Unit 7 - Parameters, Return, and Libraries

Unit Overview

Students learn how to design clean and reusable code that can be shared with a single classmate or the entire world. In the beginning of the unit, students are introduced to the concepts of parameters and return, which allow for students to design functions that implement an algorithm. In the second half of the unit, students learn how to design libraries of functions that can be packaged up and shared with others. The unit concludes with students designing their own small library of functions that can be used by a classmate.

Course Master Vocabulary

Unit Philosophy and Pedagogy

- **Learning by Building Libraries:** In the second EIPM sequence of this unit, students learn to use the Student-Create Libraries tool in App Lab. This tool allows them to build and share libraries of functions that can be used in many different projects . This tool serves many purposes besides simply teaching students about libraries. By having to write functions that other students find useful, they'll need to think about common patterns or situations that they've seen in projects across the course. Students will also practice commenting their code so others can understand how it works, practice designing functions that use parameters and return.

Teaching Tip

Modifications for Virtual and Socially-Distanced Classrooms

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

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

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

- **Final Preparation for the Create PT:** Students learn very few new concepts in this unit; nevertheless, it can be challenging because students need to learn how to integrate the ideas of parameters and return with every other concept they've learned in this course so far. This unit presents a good opportunity to do a final review of every programming construct covered in the course as students prepare to demonstrate what they've learned on the Create PT.

Major Assessment and Projects

The unit project asks students to design a library of functions that they can share with classmates. Their library must contain at least two functions and at least one of those functions must include a parameter, return, a loop, and an if-statement. This requirement ensures students practice skills they'll use in the Create PT. Using a project guide, students choose a theme for their library, build it, test it, and exchange feedback with other students. Students submit their library code, project guide, and written responses to reflection questions about how the app is designed and the development process they used to make it. They will also complete an end-of-unit assessment aligned with CS Principles framework objectives covered in this unit.

AP Connections

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

- CRD-2: Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.

- AAP-2: The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.
- AAP-3: Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.

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

- 3.12 Calling Procedures
- 3.13 Developing Procedures
- 3.14 Libraries

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

- 3.c.i
- 3.c.ii
- 3.c.iii
- 3.c.iv
- 3.d.i
- 3.d.ii
- 3.d.iii

Week 1

Lesson 1: Parameters and Return Explore

Develop a mental model for how functions can be generalized using parameters and return values.

Lesson 2: Parameters and Return Investigate

App Lab

Investigate and modify sample apps that use parameters and return values.

Lesson 3: Parameters and Return Practice

App Lab

Practice programming with parameters and return values through a set of programming puzzles.

Lesson 4: Parameters and Return Make

App Lab

Make an app that uses functions with parameters and return.

Lesson 5: Libraries Explore

Develop a mental model for how libraries work.

Week 2

Lesson 6: Libraries Investigate

App Lab

Investigate and modify sample apps that use libraries learn how libraries help programmers simplify and reuse their code.

Lesson 7: Libraries Practice

App Lab

Practice programming with libraries through a set of programming puzzles.

Lesson 8: Project - Make a Library Part 1

Project | App Lab

The first day of a three day project where students build and test their own libraries.

Lesson 9: Project - Make a Library Part 2

Project | App Lab

The second day of a three day project where students build and test their own libraries.

Lesson 10: Project - Make a Library Part 3

Project | App Lab

The third day of a three day project where students build and test their own libraries.

Week 3

Lesson 11: Assessment Day

Project

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

Lesson 1: Parameters and Return Explore

Overview

Students work with envelopes and paper to model functions with parameters and return values. Students create their own physical function envelope for drawing a house that takes in different parameters, and then build another function to calculate and return the cost of building that house.

Purpose

In this Exlore lesson, students follow a presentation and use manipulatives to understand the basics of creating a function with parameters and return values.

Agenda

Lesson Modifications
Warm Up (5 mins)
- **Preview Parameterized Functions**

Activity (30 mins)
- **Functions with Parameters and Return**

Wrap Up (10 mins)
- **Assessment: Check For Understanding**

View on Code Studio

Objectives

Students will be able to:

- Use appropriate vocabulary to describe parameters and return values.
- Remove specifics from a function so that it can be used in a variety of situations

Preparation

- [ ] Collect various sizes of sticky notes. Small sticky notes are particularly useful for this activity.
- [ ] Two envelopes per group
- [ ] Pencils/pens
- [ ] One baggy for a teacher demonstration (optional)
- [ ] Print out copies of the Activity Guide for students
- [ ] Review the Unit 7, Lesson 1 slides prior to the class

Links

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

For the Teachers

- **CSP Unit 7 - Parameters, Return, and Libraries** - Presentation

For the Students

- **Activity Guide - Function Houses**
  Make a Copy

Teaching Guide

Lesson Modifications

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

Warm Up (5 mins)

Preview Parameterized Functions

Remarks

> Up to this point, we've learned the basics of programming: variables, conditionals, functions, lists, loops, and traversals. As you grow in your programming skills, you'll start to notice ways that you can organize your code so that it is easier to work with. That's what this unit is all about!

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

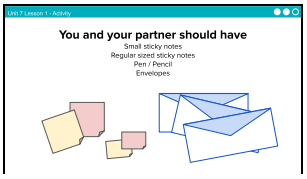
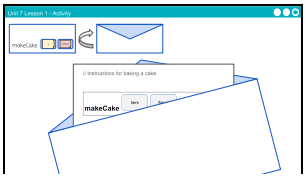
Discussion Goal

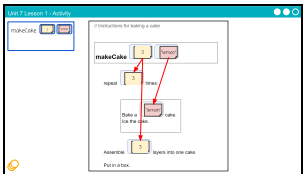
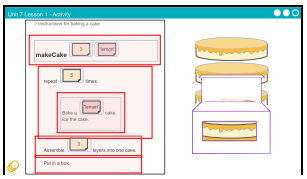
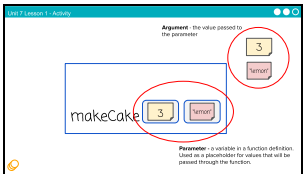
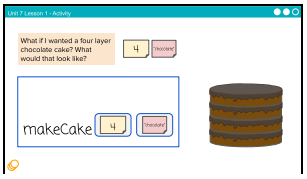
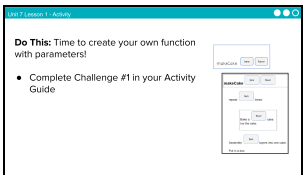
Goal: This is a good time to remind students that when we write code, we aren't just writing for a computer - we are also writing for humans who will need to understand our code. Better organization can help others quickly understand our code so they can help debug or build upon the code.

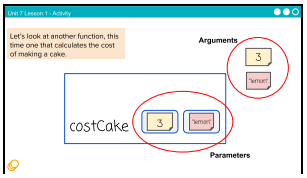
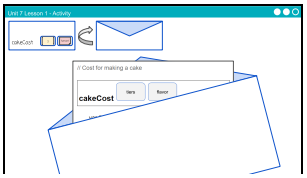
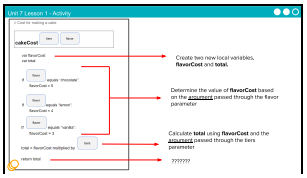
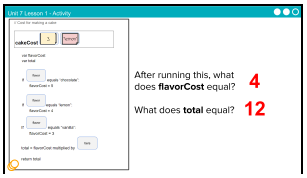
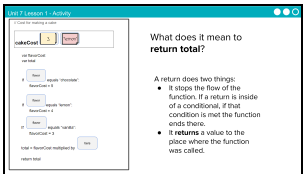
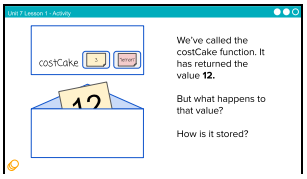
Activity (30 mins)

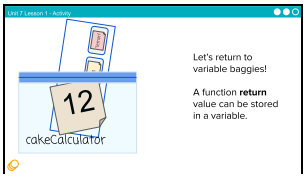
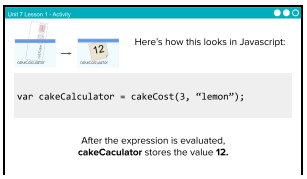
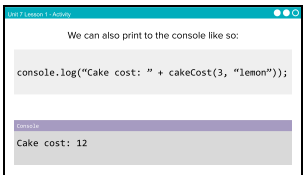
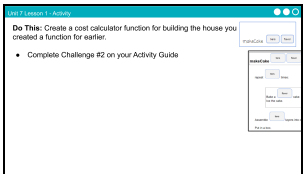
Functions with Parameters and Return

Display: Use the activity slides for this lesson to guide the unplugged activity on Functions with Parameters and Return.

Slides	Speaker Notes
	**Distribute:** large and small sticky notes and envelopes. Hold off on distributing the Activity Guide until later in the activity.
	**Say:** Let's pretend that I want to make a cake. I need to decide how many tiers the cake will have, and what flavor the cake will be. I've decided to make a 3 tiered cake. **Click for animation**
	**Say:** If I flip over the envelope, and open it up there's a recipe inside. This recipe is going to tell me how to make the cake I've specified.

Slides	Speaker Notes
	**Say:** To do this, I'm going to replace `tiers` with `3` and `flavor` with `lemon`, every time those words show up in the recipe. But notice - this recipe is pretty general. I could use it to make lots of different types and sizes of cake. **Click for animation** **Do This:** Read through the recipe as a class. Make sure everyone understands how it works before moving on.
	**Say:** Ok, let's make our cake. Remember, each time I come across `tiers` or `flavor`, I'm going to replace that word with the appropriate value. Once I'm done, I will have baked a 3 layer lemon cake! **Click for animation:** There are several steps to click through here. Take it slow so students can see each part.
	**Say:** This is a function! It should look familiar, but it has a few extra parts. This function has parameters. Parameters are used as plaeholders for values that will be passed through the function. **Click for animation** **Say:** Those values passed to the parameter are called arguments. **Click for animation**
	**Say:** Let's think through how changing the arguments affects the output. Now I want a four layer chocolate cake. With a partner, discuss what my arguments will look like. **Click for animation** **Say:** I pass through the arguments `4` and `chocolate` to the parameters `tiers` and `flavor`. Then my function is ready. What will my cake look like? **Click for animation**
	**Do This:** Distribute Activity Guides and direct students to Challenge #1. Students will create their own function envelopes and write a simple function that uses parameters which can be used to draw a house. When students are finished building their function envelopes, a partner tests out the code and draws the house on a spare sticky note.

Slides	Speaker Notes
	**Say:** Here's another function. This function is used to calculate the total cost of making a cake. Take note again of the arguments and parameters. **Click for animation**
	**Say:** Let's open the envelope and check out how this function works.
	**Say:** There are a few new things here. First, we've created two local variables. Remember, the local variables are contained within the function and can't be updated or accessed outside of the function. We learned about this when we discussed variable scope. **Click for animation** In this function, a decision (in this case, how much to charge per flavor) is made based on the argument passed through the parameter. **Click for animation** Finally, the total is returned, but returned where? What does that mean? **Click for animation**
	**Say:** Before we talk more about that return, let's calculate the total cost of our original cake. **Do This:** With a partner, determine what `flavorCost` and `total` equal after the function is run. **Click for animation**
	**Say:** Now let's think about the return. If the program hits a return at any point in running a function, it will stop and a value will be "returned". Where is it returned? To the place in the program where the function was called.
	**Say:** So to get back to our example, the value 12 is returned, but where is it returned to? Where is it stored? **Do This:** Take a moment to brainstorm with a partner.

Slides	Speaker Notes
	**Say:** Let's return to variable baggies! A function return value can be stored in a variable. Notice how the envelope is placed inside of a baggie. That means that the function would be evaled and whatever is returned would be stored in the variable `cakeCalculator`. **Click for animation**
	**Say:** Here's how this looks in JavaScript. It's ok if it's a little confusing. You'll get to play with this in Code Studio in the next lesson. For now, notice how the statement is set up. A variable is set up with the name `cakeCalculator`. This gets the value that is returned from the function `cakeCost` which has two arguments `3` and `lemon` passed through the parameters. After the function is evaluated, `cakeCalculator` now stores the value 12.
	**Say:** We can also print to the console like so. **Do This: Read through as a class.
	**Do This:** Direct students to complete Challenge #2 in their Activity Guides. Students create a new envelope function which calculates the cost of the house from Challenge #1. After students are finished, they share the envelopes with a partner who tests the code.

Wrap Up (10 mins)

Takeaways: Review the slide with the class.

- 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

Journal: Students should add the following words and definitions to their journals: parameter, argument, return.

Assessment: Check For Understanding

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

Question: Read through the code and choose the correct answer.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

CSP2021

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

Lesson 2: Parameters and Return Investigate

Overview

In this lesson students work with partners to investigate two different apps that use parameters and return values. Students are also introduced to the mod operator as part of one of the apps that they use.

Purpose

As with all Investigate lessons, this is an opportunity for students to dig into programs that use a new concept, in this case parameters and return, in the context of working apps. Encourage students to read the code carefully, discuss their findings with classmates, make connections to the Explore activity from the day before, and start making simple modifications to the program. Students will not leave this lesson as experts in writing programs with parameters and return, but they should understand the high level context and know they can refer back to the code in these investigate projects when they need help getting unstuck in the coming Practice and Make lessons.

Agenda

Lesson Modifications
Warm Up (5 mins)
- **Who is Clean Code For?**

Activity (35 mins)
Wrap Up (5 mins)
- **Assessment: Check For Understanding**

View on Code Studio

Objectives

Students will be able to:

- Identify situations in which a function with a parameter or return value would be necessary
- Explain the benefits of using a function with a parameter or return value in the context of a specific program
- Modify programs that use functions with parameters and return
- Use the modulus operator in a program

Preparation

- [ ] Review the two apps that students will investigate in the lesson and make sure you understand the goals of the discussion prompts

Links

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

For the Teachers

- **CSP Unit 7 - Parameters, Return, and Libraries** - Presentation

Teaching Guide

Lesson Modifications

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

Warm Up (5 mins)

Who is Clean Code For?

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.

Discuss: Have students silently journal their response, then discuss with a partner, and finally share with the class.

Discussion Goal

Goal: Parameters and return values allow students to write programs that are more organized and cleaner. Naming functions helps students write programs that read more like descriptions of what they do, and they also help students reuse code.

This quick discussion sets up many of these realizations. Students will not leave this lesson able to "do more" than previously. Instead they'll be able to write programs that are cleaner, better organized, and better able to be used elsewhere. Long term, writing programs like this will actually help students write more complex code, but because they as programmers will better be able to understand their programs and reuse the code they write. This quick prompt does not need to cover all of those points and should just be a reminder of things they've already seen. Clean code is for humans, but in the long term it helps them write better programs.

Remarks

> Clean programs may sometimes run "better" but usually they're just helpful for us as we read them and write them. In other words, clean code is for people, but in the long run writing good code can help us write larger and more complicated programs because we can just read them and understand what they're doing.
>
> Today we're going to learn more about parameters and return values. Something important to notice is that they might not actually change the way that your programs run, but they definitely will change the way we read and write programs. In general they'll make programs cleaner and easier to understand, but let's go look at some examples.

Activity (35 mins)

Do This: Take a few minutes to review the takeaways from the previous lesson.

Group: Place students with partners

Do This: Students navigate to Level 2 and read and respond to the questions there. If students have extra time, they can try the modify activity.

Teaching Tip

Make sure you leave time to bring the class back together for discussion and code walkthroughs. There's a lot to dig into with these Investigations.

- How does calculate() work?
- What are the arguments passed through the parameter in calculate() when it is called?
- What types of data does the parameter require in the upgradeClickerCheck() function? Where can you find that information?
- What is returned? What type of data?

Discuss: As a class, walk through the questions and the code. Make sure students understand the parts of the function.

Do This: Now look together at lines 30-36. Students discuss with a partner how they think the MOD operator works.

Do This: The MOD operator is just like any other operator - it takes two numbers, processes them, and then returns a value. Click through the animations one by one as students give answers to each arithmetic statement.

> **Teaching Tip**
>
> Students need to know how the MOD operator works for the AP CSP exam. The explanations here are enough to help them answer those questions.
>
> There are many wonderful and interesting uses of MOD in apps, but students are not expected or required to use the operator themselves when programming apps for this course.

Do This: Give students a few minutes to practice using the MOD operator with a partner, solving the problem on the screen. Then click through to see the answer.

Remarks

> MOD can be a little tricky, but here's what you need to remember: it's the reaminder that is left after a number is divided by another number.
>
> Why 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 also use MOD to determine if a number is divisible by another number.
>
> Now let's get back to another app investigation with functions with parameters and return values.

Do This: Students navigate to Level 3, read likes 1-14 and discuss with their partner what's happening in these lines.

Discuss: Read through the lines together as a class, and clear up any misconceptions.

Do This: Students carefully read the function on lines 15-34 before explaining to their partners how it works. If students have extra time, they can try the modify activity.

Discuss: Talk through the app together as a class. Make sure students understand all the parts of the function and understand where to look in the comments for information.

Wrap Up (5 mins)

Review: Discuss the takeaways on the Wrap Up slide.

> **Discussion Goal**
>
> **Goal:** Students are introduced to procedural abstraction on this slide, a concept they will return to throughout the unit. Spend a few minutes discussing what "generalize" means - focusing on how a single function can be used to accomplish many different tasks by using parameters.

Assessment: Check For Understanding

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

Question: What are the benefits of writing functions that use parameters and return? Try to list at least two.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

CSP2021

- **AAP-2** - The way statements are sequenced and combined in a program determines the computed result
- **AAP-3** - Programmers break down problems into smaller and more manageable pieces

Lesson 3: Parameters and Return Practice

Overview

Students practice writing programs with parameters and return values by creating and debugging functions that use them.

Purpose

This is students' opportunity to practice using parameters and return values in a variety of contexts. The progression of levels builds in complexity from students merely calling functions with parameters to designing functions and debugging complex programs that use parameters and return values. This lesson should build students familiarity with these concepts as they prepare for the Make project in the next lesson.

Agenda

Lesson Modifications
Warm Up (5 mins)
- **Quick Warm Up**

Activity (35 mins)
- **Practice Time**

Wrap Up (5 mins)
- **Assessment: Check For Understanding**

View on Code Studio

Objectives

Students will be able to:

- Correctly set up a parameter in a function
- Correctly set up a return value in a function
- Write comments to explain the function purpose, parameters, and return values

Preparation

- [ ] Review the programming levels students will complete

Links

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

For the Teachers

- **CSP Unit 7 - Parameters, Return, and Libraries** - Presentation

Teaching Guide

Lesson Modifications

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

Warm Up (5 mins)

Quick Warm Up

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?

Discuss: Have students share responses with a neighbor before discussing with the whole class.

Remarks

> Parameters and return values allow us to write code that is neater and works in a larger number of scenarios. We're going to have to practice reading, writing, and debugging code like this however, because it works a little a differently than the programs we've gotten used to writing. Let's dive in!

Discussion Goal

Goal: Use this quick prompt to review what students should already have seen about parameters and return values while also prompting them to think about how they'll need to use those skills in today's lessons. You should use your remarks to review the fact that parameters and return help write neater programs, but they may be a little tricky at first to read and understand. That's why we practice!

Up to this point, students have only seen complete traversals - that is, traversals where every element of a list is accessed. Now that they've been introduced to return, students can build partial traversals where a list is only partially accessed. For example: A list could be traversed to find a specific element. Once that element is found, "found element x" is returned, and the traversal stops. While students don't actually need to build partial traversals themselves, they now should know how it theoretically could be created.

Activity (35 mins)

Practice Time

Group: It is recommended that students work in pairs for this lesson.

Remarks

> Today you're mostly going to practice what we've learned about programming with parameters and return values. In general we want to use the debugging process we've practiced all year, but here's some specific things to look out for in today's lesson.
>
> - Use console.log to call functions with parameters with different values to see how they run
> - Slow down the code with the speed slider to see how the code is running once your function is called

Levels 2-6: These levels involve functions with parameters and return values, working in the console

- Level 2: Calling a function multiple times with a single parameter
- Level 3: Students follow a pattern to choose a randomCompliment from a list that is concatenated with a name passed through a parameter.
- Level 4: This is the same as Level 2, except now the functions use return values. Again, students follow a pattern given in another working function.
- Level 5: Students write a function that returns the season a month is in. By this point, students should be noticing the type of commenting that is happening above each function. This format previews how students will

Teaching Tip

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

need to write comments for libraries later on in the unit.
- Level 6: This level traverses lists to find the longest and shortest words. Parameters and return values are used.

Levels 7-8: Students continue practicing, now working with apps

- Level 7: Students debug and comment an ordering app. Completing this level will require students to read programs that use functions with parameters and return carefully.
- Level 8: We return to the compliment code, this time reworking it into an app. Again students write their own comments, this time including comments for a return value. Students also complete the `updateScreen` function, which calls both of the functions with parameters.

Level 9: Students do a quick practice with the MOD operator.

Extension Opportunities:

- Level 8: Add more greeting and compliment options to the lists

Wrap Up (5 mins)

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?

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

Discussion Goal

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

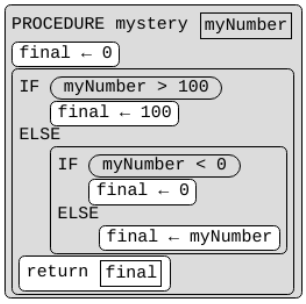
Remarks

> Working with parameters and return values can be tricky. We will get more practice tomorrow by making an app that uses lists to store information.

Assessment: Check For Understanding

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

Question: The procedure below will be called twice with two numbers provided as arguments to the parameter `myNumber`. Which of the following pair of numbers will `mystery` return the same value?



Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result
- ▶ **AAP-3** - Programmers break down problems into smaller and more manageable pieces

Lesson 4: Parameters and Return Make

Overview

Students use the knowledge and skills they've developed working with parameters and return values to develop a Rock Paper Scissors App. Unlike typical Make projects, students are given a significant portion of the starter code but are given three functions with parameters and return values that they'll need to develop themselves. At the end students submit their apps which can be assessed using a provided rubric.

Purpose

Make lessons typically ask students to take on an entirely blank screen when writing their program. In this instance students have to take on the "blank function" which needs to fit within the larger program they're developing. This is part of a broader mindset shift that students develop in this unit. Functions with parameters and return values operate like their own "mini programs" which can be written and tested independently. There is actually less difference than students might think between previous make projects, in which students get the user interface elements and no code, and this project, in which they get some starter code but blank functions with parameters. In each case they need to develop programs that integrate into a larger portion of the project. The focus of this unit is how functions with parameters and return values help further split programs into logical chunks like this.

Agenda

Lesson Modifications
Warm Up (5 mins)
Activity (35 mins)
- **Build the Rock Paper Scissors App**

Wrap Up (5 mins)
- **Assessment: Grading the Project**

View on Code Studio

Objectives

Students will be able to:

- Write functions with parameters and return values that meet a set of specified requirements
- Debug programs that use functions with parameters and return

Preparation

- [ ] Review the app students will need to make and ideally spend some time working on the project yourself to anticipate the challenges students will encounter

Links

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

For the Teachers

- **CSP Unit 7 - Parameters, Return, and Libraries** - Presentation

For the Students

- **CSP Parameters and Return Make - Rock Paper Scissors App** - Activity Guide

Make a Copy

Teaching Guide

Lesson Modifications

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

Warm Up (5 mins)

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

Discuss: Have students share their ideas with a partner and then discuss as a class.

Remarks

> Today we're going to be working on another Make project, but this one is a little bit different than others we've done in the course. A lot of the code will already be written for us, and we'll be working on blank functions, rather than entirely blank programs. Thanks to the way that parameters and return work, it's easy to design programs this way.

Discussion Goal

Goal: This discussion should prompt students to better understand what turns out to be a slightly different make project. Today they're going to be taking on "blank functions" rather than "blank screens". This is because with parameters and return values you can chunk out your programs into pieces that have well-defined behavior. Remind students that this is known as procedural abstraction. If students don't land on these ideas don't aim to force it but return to this idea in the wrap up.

Activity (35 mins)

Build the Rock Paper Scissors App

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

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

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

Review: Quickly review what students will be doing again before they move on to the next step.

Distribute: Give students copies of **CSP Parameters and Return Make - Rock Paper Scissors App - Activity Guide** if you will be using it during the class.

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

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

Wrap Up (5 mins)

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

Remarks

> Parameters and return help us chunk programs into pieces. Each important piece can be made into its own function which can be written and tested separately from the others. If you and your partner agree in advance how to split up the way a program is written and are clear on what each function needs to do, you could develop a program just like this make project.

Assessment: Grading the Project

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

Teaching Tip

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

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

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

Discussion Goal

Goal: This prompt should help review ideas that students saw in this project, in particularly that parameters and return help break programs into chunks. It should also help them see that this approach could be used to collaboratively design programs. If teammates agree in advance what each function should do they can go off and write them separately before coming back together to make the entire app work.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

CSP2021

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

Lesson 5: Libraries Explore

Overview

Students learn the basics of libraries while building upon the envelope model of a function with a folder to represent a library.

Purpose

The mental model of a folder containing a group of functions is used to introduce the concept of a library. Throughout this lesson student explore the requirements for working with libraries and documentation.

Agenda

Lesson Modifications
Warm Up (5 mins)
- **Preview Libraries**

Activity (35 mins)
Wrap Up (10 mins)
- **Assessment: Check For Understanding**

View on Code Studio

Objectives

Students will be able to:

- Use appropriate vocabulary to describe libraries
- Explain the process of preparing a function to be added to a library
- Clearly write documentation for functions in a library

Preparation

- [ ] Review the slides and click through all the animations
- [ ] Watch the Libraries video which introduces the Libraries tool in App Lab

Links

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

For the Teachers

- **CSP Unit 7 - Parameters, Return, and Libraries** - Presentation

Teaching Guide

Lesson Modifications

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

Warm Up (5 mins)

Preview Libraries

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

Remarks

> Today we are going to learn a new way to share code.

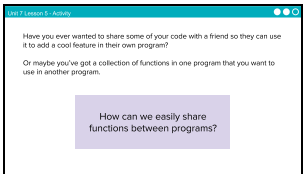
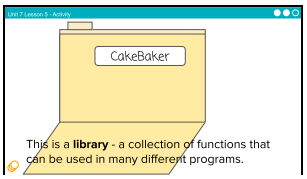
Discussion Goal

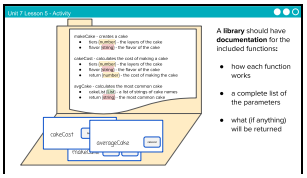
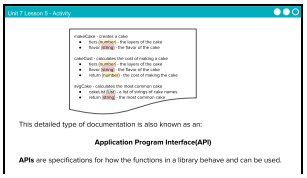
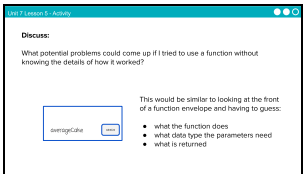
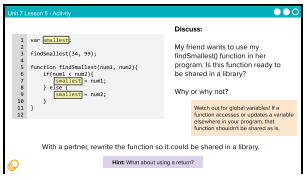
Goal: Guide student answers towards possible answers like these -

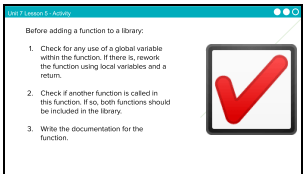
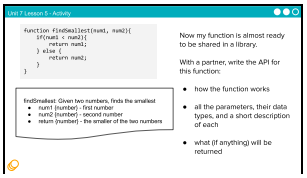
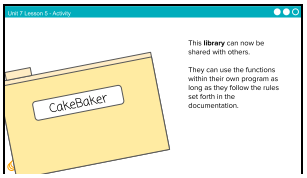
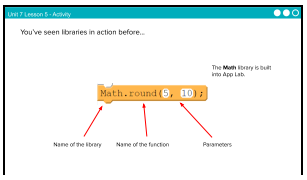
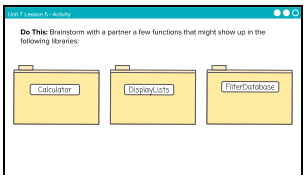
- email code to each other
- used a shared document
- copy/paste functions from one tab to another

Activity (35 mins)

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

Slides	Speaker Notes
	**Say:** 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 program? Or maybe you've got a collection of functions in one program that you want to use in another program. Today we are going to explore the answer to this question: How can we easily share functions between programs?
	**Click for animation:** Stop once all three envelopes are on the screen. **Say:** These are functions. We recently learned about functions with parameters and return values. **Click for animation:** Click until the library definition appears on the screen. **Say:** The envelopes have been grouped together in a folder. This represents a library. A library is a collection of functions that can be used in many different programs.

Slides	Speaker Notes
	**Say:** Let's take a look at what's inside the library. If we open up the folder, we see the functions and some documentation. A library should have documentation for each of the included functions. The documentation should include: • how each function works • a complete list of the parameters • what (if anything) will be returned
	**Say:** This detailed type of documentation is also known as **Application Program Interface(API)**. APIs are specifications for how the functions in a library behave and can be used. **Discuss:** With a partner, review the parts of the documentation.
	**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 **Note:** If students are struggling to come up with ideas, give a few hints: • what if there's an error when you run the code which calls this function? • what happens to a return if you don't store it in a variable or print it to the console?
	**Discuss:** My friend wants to use my `findSmallest()` function in her program. Is this function ready to be shared in a library? **Click for animation** **Say:** 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. **Click for animation** **Do This:** With a partner, rewrite the function so it could be shared in a library. **Click for animation** **Note:** The problem here, is that the function uses a global variable. Students should try to rewrite the function so only local variables and a return are used. The answer is shown later on in the slides.

Slides	Speaker Notes
	**Say:** These are the things you should consider 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. 2. Check if another function is called in this function. If so, both functions should be included in the library. 3. Write the documentation for the function.
	**Say:** Here's the same function, now written to be shared in a library. How does it look compared to yours? It may look a little different - and that's ok, as long as the function works as expected. 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 **Click for animation** When the class is ready, click through to see a possible answer.
	**Say:** A library needs a name. For our libraries in this class, we are going to follow the following rules: • No spaces • Capitalize the first letter **Click for animation**
	**Say:** This library can now be shared with others. They can use the functions within their own programs as long as they follow the rules set forth in the documentation. **Click for animation**
	**Say:** You've seen libraries in action before. The Math library is built into App Lab. Notice the name of the library first, then after a dot, the name of the function and finally the parameters. This is the same formatting you will use to call functions in libraries: Library name, followed by function name, and finally the parameter(s)
	**Do This:** Brainstorm with a partner a few functions that might show up in the following libraries: Calculator, DisplayLists, FiterDatabase. **Note:** Here are some ideas: • **Calculator:** functions similar to what you would find in a common calculator • **DisplayLists:** functions that help you display a list on a screen in different ways, such as one list item per line or divided with bullet points • **FilterDatabase:** functions that allow you to filter a database for different things, like all the items that start with the letter "a"

Remarks

> Now let's take a look at how libraries work in App Lab.

Video: Watch the Libraries video.

Wrap Up (10 mins)

Journal: Have students add the following words to their journals: library, API

Remarks

> Great work today! Libraries are different than other programming concepts we've explored like variables and conditions. They offer a way to organize and share your code with others.

Assessment: Check For Understanding

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

Question: Evaluate if the following function is a good candidate to be placed in a library. Why or why not?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

CSP2021

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

---

Lesson 6: Libraries Investigate

Overview

In this lesson students work with partners to investigate two apps that use libraries as well as the code used to make a library. Through a series of guided discussions they learn how libraries help programmers simplify and reuse their code.

Purpose

Students get a chance to see how libraries are used in actual apps before they begin building libraries themselves in the subsequent project. The sample apps are written in a clean and simple manner, helping students understand the usefulness of well-documented libraries.

Agenda

Lesson Modifications
Warm Up (5 mins)
Activity (35 mins)
- **Level 2**
- **Level 3**

Wrap Up (5 mins)
- **Synthesis**
- **Assessment: Check For Understanding**

View on Code Studio

Objectives

Students will be able to:

- Identify the use of a library within a program.
- Explain the purpose of libraries as a way to simplify programs, allow for code reuse, and enable collaboration.
- Test the functions in libraries in order to understand their behavior

Preparation

- [ ] Read the code for both sample apps and their associated libraries
- [ ] Practice sharing and importing libraries yourself in order to understand the process

Links

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

For the Teachers

- **CSP Unit 7 - Parameters, Return, and Libraries** - Presentation

Teaching Guide

Lesson Modifications

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

Warm Up (5 mins)

Prompt: 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?

Discuss: Have students share responses.

Discussion Goal

Some points that might come up:

- Parameters and return values allow functions to run in predictable ways without impacting other parts of the app unexpectedly.
- Parameters and return values help communicate what the code is supposed to do.
- Parameters and return values make functions more flexible so that they can be used in a variety of situations.

Activity (35 mins)

Group: Place students in pairs. One student per group should navigate to the lesson on Code Studio.

Level 2

Do This: Students navigate to Level 2 where they should run the app and try several different inputs.

Discuss: With their partners, students look at the project code and discuss what happens when the button is clicked. After a few minutes, pull the class back together and discuss as a whole.

Discussion Goal

When the button is clicked, the screen is updated by calling several functions from a library which return text and images that is displayed.

Do This: Now students should open the functions drawer and look specifically at each StateLibrary function. They should mouseover the functions for the documentation before discussing with a partner how these functions work.

Do This: Then students click "Manage Libraries" and "view code" following the screenshots on the slide. Students read the library code and discuss how the functions actually work. They should consider if they were accurate in their predictions before looking at the library code.

Discuss: 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?

Discussion Goal

Benefits: The project code is neat and organized.

Information: The end user does not have to know all the inner workings of the functions - they just need to know the documentation for the functions so they can call them correctly and understand what will be returned (if anything).

Level 3

Do This: Students navigate to Level 3 where they run the Pigify app and try several inputs.

Do This: Again, students open the functions drawer, look at the documentation for each function, and discuss how they work.

Test the Functions: Now students practice testing the functions to understand how they work. Here are the steps:

- 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: Now students navigate back to the States App and use `console.log` to test all of the library functions there.

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?

Review: Up to this point, students have either created their own algorithms from scratch, or modified existing ones (usually in Investigate Lessons). Now they have another tool to use: combining existing algorithms to make new algorithms. Here, this is accomplished with a library.

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

Discussion Goal

Goal: Testing the functions allows the end user to understand the behavior of the function. It's helpful for debugging functions, in addition to looking at the code in the library.

For example: If I call a function whose documentation indicates that it will return the first letter of a string, and instead it returns the last number of a string, I know that there is a problem with the library function and not my project code.

Discussion Goal

Goal: Hopefully through the process of testing, students will understand the importance of well-documented functions that do what they are expected to do!

A good library function contains everything it needs within the function. Beware of global variables or references to element IDs that might not be present in the end user's projects

Discussion Goal

Goal: The benefits of using existing algorithms as building blocks for constructing other algorithms include:

- reduced development time
- reduced testing
- simplification and identification of errors

We demonstrated the last two bullets in the exercise using console.log to test functions and isolate errors.

Remarks

> Now let's review Procedural Abstraction. In Lesson 2, we learned about Procedural Abstraction, where shared features of functions (also known as procedures) are extracted in order to generalize use. Procedural Abstraction encourages code reuse and manages complexity, because generalized functions can be used to accomplish many different tasks.
>
> Now let's extend that definition. Procedural Abstraction provides a name for a process and allows the procedure 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 ogranization in a program: Modularity - the subdivision of a computer program into separate subprograms. This is what we are doing when we organize functions into a library and then call them in another program.
>
> The end user of the library only needs to know how the functions work, which they can learn through the documentation. This means that the creator of the library can udate the functions for any number of reasons, such as making the functions more efficient, without having to notify the end users, as long as the function documentation does not need to be changed. The end users can expect the library to work as intended.

Do This: Review the takeaways on the slide. These are focused on the practicalities of creating and using libraries.

Wrap Up (5 mins)

Synthesis

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

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

Discuss: Have students discuss together before sharing with a class.

Journal: Students add to their journal the following word and definition: modularity. They may also want to update their definition for procedural abstraction based on earlier conversations and slides.

Discussion Goal

As you discuss focus conversation on the following points.

- Libraries help programmer collaborate because they can design code with the idea that other people will use them.
- Libraries help programmers reuse code because they can be imported into many different programs
- Libraries help programmers write simpler programs because library code can be used to handle many common (and sometimes complex) behavior and remove it from your core program.

Assessment: Check For Understanding

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

Question: Why is it important to use meaningful names for the functions in your library?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

CSP2021

- **AAP-2** - The way statements are sequenced and combined in a program determines the computed result
- **AAP-3** - Programmers break down problems into smaller and more manageable pieces

Lesson 7: Libraries Practice

Overview

Students practice important skills for working with libraries, including testing and debugging libraries, and using libraries to help develop apps. After a brief introduction to these practices by the teacher, students spend the majority of their time programming in a level progression.

Purpose

Using libraries highlights a number of important concepts, skills, and mindsets. Here is a short list

- Testing: Before sharing a library it is critical to make sure your functions actually work as intended. The best way to do this is to write tests that try out different cases of your program. By doing this like testing positive and negative numbers, or testing out edge cases, it's possible to assure yourself that your code works as you expect.
- Debugging: Closely related to the point above, debugging libraries can be tricky once the program is shared with another student. While students can test functions that were shared with them, it will be up to the original author of the code to make the changes and ensure their library is bug free.
- Abstraction: Libraries allow you to make blocks that are "higher level" than other blocks in the toolbox. The result is that the actual programs you write either will be shorter, since library code does the heavy lifting, or more complex, because library code allows you to focus on bigger problems. In either case, libraries help you focus on the high level problem you're solving rather than focus on the low level details, and this is an example of abstraction in programming.

Agenda

Lesson Modifications
Warm Up (5 mins)
    Quick Warm Up
Activity (35 mins)
    Practice Time
Wrap Up (5 mins)
    Assessment: Check For Understanding

View on Code Studio

Objectives

Students will be able to:

- Test functions designed to be used in a library using different input values
- Debug library code to remove any errors
- Read library code documentation in order to select the proper functions in the library to develop an app

Preparation

- [ ] Review the programming levels students will complete

Links

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

For the Teachers

- **CSP Unit 7 - Parameters, Return, and Libraries** - Presentation

Teaching Guide

Lesson Modifications

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

Warm Up (5 mins)

Quick Warm Up

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

Discuss: Have students share responses with a neighbor before discussing with the whole class.

Remarks

> When we program with libraries we can use functions that take care of difficult or repetitive tasks. Instead of focusing on these details we can focus on the big picture of what it is that we want our programs to do and just assume that those details are handled by library functions. As we'll see in today's lesson that leads to two important realizations.
>
> - Our programs may often be shorter and easier to read
> - We need to debug libraries carefully before we share them to make sure they really work

Discussion Goal

Goal: You'll return to this prompt later in the lesson so use this discussion as much to plant the ideas here in students heads as to really hit them home. The main goal you're aiming for is the first bullet in the remarks at the end of the warm up. Libraries take care of detailed tasks so you can focus on the big picture. Students likely will not mention the fact that this increases the need to debug and test code in the first place but this is another idea that will be explored in the lesson.

Activity (35 mins)

Practice Time

Group: It is recommended that students work in pairs for this lesson.

Remarks

> Today you're mostly going to practice what we've learned about programming with libraries. In general we want to use the debugging process we've practiced all year, but here's some specific things to look out for in today's lesson.
>
> - Test library code using console.log. Call the function with many different values to make sure it works as you expect
> - It's much harder for someone to debug library code once it's sent to them. You'll need to make sure it works on your end.

Levels 2-3 Using Libraries: These levels involve students using libraries. These levels focus on students understanding that libraries make the programs that they write simpler and higher level.

- Level 2: Students build a simple app with the states library from the investigate lesson
- Level 3: Students build a simple app with a new library that uses the cats data set.

Teaching Tip

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

Levels 4-6 Debugging Libraries: Students practice debugging code that will be used in a library before sending it to a classmate. This is useful help before the project in the next lesson where students will develop a library.

- Level 4: Students debug a simple library for manipulating numbers
- Level 5: Students debug a simple library for manipulating strings
- Level 6: Students debug and finish writing a simple library for manipulating strings

Extension Opportunities:

- Levels 2 and 3: Students can add additional features to the two apps in levels 2 and 3 using the library functions made available.

Wrap Up (5 mins)

Prompt: How do libraries let you write programs at a "higher level"? Why is testing important when building and sharing libraries?

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

Discussion Goal

Goal: These prompts review the main things students should have seen in today's lesson and that were previewed in the Warm Up. Students should be able to better speak from experience about the fact that the programs they wrote were significantly simpler because they could just use library code. They should also have seen how testing of libraries is important because the user of their library is counting on the functions working exactly as they say they're supposed to.

Remarks

> Libraries are yet another level of procedural abstraction. They allow us to write programs that are high level and focus on the big picture. Our programs are shorter, and they read more like exactly what they are doing. This doesn't mean that there isn't more detailed code being run, it just means that once the library is written, perhaps by someone else, we don't have to think about it as much anymore.
>
> We're about to do a class project that will test our ability to think in this new way.

Assessment: Check For Understanding

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

Question: The following procedure was developed to determine if a list contains a negative number. If the list contains a negative number it should return true, otherwise it should return false.

An error was made in writing this function so that it does not work as intended. Which line of code would need to be fixed in order for the function to work as designed?

```
01 PROCEDURE checkNegative(list)
02 {
03  hasNegative <- true
04    FOR EACH number IN list
05    {
06      IF(number < 0)
07      {
08        hasNegative <- true
09      }
10    }
11  RETURN(hasNegative)
12 }
```

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Lesson 8: Project - Make a Library Part 1

Overview

In this lesson students begin a multi-day project designing a library of functions. Students will brainstorm common problems they've encountered while programming this year and begin to design functions that address those common problems.

Purpose

This project caps off everything students have learned about programming as they head into the Create PT. Students are literally "building blocks" that classmates will literally have access to in order to create projects of their own. This kind of thinking requires not only a mastery of programming concepts and skills, but the ability to think more abstractly about how programs are built. Students need to identify common situations or problems that they and other programmers may encounter, and help build commands to address that problem.

This project also caps off a unit that is all about abstraction. On one level students are thinking about procedural abstraction, and simply learning how to design functions. On another level, however, they're learning to think abstractly. They're not merely designing a function, but they're thinking about the entire process of how programs are developed and common problems or situations that arise. This mental approach to thinking about why to build a function is just as important as knowing the steps to design one.

Agenda

Lesson Modifications
Warm Up (5 mins)
Activity (40 mins)
Wrap Up (0 mins)

View on Code Studio

Objectives

Students will be able to:

- Select a theme for a library of functions
- Design the API for a library of functions, including the function names, purpose, and parameters, and types of values each function will return

Preparation

- [ ] Review the Project Guide to make sure you understand what students will be expected to do in the project
- [ ] Review the Examples Submission for the Project found in the last lesson of this project

Links

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

For the Teachers

- **CSP Unit 7 - Parameters, Return, and Libraries** - Presentation

For the Students

- **Project Guide - Make a Library** - Project Guide Make a Copy

Teaching Guide

Lesson Modifications

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

Warm Up (5 mins)

Prompt: 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?

Discuss: Have students brainstorm either individually or with a partner. Then have a few volunteers share their ideas with the class.

Remarks

> Today we're going to begin our final project before the Create PT. You'll be building a library of functions that you and others will be able to import and use. You'll get to customize App Lab.

Discussion Goal

Goal: This prompt foreshadows the project students will complete in which they'll be building a library of functions. Sharing this library will allow them to actually add new blocks to App Lab for themselves or their classmates.

Activity (40 mins)

Group: Students may complete this project individually or in pairs.

Distribute: Give students each a copy of the **Project Guide - Make a Library - Project Guide**

Project Description - 5 mins: As a class review the project description, what they'll submit, and the steps. Make sure students are also aware that they have access to the rubric.

Step 1 - Brainstorm - 5 Mins: Give students 5 minutes to brainstorm a theme for their library. Many specific ideas are given to students for the types of functions they could write, but ideally students will develop functions that are focused on a specific context they find interesting.

Step 2 - Design - 10 mins: Before students start writing the code for their project they should stend 10 minutes defining the way the different functions in their library should work. This means they'll need to write out the comments that describe the function, the different parameters, and what will be returned.

Teaching Tip

Differentiation: The size of library is a good way to differentiate in this project. Students who are more comfortable programmers should be encouraged to develop larger libraries of functions.

Difficulty of Suggested Functions: The functions suggested in the project guide are loosely in difficulty order. Functions that appear near the bottom of that list are actually quite difficult and will require students to build substantially upon things they've learned in the course.

Step 3 - Build - 20 mins: Give students the remainder of time in class to work on building out their library.

- Students should start by working on one of their functions that includes all four features. Students will need to develop a function like this as part of the Create PT and also will be answering questions about their function in the free response questions at the end of the project.
- Circulate the room offering support to students. If they have successfully completed step 2 of their project guides then they should be able to write a lot of code very quickly to set up their functions and comments.

Wrap Up (0 mins)

Remarks

> Great work today. Next time we meet you'll have more time to work on your libraries and then you'll need to share them with another group for feedback.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

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

Lesson 9: Project - Make a Library Part 2

Overview

Students continue to develop their library of functions. Students are encouraged to begin developing tests for their functions to make sure they return values as expected. Late in the lesson students exchange their library with a classmate for feedback.

Purpose

In this lesson students continue to write the code for their libraries, but a big focus is on making sure it works as expected. Through writing tests of their own students practice thinking through how different inputs into a function lead to expected outputs. Sharing their libraries with classmates helps check not only that their code works as expected, but that their library is clear, approachable, and useful, to someone else.

Agenda

Lesson Modifications
Warm Up (2 mins)
Activity (40 mins)
Wrap Up (0 mins)

View on Code Studio

Objectives

Students will be able to:

- Write tests for functions with a library that they designed
- Provide feedback to their classmate about a library they designed

Preparation

- [ ] Ensure students have access to their project guides from last class.

Links

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

For the Teachers

- **CSP Unit 7 - Parameters, Return, and Libraries** - Presentation

For the Students

- **Project Guide - Make a Library** - Project Guide Make a Copy

Teaching Guide

Lesson Modifications

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

Warm Up (2 mins)

Remarks

> Today you are going to have more time to work on your library, but you're going to focus on testing it in two very different ways. In the first half of the lesson you'll still have work time to write your library, but you should develop tests like we practiced in this unit to make sure they're working as you expect.
>
> In the second half of the lesson you'll share your library with a classmate who will try using it themselves and will give feedback. Next time we meety you'll be able to use this feedback and testing to make the final edits to your library.
>
> Let's get to it!

Activity (40 mins)

Step 4 - Test - 20 mins: In this section students briefly review how to write tests for functions and then actually go write them.

- **Tests Refresher:** Briefly review with students how to use `console.log` to write tests for their functions.
- **Test:** Give students time to write tests for their functions and then continue to program to fix any errors they encounter.

Group: Have students find a classmate with whom to share their library and give feedback. Students should trade project guides with their classmate.

Step 5 - Feedback - 20 mins: Students will need to go through a series of steps to share their libraries and given feedback

- **Export Your Library:** Walk students through the steps necessary to export their own library
- **Import a Library:** Walk students through the steps necessary to import a library from their classmate.
- **Give Feedback:** Walk students through ways that they can use a classmate's library to test it out. Make sure they give feedback on their classmate's project guide under Step 5.

Teaching Tip

Writing Tests: Remind students of these important strategies while testing their program code.

- They should actually call their functions with different inputs
- They should try out values that try out different ways their functions work, or right below, at, and after the cut offs of the conditional statements within their functions

Supporting the Feedback Process: There are a few logistical steps involved in getting students to provide feedback to one another. They'll need to exchange project guides, navigate to a different level from the one they're working in, and import a classmate's library. Once this is done make sure you encourage good processes in testing one another's libraries by asking questions

- Can you tell what the library is supposed to do?
- Do all the functions work as they're suppoed to? How do you know?
- What are ways the library could be improved?

Step 6 - Improve: Students should return project guides to their classmates and review the feedback. Next class they'll have time to improve their libraries and prepare to submit them.

Wrap Up (0 mins)

Remarks

> Great work today. At this point you should have a good sense of how your library is working and what edits you'll need to make. Tomorrow you'll finish your library and answer some questions about them.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

Lesson 10: Project - Make a Library Part 3

Overview

Students complete their library project, finalizing their code and writing written responses explaining the way one of the functions in their library works.

Purpose

This lesson wraps up this quick project but also gives students a chance to practice responding to questions that appear in the Create PT. This lesson and project should therefore serve as a cap on students' study of procedural abstraction and if you are teaching this class as an AP, prepare them for the Create Task that they will complete in the following unit.

Agenda

View on Code Studio

Objectives

Students will be able to:

- Debug a library of functions based on testing and classmate feedback
- Explain the purpose and functionality of a function they developed
- Explain two different calls to a function they developed

Preparation

- [ ] Ensure students have access to their project guides
- [ ] Review the **Example Project Guide Submission** and **Example Library** (in the blue teacher panel)

Links

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

For the Teachers

- **CSP Unit 7 - Parameters, Return, and Libraries** - Presentation

For the Students

- **Project Guide - Make a Library** - Project Guide Make a Copy

Teaching Guide

Lesson Modifications

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

Warm Up (2 mins)

Remarks

> Today is your final day to work on your library. Based on the feedback you received from your classmates and the tests that you wrote, you should know where you need to improve your functions. Today you'll also need to finish a set of written response questions about one of your functions expxlaining how it works and describing two different calls to that function with different arguments.
>
> Let's get to it!

Activity (40 mins)

Step 7 - Acknowledge Collaborators / Sources - 2 mins: If students worked in pairs or imported some code from elsewhere they should acknowledge the sources of that code.

Step 8 - Free Response - 20 mins: With students quickly review the questions students need to answer about one of the functions in their library. These questions are taken nearly exactly from the Create PT and should help students prepare to independently answer these questions when they complete that task.

Teaching Tip

Scheduling Time: Depending on where students are in the process of developing their libraries they may need to program a little more prior to starting their free response answers. Remind students that in order to finish their responses they only need to have written a single function that has a parameter, return value, loop, and conditional. As soon as students have completed a function like that they should move to free response answers. They can subsequently come back to finish developing the rest of their libraries.

Work Time - 18 mins: Students should work on their free response answers early in the lesson. Once they've finished working on their responses they can go back to their project to make any final edits to their project code. Encourage students to review the Scoring Guidelines to make sure they're not missing anything in their projects.

Wrap Up (3 mins)

Submit Projects: At the end of the lesson students should submit their projects in Code Studio and submit their completed project guides, including free response questions.

Assessment

Rubric: Evaluate the projects using the provided rubric in the Project Guide

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

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

Lesson 11: Assessment Day

Overview

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

Agenda

Lesson Modifications
Assessment (25 mins) (5 mins)
    Topic Coverage
Assessment Review (20 mins) (35 mins)

View on Code Studio

Preparation

Preview the assessment questions

Links

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

For the Teachers

- **CSP Unit 7 - Parameters, Return, and Libraries** - Presentation

Teaching Guide

Lesson Modifications

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

Assessment (25 mins) (5 mins)

Administer the Unit 7 Assessment, found on Code Studio. Make sure to unlock the assessment following instructions **here**

Assessment Review (20 mins) (35 mins)

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

Teaching Tip

Topic Coverage

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

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

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

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

Unit 7: Parameters, Return, and Libraries	Topic 3.12 Calling Procedures
	Topic 3.13 Developing Procedures
	Topic 3.14 Libraries

Click for more info: **Code.org CSP Topic Coverage**