Defining Functions

Python programs typically have one or more functions, each of which has one or more statements. Defining new functions allows you to break down a complex program into smaller blocks of reusable code.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain the flow of execution from one function to another.
- Describe the syntax of function definitions and function calls.
- Write short functions that have parameters and return results.

Process Skill Goals

During the activity, students should make progress toward:

- Tracing the execution of functions with Python Tutor. (Information Processing)

Facilitation Notes

This activity uses Python Tutor (by Philip Guo) to visualize the flow of execution. Each model is interactive—to encourage discussion, have only one student per team run the visualization. It's particularly important in this activity that managers keep track of time. You can save time by providing students with the source code in advance.

Check in with each team during Model 1, and make sure they can run the visualization. Double check their answers for the first several questions. Have the first team that finishes draw their answer to #12 on the board. Invite another team to draw their answer to #8 on the board.

At the end of Model 2, invite teams to explain the difference between parameters and arguments. Discuss how arguments are passed to their corresponding parameters by assignment. If time permits, demonstrate the solution for the last two questions in Python Tutor.

On Model 3, remind students to work quickly. During report out, invite teams to explain what happened in #25. Emphasize that a value is always returned when a function call completes. Explain that `None` is a built-in constant like `False` and `True`.

Model 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 (20 min) Start time: _____

1. Based on the program in Model 1:

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

   b) On what line is the `model_one` function defined? _____ 1 called? _____ 9

   c) On what line is the `main` function defined? _____ 7 called? _____ 12

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? The next line to execute.

   b) What does the **green** arrow indicate? The line that just executed.

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? Line 7

   b) After line 9 of the program executes (Step 6), what is the next line that executes? Line 1

5. Go back to the beginning of the program execution. This time as you step through the program, pay attention to what changes on the **right side** of the visualization.

a) Describe what changes in the visualization after Step 1.

Under Frames, a box labeled "Global Frame" appears. Inside the box, it lists `model_one` with an arrow pointing to `function model_one()` under Objects.

b) Describe what changes in the visualization after Step 2.

The name `main` is added to the "Global Frame" box, with an arrow pointing to `function main()` under Objects.

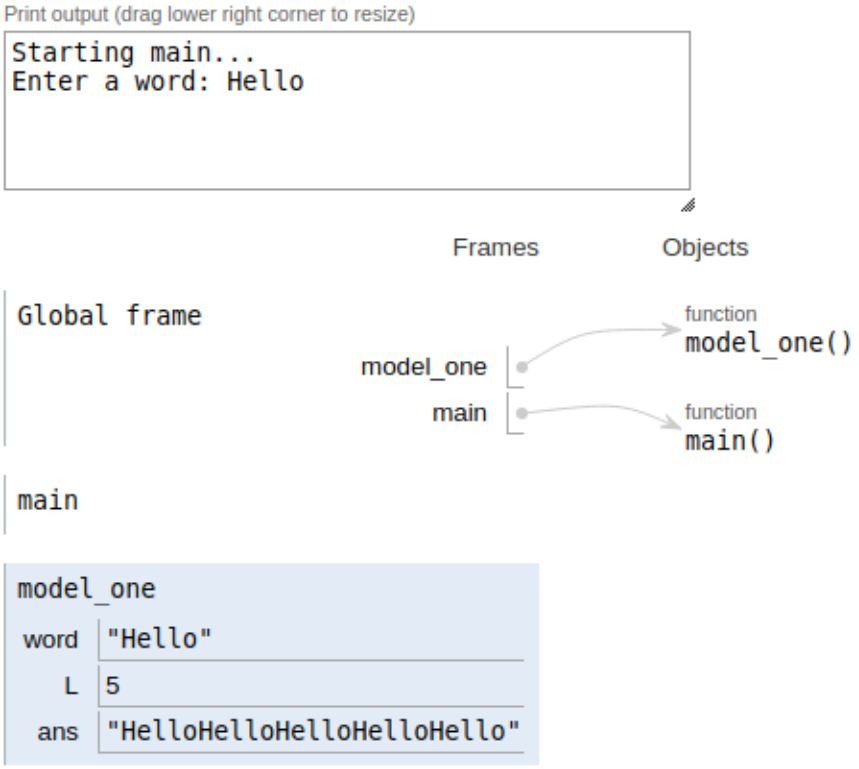
6. In general, what happens on the right side of the visualization when a function is called?

A box with the function name appears under Frames.

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

The code for the function will begin executing.

8. Draw the right side of the visualization for Step 11 in the space below.



9. Notice that the variable `ans` is printed from within the `model_one` function. What happens if you try to `print(ans)` inside the main function?

NameError: name 'ans' is not defined

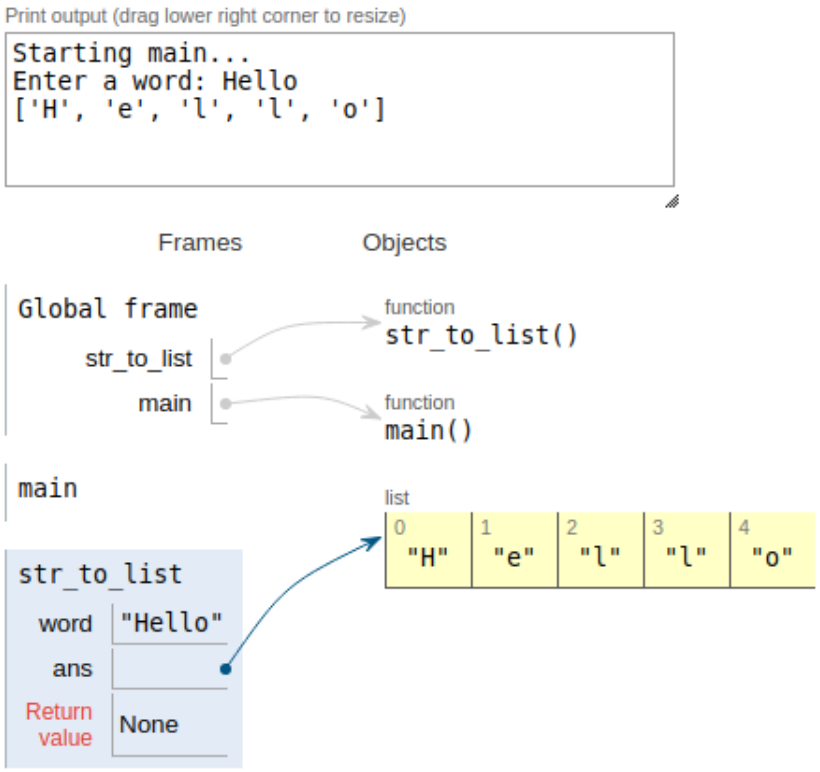
10. Explain what happened in the previous question in terms of frames in the visualization.

The variable `ans` never shows up in the frame for `main`, because it is local to `model_one`. When `model_one` finishes executing, everything associated with its frame disappears.

11. In the space below, write a definition for a function called `str_to_list` that prompts the user to enter a word. The function should convert the string to a list and print the list.

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

12. 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.



Model 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 (15 min) Start time: _____

13. Underline the parameter in the `model_two` function definition, then circle each use of the parameter inside the function. two uses of the parameter `word` should be circled in line 2

14. Find the `model_two` function call in `main`, and underline the argument being passed by the function call. word should be underlined on line 1, and w should be underlined on line 8

15. 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?

    The parameter `word` is already showing with a value in the frame for `model_two`.

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

    `word = w`

    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?

    No

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

a) What is the argument for the function call? s1 + s2 (the expression)

b) Write the implied assignment statement that happens during the call. word = s1 + s2

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

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

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

The value of the argument is passed; it gets assigned to the parameter. Note that the argument itself is not passed.

18. 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.

```
def model_two(word, times):
    ans = word * times
    print(ans)
```

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

A second argument (of type `int`) must be added to the function call: `model_two("Hi", 2)`

Model 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 (10 min) Start time: _____

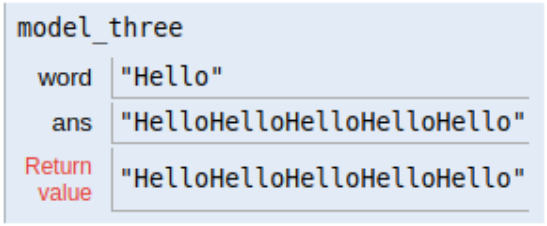
20. Aside from the function name, how does line 8 in Model 3 differ from line 8 in Model 2?

It is now an assignment statement.

21. At what step number (in the simulation) has `model_three` completed its execution, but control has not yet returned to the `main` function?

Step 11.

In the space below, draw the frame for `model_three` after this step.



22. In general, what value will be returned by `model_three`?

Whatever is the value of the variable ans.

23. What changes in the frame for `main` at Step 12 of the execution?

The variable `result` has been added to the frame. It has the value returned by `model_three`.

24. 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?

The value `None`.

25. 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?

Because `result` is not being assigned the value returned by the `model_three` function call, trying to print the value of `result` will cause a `NameError`.

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

Printing a value to the screen only benefits the program in that moment. If you instead return the value and assign it to a variable, you can continue using it in the calling function.