James Henderson

8/10/2022

IT FDN 110 B

Assignment05

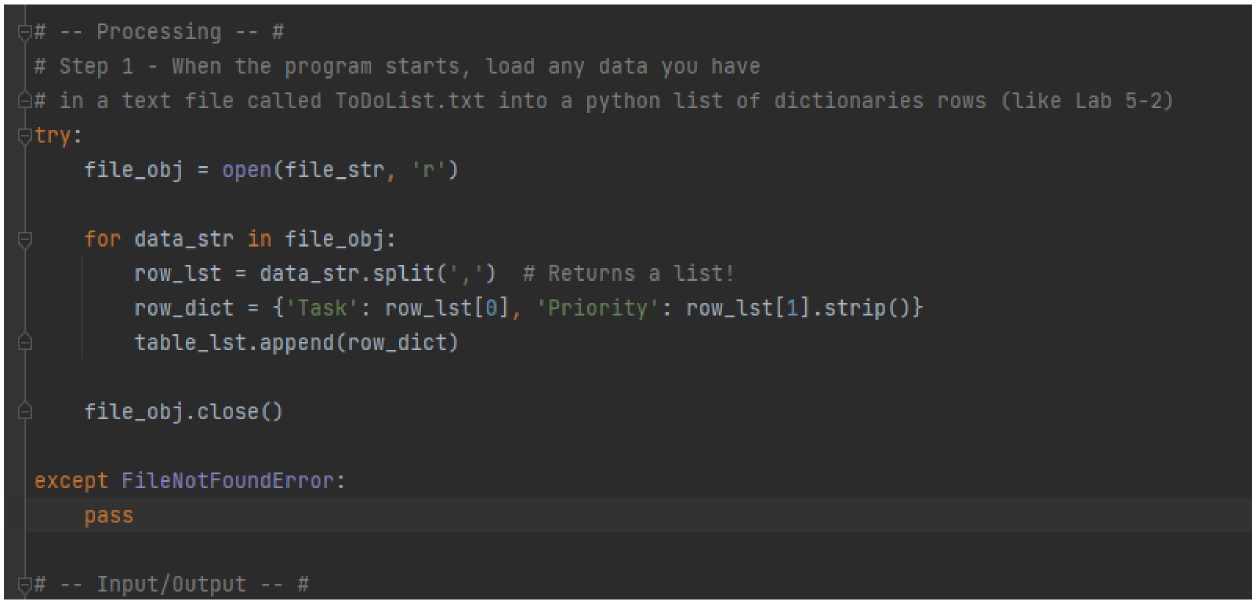
Assignment 5 – Task List Script

Introduction

For this assignment, I added a folder to my PyCharm project on my Windows computer. Within the project folder, I copied a script, Assignment05_Starter.py, from our class files. I then modified the script to make a fully functioning program. The script captures input from the user to make a list of tasks, along with an assigned priority for each task, and stores it in a file.

Task List: Processing

To accomplish the goals of this assignment, I added code to make the script run correctly. First, the program attempts to read in / pick up existing task data from a text file in the same folder as the script, "ToDoList.txt". If the file doesn't exist when the user runs the script, that's OK – no data will be picked up to start with. This potential error is managed with a **try / except** framework (***Figure 1***):



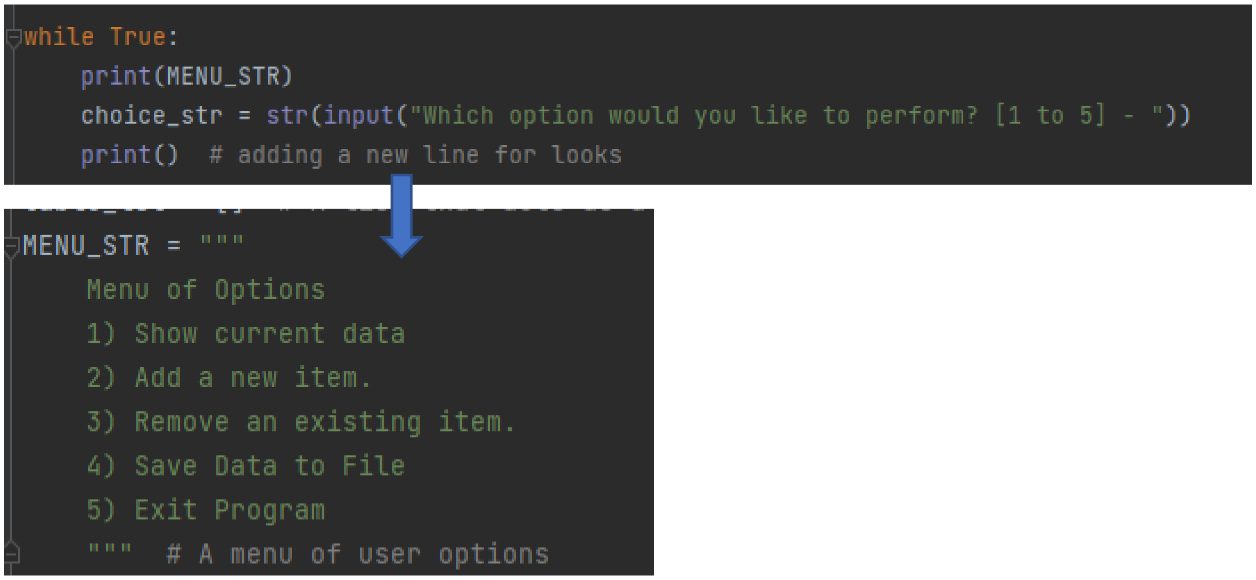
```
# -- Processing -- #
# Step 1 - When the program starts, load any data you have
# in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)
try:
    file_obj = open(file_str, 'r')

    for data_str in file_obj:
        row_lst = data_str.split(',')  # Returns a list!
        row_dict = {'Task': row_lst[0], 'Priority': row_lst[1].strip()}
        table_lst.append(row_dict)

    file_obj.close()

except FileNotFoundError:
    pass

# -- Input/Output -- #
```

Figure 1 – Reading in data to start – catching a potential missing file error with try / except

If the file does exist, the script reads in each line of data from the file (must be comma-delimited) and loads the data into Dictionary objects, which are then appended to a list (**table_lst)**. Note that newline characters from the text file are explicitly stripped out as the data is read in.

Task List: Input / Output

The main body of the interactive program is contained within a **while** loop. As shown in ***Figure 2,*** the user's presented with a menu of choices and must enter a selection (5 is to exit). I moved the menu text to the top of the script and defined it as a constant to keep this section of the code more readable.



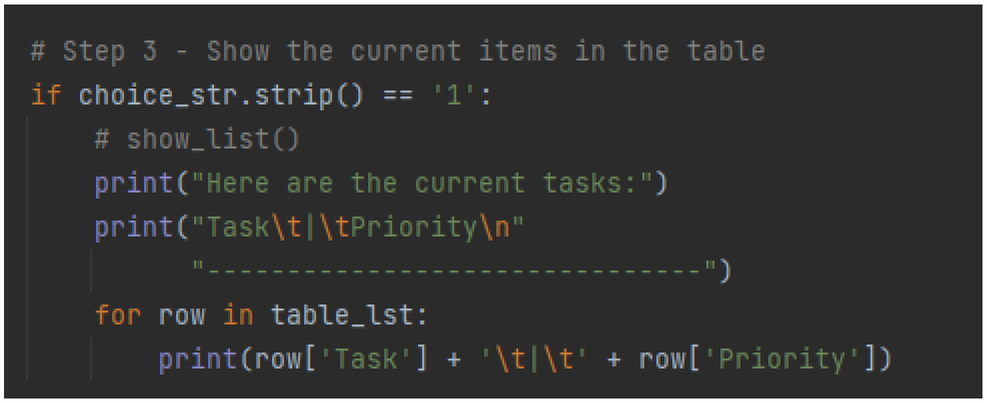
```
while True:
    print(MENU_STR)
    choice_str = str(input("Which option would you like to perform? [1 to 5] - "))
    print()  # adding a new line for looks
```

```
MENU_STR = """
    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
    """  # A menu of user options
```

Figure 2 – Moving the menu text to the top of the code and writing in uppercase to indicate it's a constant rather than a variable improves readability

Showing Current Tasks

Within the body of the input / output section, the user can choose to display the current items in the table. These are printed out with a bit of formatting by looping through the Dictionary objects within the List of tasks held in memory (***Figure 3***). I avoided using a function to accomplish this, since we are supposed to wait for Module 6 to implement the code with functions.

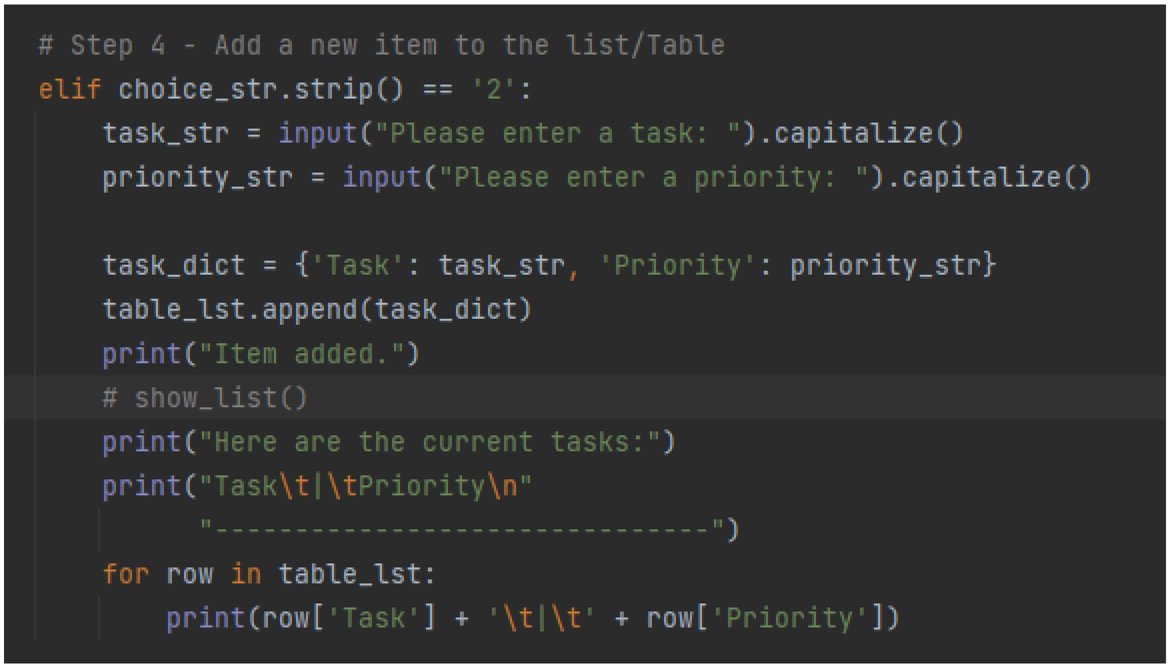


```
# Step 3 - Show the current items in the table
if choice_str.strip() == '1':
    # show_list()
    print("Here are the current tasks:")
    print("Task\t|\tPriority\n"
          "------------------------------")
    for row in table_lst:
        print(row['Task'] + '\t|\t' + row['Priority'])
```

Figure 3 – printing out the current list of tasks

Entering a New Task

If the user chooses to enter a new task, they are prompted to enter the name of the task and its priority. Both of these text inputs are converted to capitalized strings after the user enters them. The input data is used to construct a new Dictionary item, which is added to the main list of tasks in memory (**table_lst**). After the task is added, the script lets the user know and displays the updated list of tasks ***(figure 5).***

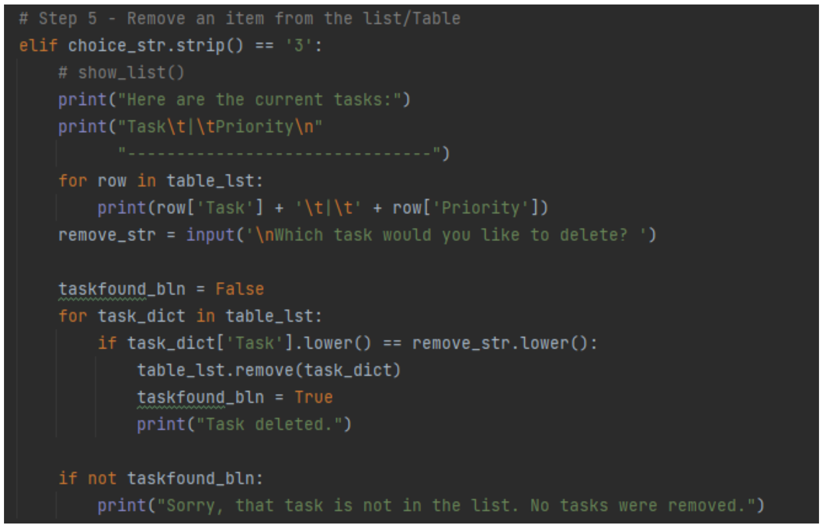


```
# Step 4 - Add a new item to the list/Table
elif choice_str.strip() == '2':
    task_str = input("Please enter a task: ").capitalize()
    priority_str = input("Please enter a priority: ").capitalize()

    task_dict = {'Task': task_str, 'Priority': priority_str}
    table_lst.append(task_dict)
    print("Item added.")
    # show_list()
    print("Here are the current tasks:")
    print("Task\t|\tPriority\n"
          "-------------------------------")
    for row in table_lst:
        print(row['Task'] + '\t|\t' + row['Priority'])
```

Figure 5 – adding a new item to the list

Removing a Task

If the user instead chooses to remove an item from the list, they are shown the current tasks in the list and prompted for which one to remove. The user must then enter a String matching (not case-sensitive) the name of one of the tasks in the list. If there is no match, the user is notified and returned to the main menu. However, if the user's input matches a task from the list, that task is removed and the user is notified that the task was deleted.

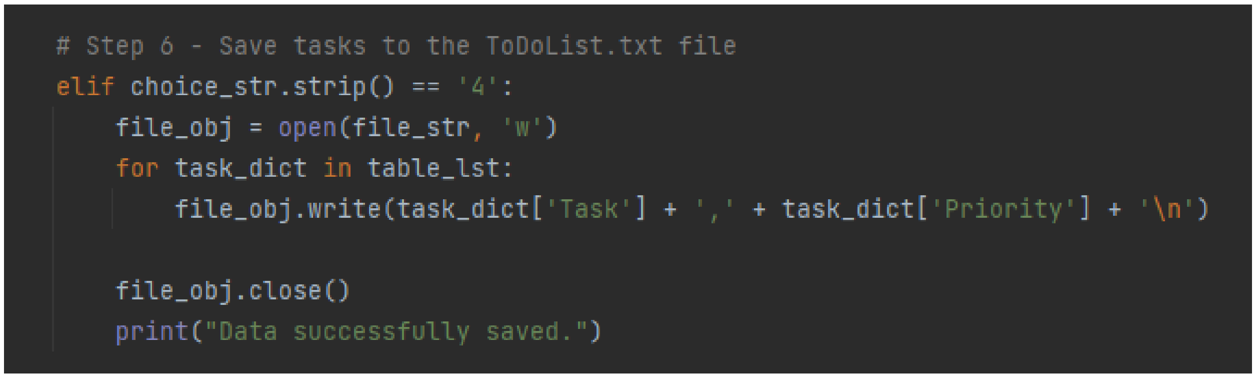


```
# Step 5 - Remove an item from the list/Table
elif choice_str.strip() == '3':
    # show_list()
    print("Here are the current tasks:")
    print("Task\t|\tPriority\n"
          "-------------------------------")
    for row in table_lst:
        print(row['Task'] + '\t|\t' + row['Priority'])
    remove_str = input('\nWhich task would you like to delete? ')

    taskfound_bln = False
    for task_dict in table_lst:
        if task_dict['Task'].lower() == remove_str.lower():
            table_lst.remove(task_dict)
            taskfound_bln = True
            print("Task deleted.")

    if not taskfound_bln:
        print("Sorry, that task is not in the list. No tasks were removed.")
```

Figure 6 – removing a task from the list

Saving Tasks to a File

If the user chooses to save the tasks to the ToDoList.txt file, the text file is opened in “write” mode, rather than “append” mode. This is because if there were any existing tasks to start with, they would have been loaded into memory at the beginning of the script, and would be written to the file again (unless the user removed them while running the program). Therefore, it’s OK to replace the entire contents of the file. The script iterates through the Dictionaries in the main list and writes out the tasks with their priorities in a comma-delimited format (***Figure 7***).

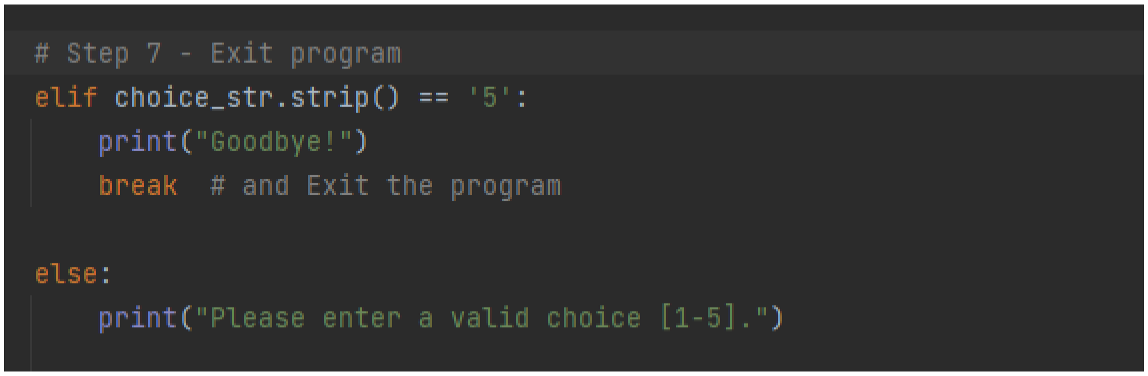


```
# Step 6 - Save tasks to the ToDoList.txt file
elif choice_str.strip() == '4':
    file_obj = open(file_str, 'w')
    for task_dict in table_lst:
        file_obj.write(task_dict['Task'] + ',' + task_dict['Priority'] + '\n')

    file_obj.close()
    print("Data successfully saved.")
```

Figure 7 – writing data out to the ToDoList.txt file

Stopping the Program / Catching Invalid Menu Choices

When the user decides to stop running the program, they just need to enter “5” from the main menu, and the program bids them farewell. If the user ever makes an invalid selection from the main menu, they are reminded of the valid choices and the main menu is presented again (***Figure 8****).*



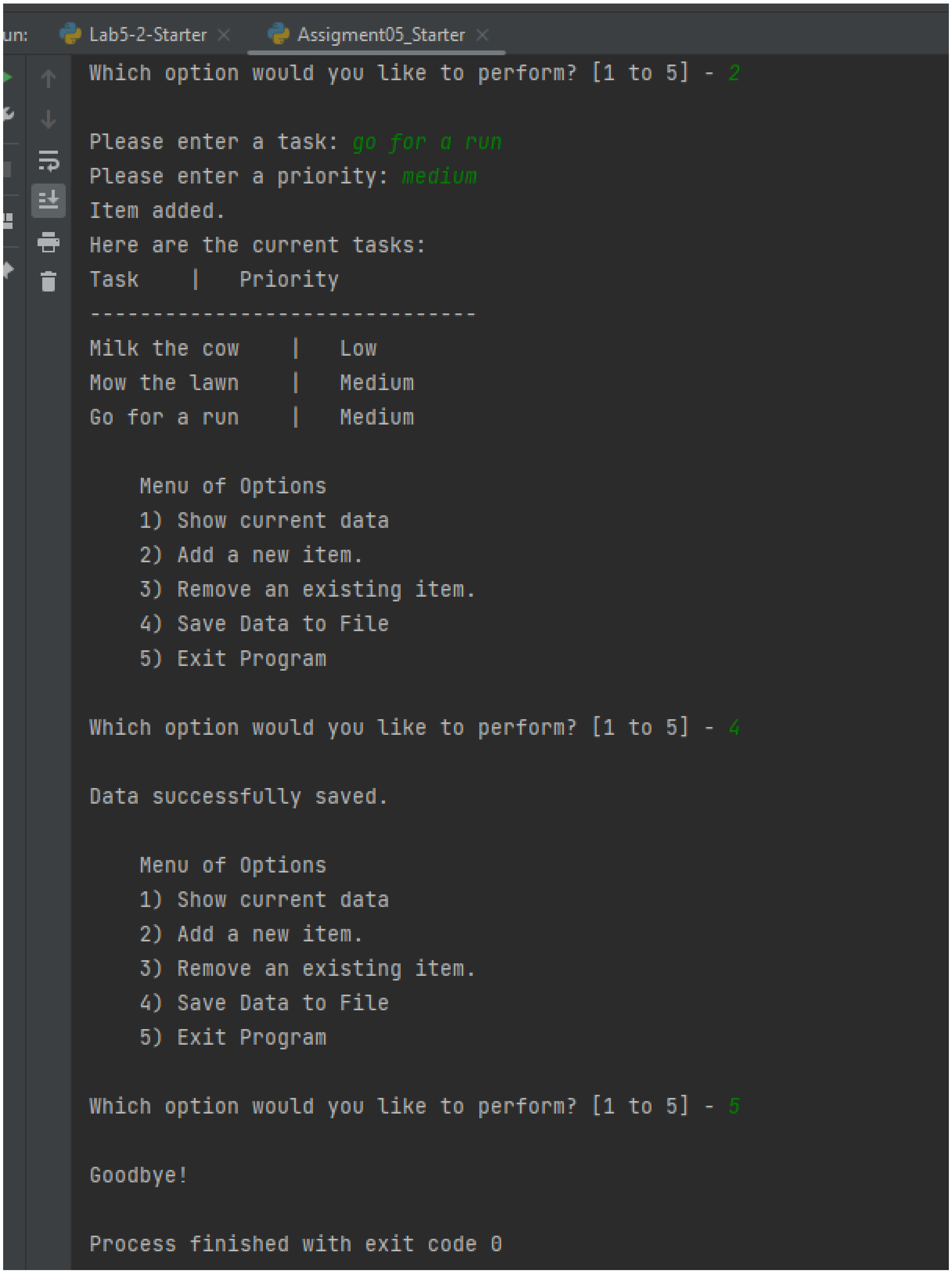
```
# Step 7 - Exit program
elif choice_str.strip() == '5':
    print("Goodbye!")
    break  # and Exit the program

else:
    print("Please enter a valid choice [1-5].")
```

Figure 8 – exiting the program and checking for invalid menu choices

Task List: Running the Script

To test the script, I first ran the Assignment05_Starter.py file within PyCharm, using the latest available Python interpreter on my machine (Python 3.10).

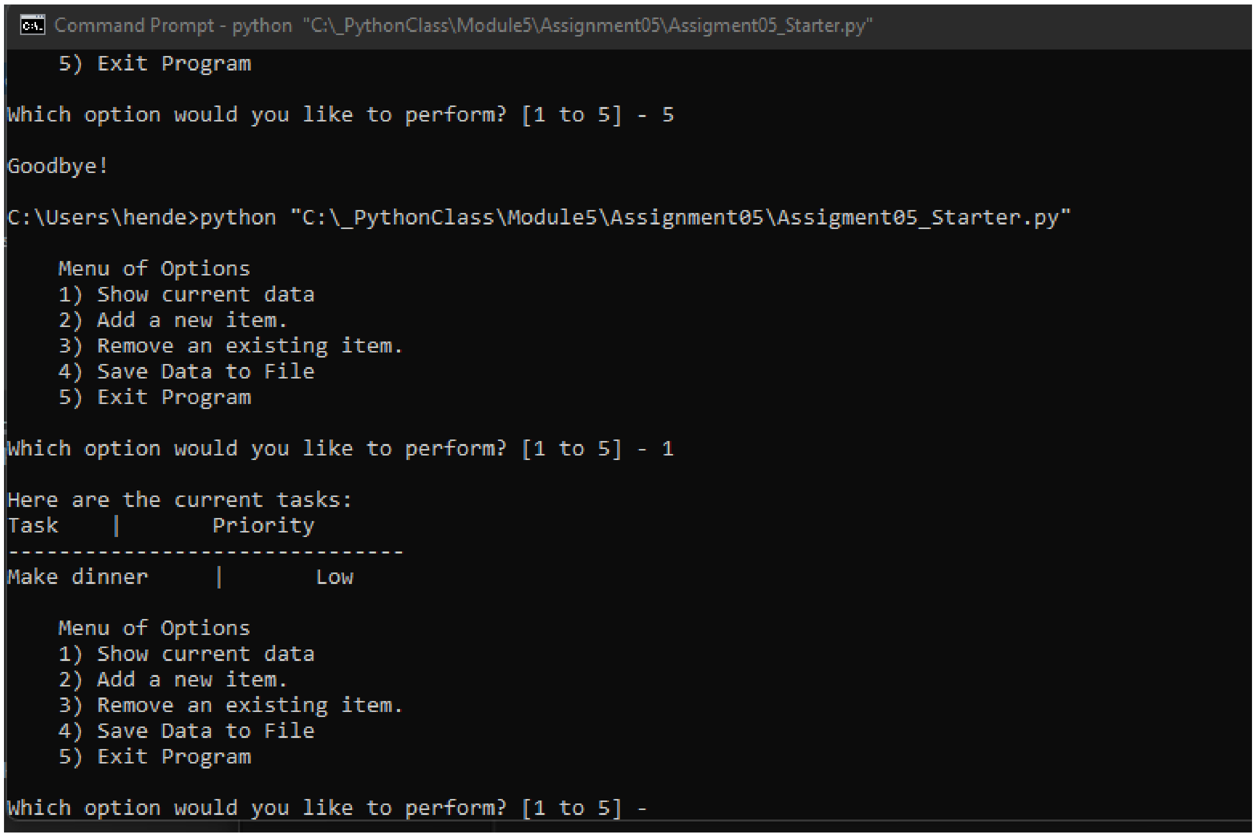
You can see part of the script’s input / output in process in ***Figure 9*** below***.***



```
Which option would you like to perform? [1 to 5] - 2

Please enter a task: go for a run
Please enter a priority: medium
Item added.
Here are the current tasks:
Task    |   Priority
-------------------------------
Milk the cow     |    Low
Mow the lawn     |    Medium
Go for a run     |    Medium

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program

Which option would you like to perform? [1 to 5] - 4

Data successfully saved.

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program

Which option would you like to perform? [1 to 5] - 5

Goodbye!

Process finished with exit code 0
```

Figure 9 – running the script in PyCharm

I also ran the script from the Windows Command Prompt window (***Figure 10).*** I verified that data from a previous run of the script was correctly picked up at the start of the program.



```
Command Prompt - python  "C:\_PythonClass\Module5\Assignment05\Assigment05_Starter.py"
    5) Exit Program

Which option would you like to perform? [1 to 5] - 5

Goodbye!

C:\Users\hende>python "C:\_PythonClass\Module5\Assignment05\Assigment05_Starter.py"

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program

Which option would you like to perform? [1 to 5] - 1

Here are the current tasks:
Task    |       Priority
-------------------------------
Make dinner     |       Low

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program

Which option would you like to perform? [1 to 5] -
```

Figure 10 – running the script from the Windows command prompt

Summary

In this assignment, I used variables to store user input and then store a list of tasks in memory using a variety of objects, like Lists and Dictionaries. I also wrote out data from the program's memory into a text file and read data back in whenever the script was run.