Carleton University
Department of Systems and Computer Engineering
ECOR 1041 - Computation and Programming

Lab 3 - Understanding Function Definitions and Function Execution

Objectives

To gain experience using **Python Tutor** to understand what happens when Python:

- executes a function definition;
- executes a function call (a *call expression*).

Getting Started

You will work in the same way as previously. You will perform a series of exercises in Python Tutor, writing your name, student number and answers in a simple text file. The template for this file is provided on Brightspace as **`lab3.txt`**.

There is one difference: The descriptions of the exercises will be provided in this document (not in the text file) because the problems are getting more complex, and we need stronger formatting for their equations.

Exercise 1 - Defining a Simple Function and Tracing its Execution

Step 1: Recall the formula for the area of a disk with radius *r*:

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

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 that has a radius of 7.0, we substitute 7.0 for *r* in the equation $f(r) = \pi r^2$ and evaluate the expression:

$$f(7.0) = \pi \times 7.0^2 = \pi \times 49.0 = 153.9380 \text{ (approximately)}$$

We can easily *implement* this function in Python (alternatively, called "write the function *definition*"):

```
import math
def f(r):
    return math.pi * r ** 2
f_at_7 = f(7.0)
```

The function *header*, `def f(r):`, specifies that the function is named `f` and has one parameter, `r`, which is the radius of a disk. The function *body*, `return math.pi * r ** 2`, 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 that has a radius of 7.0, we *call* the function with 7.0 as the *argument*:

```
f_at_7 = f(7.0)
```

This code would be easier to understand if we used *descriptive names* for the identifiers; for example:

```
import math
def area_of_disk(radius):
    return math.pi * radius ** 2
area = area_of_disk(7.0)
```

You are 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.

Type the above script into Python Tutor's editor window. Click the Visualize Execution button. Execute the script, one statement at a time, by clicking the Forward > button. **Observe the frames pane** as the script is executed step-by-step. **Answer all the questions pertaining to Exercise 1 in lab3.txt.**

Exercise 2 - Learning More about Function Arguments

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

Make the following changes to the code in Python Tutor (changes in bold-red)

```
import math

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

area = area_of_disk(7.0)      # Literal float

area = area_of_disk(4.0 + 3.0)  # Expression
r = 4.0 + 3.0
area = area_of_disk(r)  # Variable
area = area_of_disk(7)  # Literal integer
```

The first-time `area_of_disk` is called, the argument is a literal `float` (`7.0`). The second time this function is called, the argument is an expression (`4.0 + 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` (7).

Click Visualize Execution and observe the frames as the script is executed step-by-step. **Answer all the questions in lab3.txt** pertaining to Exercise 2.

Exercise 3 - Composing Functions

A ring is a disk with a hole in the center (i.e., a doughnut). The function `area_of_ring` 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 the ring, the function first calculates the areas of the disk and the hole, then calculates the difference of the two areas:

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

We can simplify the `area_of_ring` function by having it call our `area_of_disk` function 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)
```

Write the following script in Python Tutor.

```
import math

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

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

area = area_of_ring(14.0, 7.0)
```

Click Visualize Execution and observe the frames pane and the objects pane as the script is executed step-by-step. **Answer all the questions in lab3.txt** pertaining to Exercise 3.

Exercise 4 - The Point of "No `return`"

All our functions so far have ended with a `return` statement. What happens if it is missing?

Edit the script in Python Tutor so that it looks like this (delete the definition of `area_of_ring` and the statement that calls that function):

```
import math

def area_of_disk(radius):
    math.pi * radius ** 2  # No return!

area = area_of_disk(7.0)
```

Click Visualize Execution and observe the frames as the script is executed step-by-step. **Answer all the questions in lab3.txt** pertaining to Exercise 4.

Exercise 5 - Assigning Values to Function Parameters

Some programming languages allow a function to "return" values through the function's parameters. If a function assigns a new value to a parameter, that value is copied to the corresponding argument, which must be a **variable**. One example of a modern language that supports this is C++, which provides *reference parameters*.

Let's run an experiment to determine if Python also provides this feature.

Delete all the code in Python Tutor's editor window and type this script:

```
def square(x):
    x = x ** 2

a = 4
square(a)
```

The programmer is hoping that the function's argument, `a`, will be assigned the value of x^2 when the value of `x ** 2` is assigned to parameter `x`. In other words, the programmer is hoping that variable `a` will be bound to `16` (the calculated value of 4^2) when Python executes the assignment statement in the body of `square`.

Click Visualize Execution and observe the frames as the script is executed step-by-step.
Answer all the questions in lab3.txt pertaining to Exercise 5.

Wrap Up

- Make sure you wrote your name and student number were indicated.
- Submit lab3.txt on Brightspace.

You are required to keep a backup copy of (all) your work for the duration of the term.

Edited: 23rd Dec, 2021