COMP 1510 Programming Methods Lab 02

Christopher Thompson
cthompson98@bcit.ca

BCIT CST — January 2022

Welcome!

Welcome to your second COMP 1510 lab. In today's lab we will begin our exploration of the Python function. In Python, the function is an important way 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.

Ordinarily, if we have to use the word AND when we describe what a function does, it is not atomic. It does more than one thing. It is too complex, and the function probably needs to be decomposed (divided) into two (or more!) separate functions.

You will have three hours to work on the lab this week. During the tutorial hour after the labs, 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, ensuring your functions are decomposed into atomic functions that each do a single thing.
2. (1 point) Correctly format your code. Use the examples from the slides in lecture as a guide
3. (1 point) Correctly submit your work for marking using GitHub.

2 Requirements

1. Visit this URL to copy the template I created to your personal GitHub account, and then clone your repository to your laptop:

   `https://classroom.github.com/a/tnxspdX_`

2. Populate the README.md with your full name and student number. Commit and push your change. I'd like you to commit and push your changes to GitHub after you complete each of the following exercises. All of your work must go into the functions.py source file.

3. Define a function called triple that accepts a single number called value and returns that number tripled. Your function is not responsible for how it acts in response to any other sort of argument. Your function must not print anything.

4. Define a function called absolute_difference that accepts two numbers and returns the absolute value of the difference between the two. Your function is not responsible for how it acts in response to any other sort of arguments. Your function must not print anything.

5. Define a function called average_of_3 that accepts three numbers and returns the average. Your function is not responsible for how it acts in response to any other sort of arguments. Your function must not print anything.

6. Define a function called average_of_top_3 that accepts four numbers and returns the average of the top three. Hint: this function probably needs to be decomposed, and it will probably use the function you just created as a helper function. Your function must not print anything.

7. Define a function called base_convert that accepts a base-10 number and a number representing the destination base, in that order, and returns the base-10 number in the destination base. This function only needs to work for destination bases [2, 10] with a maximum of 2 (yes only 2!) digits in the destination base. Your function is not responsible for how it acts in response to any other sort of arguments. If the base_convert function is decomposed into smaller functions, define them adjacent to the base_convert function (do they have to come before base_convert, or can you define them after base_convert?). Your function must not print anything.

8. Design a function called name. This function must accept a single parameter, a positive integer called length. Your function is not responsible for how it acts in response to any other sort of argument. Return (don't print) a string of random letters, i.e., a name, of the specified length. Ensure the first letter is capitalized and the rest are in lower case. 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`. Your function must not print anything.

9. The Wikipedia article tells me there are eight temperature scales (wow, I had no idea!):

   `https://en.wikipedia.org/wiki/Comparison_of_temperature_scales`

   Define a function called convert_temperature that accepts three parameters: a floating point number representing source units, a string representing the source scale, and a string representing the target scale. The strings must be one of the following: "Kelvin", "Celsius", Fahrenheit", "Rankine", "Delisle", "Newton", "Reaumur", and "Romer". Your function is not responsible for how it acts in response to any other sort of arguments. Your function must convert the value from source units to target units and return the temperature as a float. Your function must not print anything. (Hint: there's a lot here and it could be a spaghetti tangle of if-statements. Too crazy. Instead, consider choosing a canonical scale like Celsius. Your conversion function could work in two steps: convert to Celsius and then convert from Celsius to the target scale.

10. Add a main function to the bottom of your functions.py source file and prove to me that all of your functions work (including helper functions) by invoking them with meaningful arguments.

That's it! Good luck, and have fun! Remember to Commit and Push your work after each of the programming exercises. Ask me for help if you're not sure how!