Carleton University
Department of Systems and Computer Engineering
SYSC 1005 - Introduction to Software Development - Fall 2018

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;
- using Wing 101 to edit and interactively test functions.

Demo/Grading

After you finish all the exercises, a TA 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 grade 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.**

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:

$$f(5.0) = \pi \times 5.0^2 = \pi \times 25.0 = 78.5398 \text{ (approximately)}$$

We can easily implement this function in Python[1]:

```
import math

def f(r):
    return math.pi * r ** 2
```

[1]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/.)

This function is named `f` and has one parameter, `r`, which is the radius of a disk. The function calculates and returns the disk's area.

Notice that we import variable `pi` from Python's `math` module. The value bound to this variable is an approximation of the mathematical constant π.

To calculate the area of a disk with radius 5.0, we *call* the function with 5.0 as the *argument*:

```
f_at_r = f(5.0)
```

It would be easier to understand this code if we rewrote it, using descriptive names for the identifiers; for example:

```
import math

def area_of_disk(radius):
    return math.pi * radius ** 2

area = area_of_disk(5.0)
```

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

Find the link Open Python Tutor in a new window in the *Labs* section of the course cuLearn page. Launch Python Tutor, and verify that it is configured this way: "Write code in Python 3.6", "hide exited frames", "render all objects on the heap (Python/Java)" and "draw pointers as arrows". If necessary, select the correct options from the drop-down menus.

Type this *script* in PyTutor's editor window:

```
import math

def area_of_disk(radius):
    return math.pi * radius ** 2

area = area_of_disk(5.0)
```

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. Record your answers to these questions (a TA may review your answers):

- 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 the type and value of the object 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 `area`?
- When does `area` appear in the frame?
- What is the type and value of the object bound to `area`? What caused that object to be bound to `area`?

Step 2: Function arguments are expressions. When the function is called, the expression is evaluated and that value is used as the argument.

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

```
import math

def area_of_disk(radius):
    return math.pi * radius ** 2

area = area_of_disk(5.0)

area = area_of_disk(2.0 + 3.0)
r = 2.0 + 3.0
area = area_of_disk(r)
area = area_of_disk(5)
```

The first time `area_of_disk` is called, the argument is a literal `float` (`5.0`). The second time this function is called, the argument is an expression (`2.0 + 3.0`). The third time the function is called, the argument is a variable, `r`. In the fourth function call, the argument is a literal `int` (`5`).

Click Visualize Execution and observe what happens when the script is executed step-by-step. What value does `area_of_disk` return each time it is called?

Exercise 2 - Composing Functions

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

```
def area_of_ring(outer, inner):
    return math.pi * outer ** 2 - math.pi * inner ** 2
```

The body of `area_of_ring` calculates the areas of the disk and the hole "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)
```

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

```
import math

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

def area_of_disk(radius):
    return math.pi * radius ** 2

area = area_of_ring(10.0, 5.0)
```

Click Visualize Execution and observe what happens when the script is executed step-by-step. Record your answers to these questions (a TA may review your answers):

- What is the name of the frame containing parameters `inner` and `outer`?
- When does the frame appear in the memory diagram?
- What are the types and values of the objects 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 the type and value of the object

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 `area`?
- When does `area` appear in the frame?
- What is the type and value of the object bound to `area`? What caused that object to be bound to `area`?

Exercise 3 - Defining Functions in a Module

PyTutor is a great tool for visualizing the execution of short pieces of code, but it is not a complete program development environment, and it doesn't provide a way for us to save our code in a file. 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: 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 101. When Python starts, it prints a message in the shell window. Check the message and verify that Wing is running Python version 3.7. If another version is running, ask a TA for help to reconfigure Wing to run Python 3.7.

Step 3: Click the New button in the menu bar. A new editor window, labelled untitled-1.py, will appear.

Step 4: 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 5: Type the definitions of `area_of_disk` and `area_of_ring` in the editor window, exactly as they're shown on the following page. 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 use the shell to test the function.

```
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

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)
```

The type contract for `area_of_disk` is: `(number) -> float`. `(number)` specifies that this function should be called with exactly one argument, and this argument should be an `int` or a `float`; that is, a number. The arrow followed by a type, `-> float`, specifies that the function returns a `float`.

The type contract for `area_of_ring` is: `(number, number) -> float`. This contract specifies that function should be called with exactly two arguments, and each argument should be an `int` or a `float`. It also specifies that the function returns a `float`.

Step 6: Click the Save button to save the edited module.

Step 7: 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 8: You can now call the functions from the shell. Try these experiments:

```
>>> area_of_disk(5.0)
>>> area_of_ring(10.0, 5.0)
>>> area_of_disk(5)
```

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

What happens if you call a function with an argument that violates the function's type contract? Try this experiment, which calls `area_of_disk` with an argument of type `str`:

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

Exercise 4 - The Point of No `return`

For this exercise, use Python Tutor, not Wing 101.

Some programming languages don't use a `return` statement. Instead, the value returned by a function is the value of the last expression that is evaluated when the function executes. Is this true of Python?

Type this script in PyTutor's editor window:

```
import math

def area_of_disk(radius):
    math.pi * radius ** 2

area = area_of_disk(5.0)
```

Click Visualize Execution and observe what happens when the script is executed step-by-step. Does `area_of_disk` automatically return the value of the expression

```
math.pi * radius ** 2
```

even though it isn't preceded by the keyword, `return`? If not, what is the type and value of the object returned by the function?

Exercise 5 is on the next page.

Exercise 5 - Assignment Statements, Function Parameters and Function Arguments

For this exercise, use Python Tutor, not Wing 101.

Consider this script:

```
def cube(x):
    x = x ** 3

a = 2
cube(a)
```

When `cube` is called, parameter `x` will be bound to `2`. We want to know what will happen when `cube` executes:

```
x = x ** 3
```

Predict the effect that this assignment statement will have on `x`. Predict the effect that this statement will have on the corresponding function argument, `a`.

Type this script in PyTutor's editor window. Click Visualize Execution and observe what happens when the script is executed step-by-step. Were your predictions correct?

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*.

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 volume_of_sphere(radius):
    """ (number) -> float

    Return the volume of a sphere with the specified
    non-negative radius.

    Put one or more function call 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.

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

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.

    Put one or more function call 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.

Wrap-up

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