

LESSON: Design and Call Functions		Time: 50 minutes
<p>Overview:</p> <p>This lesson introduces user-defined functions. Students define and call functions to organize their code and reduce repetition. Functions also help the readability of code by naming sections of code that accomplish a task. Abstraction is also introduced, which is an important concept in computer science. Students will define and call functions in three programs.</p>		<p>Objectives:</p> <ul style="list-style-type: none"> • I can define “abstraction” and “function” • I can define (create) a function • I can call a function • I can determine when a function can be written • I can state benefits for using functions
<p>Standards:</p> <p>2-CS-03 Systematically identify and fix problems with computing devices and their components.</p> <p>2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.</p> <p>3A-AP-17 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.</p>	<p>CSP Framework:</p> <p>Computational Thinking Practices:</p> <p>2.B Implement and apply an algorithm.</p> <p>3.B Use abstraction to manage complexity in a program.</p> <p>3.C Explain how abstraction manages complexity.</p> <p>4.C Identify and correct errors in algorithms and programs, including error discovery through testing.</p>	<p>Key Concepts:</p> <ul style="list-style-type: none"> • Abstraction is an integral part of computer science and coding • Programmers can write functions that accomplish a task • Functions must be called in order for their code to be run • Functions can be called in any order, and multiple times • There are many benefits to using functions
<p>Preparation:</p> <p>Make a copy of the assignment or put it in the LMS.</p> <p>Decide how you will have students turn in or show you their code (turn in text file, submit through LMS, paste into assignment, show teacher)</p> <p>Prepare any formative assessments you want to use in the wrap-up</p>	<p>Links:</p> <ul style="list-style-type: none"> • Assignment • Instructions slide deck <p>Resources:</p> <p>Sample code for mission 3, mission 4 and mission 6</p> <ul style="list-style-type: none"> • Folder with code 	<p>Agenda:</p> <ul style="list-style-type: none"> • Warm-up / Design process (5 minutes) • Modify mission 3 (10 minutes) • Modify mission 4 (15 minutes) • Challenges (15 minutes) • Wrap-up & Assessment (5 minutes)
<p>Vocabulary:</p> <ul style="list-style-type: none"> • Abstraction: the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics • Function: a named set of instructions that accomplishes a task 		
<p>Assessment:</p> <ul style="list-style-type: none"> • Daily reflection journal or Google form • Assignment completion • Demonstrate ability to define and call functions 		

Teaching Guide


Warm-up / Design Process (5 minutes)

This short warm-up is to introduce abstraction and functions. The assignment document is only used for the debugging table, and then at the end of the lesson, as a review.

Teaching tip – warm-up

- Go through slides #1-6
- Include any student discussion about abstraction and functions. Discussion can also wait until the wrap-up when students have experience with these two new terms.

Modify Mission 3 (10 minutes)


 Students will work in pairs (or individually) at their computers.

Teaching tip:

You can go over the instructions with the students, or you can use the slides (slides 7-14).

- Students are asked to modify their code from Mission 3. Students should be able to use their own code. However, if you want to give them code to use, starter code for Mission 3 is in the folder.
- The extensions were introduced in Mission 6. Adding them to this mission is good practice for students.

Modify Mission 4 (15 minutes)


 Students will work in pairs (or individually) at their computers.

Teaching tip:

You can go over the instructions with the students, or you can use the slides (slides 15-22).

- Students are asked to modify their code from Mission 4. Students should be able to use their own code. However, if you want to give them code to use, starter code for Mission 4 is in the folder.
- No extension for this part of the lesson – the extension is the mild challenge

Challenges (15 minutes)

 Students will work in pairs (or individually) at their computers.

Teaching tip:

Three challenges are given, with varying levels of difficulty. Students can pick the challenge they want. Also, depending on time, students can complete more than one challenge. (slides 23-26)

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

- Mild Challenge: an extension of Mission 4. Add a new function to reset all pixels. Then add a loop and kill switch to the code (very similar to the extension on mission 3)
- Medium Challenge: Create functions in Mission 6 (the heartbeat program). A couple of suggestions are given to the students about functions that can be created.
 - NOTE – if students are creating a function with the button presses, they are changing the value of the delay variable. We haven't talked about this yet, but they will need to add "global delay" as the first line of the function.



- Spicy Challenge: Students will create functions for their remix project. No suggestions are given, since each remix will be different.

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

Wrap-Up (5 minutes)

The wrap-up can be very short for this lesson (slides 27-28). This is the class's chance to review abstraction and functions and make their own meaning with the terms.

Formative Assessment:

- Daily reflection journal or [Google form](#)
- Class discussion on what they learned about abstraction and functions
- Assignment completion
- Defining and calling functions in Mission 3 and 4 (and Mission 6)
- Exit ticket

SUCCESS CRITERIA:

- Define "abstraction" and "function"
- Correctly define a function
- Understand how and when to call a function
- Define and call one or more functions in a program
- Determine when a function should be used in code