**Lesson 1** Functions

**How can we create our own custom functions?**

| **Overview** | |
| --- | --- |
| In this lesson, students will define their own functions for the first time. They will use them to make their code reusable in different contexts by turning a simple p5 sketch into a “stamp.” | |
| **Lesson Objectives** | |
| Students will be able to   * Define and call custom functions * Describe the advantages of defining functions | |
| **Suggested Duration** | |
| One to Two periods (45 - 90 minutes) | |
| **Blueprint Foundations Student Outcomes (**https://blueprint.cs4all.nyc/outcomes/) | |
| Abstraction Analyze | **Describe how** I might use patterns to express an idea. |
| Algorithms Analyze | **Describe how** instructions can have different outputs depending on inputs. |
| Algorithms  Prototype | **Demonstrate** the benefit of using an event, conditional, or loop in my prototype. |
| Algorithms Prototype | **Explain how** a function I prototyped can be used by someone else. |
| **Vocabulary** | |
| * **Function:** A named group of programming instructions that perform a specific task | |
| **Planning Notes** | |
| * Students will need to have a complete working function to participate in Lesson 2, so give them an extra period to finish if necessary. Students who finish early can work on the extension and also help debug their peers’ functions. * Make sure every student sends you the link to their function - you will need to compile them into a JavaScript file and share it during the next lesson. | |
| **Resources** | |
| * Video Tutorial: [Function Parameters and Arguments](https://www.youtube.com/watch?v=zkc417YapfE) | |
| **Assessments** | |
| * Assess the student activity. Check for ability to:   + Create and call functions in their programs * Assess the Wrap Up activity. Check for ability to:   + Explain the benefit of creating functions in programs | |

| **Do Now:** |
| --- |
| * **[Design Journals]** Give students the following prompt:  *Imagine that [teacher’s name] is a robot. Write a set of specific instructions for it to follow to get from the place where it is standing to outside the door.* * Call on students to read out their instructions and attempt to follow them **exactly**. Challenge them to try to get you to the door in the fewest number of commands! |
| **Discussion:** |
| * Tell students that they will be learning to write and use their own “custom” **functions**, which is one of the most fundamental programming concepts. Instead of just *calling* functions built into the p5 library, like ellipse() or fill(), they will make their own. * Show the syntax of a function definition by turning the instructions from the **Do Now** activity into a function using pseudocode: |
| **Teacher Demo:** |
| * Demonstrate how to define and call a custom function in p5 by using a simple design, like a lollipop. It only takes a few lines of code to make a single lollipop, but if you wanted to make a poster with many lollipops it would be much more efficient to use a function instead of repeating similar lines of code 30 or 40 times:      * The first step is to **define** the function underneath the draw() loop:      * Hit the “play” button to show students that nothing will happen when the code is run. This is because unlike setup() and draw() which are special p5 functions, custom functions **must be called** in order for the code to execute:      * Now try calling the lollipop() function 3 times:      * Turn and Talk: Explain that you’re trying to draw three different lollipops, but each lollipop looks exactly the same, and is appearing in the same position on the canvas. Ask: Why is this happening? Think about how the lollipop function is different from other p5 functions, like ellipse() or line(). * Students may notice that the lollipop() function is missing **parameters**. All of the values for fill(), rect() and ellipse() have been **hard-coded**, which means that the lollipop will always look the same. In order for a function to be **reusable** (for example, to make lollipops of different sizes or colors) you will need to use parameters in the function definition.   + You can make a comparison to the **Do Now** activity by explaining that the “function” that got you to the door would only work for someone **standing in your exact position**. The same instructions wouldn’t work for someone standing in the opposite corner. One of the ways that functions can be useful is their ability to work in different situations. * Begin by making a parameter to control the size of the lollipop. Explain that it’s similar to a variable, but it’s only given a value when the function is called.      * Try calling the lollipop function like this: lollipop(size); to show that p5 will display an error saying “size is not defined.”   + Ask: Why are we getting this error? *Answer: We need to use a real number when we call the lollipop function to tell the ellipse function what size to be*. * Remind students that in the same way they cannot create an ellipse by only typing ellipse(x, y, size); students will need to use an actual value, or an argument, when calling the function:      * Independent Practice: Update the function to add parameters for x and y position, and then create 3 lollipops with different positions and sizes inside draw(). * After the independent exercise, the lollipop() function should look like this (note that only one lollipop has been drawn in this example):      * Show students how to update this code to turn the lollipop into a “stamp” by moving the background to setup() and calling the lollipop() function inside mousePressed()   + Ask: What built-in p5 variables could you use to make position of the lollipop the same as the x and y coordinates of the mouse? *Answer: mouseX and mouseY*   + Ask: Why do we have to move the background() to setup() to see the lollipop? *Answer: So it is only called one time. If background() is called inside draw(), it will cover up the lollipop when the code loops.*      * Optional Pair Practice: Work with a partner to make the lollipop size a random value whenever the mouse is pressed.   + [Solution](https://editor.p5js.org/mparker/sketches/dPX6jEYPs) for teacher reference. |
| **Student Activity** |
| * In this activity, students will make their own custom functions and turn them into stamps. These functions should include:   + Two parameters for x and y position, and at least one other parameter of their choice (it might control size, color, or something else entirely).   + A comment above the function explaining what it does. * Some suggestions of possible designs are: lightbulb, fish, leaf, flower, heart, sun, or bumblebee. Encourage students to keep their designs as simple as possible! * Scaffold: If students are (unproductively) struggling to make a function from scratch, share [this sketch](https://editor.p5js.org/mparker/sketches/X5F8boGbV) with them and ask them to remix the **greeting()** function. They might rewrite the greeting message to say something other than “Hi”, include parameters to control text size or position, or add shapes to change the design of the message.   + Note: If more than one student remixes the greeting() function, they will need to change the name of the function before it is added into the class’s custom function library, since there cannot be duplicate function names. * Extension: Early finishers should review [this example](https://editor.p5js.org/mparker/sketches/0NTFt6NKy) and try to make a second version of their function that includes a conditional statement. |
| **Wrap Up** |
| * **[Design Journal]** Ask students to answer the following question:   + What is the benefit of creating functions in your code? * Make sure to collect students’ functions before they leave so you can compile them into a new JavaScript file to share during the next lesson. |
| **Extensions:** |
| * N/A |