Pavel (Pasha) Beresnev

11/16/20

IT FDN 110A

Assignment 05

# How to Create a To-Do List in Python Using Lists/Dictionaries

## Introduction

In this module I discuss how to use a menu of options presented to the user to upload a to-do list from a text file, allow the user to show the data, add to it or remove from it, and then re-save to the list. This will be performed by introducing the concept of lists and dictionaries. Finally, I will demonstrate how this looks in both PyCharm and the Command Prompt.

## Header/Data:

As always, I begin by creating a header. In this case, I started with some starter code created by the instructor and updated the log to add to the template as shown in Figure 1:

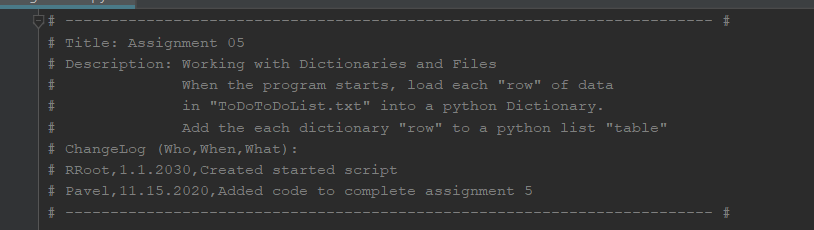


Figure 1: Code Header showing added change log item

Next I define the variables to be used in this code, as shown in Fig2:

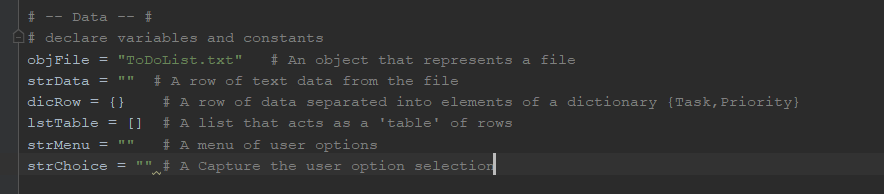


Figure 2: Data section of code

This is in line with the Separation of Concerns practice, where we try to separate out the data, the processing, and the presentation code in our scripts for ease of reading.

## Processing

The next section of the code is for the processing. We first write out the code that loads the data from the existing text file. We create a variable to first open the file and read from it, using ‘r’ as the second parameter in the ‘open’ function. Then we iterate over each row in the file. For each iteration we split the data using the comma and put that in a new list. Then we create a dictionary using the key “Task” for the first element of the list and add a second key “priority” for the next list element. We also use the .strip() function to delete the carriage return. Finally, we add this new dictionary to our table list using “.append()” and close the file as shown in Figure 3. Note, we use a try/except clause to avoid an error in case there is no text file yet, in which case the user sees a message that no data exist.

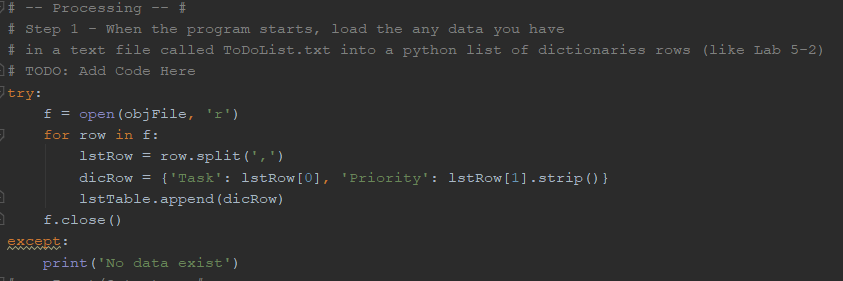


Figure 3: Processing code

## Input/output

Our next code section is for input/output. We start by displaying the menu of choices to the user with the print function. Then we ask the user which option they want to perform and save that in a variable. Then we add code for each option.

First, we start with the option for showing the current items in the table. This is option 1. Since we previously saved those items in our table list, we can simply iterate over the list elements and print each one on a new line with the ‘print’ function.

If the user selects 2, we want to add code for adding a new item to the list/table. To do this, we store the task they want to add as a variable using input and do the same for the priority. Then I created a new dictionary using the task and priority keys with the spaces stripped and stored inputs from the user. Finally, I appended this new row of data (dictionary) to the table list. Note, I also check to see if this task is already in the list so the user can’t add a duplicate. I create a list of existing tasks by iterating over the existing table and appending the values in lowercase for the task key. Then I check if the task the user entered in lowercase and stripped is in the list of tasks using “if” and “in”. If it is, then I print an appropriate message to the user and we continue to the next iteration of the menu loop. Otherwise the code continues to add the item to the to-do list.

If the user picks 3, we want to delete an item from the table. To do this, we store the item the user wants to delete using “input” and a variable. Then we loop through the table. For each element, if the value of ‘task’ in lowercase matches the item the user wanted to delete in lowercase and stripped, then we remove that element from the list and break out of the for loop so we don’t need to keep iterating and looking for that task. Here I also check to see if the input is in the to-do list and display if not to the user. Similar to the code to add an item, here I also create a list of tasks by iterating over the table and appending the value for each element’s “task” key in lowercase. Then if the task the user wants to delete in lowercase and stripped is not in that task list, I print an appropriate message and continue to the next loop of the menu. Otherwise, the remaining code in this block runs as described above. See Figure 4 below:

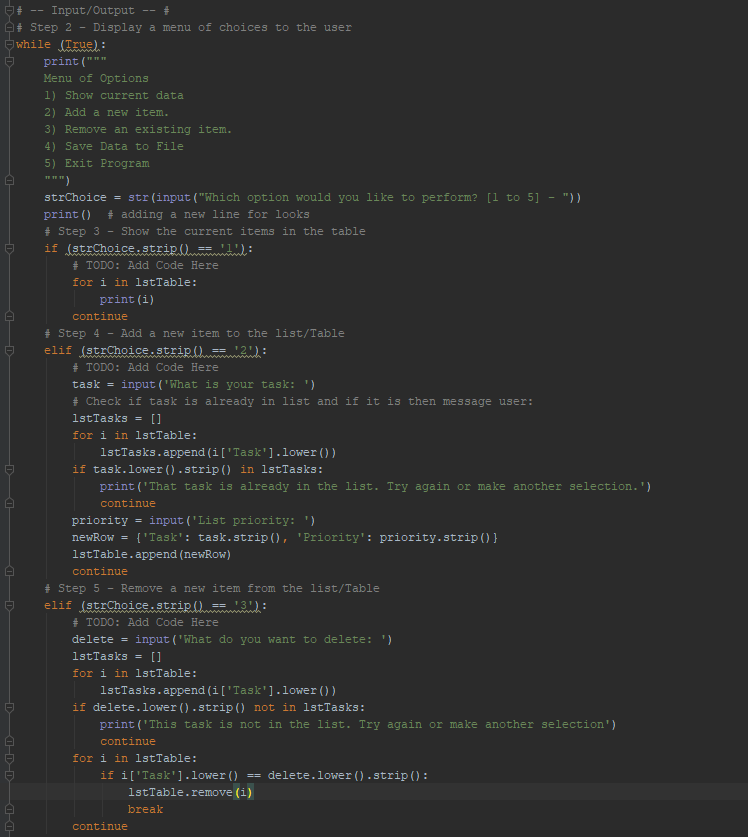


Figure 4: Input/output for showing, adding and removing items related to list

Finally, we add code for saving the tasks to the text file. If the user picks that option we open the file, this time in write mode. Then we loop through the table, writing the task value and the priority value separated by a comma with a carriage return at the end. We then close the file. If the user selects 5 we simply break the while loop that runs through the options and print that the user has exited as shown in Fig 5:

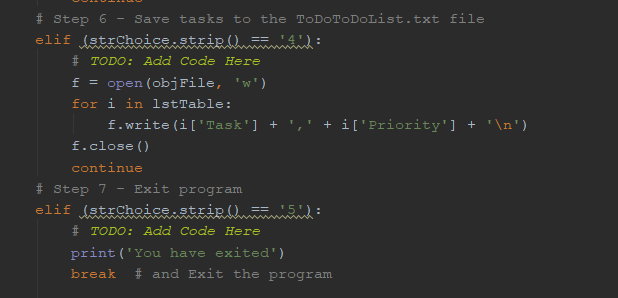


Figure 5: Input/output for saving data and exiting the program

## Results

Below is an example of the program running in PyCharm. In this case I started with no text file, hence you can see in Fig 6 that the appropriate message is displayed:

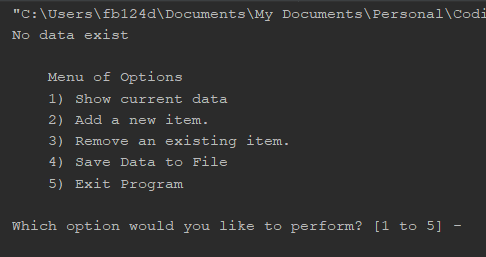


Figure : Example run in PyCharm showing no existing data

I then ran it again and imported the data from an existing text file. Then I added a “write” task, deleted the “math” task and saved/exited as shown in figures 7-9.

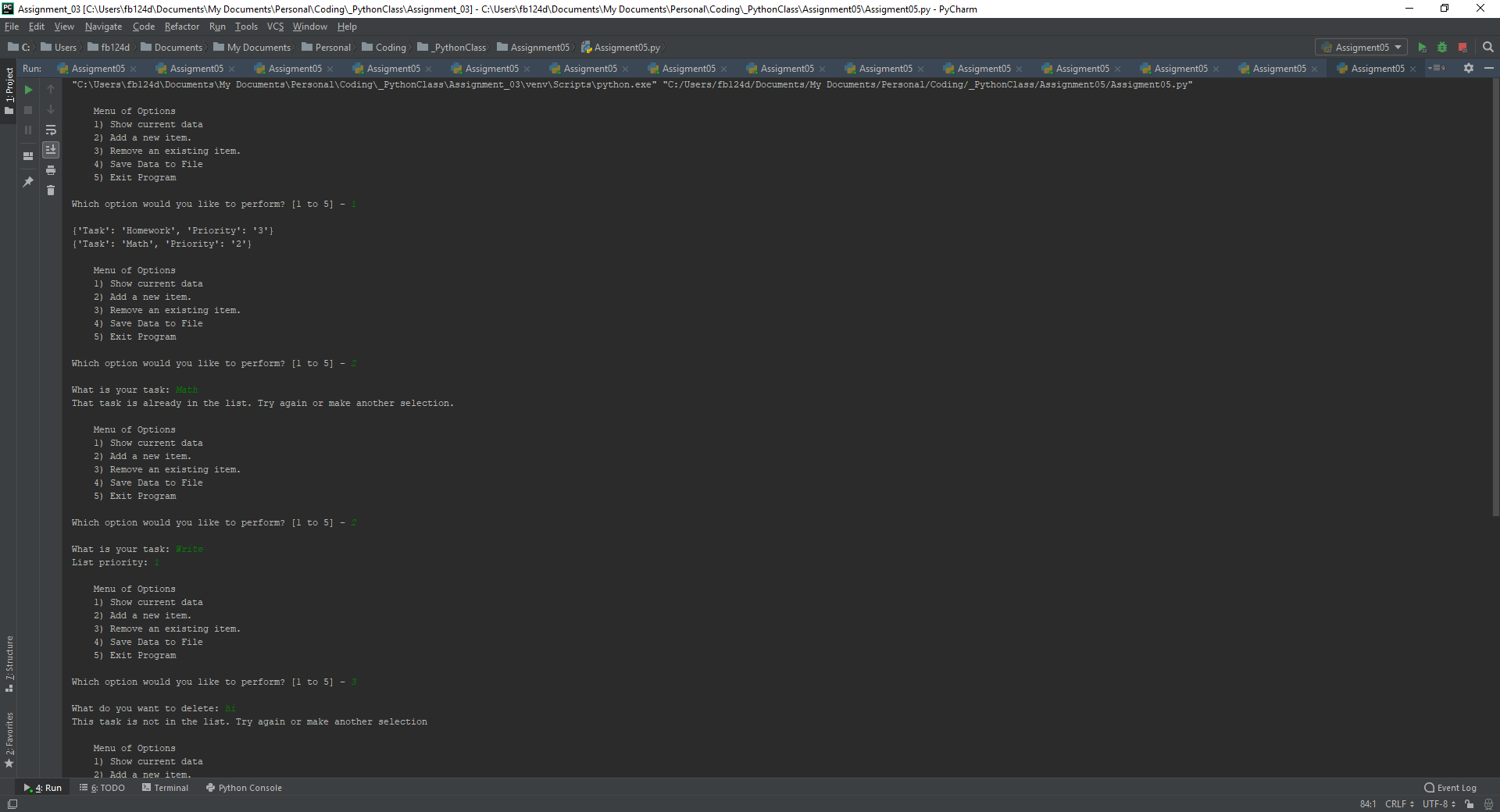


Figure 7: Example Session in PyCharm

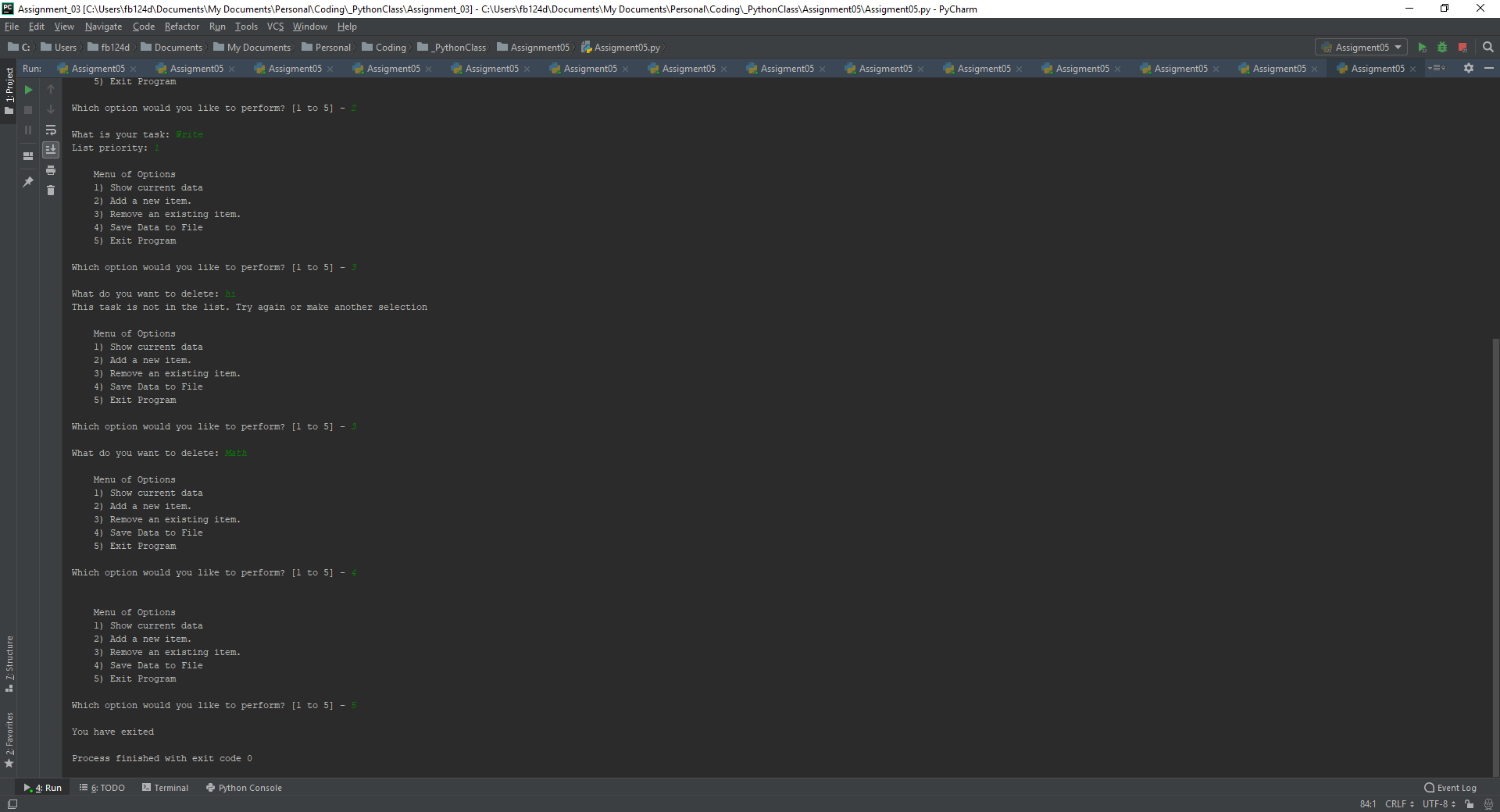


Figure 8: PyCharm session continued

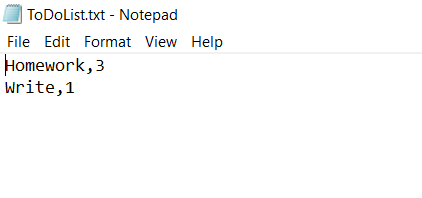


Figure : Resulting text file

Below is a similar session but in the Command Prompt. I loaded a similar text file with slightly different data, then added a wash task, deleted the call task and saved/exited as shown in Figures 10-12:

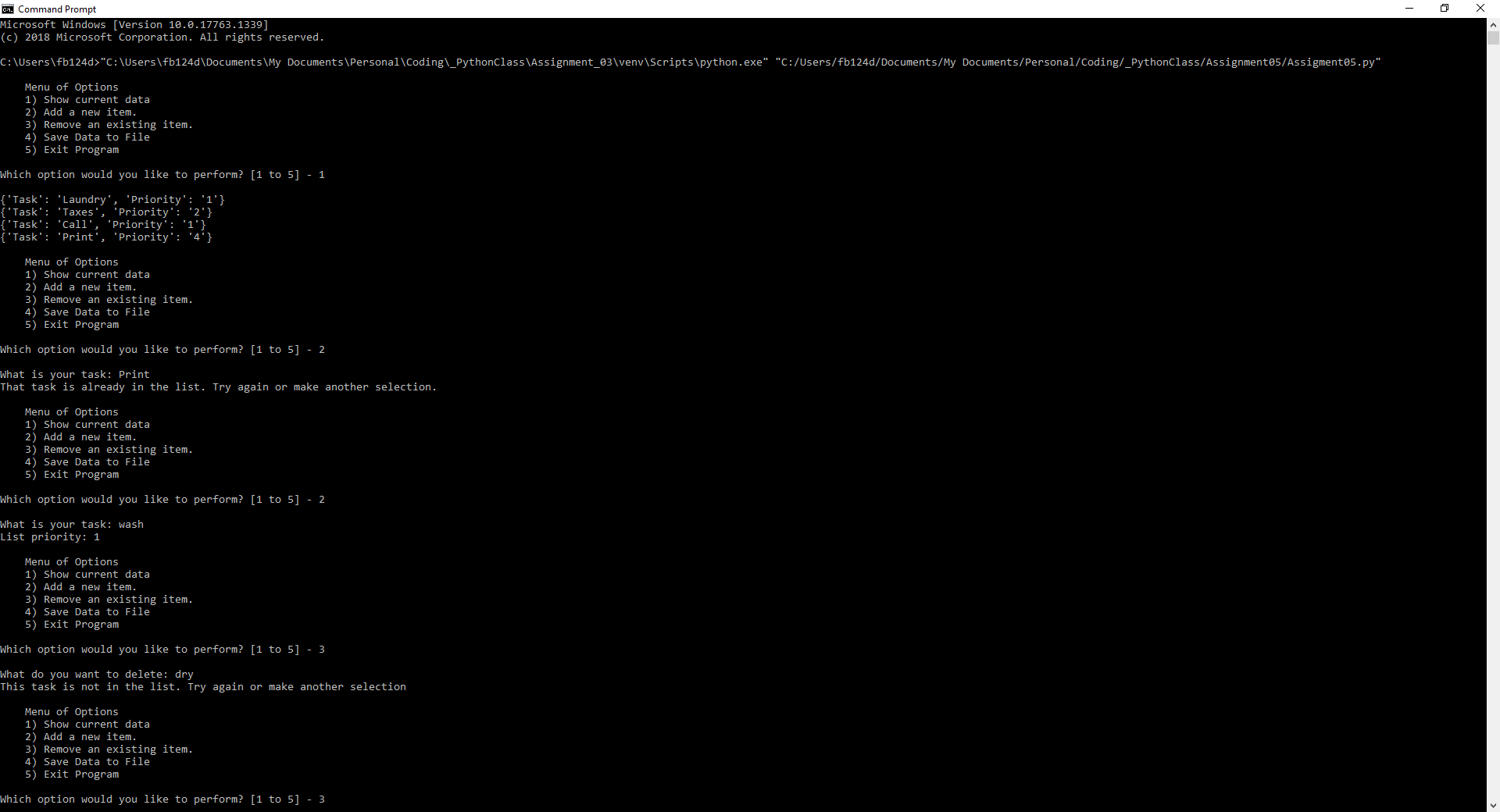


Figure 10: Command Prompt Session



Continuation

Figure : Command Prompt Session Continued

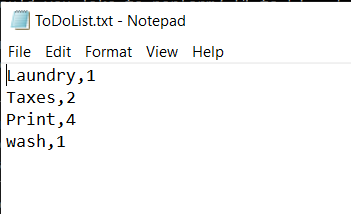


Figure 12: Command Prompt Session Output

Fig 13 demonstrates the Command Prompt version of when the text file doesn’t yet exist:

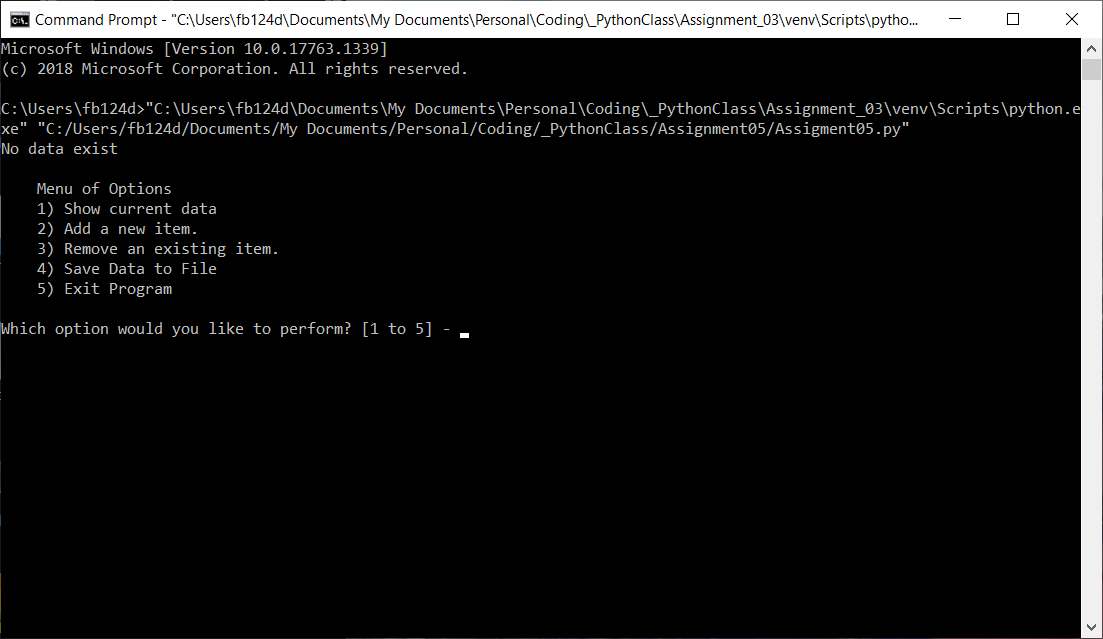


Figure : Command Prompt Showing run when text file doesn’t exist

Finally, here is a demonstration of the strip and lower methods used to add/delete data. Figures 14-15 in PyCharm show that if “Wash” is in the list then if we try to add “ wash “ it’s treated as in the list. Then if we add “Math” with spaces before/after and display the result we get a stripped version. Similarly if I try deleting “homework” with spaces we still delete the existing list item “Homework”.

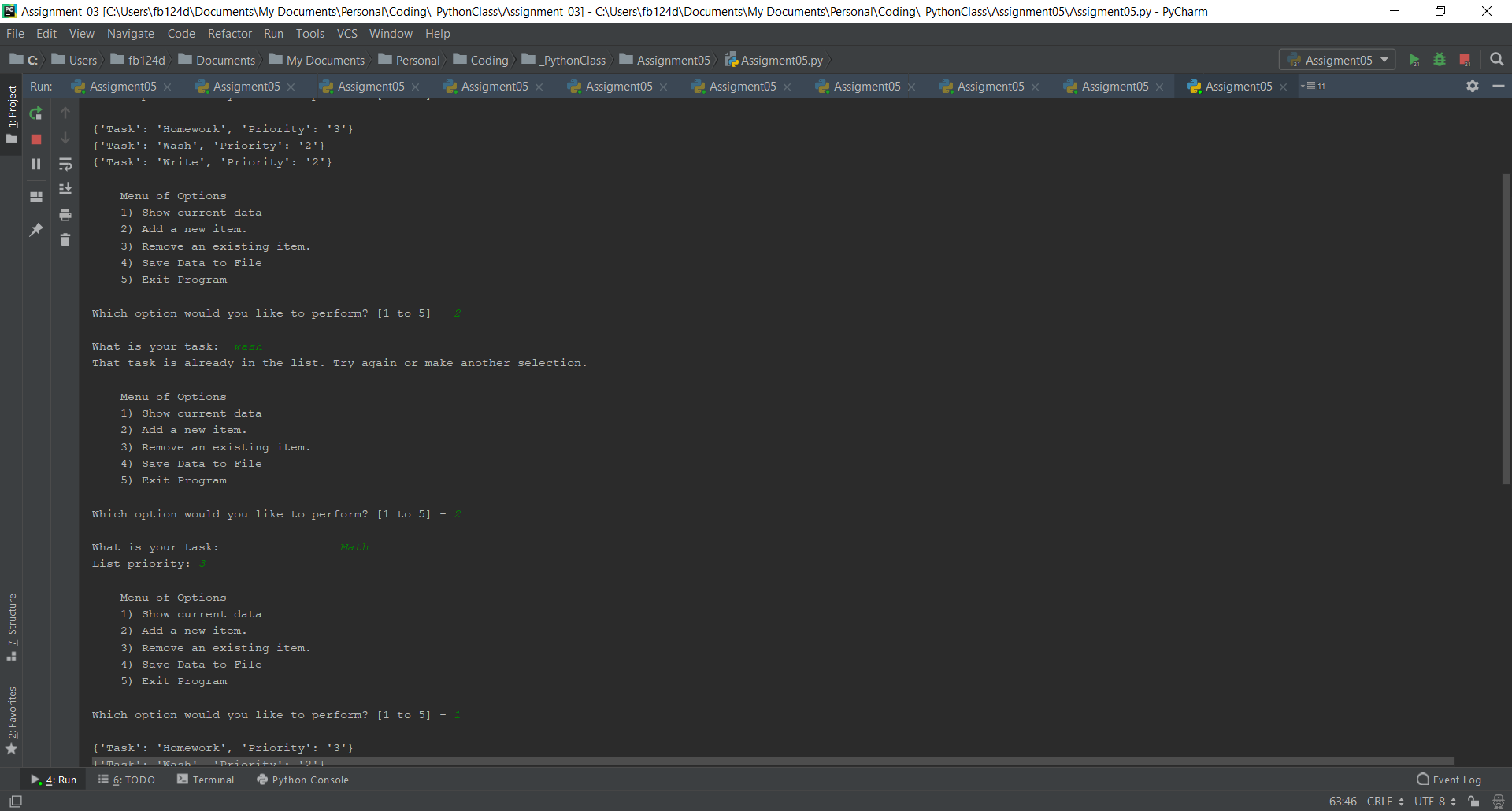


Figure : Output with spaces and multi-case

## 

Figure : Output in PyCharm with spaces and multi-case (cont.)

Figure 16 shows a similar output but in the Command Prompt. We first add “math” to the list. Then we try to add it in caps with spaces and we see it’s in the list. Then we try to delete it in a different case again with spaces and we see it is successfully deleted.

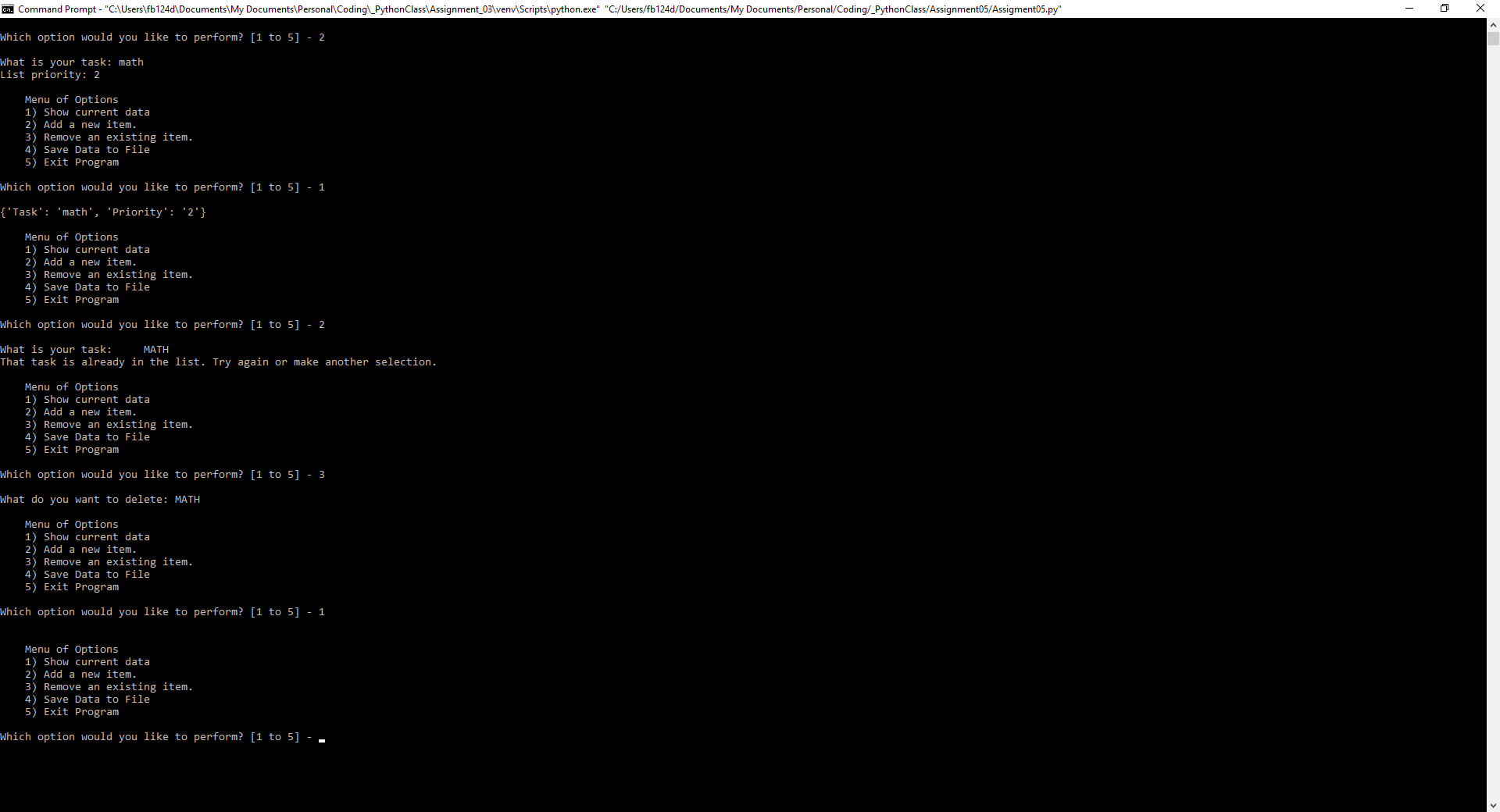


Figure : Command Prompt Output with multi-case input with spaces

## Conclusion

In this module we learned how to use dictionaries and lists to represent data in a table for a to-do list. We created a menu allowing the user to see existing data, add more, delete data, save it to a txt file and then exit. We also introduced the concept of Separation of Concerns, and separated our code in that fashion to make it more organized and professional.