Kelsey Kawaguchi

Nov. 6, 2019

IT FDN 100

Assignment 05

Adding Items to a To Do List Script

Introduction

This paper explains the logic on how I completed the To Do List Script assignment, which allows the user to compile a “To Do” list based on task to be achieved and the priority level of the task. This code would be valuable to someone who needs to build a To Do list and prioritize each task. In the code, the user has the ability to add, delete, save, and print the list to an external text file.

Methodology

Per Figure 1, the first part of the code identifies what variables will be used in the program.

The next step, I combined both the process and the Input/Output sections into one section. The first part of this section prompts the user to make a selection. A while loop is created to loop through and allow the data to perform any task before the program is killed. The next item tasks the user to select an option to be performed.

If 1 was selected, then the existing To Do list items are printed to the user. After printing the list out, the user is prompted with the menu again.

If 2 is then selected, the user can add a task and priority number to which the task needs to be completed. At this point, the file is opened into “append” mode. For every task and priority entered, a dictionary row is created with the strID (or priority) associated with the variable of “Priority” and the strTask is associated with the variable “Task”. Lastly, the list of rows is appended with the new dictionary row input by the user.

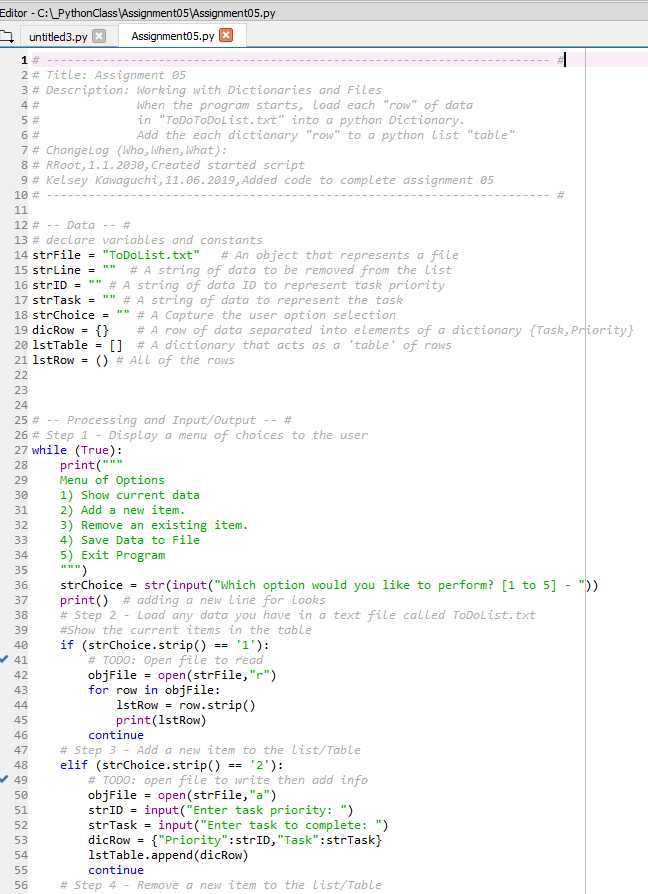


Figure 1 – “To Do” List Code Part 1

If the user selects option 3, then the user is prompted with the option to remove a task or priority. When entered, the strFile will open and have both read and write access and will be called out as objFile. New\_objFile will act as a variable for the each line or row in the objFile (as identified by readlines). Next, the objFile starts from the beginning of the file and for each line identified as new\_objFile, if the strLine isn’t located, then print the line in the file. The last feature truncates the file to remove any blank lines.

If 4 is selected, then the files are saved or “appended” based on the objFile Priorities and Tasks as identified in the dictionary row. The file is then closed.

Lastly, if 5 is selected then the user will exit the while loop and program.

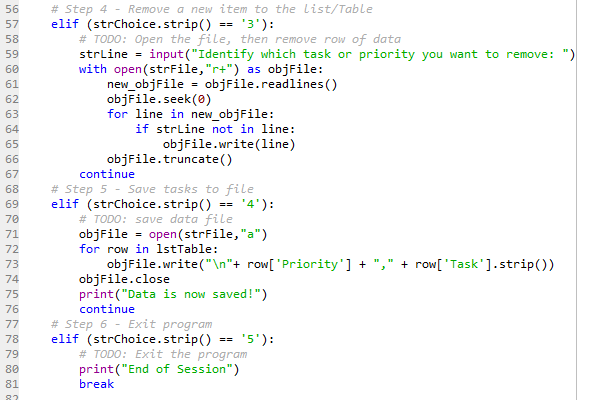


Figure – “To Do” List Code Part 2

Command Results

In the command window shown in Figure 3 below, after the user runs the program, they are prompted to select an option. If a text file is not already created, the user will need to first input some data into a text file, and need to select option 2 as shown below.

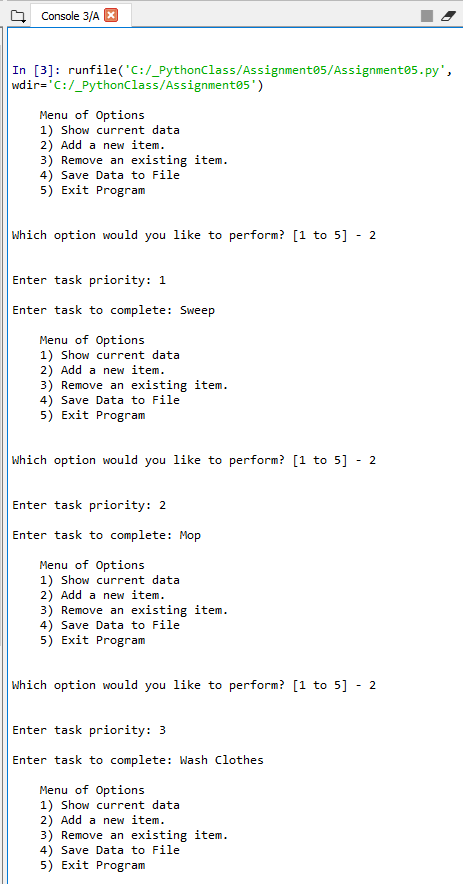


Figure 3 – Spyder Python Console

Once data is entered, the user can now save the data. By selecting option 4. To view all of the data saved, the user then can input option 1 as shown in figure 4. Next, the user can also input option 3 if they want to delete a row and the results can be seen in the Figure below.

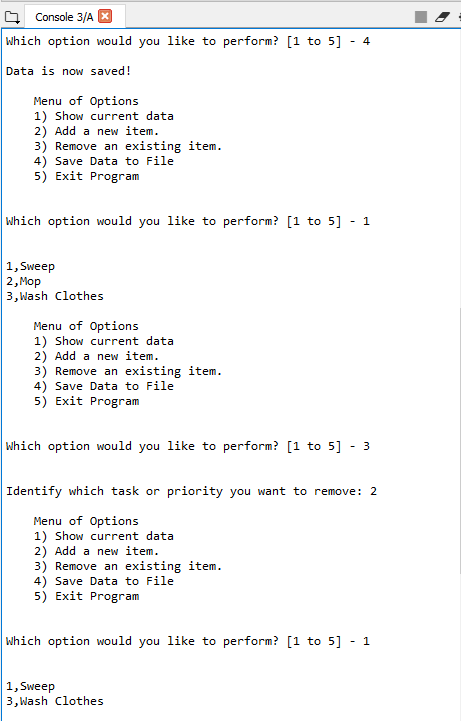


Figure – Spyder Python Console 2

Lastly, Figure 5 shows that the program successfully ends by inputting option 5.

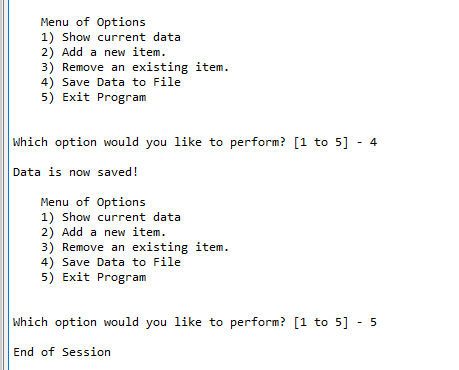


Figure – Spyder Python Console 3

Figure 6 displays that the output text file is located in the same file as the program.

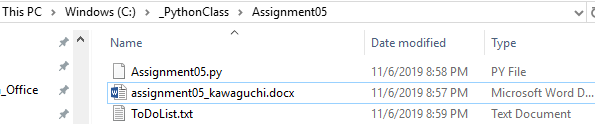


Figure 6: Proof of the Text File location

Figure 7 shows the data stored in the textfile. The data matches what was seen last in the command window, so the program was successful!

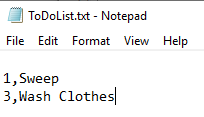


Figure 7: To Do List Data

Summary

In Conclusions, I was able to successfully create a program that allows a user to build a list of Tasks and priorities using the dictionary and list functions. The program was then able to successfully add, remove, print, and save the data input by the user.