*Yong Son*

*Nov 14, 2020*

*IT FDN 110 A Au20: Foundations of Programming: Python*

*Assignment 05*

**Collection of Data: DICTIONARIES**

**SEPARATION OF CONCERNS**

Introduction

In Module 05, I’ve learned about Dictionaries and how it is different from using other elements like Tuples and Lists when working with collection of data. I’ve also learned about separation of concerns and how we should be dividing entire code into major sections of branches. Coding part of Module 5 assignment is similar to assignment 04, I will be saving and importing data from text file using different user inputs.

Lists and Dictionaries

Lists are a simple way to hold collection of objects and it is most common one from other elements such as Tuple or Strings. Dictionaries are very similar to Lists but Dictionaries replace the index subscripts with “key (character) subscripts”.

Below are some of the notes saved from assignment 04:

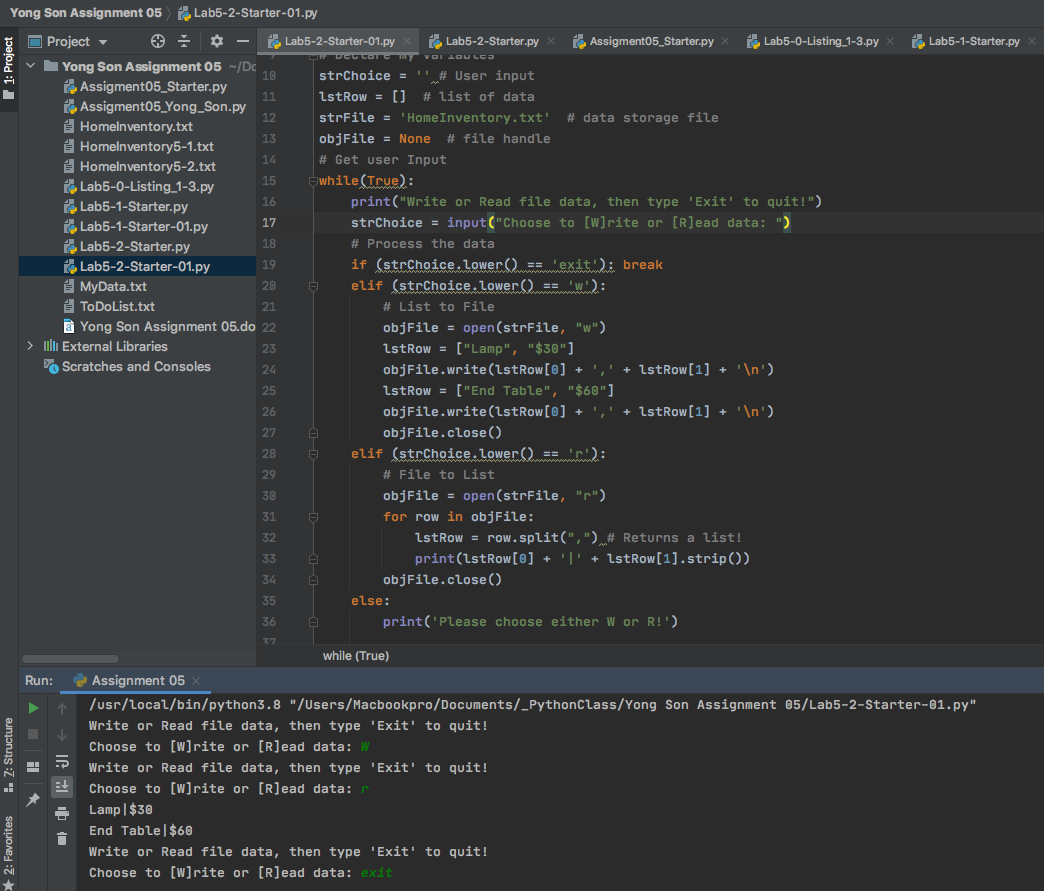
Lists = use [ ] operator for collection of data, flexible usage

Tuples = use ( ) operator or nothing for collection of data, immutable

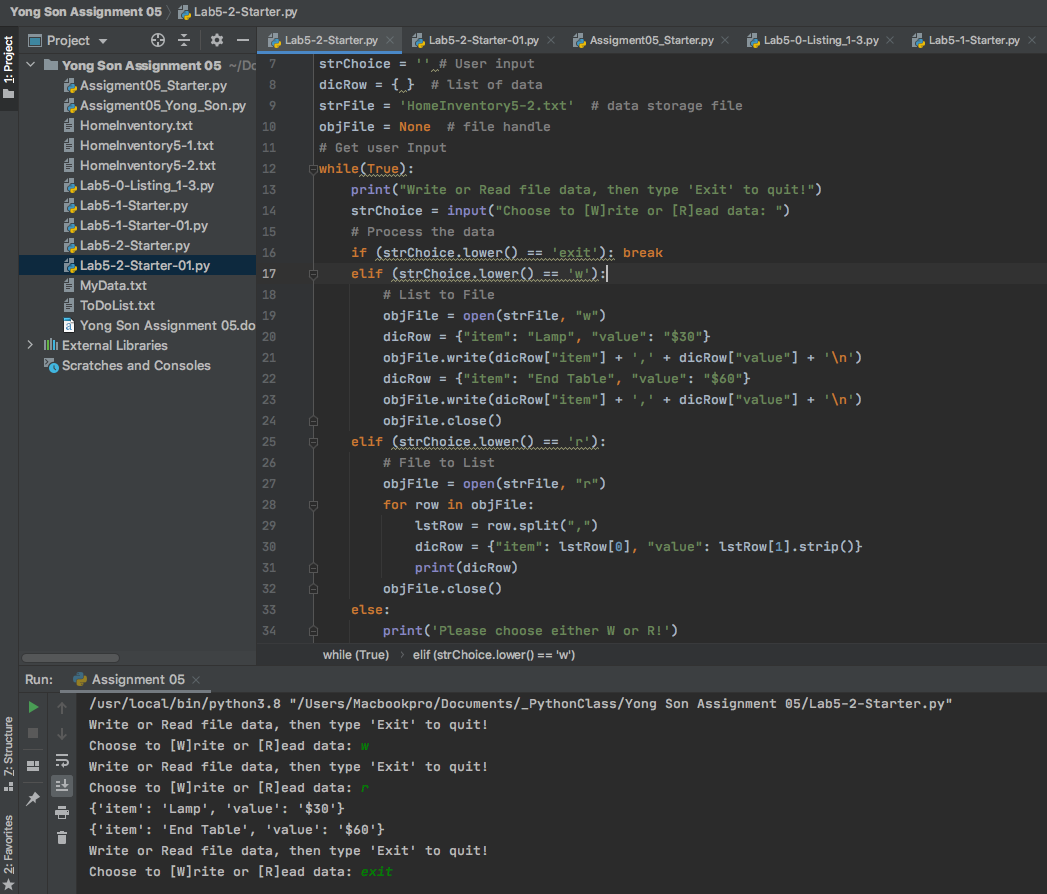
Strings = collection of individual characters

Range = List of numbers within a given range

We use { } operator to indicate that we want collection of the data to be dictionaries. As you can see the comparison between Figure5-1 (use Lists to store data) and Figure5-2 (use Dictionaries to store data), there are few differences between Lists and Dictionaries. List stores data into a row with index (numeric) subscripts but Dictionaries replace the index subscripts with key (character) subscripts.



***Figure 5-1: Home Inventory script using List***



***Figure 5-2: Home Inventory script using Dictionaries with key subscript***

Separation of concerns

“In computer science, separation of concerns (SoC) is a design principle for separating a computer program into distinct sections, so that each section addresses a separate concern. A concern is a set of information that affects the code of a computer program.” <https://en.wikipedia.org/wiki/Separation_of_concerns>, 2019

Generally, program scripts can be divided into three major sections: Data, Processing and Presentation. Data section will contain all declare variables, Processing section will contain performance tasks to deal with data, and Presentation section will contain all input and output commands.

Pseudo-code

Following the SoC requirements, my Pseudo-code is also divided into three major sections: Data, Processing, and Presentation (Input and Output). Assignment-05-starter file also contains the pre-written scripts with Data, Processing, and Presentation sections. Following are Pseudo-code instruction that I wrote for Assignment 05. Goal: create a program script that imports the data from text file and displays it on the screen. Script has 5 different menu options: 1. Display current data, 2. Ask for input, 3. Delete items per user’s input, 4. Save new data into text file, 5. Terminate the program.

Following the SoC guidelines, following Pseudo-code has been created.

Data: Declare variables and constants. Also declare list, dictionaries and Table.

Use Str, File, Row and Table to declare variables and constants

Processing: Data load instructions using dicFile and for-loop from ToDoList.txt

Use For loop to go over all rows in data table

Change data into list row and create dictionaries in the row

Append to the table

Presentation (Input/Output):

Display menu of options to user

Ask for a user input for menu choices

Create 5 if statement for all 5 options

Option#1: Show current data

For loop to go over the list table and show all rows in the table

Option#2: Ask for input data to add

Save input data into the Row

Append to Table

Option#3: Ask for input data to delete

For loop to go over all rows in the table

Compare Key and Values with input data

If “True” delete the entire row, if “false” continue to next row

Option#4: Save the table of data into text file

File “write”

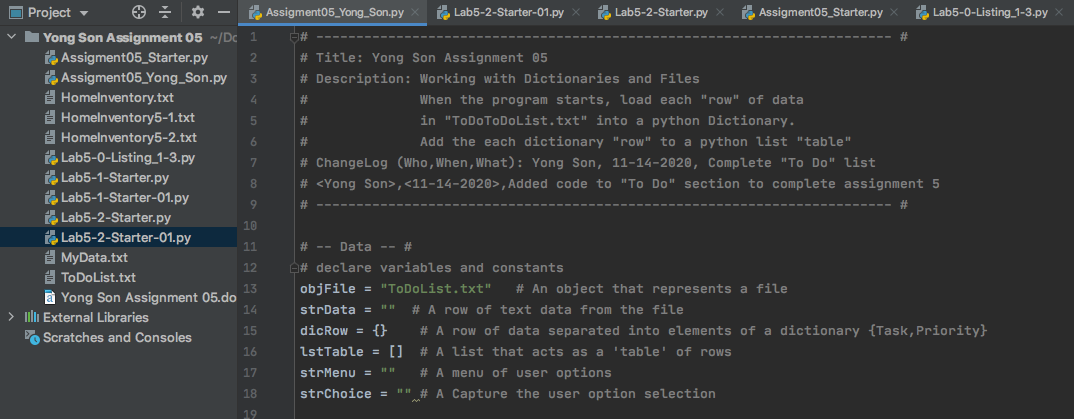
For loop to go through all rows in the table

Option#5: Exit

Exit out the program

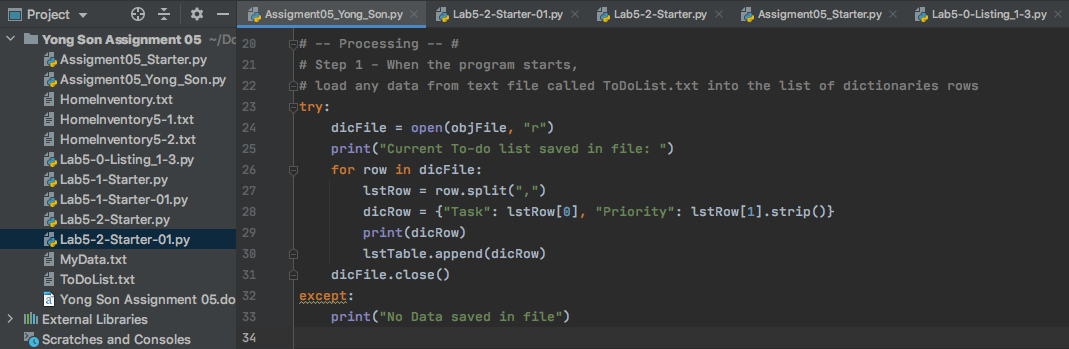
Coding

Data: All variables, lists, dictionaries and tables are pre-declared (Figure 5-3).



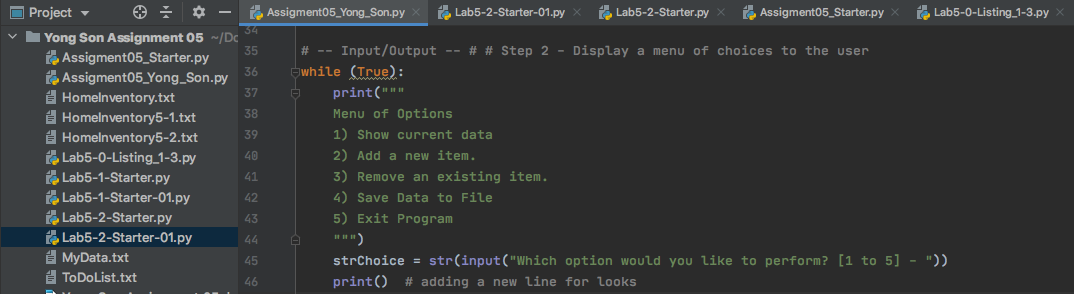
***Figure 5-3: Data section with declare values***

Processing: I used open (read) function to open the text file and import all data into dicRow { } and save it in lstTable [ ] (Figure 5-4). For loop has been used to go over each rows in the text file and save each items and priorities into dicRow. These data saved in dicRow then will be printed and added to lstTable. If there’s no data saved in text file, program will print out “No Data saved in file”.



***Figure 5-4: Processing section to open and read text file from hard drive***

Presentation: All menu options and input and output options of Step 2 are predefined (Figure 5-5).



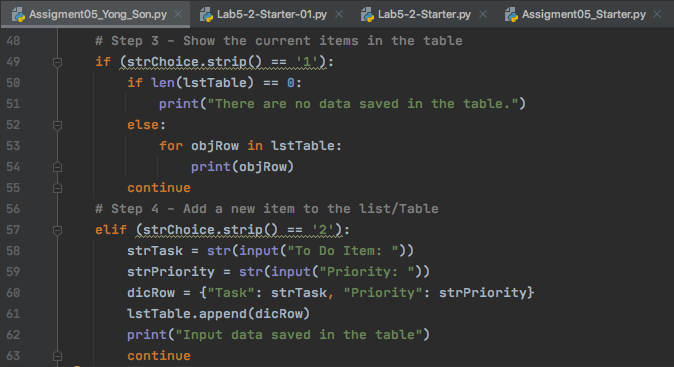
***Figure 5-5: Display all menu options***

Presentation: Option#1 – Display current data

If the user choose menu option 1, program will go through “if and else” statement to display current items in the table. If lstTable does not carry any index value then program will print out “there are no data saved in the table”. Other wise program will go through each row in the lstTable and print items and priorities saved in each row. (Figure 5-6)

Presentation: Option#2 – Add new item to the data table

If the user chooses menu option 2, program will ask for new to-do item and its priority and save these data as string characters. String character item and priorities then be added to dicRow as dictionaries and append to lstTable. New row will be added to lstTable’s existing data. (Figure 5-6)



***Figure 5-6: Menu option 1 & 2***

Presentation: Option#3

When menu option 3 is entered in program will go through the “for loop” and “nest for loop” to compare input data with existing to-do items. “For loop” will loop through each rows in lstTable and “nest for loop” will go through each items saved in the row and compare it with input value. (Figure 5-7)

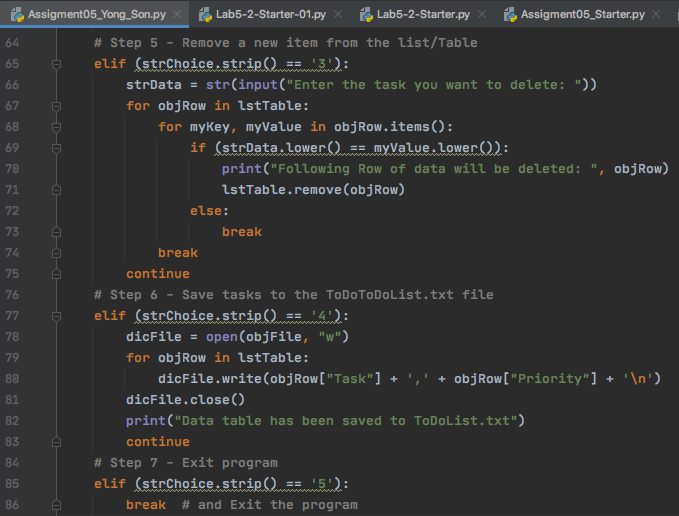
If program finds any matching items, program will print out the data saved in that row and remove the row from the lstTable. If there’s no matching item then “for loop” will move on to the next row in the table and continue. (Figure 5-7)

Presentation: Option#4

Menu option 4 will save all existing data into text file. Open “file” with “w” (writing) option will write all data into the text file. “For loop” will go through each row in lstTable and write each Task and Priority items in to the text file. (Figure 5-7)

Presentation: Option#5

I did not write any code for menu option 5 because “break” should be good enough to terminate the program when user inputs option 5 to exit the program. (Figure 5-7)

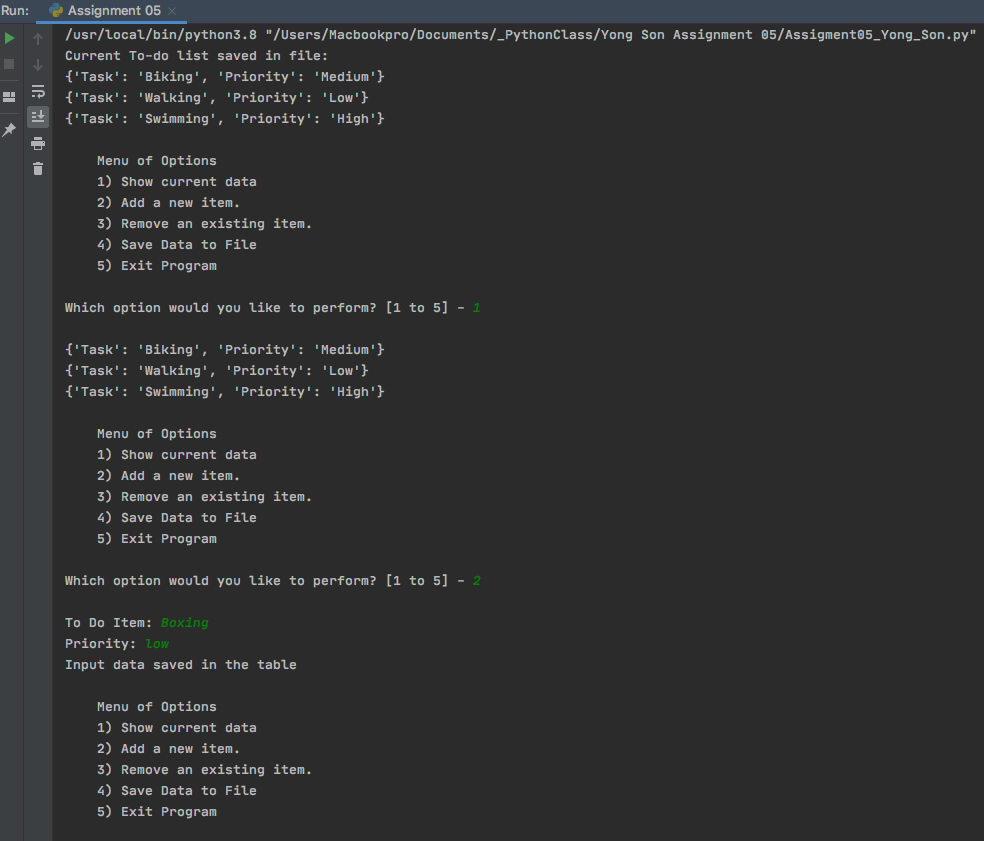


***Figure 5-7: Menu option 3, 4 & 5***

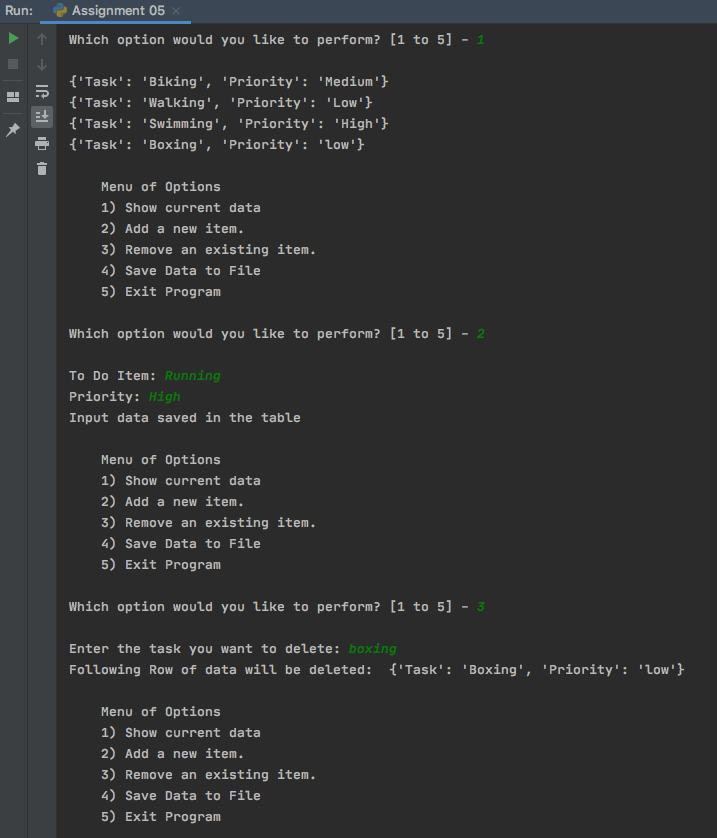
Result

Program result is showing on Figure 5-8a to Figure 5-8c. When program starts, program will import all to-do lists from the text file and display it on the screen. Menu option will show the same result since there’s no new to-do list added at this moment. When user enters menu option#2, program will take new to-do list item and its priority and save it in the lstTable. (Figure 5-8)

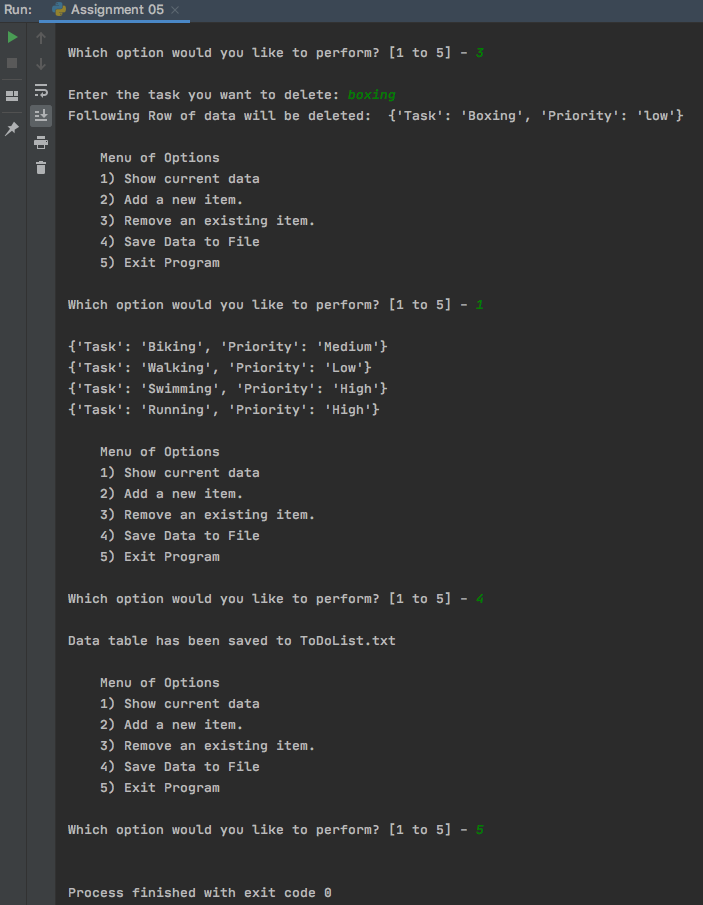
When user chooses menu option#3, delete item must exist in the current table otherwise program will not delete anything from the lstTable. Menu option #4 will save all data in the lstTable into text file and option #5 will exit out the program. (Figure 5-8)



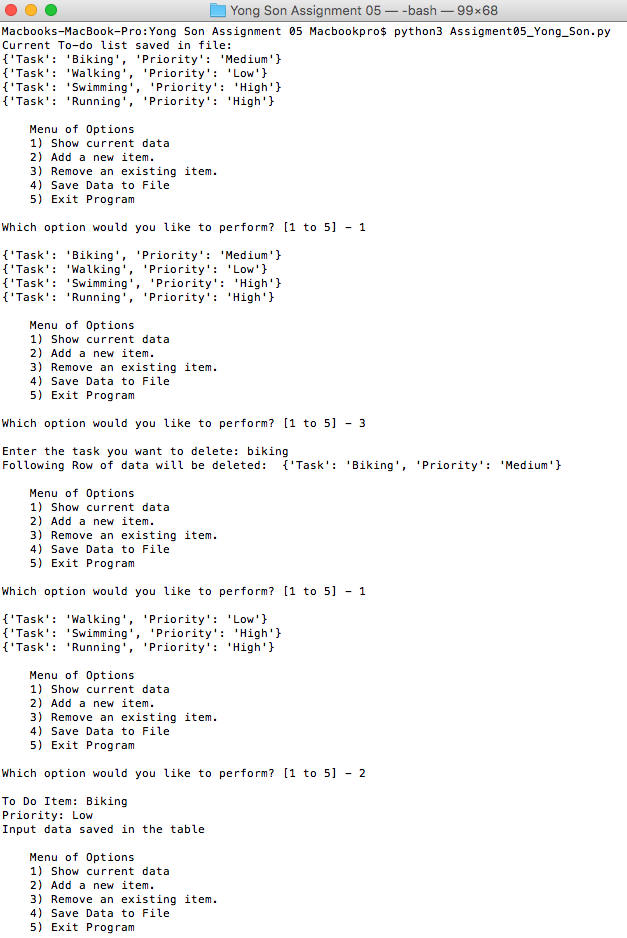
***Figure 5-8a: Result showing in Pycharm***

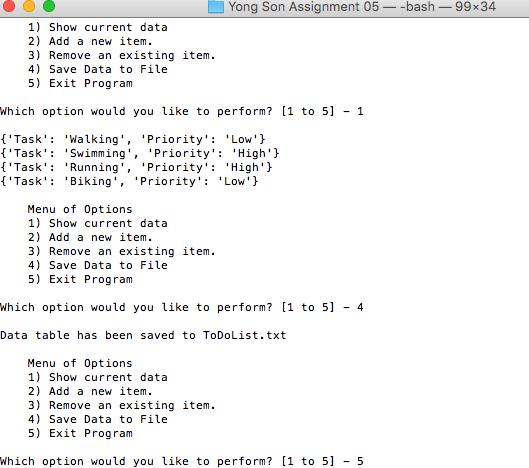


***Figure 5-8b: Result showing in Pycharm***



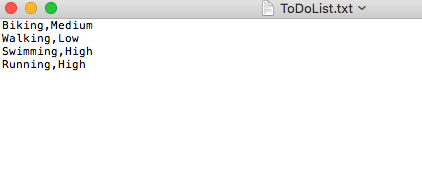
***Figure 5-8c: Result showing in Pycharm***

******

******

***Figure 5-9: Result showing in Terminal***

Final product saved as text file format. (Figure 5-8)



***Figure 5-10: text file saved in hard drive for assignment 05***

Conclusion

In this assignment, I’ve learned about Dictionaries and how it is different from using Lists or Tuples. I’ve also learned about separation of concerns and why it is important to divide the script into Data, Processing, and Presentation sections. It is much easier for the programmer to write scripts since all sections are organized in standard format and it is also much easier for other programmers to read and find out the intention of the program by reviewing each section of the scripts.