Hannah Chung

May 16, 2020

IT FDN 100 Foundations of Programming: Python

Assignment 05

<https://github.com/hannahinissaquah/IntroToProg-Python>

# **Assignment05: Working with Dictionaries and Files**

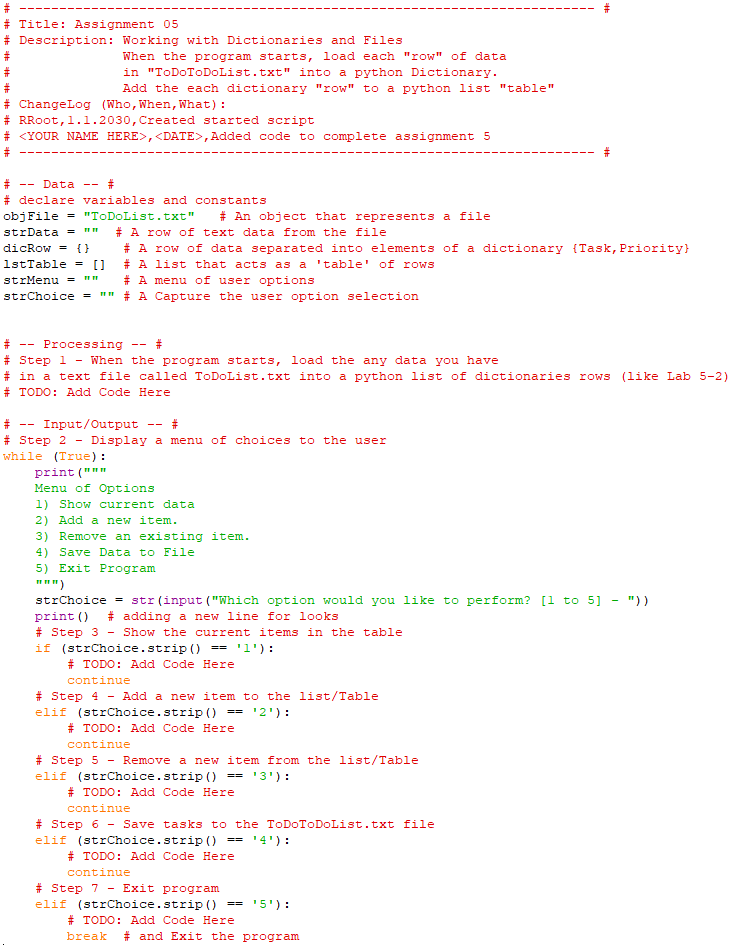
Introduction

In this document, I will explain the steps I took to modify a script that manages a “To Do list” containing two columns of data: “Task” and “Priority”. The columns of data are loaded into a Python dictionary object, each dictionary object representing one row of data, and these rows are added to a Python List object to create a table of data. I was provided with a file, “Assignment05\_Starter.py” that contained starter code, and my assignment was to add code to complete the Processing and Presentation layers of the program that would allow the user to review current data, add a new item to the list, remove an existing item, save data to the file, and to exit the program.

Assignment Steps

**Getting started**

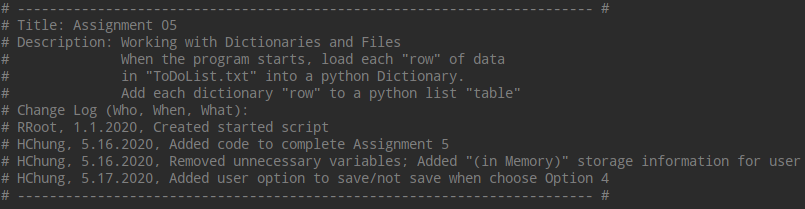
First, I created a new PyCharm project that uses the \_PythonClass\Assignment05 folder as its location. I added the Assignment05\_Starter.py file to my project and reviewed each line of starter code. Figure 1 shows a screenshot of the starter code file opened in IDLE. The code has been organized into three main sections: Data, Processing, and Presentation (or Input-Out). Organizing code into sections in this manner is known as “Separation of Concerns”, each section addressing a separate concern.



**Figure 1 Assignment05 Starter Code**

After reviewing the starter code multiple times, I began to make a plan for how I was going to use the starter code to complete Assignment05.

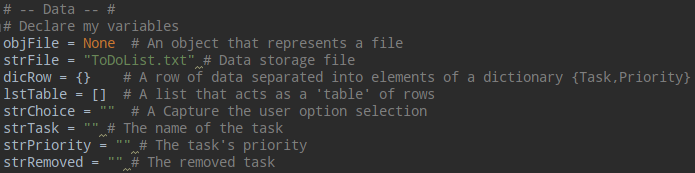
There were a few typos in the Script Header, so I corrected those and made my first entry into the change log as shown in Figure 2. Over time, I continued to make updates to the change log by recording the changes I had made to the previous version of the script.



**Figure 2 Script header**

**Data**

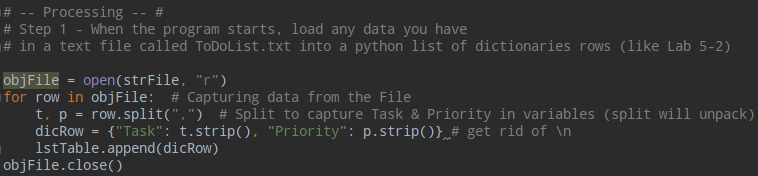
The program is separated into three main sections: Data, Processing, and Input/Output (or Presentation). As you can see from Figure 1, the Data section was already populated with a list of declared variables to be used throughout the code. I did end up making a few changes to the list, including adding a few new variables and reassigning the data storage file “ToDoList.txt” to a new strFile variable. Moreover, there were a couple of variables strData and strMenu that I did not end up needing for the program so I removed those variables from this section. Figure 3 shows the contents of my Data section.



**Figure 3 Data Section**

**Processing**

In the Processing section, the goal was to load any data I have in a text file called “ToDoList.txt”, which I’ve assigned to the strFile variable, into a Python list of dictionary rows. I created a text file and saved it as “ToDoList.txt” in the \_PythonClass\Assignment05 folder. I used the open() function to open the file and assign the results to objFile. I provided the “r” for the access mode, which tells Python that I want to open the file for reading. Next, I created a for loop to loop through the file and used the split() command to split each row on the comma “,” so that I end up with a list of two strings: the task and priority assigned to t and p, respectively. Next, I put the items into a dictionary row. Dictionaries are similar to lists except rather than storing information in a sequence, it is stored in pairs—the key and its value. I named the primary key “Task” and its value is t, and I named the second key “Priority” and its value is p. So we can think of “Task” and “Priority” as the column names and t and p are the values stored in each column. I used the strip() command to strip any spaces and tabs, and in this case carriage returns (\n). Finally, the dictionary row is appended to the table (lstTable) and when the for loop is done, the object file is closed. Figure 4 shows the contents of my Processing Section.



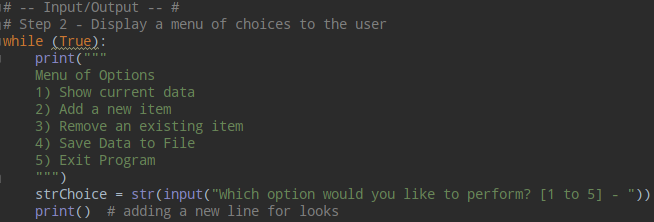
**Figure 4 Processing Section**

**Input/Output**

The Input/Output section was divided into 6 sub-sections. The first sub-section contained starter code to display a menu of choices and to give the user the ability to choose from 5 options:

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

The user’s choice [1 to 5] is then assigned to the variable strChoice. No changes were made to the code for this first sub-section (see Figure 5).

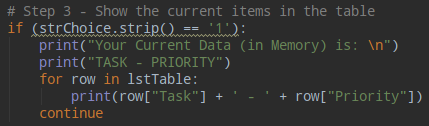


**Figure 5 Display a menu of choices to user**

The following sub-sections contained pseudo-code providing a description of each step the programmer is to take, starter code with if statement and elif clauses, and TODO comments that describes the task the programmer is to complete for each sub-section. PyCharm’s TODO comments allows the programmer to mark parts of the code that require attention and all TODO items will be listed in the TODO tool window.

Option 1 – Show Current Data

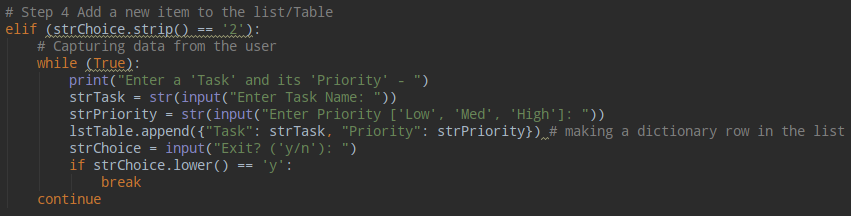
If the user chooses Option 1, the current items in the table will be shown. To display the items in the table, I created a for loop to loop through the table and print each dictionary row, specifying the keys “Task” and “Priority” to retrieve their values and using a hyphen (-) to separate their values. As mentioned above, accessing dictionary values is similar to accessing list values except you access a dictionary value with a key whereas you access the list value using an index or subscript. I also added a couple of print statements above the for loop to tell the user what is being presented. Figure 6 shows the final code for this sub-section.



**Figure 6 Show current data**

Option 2 – Add a New Item

If the user chooses Option 2, the user will be able to add a new item to the list. I added a while loop by using while (True), which means that the loop will continue to ask the user to enter a task name and priority until there is an exit condition in the loop body. The user’s input for the task name will be assigned to the variable strTask and the user’s input for its priority will be assigned to the variable strPriority. A new dictionary row containing the new values will be appended to the table lstTable. The user will have the option of continuing to add new items or exiting. The loop will only exit if the user chooses ‘y’. If the user enters ‘y’, the user will be returned to the main menu of options. If the user types in any character other than ‘y’, it will return to the top of the while loop, prompting the user to enter a task name and priority. See Figure 7 for the code to add new item(s) to the list.

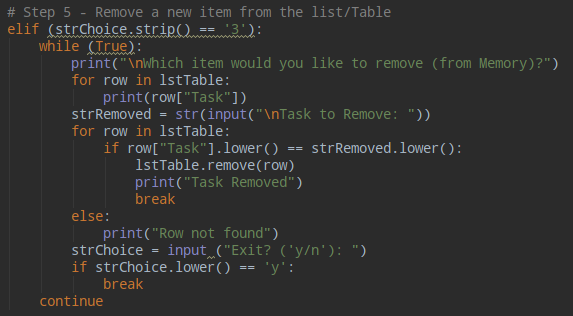


**Figure 7 Add new item to list**

Option 3 – Remove an Existing Item

If the user selects Option 3, the user can remove an existing item from the list. This sub-section took me the most amount of time and trial-and-error to figure out.

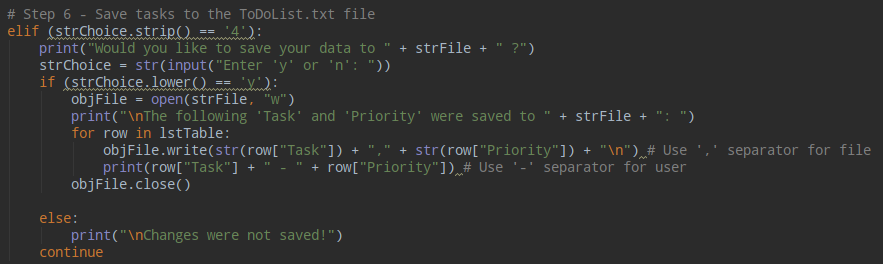
Again, I added a while loop to allow the user to remove tasks until the user chooses to exit this option. While testing this section, I thought it would be helpful to first display to the user the current list of tasks stored in memory. I then ask the user to enter the name of the task he/she would like to remove and assign the user’s input to the variable strRemoved. If the user’s entry, converted to a lower case string, matched a dictionary item, also converted to a lower case string, then the program would enter into a for loop and remove the user-entered item from the table. If the user enters in an item that does not exist in the table, information will be displayed to the user that the row was not found. Again, the user is asked whether he/she would like to exit the option. Until the ‘y’ is entered, the user will be prompted to enter in the name of the item he/she would like to remove.



**Figure 8 Remove an item from the list**

Option 4 – Save Data to File

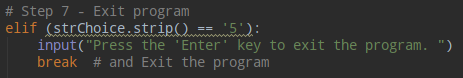
If the user selects Option 4 to save the data to the file, I prompt the user again to verify he/she would like to save the data to ToDoList.txt. I added this as a safety measure to ensure the user really wants to overwrite the existing file with the items stored in memory. If the user chooses to save the data, the ToDoList.txt file is opened in write access mode, and the dictionary rows are written to the file using the write () method. I also added print statements to display to the user what data was saved to the file. If the user chooses not to save the data, changes are not saved and the user will be returned to the menu of options. See Figure 9 for the code to save changes to the ToDoList.txt file.



**Figure 9 Save tasks to the ToDoList.txt file**

Option 5 – Exit the Program

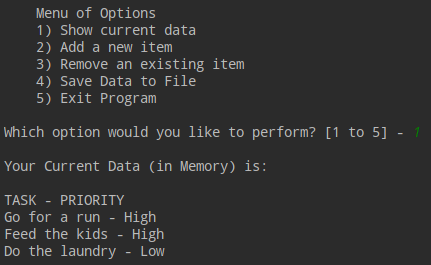
Finally, if the user chooses Option 5, the user will be prompted to press the ‘Enter’ key to exit the program. When the Enter key is pressed, the subsequent break statement will end the first while loop and exit the program.



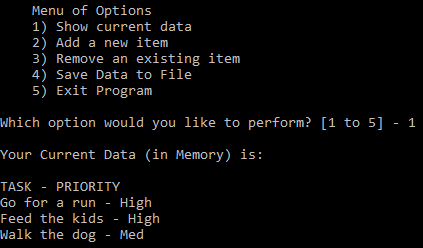
**Figure 10 Exit the program**

**Running the Script in PyCharm and the Command Window**

When the program is executed, the Menu of Options will appear. Figure 11 shows the script running in PyCharm when Option 1 is selected, and Figure 12 shows the script running in the Command Window when Option 1 is selected.

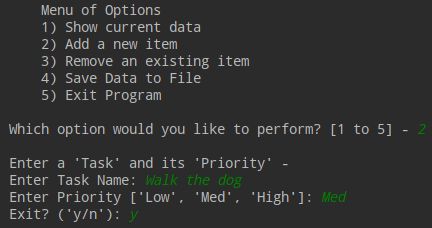


**Figure 11 PyCharm - User selects Option 1**

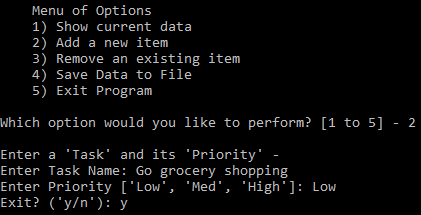
****

**Figure 12 Command - User selects Option 1**

When user selects Option 2, he/she is prompted to enter a task name and priority. See Figure 13 to see script running in PyCharm when the user selects Option 2, and see Figure 14 to see the script running in the Command Window.

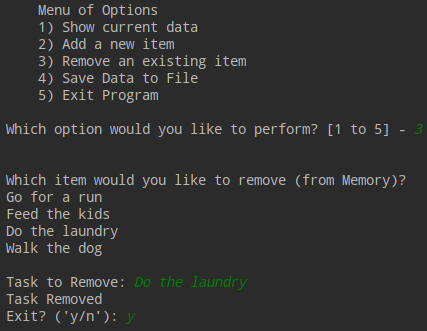
****

**Figure 13 PyCharm - User selects Option 2**

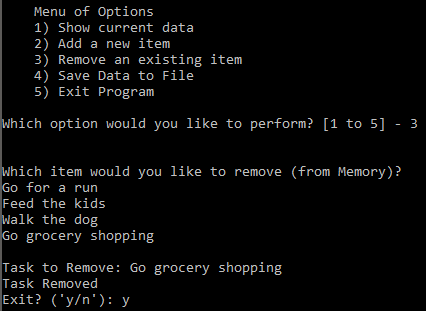


**Figure 14 Command - User selects Option 2**

When user selects Option 3, he/she is prompted to enter an existing item to remove. Figure 15 shows the script running in PyCharm when the user selects Option 3, and Figure 16 shows the script running in the Command Window when Option 3 is selected.

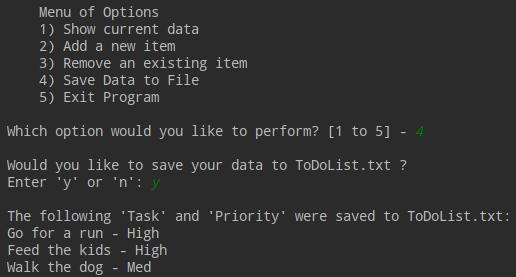
******

**Figure 15 PyCharm - User selects Option 3**

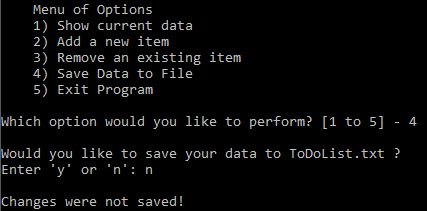


**Figure 16 Command - User selects Option 3**

When the user selects Option 4, the user is asked whether he/she would like to save the data to the file. Figure 17 shows the script running in PyCharm when the user selects Option 4, and Figure 18 shows the script running in the Command Window. Figure 17 shows the user choosing to save to the file whereas Figure 18 shows the user choosing not to save the data to the file.

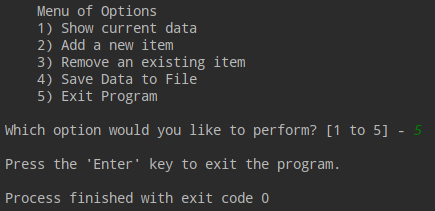


***Figure 17 PyCharm - User selects Option 4***

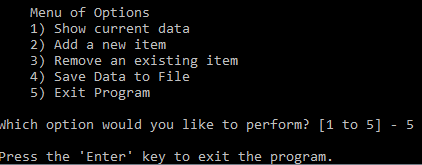


**Figure 18 Command - User selects Option 4**

When user selects Option 5, he/she is asked to press the ‘Enter’ key to exit. Figure 19 shows the script running in PyCharm when the user selects Option 5, and Figure 20 shows the script running in the Command Window when Option 5 is selected.



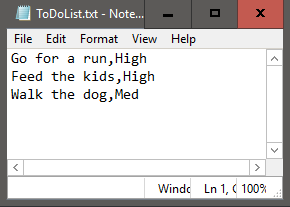
**Figure 19 PyCharm - User selects Option 5**



**Figure 20 Command - User selects Option 5**

**Verifying the Script Worked**

I verified that the script worked by opening the ToDoList.txt located in the \_PythonClass\Assignment05 folder. Figure 21 shows a screenshot of its contents.



**Figure 21 Screenshot of ToDoList.txt**

Summary

For this assignment, I modified a starter script “Assignment05\_Start.py” that manages a “To Do list” containing two columns of data: “Task” and “Priority”. The columns of data are loaded into a Python dictionary object, each dictionary object representing one row of data, and these rows are added to a Python List object to create a table of data. I found this assignment to be the most difficult assignment so far, and working with starter code had its advantages and disadvantages. It was helpful to have the separated sections with pseudo-code and TODO comments; however, because I did not use all the declared variables, I kept second-guessing myself.