COMP 1510 Programming Methods Lab 02

Christopher Thompson
`cthompson98@bcit.ca`

BCIT CST — September 2019

Welcome!

Welcome to your second COMP 1510 lab. Today's lab is all about familiarizing ourselves with operators and the Python 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 short programs and functions in this lab
2. (1 point) Correctly and fully test the functions in the `if __name__ == "__main__"` block
3. (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.

2 Requirements

Please complete the following:

1. Create a new project in PyCharm called COMP_1510_Lab_02. Remember to use your existing project interpreter.
2. Select the main project folder in the left-hand project pane of PyCharm. From the main menu, select VCS > Import into version control > Create git repository... Select the folder called COMP_1510_Lab_02 which contains the project we just created. We want to add this project to git version control. Select open.

3. We want to keep a copy of our project on the cloud, too. Select VCS again from the main menu, and then Import into version control and Share project on GitHub. Make sure the project is private, and select Share.

4. When you click Share, a window opens that suggests you add some files in the .idea folder to version control. Let's not. I'd like you to make sure the files are **not** checked. We will only put our source code (.py files) into version control. This way, someone else using our code can add it to a project on their laptop using their own PyCharm settings.

5. For this lab, I'd like you to implement some short programs and functions. I'd like you to commit and push your changes to GitHub after you complete each short program and function.

6. Create a new Python file (program) called my_circle.py in your project:

   (a) Let's write a script in this file. No functions, just a series of definitions and commands.

   (b) A circle isn't a circle without some PI. Declare a variable called PI and assign the value 3.14159.

   (c) Every circle has a radius. Declare a variable called radius. Give it a value of zero for now. We will ask the user to provide a new value.

   (d) Prompt the user to enter a radius value for a circle, and assign the value (as a float) to the radius variable.

   (e) The formula for the circumference of a circle is 2 x PI x radius. Calculate the circumference of the circle and store the result in a new variable called circumference. Print the circumference for the user.

   (f) The formula for the area of a circle is PI * radius * radius. Calculate the area of the circle and store the result in a new variable called area. Print the area for the user.

   (g) Let's have some fun. Suppose we double the radius of a circle. What happens to the area of the circle, does it also double? What about the circumference, does it double?

   (h) Modify your program so that when the user enters a value for the radius, a variable called double_radius stores double the value stored in radius.

   (i) Calculate the area and circumference of a circle using the double_radius value and store these values in new variables (you can choose the names).

   (j) Use division to determine how many times the area and circumference increase when the radius doubles. Print a useful message for the user.

7. Create a new Python file (program) called room_painter.py in your project. Let's write a program to determine how many cans of paint we need to paint a room. We will consider the length, width, and height of the room, and the number of coats we want to apply:

   (a) Let's write a script in this file. No functions, just use a series of definitions and commands.

   (b) Let's start by declaring a constant. Let's assume that a 4 litre can of paint will always (constantly) cover 400 square feet. Declare a variable called COVERAGE to store this value.

   (c) Prompt the user for the length of the room in feet. Store the result in a variable called length.

   (d) Prompt the user for the width of the room in feet. Store the result in a variable called width.

   (e) Prompt the user for the height of the room in feet. Store the result in a variable called height.

   (f) Prompt the user for the number of coats to enter. Store the result in a variable called coats.

   (g) Calculate the total surface area to be painted in square feet and store the result in a variable called surface_area. You can imagine the room is a cube, and we want to calculate the area of the 4 sides and the top. Don't paint the floor (oh perish the thought!).

   (h) Multiple surface_area by the number of coats to apply and store the result in a new variable called coverage_needed.

   (i) Divide coverage_needed by COVERAGE, the amount each can will cover, and store the result in a variable called cans_of_paint_required.

   (j) Print a message to the user telling them how many cans of paint they need to buy.

8. Create a new Python file (program) called functions.py in your project:

   (a) Inside functions.py, implement (create) a function called format_name. The format_name function must accept two strings, a first name and a last name. Inside the function, remove the leading and trailing whitespace, and return a concatenated string formatted in title case (first letter of each name capitalized and the rest in lower case) with one single space between the first and last name.

   (b) Implement a function called tripler. The tripler function must accept a parameter (type doesn't matter) and return a string that represents the parameter tripled. For example, invoking tripler(3) will return a string that contains 333. Invoking tripler("Python") will return a string that contains PythonPythonPython.

   (c) Implement a function called this_year. The this_year function must return the int 2019. Develop an interesting and surprising mathematical expression that evaluates to 2019, using only numbers and arithmetic operators.

   (d) Create a `if __name__ == "__main__"` block and add code that invokes a main function. Inside the main function, invoke the functions you just wrote. Prove to me that they work. Are there circumstances in which they won't work, i.e., are there combinations or types of input that will not or should not work?

9. Your final exercise is about base conversion. One algorithm for converting a base 10 number to another base b involves repeatedly dividing by b. Each time a division is performed the remainder and quotient are saved. At each step, the dividend (numerator) is the quotient from the preceding step; the divisor (denominator) is always b. The algorithm stops when the quotient is 0. The number in the new base is the sequence of remainders in reverse order (the last one computed goes first; the first one goes last, and so on):

   (a) I would like you to use this algorithm to convert a base 10 number to a 4-digit number in another base (you don't know enough programming yet to be able to convert any size number). The base 10 number and the new base (between 2 and 9) will be provided by the user.

   (b) The program will only work correctly for base 10 numbers that fit in 4 digits in the new base. We know that in base 2 the maximum unsigned integer that will fit in 4 bits is 1111_2 which equals 15 in base 10 (or 2^4– 1). In base 8, the maximum number is 7777_8 which equals 4095 in base 10 (or $8^4 - 1$). In general, the maximum base 10 number that fits in 4 base b digits is $b^4 - 1$.

   (c) Inside functions.py, implement a function called base_conversion. In the function, ask the user for their destination base (2 - 9).

   (d) Calculate the maximum base 10 number that will fit into 4 digits in the new base

   (e) Tell the user this maximum value, and prompt the user for a base 10 number that is equal to or less than the maximum.

   (f) Perform the conversion. I know, I make it sound so easy. Actually, it looks challenging at first, but once you do it a few times, it becomes easier to understand. It only takes a few steps.

   (g) Start by dividing the base 10 number by the new base. The remainder is what we will use for the right-most digit of the new base number. Save it in a variable.

   (h) When you divided the base 10 number by the new base, the result is called the quotient. Divide the quotient again by the new base number. The remainder is the second-to-right digit of the new base number.

   (i) Repeat this two more times (divide the remainder from the previous step by the new base) for the second and first digits of the new base number.

   (j) Concatenate the results into a string, convert it to an int, and print the result.

   (k) Inside the main function, invoke your base_conversion function. Make sure it works, and then add your name to the marking list.

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