Hien Ho
August 16, 2022
Foundations of Programming: Python
Assignment 06
https://github.com/HHoUW/IntroToProg-Python-Mod06

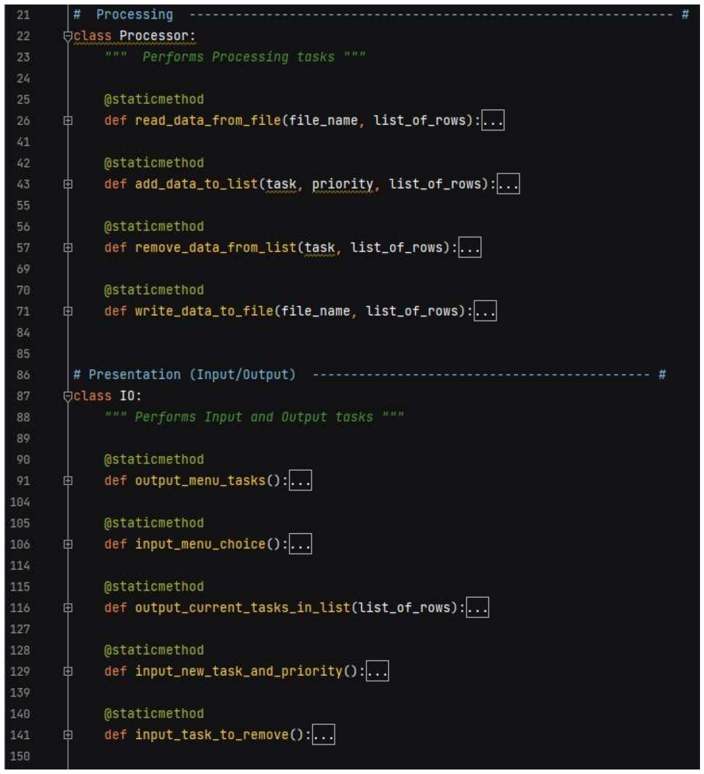
Completing To Do List Script

Introduction

In this report, I will go over how I added code to an existing Python script to create a "To-Do List" program. The program runs and shows the user a list of current tasks on the list and presents a menu of options to the user. The options allow the user to add new tasks to the list, to remove tasks from the list, to save the list to a data file and to exit the program. The program makes use of programmer-created **Functions** that groups code statements that perform a specific task together. These functions are group into **Classes** base on the type of task that they perform**.**

Writing the code

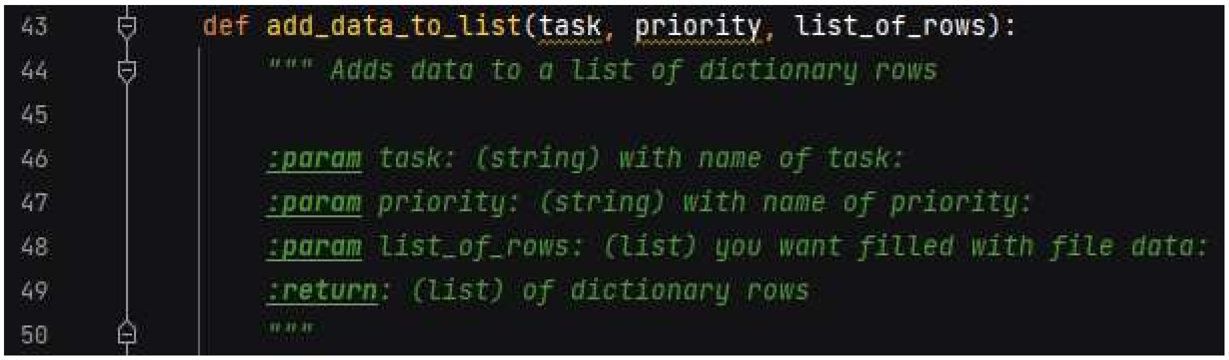
I started with an existing script with some of the code already written. The script is divided sections. There is a Data, Processing, Presentation (Input/Output) and Main code sections. The Data section is where variables and constants are declared. The Processing and Presentation sections contains all the functions created to handle all of the input/output tasks and the processing of data. (Figure 1)



```
#  Processing  -------------------------------------------------------------- #
class Processor:
    """  Performs Processing tasks """

    @staticmethod
    def read_data_from_file(file_name, list_of_rows):...

    @staticmethod
    def add_data_to_list(task, priority, list_of_rows):...

    @staticmethod
    def remove_data_from_list(task, list_of_rows):...

    @staticmethod
    def write_data_to_file(file_name, list_of_rows):...


# Presentation (Input/Output)  -------------------------------------------- #
class IO:
    """ Performs Input and Output tasks """

    @staticmethod
    def output_menu_tasks():...

    @staticmethod
    def input_menu_choice():...

    @staticmethod
    def output_current_tasks_in_list(list_of_rows):...

    @staticmethod
    def input_new_task_and_priority():...

    @staticmethod
    def input_task_to_remove():...
```

Figure 1: List of functions under the Processor and Presentation classes.

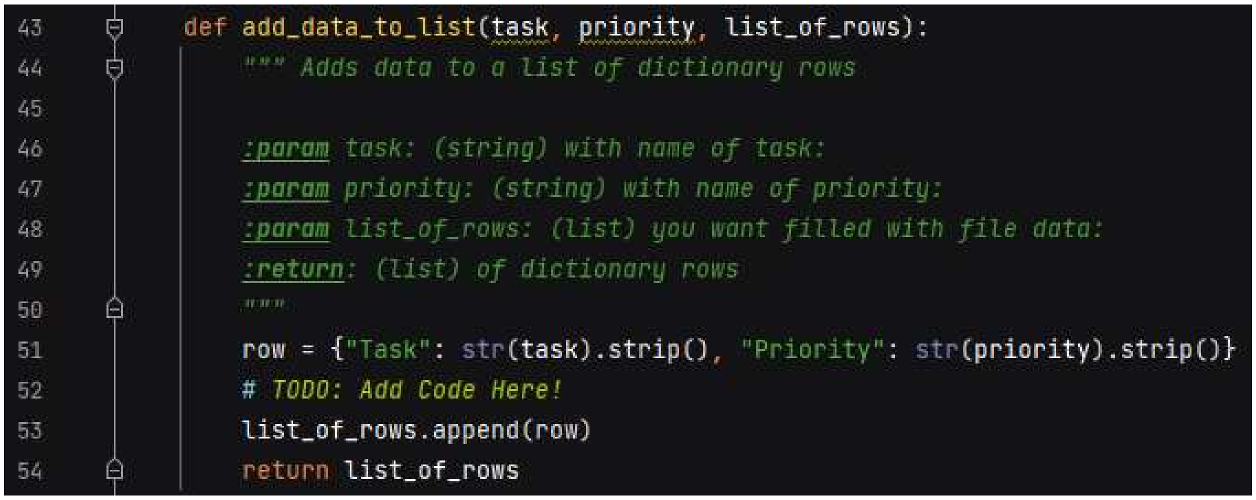
I wrote code to finish creating some of the functions. To create a new function you first have to define it using the **def** keyword follow by the name of the function and the parameters that can be pass in and out of the functions. The parameters are set within parentheses following the function name. It is common to include a header or **docstring** at the beginning of a function that provides some information or developer notes about the function. The docstring consists of text in triple quotation marks. (Figure 2)



```python
def add_data_to_list(task, priority, list_of_rows):
    """ Adds data to a list of dictionary rows

    :param task: (string) with name of task:
    :param priority: (string) with name of priority:
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
```

Figure 2: Define a function with three parameters and the docstring within the function.

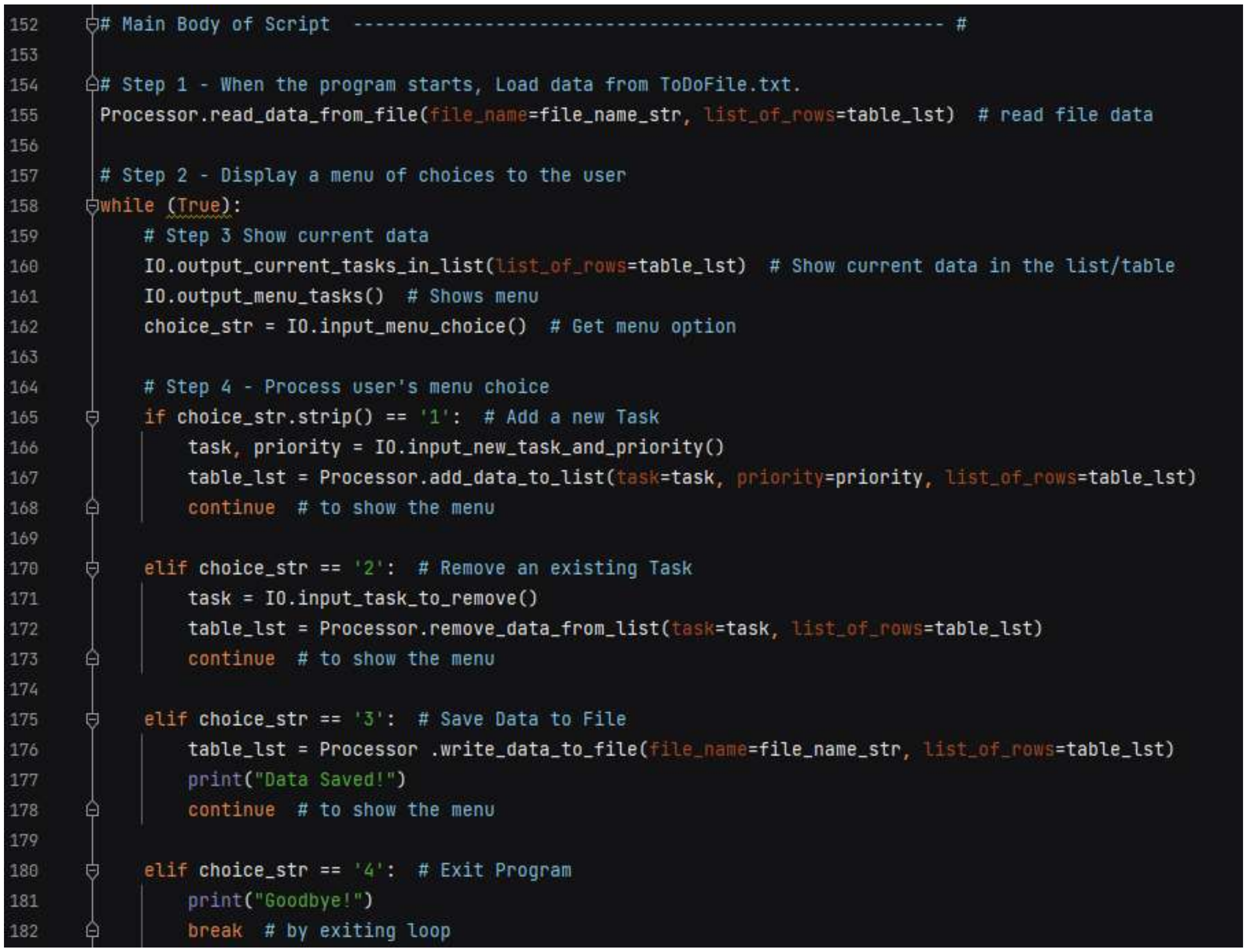
As the function can receive a value through a parameter, it can also send a value back out that then can be use in a different part of the script. This is done with the **return** statement. For the “add_data_to_list” function, a python list (*list_of_rows*) is pass into the function along with a task and its priority. The task and priority are appended to the list and then the newly updated list is return back out of the function. (Figure 3)



```python
def add_data_to_list(task, priority, list_of_rows):
    """ Adds data to a list of dictionary rows

    :param task: (string) with name of task:
    :param priority: (string) with name of priority:
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
    row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
    # TODO: Add Code Here!
    list_of_rows.append(row)
    return list_of_rows
```

Figure 3: Function to add new task to list.

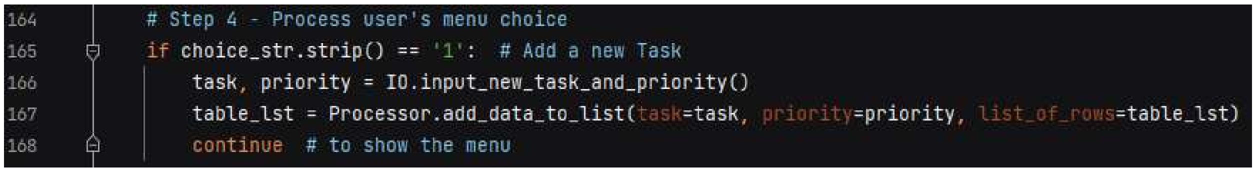
The main body of the code consist of statements calling on the various function to perform a task base on the choices the user of the program makes. A function is call by its class follow by “.” and then the function’s name, *Class.Function()*. (Figure 4)



```
152  # Main Body of Script  ------------------------------------------------------ #
153
154  # Step 1 - When the program starts, Load data from ToDoFile.txt.
155  Processor.read_data_from_file(file_name=file_name_str, list_of_rows=table_lst)  # read file data
156
157  # Step 2 - Display a menu of choices to the user
158  while (True):
159      # Step 3 Show current data
160      IO.output_current_tasks_in_list(list_of_rows=table_lst)  # Show current data in the list/table
161      IO.output_menu_tasks()  # Shows menu
162      choice_str = IO.input_menu_choice()  # Get menu option
163
164      # Step 4 - Process user's menu choice
165      if choice_str.strip() == '1':  # Add a new Task
166          task, priority = IO.input_new_task_and_priority()
167          table_lst = Processor.add_data_to_list(task=task, priority=priority, list_of_rows=table_lst)
168          continue  # to show the menu
169
170      elif choice_str == '2':  # Remove an existing Task
171          task = IO.input_task_to_remove()
172          table_lst = Processor.remove_data_from_list(task=task, list_of_rows=table_lst)
173          continue  # to show the menu
174
175      elif choice_str == '3':  # Save Data to File
176          table_lst = Processor .write_data_to_file(file_name=file_name_str, list_of_rows=table_lst)
177          print("Data Saved!")
178          continue  # to show the menu
179
180      elif choice_str == '4':  # Exit Program
181          print("Goodbye!")
182          break  # by exiting loop
```

Figure 4: Main body of script. Calling function based on user input.

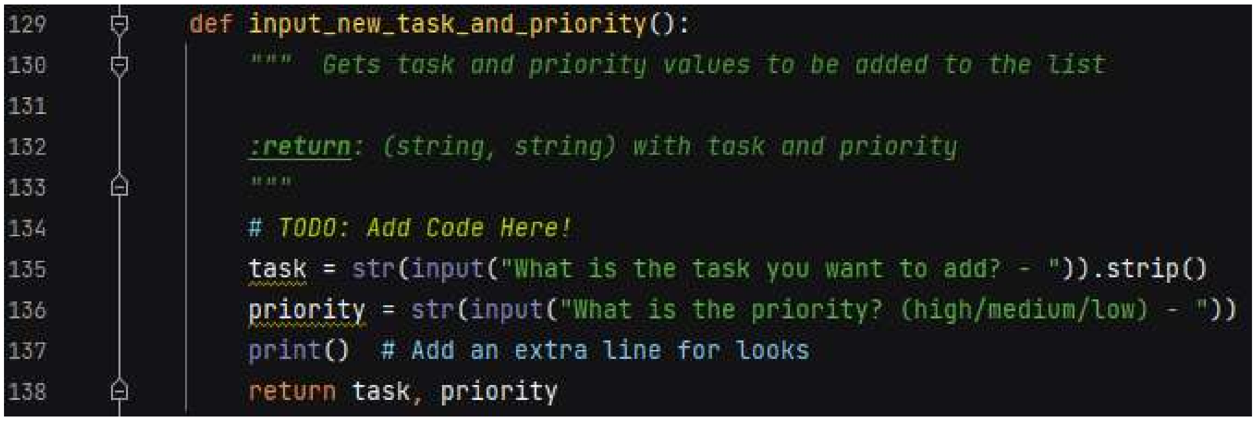
If the user chooses option 1, to add a new task, the function input_new_task_and_priority() is called. The variables task and priority will get values return by the function, shown in line 166 in the code example. (Figure 5)



```
164      # Step 4 - Process user's menu choice
165      if choice_str.strip() == '1':  # Add a new Task
166          task, priority = IO.input_new_task_and_priority()
167          table_lst = Processor.add_data_to_list(task=task, priority=priority, list_of_rows=table_lst)
168          continue  # to show the menu
```

Figure 5: User chooses option to add new task, input_new_task_and_priority() function call

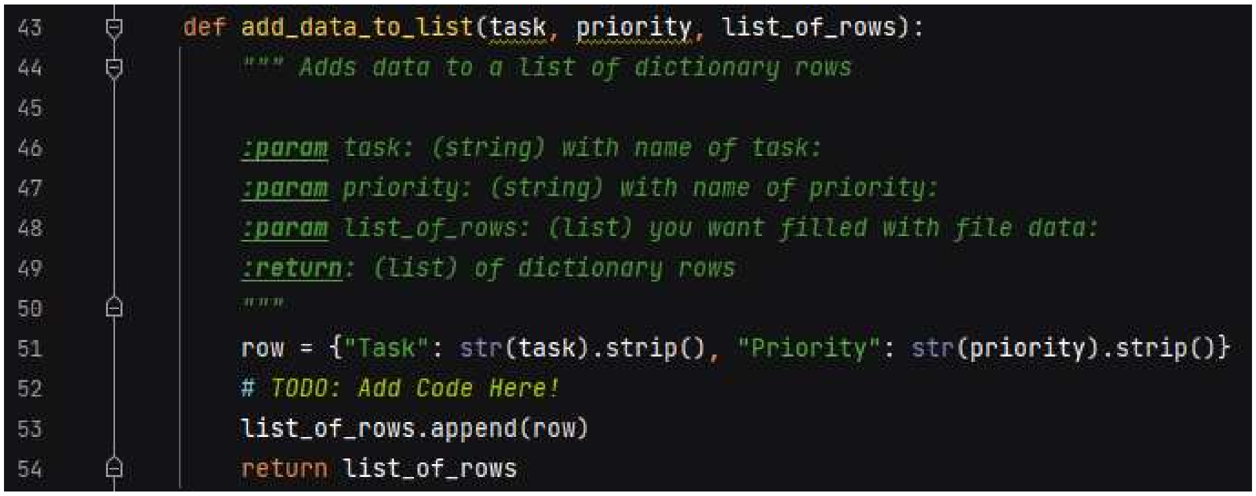
The function input_new_task_and_priority() runs. Prompt the user for inputs and then returns the user’s inputs back to the task and priority variables. (Figure 6)



```python
def input_new_task_and_priority():
    """  Gets task and priority values to be added to the list

    :return: (string, string) with task and priority
    """
    # TODO: Add Code Here!
    task = str(input("What is the task you want to add? - ")).strip()
    priority = str(input("What is the priority? (high/medium/low) - "))
    print()  # Add an extra line for looks
    return task, priority
```

Figure 6: The input_new_task_and priority() function.

Next the add_data_to_list() function is call which will return an updated list to table_lst. Three parameters are pass into the function, task, priority and a list, as shown in line 167 in the code example in Figure 5 above. The function takes the task and priority and creates a dictionary. It then appends the dictionary to the list that was passed into the function and then returns the updated list back out of the function. (Figure 7)



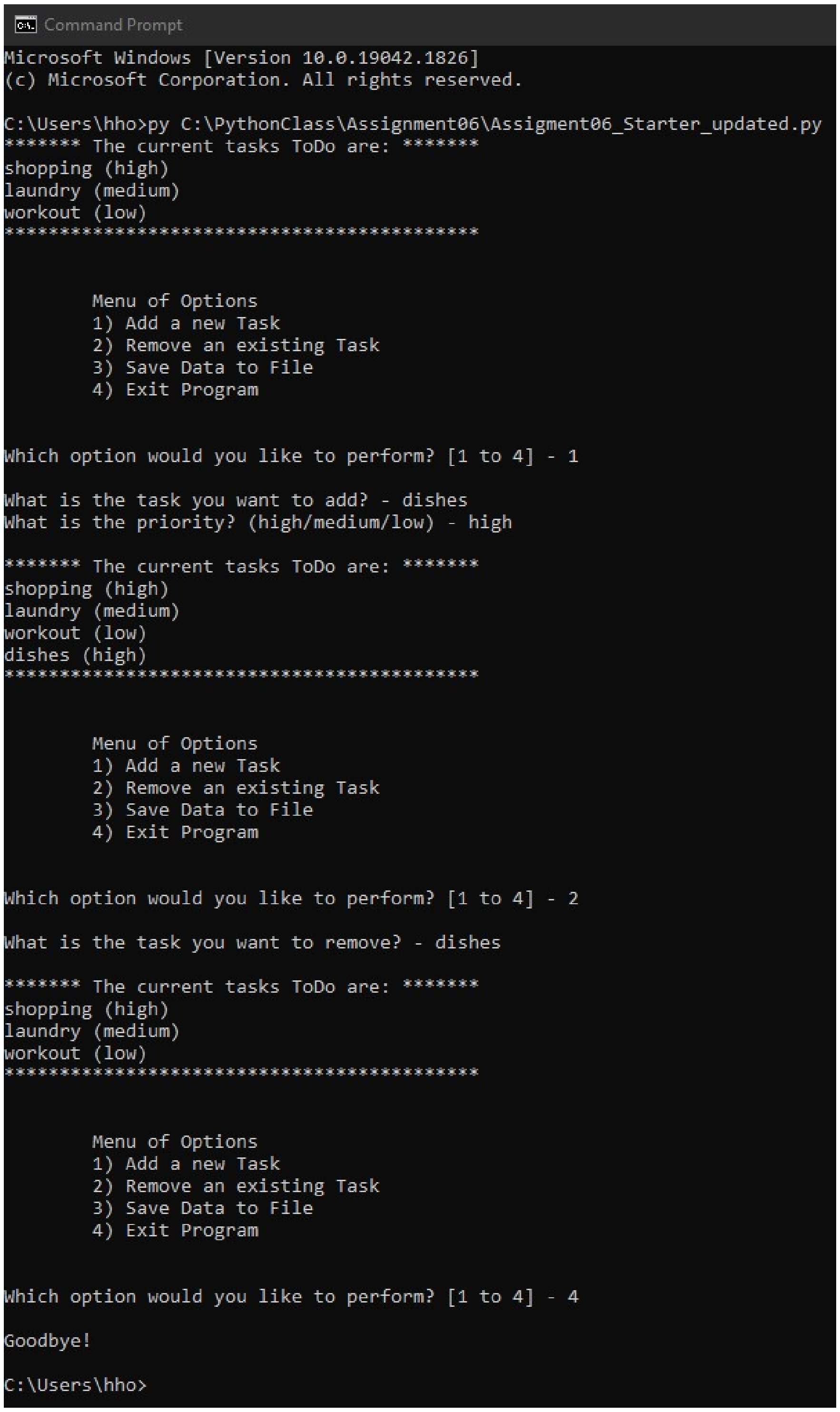
```python
def add_data_to_list(task, priority, list_of_rows):
    """ Adds data to a list of dictionary rows

    :param task: (string) with name of task:
    :param priority: (string) with name of priority:
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
    row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
    # TODO: Add Code Here!
    list_of_rows.append(row)
    return list_of_rows
```

Figure 7: The add_data_to_list() function.

After the function completes its process and return the newly updated list to table_lst, it brings the user back to the main menu. Similar process happens when the user chooses another option. Specific functions are call to complete the processes the user's request.

Running the script

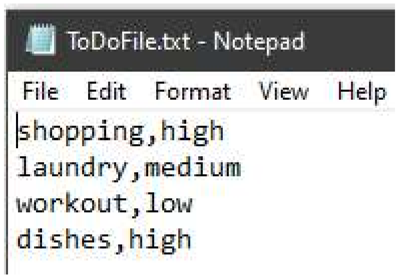
I ran the script in both PyCharm and the command window and it ran properly on both. Here is the program running in the command window. (Figure 8)



```
Command Prompt
Microsoft Windows [Version 10.0.19042.1826]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hho>py C:\PythonClass\Assignment06\Assigment06_Starter_updated.py
******* The current tasks ToDo are: *******
shopping (high)
laundry (medium)
workout (low)
*******************************************



        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

What is the task you want to add? - dishes
What is the priority? (high/medium/low) - high

******* The current tasks ToDo are: *******
shopping (high)
laundry (medium)
workout (low)
dishes (high)
*******************************************



        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 2

What is the task you want to remove? - dishes

******* The current tasks ToDo are: *******
shopping (high)
laundry (medium)
workout (low)
*******************************************



        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

Goodbye!

C:\Users\hho>
```

Figure 8: Program running in command window

This is a screenshot of the text file to verify that data is successfully written to a text file. (Figure 9)



```
shopping,high
laundry,medium
workout,low
dishes,high
```

Figure 9: Verifying the data is in the text file

Summary

Using the knowledge that I have gain so far for this course, I contributed code to an existing script to create a Python program to let a user to view and edit a to-do list. The script makes use of a number of programmer-created functions to perform specific tasks and classes to group the functions base on those tasks.