Carleton University
Department of Systems and Computer Engineering
SYSC 1005 - Introduction to Software Development - Fall 2014

Lab 3 - Understanding Function Definitions and Function Execution

Objectives

- To gain additional experience developing and interactively testing Python functions, and using the Online Python Tutor to visualize the execution of the code.

Attendance/Demo

To receive credit for this lab, you must make reasonable progress towards completing the exercises and demonstrate the code you complete. **Also, you must submit your lab work to cuLearn by the end of the lab period**. (Instructions are provided in the *Wrap Up* section at the end of this handout.)

When you have finished all the exercises, call a TA, who will review the code you wrote. For those who don't finish early, a TA will ask you to demonstrate whatever code 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`.

Step 2: Launch Wing IDE 101. Check the message Python displays in the shell window, and verify that Wing is running Python version 3.4. If another version is running, ask your instructor or a TA for help to reconfigure Wing to run Python 3.4.

Exercise 1 - `return` versus `print()`

Novices are sometimes not sure of the difference between a function that returns a value and a function that prints a value. You'll investigate this in this exercise.

Step 1: Here are two function definitions. Type both function definitions in the shell.

```
>>> def double_and_return(x):
...     return 2 * x
... Type the Enter key here

>>> def double_and_print(x):
...     print(2 * x)
... Type the Enter key here
```

Step 2: Call both functions:

```
>>> double_and_return(5)
What does Python display?

>>> double_and_print(5)
What does Python display?
```

After each function is called, Python displays a value in the shell window. Based only on these values, do the functions appear to do the same thing or do they do something different?

Step 3: We'll repeat the function calls, but this time, we'll bind the values returned by the functions to variables:

```
>>> first_result = double_and_return(5)
What does Python display?

>>> first_result
What does Python display?

>>> print(first_result)
What does Python display?

>>> second_result = double_and_print(5)
What does Python display?

>>> second_result
What does Python display?

>>> print(second_result)
What does Python display?
```

Based on your observations:

- What value is returned by a function that does not contain a `return` statement?
- What is the difference between a function *returning* a value it calculates and *printing* a value it calculates?

You're now going to define some functions in 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 the folder where you want to store the module, type lab3.py in the File name box, then click the Save button.

Exercise 2 - A function that returns a tuple

A quadratic equation has the form:

$$ax^2 + bx + c = 0, \text{ with } a \neq 0 \ .$$

The roots of a quadratic equation are given by the formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

In lab3.py, define a function named `quadratic_roots` that is passed the real-number coefficients *a*, *b* and *c* of a quadratic equation. The function should return a tuple containing both roots of the equation. If the two roots are the same, the tuple will have contain two identical values.

The function does not have to handle quadratic equations with complex roots; i.e., the function should assume that the term:

$$\sqrt{b^2 - 4ac}$$

will never be negative. (The function does not have to check for this case.)

Use the shell to interactively test your function. At a minimum, verify that the function correctly calculates the roots of these two equations:

- $x^2 - 4x - 21 = 0$ (the roots are 7.0 and -3.0)
- $4x^2 + 12x + 9 = 0$ (both roots are -1.5)

Exercise 3

For many students who state that "programming is hard", the problem often has little to do with writing the code for solutions to problems. Instead, the student frequently has difficulties developing the mathematical formulas that correctly model the solutions. This exercise (as well as Exercise 4) is an example of problem that is easily coded, once you've figured out the correct formulas.

In lab3.py, define a function named `make_change` that calculates how much change a cashier should give a customer. The function has two parameters: `amount_due` (the cost of a customer's purchases) and `amount_received` (the amount of money provided by the customer).

The function should return a tuple containing the dollars, quarters, dimes, nickels and pennies that the customer should receive as change. (For those students who are not familiar with Canadian coins, a dollar is worth 100 cents, a quarter is worth 25 cents, a dime is worth 10 cents, a nickel is worth 5 cents, and a penny is worth 1 cent.)

Ignore the fact that, in Canada, stores now round the cost of purchases up or down to the nearest nickel. In other words, assume that if the cost of your purchase is $17.33, that's exactly the amount that you have to pay (not $17.35).

The function should calculate the minimum number of coins that provide the correct amount of change; for example, if the customer should receive 80 cents in change, the function should calculate 3 quarters and 1 nickel, not 3 dimes, 8 nickels and 10 pennies.

To avoid roundoff errors, each function parameter should specify an amount in pennies (integers); for example, 3 dollars and 78 cents should be specified as 378, not 3.78.

Do not use loops or `if` statements in this function (we haven't covered those constructs in class). Hint: which Python operators perform integer division and calculate remainders? If you're not sure, check Lab 1.

Use the shell to interactively test your function for several pairs of input values.

Exercise 4

In lab3.py, define a function named `area_of_pipe` that returns the *total* surface area of a pipe, which is an open cylinder. The function has three parameters: the pipe's inner radius, its length and the thickness of its wall. Your function must call the `area_of_disk` and `area_of_ring` functions from Lab 2 (copy the definitions of the final versions of those functions from lab2.py to lab3.py). You must also define a function named `circumference`, which is passed the radius of a circle and returns its circumference. Your `area_of_pipe` function must call `circumference`.

Use the shell to interactively test your function for several pairs of input values. At a minimum, verify that when called this way:

```
>>> area_of_pipe(2, 3, 4)
```

the value returned by your function is (approximately) `351.9`.

Exercise 5

In the Labs section of the cuLearn course page, click the link Online Python Tutor - Lab 3. Copy your `quadratic_roots` function from Exercise 2 to the OPT editor. Remember, OPT does not have a shell window, so you'll have to add calls to your `quadratic_roots` function to the script. Trace the execution of your script, step-by-step. Make sure you understand the diagrams OPT displays questions dealing with visualization of code execution will be on the exams.

Repeat this exercise for your `make_change` and `area_of_pipe` functions.

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

Imagine the owner of a movie theater who has complete freedom in setting ticket prices. The more she charges, the fewer the people who can afford tickets. In a recent experiment the owner determined a precise relationship between the price of a ticket and average attendance. At a price of $5.00 per ticket, 120 people attend a performance. Decreasing the price by a dime ($.10) increases attendance by 15. Unfortunately, the increased attendance also comes at an increased cost. Every performance costs the owner $180. Each attendee costs another four cents ($0.04). The owner would like to know the exact relationship between profit and ticket price so that he can determine the price at which he can make the highest profit.

While the task is clear, how to go about it is not. All we can say at this point is that several quantities depend on each other.

When we are confronted with such a situation, it is best to tease out the various dependencies one at a time:

1. *Profit* is the difference between revenue and costs.
2. The *revenue* is exclusively generated by the sale of tickets. It is the product of ticket price and number of attendees.
3. The *costs* consist of two parts: a fixed part ($180) and a variable part that depends on the number of attendees.
4. Finally, the problem statement also specifies how the number of attendees depends on the ticket price.

You're now going to define four functions. Some of the functions will call one or more of the other functions.

1. Write a function named `attendees` that is passed the price of a ticket. The function returns

the number of people who will attend a performance at the specified ticket price. Test your function by calling it with ticket prices of $3.00, $4.00 and $5.00. Does your function return the correct attendance for each ticket price? (You will have to figure out the results that a correctly-designed function should return, yourself.)

2. Write a function named `cost` that is passed the price of a ticket. The function returns the cost of a performance, given the specified ticket price. Test your function by calling it with ticket prices of $3.00, $4.00 and $5.00. Does your function return the correct performance cost for each ticket price?

3. Write a function named `revenue` that is passed the price of a ticket. The function returns the total revenue from a performance, given the specified ticket price. Test your function by calling it with ticket prices of $3.00, $4.00 and $5.00. Does your function return correct revenue for each ticket price?

4. Write a function named `profit` that is passed the price of a ticket. The function returns the total profit from a performance, given the specified ticket price. Test your function by calling it with ticket prices of $3.00, $4.00 and $5.00. Does your function return correct profit for each ticket price?

Call your `profit` function several times, using a different ticket price each time. What price should the owner charge for tickets, in order to maximize the profit?

Wrap-up

1. Remember to have a TA review your solutions to the exercises, assign a grade (Satisfactory, Marginal or Unsatisfactory) and have you initial the demo/sign-out sheet.

2. Before you leave the lab, log in to cuLearn and submit lab3.py. To do this:

    2.1. Click the Submit Lab 3 link. A page containing instructions and your submission status will be displayed. After you've read the instructions, click the Add submission button. A page containing a File submissions box will appear. Drag lab3.py to the File submissions box. Do not submit another type of file (e.g., a zip file, a RAR file, a .txt file, etc.)

    2.2. After the icon for the file appears in the box, click the Save changes button. At this point, the submission status of your file is "Draft (not submitted)". If you're ready to finish submitting the file, jump to Step 2.4. If you instead want to replace or delete your "draft" file submission, follow the instructions in Step 2.3.

    2.3. You can replace or delete the file by clicking the Edit my submission button. The page containing the File submissions box will appear.

        2.3.1. To overwrite a file you previously submitted with a file having the same name, drag another copy of the file to the File submissions box, then click the Overwrite button when you are told the file exists ("There is already a file called..."). After the icon for the file reappears in the box, click the Save changes button.

        2.3.2. To delete a file you previously submitted, click its icon. A dialogue box will appear. Click the Delete button., then click the OK button when you are asked, "Are you sure you want to delete this file?" After the icon for the file disappears, click the Save changes button.

    2.4. Once you're sure that you don't want to make any changes, click the Submit assignment button. A Submit assignment page will be displayed containing the message, "Are you sure you want to submit your work for grading? You will not be able to make any more changes." Click the Continue button to confirm that you are ready to submit your lab work. This will change the submission status to "Submitted for grading". **Make sure you do this before you leave the lab.**

Acknowledgments: Exercise 3 was based on a programming exercise by Cay Horstmann and Rance Necaise. Exercise 4 and the Extra Practice exercise were based on exercises by Matthias Felleisen, Robert Bruce Findler, Matthew Flatt and Shriram Krishnamurthi.