Carleton University
Department of Systems and Computer Engineering
SYSC 1005 - Introduction to Software Development - Fall 2013

Lab 3 - Python Functions

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 these functions.

Attendance/Demo

To receive credit for this lab, you must demonstrate your work. When you have finished all the exercises, call your instructor or a TA, who will review the code you wrote. For those who don't finish early, the TAs will ask you to demonstrate whatever code you've completed, starting at about 30 minutes before the end of the lab period.

Getting Started

Step 1: Create a new folder named `Lab 3` and launch Wing IDE 101.

Exercise 1 - Python Functions (Review of Concepts from Weeks 2 and 3)

Step 2: Recall that the area of a disk with radius r is approximately $3.14 \times r^2$, where variable r represents any positive number. To calculate the area of a disk with radius 5, we substitute 5 for r in the formula, and evaluate the expression:

$$\begin{aligned} & 3.14 \times 5^2 \\ = \; & 3.14 \times 25 \\ = \; & 78.5 \end{aligned}$$

Performing this calculation in Python is easy. At the shell prompt (`>>>`) type this expression:

```
>>> 3.14 * 5 ** 2
```

What value is displayed when this expression is evaluated?

Step 3: Here is the definition for a Python *function* that is passed one argument, `r`, 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 first statement, the shell recognizes that you are defining a function, and repeatedly displays the secondary shell prompt (`...`) to prompt you to enter another line of the function body.

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

Step 4: To execute the statements in this function, we call it by name. Type this function call to calculate the area of a disk with radius 5:

```
>>> area_of_disk(5)
```

The value returned by the function (and displayed by the shell) should be the same as the result in Step 2. Is it?

Step 5: Of course, we can pass a different argument to the function when we call it; for example, 10. Try it!

```
>>> area_of_disk(10)
```

Here is a summary of the syntax rules for functions. Once again, here's the code:

```
def area_of_disk(r):
    area = 3.14 * r ** 2
    return area
```

- The first line is the function *header*, which always begins with the *keyword* `def`
- Every function has a name; e.g., `area_of_disk`
- The header must end with a colon (`:`)
- `r` is a variable (it corresponds to the variable *r* in the formula shown earlier)
- Variables listed in function headers; for example, `r`, are *parameters*
- The parameter list is enclosed in round brackets, between the function name and the `:`
- The remaining statements are the function *body*
- Statements in the body **must** be indented (the usual Python convention is 4 spaces per indentation level).
- `area` is known as a *local variable* because it's defined inside (is local to) the function
- The function binds a value (the calculated disk area) to `area`
- The `return` statement means: *stop executing the function, and return the value of the following expression*
- Here, the value returned by the function is the value bound to variable `area`

Step 6: Function arguments are expressions. When the function is called, the argument expression is evaluated, and that value is bound to the corresponding. Try these calls, and note the results:

```
>>> area_of_disk(2 + 3)

>>> radius = 2 + 3
>>> area_of_disk(radius)

>>> area_of_disk(radius * 2)
```

Step 7: What happens when you call the function this way?

```
>>> result = area_of_disk(5)
>>> result
```

The function calculates and returns the disk's area (78.5). This value is bound to `result` after `area_of_disk` returns. Notice that a function call represents a value: the value returned by the function.

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

On the cuLearn page for this course, look for the link Online Python Tutor - Lab 3. Click the link. The Online Python Tutor editor will open in a new window. Type this code in the editor:

```
def area_of_disk(r):
    area = 3.14 * r ** 2
    return area

result = area_of_disk(5)
```

Ensure that OPT is configured this way: "Python 2.7", "hide frames of exited functions", "render all objects on the heap", "hide environment parent pointers", "draw references using arrows", and "show everything". If necessary, select the correct options from the drop-down menus.

Click the Visualize Execution button. Execute the code, one statement at a time, by clicking the Forward> button. When does the frame containing parameter `r` and variable `area` appear in the diagram? When does the frame disappear? Hows does what you've observed demonstrate that a function's parameters and local variables exist only while the function is executing.

Exercise 2 - A More Concise Version of `area_of_disk`

Step 8: In a `return` statement, the `return` keyword is followed by an expression. In the `area_of_disk` function, we can change the `return` statement, replacing variable `area` with the expression that calculates the area. So,

```
def area_of_disk(r):
    area = 3.14 * r ** 2
    return area

result = area_of_disk(5)
```

becomes:

```
def area_of_disk(r):
    area = 3.14 * r ** 2
    return 3.14 * r ** 2

result = area_of_disk(5)
```

Click the Edit code link in OPT, and make this change (highlighted in **boldface**) to your function definition. Click Visualize Execution and execute your modified function step-by-step.

The result returned by the function should be the same as in Step 7. Is it?

Step 9: Notice that in the modified function, variable `area` is never used after a value is bound to it. As such, `area` is is no longer needed, so we can further simplify the function by deleting the statement: `area = 3.14 * r ** 2`:

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

result = area_of_disk(5)
```

Make this change to your function definition. Execute the modified function, step-by-step. The result returned by the function should be the same as in Step 8. Is it?

Exercise 3 - Functions Can Call Functions

Step 10: 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.

Edit the code in OPT so that it looks like this:

```
def area_of_ring(outer, inner):
    area_outer = 3.14 * outer ** 2
    area_inner = 3.14 * inner ** 2
    area = area_outer - area_inner
    return area

# code continues on the next page
```

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

result = area_of_ring(10, 5)
```

`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.

Also, notice that the call to `area_of_disk` has been replaced by a call to `area_of_ring`.

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 `area_of_ring` is executed step-by-step.

Step 11: The first two statements in `area_of_ring` calculate the areas of two disks. Why not call `area_of_disk` to do those calculations? Here is a revised function definition that does this. Notice how the two parameters of `area_of_ring` (`inner` and `outer`) are used as the arguments of the two calls to `area_of_disk`.

```
def area_of_ring(outer, inner):
    area_outer = area_of_disk(outer)
    area_inner = area_of_disk(inner)
    area = area_outer - area_inner
    return area

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

result = area_of_ring(10, 5)
```

Use the OPT editor to make these changes (highlighted in **boldface**) to your function definition. Click `Visualize Execution`, and observe what happens as the code is executed step-by-step. The result returned by the function should be the same as in Step 10. Is it?

Step 12: Now that we know the function works, we can make it more concise (just like we did in steps 8 and 9), eliminating all the local variables `area_outer`, `area_inner` and `area`. Edit the code once more, so that it looks like this:

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

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

```
result = area_of_ring(10, 5)
```

Test the function. The result returned by the function should be the same as in Step 11. Is it?

Exercise 4 - Defining Functions in a Module

While the Online Python Tutor is a great tool for visualizing the execution of short pieces of code, it is not a complete program development environment. In addition, the functions we type interactively in Wing IDE's Python shell are lost when we exit Wing 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 the source code in a 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 13: Click the New button in the menu bar. A new editor window, labelled untitled-1.py, will appear.

Step 14: 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.

Step 15: Here are the definitions of `area_of_disk` and `area_of_ring`. Type this code in the editor window, exactly as it's shown here. Once again, each statement in the function body should be indented four spaces.

```
def area_of_disk(r):
    """
    Return the area of a ring with radius r.
    """
    return 3.14 * r ** 2

def area_of_ring(outer, inner):
    """
    Return the area of a ring with radius outer.
    The radius of the hole is inner.
    """
    return area_of_disk(outer) - area_of_disk(inner)
```

We've made one addition to the functions. The triple-quoted string immediately after the function header is a *documentation string (docstring)*. This string summarizes what the function does.

Step 16: Click the Save button to save the edited module.

Step 17: 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 18: You can now call the functions from the shell. Try these experiments:

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

Exercise 5 - Importing Modules

Step 19: Python is shipped with several modules (hence the phrase "batteries included", which 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 (the changes are highlighted in **boldface**):

```
import math

def area_of_disk(r):
    """
    Return the area of a disk with radius r.
    """
    return math.pi * r ** 2
```

The statement:

```
import math
```

makes the definitions in `math` 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`.

Make these changes to the module, then test `area_of_disk`.

Exercise 6

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

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

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

Define a function named `area_of_cone` in `lab3.py` that calculates and returns the lateral surface area of a right-circular cone. Here is the function header and docstring:

```
def area_of_cone(h, r):
    """
    Returns the lateral surface area of a right circular
    cone with height h and radius r.
     """
```

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.

Exercise 7

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

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

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

Use the shell to interactively test your function.

Exercise 8

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. Your function must call the function you wrote for Exercise 7. Use the shell to interactively test your function.

Wrap-up

1. Remember to have a TA review your solutions to the exercises and give you a grade (Satisfactory, Marginal or Unsatisfactory) before you leave the lab.
2. Log in to cuLearn and click the Submit Lab 3 link. Submit lab3.py. Instructions for submitting file-based assignments in cuLearn can be found here:

   www5.carleton.ca/culearnsupport/students/

   Click the link Evaluation Tools, then click the link Assignments.

   After you upload the file, remember to click the Send for marking button (See Step 6 of the instructions).