Hyo Park

August 16, 2020

Foundations of Programming: Python

Assignment 06

Assignment06\_Starter.py

# Introduction

For module 6, we were asked to expand on previous week’s assignment. We were again provided with a starter file. The file was further broken down into classes and functions, and we were asked to fill in the missing codes. In this paper, I will explain my reasonings and steps I took in order to finish my assignment.

Code Setup

Because we were provided with a starter file with foundations already in place, I did not see a need to come up with a pseudocode this time around. I decided to make the variables more “Pythonic” and rename them without the type pre-fixes. I originally started our class naming the variables following the conventions recommended in the book and the Python PEP. I switched over to using Randal’s way of using pre-fixes around week 3 to familiarize myself with different variable types. Using the pre-fix naming convention was very helpful in getting myself acquainted with the different types. I will use the more Pythonic way going forward as this is the norm in the Python and programming community.

A screenshot of a cell phone

Description automatically generated

Figure 1: Variables of Assignment06\_Starter.py with more Pythonic convention

Main Code

Added to the fact that I am still having hard time grasping the concept of separation of concerns and that the script was now split into numerous functions, the assignment seemed very daunting at first. What gave me the most trouble was initially recognizing that some of the functions were “related” to each other, meaning in order for one to be used the other had to be called first. With Randal’s examples and breaking down the functions one by one, I was able to figure out the correct structure. As the basis of this assignment is transferring my module 5 codes into workable format, I will go in depth to how I figured out how to work with multiple functions.

*add\_data\_to\_list and input\_new\_task\_and\_priority* functions

As add\_data\_to\_list function was the very first “TODO” part of the assignment, I decided to tackle it first. Using a demo test script file, I was able to pass an argument into the function so that a new task and priority was added to a list of dictionaries. In the example below, I am passing strings “wash dishes” and priority “low” with a list of dictionaries called “list\_of\_tasks”, as these are the three required parameters. Variable returned\_data is assigned to the function call and printed so that I am able to validate that it is working properly. The new task and priority are included in the list of dictionaries with word “Success” printed as a tuple.

A screenshot of a computer

Description automatically generated

Figure 2: Testing add\_data\_to\_list function in a demo test file.

While this allowed me to add passed arguments into the list of dictionaries, this did not result in the user actually inputting the data. I then realized add\_data\_to\_list must work in conjunction with another function. There was only one function under the IO class, the input\_new\_task\_and\_priority function, made sense to work as it returned task and priority variables while not passing any arguments. The code for the function itself is very basic, as it just gets the input from the user for the task and variable.

A screen shot of a social media post

Description automatically generated

Figure 3: Function input\_new\_task\_and\_priority in demo test file.

Now that I had working input\_new\_task\_and\_priority function, I had to call it somehow and incorporate it with the add\_data\_to\_list function. Since functions can be called inside another function, I called input\_new\_task\_and\_priority inside add\_data\_to\_list function. While I was now able to enter a new task and priority, printing the result returned an empty key and value in the last dictionary. This was because I was not assigning any variables to the input\_new\_task\_and\_priority function when it was being called. After assigning variables task and priority, I got the expected results. This was outlined very well in the Tic-Tac-Toe example in our textbook (Python Programming for The Absolute Beginner, Michael Dawson).

A screenshot of a computer

Description automatically generated

Figure 4: add\_data\_list and input\_new\_task\_and\_priority working in demo test file. Task and priority are now user added.

After incorporating the two functions, I now had to come back to the main body of the script, where the user inputs their choices. The add\_data\_to\_list function was added under step 4 while. This did not work at first as the two functions are under different classes. After referencing Processor and IO classes in the call, the starter file behaved the same way as in the test file when I chose to enter new task and priority. Following the methods described above, I was able to finish rest of the script.

A screenshot of a cell phone

Description automatically generated  
Figure 5: Class name must be referenced when functions are not residing in the main body of the script.

Code Output

I am happy with how the program turned out. As expected, using the Mac OS command line to test the code created the ToDoList.txt in my user home directory folder as my current working Python directory is located there. Testing on PyCharm, however, the text file was created in the same folder as my .py file.

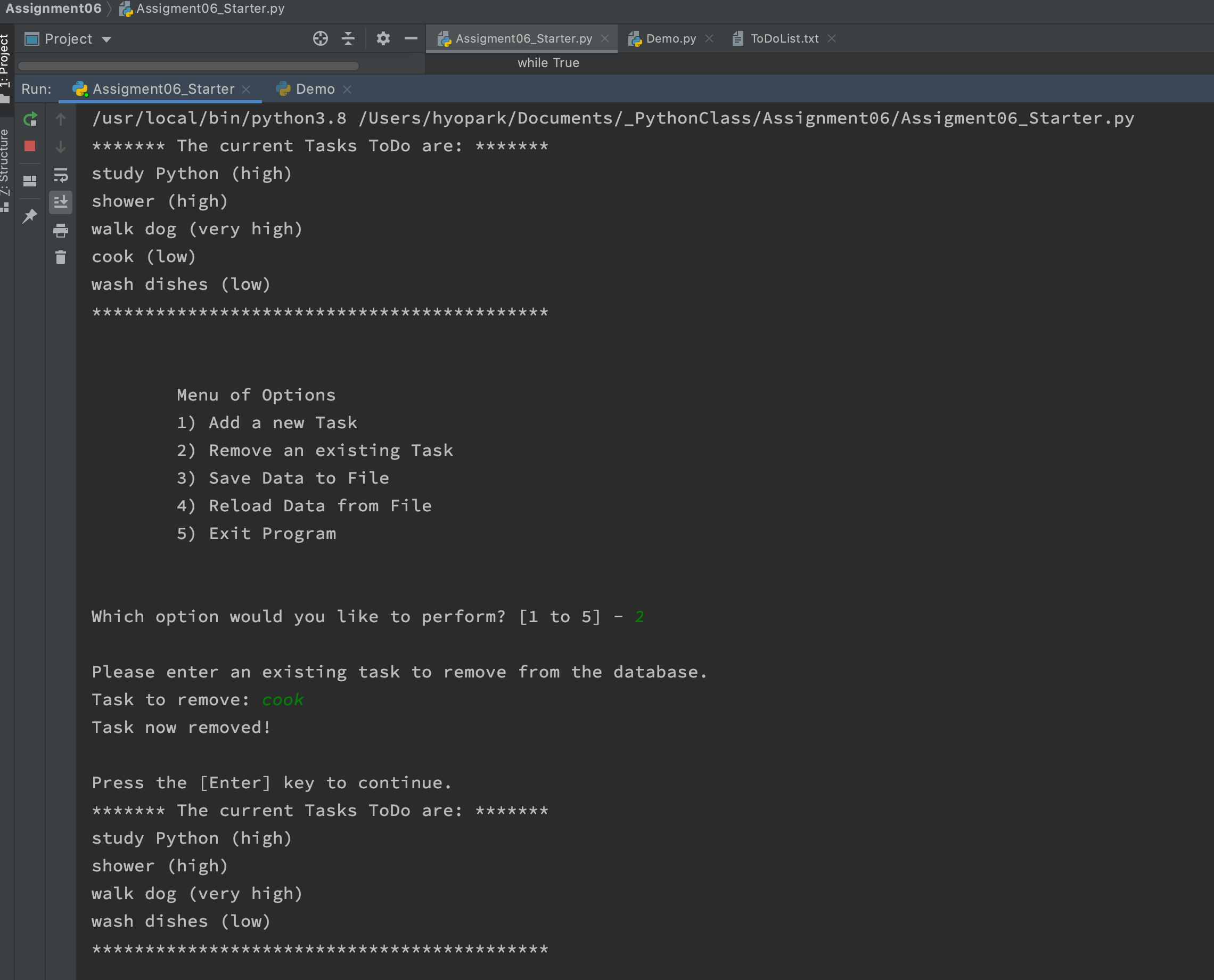


Figure 6: Assignment06\_Starter.py running on PyCharm.

A screenshot of a cell phone

Description automatically generated

Figure 7: Assignment06\_Starter.py running on PyCharm.

A screenshot of a cell phone

Description automatically generated

Figure 8: ToDoList.txt in directory with saved results.

# A screenshot of a cell phone Description automatically generated

Figure 9: Assignment06\_Starter.py running on MacOS command line.

# Summary

As previously mentioned, this assignment seemed very daunting at first. It actually turned out to be fun and exciting once I figured out how the functions co-exist each other. It was nice to see the concepts come together and make the program work as intended. There are some concepts I still don’t fully understand. For example, the variables declared in the beginning of the scrip are getting flagged by PyCharm as global variables. Should we be using local variables inside each function? Why are we using staticmethods when the functions are under classes? I am hoping to figure out these soon.