

Trevor Huynh

Nov 23, 2022

IT FDN 110 A

Assignment 06

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

[Module06 Website | IntroToProg-Python-Mod06 \(trevwin.github.io\)](#)

## **Assignment 06: Functions Script**

### **Introduction**

In this week, functions/parameters, global/local variables, error handling/debugging, and some additional github basics were covered. For this week's assignment, a starter script with multiple functions is provided. The script runs similar to the script in assignment 5 with the difference being the inclusion of functions to do the data processing and input/output. Some of the starter file's functions are missing code to allow it to run properly.

### **Starter File Breakdown / Structure**

As this assignment requires working off an existing starter code, the starter code structure is analyzed first. The basic structure is summarized below:

---

#### DATA

Declaration / initialization of global variables

#### PROCESSOR CLASS FUNCTIONS

- read\_data\_from\_file(file\_name, list\_of\_rows)
- add\_data\_to\_list(task, priority, list\_of\_rows)
  - o ADD CODE
- remove\_data\_from\_list(task, list\_of\_rows)
  - o ADD CODE
- write\_data\_to\_file(file\_name, list\_of\_rows)
  - o ADD CODE

#### I/O CLASS FUNCTIONS

- output\_menu\_tasks()
- input\_menu\_choice()

- output\_current\_tasks\_in\_list(list\_of\_rows)
- input\_new\_task\_and\_priority()
  - o ADD CODE
- input\_task\_to\_remove()
  - o ADD CODE

MAIN BODY OF SCRIPT (similar to assignment 5)

- 1) Read data from file
- 2) Display menu to user
- 3) Show current data (IO functions called)
- 4) Process menu choices
  - a. Choice = 1, add new task
  - b. Choice = 2, remove existing task
  - c. Choice = 3, save data to file
  - d. Choice = 4, exit program

---

*Table 1: Starter File Basic Structure.*

From analyzing the starter file, the overall script works like the script in Assignment 5. The major difference is utilizing various functions to accomplish the task. These functions are separated into processor or IO class functions.

There are a total of 5 functions that require code to complete the starter file script. The strategy is to isolate each function and verify that each function works, then doing an overall test with the script.

### **Individual Function Testing**

#### **IO.input\_new\_task\_and\_priority – Test Function**

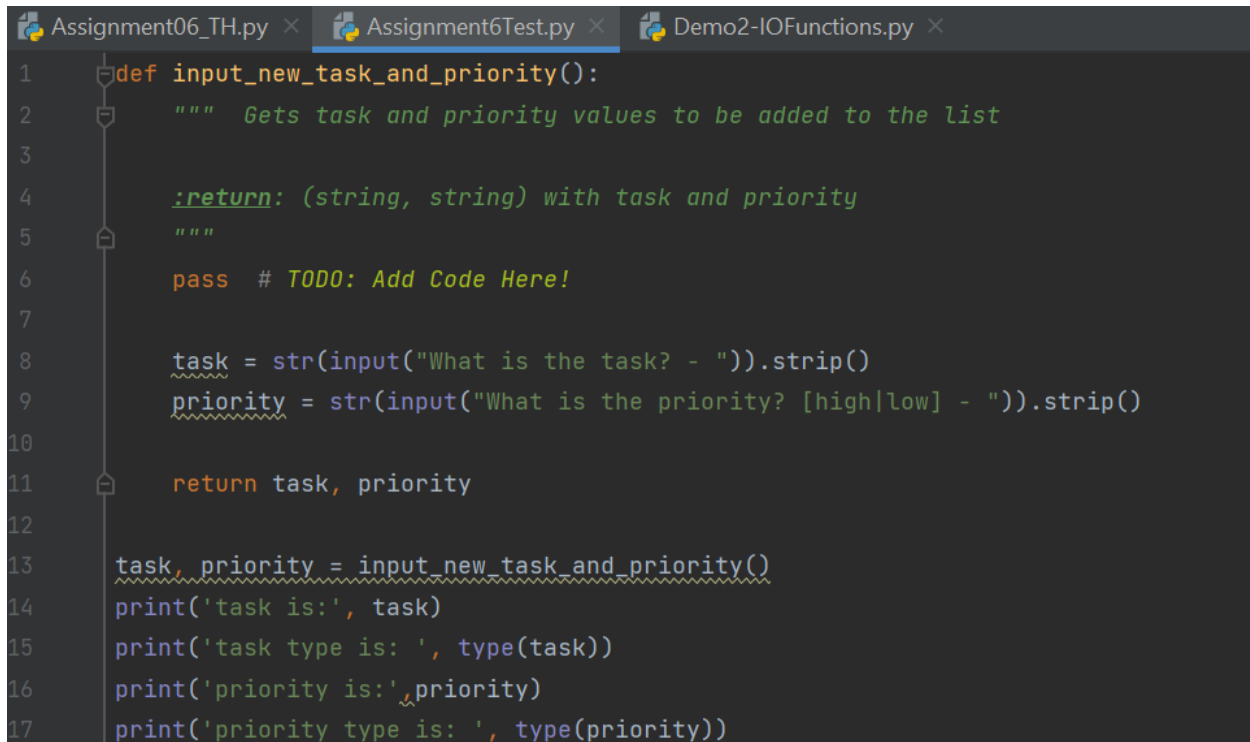
This function's goal is to get task and priority values and add to a list. The function returns a list (string) with variables task and priority.

This function shows up in Step 4 in the main body of the script along with a processor class function. The returned values are then assigned to task, priority variables.

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

*Figure 1: Main Body of Script (Choice 1)*

To finish this function, the function was isolated in a new python file and fed user input data. The user input data was then printed out to verify correct assignment and data type.

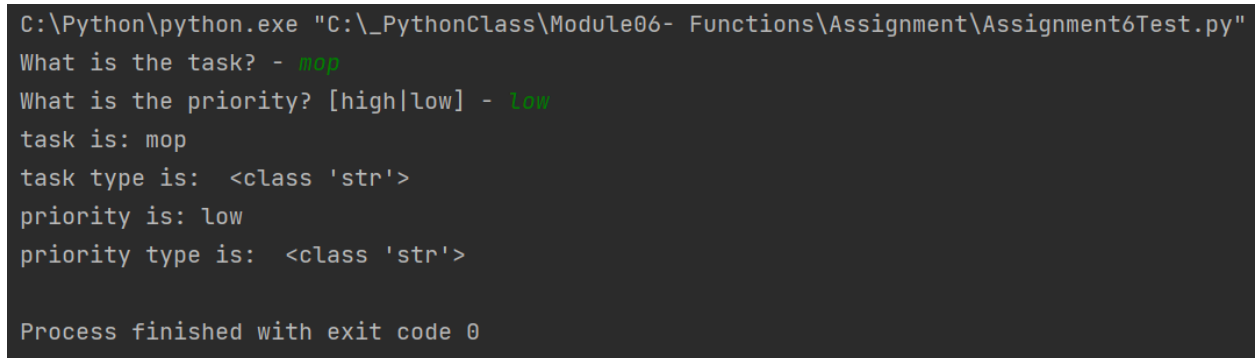


```

1 def input_new_task_and_priority():
2     """ Gets task and priority values to be added to the list
3
4     :return: (string, string) with task and priority
5     """
6     pass # TODO: Add Code Here!
7
8     task = str(input("What is the task? - ")).strip()
9     priority = str(input("What is the priority? [high|low] - ")).strip()
10
11     return task, priority
12
13 task, priority = input_new_task_and_priority()
14 print('task is:', task)
15 print('task type is: ', type(task))
16 print('priority is:', priority)
17 print('priority type is: ', type(priority))

```

Figure 2: Function IO.input\_new\_task\_and\_priority() Test Code



```

C:\Python\python.exe "C:\_PythonClass\Module06- Functions\Assignment\Assignment6Test.py"
What is the task? - mop
What is the priority? [high|low] - low
task is: mop
task type is: <class 'str'>
priority is: low
priority type is: <class 'str'>

Process finished with exit code 0

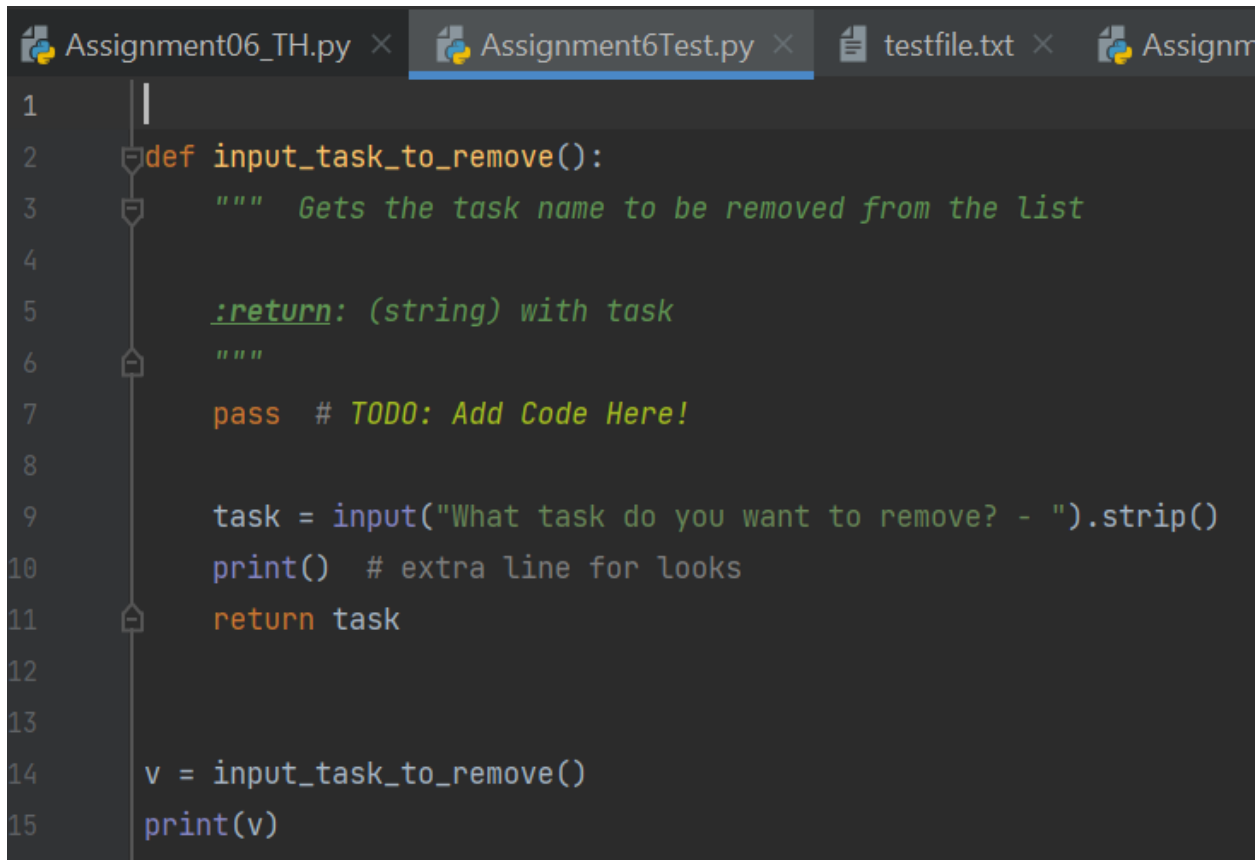
```

Figure 3: Function IO.input\_new\_task\_and\_priority() Test Results

## IO.input\_task\_to\_remove – Test Function

This function's goal is to just return the user's inputted task that is stripped of all hidden spaces. As this is an IO class function, it is important to not confuse it with the processor task function that does the actual data processing. This function only cleans up the user inputted task and returns it, whereas the processor class variant does the search in the dictionary list, finds the match and then removes it.

A simple test was completed to verify this function. The function worked as expected.

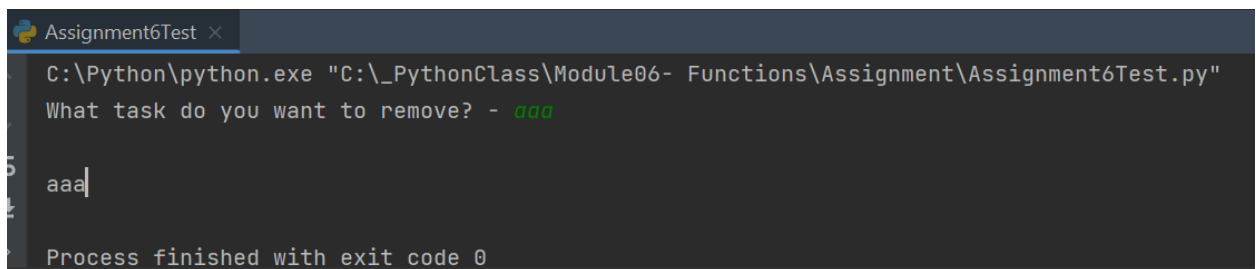


```

1 |
2 def input_task_to_remove():
3     """ Gets the task name to be removed from the list
4
5     :return: (string) with task
6     """
7     pass # TODO: Add Code Here!
8
9     task = input("What task do you want to remove? - ").strip()
10    print() # extra line for looks
11    return task
12
13
14 v = input_task_to_remove()
15 print(v)

```

Figure 4: Function IO.input\_task\_to\_remove() Test Code



```

Assignment6Test x
C:\Python\python.exe "C:\_PythonClass\Module06- Functions\Assignment\Assignment6Test.py"
What task do you want to remove? - aaa

aaa
Process finished with exit code 0

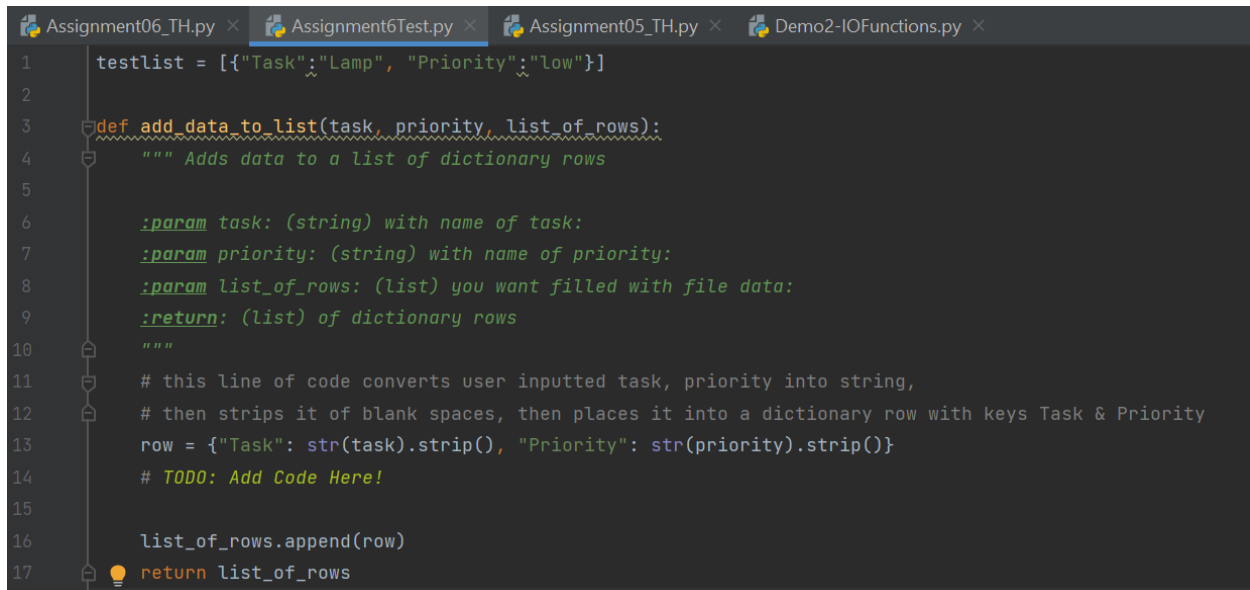
```

Figure 5: Function IO.input\_task\_to\_remove() Test Results

## Processor.add\_data\_to\_list – Test Function

The next function in step 4 (add new task) of the main script is the processor class function add\_data\_to\_list (See Figure 1).

This function's goal is to add data to a list of dictionary rows. The inputs/parameters of this function are: task, priority, and list\_of\_rows. As the starter code for this specific function already converts user inputted task/priority into a string, strips it of blank spaces, and then places it into a dictionary with keys, the only additional code required is to append it to variable list\_of\_rows using the append function.

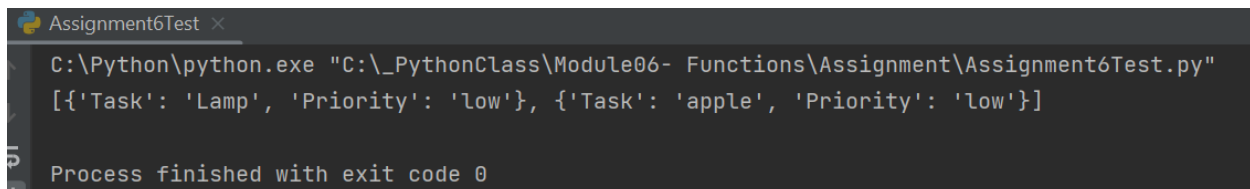


```

1  testlist = [{"Task": "Lamp", "Priority": "low"}]
2
3  def add_data_to_list(task, priority, list_of_rows):
4      """ Adds data to a list of dictionary rows
5
6      :param task: (string) with name of task:
7      :param priority: (string) with name of priority:
8      :param list_of_rows: (list) you want filled with file data:
9      :return: (list) of dictionary rows
10     """
11     # this line of code converts user inputted task, priority into string,
12     # then strips it of blank spaces, then places it into a dictionary row with keys Task & Priority
13     row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
14     # TODO: Add Code Here!
15
16     list_of_rows.append(row)
17     return list_of_rows

```

Figure 6: Function Processor.add\_data\_to\_list() Test Code



```

Assignment6Test x
C:\Python\python.exe "C:\_PythonClass\Module06- Functions\Assignment\Assignment6Test.py"
[{'Task': 'Lamp', 'Priority': 'low'}, {'Task': 'apple', 'Priority': 'low'}]

Process finished with exit code 0

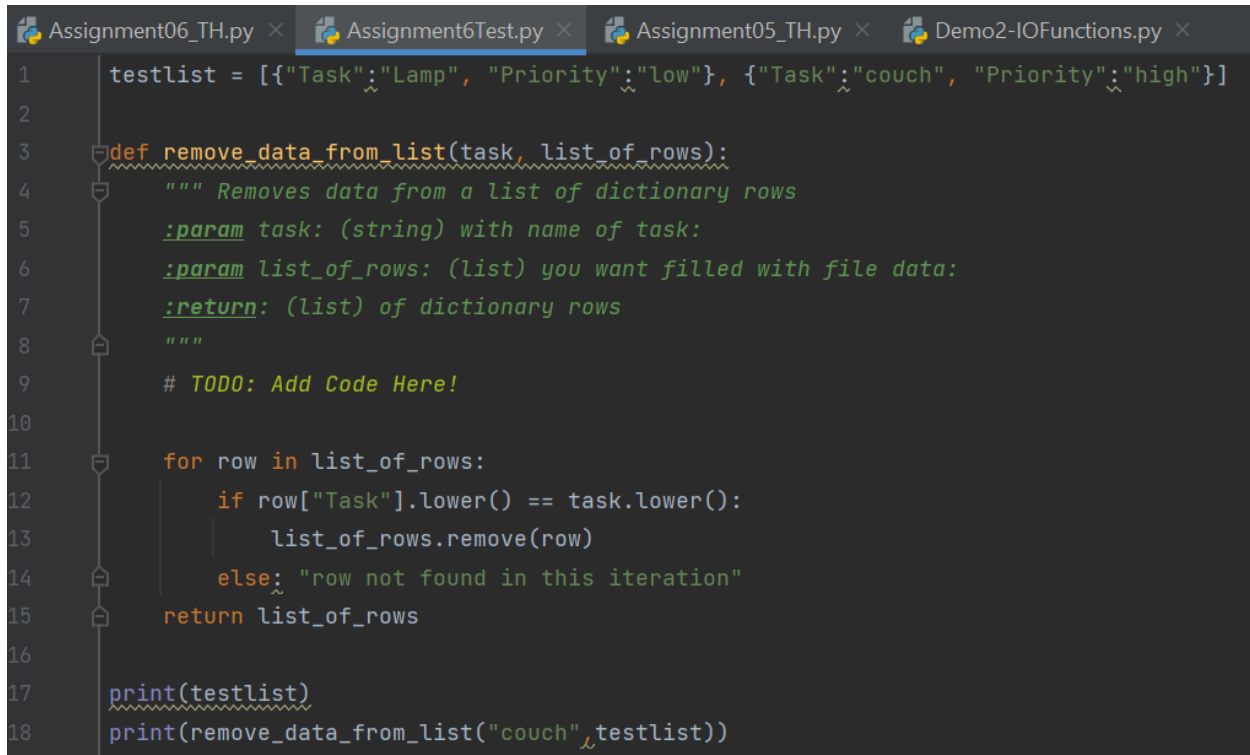
```

Figure 7: Function Processor.add\_data\_to\_list() Test Results

## Processor.remove\_data\_from\_list – Test Function

This function's goal is to match the inputted task with the task in the list for removal. This function returns list\_of\_rows (dictionary) with the inputted task removed.

Similar to assignment 5, a for loop is used to loop through each row in the list while searching for a match between the inputted task and an existing task in the list. If a match is found, the remove function (python default) is used to remove it from the list.

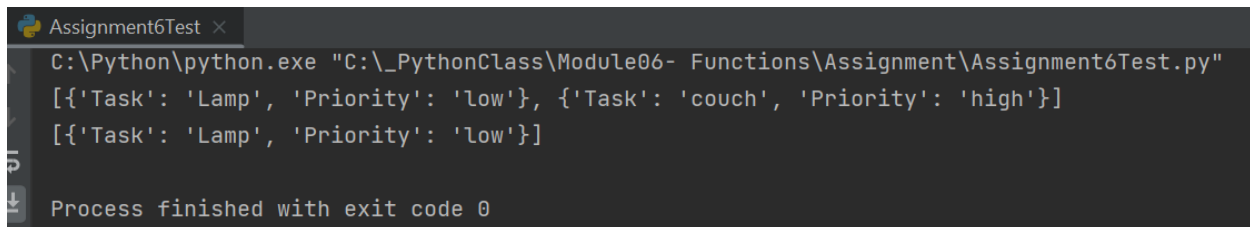


```

1  testlist = [{"Task": "Lamp", "Priority": "low"}, {"Task": "couch", "Priority": "high"}]
2
3  def remove_data_from_list(task, list_of_rows):
4      """ Removes data from a list of dictionary rows
5          :param task: (string) with name of task:
6          :param list_of_rows: (list) you want filled with file data:
7          :return: (list) of dictionary rows
8          """
9      # TODO: Add Code Here!
10
11     for row in list_of_rows:
12         if row["Task"].lower() == task.lower():
13             list_of_rows.remove(row)
14         else: "row not found in this iteration"
15     return list_of_rows
16
17     print(testlist)
18     print(remove_data_from_list("couch", testlist))

```

Figure 8: Function Processor.remove\_data\_from\_list() Test Code



```

Assignment6Test x
C:\Python\python.exe "C:\_PythonClass\Module06- Functions\Assignment\Assignment6Test.py"
[{'Task': 'Lamp', 'Priority': 'low'}, {'Task': 'couch', 'Priority': 'high'}]
[{'Task': 'Lamp', 'Priority': 'low'}]
Process finished with exit code 0

```

Figure 9: Function Processor.remove\_data\_from\_list() Test Results

## Processor.write\_data\_to\_file – Test Function

This function's goal is to write the data to the file with parameters file\_name, and list\_of\_rows. The specified dictionary list is written to the specified file (file\_name). This function code uses a similar code as in assignment 05. It uses a for loop and writes the dictionary list, list\_of\_rows, using key values "Task" and "Priority" along with a carriage return for the next line.

```

1  file_name = 'testfile.txt'
2  list_of_rows = [{"Task": 'hello', "Priority": 'bye'}]
3
4  def write_data_to_file(file_name, list_of_rows):
5      """ Writes data from a list of dictionary rows to a File
6
7      :param file_name: (string) with name of file:
8      :param list_of_rows: (list) you want filled with file data:
9      :return: (list) of dictionary rows
10     """
11     # TODO: Add Code Here!
12
13     objFile = open(file_name, "w") #store file name
14     for row in list_of_rows:
15         objFile.write(str(row["Task"]) + ',' + str(row["Priority"]) + '\n')
16     objFile.close()
17     print("Tasks written to: -", file_name)
18     return list_of_rows
19 write_data_to_file(file_name, list_of_rows)

```

Figure 10: Function Processor.write\_data\_to\_file Test Code

```

Assignment6Test x
C:\Python\python.exe "C:\_PythonClass\Module06- Functions\Assignment\Assignment6Test.py"
Tasks written to: - testfile.txt

Process finished with exit code 0

```

Figure 11: Function Processor.write\_data\_to\_file Test Result

```

1  hello,bye
2

```

Figure 12: Function Processor.write\_data\_to\_file Text File

## Overall Script Testing

The overall test plan involves testing each menu option. A text file was created with 1 line of task/priority of Walk,High to start. The script was tested in both PyCharm and Windows CMD.

```

What is the Task? - cook
What is the priority? [high|low] - high
***** The current tasks ToDo are: *****
Walk (High)
aaa (bbb)
cook (high)
*****

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

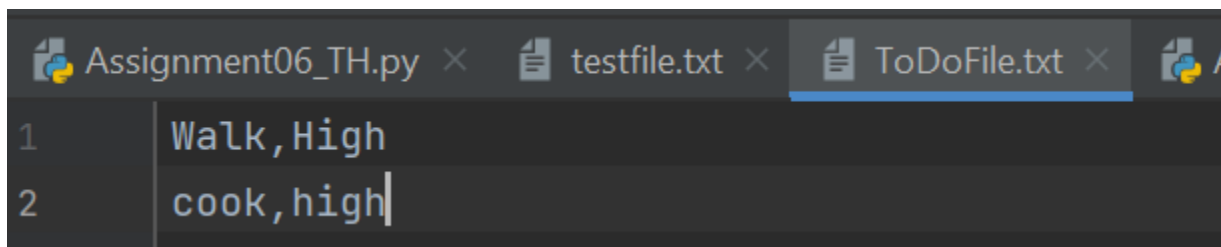
Which option would you like to perform? [1 to 4] - 2

What task do you want to remove? - aaa

***** The current tasks ToDo are: *****
Walk (High)
cook (high)
*****

```

Figure 13: Addition and Removal of Tasks



The screenshot shows a text editor window with the file 'ToDoFile.txt' open. The editor contains two lines of text: 'Walk, High' on line 1 and 'cook, high' on line 2. The cursor is positioned at the end of the second line.

Figure 14: Text File



```

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

What is the Task? - aaa
What is the priority? [high|low] - bbb
***** The current tasks ToDo are: *****
Walk (High)
cook (high)
aaa (bbb)
*****

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 task do you want to remove? - aaa

***** The current tasks ToDo are: *****
Walk (High)
cook (high)
*****

```

Figure 15: Windows CMD Verification

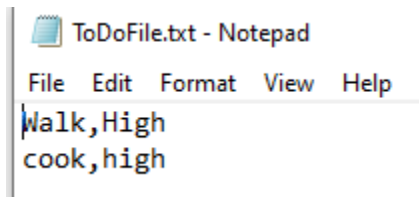


Figure 16: Windows CMD – Text File

## Summary

This week's assignment improved on assignment 5's script where a task and priority are inputted and stored by the user. The difference with this week is the encapsulation of the various code sections in assignment 5 into functions and function classes for basic I/O processing and actual data processing. The majority of the script was created by another person and therefore it was important to understand the logic flow of the original creator in order to plan out how to complete the various functions. I also learned it is very tough to get married and finish assignments at the same time.