Adam Gibbs

5/24/2020

IT FDN 100 A Sp 20: Foundations of Programming: Python

Assignment 06

<https://github.com/agibbs-uscg/-IntroToProg-Python-Mod06>

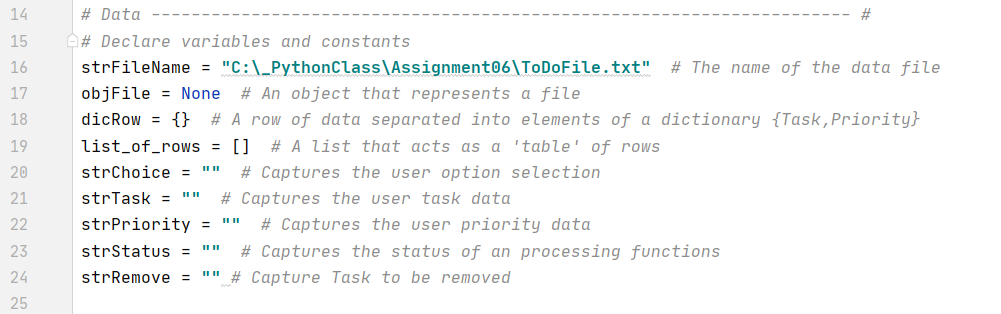
Create a Python Task Table/Dictionary Script with Functions

**Introduction**

This paper will discuss how to create a task table script file in Python that allows a user to choose options from a menu on what they want to do with their task table. The user can enter data, view data, edit data, or save/exit. If the user wishes, their data will be saved to a .txt file. This script also imports data from the previous task .txt file so it can be edited and updated. I use functions to clean-up the script.

**How the script works**

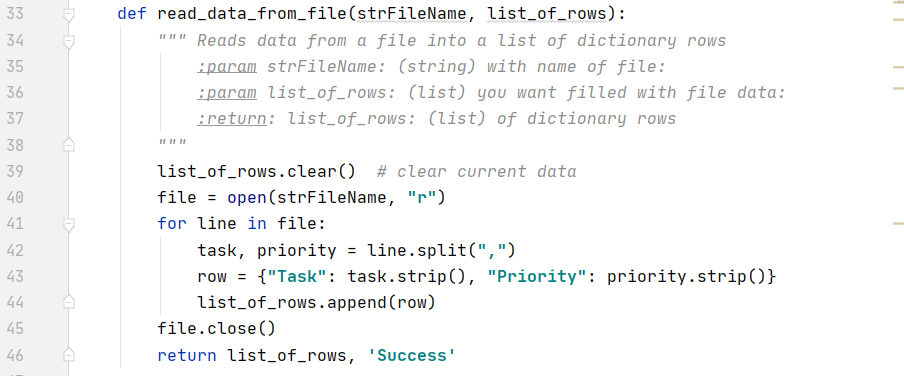
Using the Python script editing tool PyCharm, I started my script declaring variables that I used throughout the script. Figure 1. Shows the variables I defined.

**Figure 1.**

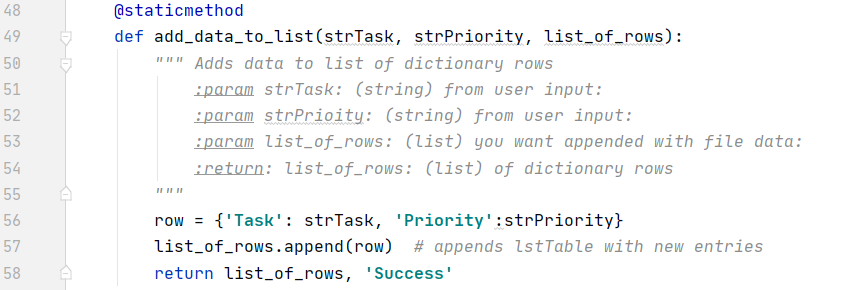
The next step in my script was to define functions that will be used later in my script. I broke these functions into two classes: Processor and IO. The processor class holds functions the process data. The IO class holds function that ask for user input and/or prints outputs to the user.

Class Processor

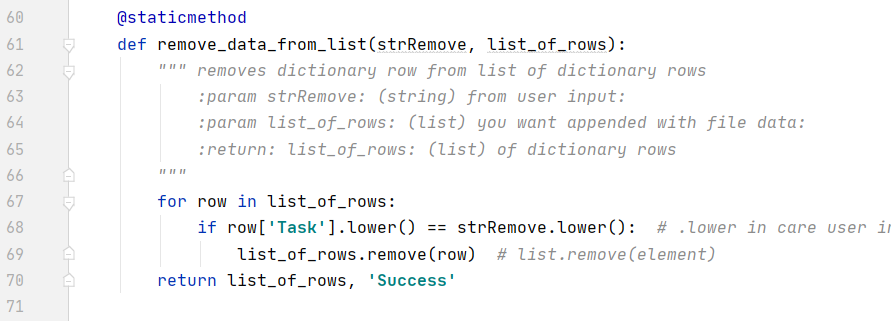
Under the processor class I built functions to read data from a saved file, add data to a list, remove data from a list, and write data to a .txt file. The read data function opens a saved .txt file, assigns each task + priority to a dictionary called row, and appends a table called list\_or\_rows. It then closed the file. This newly created list\_of\_rows is now in memory and ready for use, see Figure 2.

**Figure 2.**

The second function under class ‘Processor’ adds new tasks and priorities to the list when called. This function has parameter strTask, strPriority, and list\_of\_rows. The first two parameters are defined in a later function through user input while the list\_of\_rows is the table in memory that was created earlier. I used the ‘append’ tool to append the current list with the new input as shown in Figure 3.

**Figure 3.**

The third function under the class ‘Processor’ removes a row from the list in memory. This function has parameters strRemove (who’s value is a user input from another function) and list\_of\_rows (which is the table saved in memory. This function looks for a dictionary that has a value for key ‘Tasks’ that matches the user input. Once located, the row in the table is removed, see figure 4.



**Figure 4.**

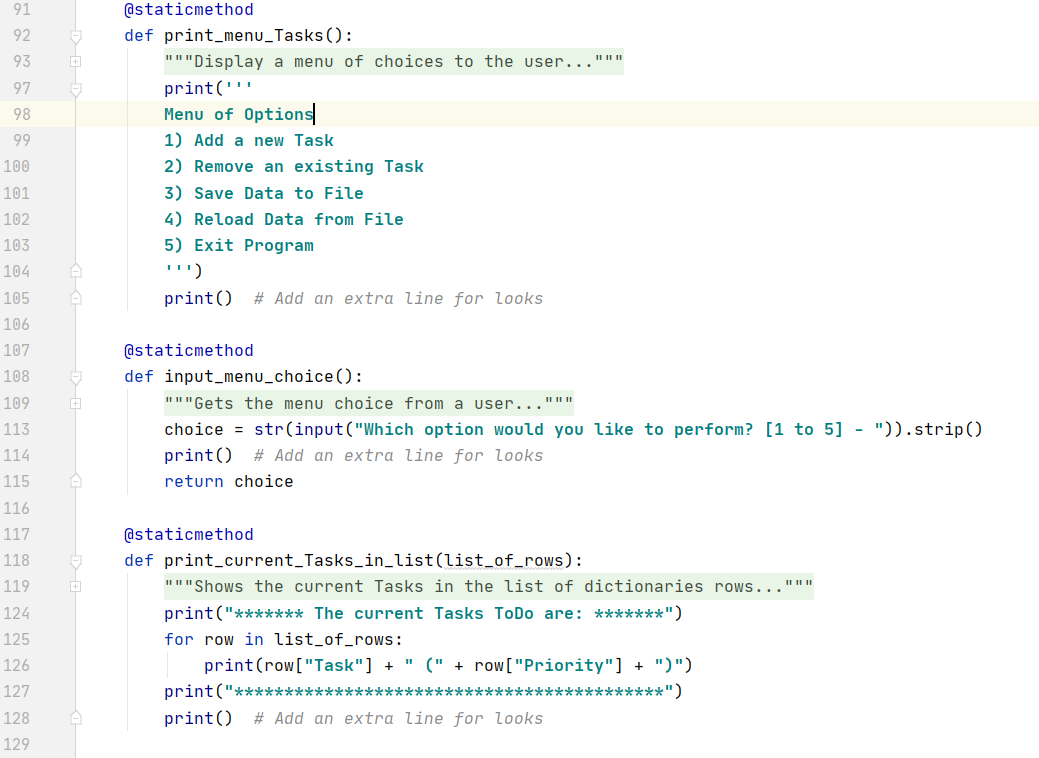
The final function within the class ‘Processor’ writes the table currently saved in memory to a .txt file. This function opens the file and writes the table in rows, with each row having a Task, comma, Priority, and character turn. It then closes the file, see Figure 5.



**Figure 5.**

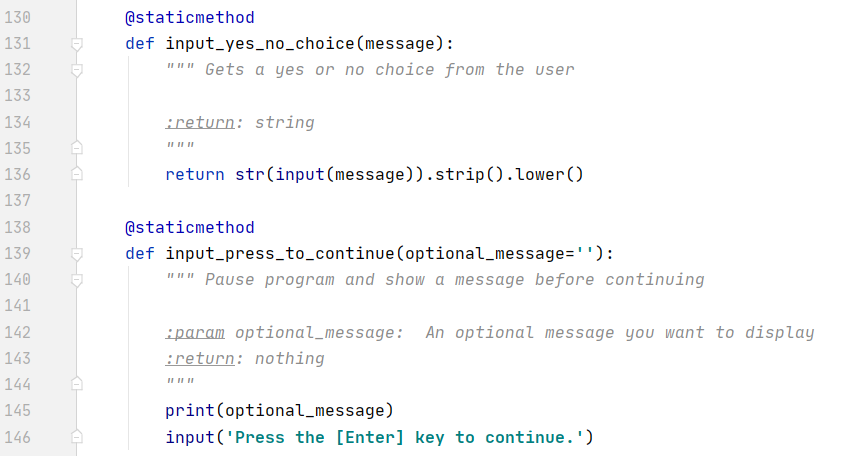
Class IO

The class ‘IO’ contains functions that ask for user inputs and/or print data for the user to see. The print\_menu\_Tasks function uses the print tool to display a menu of options when called. The Input\_menu\_choice function uses input to prompt the user to choose a menu option. The print\_current\_Tasks\_in\_list function has parameter ‘list\_of\_rows’ and displays this table in rows with a task and priority in each row. Figure 6 show the script for these three functions.



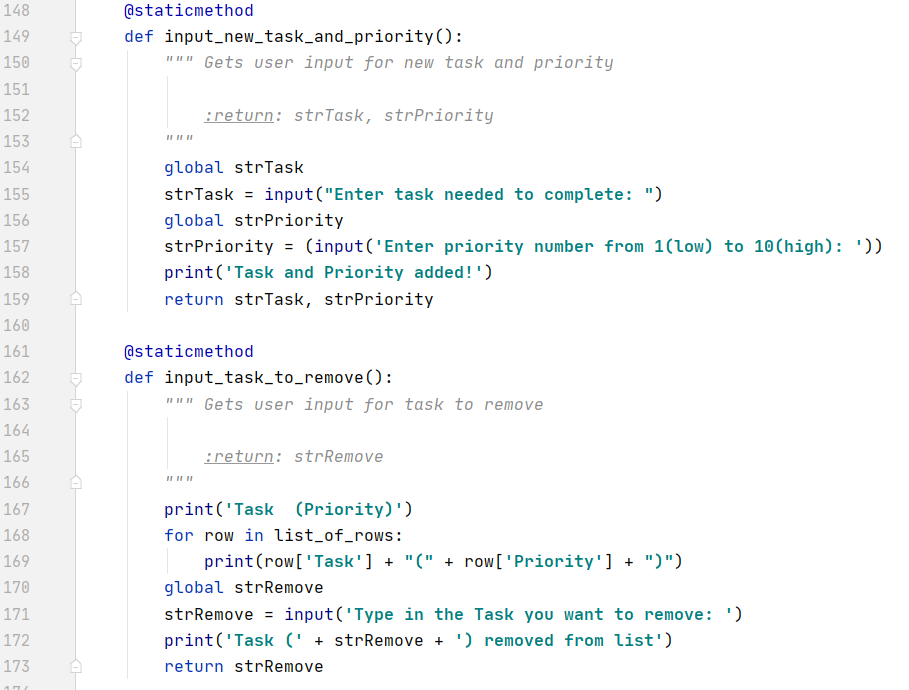
**Figure 6.**

Next, the input\_yes\_no\_message asks the user to input a y or n. the input\_press\_to\_continue function prompts the user to hit the enter key before moving on in the script. This is shown in Figure 7.



**Figure 7**.

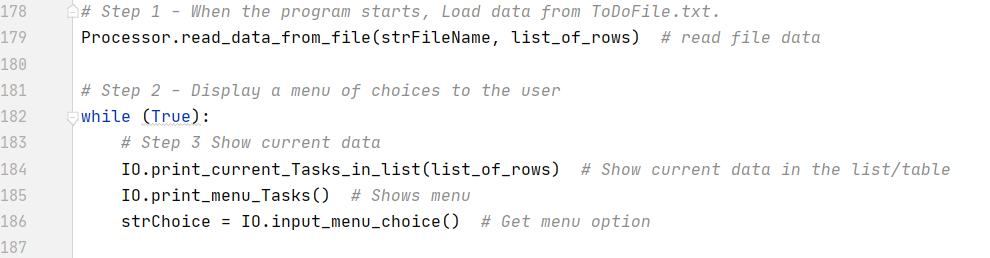
Finally, the input\_new\_taks\_and\_priority function asked for the user to input a new task with its priority. These inputs set the values for variables strTask and strPriority. The last function input\_task\_to\_remove first prints the current table then asks the user to input a task they want to remove. This input sets the value for variable strRemove. The script for these function are shown in Figure 8.



**Figure 8.**

**Main script**

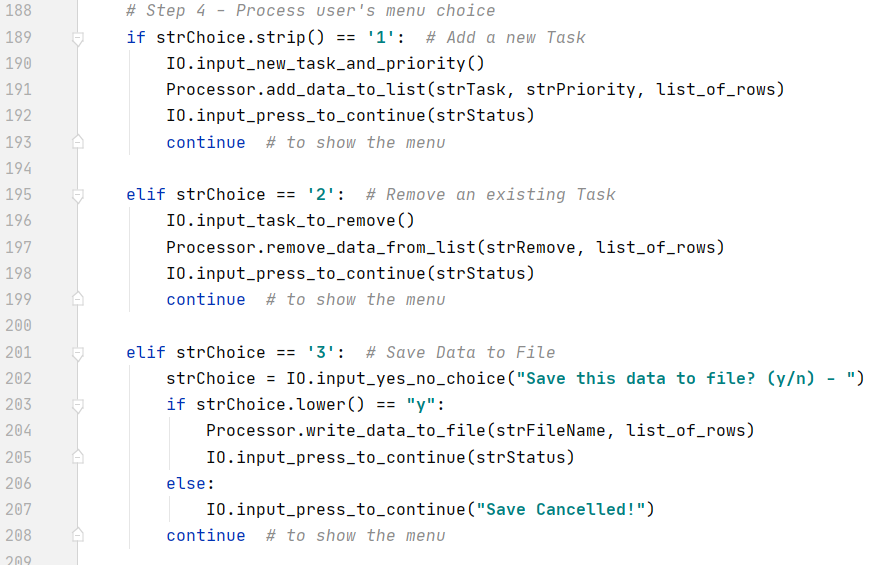
With these function set I then move to the body of the script. First, the script calls for function read\_data\_from\_file in class ‘Processor’. This, as discussed above, reads data from a file and creates a table in memory. The remainder of the script is a ‘while’ loop that continues to run until the user asks to exit. The loop first calls for the display of the current list, followed by the menu and finally the prompt for the user to choose from the menu. This is accomplished by calling for functions print\_current\_Tasks\_in\_list, print\_menu\_Tasks, and input\_menu\_choice which are all in the class ‘IO’ (shown in Figure 9.). This will display upon completion for each ‘if’ statement until exited.



**Figure 9.**

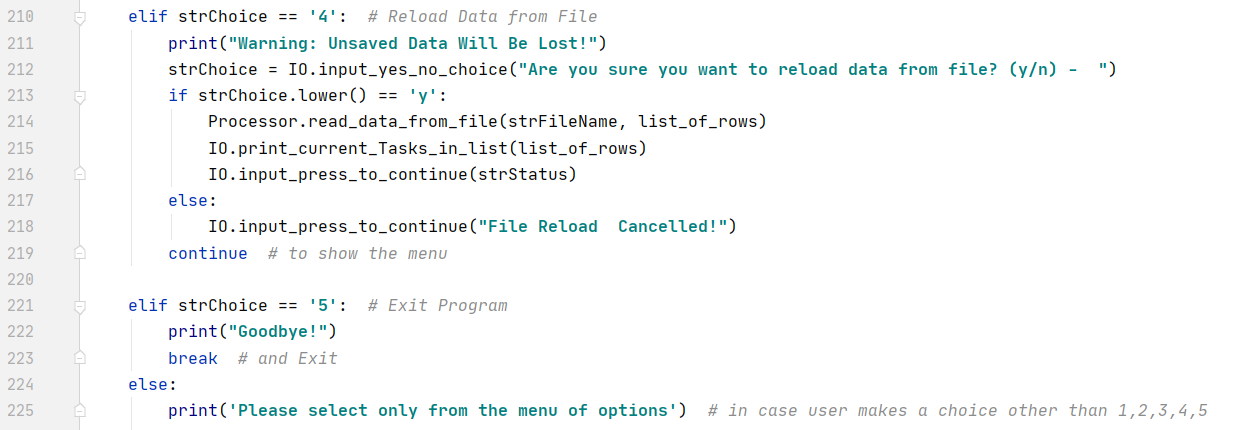
A series of ‘If’ statements follows corresponding to the menu choice by the user. Each if statement ends with the input\_press\_to\_continue function to prompt the user to hit enter to continue the script. Choice 1 calls the functions input\_new\_task\_and\_priority and add\_data\_to\_list. Choice 2 calls the functions input\_task\_to\_remove and remove\_data\_from\_list. Choice 3 calls the functions IO.input\_yes\_no\_choice. A nested ‘If’ statement then calls function write\_data\_to\_file if the user answered ‘y’ or function input\_press\_to\_continue is the user enter anything but ‘y’.

Using the Python script editing tool PY CHARM, this assignment involved using the print and input functions with string variables, the append and write function to store keystrokes in a .txt file, and a ‘while’ loop and ‘if’, ‘elif’, ‘else’ functions. I started the script by identifying variables that will be used in future steps. After naming the variables, the following steps were completed. These are shown in Figure 10.



**Figure 10.**

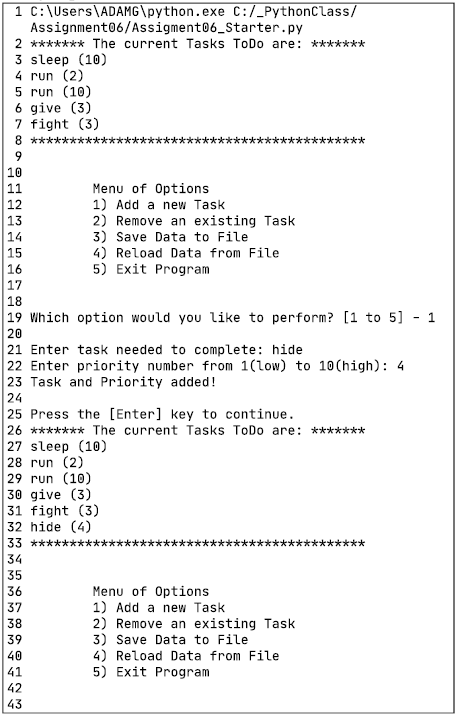
If the user chooses 4, the script warns the user that reloading the data will delete unsaved data the calls function input\_yes\_no\_choice. A nested ‘If’ statement then calls functions read\_data\_from\_file, print\_current\_Tasks\_in\_list, and input\_press\_to\_continue if the user types ‘y’. If the user types anything else the script calls function input\_press\_to\_continue with the message ‘File Reload Cancelled’. A choice of 5 will simply exit the loop and close the script. If the user types anything but 1-5, a statement prints that they must chose from the menu. This is accomplished with an ‘else’ statement at the end of the script. See Figure 11.



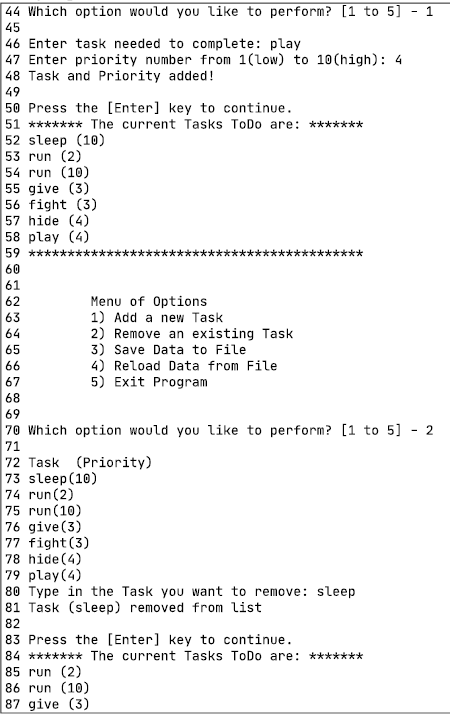
**Figure 11.**

**Script running**

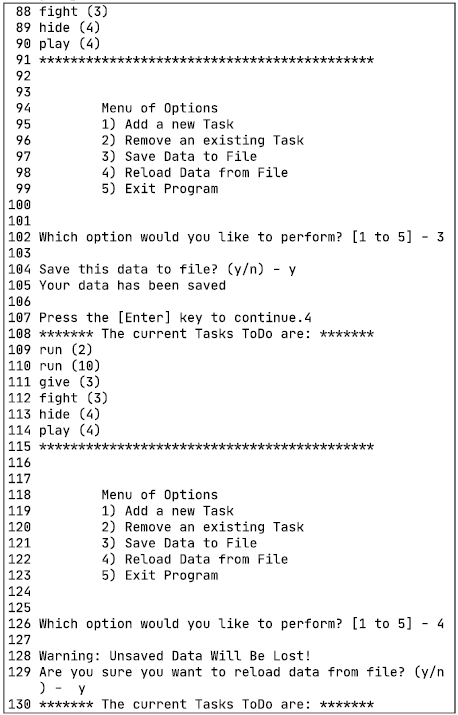
Figures 12-15. below shows the script running in Pycharm. For this run I first show what is already on the table. I then add task ‘hide’ with priority ‘4’ and task ‘play’ with priority ‘4’. Then I remove task ‘sleep‘, show the new list, save the file and exit the program. Figure 16. Shows the contents of the created .txt file.



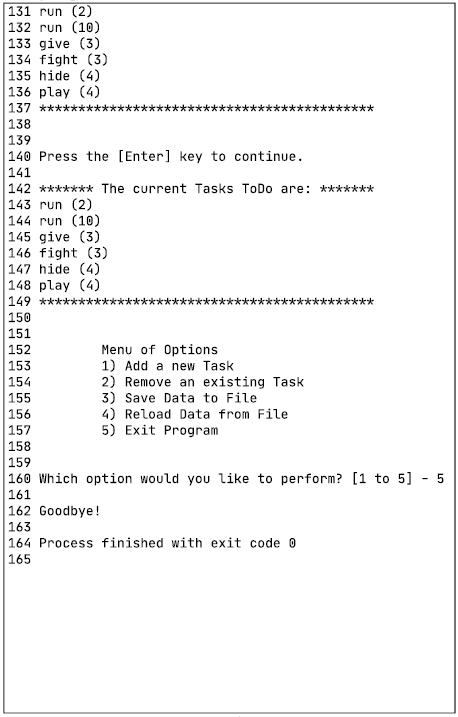
**Figure 12.**



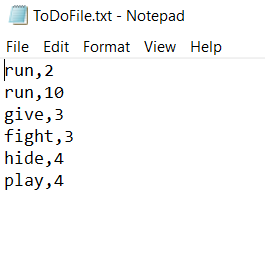
**Figure 13.**



**Figure 14.**

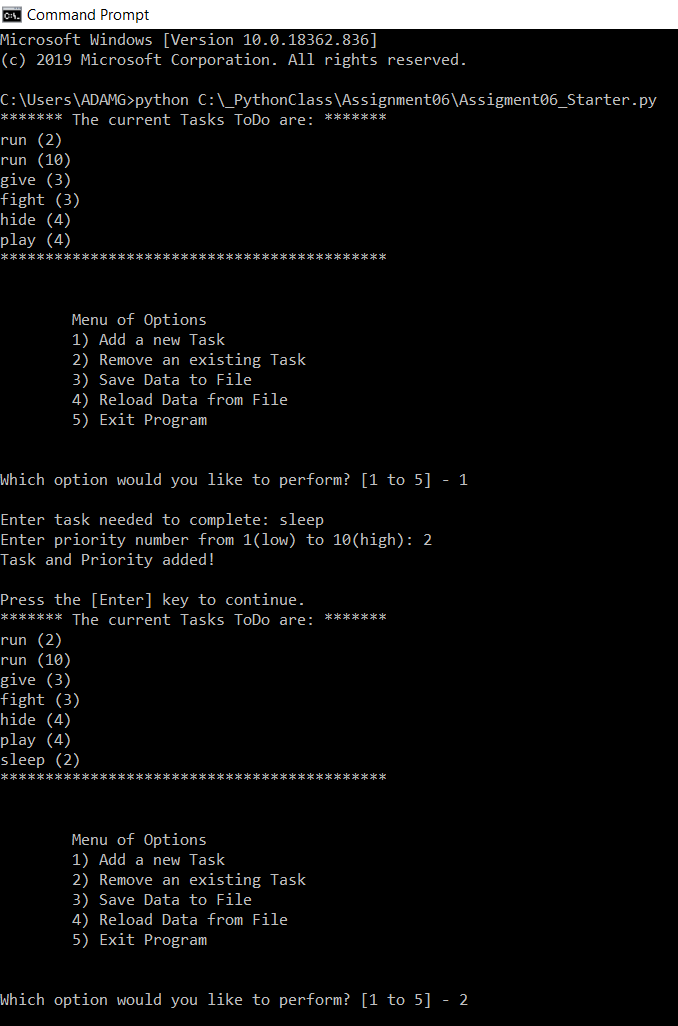


**Figure 15.**

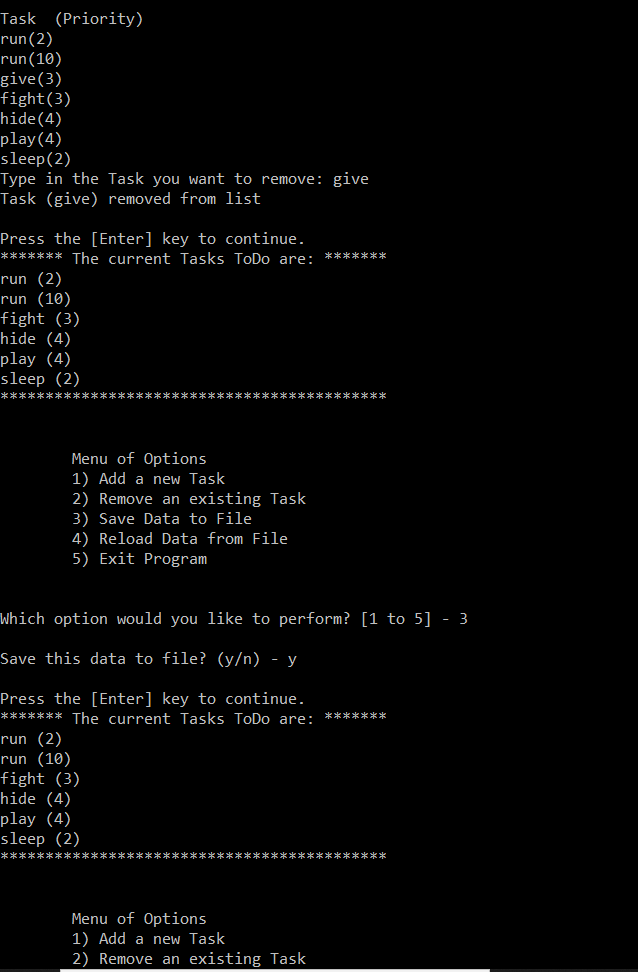


**Figure 16.**

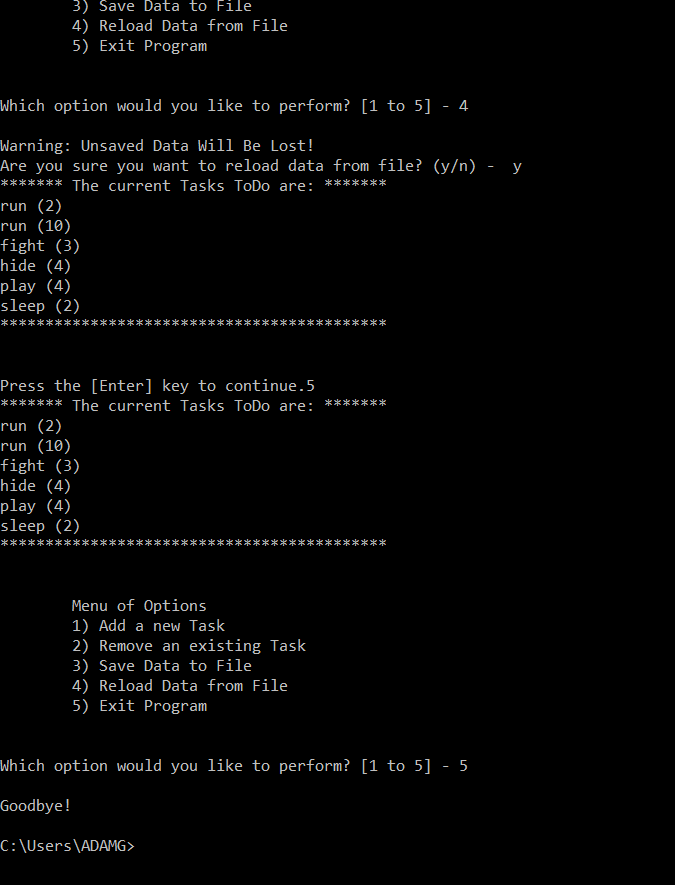
Figures 17-19. below shows the script running in CMD. For this run I first show what is already on the table. I then add task ‘sleep’ with priority ‘2’. Then I remove task ‘give ‘, show the new list, save the file and exit the program. Figure 20. Shows the contents of the created .txt file.



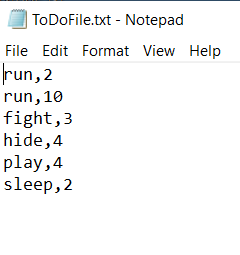
**Figure 17.**



**Figure 18.**



**Figure 19.**



**Figure 20.**

**Summary**

Through this assignment I created a task table file in Python that allows a user to choose options from a menu on what they want to do with their task table. The user can enter data, view data, edit entries or save/exit. I used functions to clean and organized the script.