Geoffrey Sears  
November 13, 2019  
Foundations of Programming, Python  
Assignment 06

# Introduction

The assignment this week was to take the starter file “Assignment06\_Starter.py” and modify it to re-arrange the provided code to perform “Separation of Concerns” between “Data”, “Processing” and “IO” concerns.

This document will walk you through the process I followed to complete the assignment, and end in a visual demonstration of the capabilities of the script after the provided code was placed into functions organized into two classes defined within the original starter script.

# Apply Your Knowledge

When I first approached the “Assignment06\_Starter.py” script, I copied the contents into a new “Assignment06.py” file and began to read the provided code. The structure of what I would need to do was conveyed to me via “Todo” comments which were detected by PyCharm into a ToDo list which could be parsed at the bottom of the interpreter.



I printed out the script so I could review at my desk while highlighting each of these items for follow-up. The screenshot above was taken after I already had finished modifying the script, so the locations and labels have been changed.

Having been informed by the previous developer’s handy commentary I started to work my way down through the original code that did not make use of functions to separate out the specific areas of concern within each block.

The other thing I noted about the structure of the script which had been provided was the introduction of two Classes and a few Functions which gave an indication of the original programmer’s intent in re-organizing this code. The first Class “FileProcessor” had a function “ReadFileDataToList” which would read in the provided text file (provided it resided in the same folder) and pass the items listed within that file into the list “lstTable” as dictionary elements that represented rows.

Following this function was another function entitled “WriteListDataToFile” which contained only a “Pass” statement, indicating that this was a placeholder. I scanned down through the script and found under Step 3.4.b a block of code designed to execute what this function appeared intended to do. I moved the code up to sit under the function definition and then modified it to align to the parameters named in the function definition:

**def** WriteListDataToFile(file\_name, list\_of\_rows):  
 objFile = open(file\_name, **"w"**)  
 **for** row **in** list\_of\_rows:  
 objFile.write(row[**"Task"**]+**','**+row[**"Priority"**]+**'\n'**)  
 objFile.close()

Once this was completed, I inserted a function call in place of this block of code in the original While loop as follows:

FileProcessor.WriteListDataToFile(strFileName,lstTable)

Having completed this step, I changed the “# TODO: Add Code Here” comment to include a “(Done)” to allow me to track my progress through the assignment of modifying all the code. In general each section followed the same steps as above, I described the specific elements below under each Menu Option.

## Option 1: Show Current Data

This option worked out of the box through invoking “IO.OutputMenuItems()” and “IO.InputMenuChoice()”. I made no modifications here.

## Option 2: Add New Item

This option proved to be the most difficult for me. Like the first example (Option 4), it requested that code that had originally been written within the While loop be moved into a function, this step wanted to break the code presented into two functions contained within separate Classes of code. To do this, we would have to make good use of parameters, and still respect the separation of local and global variables.

Initially as I tried to define two separate functions, one to capture the user’s input for Task and Priority (IO.AddItemToList()) and the other (FileProcesor.WriteNewItemToList()) to add that input to the lstTable list variable. Initially, I was having difficulty passing the returned output from the IO.AddItemToList() function to the FileProcessor.WriteNewItemToList() function due to the code not capturing the returned values into a variable to act upon. After considering the problem, I decided I would assemble the dictionary object within the IO.AddItemToList() function and then pass that object as a returned value to a dictionary variable to be used as a parameter into the FileProcessor.WriteNewItemToDoList() function. I am conscious that SoC would dictate that the assembling of the dictionary variable would happen internal to the second function, but this seemed cleaner to me.

## Option 3: Remove an existing item

Initially when I tried to modify this section, I ran into problems concerning the placement of the following rows:

blnItemRemoved = **False** *# Create a boolean Flag for loop*intRowNumber = 0 *# Create a counter to identify the current dictionary row in the loop*

Initially, I left them in the code block within the Wile loop, but this presented a problem concerning Global and Local Variables. If they resided outside of the FileProcessor.RemoveItemFromList() function, then the function when called would error out on the local variable binItemRemoved not having been assigned. I relocated them to be within the function and things worked better.

## Option 4: Save Data to File

This was described above in the “Apply Your Knowledge” section. I moved the provided code for writing the lstTable list to a text file to its own function FileProcessor.WriteListDataToFile() and modified the variables to conform to the parameters supplied with that function call. This worked as designed so didn’t require much other modification.

## Option 5: Reload Data From File

I did not have to make any changes to this section of code.

## Option 6: Exit Program

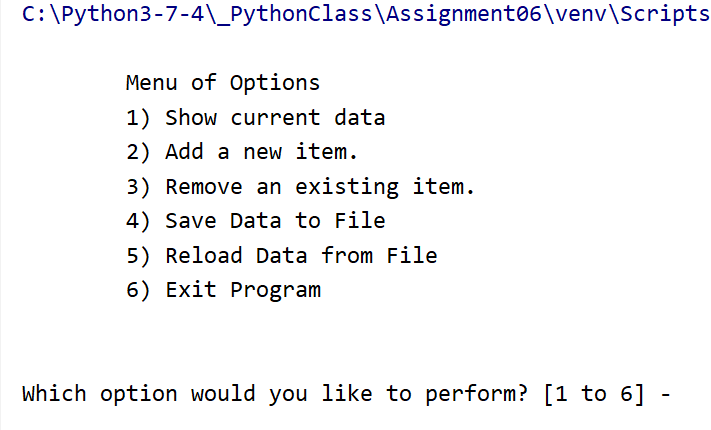
I did not have to make any changes to this section of code.

# Demonstrating the Script Worked

Below are screen captures illustrating the code being run through its paces via each menu option. I conducted my testing in PyCharm and in Windows Command Prompt. In order to get the program to work initially, I had to create a “ToDoList.txt” file for it to manage.

## Pycharm Testing:

On initial execution of the script, I was presented with a menu of six choices:



I selected Choice 1 to demonstrate that the program had read into memory the contents of the ToDoList.txt file which still contained some data from the previous week’s assignment as well as new data I had entered in during this assignment.

Which option would you like to perform? [1 to 6] - 1

\*\*\*\*\*\*\* The current items ToDo are: \*\*\*\*\*\*\*

sleep (3)

review (100)

Pontificate (high)

lol (low)

postulate (low)

I then proceeded to option “2” which permits adding items to the list. After entering a task and priority pair of values, it processed them and added them to a revised list which was outputted for consumption:

Which option would you like to perform? [1 to 6] - 2

What is the task? - postulate

What is the priority? [high|low] - low

\*\*\*\*\*\*\* The current items ToDo are: \*\*\*\*\*\*\*

Write (1)

sleep (3)

review (100)

Pontificate (high)

lol (low)

postulate (low)

Option 3 permits you to remove an item from the list. Because it also had error detection written into it, I wanted to test both the successful and error results of this selection.

Which option would you like to perform? [1 to 6] - 3

Which TASK would you like removed? - Write

The task was removed.

\*\*\*\*\*\*\* The current items ToDo are: \*\*\*\*\*\*\*

sleep (3)

review (100)

Pontificate (high)

lol (low)

postulate (low)

Which option would you like to perform? [1 to 6] - 3

Which TASK would you like removed? - laugh

I'm sorry, but I could not find that task.

\*\*\*\*\*\*\* The current items ToDo are: \*\*\*\*\*\*\*

sleep (3)

review (100)

Pontificate (high)

lol (low)

postulate (low)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Then Option 4 writes the contents of the table in memory (lstTable) to the text file (“ToDoList.txt”):

Which option would you like to perform? [1 to 6] - 4

\*\*\*\*\*\*\* The current items ToDo are: \*\*\*\*\*\*\*

sleep (3)

review (100)

Pontificate (high)

lol (low)

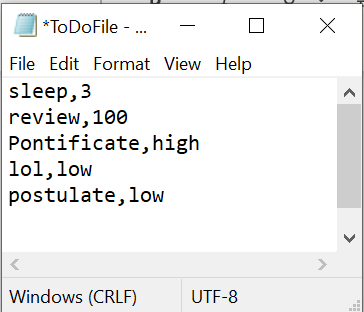
postulate (low)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Save this data to file? (y/n) - y

Data saved to file! Press the [Enter] key to return to menu.

I captured a screen shot of the text file after this modification:



Option 5 reads what is stored in the text file back into memory, overwriting what you are currently working with within the lstTable variable.

Which option would you like to perform? [1 to 6] - 5

Warning: This will replace all unsaved changes. Data loss may occur!

Reload file data without saving? [y/n] - y

\*\*\*\*\*\*\* The current items ToDo are: \*\*\*\*\*\*\*

sleep (3)

review (100)

Pontificate (high)

lol (low)

postulate (low)

Finally, Option 6 exits out of the program as expected:

Which option would you like to perform? [1 to 6] - 6

Process finished with exit code 0

## Microsoft Windows command prompt

Executing the Python script from the command prompt:

c:\Python3-7-4\\_PythonClass\Assignment06>python.exe Assignment06.py

Menu of Options

1) Show current data

2) Add a new item.

3) Remove an existing item.

4) Save Data to File

5) Reload Data from File

6) Exit Program

Which option would you like to perform? [1 to 6]

Option 1:

Which option would you like to perform? [1 to 6] - 1

\*\*\*\*\*\*\* The current items ToDo are: \*\*\*\*\*\*\*

sleep (3)

review (100)

Pontificate (high)

lol (low)

postulate (low)

Option 2:

Which option would you like to perform? [1 to 6] - 2

What is the task? - Walk Dog

What is the priority? [high|low] - high

\*\*\*\*\*\*\* The current items ToDo are: \*\*\*\*\*\*\*

sleep (3)

review (100)

Pontificate (high)

lol (low)

postulate (low)

Walk Dog (high)

Option 3:

Which option would you like to perform? [1 to 6] - 3

Which TASK would you like removed? - sleep

The task was removed.

\*\*\*\*\*\*\* The current items ToDo are: \*\*\*\*\*\*\*

review (100)

Pontificate (high)

lol (low)

postulate (low)

Walk Dog (high)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Menu of Options

1) Show current data

2) Add a new item.

3) Remove an existing item.

4) Save Data to File

5) Reload Data from File

6) Exit Program

Which option would you like to perform? [1 to 6] - 3

Which TASK would you like removed? - sing

I'm sorry, but I could not find that task.

Option 4:

Which option would you like to perform? [1 to 6] - 4

\*\*\*\*\*\*\* The current items ToDo are: \*\*\*\*\*\*\*

review (100)

Pontificate (high)

lol (low)

postulate (low)

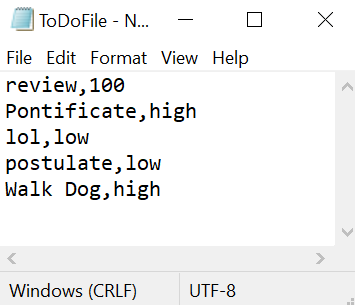
Walk Dog (high)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Save this data to file? (y/n) - y

Data saved to file! Press the [Enter] key to return to menu.

Screenshot of the ToDoList.txt file to compare with the above:



Option 5:

Which option would you like to perform? [1 to 6] - 5

Warning: This will replace all unsaved changes. Data loss may occur!

Reload file data without saving? [y/n] - y

\*\*\*\*\*\*\* The current items ToDo are: \*\*\*\*\*\*\*

review (100)

Pontificate (high)

lol (low)

postulate (low)

Walk Dog (high)

Option 6:

Which option would you like to perform? [1 to 6] - 6

c:\Python3-7-4\\_PythonClass\Assignment06>

# Summary

The assignment this week involved taking a provided script and revising it to incorporate the use of Classes and Function as well as practicing Separation of Concerns. Most of the modifications made to the script were to move blocks of code out of the central While loop of the script and place them into Functions which allow for better separation of concerns, and modeled the passing of parameters and returned values from functions which are necessary for more complicated program solutions.

This week’s assignment highlighted the way a programmer’s intent can be conveyed through the use of TODO comments, which Pycharm helpfully collects as a list within the program for review by the next person to modify the script. I was able to make use of this feature this week to keep track of the items which were passed to me by the instructor and ensure that all the items which were requested were completed. We also made use of docstrings within our functions to provide guidance to subsequent developers as to the purpose of the defined parameters, and return values.

The organization of our script also took a step forward with the inclusion of Classes and Functions. This permitted separating the different functionality of our code into functions, and then grouping them into separate classes in alignment with their functionality (Data, Processing, Presentation). Doing this highlighted a number of potential issues to focus upon.

* The use of global and local variables. As code is shifted inside of a function, a separate list of local variables is needed to support the execution of that code. This prevents the function from polluting global variables with unintended assignment, and simplifies future troubleshooting.
* The passing of values in the form of parameters into the function and return of values requires the programmer to think carefully about what distinct components of data are needed to meet their programming goals for the function. This also reinforces the programmer reducing the function to the most specific purpose possible for that block of code.
* When the function completes its block of code, the returned value needs to be captured into a variable to be used elsewhere in the program. As referenced in the first bullet, the variables used within the function, even if they share the same name as a global variable, do not share the same namespace, and will not influence global variables unless specifically designated to do so.

Having completed the requested changes to the program, I was able to test it successfully and log the results in the “Demonstrating the Script Works” section above.