COMP 1510 Programming Methods Lab 03

Christopher Thompson
cthompson98@bcit.ca

BCIT CST — September 2019

Welcome!

Welcome to your third COMP 1510 lab. In today's lab we will continue our exploration of the function. Remember that we use the function to implement indirection.

We like to say that functions are atomic. An atomic function cannot be broken down any further. That is, every function does only one thing. This makes functions easier to implement, test, and debug.

You will have two hours to work on the lab on Tuesday. On Thursday in lab, I will spend two minutes with each student in a lightning-round face to face marking meeting. After a short break, we'll spend some time reviewing some important topics from the lab and lectures, and finish with our weekly quiz.

Take your time. Read each step. Don't skip anything. Good luck, and have fun!

1 Grading

Figure 1: This lab is graded out of 5

This lab will be marked out of 5. For full marks this week, you must:

1. (3 points) Correctly implement the functions, etc., in this lab
2. (1 point) Correctly format and comment your code. Use the examples from the slides in lecture to as a guide and add a docstring to your program and to each function inside the program
3. (1 point) Correctly and fully test the functions using doctests inside your docstrings.

2 Project Setup

Please complete the following:

1. Create a new project in PyCharm called A########_1510_labs, where A######## is your student number. We will use this project for the rest of our labs this term.
2. Inside the new project, create five Python packages called Lab01, Lab02, Lab03, ..., Lab12, respectively.
3. Copy your code from your original Lab 1 to the Lab01 folder.
4. Copy your code from your original Lab 2 to the Lab02 folder.

5. Create a new file in Lab03 called base_conversion.py. Copy your base_conversion function from function.py in Lab02 to this new file in Lab03.
6. Add this project to git. From the main menu select VCS > Import into version control > Create git repository. Select the project folder (A########_1510_labs, not Lab01 or Lab02 or Lab03).
7. After adding the project to git, select the folders Lab01, Lab02, ..., Lab12 in the PyCharm project pane. Right-click and select Add to add them to version control.
8. Now add the project to GitHub in the cloud. Select VCS > Import into version control > Share project on GitHub. Only Ensure the project is private. I will not mark any repository which is public or which has ever been public.
9. For your third lab, all files must go into the Lab03 folder.
10. After you complete each task, commit and push your change to version control. In order to earn full marks, you must commit and push after each function is complete. Your commit message must be a short message to me that describes what you completed for this commit, i.e., "deconstructed base_conversion", or "debugged function_name", etc.
11. When you are finished, invite me as a collaborator. You will only need to do this once for this project. For future labs, since I am already a collaborator I will pull your future work automatically.

3 Requirements

Let's start by intentionally generating some errors:

1. Create a new Python file called errors.py in the Lab03 folder.
2. Add a main method to errors.py.
3. Inside the main method, write code that produces a ZeroDivisionError.
4. Write code that produces an IndexError. Produce this error in two different ways.
5. Write code that produces a TypeError. Produce this error in two different ways.

Let's continue by breaking down a large function into a group of smaller, easily tested functions:

1. Start by examining your base_conversion function that you copied to Lab03 from Lab 02.
2. Last week we talked about functional decomposition. Open the slides from last week and refer to the slides about functional decomposition and decomposition techniques. Apply what we have learned to break down your base_conversion function into a main function called base_conversion and helper functions. Each function must be short, discrete, independent, and easily described with a short description using a single verb in the docstring.
3. Create a flowchart for the algorithm we are using to convert values to a new base. If you don't like MS Visio, check out https://www.draw.io/ which is a free online flowchart maker. Indicate using colour or outlines, etc., how you have divided this algorithm into separate functions, and where the boundaries of those functions meet.

Finally, let's flex those mental muscles of yours:

1. Create a file called functions.py and inside it design a function called roll_die:
   (a) Die is the singular form of the word dice. (Suppose I have two dice in my hand. If I remove one, I now only have one die.)
   (b) This function accepts two parameters, both integers. The first is called number_of_rolls, and the second is called number_of_sides.

(c) This function must simulate rolling a die of the specified size the specified number of times. That is, if I invoke the function like this: roll_die(3, 6), then I mean to roll a six-sided die three times, for a random total that will be anywhere from 3 to 18 inclusive [3, 18].

(d) This function must return that random total. Note that both parameters must be positive integers or the function should return 0. It's impossible to roll a zero with any known die in this universe, so this is a good return value that signifies something went wrong.

(e) How will you test this function to prove it works?

2. Inside your functions.py file, design a function called create_name:

(a) This function must accept a single parameter, a positive integer called length. If this function is passed a parameter that is not a positive integer, return None.

(b) Return a string of random letters, i.e., a name. This string must have the number of letters specified by the parameter. Ensure the first letter is capitalized and the rest are in lower case.

(c) You may find the sample or choices functions in the random library helpful. They can be studied here: `https://docs.python.org/3/library/random.html`

That's it! Good luck, and have fun!