Jonathan Ou

May 18, 2021

Foundations of Programming: Python

Assignment 06

Todo List Functions Documentation

# Introduction

In this exercise, this script utilizes the same operations as the script in Assignment 05, but this time, the operations are established into functions which are called in the if cases. This allows for a cleaner script in the main body of the script. Having the functions separated out in the beginning also allows for multiple instances of the same function to be called in the main body without taking too many lines.

# Data Declaration

Figure 1 shows the variable callout that will be used in the rest of the script. Calling it ahead of time allows the script to store a blank value of a data type for its first use. Knowns such as the file name is populated rather than given a blank value.

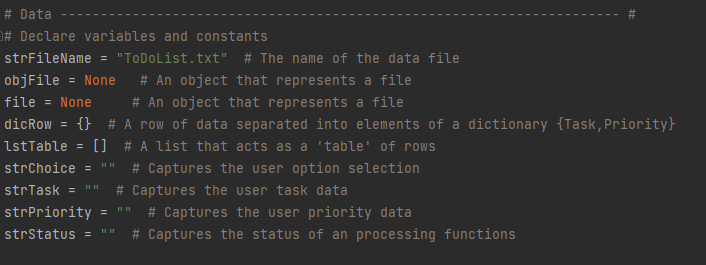


Figure : Data Declare

# Processing Functions

The first set of functions is set behind a class which is used to further identify the function for later use. The functions stored in this class is shown in Figures 2 and 3. Figure 2 has the function which reads a file and stores the values into a dictionary labelled with task and priority. This function takes in two inputs, the file\_name which is the file that the information is drawing from and list\_of\_rows, which is the variable in which the values will be stored. This was utilized the same way as in Assignment 05. The append function was used to store the rows of information into the list, list\_of\_rows. The next function takes in 3 inputs for the act of adding data into some list. The inputs of task and priority will be added into the variable list\_of\_rows, which are stored as a dictionary with labels of task and priority.

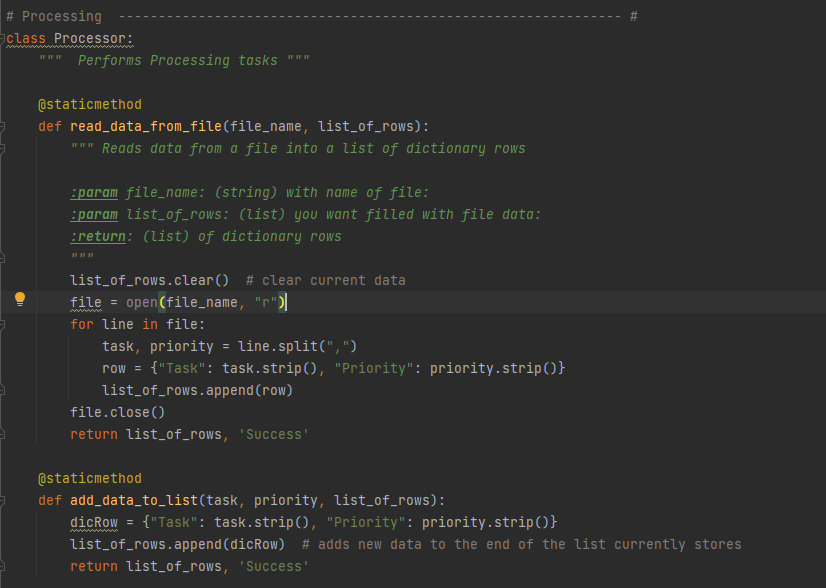


Figure : Processing Functions 1

The next set of functions in Figure 3 go over removing data from a list as well as writing data from a list into a file. The function to remove data iterates through the rows of data in list\_of\_rows and then checks if the input task is found in any of the values. If the input task is found, then the row in which that value is found is then removed from the variable. Writing data to file function takes in two inputs which are the file\_name which is the file which the information will be stored and the variable list\_of\_rows, which is the information that will be stored. In this function, the file is open with the open function. The variable, list\_of\_rows, is scanned per row and extracting the dictionary values into a string.

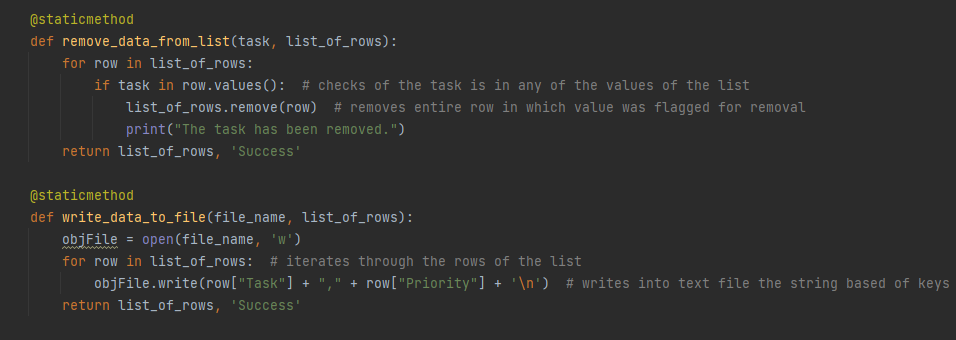


Figure : Processing Functions 2

# Input/Output Functions

Figures 4-6 show the input/output functions of the script which relate to displaying information to the command window of either Pycharm or CMD for the purpose of this script. These functions are stored into the class IO which will be referenced later in the main body. The first two functions in Figure 4 display the menu of options to the user and collects an input from the user for later use. The menu display is simply the print function with the available options. The input collection function utilizes the input function, no surprise, with a string that informs the user that there are options from 1 to 5.

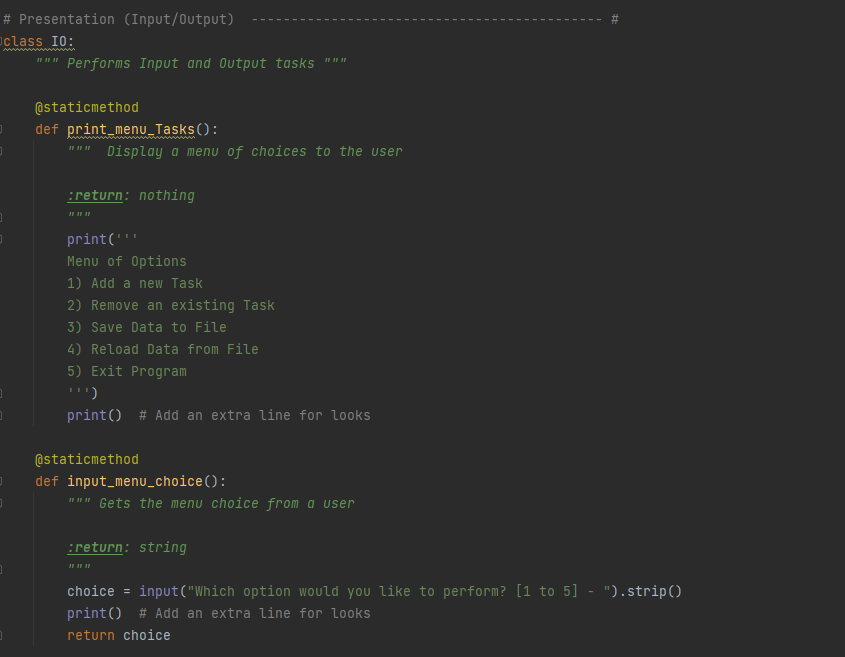


Figure : Input/Output Functions 1

Figure 5 has functions printing the current dictionary for the user, collects an input from the user of yes/no, and a pause function which asks the user to press enter before continuing with the code. The print task function utilizes the print function as well as a for loop which cycles through the rows inside the variable, list\_of\_rows, and then prints the values into a string after taking the values from the dictionary. The last function in the script simply pauses the script with an input function. Since the input is not utilized, it is not assigned a variable and only has a string which prompts the user to press enter before continuing.

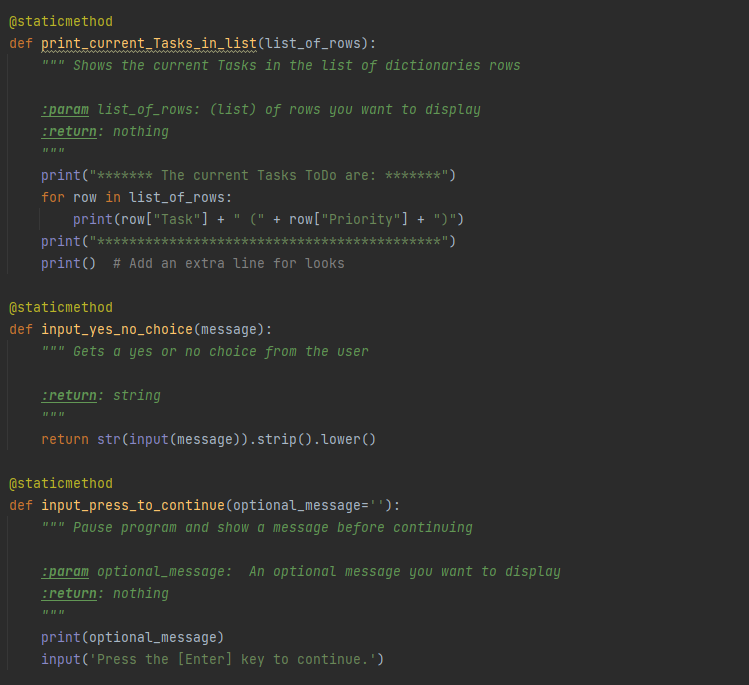


Figure : Input/Output Functions 2

Figure 6 captures the last set of functions in the Input/Output Class. The last two functions are functions which call the user to input either a new task and priority which will be used to adding to the list or call the user to input a task which will be removed from the list. Both these functions utilize the input function and then stores the input into a variable which can be called later.

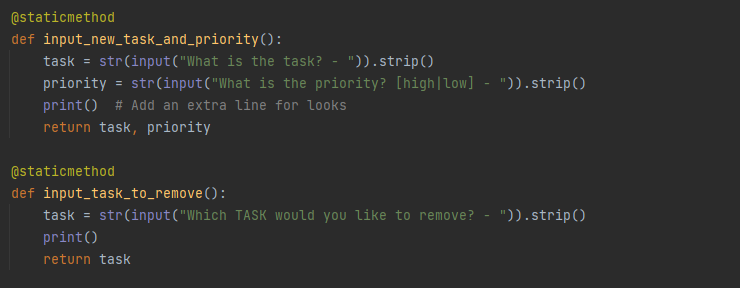


Figure : Input/Output Functions 3

# Main Body of Script

The next Figures show the main script which calls up all the functions which were assembled in the previous figures. The first step is to utilize the read\_data\_from\_file function to pull in information into Python’s memory for the first time. Step 3 then shows the current tasks which are in memory after being pulled in from the file, prints the menu of options, and then collects the user’s input for option choice. Step 4 goes through the function callouts which are associated with all of the potential options for the user. When selecting 1, the script will prompt the user for two inputs of task and priority which will then be added into the list in the memory. All functions will be given the function press\_enter\_to\_continue which pauses the script at the conclusion of all option choices. Option 2 allows the user to remove a task via input from the list. This is accommodated through the two functions Input\_task\_to\_remove and Remove\_tasks\_from\_list.

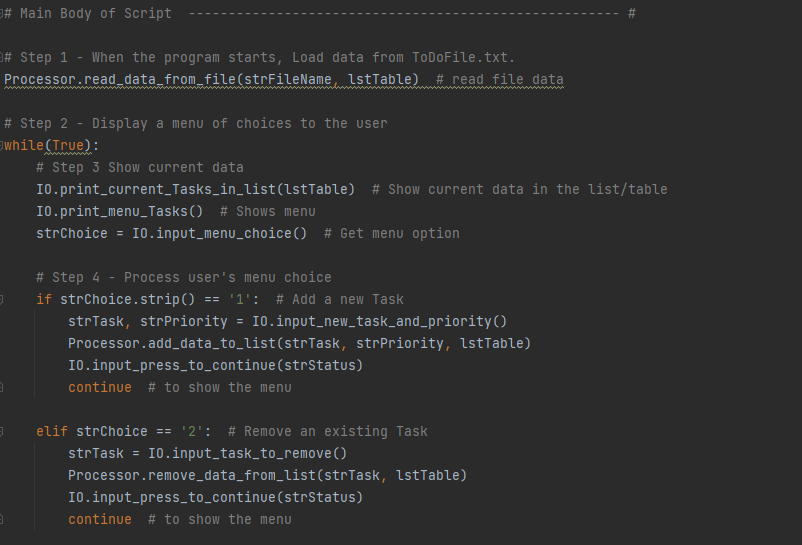


Figure : Main Script 1

Figure 8 shows the functions which will correspond with menu choices 3 and 4. Choice 3 will allow the user to save the current variable lstTable into a file. There is an input step which will allow the user to exit this save feature without saving. Choice 4 will allow the user to reload the data from file. Again there is an input step which will allow the user to exit this reload feature without loading to preserve values in the current memory.

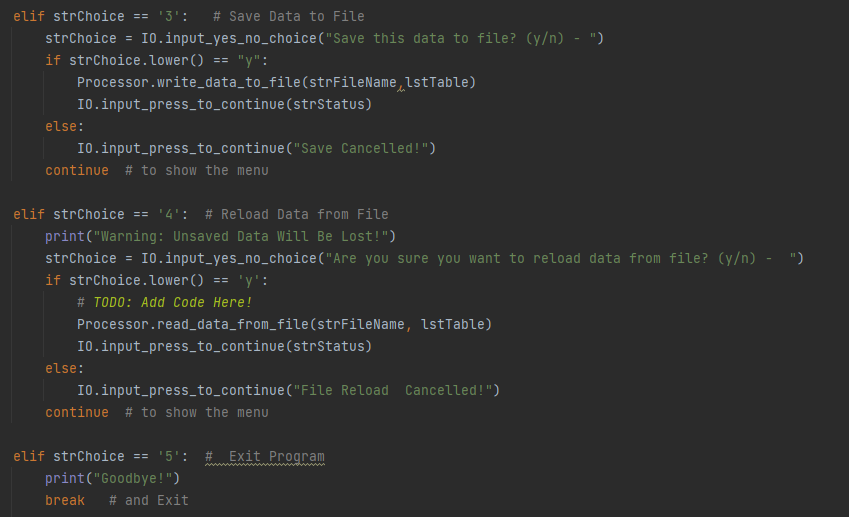


Figure : Main Script 2

# Script Output

The following figures show the output of the script in PyCharm. As shown in Figure 9, the list of tasks get an additional value with the use of choice 1.

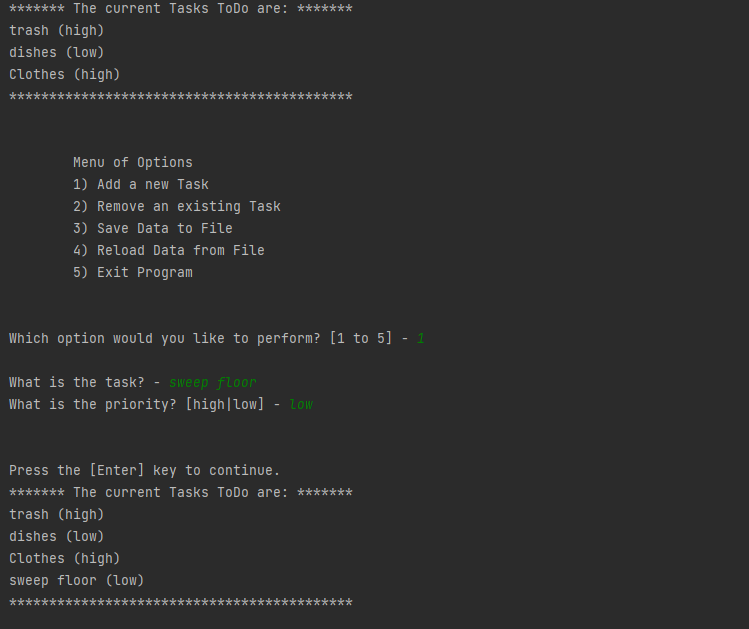


Figure : Pycharm Output 1

Figure 10 shows the output after selecting choice 2. In this step, the previous task of Clothes with high priority has been removed.

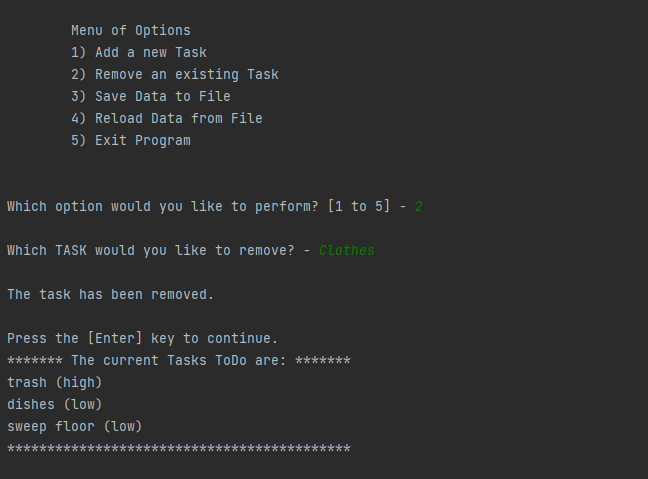


Figure : Pycharm Output 2

Figure 11 shows the output for choice 3 and 4. Choice 3 simply saves the data and provide a string display acknowledging the act. Choice 4 reloads the data in the file but since the file was recently saved. The data in memory will be the same as what was shown after selecting choice 3.

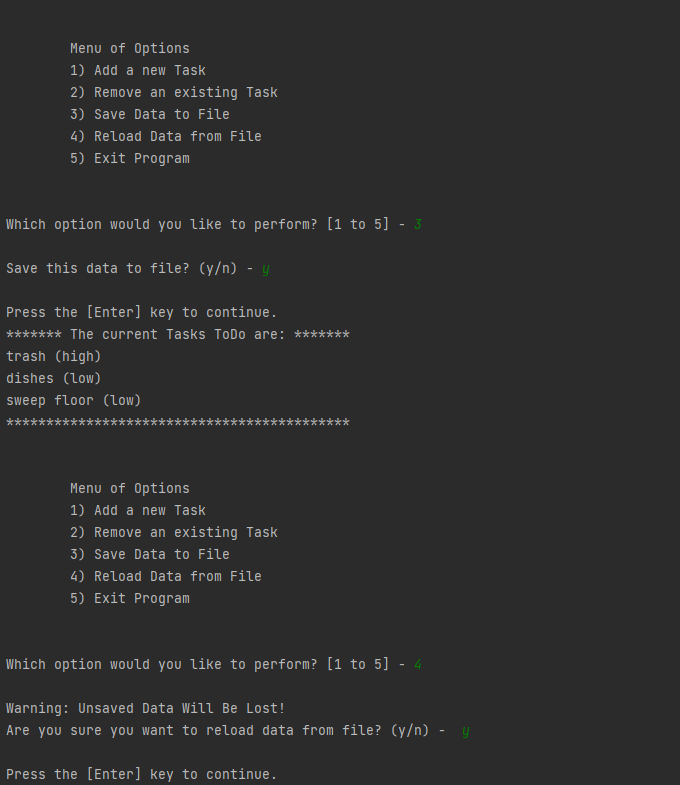


Figure : Pycharm Output 3

Figure 12 shows the output after selecting option 5. This ends the script with a string wishing the user goodbye.

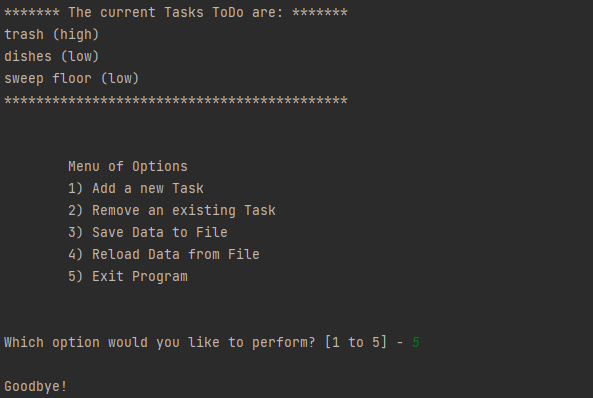


Figure : Pycharm Output 4

The following figures show the output in CMD as opposed to Pycharm. Figure 13 shows the output for selecting choices 1 and 2.

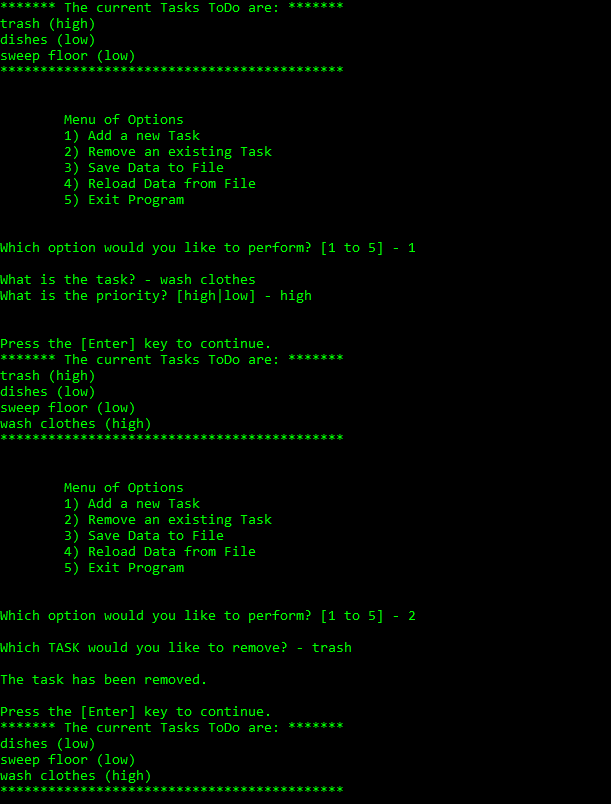


Figure : CMD Output 1

Figure 14 shows the output for options 3 and 4. In contrast to Figure 11 in Pycharm, Figure 14 shows the output when electing not to reload the data.

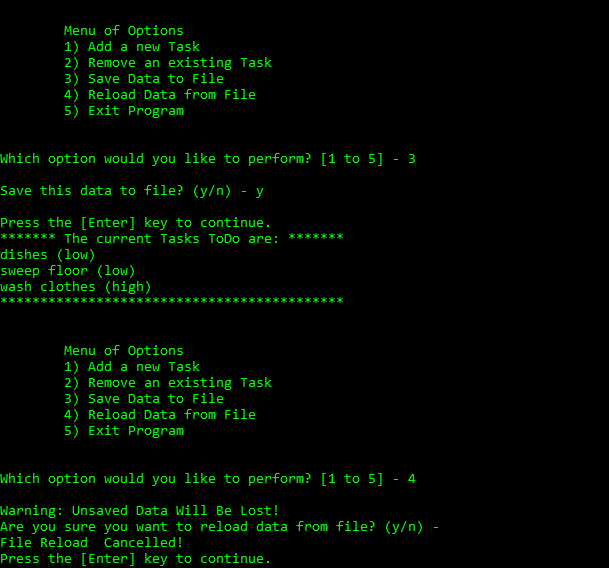


Figure : CMD Output 2

Lastly Figure 15 shows the output for selecting yes for option 4 and the output for option 5. Like in the last, option 5 only wishes the user goodbye!

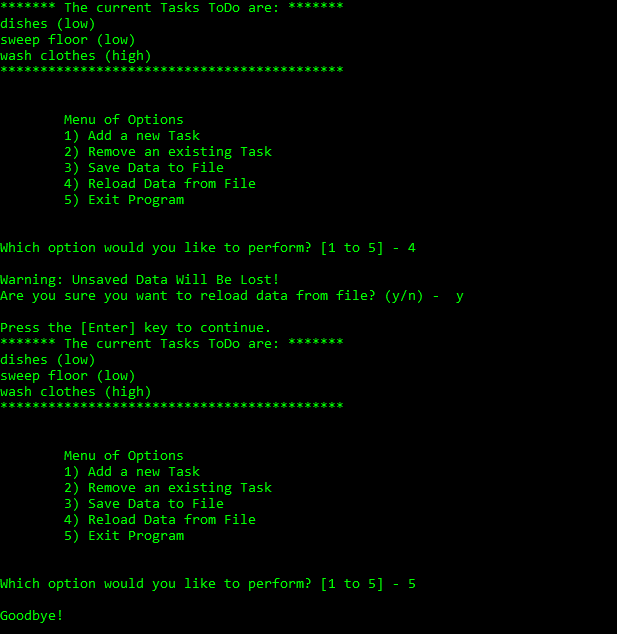


Figure : CMD Output 3

The very last Figure 16, shows the text file output at the conclusion of the CMD prompted script. The text file shows the extracted values from the list into a formatted string.

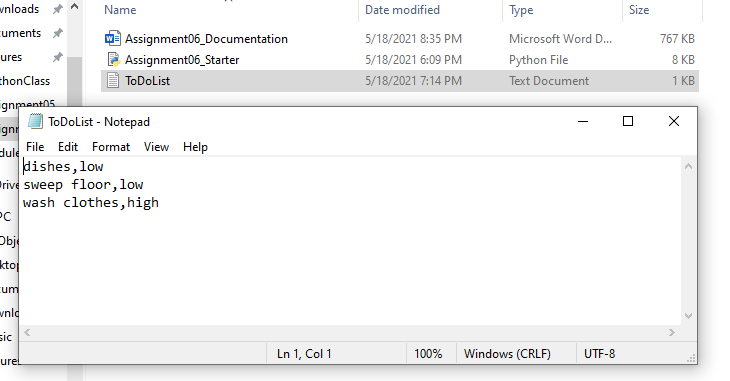


Figure : Text File Ouptut

# Summary

Utilizing functions enable the script to be much cleaner and allow for easier accessibility of common functions/commands. In this exercise, the end result was very similar to that of Assignment 05, but the script was constructed in a way that future edits will be simpler or extractions of pertinent functions will be easier. In the future, this will be the common structure of scripts as opposed to having the function coded out in the execution of the script.