

Chase Richardson

May 22, 2023

Foundations of Programming: Python

Assignment 06

<https://github.com/jcrich13/IntroToProg-Python-Mod06>

Creating a Python Script for To Do List 2.0

Introduction

The primary purpose of Assignment 06 is to modify a more advanced 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 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 Assignment 05 pertaining to dictionaries. Furthermore, I reviewed the course material by modules, book, and other internet sources provided within the course to learn more about functions and classes. The information provided below outlines the steps I completed in order to create Assignment 06, To Do List 2.0 script.

Getting Started

To modify the To Do List 2.0 Python script in Assignment 06, 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 Assignment06 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).

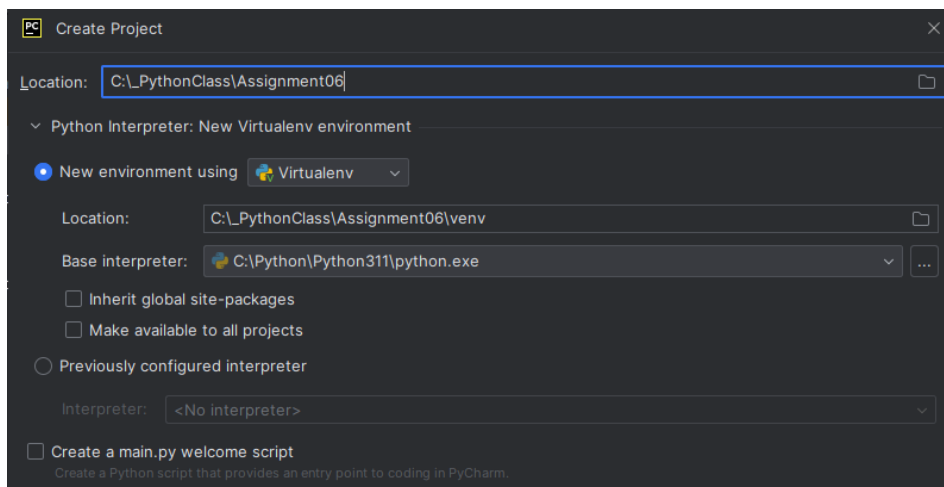
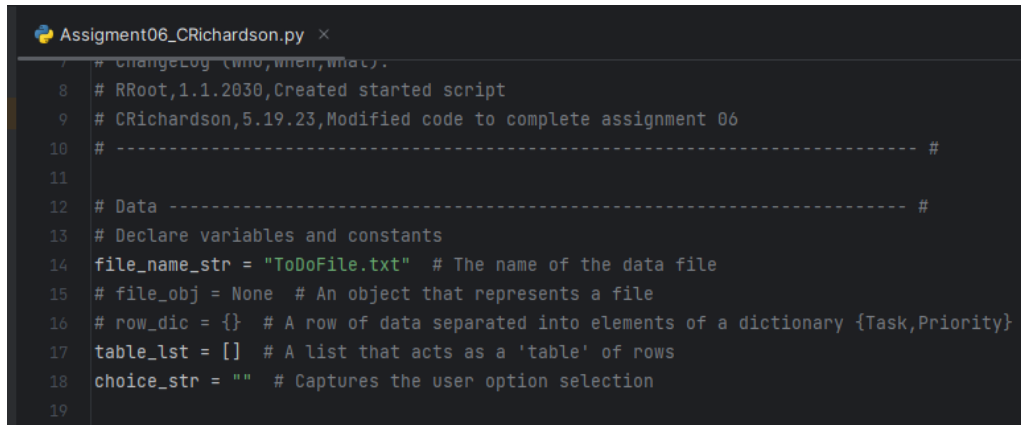


Figure 1: Creating a New Project within PyCharm for Assignment06

Since there was a starter script available I copied the “Assignment06_Starter.py” file into the Assignment06 subfolder to start. And then I created a copy of the “Assignment06_Starter.py” and renamed it to be called “Assignment06_CRichardson.py” to complete the subsequent steps.

Once I opened the file in PyCharm, I added my name and date to the Developer Notes section. I kept any prior existing variables in this section even if they were no longer being referenced (Figure 2).

A screenshot of a PyCharm code editor window titled "Assignment06_CRichardson.py". The code is in Python and shows the initial variable declarations. Line 7 has a comment "# ChangeLog (who,when,what)". Line 8 has a comment "# RRoot,1.1.2030, Created started script". Line 9 has a comment "# CRichardson,5.19.23, Modified code to complete assignment 06". Line 10 has a comment "# ----- #". Line 11 is blank. Line 12 has a comment "# Data ----- #". Line 13 has a comment "# Declare variables and constants". Line 14 declares "file_name_str = 'ToDoFile.txt'" with a comment "# The name of the data file". Line 15 declares "file_obj = None" with a comment "# An object that represents a file". Line 16 declares "row_dic = {}" with a comment "# A row of data separated into elements of a dictionary {Task,Priority}". Line 17 declares "table_lst = []" with a comment "# A list that acts as a 'table' of rows". Line 18 declares "choice_str = """ with a comment "# Captures the user option selection". Line 19 is blank.

```
7 # ChangeLog (who,when,what).
8 # RRoot,1.1.2030, Created started script
9 # CRichardson,5.19.23, Modified code to complete assignment 06
10 # ----- #
11
12 # Data ----- #
13 # Declare variables and constants
14 file_name_str = "ToDoFile.txt" # The name of the data file
15 # file_obj = None # An object that represents a file
16 # row_dic = {} # A row of data separated into elements of a dictionary {Task,Priority}
17 table_lst = [] # A list that acts as a 'table' of rows
18 choice_str = "" # Captures the user option selection
19
```

Figure 2: 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 reviewed each of the defined variables within this section that needed to be modified. There were four functions created “read_data_from_file”, “add_data_to_list”, “remove_data_from_list”, and “write_data_to_file” and the latter three needed to be revised in order to work properly. Using the lessons learned and code from the Assignment 05, I replicated just the processing element to append the new row into the code since the dictionary row and return code was already available in the “add” section. For the “remove” section, the process was similar, but the remove code used a “for” loop to determine if a task exists and if it does it will be removed from the table. The return code was also already available in the “remove” section. The Add and Remove Processing the Data section is captured below (Figure 3).

```

Assignment06_CRichardson.py x
47     :param priority: (string) with name of priority:
48     :param list_of_rows: (list) you want to add more data to:
49     :return: (list) of dictionary rows
50     """
51     row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
52     list_of_rows.append(row)
53     return list_of_rows
54
55     1 usage
56     @staticmethod
57     def remove_data_from_list(task, list_of_rows):
58         """ Removes data from a list of dictionary rows
59
60         :param task: (string) with name of task:
61         :param list_of_rows: (list) you want filled with file data:
62         :return: (list) of dictionary rows
63         """
64         for row in list_of_rows:
65             if row["Task"].lower() == task.lower():
66                 list_of_rows.remove(row)
67         return list_of_rows

```

Figure 3: Add and Remove Processing the Data Section

When writing data to a file, I once again referenced the Assignment 05 deliverable and mirrored the processing component of this information. I requested to open the file with write option, used the “for” loop for each row that was added or removed, and asked the code to write the task and priority and then close the file when completed. The return code was already provided. The entire Write Data to File Processing the Data section is captured below (Figure 4).

```

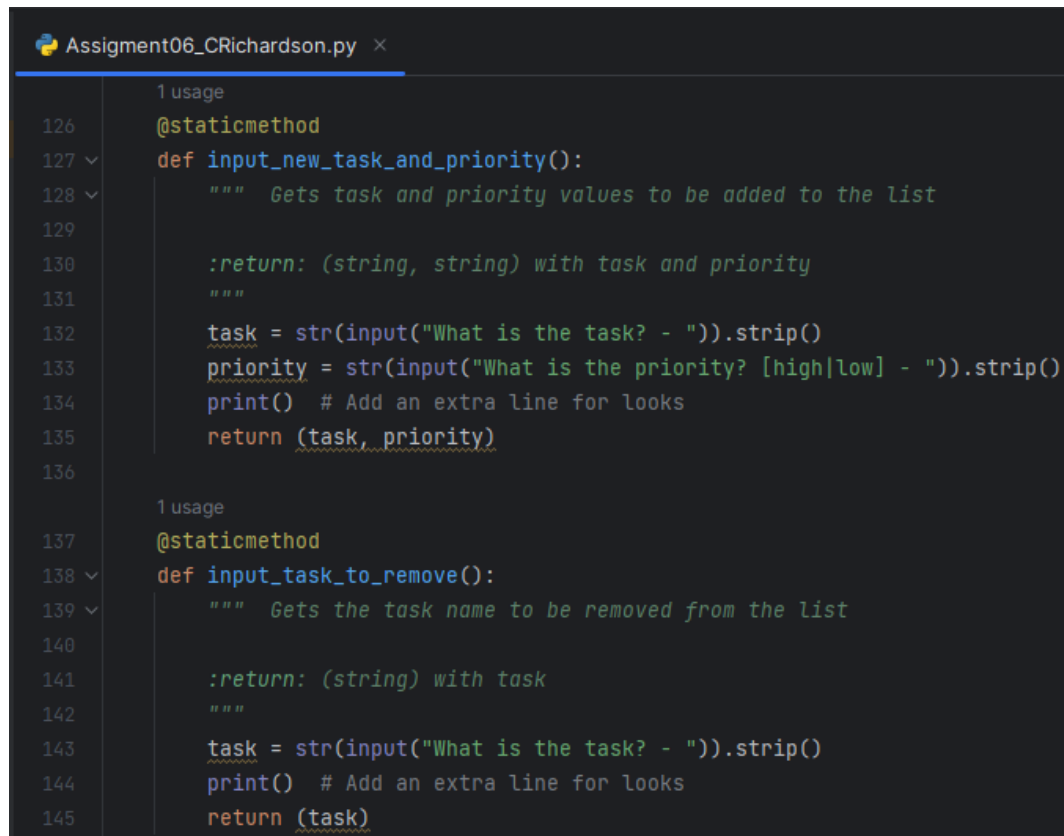
Assignment06_CRichardson.py x
1 usage
68     @staticmethod
69     def write_data_to_file(file_name, list_of_rows):
70         """ Writes data from a list of dictionary rows to a File
71
72         :param file_name: (string) with name of file:
73         :param list_of_rows: (list) you want filled with file data:
74         :return: (list) of dictionary rows
75         """
76         file = open(file_name, "w")
77         for dicRow in list_of_rows:
78             file.write(dicRow["Task"] + "," + dicRow["Priority"] + "\n")
79         file.close()
80         return list_of_rows

```

Figure 4: Write Data to File Processing the Data Section

Presentation (Input and Output) of the Data

In the Presentation section, there were five functions created that I reviewed to start “output_menu_tasks”, “input_menu_choice”, “output_current_tasks_in_list”, “input_new_task_and_priority”, and “input_task_to_remove”. Luckily, of the five presentation functions only the last two needed to be revised in order to work properly. For the input new section, the user will be prompted to input the task and priority in string format and return the task and priority. Conversely, for the input task to remove, the user will be prompted to input the task only in string format and return the task. The remainder of the Presenting the Data section is captured below (Figure 5).



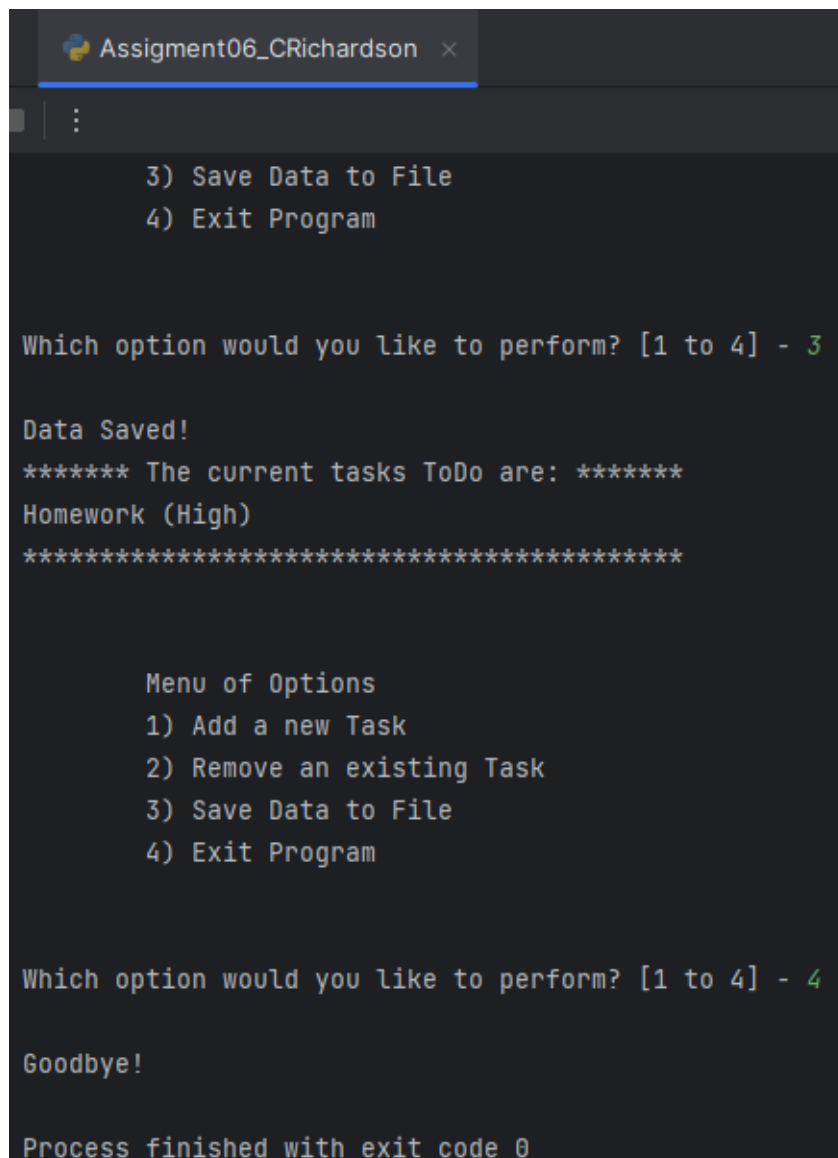
```
Assigment06_CRichardson.py x
1 usage
126 @staticmethod
127 def input_new_task_and_priority():
128     """ Gets task and priority values to be added to the list
129
130     :return: (string, string) with task and priority
131     """
132     task = str(input("What is the task? - ")).strip()
133     priority = str(input("What is the priority? [high|low] - ")).strip()
134     print() # Add an extra line for looks
135     return (task, priority)
136
137 1 usage
138 @staticmethod
139 def input_task_to_remove():
140     """ Gets the task name to be removed from the list
141
142     :return: (string) with task
143     """
144     task = str(input("What is the task? - ")).strip()
145     print() # Add an extra line for looks
146     return (task)
```

Figure 5: Presenting the Data Section

Running the Program with Functions

The remainder of the code, is the consolidation of the processing and presentation components together. While this is considered the “main body” of the program, the code here was already completed and didn’t require any editing or additional changes. Most of the information shared here is very similar to the prior assignment except it is more concise and repeatable, because the functions have been defined going forward. Furthermore, our course textbook states, “One of the biggest (advantages) is that it allows you to break up your code into manageable, bit-sized chunks” (Dawson, 159).

Figure 6 displays the last part of the solution using the Run Feature within PyCharm Community Edition.



```
Assigment06_CRichardson x
⋮
3) Save Data to File
4) Exit Program

Which option would you like to perform? [1 to 4] - 3

Data Saved!
***** The current tasks ToDo are: *****
Homework (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] - 4

Goodbye!

Process finished with exit code 0
```

Figure 6: To Do List 2.0 Code in Run mode of PyCharm Community Edition

Figure 7 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.

```

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

What is the task? - Laundry
What is the priority? [high|low] - Low

***** The current tasks ToDo are: *****
Homework (High)
Laundry (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] - 3

Data Saved!
***** The current tasks ToDo are: *****
Homework (High)
Laundry (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:\_PythonClass\Assignment06>

```

Figure 7: To Do List Code 2.0 in Command Prompt

Lastly, Figure 8 confirms the ToDoList.txt file was correctly modified based on the user inputs from the Python script in PyCharm or Command Prompt.

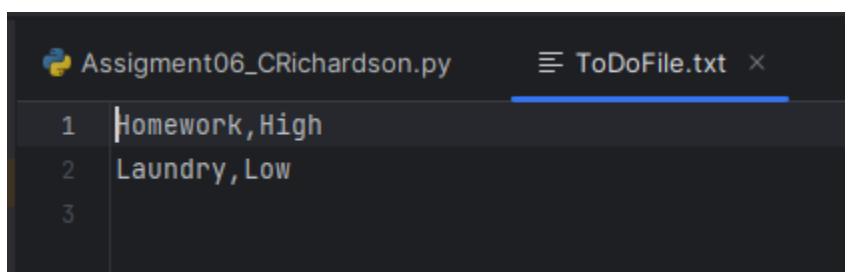


Figure 8: To Do List 2.0 Text File in PyCharm Community Edition

Summary

In conclusion Assignment 06, To Do List 2.0 Python script, provided a refresher once again on loops, processing data to a file, and earlier concepts. New concepts like functions and classes were introduced. The end result was a to do list resource that allows the user to keep track of various tasks and how important they are to be completed with more efficient code behind the scenes. Understanding the concept of defining functions, classes and breaking out the processing and presentation components was quite difficult to figure out in this assignment.

Bibliography

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