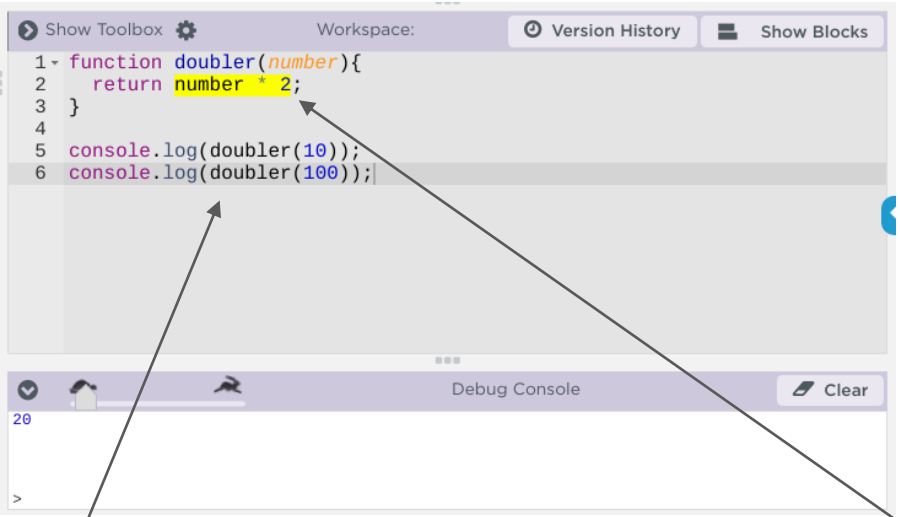
Make a small change

Document

As You Go

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

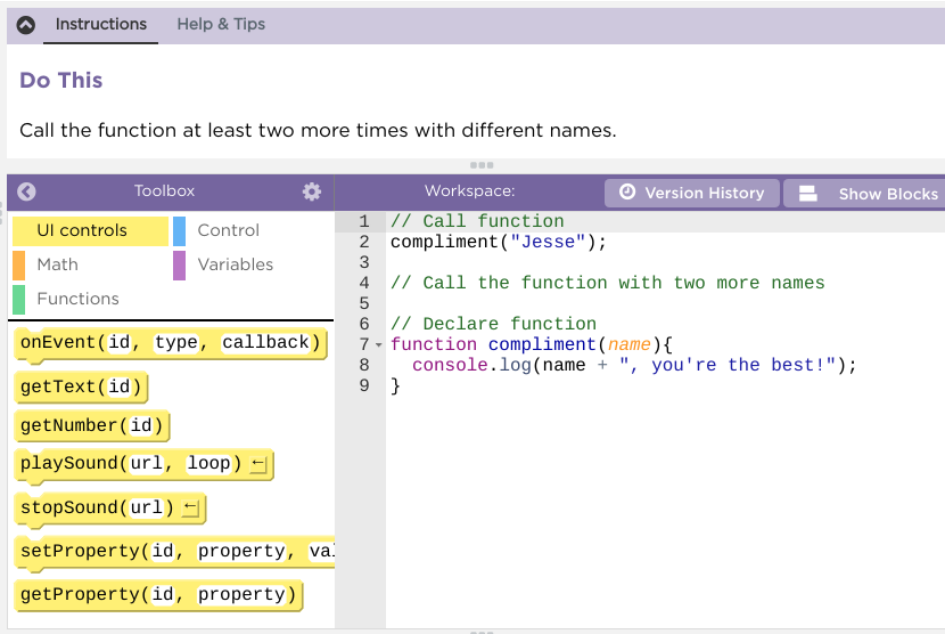
Debugging Functions with Parameters and Return

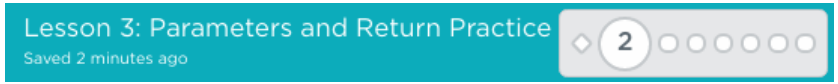


Use `console.log` to call functions with parameters and see how calling them with different values returns different values

Use the speed slider to slow down code so you can watch how functions are being called.

Parameters and Return Practice





Do This:

- Navigate to Lesson 3, Level 2 on Code Studio

Wrap Up

Prompt:

What aspects of working with parameters and return values do you feel like clicked today?

What do you still feel like you have trouble with?

Wrap Up

Prompt:

How could using parameters and return help you write programs collaboratively?

Unit 7 - Lesson 4
Parameters and Return Make

Warm Up

Prompt:

How do parameters and return change the way you write programs?

This Make Project is different

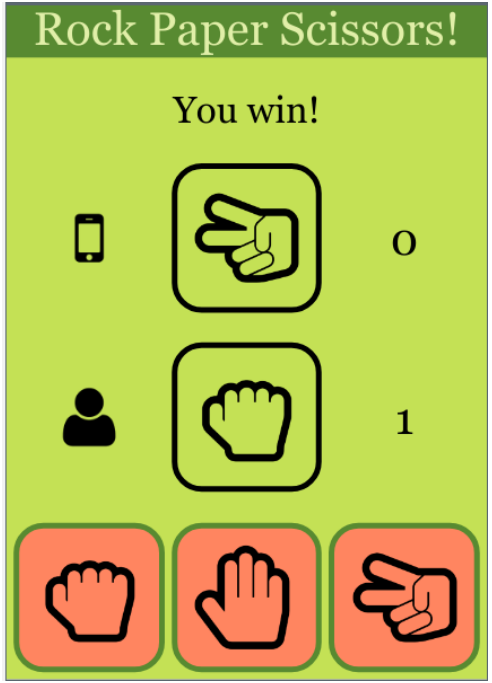
You'll get most of the code but three functions are incomplete. They always return the same value.

You'll need to rewrite these three functions using the comments provided

```
// iconName {string} - either "Rock", "Paper", or "Scissors"
// return {string} - the icon associated with the string provided
function findIcon(iconName){
    return "icon://fa-hand-rock-o";
}
```

Activity

Parameters and Return Make: Rock Paper Scissors App



Do This:

- Navigate to Lesson 4, Level 2 on Code Studio

Prompt:

- What does each button do
- How does the screen get updated after clicking each button

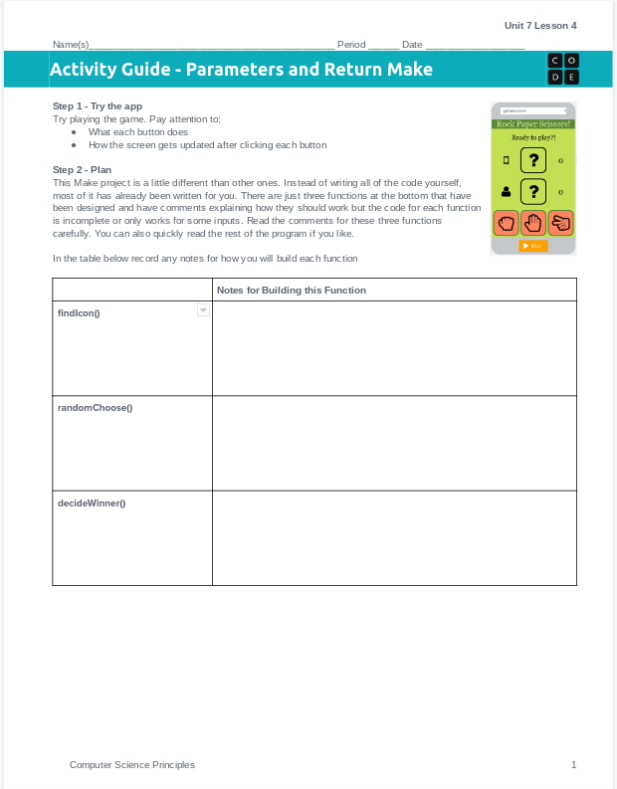
This Make project is different

You'll have most of the code but three functions are incomplete. They always return the same value.

You'll need to rewrite these three functions using the comments provided. No other edits in the program are necessary.

```
// iconName {string} - either "Rock", "Paper", or "Scissors"
// return {string} - the icon associated with the string provided
function findIcon(iconName){
    return "icon://fa-hand-rock-o";
}
```

Do This: Make the Rock Paper Scissors App!



Unit 7 Lesson 4

Name(s)_____ Period _____ Date _____

Activity Guide - Parameters and Return Make

Step 1 - Try the app

Try playing the game. Pay attention to:

- What each button does
- How the screen gets updated after clicking each button

Step 2 - Plan

This Make project is a little different than other ones. Instead of writing all of the code yourself, most of it has already been written for you. There are just three functions at the bottom that have been designed and have comments explaining how they should work but the code for each function is incomplete or only works for some inputs. Read the comments for these three functions carefully. You can also quickly read the rest of the program if you like.

In the table below record any notes for how you will build each function

	Notes for Building this Function
findIcon()	
randomChoose()	
decideWinner()	

Computer Science Principles 1

Use the activity guide to plan out your code.

Don't forget you're only working inside of the three functions at the bottom of the program. Use programming patterns to help and test your code as you go.

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

Don't forget: check the rubric on the last level before hitting submit

Category	Extensive Evidence	Convincing Evidence	Limited Evidence	No Evidence
findIcon Function	The function returns the correct values for all input values.	The function returns the correct values for most input values.	The function returns the correct values for some input values.	The function does not return correct values for any input values.
randomChoose Function	The function returns the correct values for all input values.	The function returns the correct values for most input values.	The function returns the correct values for some input values.	The function does not return correct values for any input values.
decideWinner Function	The function returns the correct values for all combinations of inputs.	The function returns the correct values for most combinations of inputs.	The function returns the correct values for some combination of inputs.	The function does not return correct values for any combination of inputs
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.
Coding Comments	Comments are used to correctly explain the purpose and functionality of all functions.	Comments are used to explain the purpose and functionality of most functions.	Comments are present, but are not used to explain the purpose or functionality of any functions.	Comments are not present.

Submit

Wrap Up

Prompt:

How could using parameters and return help you write programs collaboratively?

Unit 7 - Lesson 5
Libraries Explore

Warm Up

Prompt:

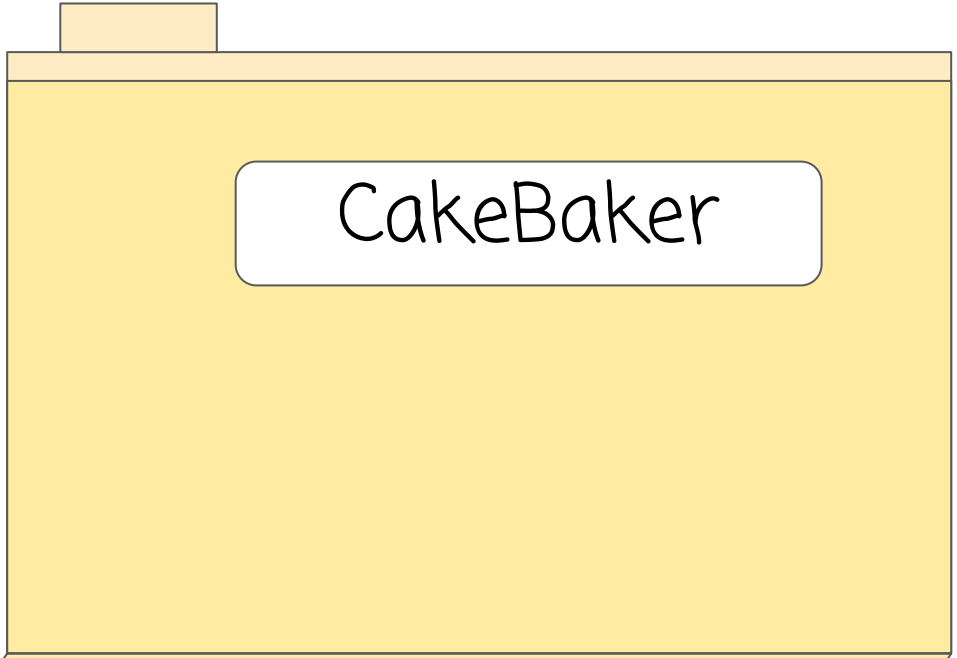
How could you share a function with another person so they could use it in their own program?

Activity

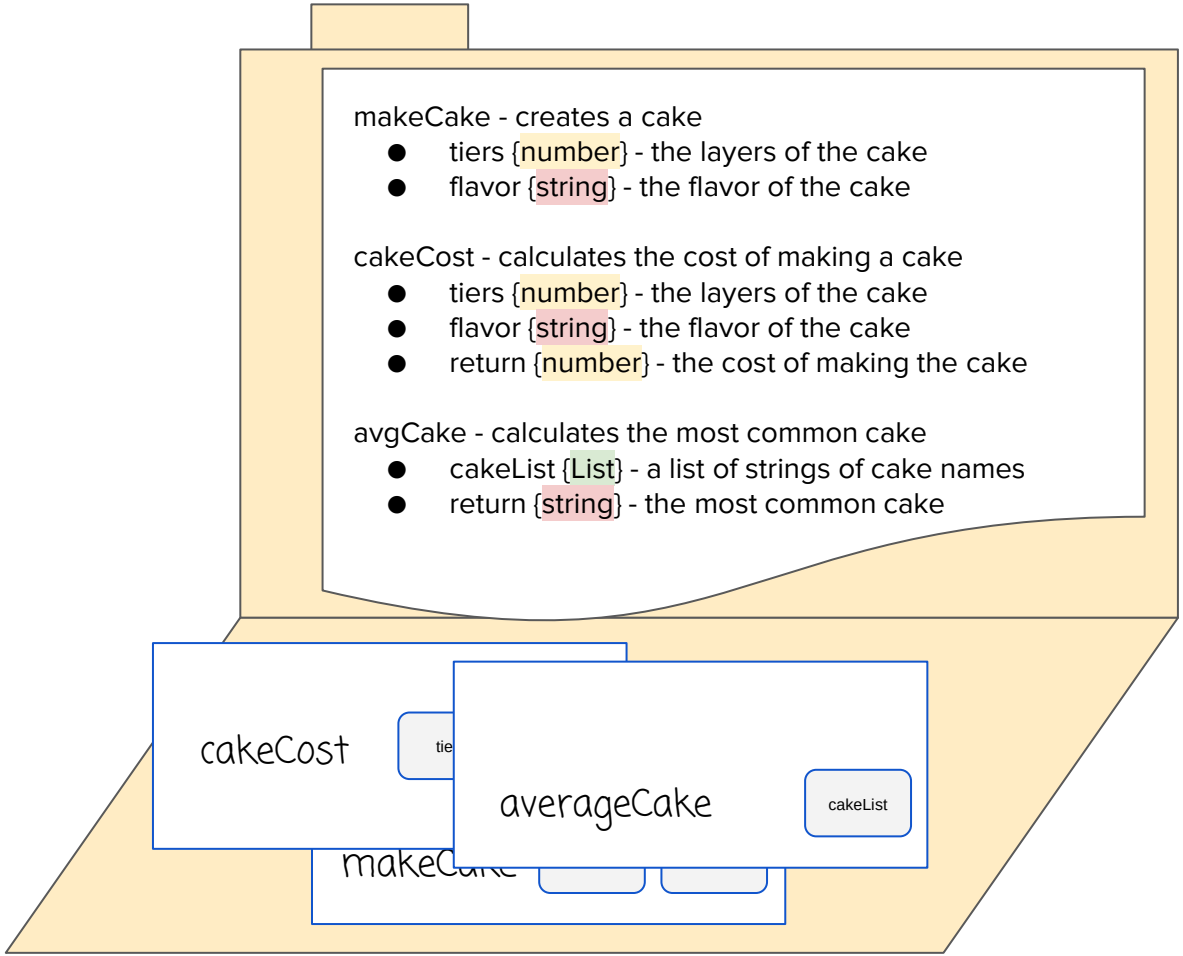
Have you ever wanted to share some of your code with a friend so they can use it to add a cool feature in their own program?

Or maybe you’ve got a collection of functions in one program that you want to use in another program.

How can we easily share functions between programs?



This is a **library** - a collection of functions that can be used in many different programs.



A **library** should have **documentation** for the included functions**:**

- how each function works
- a complete list of the parameters
- what (if anything) will be returned

makeCake - creates a cake

- tiers {number} - the layers of the cake
- flavor {string} - the flavor of the cake

cakeCost - calculates the cost of making a cake

- tiers {number} - the layers of the cake
- flavor {string} - the flavor of the cake
- return {number} - the cost of making the cake

avgCake - calculates the most common cake

- cakeList {List} - a list of strings of cake names
- return {string} - the most common cake

This detailed type of documentation is also known as an:

Application Program Interface(API)

APIs are specifications for how the functions in a library behave and can be used.

Discuss:

What potential problems could come up if I tried to use a function without knowing what it does or how to interact with it?

This would be similar to looking at the front of a function envelope and having to guess:

- what the function does
- what data type the parameters need
- what is returned

```
var smallest;

findSmallest(34, 99);

function findSmallest(num1, num2){
    if(num1 < num2){
        smallest = num1;
    } else {
        smallest = num2;
    }
}

```

Discuss:

My friend wants to use my findSmallest() function in her program. Is this function ready to be shared in a library?

Why or why not?

Watch out for global variables! If a function accesses or updates a variable elsewhere in your program, that function shouldn't be shared as is.

With a partner, rewrite the function so it could be shared in a library.

Hint: What about using a return?

Before adding a function to a library:

1. Check for any use of a global variable within the function. If there is, rework the function using local variables and a return.

1. Check if another function is called in this function. If so, both functions should be included in the library.

1. Write the documentation for the function.

```
function findSmallest(num1, num2){
    if(num1 < num2){
        return num1;
    } else {
        return num2;
    }
}
```

findSmallest: Given two numbers, finds the smallest

- num1 {number} - first number
- num2 {number} - second number
- return {number} - the smaller of the two numbers

Now my function is almost ready to be shared in a library.

With a partner, write the API for this function:

- how the function works
- all the parameters, their data types, and a short description of each
- what (if anything) will be returned

A library needs a name.

Follow these rules:

- No spaces
- Capitalize the first letter

This **library** can now be shared with others.

They can use the functions within their own program as long as they follow the rules set forth in the documentation.

You've seen libraries in action before...

The **Math** library is built into App Lab.

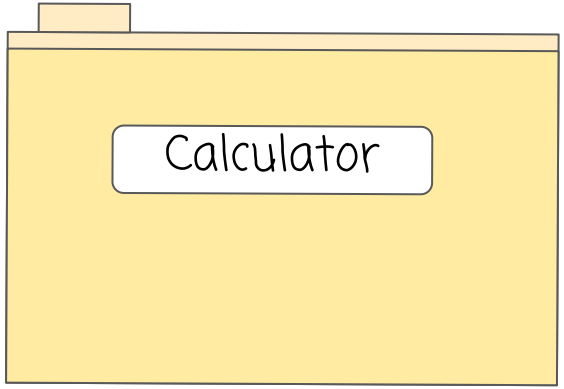
```
Math.round(5, 10);
```

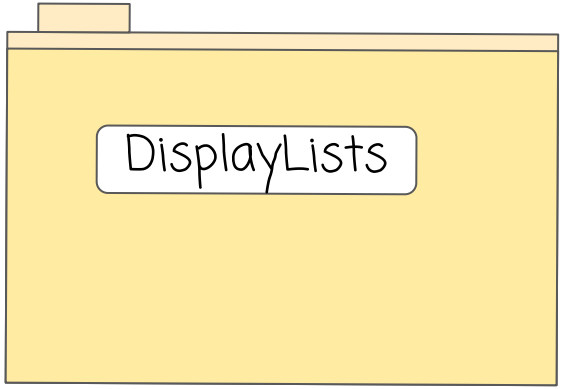
Name of the library

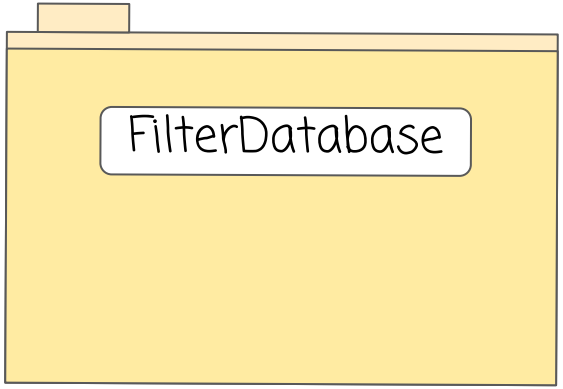
Name of the function

Parameters

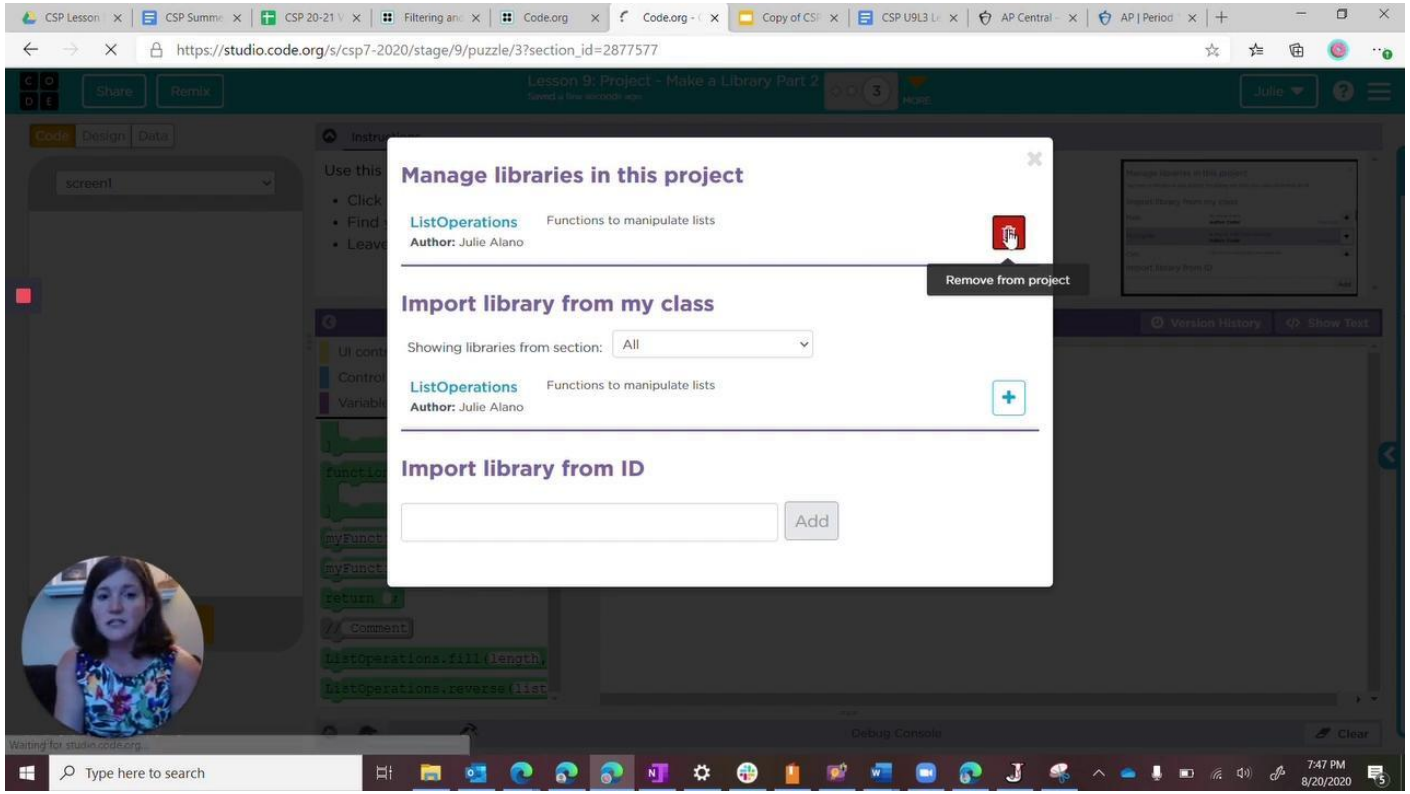
Do This: Brainstorm with a partner a few functions that might show up in the following libraries:







Libraries in App Lab



Wrap Up

Vocabulary

Library: a group of functions (procedures) that may be used in creating new programs

API: Application Program Interface - specifications for how functions in a library behave and can be used

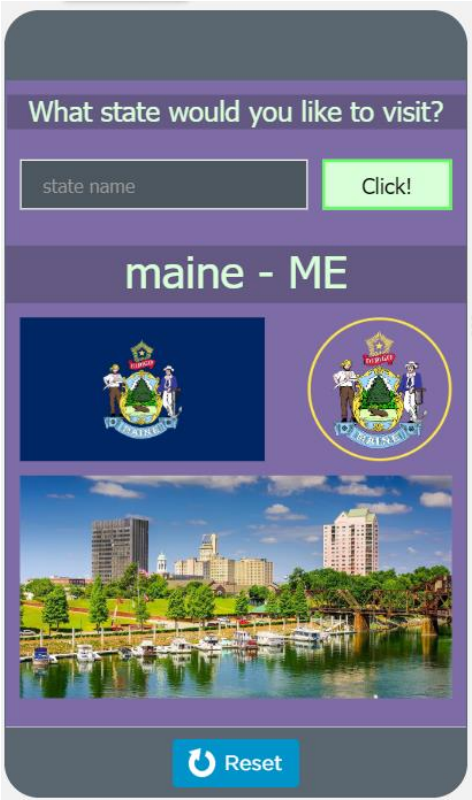
Unit 7 - Lesson 6
Libraries Investigate

Warm Up

Prompt:

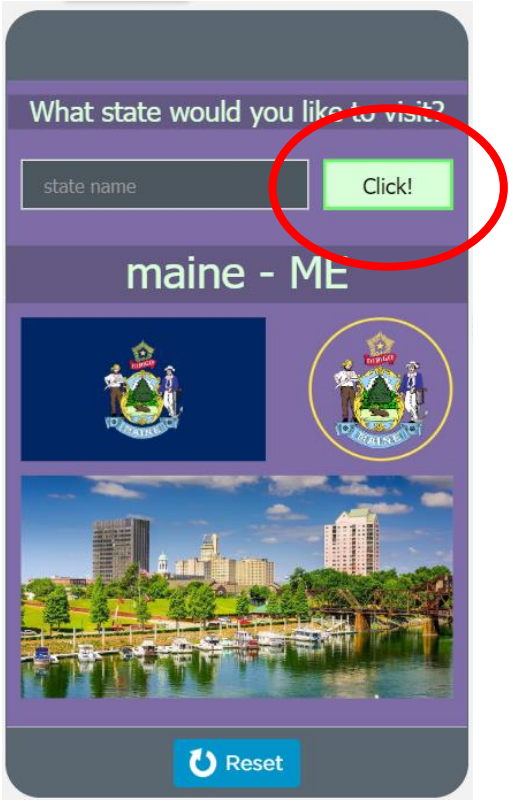
Today we are going to learn how to use libraries to share code with one another. Usually you do this by writing functions with parameters and return values. Why do you think it's important to use parameters and return values when writing code for other people to use?

Activity



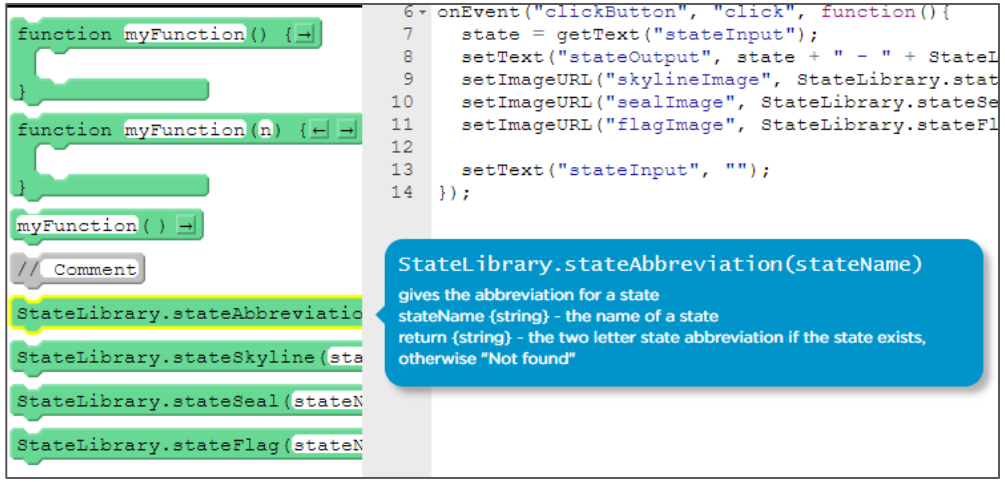
Do This:

- Navigate to Level 2
- Run the app
- Try several different inputs



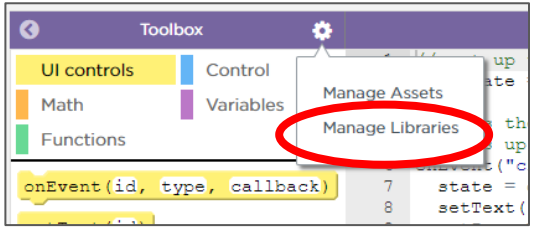
Discuss:

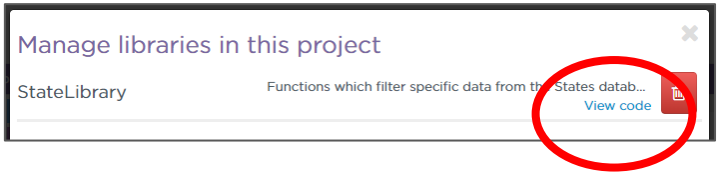
- With a partner, look at the project code.
- Discuss what happens when the button is clicked.



Do This:

- Open the functions drawer
- Look at each of the StateLibrary functions. Mouseover for the documentation.
- Discuss with a partner how you think these functions work.





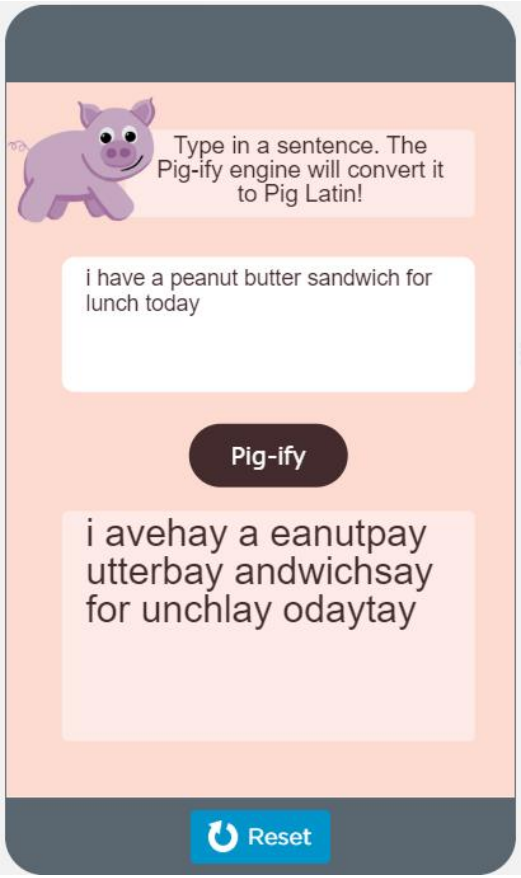
Do This:

- Click "Manage Libraries"
- Click "view code" for the State Library
- With your partner, read through the library and discuss how the functions work. Were you accurate in your predictions?

Prompt:

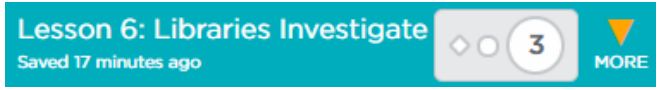
What are the benefits of hiding all of the code for filtering the dataset in a library?

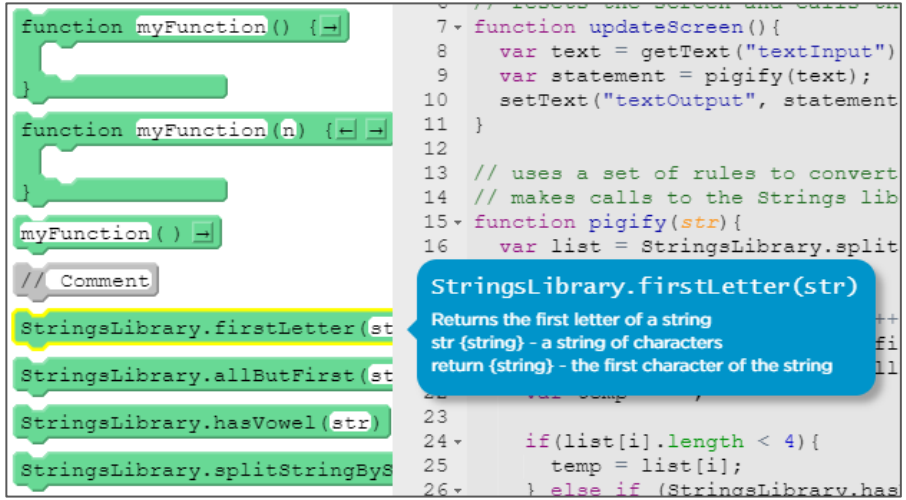
What information does the user need to know in order to use the library functions?



Do This:

- Navigate to Level 3
- Run the app
- Try several different inputs





Do This:

- Open the functions drawer
- Look at each of the StringsLibrary functions. Mouseover for the documentation.
- Discuss with a partner how you think these functions work.

Test the functions:

- Re-read the documentation for each library function
- Add a `console.log()` statement to the end of the program and call a function. Put in a reasonable argument in the space for the parameter.
  - For example:
    `console.log(StringsLibrary.firstLetter("pizza"));`
- Hit run to see the output.
- Now add `console.log()` statements to test the rest of the functions. Is the output what you would expect? Try several different inputs.

Prompt:

Why should we test the functions in the library?
What does this help us to know?

Do This:

- Navigate back to the States App
- Add `console.log` statements for each of the functions and test them out. Is the output as expected?

Prompt:

What makes a good library function? How can you make sure that the end users of your library have what they need in order to use your functions?

Let's quickly review Algorithms:

Up to this point, most of the algorithms you've used you created yourself, or you modified existing code.

Do This: Look at the functions in Levels 2 & 3 that call the library functions.

Notice here how we can build new functions by combining the existing library functions - essentially we are creating new algorithms by using existing algorithms (library functions).

Prompt: What are the benefits of using existing algorithms instead of brand new algorithms?

Examples of existing algorithms you may have seen:

- the maximum or minimum of 2 or more numbers
- the sum or average of 2 or more numbers
- an algorithm that determines if an integer can be evenly divided by another integer
- a robot's path through a maze

Let's quickly review Procedural Abstraction:

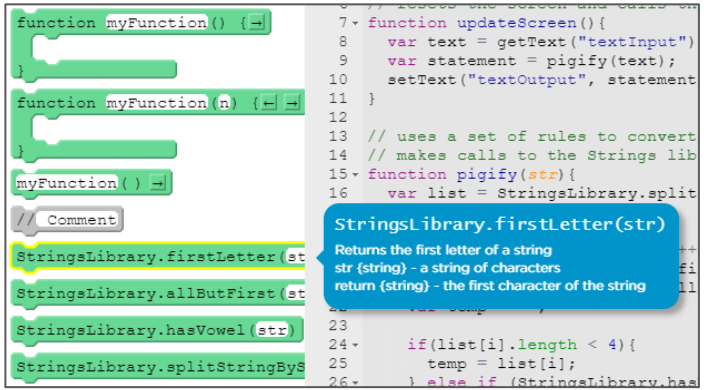
Procedural Abstraction provides a name for a process and allows the procedure (function) to be used only knowing what it does, and not necessarily how it does it.

This is how our libraries work!

There's a term for using libraries or other forms of organization in a program:

Modularity - the subdivision of a computer program into separate subprograms.

Takeaways:



- **Creating a library:**
  - Build functions
  - Add documentation
  - Share as a Library
- **Using a library:**
  - Click "Manage Libraries"
  - Either choose a classmate's library, or paste in a library code
  - Call the functions by writing the library name, a dot, the name of the function, and including any arguments for the parameters
- **Testing a library:**
  - Use console.log as the end user to test functions in a library
  - Check that the output is what you would expect
  - Read the library code if something does not work correctly, and contact the library owner if something needs to be changed.

Wrap Up

Prompt:

Based on what you saw today, add reasons **why** someone would argue for the following three statements

- Libraries help programmers collaborate because...
- Libraries help programmers reuse code because...
- Libraries help programmers writer simpler programs because...

Vocabulary:

Modularity: the subdivision of a computer program into separate subprograms

Unit 7 - Lesson 7
Libraries Practice

Warm Up

Prompt:

How does using a library allow you to think about programming at "a higher level"?

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?

Testing Functions

Code | Design | Data

Instructions

Show Toolbox | Workspace: | Version History | Show Blocks

```
1  // Decides if a number is even
2  // number {number} - a number to decide is even or not
3  // return {boolean} - whether the number is even
4  function isEven(number){
5    return (number%2) == 0;
6  }
7
8  console.log("10 is even? " + isEven(10));
9  console.log("5 is even? " + isEven(5));
10 console.log("-2 is even? " + isEven(-2));
11 console.log("-5 is even? " + isEven(-5));
12
13 // Decides if a number is odd
14 // number {number} - a number to decide is odd or not
15 // return {boolean} - whether the number is odd
16 function isOdd(number){
```

Debug Console | Clear

```
"10 is even? true"
"5 is even? false"
"-2 is even? true"
"-5 is even? false"
>
```

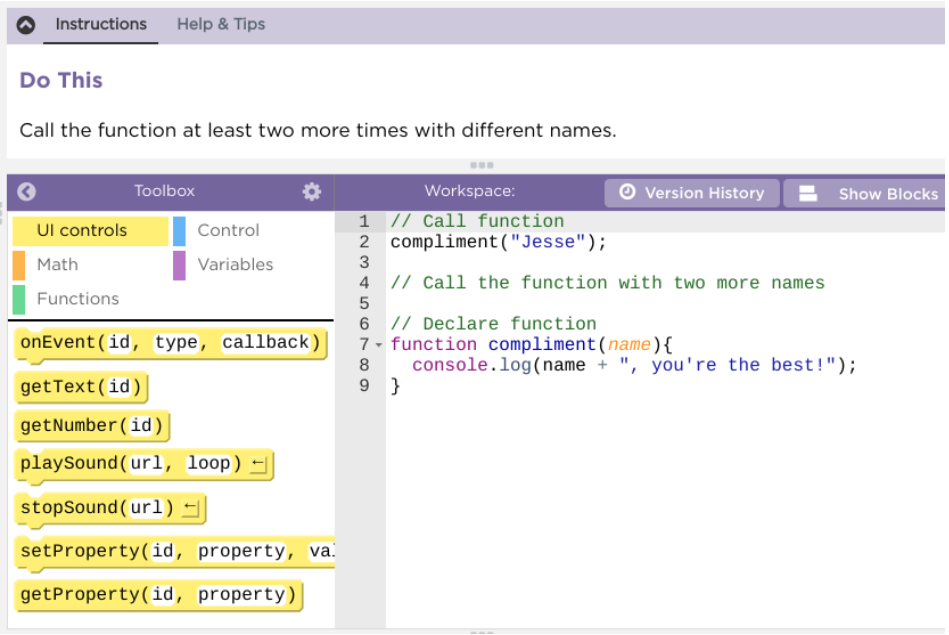
Reset

Privacy Policy | Copyright | More

Use `console.log` to **write tests** of the function. Try different values to make sure your function works in many cases.

Check the results in the console to make sure that the functions pass the test.

Libraries Practice



Do This:

- Navigate to Lesson 7, Level 2 on Code Studio

Wrap Up

Prompt:

How do libraries let you write programs at a "higher level"?

Why is testing important when building and sharing libraries?

Unit 7 - Lesson 8
Project - Make a Library Part 1

Warm Up

Prompt:

Think back over all the different apps you've built this year. What blocks do you wish already came with App Lab to help you build those apps?

Activity

Project - Make a Library

- Read the project description
- Review what you'll submit, steps of the project, and rubric

Step 1 - Brainstorm

Brainstorm a theme for your library.

- What kind of blocks do you want to add to App Lab?
- What situations do you want to make easier?

Step 2 - Design

- Choose 2 or more functions you'd like to build.
- At least one needs a parameter, return, loop, and if-statement
- Fill in step 2 of the Project Guide

Step 3 - Build

Use the rest of your time today to build out the functions you designed.

Wrap Up

Unit 7 - Lesson 9
Project - Make a Library Part 2

Warm Up

Two ways to test your library

1. Write tests!
2. Have a classmate try it out

Activity

Tests refresher

Code | Design | Data

Instructions

Show Toolbox | Workspace: | Version History | Show Blocks

```
1  // Decides if a number is even
2  // number {number} - a number to decide is even or not
3  // return {boolean} - whether the number is even
4  function isEven(number){
5    return (number%2) == 0;
6  }
7
8  console.log("10 is even? " + isEven(10));
9  console.log("5 is even? " + isEven(5));
10 console.log("-2 is even? " + isEven(-2));
11 console.log("-5 is even? " + isEven(-5));
12
13 // Decides if a number is odd
14 // number {number} - a number to decide is odd or not
15 // return {boolean} - whether the number is odd
16 function isOdd(number){
```

Debug Console | Clear

```
"10 is even? true"
"5 is even? false"
"-2 is even? true"
"-5 is even? false"
>
```

Reset

Privacy Policy | Copyright | More

Use `console.log` to **write tests** of the function. Try different values to make sure your function works in many cases.

Check the results in the console to make sure that the functions pass the test.

Step 4 - Test

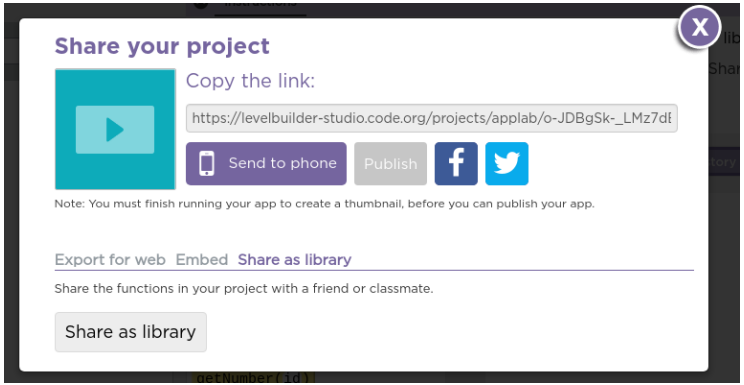
- As your program today, add tests to your functions to make sure they're working as you expect
- Keep writing your library

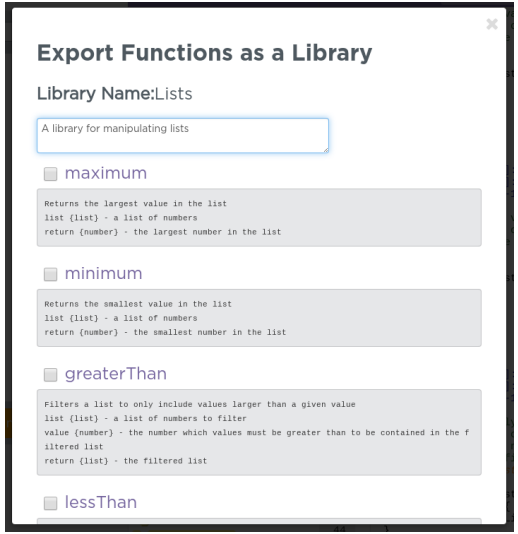
Step 5 - Feedback - Export Your Library

1. In Lesson 9, Level 2 click Share → Show Advanced Options → Share as Library

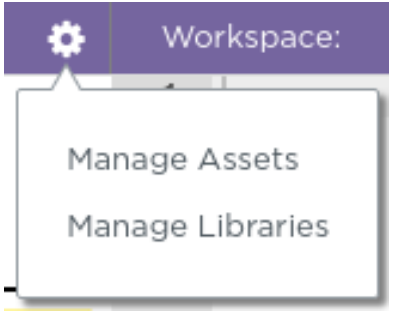
1. Choose the functions you'd like to export. If you need to edit the comments before your functions do so.

2. Hit Publish





Step 5 - Feedback - Import a Library

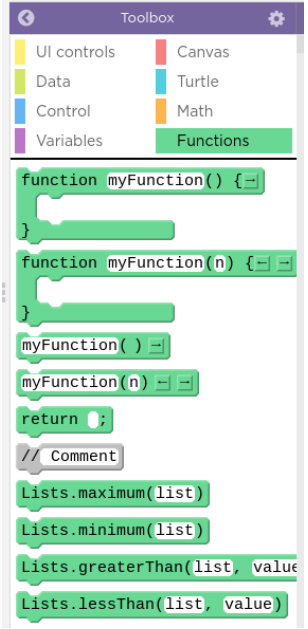


1. Go to the next level, Lesson 9 Level 3

1. Click the gear and then Manage Libraries

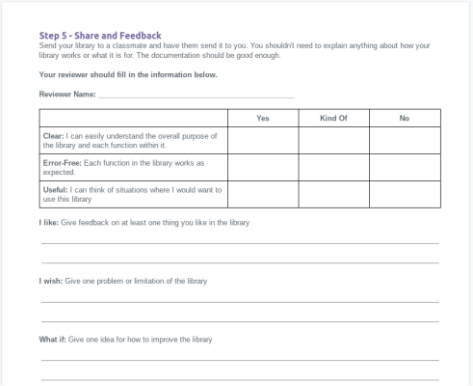
1. Find your partner's library and import it

1. Start testing the different functions they shared with you. They'll be in the "Functions" drawer



Step 5 - Feedback - Give Feedback

- On your classmate's project guide give them feedback about their library.
- Hover over blocks to read their documentation
- You can view all the library code by clicking "View Code" from the "Manage Libraries" window



Step 5 - Share and Feedback

Send your library to a classmate and have them send it to you. You shouldn't need to explain anything about how your library works or what it is for. The documentation should be good enough.

Your reviewer should fill in the information below.

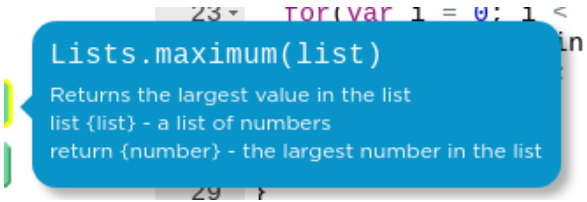
Reviewer Name: _____

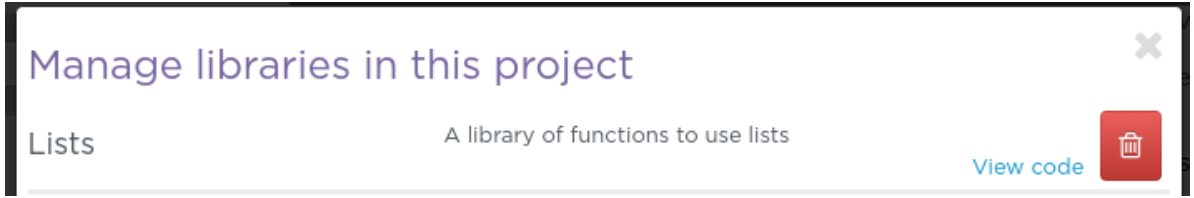
	Yes	Kind Of	No
Clear: I can easily understand the overall purpose of the library and each function within it			
Error-Free: Each function in the library works as expected.			
Useful: I can think of situations where I would want to use this library			

I like: Give feedback on at least one thing you like in the library

I wish: Give one problem or limitation of the library

What if: Give one idea for how to improve the library





Step 6 - Improve

- If you have time continue working on improving your library based on your feedback and testing from today

Wrap Up

Unit 7 - Lesson 10
Project - Make a Library Part 3

Warm Up

Activity

Step 7 - Acknowledge Collaborators

- Fill in the table acknowledging the source of any code your partner wrote or that you got from another source

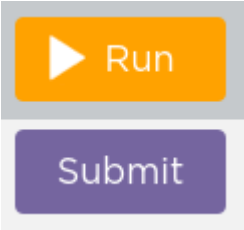
Step 8 - Free Response

- Complete the free response questions about one of the functions in your project
- If you have more time keep working on your library and check the Scoring Guidelines to make sure you're ready to submit.

Wrap Up

Submit

- Turn in your project guide
- Hit "Submit" on Lesson 10 Level 2



Unit 7 - Lesson 11
Assessment Day

Activity

Unit Assessment