Carleton University
Department of Systems and Computer Engineering
SYSC 1005 - Introduction to Software Development - Fall 2017

Lab 3 - Understanding Function Definitions and Function Execution

Objectives

- To gain experience developing Python functions, using the online Python Tutor to visualize the execution of the code, and using the shell to interactively test functions that are stored in a script.

Demo/Grading

When you have finished all the exercises, call a TA, who will review your solutions, ask you to demonstrate some of them, and assign a grade. For those who don't finish early, a TA will review the work you've completed, starting about 30 minutes before the end of the lab period. **Any unfinished exercises should be treated as "homework"; complete these on your own time, before your next lab.**

Getting Started

Step 1: Create a new folder named Lab 3 on the lab computer's hard disk (on the desktop or in your account's Documents folder) or in the M: drive on the network server.

Step 2: Launch Wing IDE 101. Check the message Python displays in the shell window, and verify that Wing is running Python version 3.6. If another version is running, ask a TA for help to reconfigure Wing to run Python 3.6.

Exercise 1 - Defining a Simple Function and Tracing its Execution

Step 1: Recall that the area of a disk with radius r is π multiplied by the square of r, where r is any positive real number. This can be represented by a mathematical function that maps each value of r to the corresponding area (which is also a real number). A common notation for representing this function is:

$$f : r \rightarrow f(r), \text{where } f(r) = \pi r^2$$

The symbol f represents the function and the symbol $f(r)$ is the value that f associates with r; in other words, $f(r)$ is the value of f at r.

To calculate the area of a disk with radius 5.0, we substitute 5.0 for r in the equation $f(r) = \pi r^2$ and evaluate the expression:

$$f(5.0) = \pi \times 5.0^2 = \pi \times 25.0 = 78.5 \text{ (approximately)}$$

Performing this calculation in Python is easy. Type the following expression at the shell prompt (>>>) (for now, we're using 3.14 as the approximate value of π):

```
>>> 3.14 * 5.0 ** 2
```

What value is displayed when this expression is evaluated?

Step 2: Here is the definition for a Python *function* that has one argument, `radius`, which is the radius of a disk. The function calculates and returns the disk's area. Type this function definition in the shell. When you type the function header (the first line), the shell recognizes that you are defining a function, and repeatedly displays the secondary shell prompt (`...`) while you type the the function body. The function body should be indented four spaces.

```
>>> def area_of_disk(radius):
...     return 3.14 * radius ** 2
...     # Press the Enter key here to indicate that you have
        # finished defining the function
>>>
```

Step 3: To calculate the area of a disk with radius 5.0, we *call* this function with 5.0 as the *argument*. Type this statement:

```
>>> area_of_disk(5.0)
```

What value is returned by the function and displayed by the shell? Is it the same as the area that was displayed in Step 1?

Step 4: To calculate the area of a disk with a different radius, we call the function with a different argument. Predict the value that `area_of_disk` should return when it calculates the area of a disk with radius 10.0. Call the function to do this. Is the result displayed by Python what you predicted?

Step 5: Function arguments are expressions. When the function is called, the expression is evaluated and that value is used as the argument. Try these function calls, and note the results:

```
>>> area_of_disk(2.0 + 3.0)
>>> r = 2.0 + 3.0
>>> area_of_disk(r)
>>> area_of_disk(r * 2)
```

Step 6: Often, we want to save the value returned by a function for use in subsequent calculations. We do this by binding the returned value to a variable. Try this:

```
>>> result = area_of_disk(5.0)
>>> result
```

What value was bound to `result`?

This experiment demonstrates that a function call represents a value; specifically, the value

returned by the function. This means that function calls can appear in expressions.

Step 7: We're now going to use the Python Tutor to help you understand what happens as the computer executes each line of the source code, step-by-step.

In the *Lab Materials* section of the course cuLearn page, look for the link Open Python Tutor in a new window. Click the link. The PyTutor editor will open in a new window. Type this *script* in the editor. The function body should be indented four spaces:

```
def area_of_disk(radius):
    return 3.14 * radius ** 2

result = area_of_disk(5.0)
```

Ensure that PyTutor is configured this way: "Write code in Python 3.6", "hide exited frames [default]", "render all objects on the heap (Python/Java)" and "draw pointers as arrows [default]". If necessary, select the correct options from the drop-down menus.

Click the Visualize Execution button. Execute the script, one statement at a time, by clicking the Forward> button. Observe the memory diagram as the script is executed, step-by-step. Make sure you can answer these questions:

- What is the name of the frame containing variable `area_of_disk`?
- What does variable `area_of_disk` refer to?
- What is the name of the frame containing parameter `radius`?
- When does the frame appear in the memory diagram?
- What is bound to `radius`? What caused that object to be bound to `radius`?
- How does PyTutor depict the value the function will return?
- When does the frame containing `radius` disappear?
- What is the name of the frame containing variable `result`?
- When does `result` appear in the frame?
- What is bound to `result`? What caused that object to be bound to `result`?

Exercise 2 - Composing Functions

Step 1: A ring is a disk with a hole in the center. To calculate the area of a ring, we simply subtract the area of the hole from the area of the disk.

Click the Edit code link in PyTutor and edit the script so that it looks like this (changes are highlighted in **boldface**):

```
def area_of_ring(outer, inner):
    return 3.14 * outer ** 2 - 3.14 * inner ** 2

def area_of_disk(radius):
    return 3.14 * radius ** 2

result = area_of_ring(10.0, 5.0)
```

`area_of_ring` is a function that calculates and returns the area of a ring. This function has two parameters. Parameter `outer` is the radius of the ring and parameter `inner` is the radius of the hole.

Don't delete the definition of `area_of_disk` - you'll need it in the next step!

Click Visualize Execution and observe what happens when the script is executed step-by-step.

Step 2: The body of `area_of_ring` calculates the areas of two disks "from scratch". We can simplify `area_of_ring` by having it call `area_of_disk` to perform those calculations (this is known as *function composition*).

Here is the revised definition of `area_of_ring`. Notice how the two parameters of `area_of_ring` (`outer` and `inner`) are used as the arguments of the first and second calls to `area_of_disk`, respectively.

```
def area_of_ring(outer, inner):
    return area_of_disk(outer) - area_of_disk(inner)

def area_of_disk(radius):
     return 3.14 * radius ** 2

result = area_of_ring(10.0, 5.0)
```

Use the PyTutor editor to make these changes (highlighted in **boldface**) to the function definition. Click Visualize Execution, and observe what happens when the script is executed step-by-step. Make sure you can answer these questions:

- What is the name of the frame containing parameters `inner` and `outer`?
- When does the frame appear in the memory diagram?

- What is bound to `inner` and `outer`? What caused those objects to be bound to `inner` and `outer`?
- What is the name of the frame containing parameter `radius`?
- When does that frame appear in the memory diagram? How many times does it appear?
- Each time the frame containing `radius` appears, what is bound to `radius`? What caused that object to be bound to `radius`?
- When does the frame containing `radius` disappear?
- When does the frame containing `inner` and `outer` disappear?
- What is the name of the frame containing variable `result`?
- When does `result` appear in the frame?
- What is bound to `result`? What caused that object to be bound to `result`?

Exercise 3 - Defining Functions in a Module

While the Python Tutor is a great tool for visualizing the execution of short pieces of code, it is not a complete program development environment. On the other hand, any functions we type interactively in Wing's Python shell (as you did in Exercise 1) are lost when we quit the IDE or restart the Python shell.

If we're writing more than a few lines of code, we typically prepare it using the IDE's editor and save it in a *source code file*.

A file containing a collection of Python function definitions is known as a *module*. You're now going to develop a module named lab3.py.

Step 1: Click the New button in the menu bar. A new editor window, labelled untitled-1.py, will appear.

Step 2: From the menu bar, select File > Save As... A "Save Document As" dialogue box will appear. Navigate to your Lab 3 folder, then type lab3.py in the File name: box. Click the Save button. Wing will create a new module named lab3.py.

Step 3: In the following definitions of `area_of_disk` and `area_of_ring`, the triple-quoted string immediately after the function header is a *documentation string (docstring)*. This string contains the function's *type contract*, a brief description of what the function does, and an example of how to call the function.

Type these definitions in the editor window, exactly as they're shown here. The docstrings and the function bodies should be indented four spaces.

```
def area_of_disk(radius):
    """ (number) -> float

    Return the area of a ring with the specified
    non-negative radius.

    >>> area_of_disk(2.0)
    12.57
    """
    return 3.14 * radius ** 2

def area_of_ring(outer, inner):
    """ (number, number) -> float

    Return the area of a ring with radius outer.
    The radius of the hole is inner.

    This function requires outer > inner >= 0

    >>> area_of_ring(4.0, 2.0)
    37.7
    """
    return area_of_disk(outer) - area_of_disk(inner)
```

Step 4: Click the Save button to save the edited module.

Step 5: Click the Run button. This will load lab3.py into the Python interpreter and check it for syntax errors. If any errors are reported, edit the function to correct them, then click Save and Run.

Step 6: You can now call the functions from the shell. Try these experiments:

```
>>> area_of_disk(5.0)
>>> area_of_ring(10.0, 5.0)
```

In these experiments, the function arguments are values of type `float`. Can we instead call the functions with arguments of type `int`? Try these experiments:

```
>>> area_of_disk(5)
>>> area_of_ring(10, 5)
```

What do you conclude from these experiments?

Exercise 4 - Importing Modules

Python is shipped with several modules (hence the phrase "batteries included" is often used when describing the language). The `math` module defines several common mathematical functions and two frequently-used values (`e` and `pi`). We can modify `area_of_disk` to use variable `pi` as a better approximation of the mathematical constant π. Edit your `area_of_disk` function from Exercise 3 to use `pi` (the required changes are highlighted in **boldface**, below). Don't modify your `area_of_ring` function.

Note 1: don't copy/paste the definition of `area_of_disk` and edit the copy. When you've finished this exercise, you should have only one `area_of_disk` function in your module.

Note 2: the "official" Python style guide states that all `import` statements in a module (e.g., `import math`) should be at the top of the file, before any function definitions, as shown below. (The style guide can be found here: https://www.python.org/dev/peps/pep-0008/.)

```
import math

def area_of_disk(radius):
    """ (number) -> float

    Return the area of a ring with the specified
    non-negative radius.

    >>> area_of_disk(2.0)
    12.57
    """
    return math.pi * radius ** 2
```

The statement:

```
import math
```

makes the definitions in the `math` module available throughout the module where `area_of_disk` is defined. To use variable `pi` in the calculation of the area, we must specify both the module name and the variable name; i.e., `math.pi`.

After you've made these changes to lab3.py, use the shell to call `area_of_disk` and `area_of_ring` (repeat the experiments from Step 6 in Exercise 3).

How have the values returned by these functions changed, now that you've replaced the literal value `3.14` with `math.pi`?

Exercise 5

The lateral surface area, *sa*, of a right-circular cone (a right cone with a base that is a circle) is given by the formula:

$$sa = \pi r \times \sqrt{r^2 + h^2}$$

where *r* is the radius of the circular base and *h* is the height of the cone.

Use a calculator to determine the area of a right circular cone for specific values of *r* and *h*.

Type following code in lab3.py. Add at least one example function call to the docstring (use the numbers from the calculations you just performed).

```
def area_of_cone(height, radius):

    """ (number, number) -> float

    Return the lateral surface area of a right circular
    cone with the specified non-negative height and radius.

    Add one or more examples here.
     """
```

Complete the function definition by coding the function body. **Your function body must consist of one statement: `return` followed by an expression.** Do not create any local variables. For the value of π, use variable `pi` from the `math` module. Python's `math` module also has a function that calculates square roots. To find out about this function, use Python's help facility. In the shell, type:

```
>>> help(math.sqrt)
```

Use the shell to interactively test your function, using the examples in the docstring. Compare the expected results in the examples to the values Python displays when the function is called.

Exercise 6

The volume *v* of a sphere with radius *r* is given by the formula:

$$v = \frac{4}{3}\pi r^3$$

Use a calculator to determine the area of a right circular cone for specific values of *r*.

In lab3.py, define the header and docstring for a function named `volume_of_sphere` that calculates and returns the volume of a sphere. The function has one parameter, named `radius`, which is the radius of the sphere.

The docstring must contain a type contract, a brief description of what the function does and at

least one example function call (use the numbers from the calculations you just performed).

After you've finished the function header and docstring, complete the function definition by coding the function body. **Your function body must consist of one statement: `return` followed by an expression.** Do not create any local variables.

Use the shell to interactively test your function, using the examples in the docstring. Compare the expected results in the examples to the values Python displays when the function is called.

Exercise 7

Assume that we have two spheres, one inside the other. In lab3.py, define a function named `hollow_sphere` that calculates and returns the volume of the larger sphere that is not occupied by the smaller sphere. This function has two parameters: the radius of the larger sphere and the radius of the smaller sphere.

Start by defining the function header (use descriptive names for the parameters) and the docstring. Then complete the function definition by coding the the function body. **Your function body must consist of one statement: `return` followed by an expression.** Do not create any local variables. Your function must call the `volume_of_sphere` function you wrote for Exercise 6.

Use the shell to interactively test your function, using the examples in the docstring. Compare the expected results in the examples to the values Python displays when the function is called.

Wrap-up

1. Remember to have a TA review your solutions to Exercises 5-7, assign a grade (Satisfactory, Marginal or Unsatisfactory) and have you initial the demo/grading sheet.

2. Remember to backup your project folder before you leave the lab; for example, copy it to a flash drive and/or a cloud-based file storage service.

Extra Practice (do these on your own time, after you've completed the steps in the *Wrap Up* section on the next page)

- Copy/paste your `area_of_cone` function to PyTutor and add a statement that calls the function. Use PyTutor to visualize the execution of this script.

- Copy/paste your `volume_of_sphere` function to PyTutor and add a statement that calls the function. Use PyTutor to visualize the execution of this script.

- Copy/paste your `volume_of_sphere` and `hollow_sphere` functions to PyTutor and add a statement that calls `hollow_sphere`. Use PyTutor to visualize the execution of this script.