Chase Richardson

May 15, 2023

Foundations of Programming: Python

Assignment 05

https://github.com/jcrich13/IntroToProg-Python

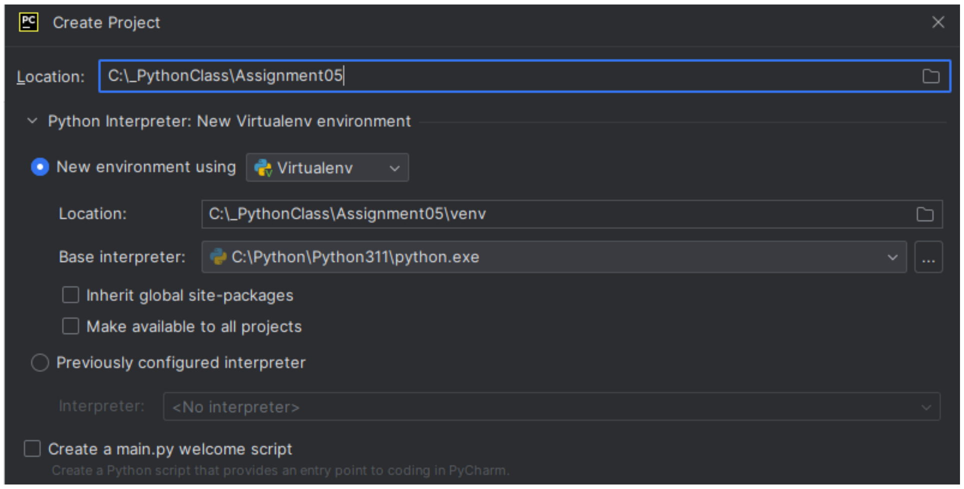
Creating a Python Script for To Do List

Introduction

The primary purpose of Assignment 05 is to modify an existing Python script that maintains the names of various tasks and priorities. It will store both pieces of data in a two-dimensional list, where each task and priority are a dictionary row of data and each dictionary row is part of a table of data. It will continue to ask the user to show data, add data, remove data, or save data until they exit the program. The information will be stored in a text file called "ToDoFile.txt." Lastly, a printed "menu" will guide the user through this process. In order to successfully complete this task, I utilized the lessons learned from Assignments 04 pertaining to lists and loops. Furthermore, I reviewed the course material by modules, book, and other internet sources provided within the course to learn more about dictionaries and GitHub. The information provided below outlines the steps I completed in order to create Assignment 05, To Do List script.

Getting Started

To modify the To Do List Python script in Assignment 05, I opened the PyCharm Integrated Development Environment (IDE) and created a brand-new project. When creating the Project, I assigned the location to be underneath the _PythonClass and Assignment05 subfolders within the C:\, created a new virtual environment, and unchecked the “Create a main.py welcome script” to avoid redundancy in the assignment folder (Figure 1).



Location: C:_PythonClass\Assignment05

Python Interpreter: New Virtualenv environment

New environment using Virtualenv

Location: C:_PythonClass\Assignment05\venv

Base interpreter: C:\Python\Python311\python.exe

Inherit global site-packages

Make available to all projects

Previously configured interpreter

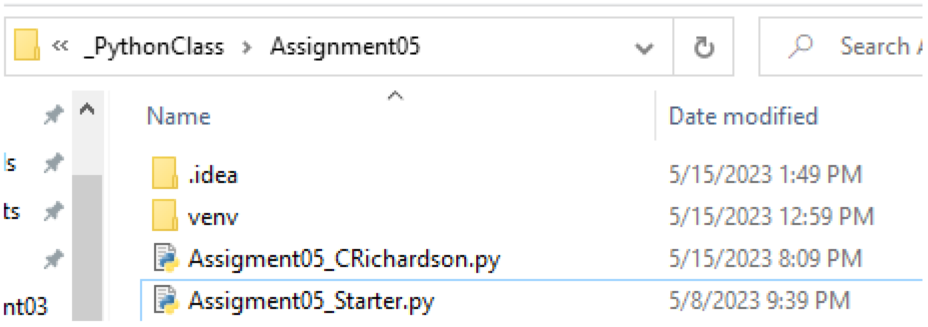
Interpreter: <No interpreter>

Create a main.py welcome script

Create a Python script that provides an entry point to coding in PyCharm.

Figure 1: Creating a New Project within PyCharm for Assignment05

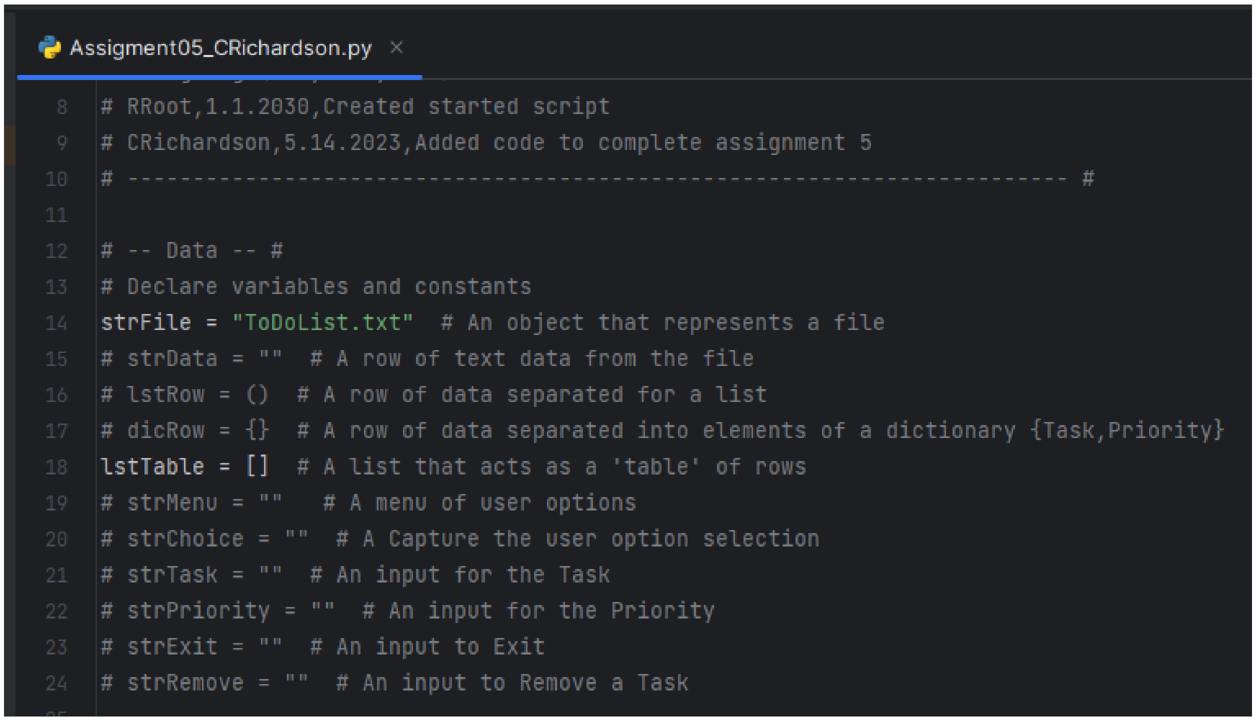
Since there was a starter script available I copied the "Assigment05_Starter.py" file into the Assignment05 subfolder to start. And then I created a copy of the "Assigment05_Starter.py" and renamed it to be called "Assigment05_CRichardson.py" to complete the subsequent steps. I kept both copies in the same location for historical/revision purposes in the event I needed to reference the original version or in the event I needed to start over. I figured this would be a good practice to maintain going forward throughout the course (Figure 2).



```
« _PythonClass › Assignment05

Name                          Date modified
.idea                         5/15/2023 1:49 PM
venv                          5/15/2023 12:59 PM
Assigment05_CRichardson.py    5/15/2023 8:09 PM
Assigment05_Starter.py        5/8/2023 9:39 PM
```

Figure 2: Assignment05 Python scripts

Once I opened the file in PyCharm, I added my name and date to the Developer Notes section. I followed the recommendation to declare the key variables at the beginning of the assignment and provided detailed commented descriptions for other variables covered in the below sections. I kept any prior existing variables in this section even if they were no longer being referenced (Figure 3).



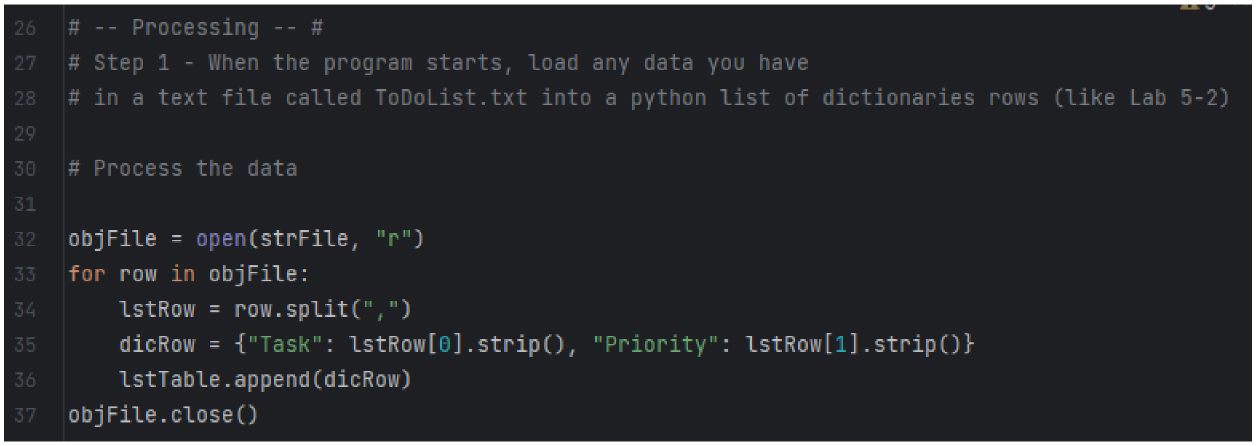
Assigment05_CRichardson.py

```
# RRoot,1.1.2030,Created started script
# CRichardson,5.14.2023,Added code to complete assignment 5
# ------------------------------------------------------------------------ #

# -- Data -- #
# Declare variables and constants
strFile = "ToDoList.txt"  # An object that represents a file
# strData = ""  # A row of text data from the file
# lstRow = ()  # A row of data separated for a list
# dicRow = {}  # A row of data separated into elements of a dictionary {Task,Priority}
lstTable = []  # A list that acts as a 'table' of rows
# strMenu = ""   # A menu of user options
# strChoice = ""  # A Capture the user option selection
# strTask = ""  # An input for the Task
# strPriority = ""  # An input for the Priority
# strExit = ""  # An input to Exit
# strRemove = ""  # An input to Remove a Task
```

Figure 3: Declaring Variables

Processing the Data

Before completing the Processing the Data section, I created a brand-new notepad file called "ToDoList.txt" and added a single record "row" to ensure the script will run successfully, "Homework,High". Once the notepad file was created, I set an objFile variable to open the "ToDoList.txt" file variable in read mode. To account for multiple records in the file I used a "for" loop to first read the existing data as a list and to recognize splits with comma separated values. Furthermore, I created a separate variable that will change the list row to a dictionary row and turn the information into a list table that appends dictionary rows going forward. Lastly, I closed the file per best practices. The course textbook states, "(Dictionary) Values don't have to be unique. Also (dictionary) values can be mutable or immutable" (Dawson, 147). Dictionaries provide a lot more flexibility within Python than traditional lists. The Processing the Data section is captured below (Figure 4).

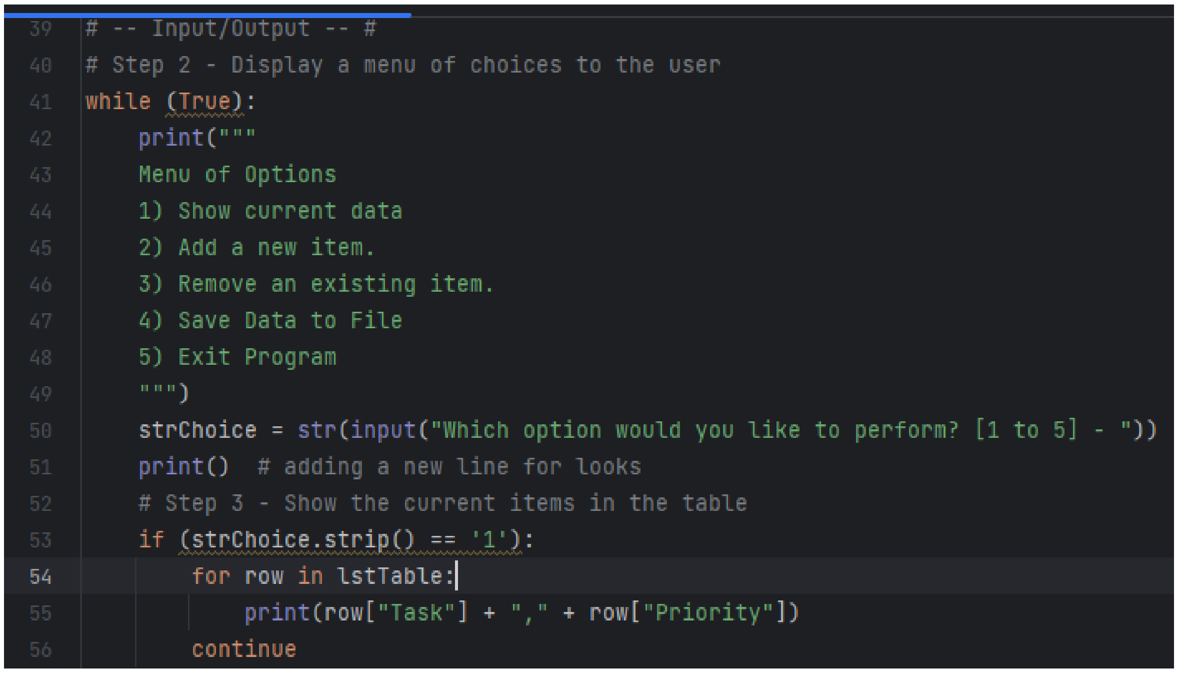


```
# -- Processing -- #
# Step 1 - When the program starts, load any data you have
# in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)

# Process the data

objFile = open(strFile, "r")
for row in objFile:
    lstRow = row.split(",")
    dicRow = {"Task": lstRow[0].strip(), "Priority": lstRow[1].strip()}
    lstTable.append(dicRow)
objFile.close()
```

Figure 4: Processing the Data Section

Displaying a Menu of Choices and the Data in the Dictionary (Table)

For the Menu of Options section, the previous code was considered complete and didn't warrant any updates, so no additional changes occurred. In the next section, if the user selects Option 1 from the Menu of Options, they will be able to display the data in the dictionary. Within the "while" loop and using the "if" and "for" loop functions, I used the dictionary row name instead of the traditional as "[0]" in the list row and the second value, value as "[1]", etc. Lastly, I used the separator function and added a comma to separate the two inputs for clarity. The remainder of the Displaying the Data to the Dictionary (Table) section is captured below (Figure 5).



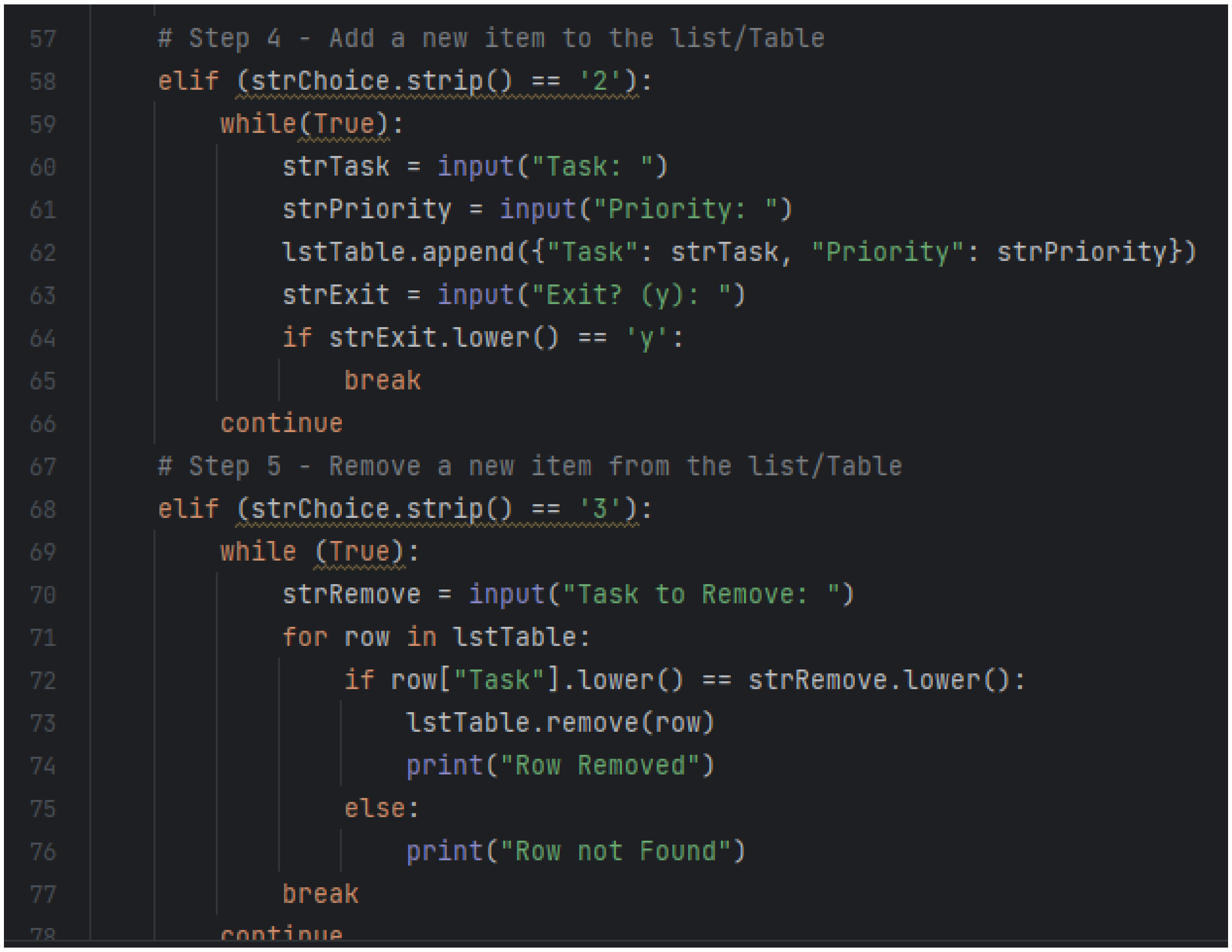
```python
# -- Input/Output -- #
# Step 2 - Display a menu of choices to the user
while (True):
    print("""
    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program
    """)
    strChoice = str(input("Which option 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'):
        for row in lstTable:
            print(row["Task"] + "," + row["Priority"])
        continue
```

Figure 5: Displaying a Menu of Choices & Data in the Dictionary (Table) Section

Adding/Removing Data to the Dictionary (Table)

If the user selects Option 2 from the Menu of Options, they will be able to add data to the table. Within the "while" loop, the user will be prompted to input the task and priority in string format. Both the task and priority will then be treated as a dictionary using curly brackets, {}. After this step is completed, an input to exit is requested before leaving this menu. Otherwise, the user will be forced to add another task and priority.

Conversely, if the user selects Option 3 from the Menu of Options, they will be able to remove data from the table. Within the "while" loop, the user will be prompted to input the task in string format. The script will look at all of the tasks using a "for" loop and if a task exists it will remove it from the table. A confirmation will be displayed if the task was removed, otherwise it will state no row was found. The remainder of the Adding/Removing Data to the Dictionary (Table) section is captured below (Figure 6).



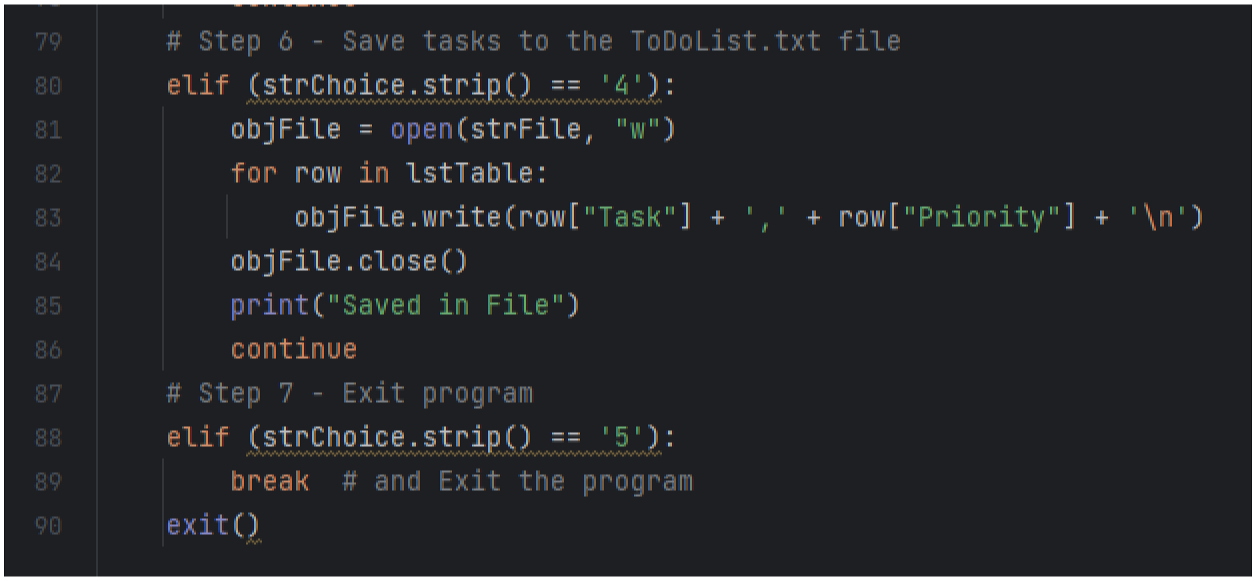
```
    # Step 4 - Add a new item to the list/Table
    elif (strChoice.strip() == '2'):
        while(True):
            strTask = input("Task: ")
            strPriority = input("Priority: ")
            lstTable.append({"Task": strTask, "Priority": strPriority})
            strExit = input("Exit? (y): ")
            if strExit.lower() == 'y':
                break
        continue
    # Step 5 - Remove a new item from the list/Table
    elif (strChoice.strip() == '3'):
        while (True):
            strRemove = input("Task to Remove: ")
            for row in lstTable:
                if row["Task"].lower() == strRemove.lower():
                    lstTable.remove(row)
                    print("Row Removed")
                else:
                    print("Row not Found")
            break
        continue
```

Figure 6: Adding/Removing Data to the Dictionary (Table) Input Section

Saving Data as a Text File and Exiting the Program

If the user selects Option 4 from the Menu of Options, they will be able to save the data. Within the "while" loop and else-if menu of options function, I set an objFile variable to open the "ToDoList.txt" file variable in write mode. Using a "for" loop, all of the prior and modified dictionary rows from Options 2 and 3 are input back into the list table and saved going forward. Lastly, I closed the file per best practices.

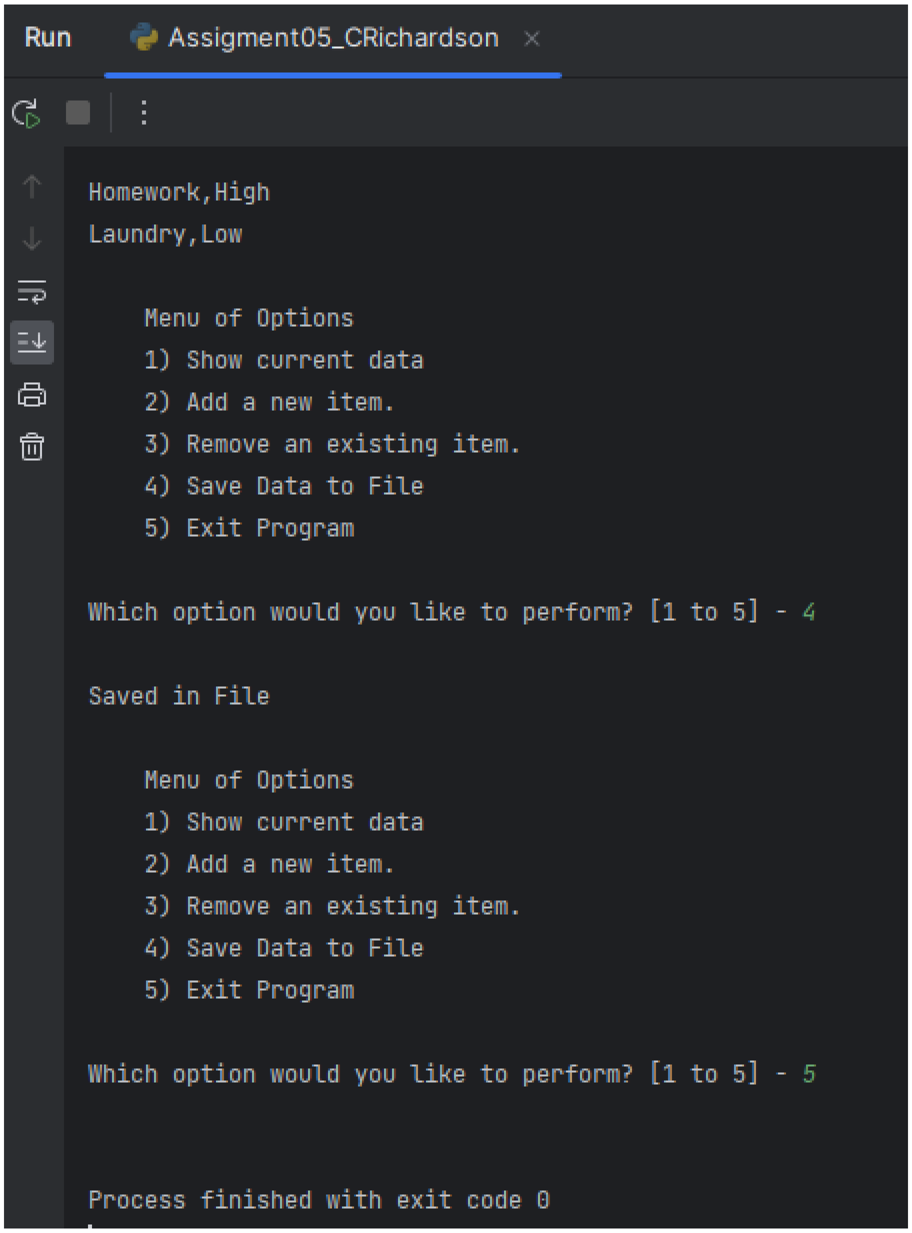
If the user selects Option 5 from the Menu of Options, they will be able to exit the program. The "while" loop from the Menu of Options concludes and the command exit() is added to completely shutdown the To Do Script. The entire Saving Data as a Test File and Exiting the Program section is captured below (Figure 7).



```
    # Step 6 - Save tasks to the ToDoList.txt file
    elif (strChoice.strip() == '4'):
        objFile = open(strFile, "w")
        for row in lstTable:
            objFile.write(row["Task"] + ',' + row["Priority"] + '\n')
        objFile.close()
        print("Saved in File")
        continue
    # Step 7 - Exit program
    elif (strChoice.strip() == '5'):
        break  # and Exit the program
    exit()
```

Figure 7: Saving Data as a Text File and Exiting the Program Section

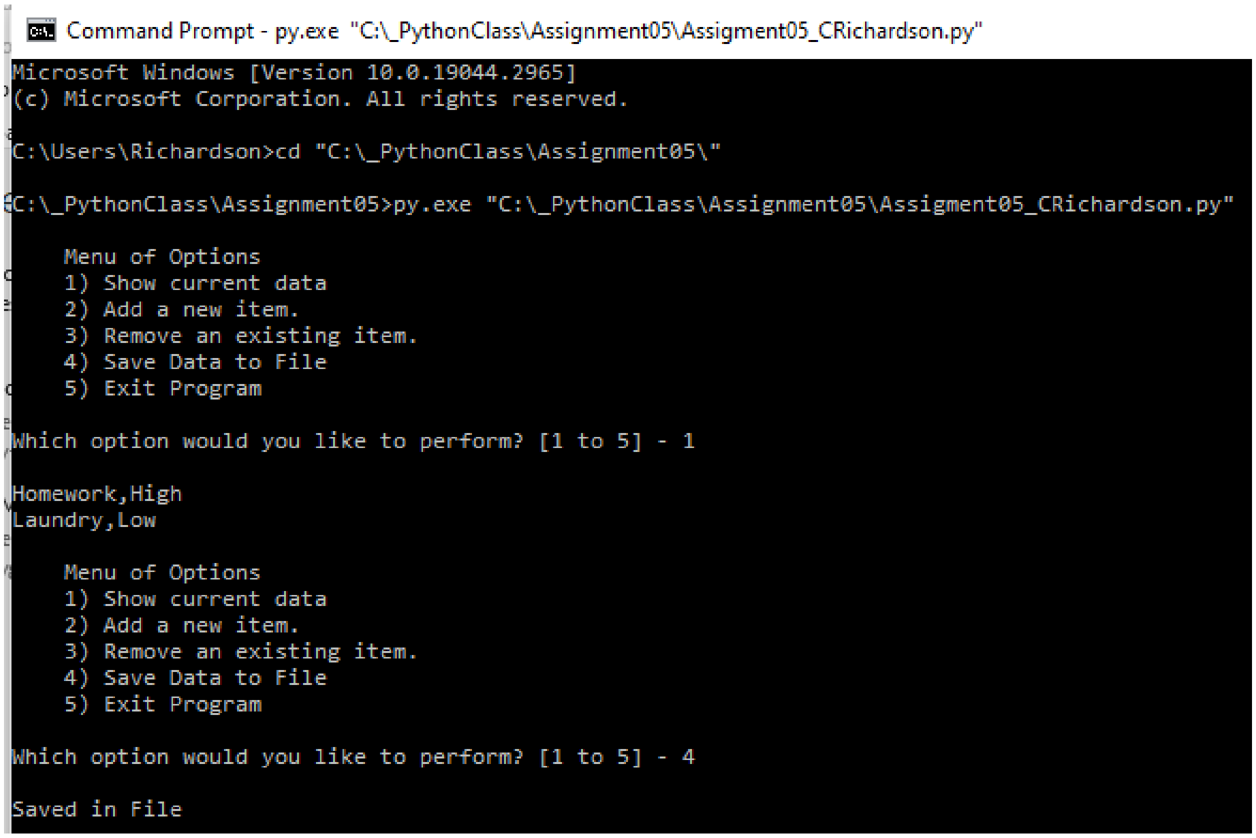
Figure 8 displays the last part of the solution using the Run Feature within PyCharm Community Edition.



```
Homework,High
Laundry,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] - 4

Saved in 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


Process finished with exit code 0
```

Figure 8: To Do List Code in Run mode of PyCharm Community Edition

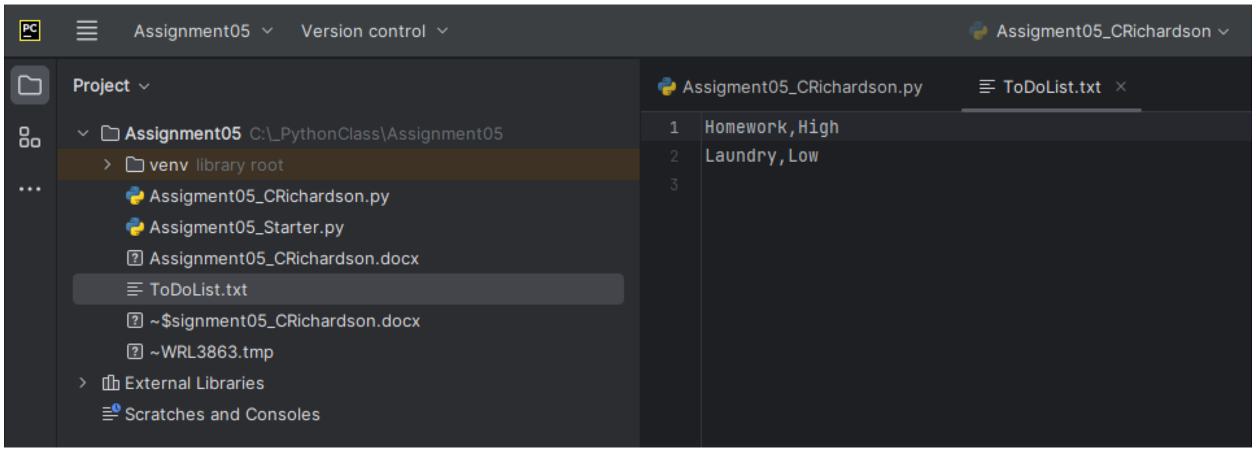
Figure 9 displays the final solution in Command prompt. I made sure to change the directory first to the location where the python script and text file relative path is located to ensure the script wrote to the file correctly.



```
Command Prompt - py.exe "C:\_PythonClass\Assignment05\Assigment05_CRichardson.py"
Microsoft Windows [Version 10.0.19044.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Richardson>cd "C:\_PythonClass\Assignment05\"

C:\_PythonClass\Assignment05>py.exe "C:\_PythonClass\Assignment05\Assigment05_CRichardson.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

Homework,High
Laundry,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] - 4

Saved in File
```

Figure 9: To Do List Code in Command Prompt

Lastly, Figure 10 confirms the ToDoList.txt file was correctly modified based on the user inputs from the Python script in PyCharm or Command Prompt.



```
Homework,High
Laundry,Low
```

Figure 10: To Do List Text File in PyCharm Community Edition

Summary

In conclusion Assignment 05, To Do List Python script, provided a refresher once again on “while” loops, “for” loops, and processing the data into a file functions. New concepts like dictionaries and GitHub were introduced. The end result is a to do list resource that allows the user to keep track of various tasks and how important they are to be completed. This week's assignment was quite a challenge and only appears to get more difficult as the weeks continue.

Bibliography

Dawson, Michael. *Python Programming for the Absolute Beginner*. Boston, Course Technology, Cop, 2008.