Chelsea A. Wright

May 30, 2019

Python – Fundamentals of Programming

IT FDN 100 A

Assignment 06

GitHub: <https://github.com/WorldAtLarge/IntroToProg-Python>

Interactive To Do List - Functions

# Introduction

This document describes creating a program using encapsulation, methods, and classes. in PyCharm. The code described herein is built off of the interactive ToDo List program from *Assignment 05*. The program written for Assignment 06 uses more detailed organization, which is necessary for more complex programs than the class has written thus far in this course.

# Starting the Assignment

I decided to reuse sections of code from *Assignment 05* in *Assignment 06*, but for testing and clarity reasons, I ported small sections of code over to the new program as I went along.

My first step was to revise my code from Assignment 05, based on feedback received from Andrew Foster during grading and further insight gained from Professor Root’s explanation of his answer for Assignment 05. I will not describe all of the revisions for brevity’s sake.

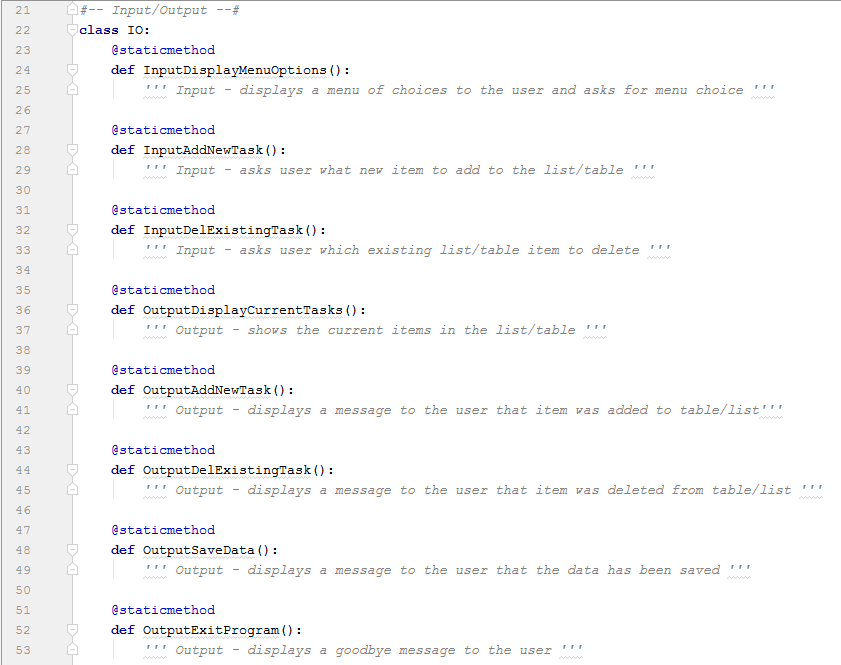
I created a new project: Assignment06and new Python file: Assignment06, in PyCharm. I copied and pasted the pseudocode provided by Professor Root in Assignment06.docx. I then divided my code into four sections: Data, Input/Output, Processing, and Main. I assigned a class to each of the Input/Output, Processing, and Main sections.

The next steps in defining structure are described in the next few sections. I knew that I would be going back through the sections and making adjustments, corrections and additions, but I felt this was a good way to get my code organized, especially for cutting and pasting from my Assignment 05 script. I did not identify any parameters during this initial stage of organizing this program.

# Input/Output Section Outline

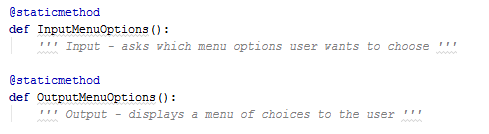
The next step of organization was to define functions for each of the Input and Output operations of the program. This was a very helpful exercise to start thinking about how to break up the code from Assignment 05. I defined functions for each of the input and output functions of the ToDo list program and created descriptive document strings (docstrings) for each function to describe the purpose of the function. (Figure 1) In hindsight, the docstrings used in this assignment are probably overcomplicated by using “Input” and “Output” as introductions.

Figure 1: Input/Output Section Outline



Note: Menu option inputs and outputs were quickly revised/edited after the above screenshot was taken. (Figure 2)

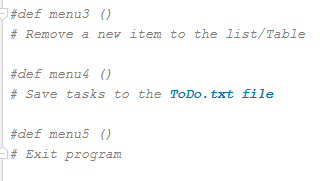
Figure 2: Revised Function Definitions - Menu Options



## A Rough Draft Structure

Perhaps I should note that the structure shown above wasn’t technically my first step in defining an outline for this script. I originally just copied and modified high-level descriptions for all of the program functions from the code blocks from my Assignment 05. This provided a sort of first step of a rough draft for the structure of the program. I did not write code directly into this rough draft structure, but I used it to get my mind organized to build the program structure in more detail. Some examples of these original lines are provided in Figure 3.

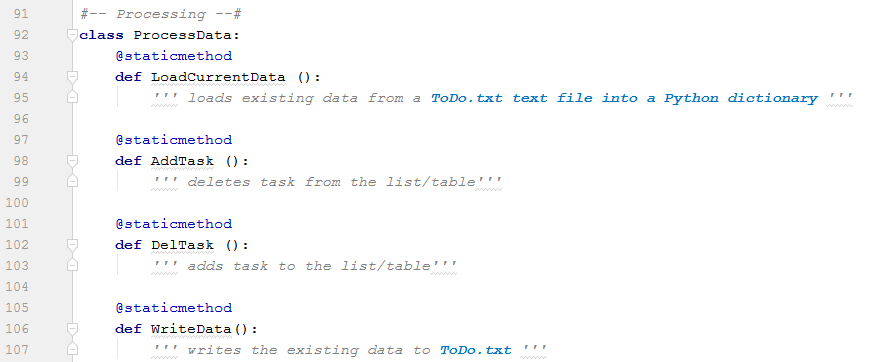
Figure 3: Early Rough Draft Outline



# Data Processing Section Outline

I next created a high-level outline for all of the data processing functions. (Figure 4)

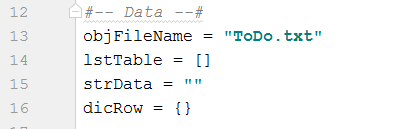
Figure : Data Processing Section Outline



# Data Section

I reused the *ToDo* text file from Assignment 05, and copied it into my Assignment 06 project folder. I then copied and pasted the data section from Assignment 05.

Figure : Data Section

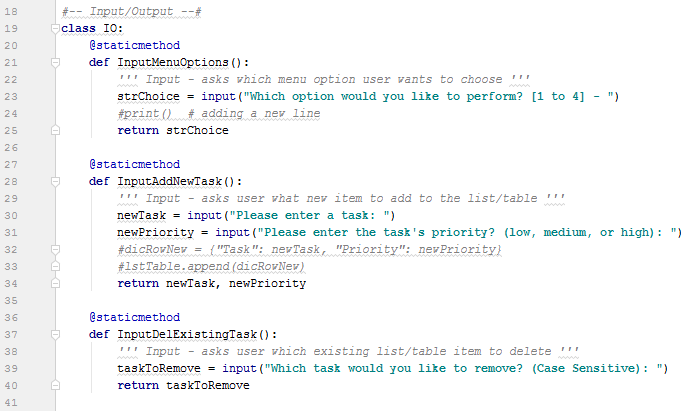


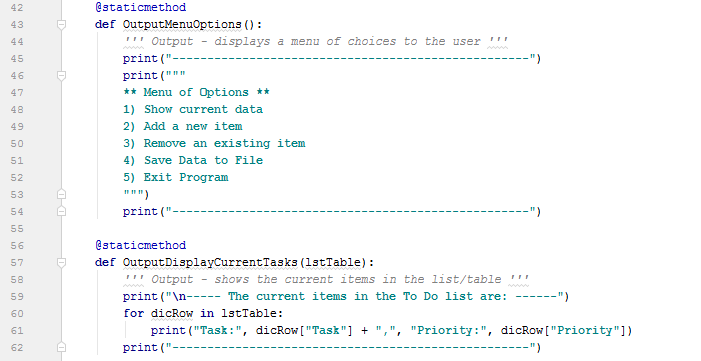
Because the objects shown in Figure 5 are used so much throughout the program, I decided to make these global variables so they can be accessed from within the various scopes of the program.

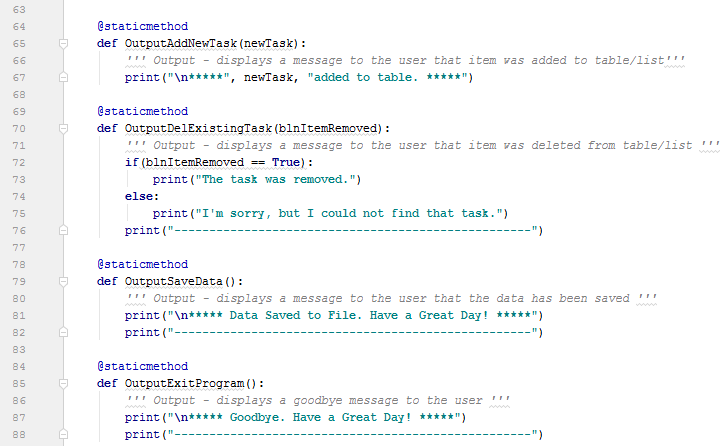
# Filling in the Program Structure

The next step was copying and pasting code lines from Assignment 05 into the (hopefully) appropriate places in the new program. My strategy was to follow the program logic as it was written for Assignment 05 to determine the parameters that each function would need to be passed in and returned throughout the program, then defining the values of the parameters with keyword arguments. Most of the code was simply copied and pasted from Assignment 05. I went through each of the program’s operations in order, following the command through each of the Input, Processing, and Output components. The Input/Output section is shown in the three sections of Figure 6.

Figure : Input/Output Section



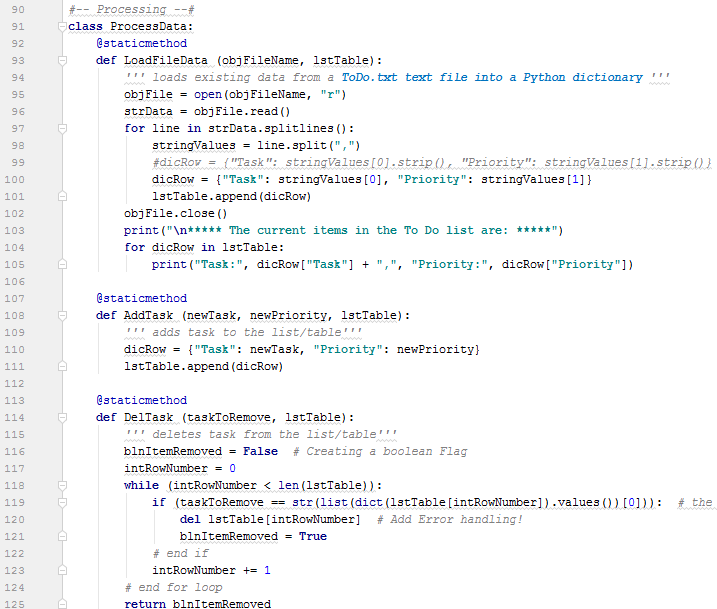


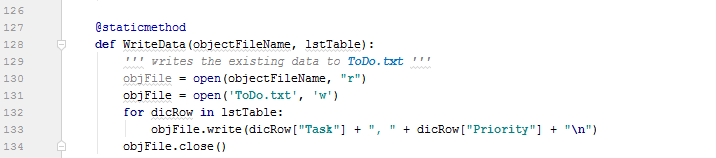


# Processing Section

I used the same technique for the Processing section (Figure 7); often going back and forth between functions in the two sections to ensure that the right parameters were being passed in and returned.

Figure : Processing Section

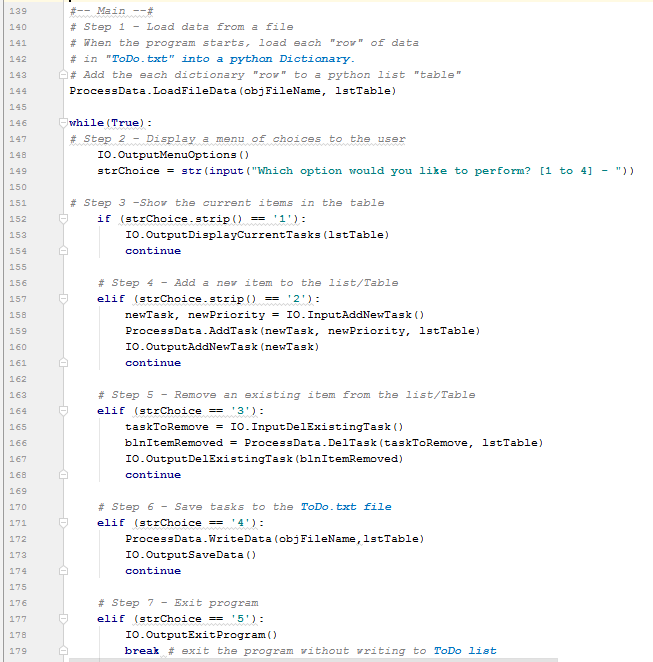




# Main Section

The last major step was to create the main section where the functions created in the Input/Output and Processing section would be called. I should note that I created static methods within classes throughout the script so that they could be called as such: *IO.OutputMenuOptions(*). This made it very clear and simple to follow the program logic. (Figure 8)

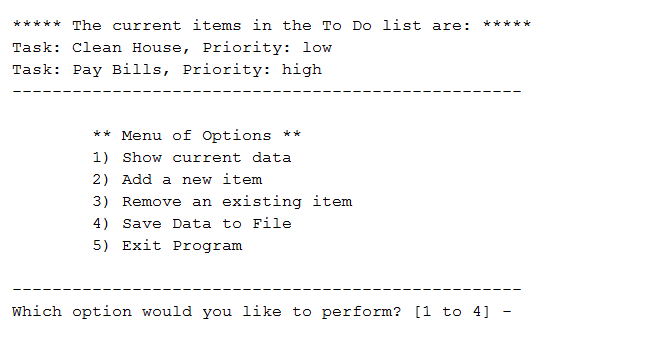
Figure : Main Section



# Running the Program

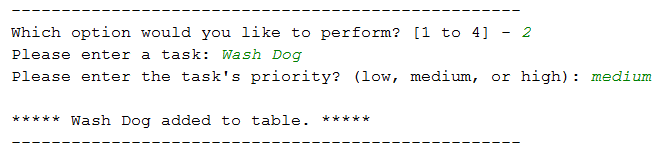
Just as in Assignment 05, when the program starts, the first operation is to load data from the ToDo.txt file and load each line of data into a Python list object. The program then proceeds to display the current items in the To Do list and a list of menu options to the user. For presentation, the program presents the key value pairs of each item in the dictionary. (Figure 9)

Figure : Starting the Program



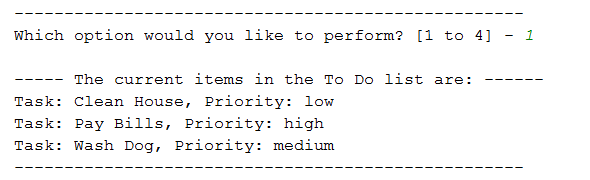
Selecting option 2 from the menu, the user can add an item to the To Do list. In the example shown in Figure 10, we’ve added the task ‘Wash Dog’ with a priority of ‘medium’ to the list object. The first function called asks the user to define values for a task and a priority, which are then returned to be passed to other functions. The second function passes these arguments to the list object, and the third function accepts the value for the new task and displays a message of confirmation to the user.

Figure : Adding a Task to the To Do List



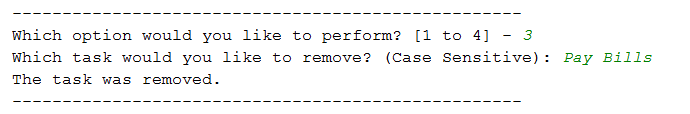
Selecting 1 from the menu, we can see that the new values have been passed to the list object. (Figure 11)

Figure : New Priority and New Value Passed to To Do List



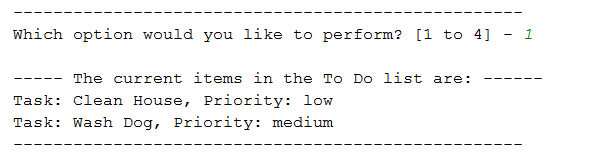
Selecting 3 from the menu, the user can choose to remove an task from the To Do list. As shown in the example in Figure 12, the user has chosen to remove ‘Pay Bills’ from the to do list.

Figure : Removing a Task from the To Do List



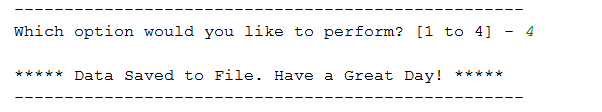
As in Assignment 05, the function removes the entire dictionary row containing ‘Pay Bills’ (Figure 13).

Figure : Current To Do List



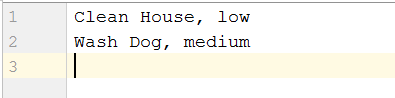
The next menu option is saving the current data to the ToDo.txt file. When option 4 is selected, the first function opens the ToDo.txt file and saves the current items in the list object and the second function (output) displays a message to the user. (Figure 14)

Figure : Saving Current Data to ToDo.txt



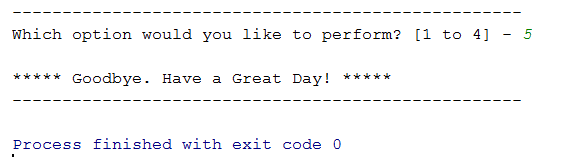
Opening the ToDo.txt file, we can see that the current data in the list object have been saved to the file. (Figure 15)

Figure : Data Has Been Saved to ToDo.txt



The last menu item (option 5) exits the program and displays a menu to the user. (Figure 16)

Figure : Exiting the Program



# Summary

In conclusion, this program is very much like Assignment 05, but with this program we have created functions and used encapsulation to organize the program. This technique will be necessary for creating more complex programs.

# Bibliography

Dawson, M. (2010). Python Programming for the Absolute Beginner, Third Edition. In M. Dawson, *Python Programming for the Absolute Beginner, Third Edition* (p. 3). Boston: Course Technology, a part of Cengage Learning.

Root, R. (2019, May). *\_Mod6PythonProgrammingNotes.docx.* Retrieved from UW Canvas: IT FDN 100 AModules.

Root, R. (2019, May).  *Assignment06.docx* Retrieved from UW Canvas: IT FDN 100 AModules.