Brandon Raupe
5/14/2023
Foundations of Programming: Python
Assignment 05
https://github.com/braupe1/Assignment05

Python Script: Task List

The Problem

The request of Assignment 5 is to build a script that will accept an entry for a task and a priority of the task. This program will consist of dictionary value pairs at points in the script but must also be translated into and from list tables.

The script will involve five potential selection options that include 1) Show current data, 2) Add a new item, 3) Remove an existing item, 4) Save Data to File, and 5) Exit the program. Each option must be part of an if statement that will include various conversion steps within each segment.

The ToDoList.txt file will display the current results of the entries or edits. The script must run not only within the IDE but the command prompt as well.

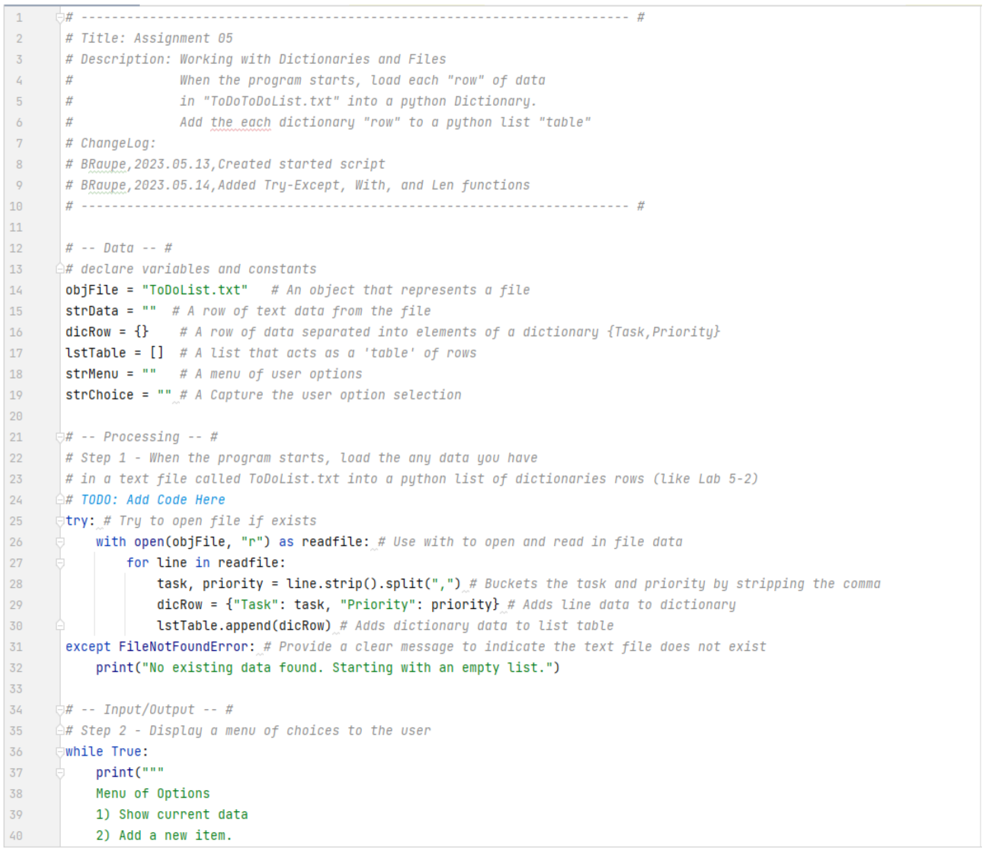
The Script

The Assignment05 script (Figures 1.1, 1.2, 1.3) contains some default segments, as provided by Professor Root. However, this submission will focus only on items from the "TODO" segments.

The below bullet points outline the essential components of the added items in the script:

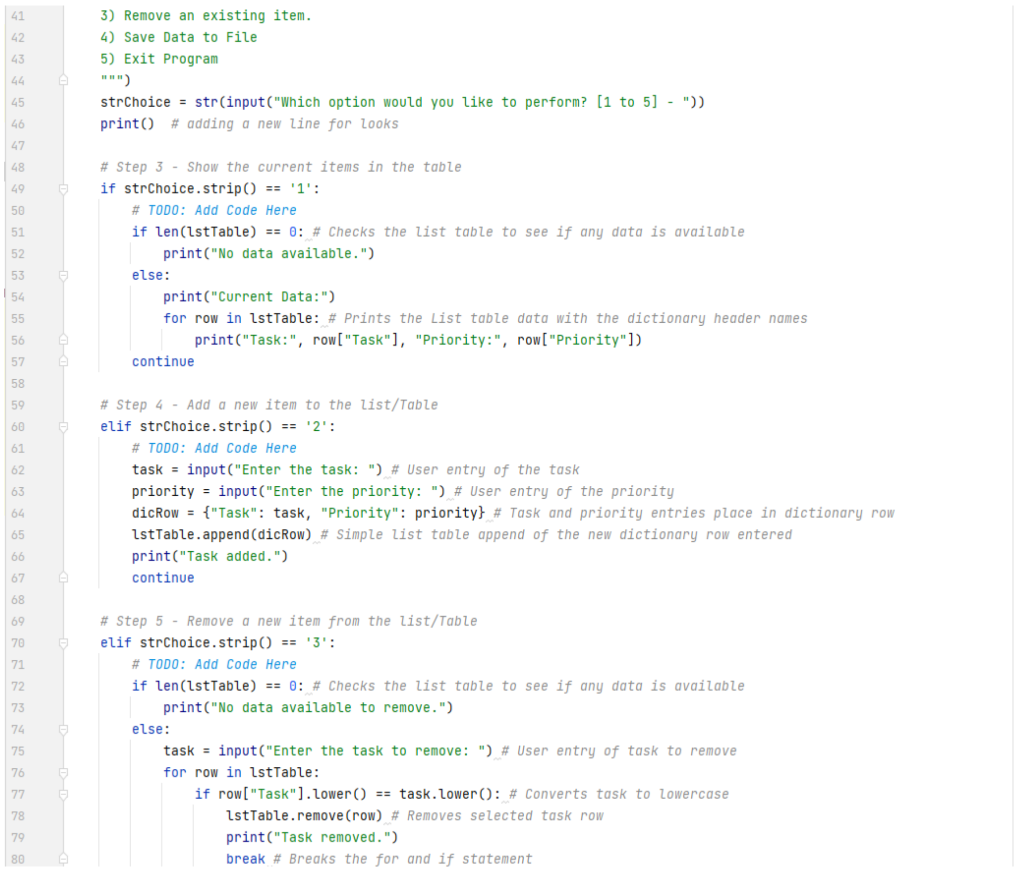
- **Step 1 TODO: (Lines 25-32)**
    - Try and Except in place to provide additional insights if the objFile does not exist.
        - The except portion uses FileNotFoundError with a printed message to indicate that the file was beginning with an empty list.
    - The "with" function was applied to open the file, then read the file and provide a separate name for the for loop.
    - The for loop separates the lines for the read file and then places the task and priority names as it strips and splits the line. This is followed by applying task and priority to the dicRow variable before finally appending the dicRow to the stable.
- **Step 3 TODO: (Lines 51-56)**
    - The show existing items in the tables begin with an if statement combined with a "len" function to ensure the table has data. In contrast, if no data is found, a print function of "No data available" is displayed.
    - The else portion of the statement prints "Current Data:" followed by a for loop the parses out the rows of the lstTable and prints them out as it might be displayed within a dictionary, but does not actually save as a dictionary in this step.
- **Step 4 TODO: (Lines 62-66)**

    - The first line requests a task to be entered, followed by the second line requesting a priority input.
    - The “dicRow” creates a dictionary row of data.
    - The dicRow is then appended to the lstTable.
    - A simple print statement to indicate the task has been added ends the operation.
- **Step 5 TODO: (Lines72-80)**
    - Item removal begins by first checking if any data exists within the table. If no data exists, a print message indicates to the user that no data can be removed.
    - The else statement accepts input from a user task.
    - A for loop then begins looping through the lstTable and adds an if statement that if the lowercase value entered equals the lowercase value.
    - If the task entered for removal equals a task in the current lstTable, it will be removed.
    - The for loop ends by printing that the task was removed, followed by a break function.
- **Step 6 TODO: (Lines 86-89)**
    - Another with statement is in place to open the file to write mode.
    - The for loop grabs each row in the lstTable then writes each row in list table format, followed by an indication that the data was saved to file.
- **Step 7 TODO: (Line 95)**
    - A simple print statement lets the user know that the program has ended, followed by an immediate break.



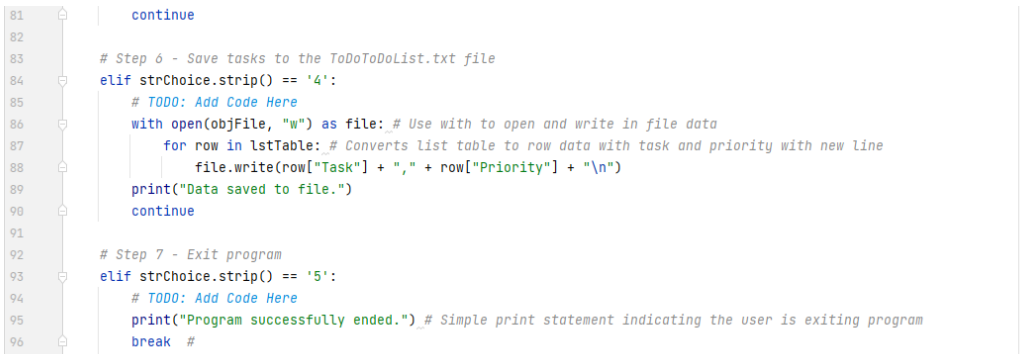
```
1   # ---------------------------------------------------------------------- #
2   # Title: Assignment 05
3   # Description: Working with Dictionaries and Files
4   #              When the program starts, load each "row" of data
5   #              in "ToDoToDoList.txt" into a python Dictionary.
6   #              Add the each dictionary "row" to a python list "table"
7   # ChangeLog:
8   # BRaupe,2023.05.13,Created started script
9   # BRaupe,2023.05.14,Added Try-Except, With, and Len functions
10  # ---------------------------------------------------------------------- #
11
12  # -- Data -- #
13  # declare variables and constants
14  objFile = "ToDoList.txt"   # An object that represents a file
15  strData = ""  # A row of text data from the file
16  dicRow = {}    # A row of data separated into elements of a dictionary {Task,Priority}
17  lstTable = []  # A list that acts as a 'table' of rows
18  strMenu = ""   # A menu of user options
19  strChoice = "" # A Capture the user option selection
20
21  # -- Processing -- #
22  # Step 1 - When the program starts, load the any data you have
23  # in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)
24  # TODO: Add Code Here
25  try: # Try to open file if exists
26      with open(objFile, "r") as readfile: # Use with to open and read in file data
27          for line in readfile:
28              task, priority = line.strip().split(",") # Buckets the task and priority by stripping the comma
29              dicRow = {"Task": task, "Priority": priority} # Adds line data to dictionary
30              lstTable.append(dicRow) # Adds dictionary data to list table
31  except FileNotFoundError: # Provide a clear message to indicate the text file does not exist
32      print("No existing data found. Starting with an empty list.")
33
34  # -- Input/Output -- #
35  # Step 2 - Display a menu of choices to the user
36  while True:
37      print("""
38      Menu of Options
39      1) Show current data
40      2) Add a new item.
```

Figure 1.1: The Assignment05.py script as it appears in PyCharm (Part 1)



```
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
    """)
    strChoice = str(input("Which option would you like to perform? [1 to 5] - "))
    print()  # adding a new line for looks

    # Step 3 - Show the current items in the table
    if strChoice.strip() == '1':
        # TODO: Add Code Here
        if len(lstTable) == 0: # Checks the list table to see if any data is available
            print("No data available.")
        else:
            print("Current Data:")
            for row in lstTable: # Prints the List table data with the dictionary header names
                print("Task:", row["Task"], "Priority:", row["Priority"])
        continue

    # Step 4 - Add a new item to the list/Table
    elif strChoice.strip() == '2':
        # TODO: Add Code Here
        task = input("Enter the task: ") # User entry of the task
        priority = input("Enter the priority: ") # User entry of the priority
        dicRow = {"Task": task, "Priority": priority} # Task and priority entries place in dictionary row
        lstTable.append(dicRow) # Simple list table append of the new dictionary row entered
        print("Task added.")
        continue

    # Step 5 - Remove a new item from the list/Table
    elif strChoice.strip() == '3':
        # TODO: Add Code Here
        if len(lstTable) == 0: # Checks the list table to see if any data is available
            print("No data available to remove.")
        else:
            task = input("Enter the task to remove: ") # User entry of task to remove
            for row in lstTable:
                if row["Task"].lower() == task.lower(): # Converts task to lowercase
                    lstTable.remove(row) # Removes selected task row
                    print("Task removed.")
                    break # Breaks the for and if statement
```

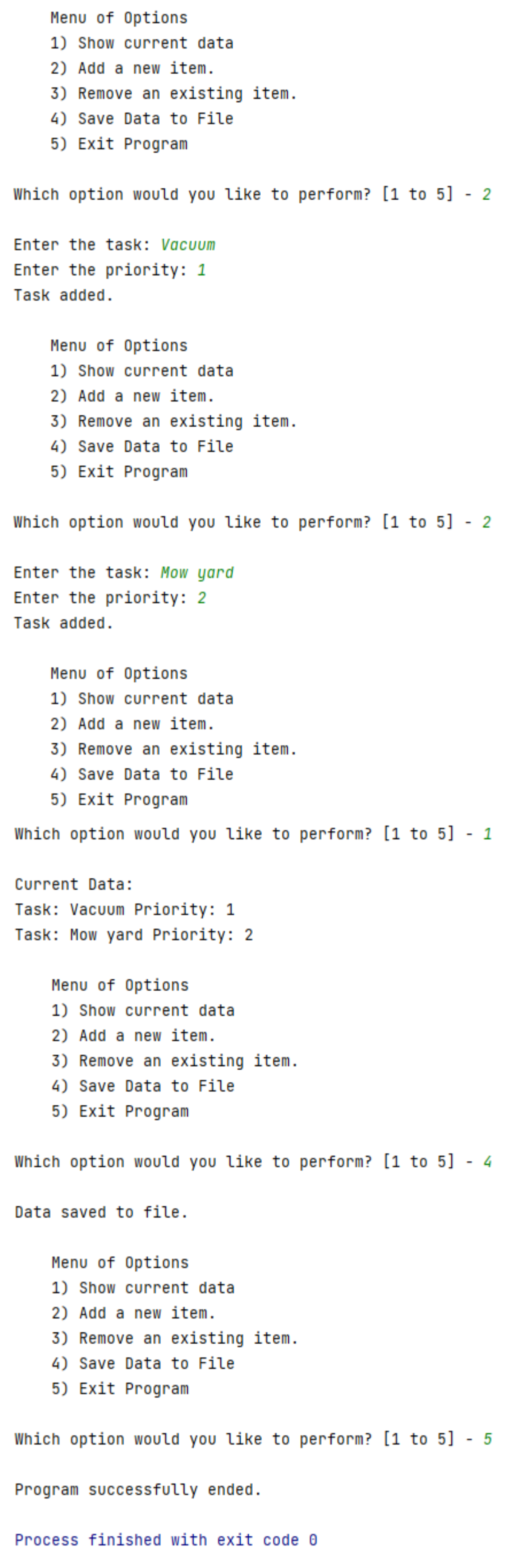
Figure 1.2: The Assignment05.py script as it appears in PyCharm (Part 2)



```
        continue

    # Step 6 - Save tasks to the ToDoToDoList.txt file
    elif strChoice.strip() == '4':
        # TODO: Add Code Here
        with open(objFile, "w") as file: # Use with to open and write in file data
            for row in lstTable: # Converts list table to row data with task and priority with new line
                file.write(row["Task"] + "," + row["Priority"] + "\n")
        print("Data saved to file.")
        continue

    # Step 7 - Exit program
    elif strChoice.strip() == '5':
        # TODO: Add Code Here
        print("Program successfully ended.") # Simple print statement indicating the user is exiting program
        break  #
```

Figure 1.3: The Assignment05.py script as it appears in PyCharm (Part 3)

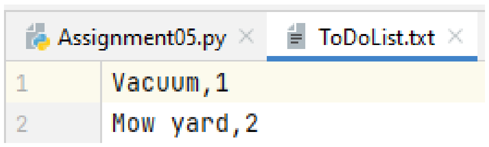
Figure 2 indicates the program being performed with various steps seen in Figure 2.



```
    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] - 2

Enter the task: Vacuum
Enter the priority: 1
Task added.

    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] - 2

Enter the task: Mow yard
Enter the priority: 2
Task added.

    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

Current Data:
Task: Vacuum Priority: 1
Task: Mow yard Priority: 2

    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 saved to file.

    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

Program successfully ended.

Process finished with exit code 0
```

Figure 2: The Assignment05.py script run within PyCharm

Figure 3 displays the results of the ToDoList.txt following the PyCharm operation and entry of two items.

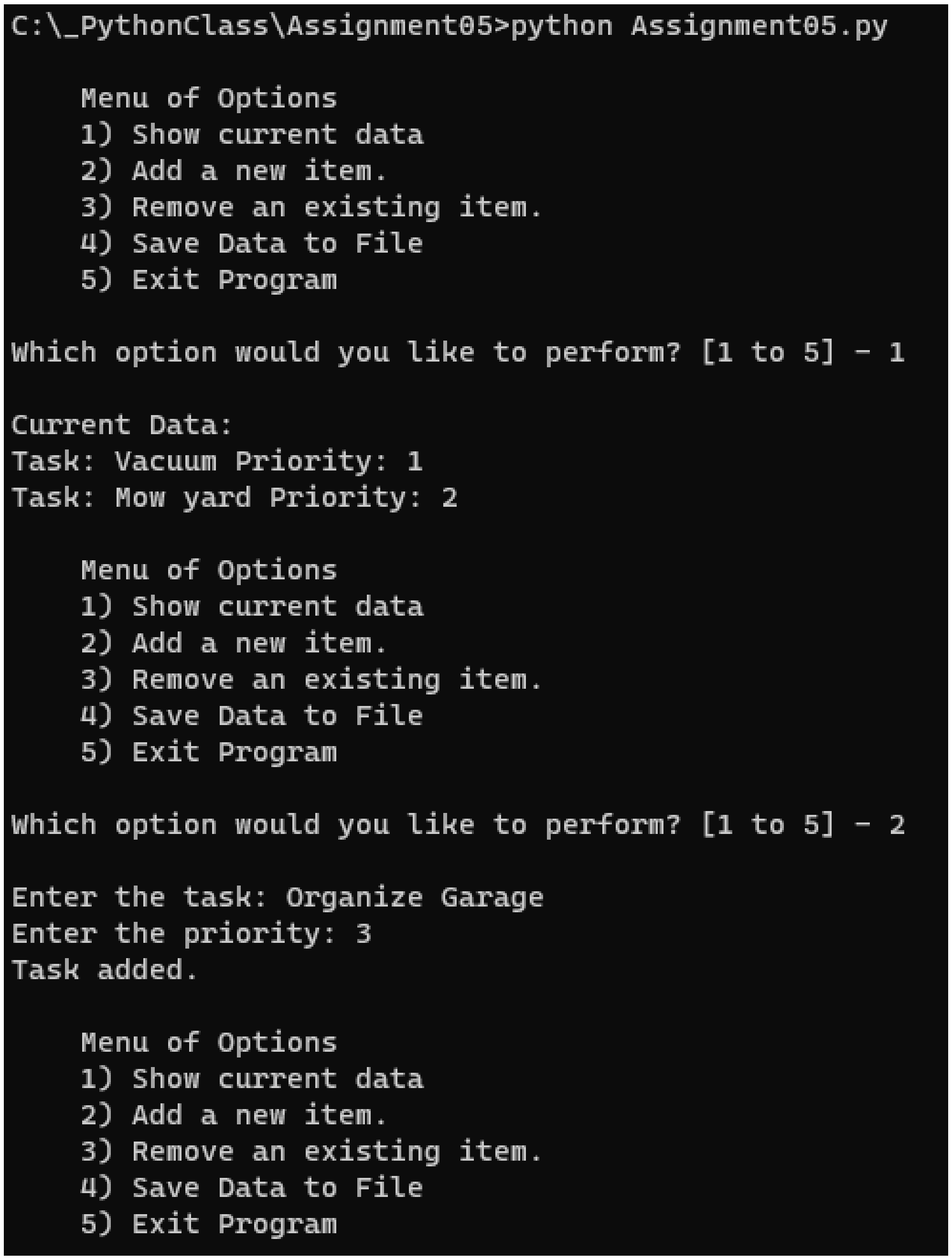


```
Vacuum,1
Mow yard,2
```

Figure 3: The ToDoList.txt file after initial items loaded

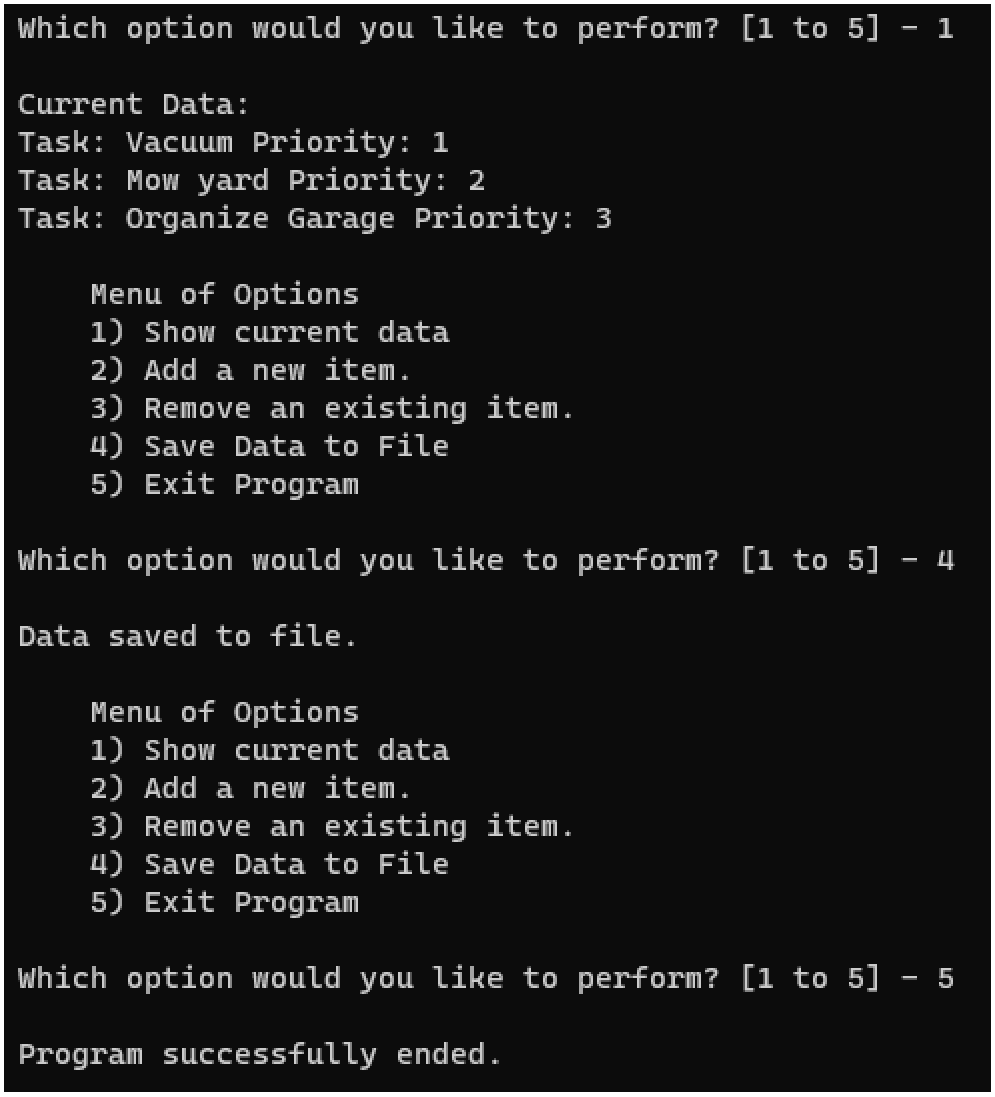
The Results in the Command Prompt

The command prompt code (Figures 4.1 and 4.2) operates as expected, as it would also operate within the PyCharm IDE. During testing, the user enters a single item for entry, and Figure 5 then displays all three added items within the ToDoList.txt file.



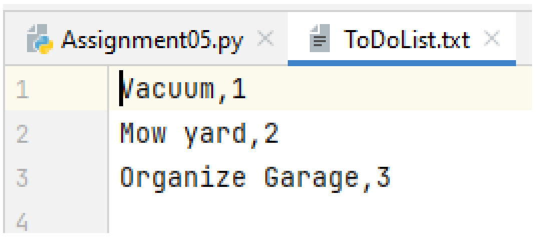
```
C:\_PythonClass\Assignment05>python Assignment05.py

    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

Current Data:
Task: Vacuum Priority: 1
Task: Mow yard Priority: 2

    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] - 2

Enter the task: Organize Garage
Enter the priority: 3
Task added.

    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
```

Figure 4.1: The Assigment05.py being run within the command prompt (Part 1)



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

Current Data:
Task: Vacuum Priority: 1
Task: Mow yard Priority: 2
Task: Organize Garage Priority: 3

    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 saved to file.

    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

Program successfully ended.
```

Figure 4.1: The Assigment05.py being run within the command prompt (Part 2)



```
Vacuum,1
Mow yard,2
Organize Garage,3
```

Figure 5: The ToDoList.txt file after command prompt item added with the PyCharm run items

Summary

Although partially provided, this script certainly provided a solid challenge for my current skills. I was able to prepare a skeleton script initially but was running into some issues identifying the best part to read and write the objFile into each segment as necessary.

I consulted with the GPT tool to offer suggestions, and then it clicked on the few remaining holes within my script. Additionally, I learned a few more things with my code to enhance the overall performance.

One of those issues was creating some guardrails if the “lstTable” did not exist. When GPT suggested adding the “len” functionality to offer an enhanced error message if the table did not exist, an appropriate error message regarding the “ToDoList.txt” could be prompted in this scenario.

After reviewing my final script for submission, I do realize there are at least a couple areas where better error handling could occur, and I will keep in mind as we move forward.