Lab 02: Python Basics

Goals for this lab:

1. Write a simple program that takes in user input and outputs formatted text to the screen
2. Identify and correct errors in a Python program
3. Write a program that performs multiple mathematical operations
4. Convert text data input numeric data using appropriate functions
5. Write and execute simple Python programs using the PyCharm IDE

General Guidelines:

- Use meaningful variable names
- Use appropriate indentation
- Use comments, especially for the header, variables, and blocks of code.

Example Header:

```
# Author: Your name
# Date: Today's date
# Description: A small description in your own words that describes what
the program does. Any additional information required for the program to
execute should also be provided.
```

1. Preparation

You will be using two premade Python files for this lab, so be sure to download the **Lab02StarterFiles.zip** that contains **lineSlope.py** and **motion.py** from the Canvas Lab02 link before you begin the lab. It is also recommended that you create a new folder Lab02 to organize all of your **.py** files for this lab.

2. Creating a New Program

For the first part of the lab you will need to create a simple Python program to input some text and output a formatted string

- Start PyCharm, create a new file, and save it as **fullName.py**
- Your Python program should perform the following steps:
    - Prompt the user to input their first name, using an appropriate message, and store the response in an appropriately named variable
    - Prompt the user to input their last name, using an appropriate message, and store the response in an appropriately named variable
    - Output the user's name in last name, first name format. i.e. if the user enters Alexia and Tarabotti, the program should output Tarabotti, Alexia
- Execute your program to test if it is working correctly, and you are receiving the desired output

Make sure you save your **fullName.py** as you will be submitting this file for Lab 02.

3. Detecting and Correcting Errors

It is an inevitability that programmers are going to make mistakes while designing and implementing programs. Some of the most important skills that programmers develop over time are the ability to identify what type of errors exists in a program, where they exists, and, most importantly, how to fix the problems. The good news is, many of the tools used to develop software are designed to aid programmers in identifying errors, though the programmer must still determine the best way to resolve the error.

For this program, start by opening the **lineSlope.py** file, which you downloaded from Canvas, in PyCharm, by using `File -> Open` option in the PyCharm menu and selecting the **lineSlope.py** file. This program is designed to compute the slope of a line given two end points (x_1,y_1) and (x_2,y_2) using the formula:

$$m = \frac{y_1 - y_2}{x_1 - x_2}$$

The expected output for this program should be:

```
Starting point: ( -2 , 1 )
Ending point: ( 5 , 36 )
Slope of the line = 5.0
```

However, **lineSlope.py** contains several syntax errors and at least one logic error. Syntax errors represent a violation of the rules of the programming language. When the Python interpreter reaches a syntax error, it cannot understand the code and will halt the interpretation of the program. Even when a program is free of

syntax errors, logic errors may still cause undesirable results. A logic error represents a mistake in the design or implementation of the program that prevents the program from producing correct results.

For this program:

- Update the header information for the program with your name and the date
- Debug the program (i.e. find and correct all of the errors) so that the results match the expected outputs above

Make sure you save your **lineSlope.py** as you will be submitting this file for Lab 02.

4. Working with Math

For the next part of this lab, you will construct your own program. You have been provided a "starter file" named **motion.py**, to which you will add the code necessary to compute and display the velocity, average velocity, and displacement of an object, given the amount of time elapsed. The formulas for each of these properties are provided below:

$$v = v_0 + at$$

$$\bar{v} = v_0 + \frac{1}{2}at$$

$$x = v_0 t + \frac{1}{2}at^2$$

After completing the calculations, your program should output the following information:

```
time = 10.0

velocity         = 60.75
average velocity = 34.5
displacement     = 345.0
```

Note that your output should be formatted exactly as shown, so you may want to use tabs and newlines (`\t` and `\n`) in addition to spaces when constructing the string for your `print` function.

Make sure you save your **motion.py** as you will be submitting this file for Lab 02.

5. User Input

As you probably noticed with the **motion.py** program, in order to calculate values for different time spans, you needed to alter the value of the `time` variable in the code. To make this more dynamic, we should prompt the user for the value. For this part of the lab:

- Make a copy of your **motion.py** file by saving a new instance of the file named **motionInput.py** using PyCharm's `File -> Save as` menu item
- Modify the program such that it prompts the user for the value of time using the `input` function and stores that value in the `time` variable
- Test your program to see if it executes correctly

You may have seen that your program outputs an error such as:

```
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
```

Or

```
TypeError: can't multiply sequence by non-int of type 'float'
```

This is because any data that is input by the user is of type string, not a numeric type. To correct for this, we must convert the text into a number (in this case we want a floating point number) by parsing the text with the `float` function as seen below:

```
time = float(input("An appropriate prompt for information "))
```

Now when the `input` function obtains the user's data, that data will be given to the `float` function (notice how the `input` function is inside the () for the `float` function), which will convert the text into a floating point number and store the number in the radius variable. Be sure to test your program on a few different inputs to ensure that your program is correct.

Make sure you save your **motionInput.py** as you will be submitting this file for Lab 02.

Submission

Once you have completed this lab it is time to turn in your work. Please submit the following files to the Lab 02 assignment in Canvas:

- **fullName.py**
- **lineSlope.py**
- **motion.py**
- **motionInput.py**

Sample Output

fullName.py

```
Please enter your first name: Alexia
Please enter your last name: Tarabotti
Tarabotti,Alexia
```

lineSlope.py

```
Starting point: ( -2 , 1 )
Ending point: ( 5 , 36 )
Slope of the line = 5.0
```

motion.py

```
time = 10.0

velocity         = 60.75
average velocity = 34.5
displacement     = 345.0
```

motionInput.py

```
Please enter the elapsed time: 15.5
time = 15.5

velocity         = 89.625
average velocity = 48.9375
displacement     = 758.53125
```

Rubric

For each program in this lab, you are expected to make a good faith effort on all requested functionality. Each submitted program will receive a score of either 100, 75, 50, or 0 out of 100 based upon how much of the program you attempted, regardless of whether the final output is correct (See table below). The scores for all of your submitted programs for this lab will be averaged together to calculate your grade for this lab. Keep in mind that labs are practice both for the exams in this course and for coding professionally, so make sure you take the assignments seriously.

Score	Attempted Functionality
100	100% of the functionality was attempted
75	<100% and >=75% of the functionality was attempted
50	<75% and >=50% of the functionality was attempted
0	<50% of the functionality was attempted