Hitakshi Verma

August 16, 2020

Foundations of Programming: Python

Assignment 06

<https://github.com/hitakshi01/IntroToProg-Python-Mod06>

**Create a script using create custom functions**

**Introduction:**

For Module06, Randal Root, my professor has introduced me with a new python’s programming concept called *Functions and Classes*.

By using this knowledge, I will be going to create a python script that manages “ToDo List” (*that we have used in our previous assignment05*) where code loads data from a file into a Python List of Dictionary objects using *functions and classes*. As a part of this assignment, I will push and commit my code on source control version software called *GitHub* and, also create a *GitHub web page.*

Along with this script*,* I am sharing the learned fundamental concepts of classes and functions that help you to quickly understand my created python script.

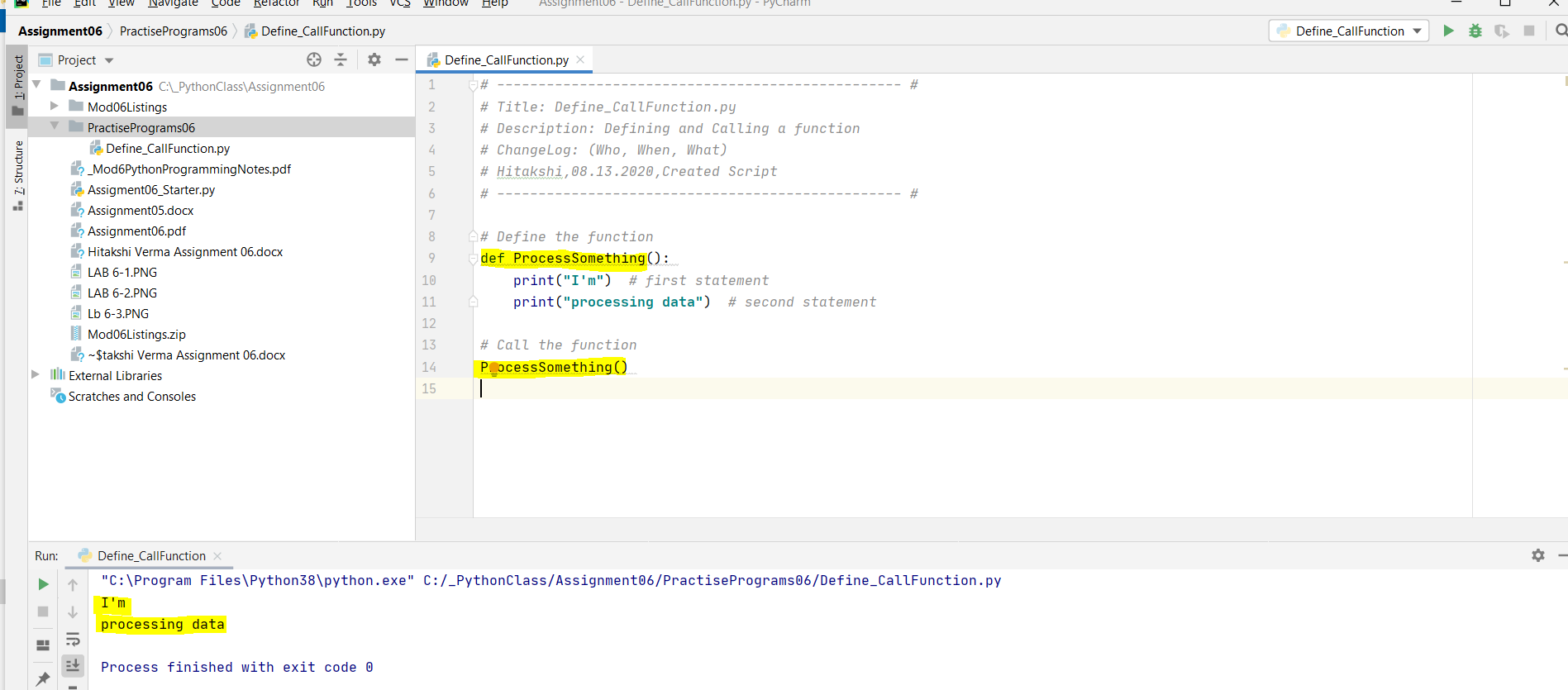
**Functions:**

*Function* in Python is used to utilize the code in more than one place in a program. It is also called method or procedures. Python provides you many inbuilt functions like print(), but it also gives freedom to create your own functions(custom functions).Functions are a way of grouping statements.

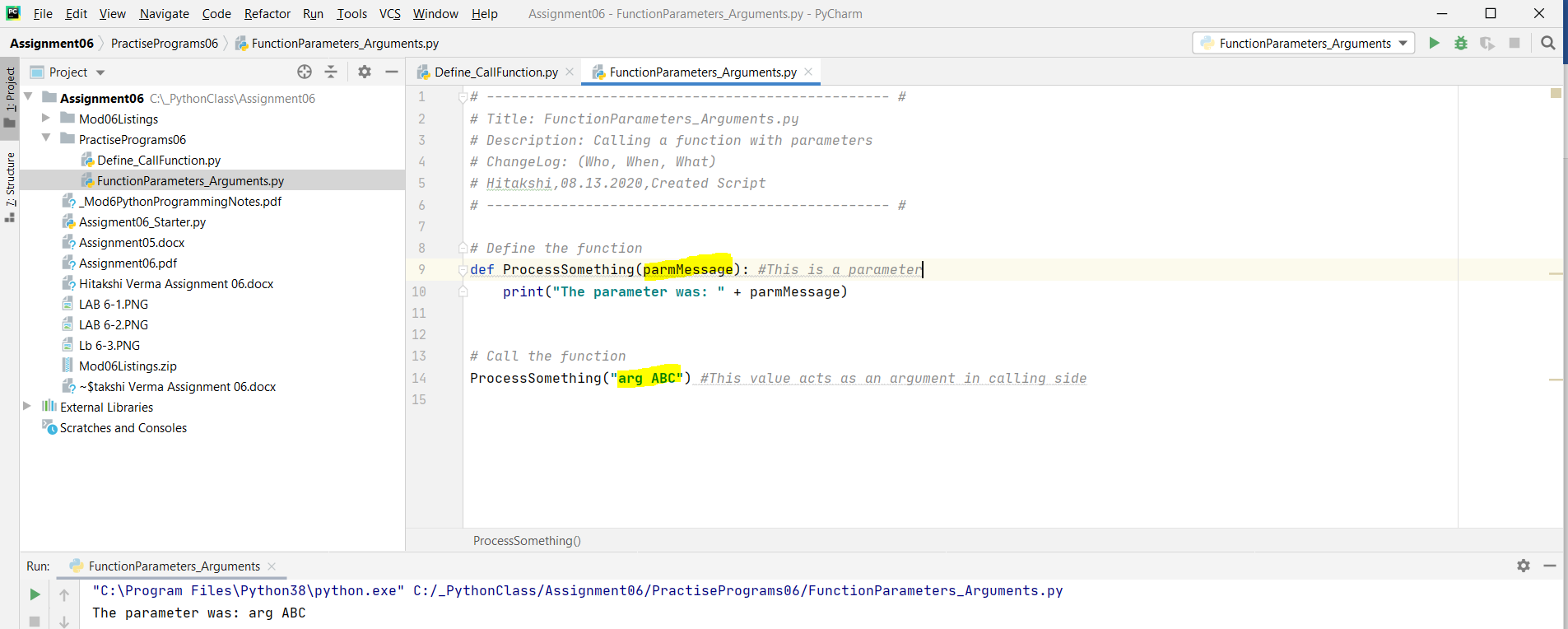
In Python, you *must define a function* before you can use code to call the function. Function in Python is defined by the**"def "**statement followed by the *function name* and *parentheses (()).*

*Calling the function* executes the statements in the function.

Below is the code snippet that shows how the function is first defined and then called.

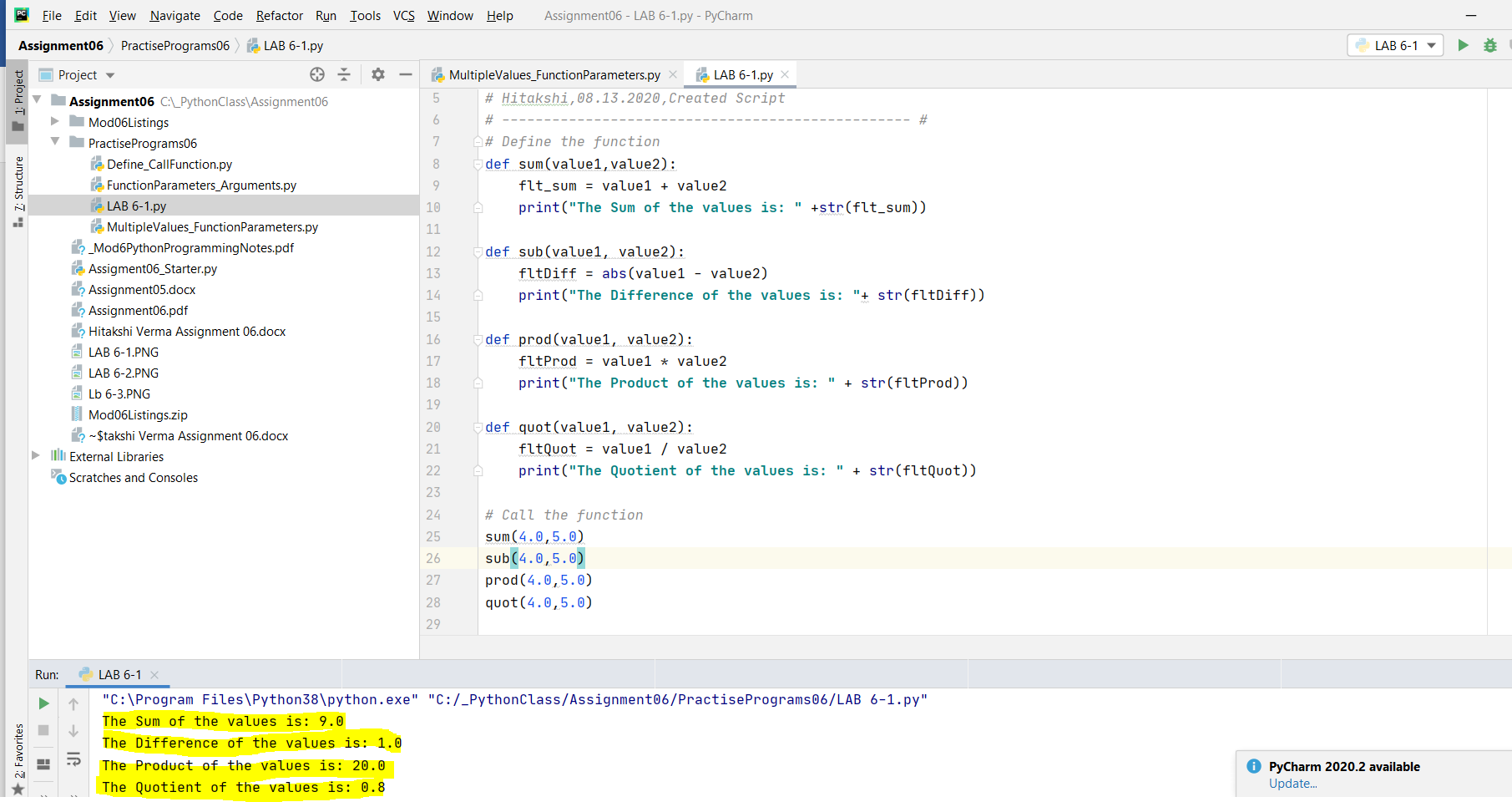
 *Figure1: Script demonstrates function defining and calling*

Function can have parameters. Where the function is defined, we call that value in parenthesis as *Parameter*. The argument is a value that is passed to the function when it's *called*. In other words, on the calling side, it is an *argument*.

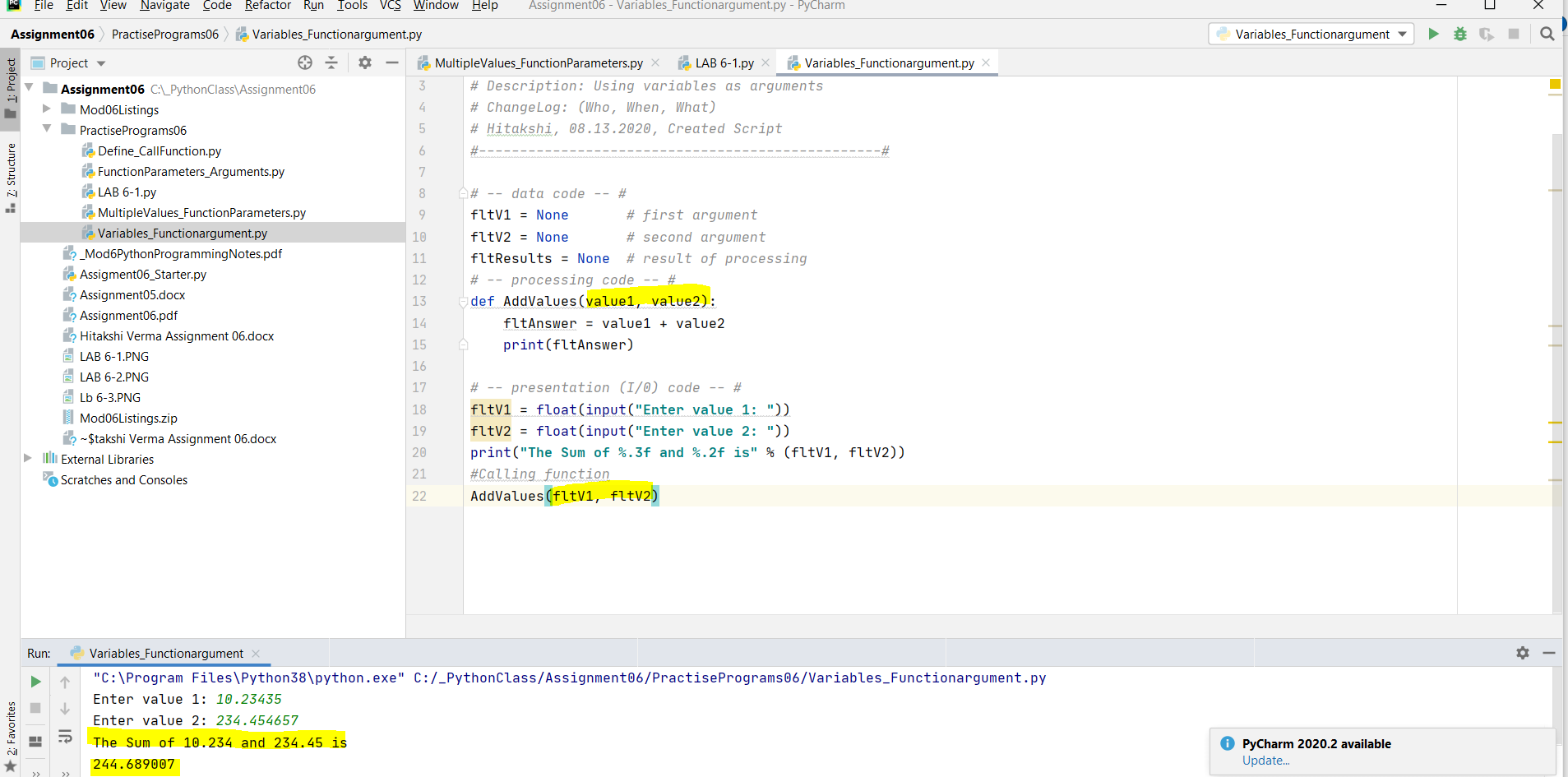
*Figure2: Script demonstrates function’s parameter and argument*

***Working on LAB 6-1:***

In this lab, I am creating a python script that calculates the sum, difference, product and quotient of two numbers using functions. Here, I am modularizing the code using separate functions like sum() for addition, sub() for subtraction, prod() for product and quot() for getting the quotient. I have also used abs() function in nested sub() for absolute result.

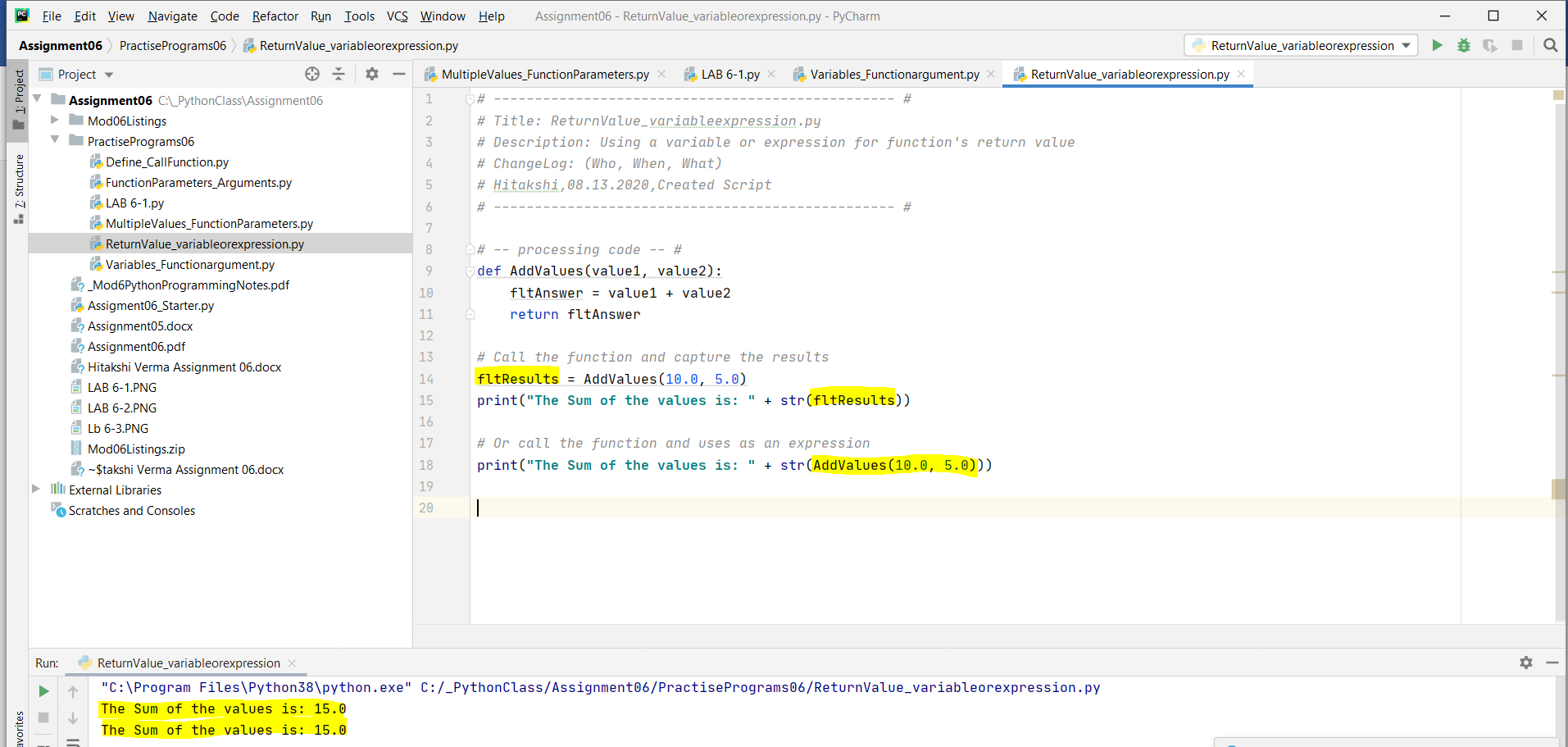
 *Figure3: Script demonstrates the calculation of addition, subtraction, product and quotient of two float numbers*

*Variables can also be used as arguments*. This is a useful approach as we can use the value of that variable for further processing.

*Figure4: Script demonstrates the use of variables as function arguments*

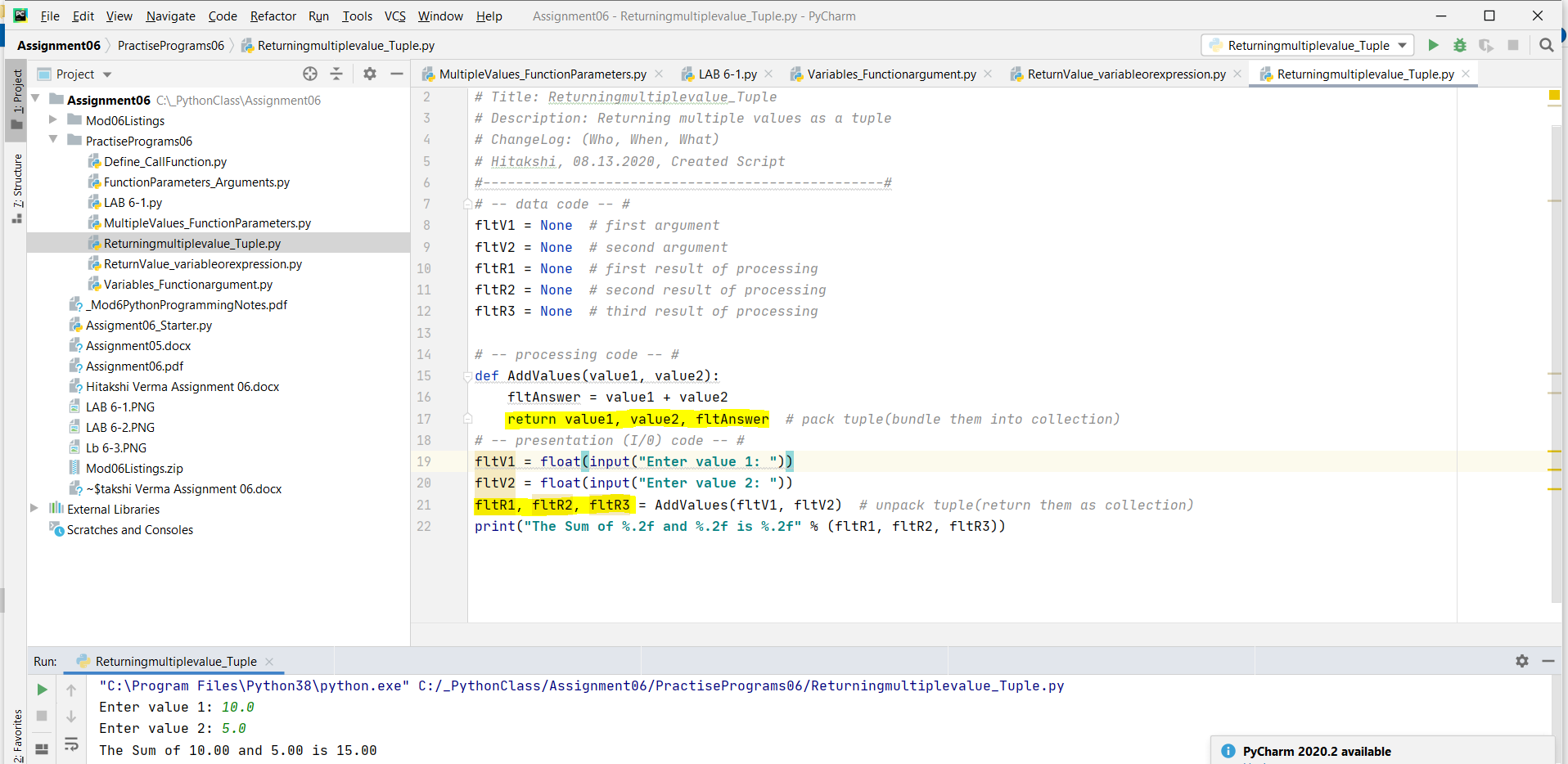
Functions can return one or more values. *Returned values* can be captured by using variable. Again, this returned value can be used further for processing at multiple times.

If we don’t want to use the variable, we can simply return the value in calling function that acts as an *expression. This will immediately reflect the result without placing the returned value in a variable.*

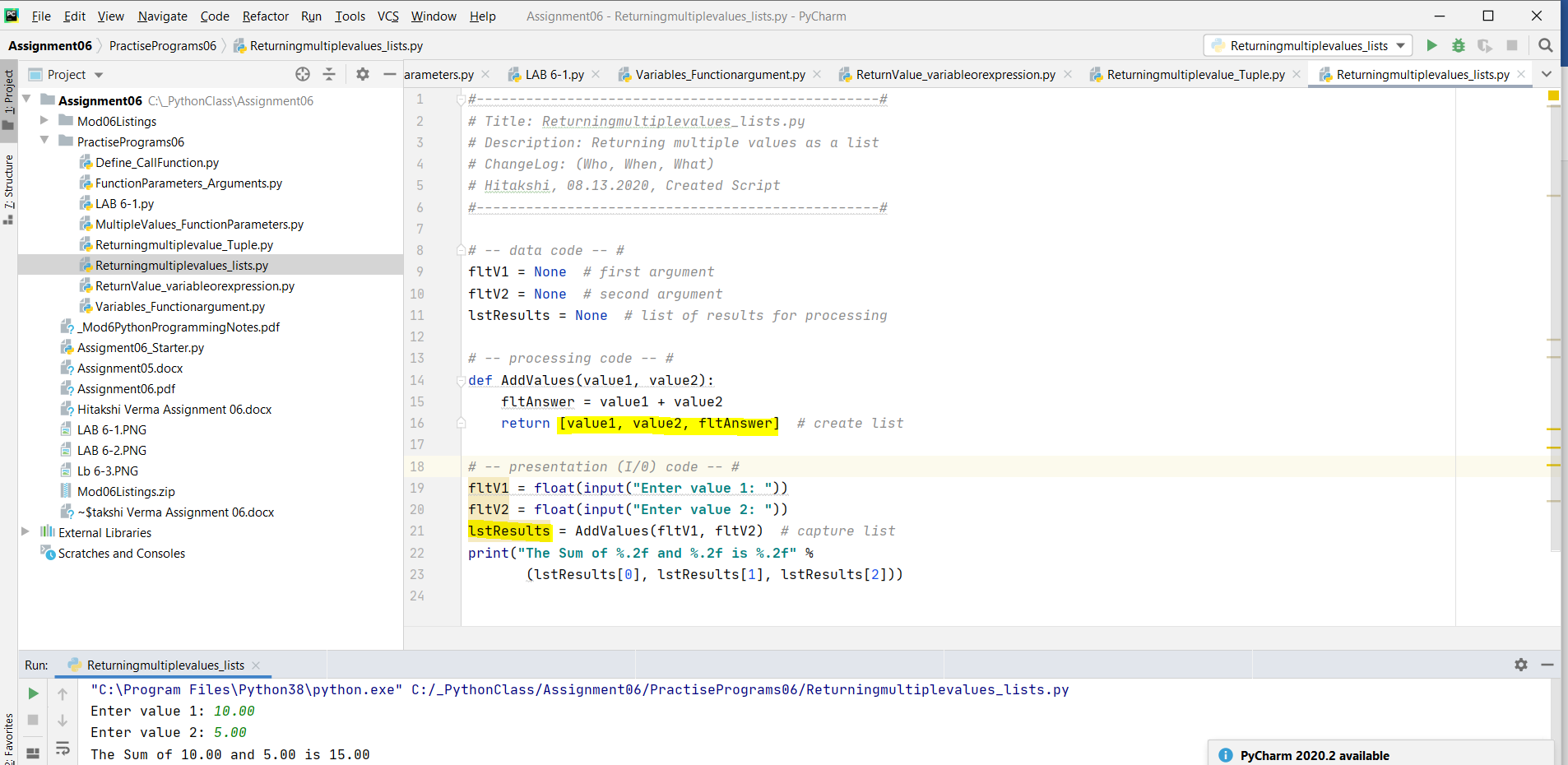
*Figure5: Script displays the use of variables or function acts expression to capture return value*

Functions can also return multiple values. To return the multiple values, we need to bundle them into collection (using tuple or lists) and return that collection.

*Returning multiple values as Tuple:*

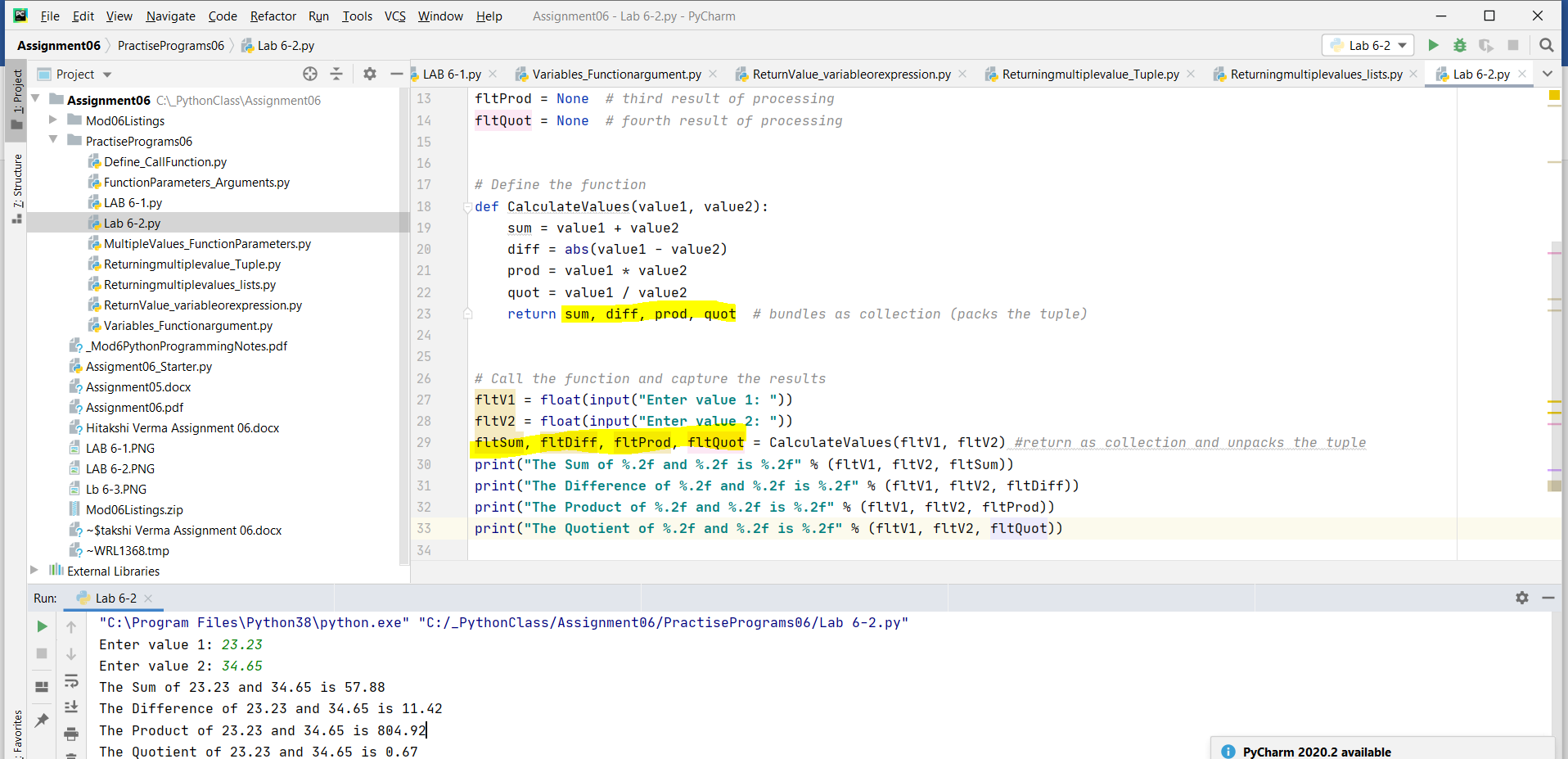
*Figure6: Script displays the bundling and returning of multiple values as tuple*

*Returning multiple values as List:*

*Figure7: Script displays the bundling and returning of multiple values as lists*

***Working on LAB 6-2:***

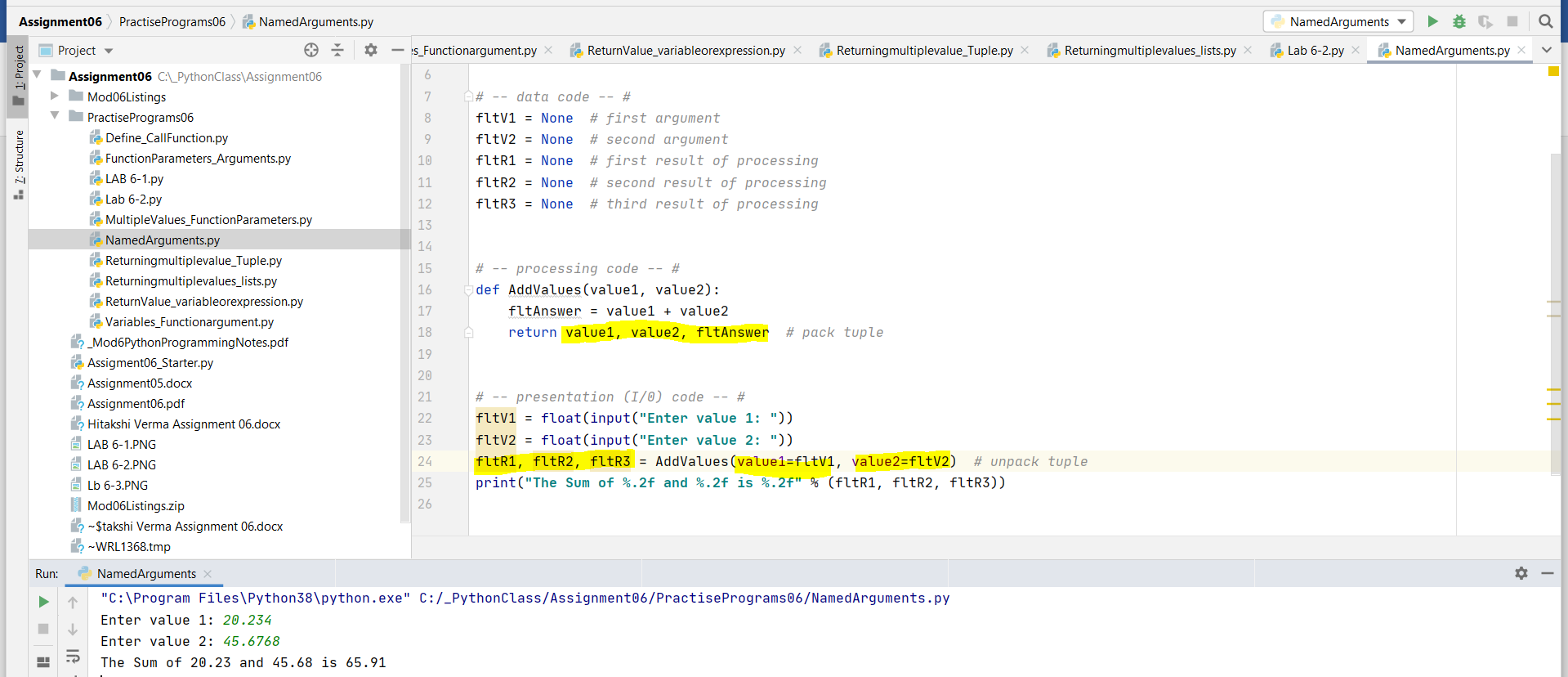
In this lab, I am creating a function that returns the Sum, Difference, Product, and Quotient of two numbers as a tuple.

*Figure8: Script displays functions that returns the bundling and returning of multiple values as tuples*

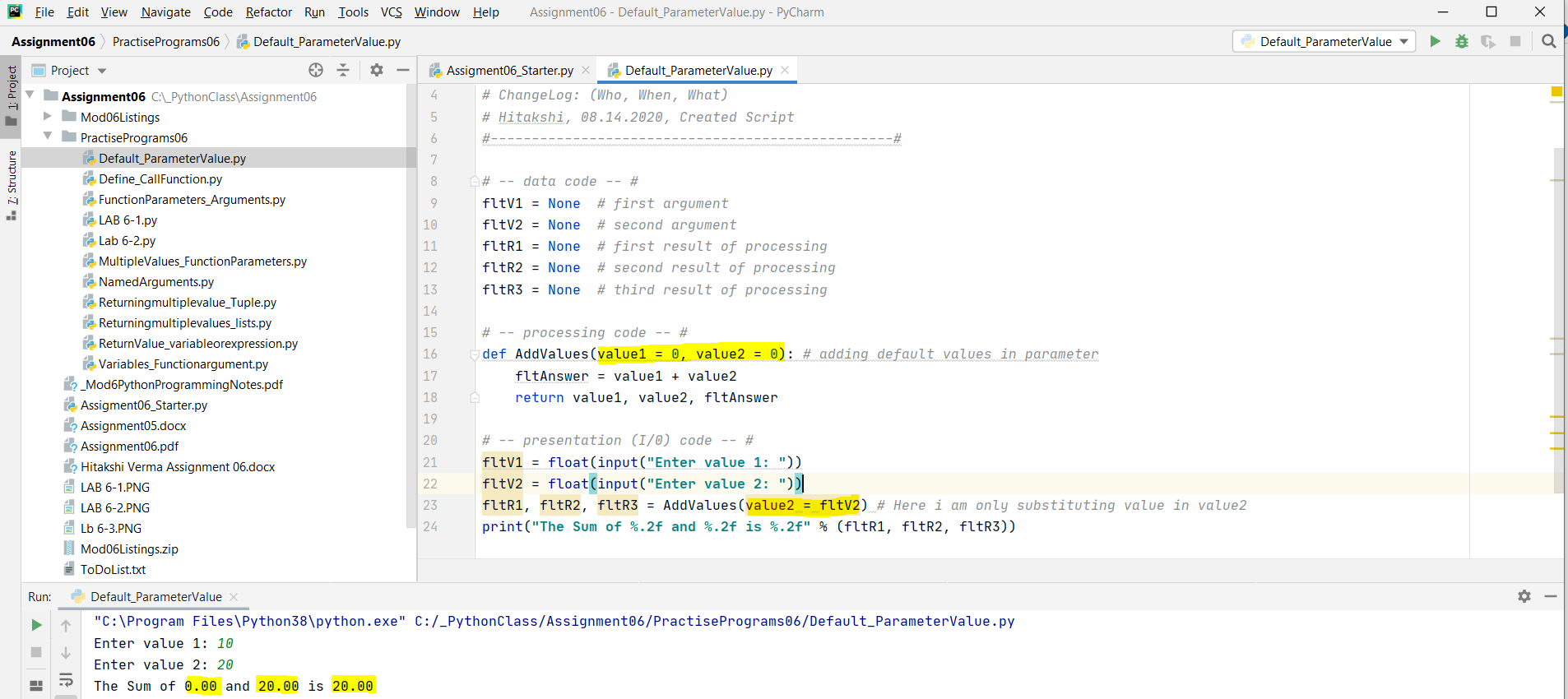
*Positional vs Named Arguments:*

Normally in all the above programs we are using variables or values that can be positioned into def function parameter as p*ositional arguments*.

So, when you call a function, you can include the name of the parameter and fill it explicitly with your arguments as *named arguments*.

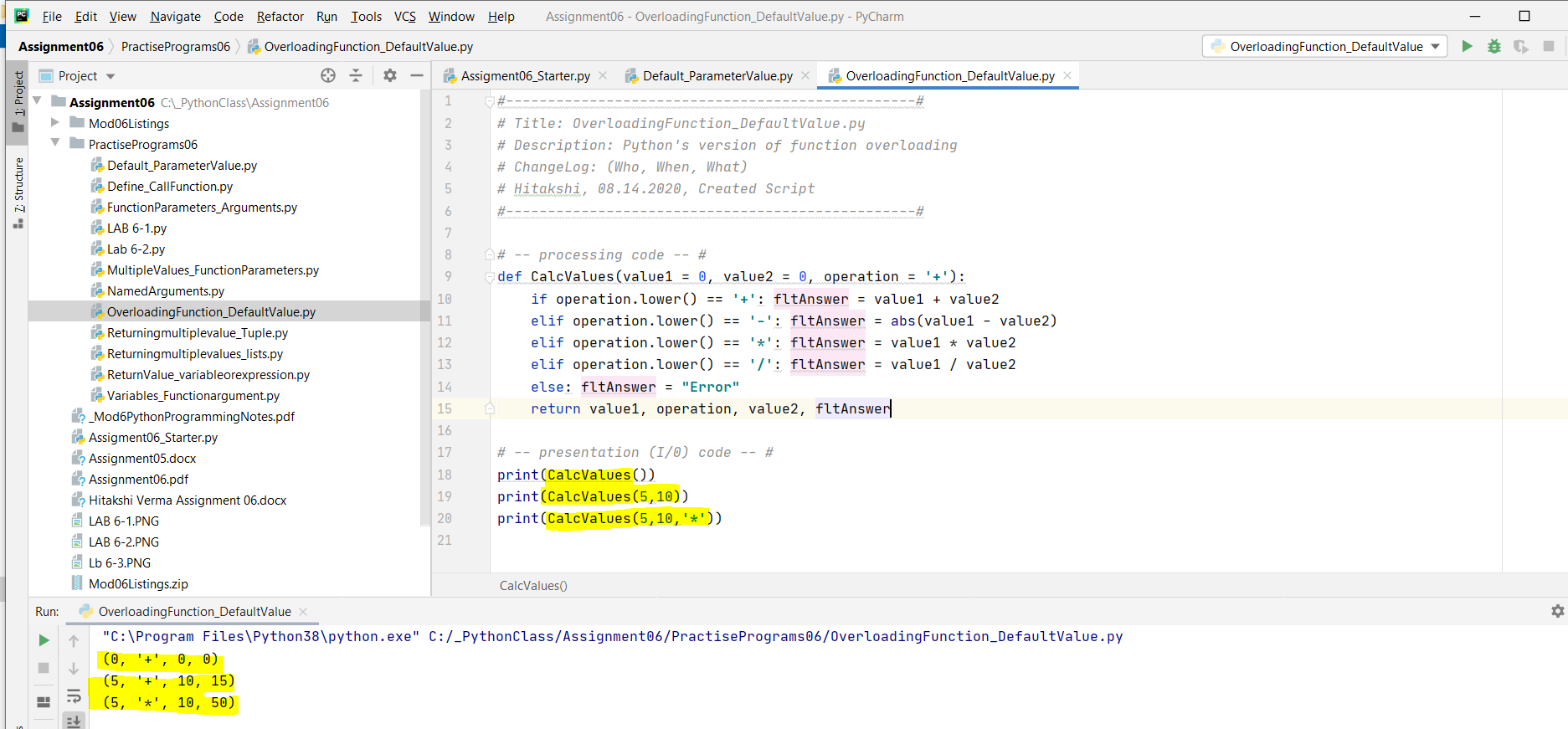
*Figure9: Script demonstrates the named arguments*

We can set default values in parameter where the function is defined. When we don’t supply the argument in a function forces the function to use parameter’s default value.

*Figure10: Script demonstrates default value in parameter*

*Overloaded Functions*

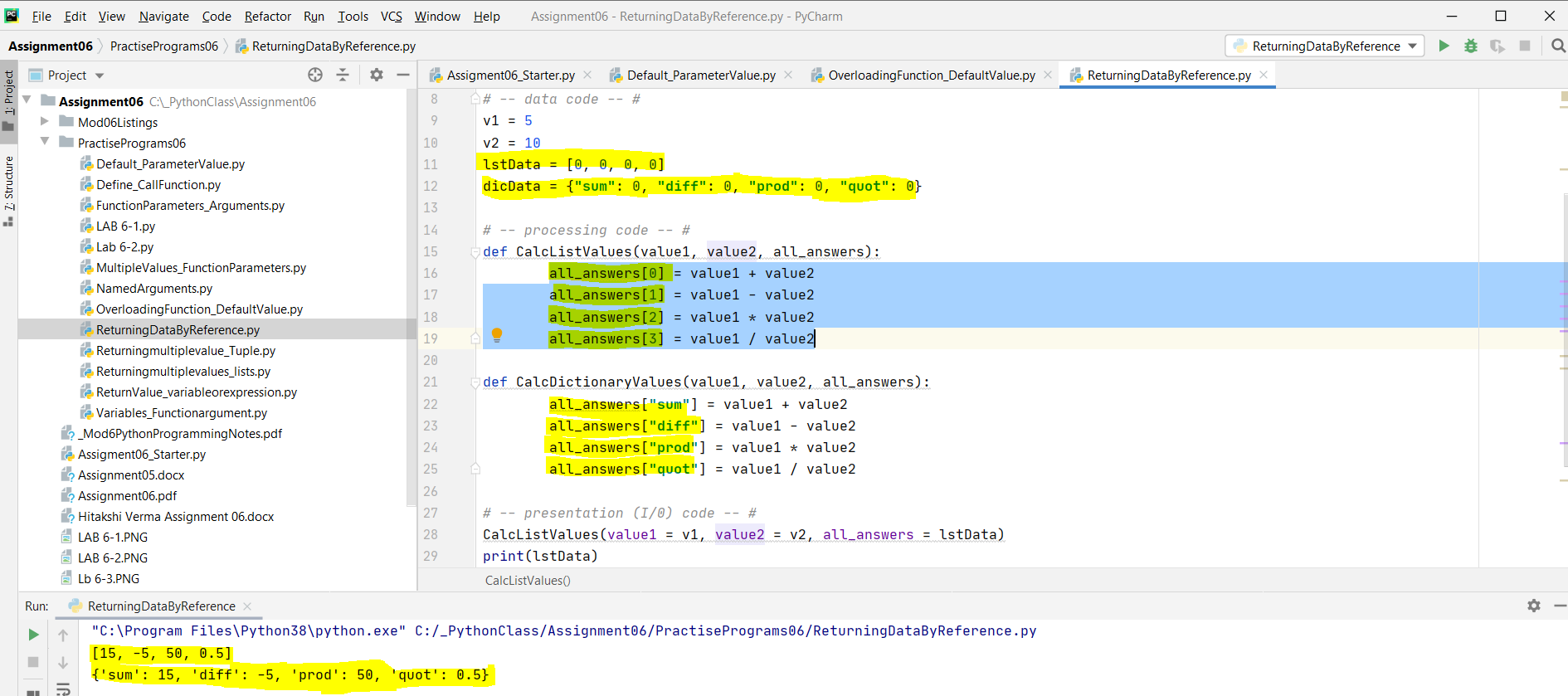
Python uses default values in parameter to accomplish overloading function concept.

*Figure11: Script demonstrates python’s version of function overloading*

*None Keyword is also used as default value in function’s parameter and can be used with is keyword to compare the values*

*Returning Data by Reference:*

In Python, simple data acts as value type and complex data acts as reference type.

*Figure12: Script demonstrates returning data type by reference*

*Global Vs Local Variable:*

Variables declared in a function are considered local to the containing function and cannot be accessed outside of that function. Variables declared in a "body" of the script are considered global to the containing script and can be used anywhere in the script.

Global as keyword should be used to represent a variable as global else local variable will "shadow" the global one whenever you assign a value to a variable with the same name.

It is a common practice to include a header at the beginning of a function, which is known as *docstring in python.*

***Below is the sample of doc strings:***

**def** AddValues(value1=**0.0**, value2=**0.0**):

""" This function adds two values

: param value1: (float) the first number to add

: param value2: (float) the second number to add

: return: (float) sum of two numbers

"""

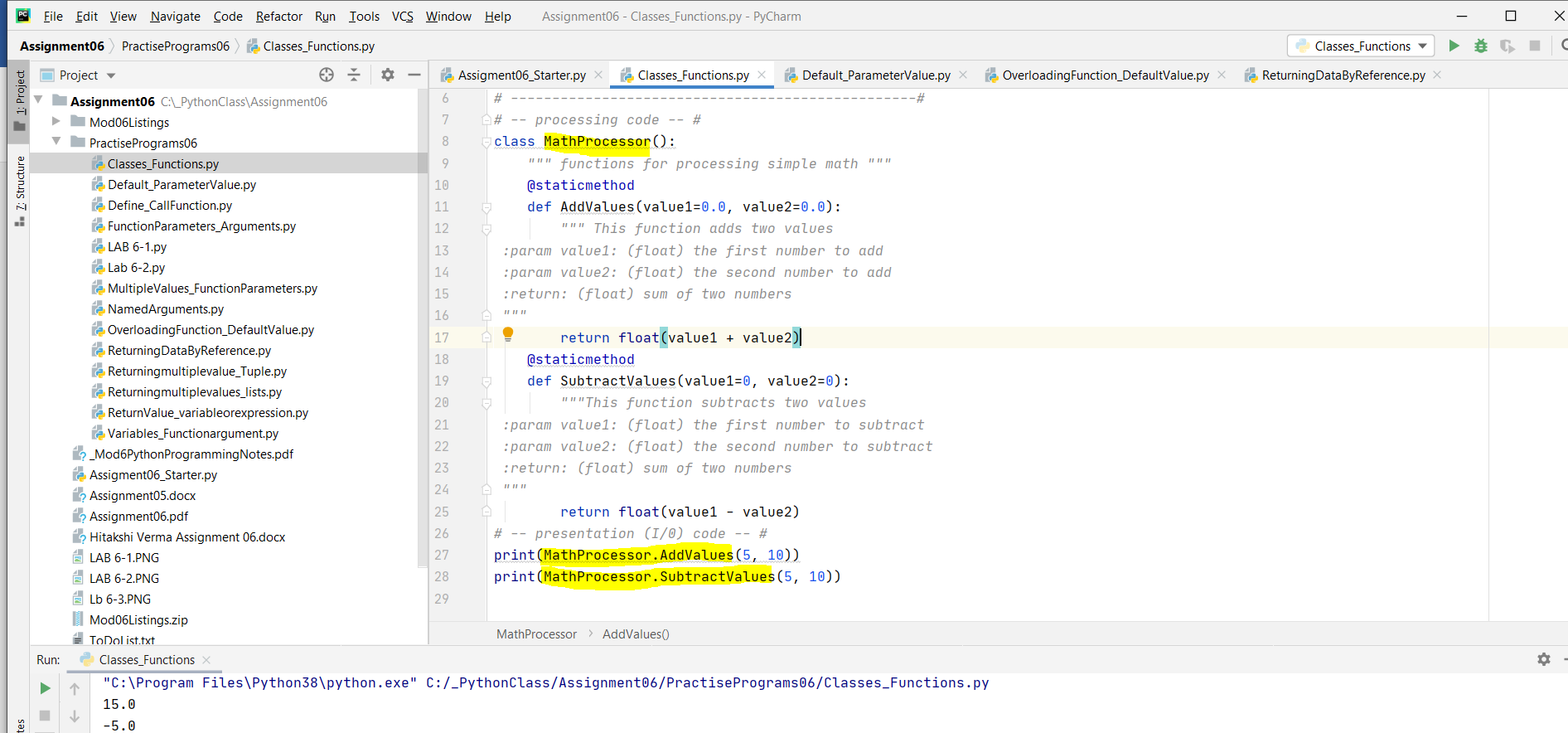
**return** value1 + value2

**print** (AddValues(5,10))

**Classes and Functions:**

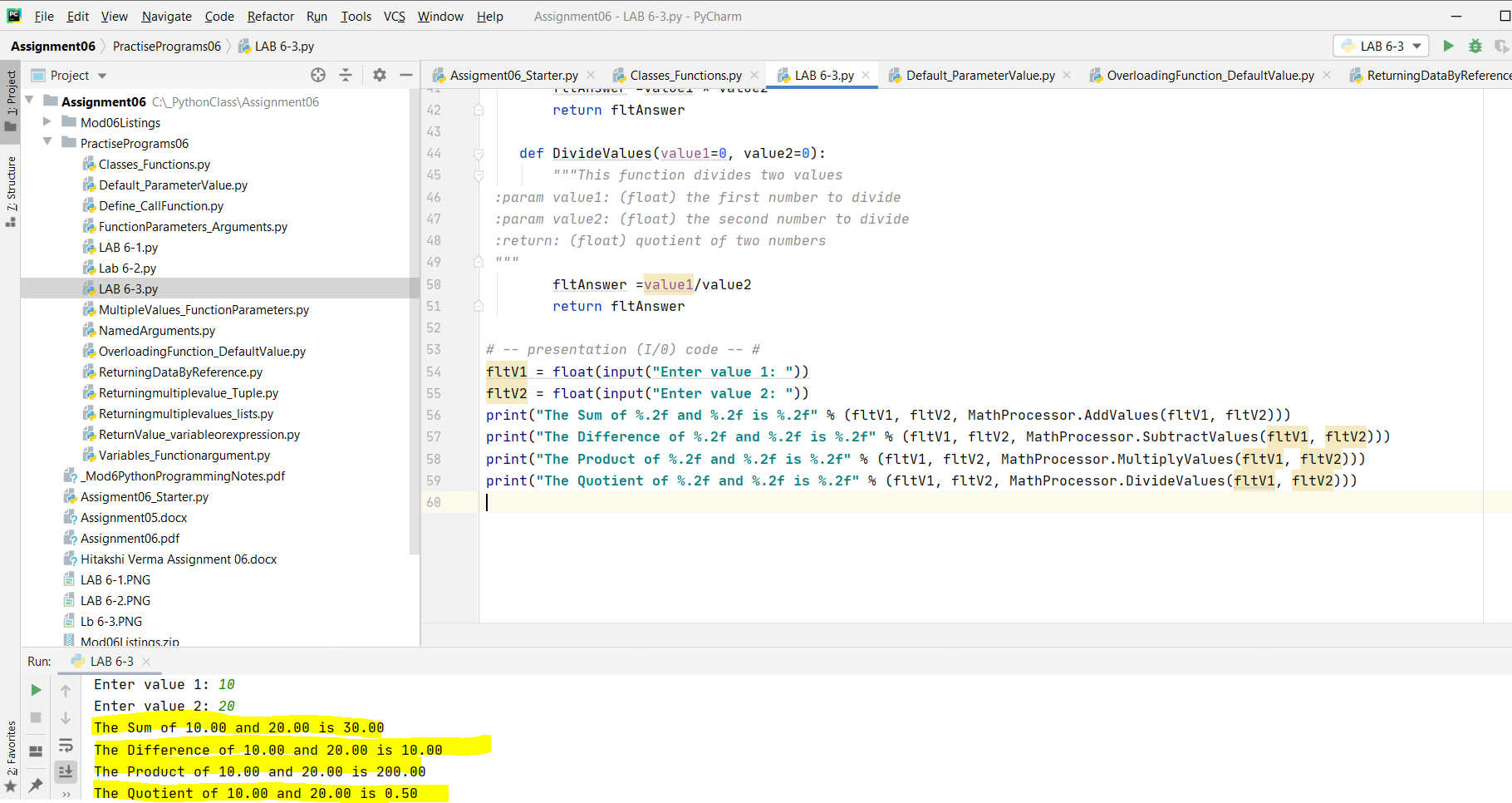
Python is an object-oriented programming language. Almost everything in Python is an object, with its properties and methods. A *Class* is like an object constructor, or a "blueprint" for creating objects.

*Classes are a way of grouping functions, variables, and constants.*

*Figure13: Script demonstrates python’s function and its encapsulation in class*

***Working on LAB 6-3:***

In this lab, I am creating a class that includes four functions, each returning either the Sum, Difference, Product, and Quotient of two numbers as a float result.

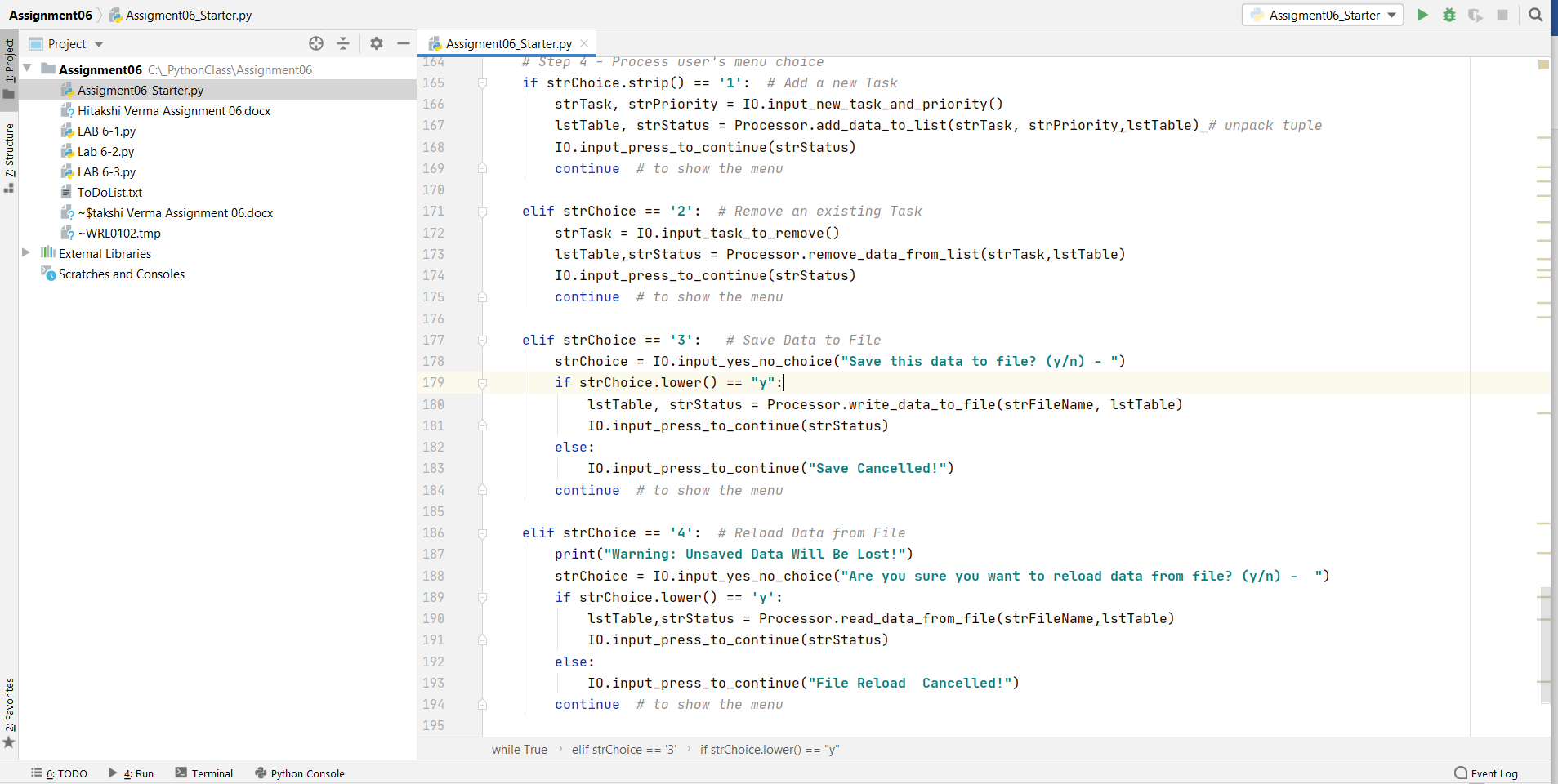
*Figure14: Script demonstrates the MathProcessor class and its functions*

**Assignment 6 – Modify data in a TodoList.txt file with lists and dictionary and functions**

This assignment was similar to the previous one with a slight enhancement of using functions. This clearly showed how clean, modular and easy a code becomes if functions are used. It was a little tricky to use the starter file as it had predefined functions and parameters. This allowed me to think backwards based on the functions’ input and output parameters. In the next sections I have laid out various steps followed to write the code.

*Step 1:*

I started from the main body of the code and in this step I loaded the data from the text file into a list of tables (lstTable). For this, there was already a function created read\_data\_from\_file which takes 2 parameters in the input (file\_name and list\_of\_rows). In this function, I opened the file in read mode and iterated through the file to split each line based on “,” and stored each task and priority in a dictionary. These dictionaries are appended into a list towards the end. I closed the file after the loop and returned the list and a success message.



*Figure 1: Overall main body of the code*

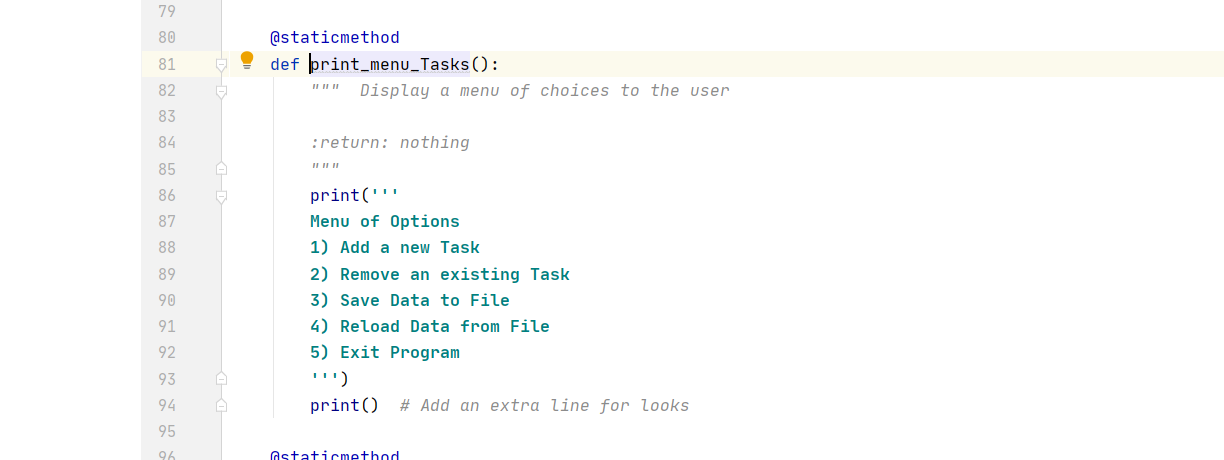
*Step 2:*

In this step, I displayed a list of menu options in a while loop which will run indefinitely unless the user selects the option 5 – Exit.

*Step 3:*

In this step, I used the following 3 functions

* print\_menu\_Tasks – displays the list of all tasks present in the list using a for loop which iterates over the list\_of\_rows
* print\_current\_Tasks\_in\_list – this is a static method which only displays the list of 5 options available to the user.
* input\_menu\_choice – this asked the user to enter one of the 5 options, removes the trailing characters using strip() and returns this value to the main body. This is stored in a variable *strChoice*.



*Figure 2: Menu tasks displayed to user*

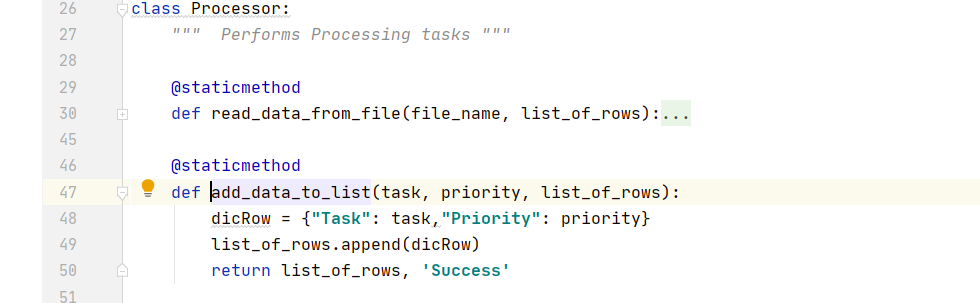
*Step 4:*

In this step, I added code for each of the user input choices. Code was similar to what was used in the previous assignment but with functions.

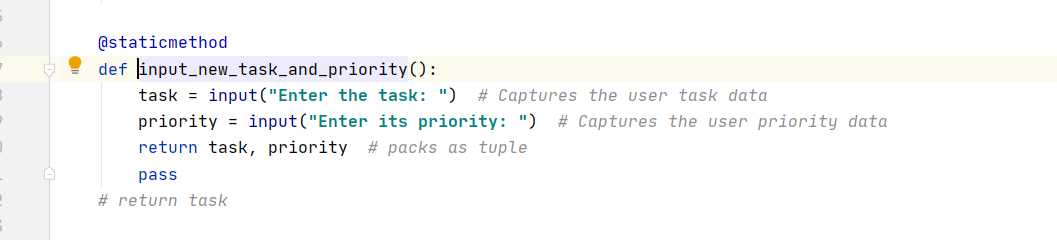
*Step 4.1*

This step added a new task to the existing list of dictionaries.

* I called a function input\_new\_task\_and\_priority which takes input from the user and returns the task name and priority.
* In the second method add\_data\_to\_list, the task and priority are added as a dictionary and appended into the list. It returns the list and the success message.



*Figure 3: Code to add data to list*

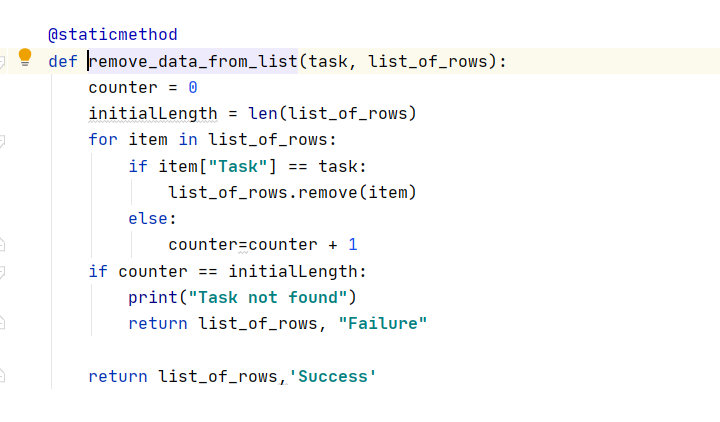


*Figure 4: Function to input new task and priority*

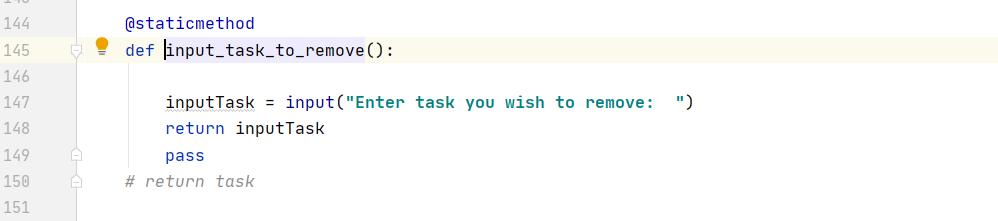
*Step 4.2*

This was the second choice which removes a task from the list. The below functions were used –

* input\_task\_to\_remove – asks the user to input the task name to be removed. This is stored in a variable inputTask and returned from this function.
* remove\_data\_from\_list – in this function, I took 2 variables counter and initialLentgh. I iterated over the for loop, and compares the user input task with the task name in the list using if condition. If the item was not found, the counter value was incremented to 1. If the counter value was equal to the initialLength then task was not found.



*Figure 5: Function to remove data from list*

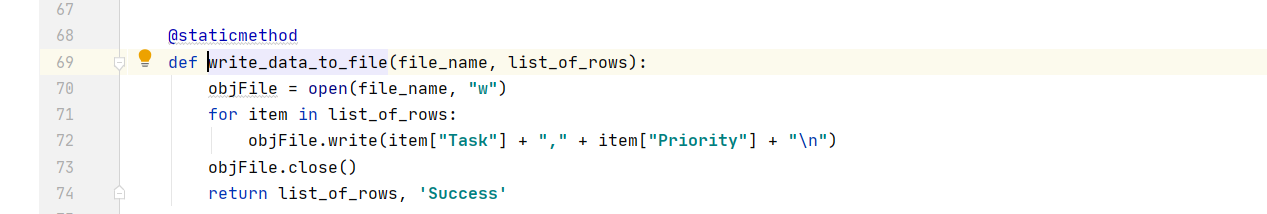


*Figure 5: Function to input task to be removed*

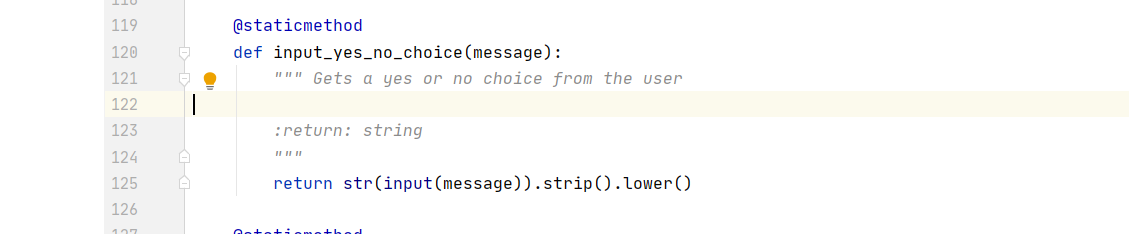
*Step 4.3*

This step asked user if he/ she wanted to save the file. The following functions were used –

* input\_yes\_no\_choice – user input to select “y” or “n”. This was a straight forward function which takes input from user, used strip() and lower() to sanitize the input and convert it to lower case.
* write\_data\_to\_file – this function opened the file in “write” mode and stored all the tasks and corresponding priorities in a comma separated format in the file. In the end, I closed the file and returned “Success”.



*Figure 6: Function to write data from list to file*



*Figure 7: Function to ask user to input “y” or “n*”

*Step 4.4*

This step reloads the data from file and prints a warning message to user that all unsaved data will be lost. Similar to the previous step, this also asks the user to confirm if he wanted to continue, if the option selected was “y” then I used the previous function read\_data\_from\_file to read the data from file once again; else do nothing and display to the user that option cancelled.

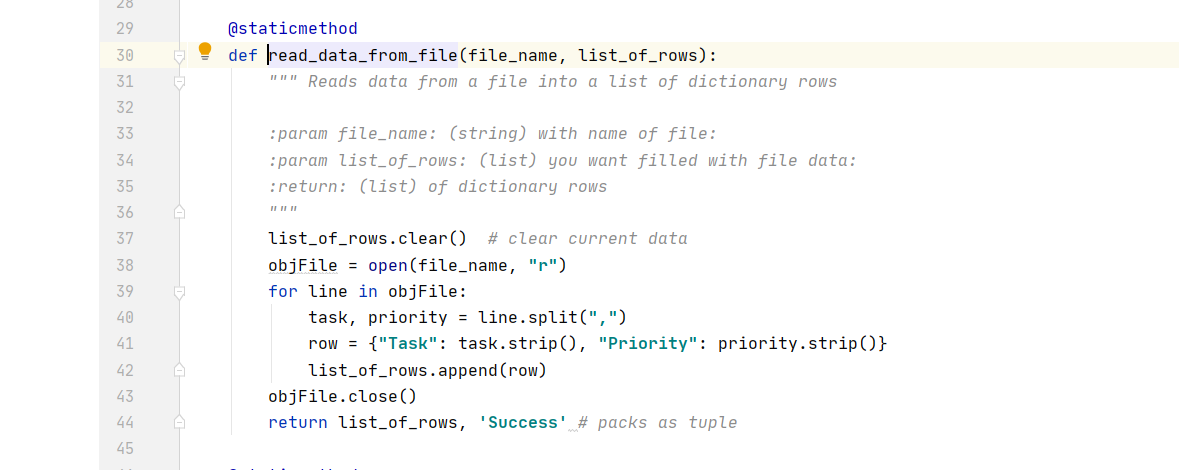
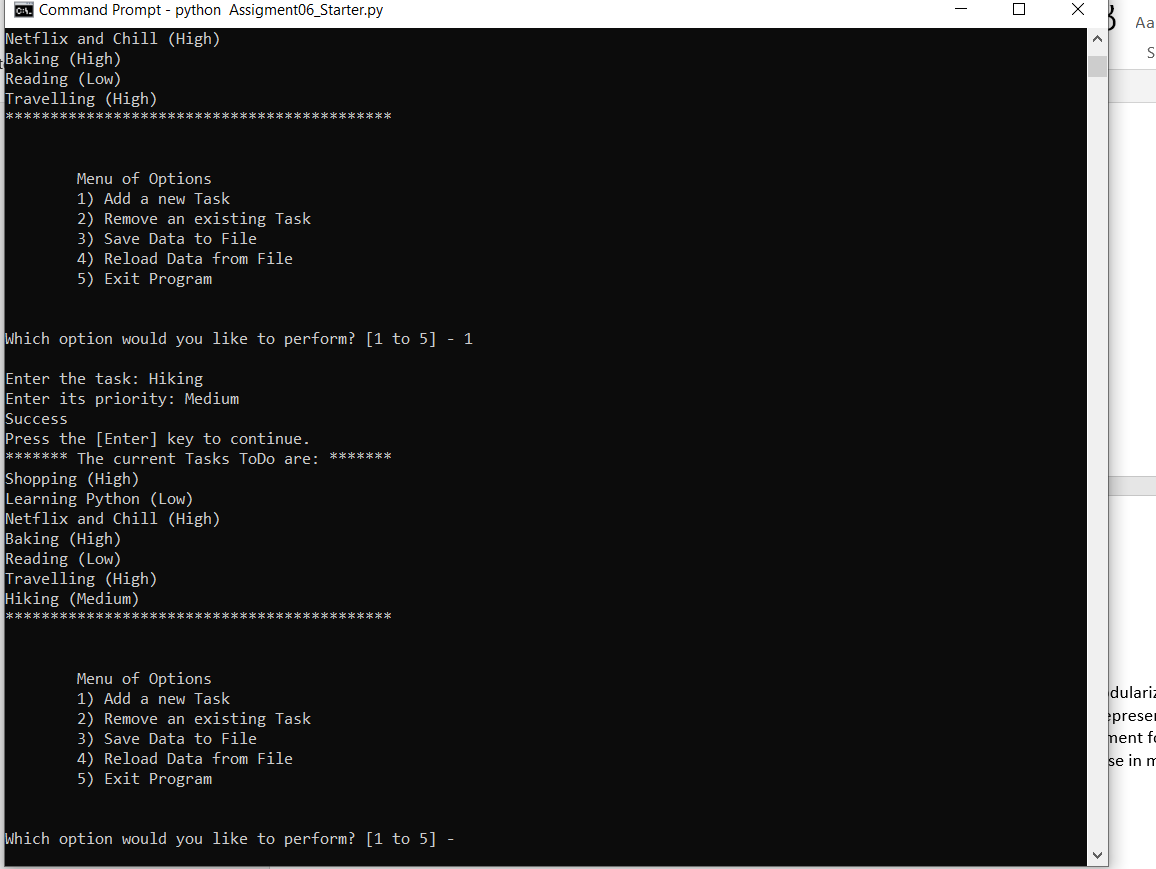


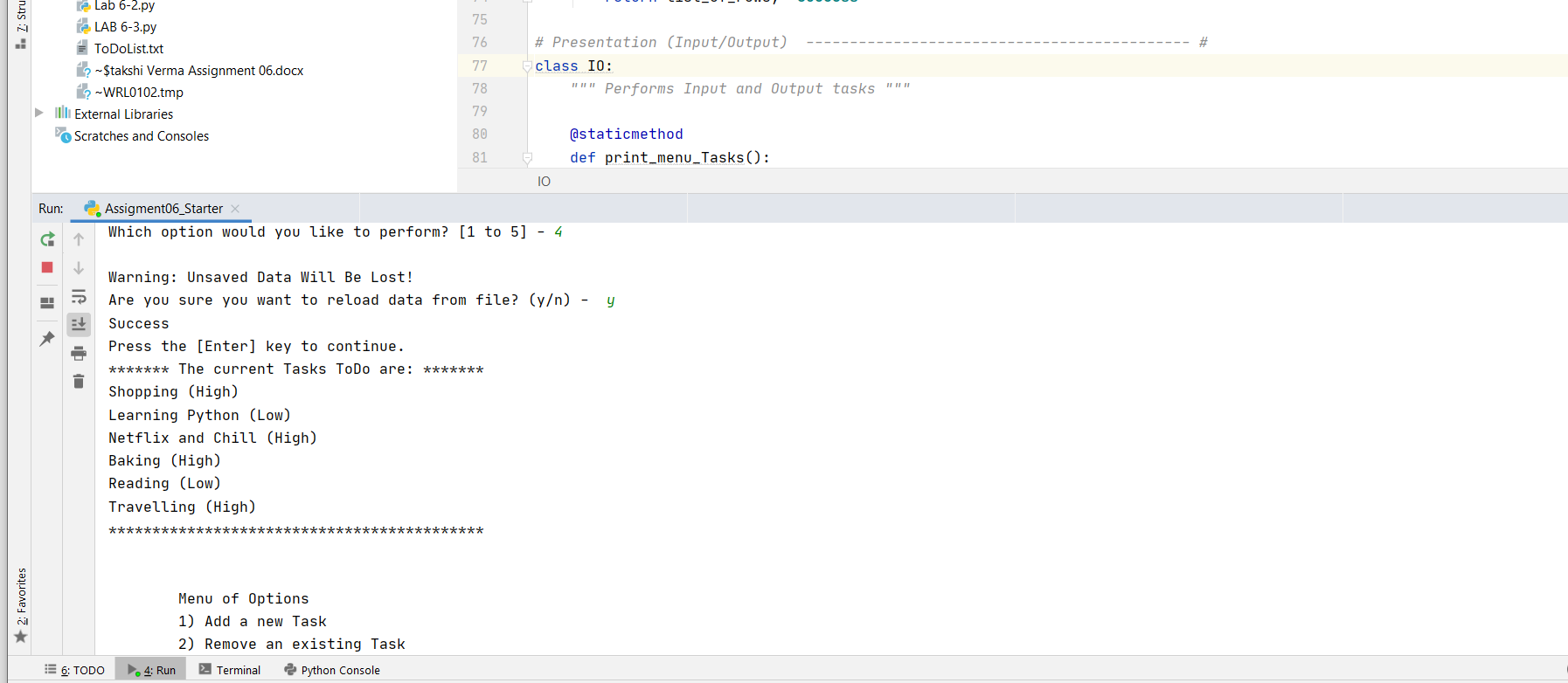
Figure 8: Function to read data from file

*Step 4.5*

If this option was chosen, then the program exits using *break* and prints “GoodBye”.



*Figure 9: Execute script from command prompt*



*Figure 10: Execute script from PyCharm*

**Summary:**

This practical exposure to new python’s programming concept called *Functions* helps me to modularize my code in an efficient way. I have also realized how we can use *Doc Strings* in our program to represent the definition of functions. I have adopted this separation of concerns in my lab work and assignment for advanced professional programming. At last, I have created GitHub webpage that I am going to use in my next work assignments.