Defining Functions

Python programs typically have multiple functions, each of which has multiple statements. Defining new functions allows you to break down a complex program into smaller blocks of reusable code.
This activity uses Python Tutor (by Philip Guo) to visualize the flow of execution. Each unit is interactive—to encourage discussion, have only one student per team run the visualization.

At the end of, explain the difference between parameters and arguments.
Remember `None` is a built-in constant like `False` and `True`.

Unit 1 Flow of Execution

In addition to using Python's built-in functions (e.g., `print`, `abs`) and functions defined in other modules (e.g., `math.sqrt`), you can write your own functions.

```
1  def model_one():
2      word = input("Enter a word: ")
3      L = len(word)
4      ans = word * L
5      print(ans)
6
7  def main():
8      print("Starting main...")
9      model_one()
10     print("All done!")
11
12 main()
```

Questions

1. Based on the program in Unit 1:

a) What is the Python keyword for defining a function?

b) On what line is the `model_one` function... defined? called?

c) On what line is the `main` function... defined? called?

2. Open a web browser and go to PythonTutor.com. Click on "Visualize your code", and type (or paste) the program above. Make sure the line numbers match.

3. Click the "Visualize Execution" button. As you step through the program, pay attention to what is happening on the **left side** of the visualization.

a) What does the **red** arrow indicate?

b) What does the **green** arrow indicate?

4. Notice the order in which the program runs:

a) After line 12 of the program executes (Step 3), what is the next line that executes?

b) After line 9 of the program executes (Step 6), what is the next line that executes?

5. In terms of execution order, what is the effect of calling a function?

6. Consider the definition for a function called `str_to_list` that prompts the user to enter a word. The function converts the string to a list and print the list.

```
def str_to_list():
    word = input("Enter a word: ")
    ans = list(word)
    print(ans)
```

Edit the program in Python Tutor so that, instead of defining and calling the function `model_one`, it defines and calls the function `str_to_list`. Verify your changes by visualizing the execution, and draw a picture of the right side immediately after the list is printed.

Unit 2 Passing Arguments

Instead of using `input` inside a function to get data, we can define a function to take a ***parameter*** (variable). When we call the function, we need to provide an ***argument*** (value). Change the program in Python Tutor as follows:

```
1  def model_two(word):
2      ans = word * len(word)
3      print(ans)
4
5  def main():
6      print("Starting main...")
7      w = input("Enter a word: ")
8      model_two(w)
9      print("All done!")
10
11 main()
```

Questions

7. Underline the parameter in the `model_two` function definition, then circle each use of the parameter inside the function.

8. Find the `model_two` function call in `main`, and underline the argument being passed by the function call.

9. Visualize the execution of `model_two` until Step 8.

a) How does the frame for `model_two` at this point in the execution differ from the frame for `model_one` previously?

b) Write the implied assignment statement to show how the parameter `word` gets its value.

c) When a variable is used as an argument, does the name of the variable need to be the same as the parameter variable name?

10. Assume that `s1 = "Hi"` and `s2 = "ya"`. In the function call `model_two(s1 + s2)`:

a) What is the argument for the function call?

b) Write the implied assignment statement that happens during the call.

c) What will be the value of parameter `word` when `model_two` begins executing?

d) Predict the output that will be produced by the function call.

11. Review the two implied assignment statements that you have written. What exactly gets "passed" when you call a function?

12. Change `model_two` so that, instead of multiplying `word` by the length of `word`, it will multiply by an integer passed as the second argument to the function. Write the new version of `model_two` in the space below. Use `times` for the name of the new integer parameter.

13. How does the call to `model_two` in `main` need to change so that it matches the new function definition? Give an example.

Unit 3 Returning Values

Functions may optionally send a value back to the calling function using a `return` statement. Change the program in Python Tutor as follows:

```
1  def model_three(word):
2      ans = word * len(word)
3      return ans
4
5  def main():
6      print("Starting main...")
7      w = input("Enter a word: ")
8      result = model_three(w)
9      print(result)
10     print("All done!")
11
12 main()
```

Questions

14. Step through the execution of this code. In general, what value will be returned by `model_three`?

15. Edit `model_three` and delete the return statement at the end of the function. Visualize the execution. What value is returned by a function when there is no return statement?

16. Edit `model_three` again, and add the return statement back to the end of the function. Then change line 8 so that `model_three` is still called but there is no assignment to `result`. What do you predict will happen in `main` after the `model_three` function call completes?

17. Why is a function that returns the value of a variable more useful than a function that simply prints the value of that variable?