Poorval Joshi

Feb 15th, 2021

IT FDN 110 B Wi 21: Foundations Of Programming: Python

Assignment 5

# Write a python script to maintain a “TO DO” list using python dictionaries and lists

# Introduction

In this assignment, I added code to the python script Assigment05\_Starter.py that allows the user to add, delete or view the contents of a To Do List and ultimately allows the user to save the contents of the list in a file on a drive. I used dictionaries and list sequence types in python for writing this code. Lists contain a collection of items and each item has an index number beginning with zero to indicate its position in the list. Lists are declared using [] and there are several in-built functions available to modify the list. Dictionaries also contain a collection of items but are declared with {} and rather than an index, they contain keys (string characters) that essentially serve as columns to hold a particular kind of data. A declared dictionary can be thought of as a row of data that can be read into a list to form a collection of rows which allows us to form a multi-dimensional list or a table. To write the script for this assignment, I launched PyCharm and opened the Class 5 project and modified the python file called Assigment05\_Starter.py as shown in Figure 1.

The script begins with a script header that includes inline comments marked by a # at their start to indicate the title, developer, date and changes made to the python script file. After the comments comes the separation of concerns which categorizes the code in different sections such as data, processing and input/output sections. Each section contains an executable code.

In the data section, all constants and variables are declared with a comment describing each declaration and the menu that will be displayed to the user is also entered here. In the processing section, I create a ToDoList.txt file and enter data into it. The data I entered are tasks and priorities declared in three separate dictionary objects. Each dictionary object is then written to the file in the form of two columns indicating the task to be performed and its priority separated by a comma. Then the file is closed.

In order to read in the data from the ToDoList.txt file, I open the file again using the open() function in the read mode (“r”) and read in the data as a python list of dictionary rows into lsttable. This allows us to read the data from the file on the drive into memory.

The input/output section begins with the use of a while loop which essentially allows to print the menu of options multiple times as long as the condition is true. Because the while loop can run indefinitely, I have used the command continue and break at the end of the if and elif statements. The continue statement forces the loop to go back and check if the condition is still true and the break statement allows to break out of the while loop and exit when user makes that choice. Then the input () functions asks the user to choose an option from the menu. The type of option chosen by the user is evaluated by the conditional if and elif statements and the equity operator (==) that allows to run a specific set of commands based on the user`s choice.

If the user chooses option 1, then the first if statement is true and the contents of the lsttable are displayed to the user by using a for loop to read each row in the lsttable and print it to the user. The for loop is a loop that runs for a definite number of times based on the number of rows that are in the lsttable and automatically breaks out of the loop after the last row.

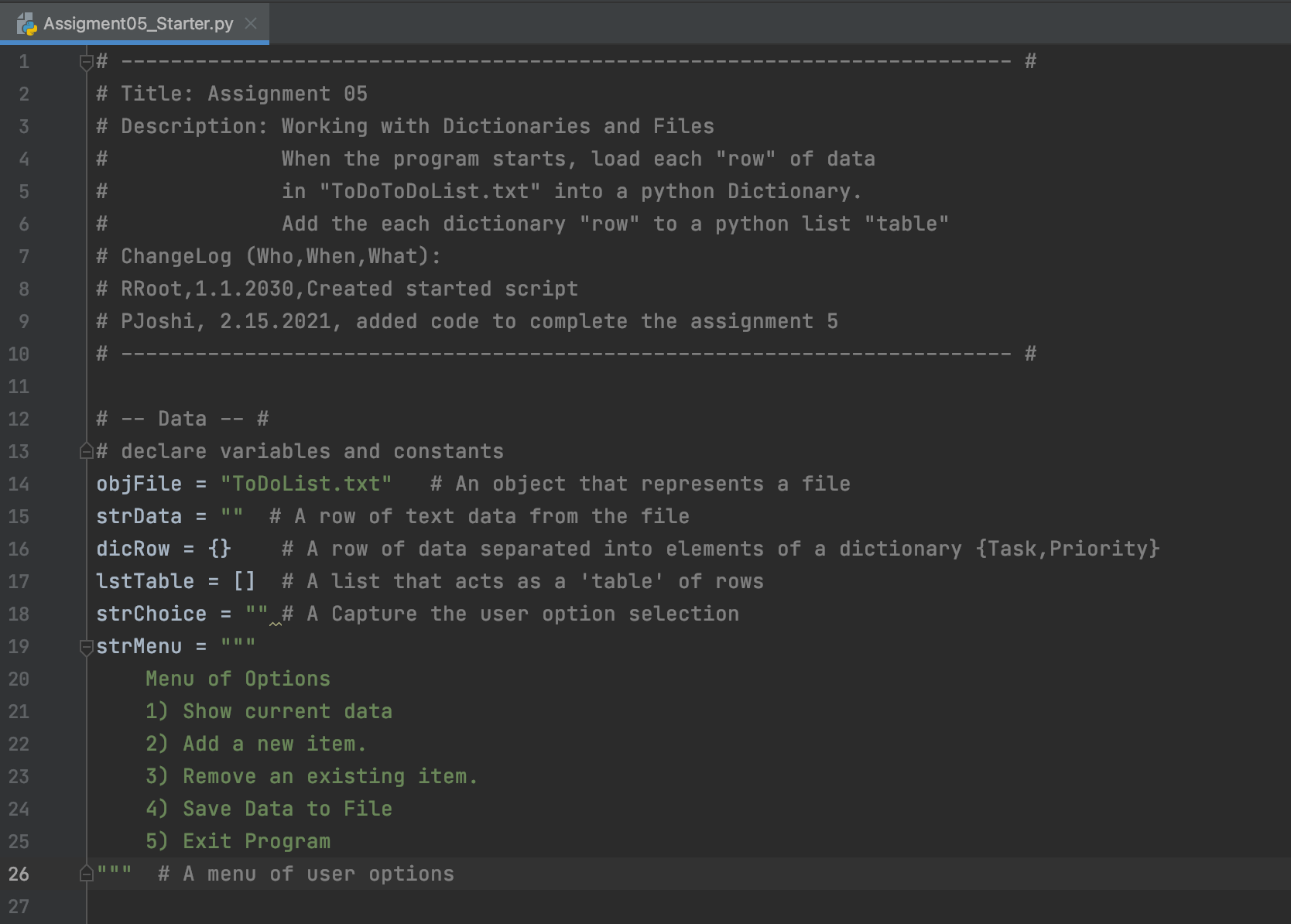
If the user chooses option 2, then the second elif statement is considered true and it allows the user to enter the name of a task and its priority as string variables. Both string variables are then used to create a dictionary object called newdicRow. This newdicRow gets appended to the lsttable using append() command such that any more entries made by the user as a newdicRow get stored as a multi-dimensional list or table in memory.

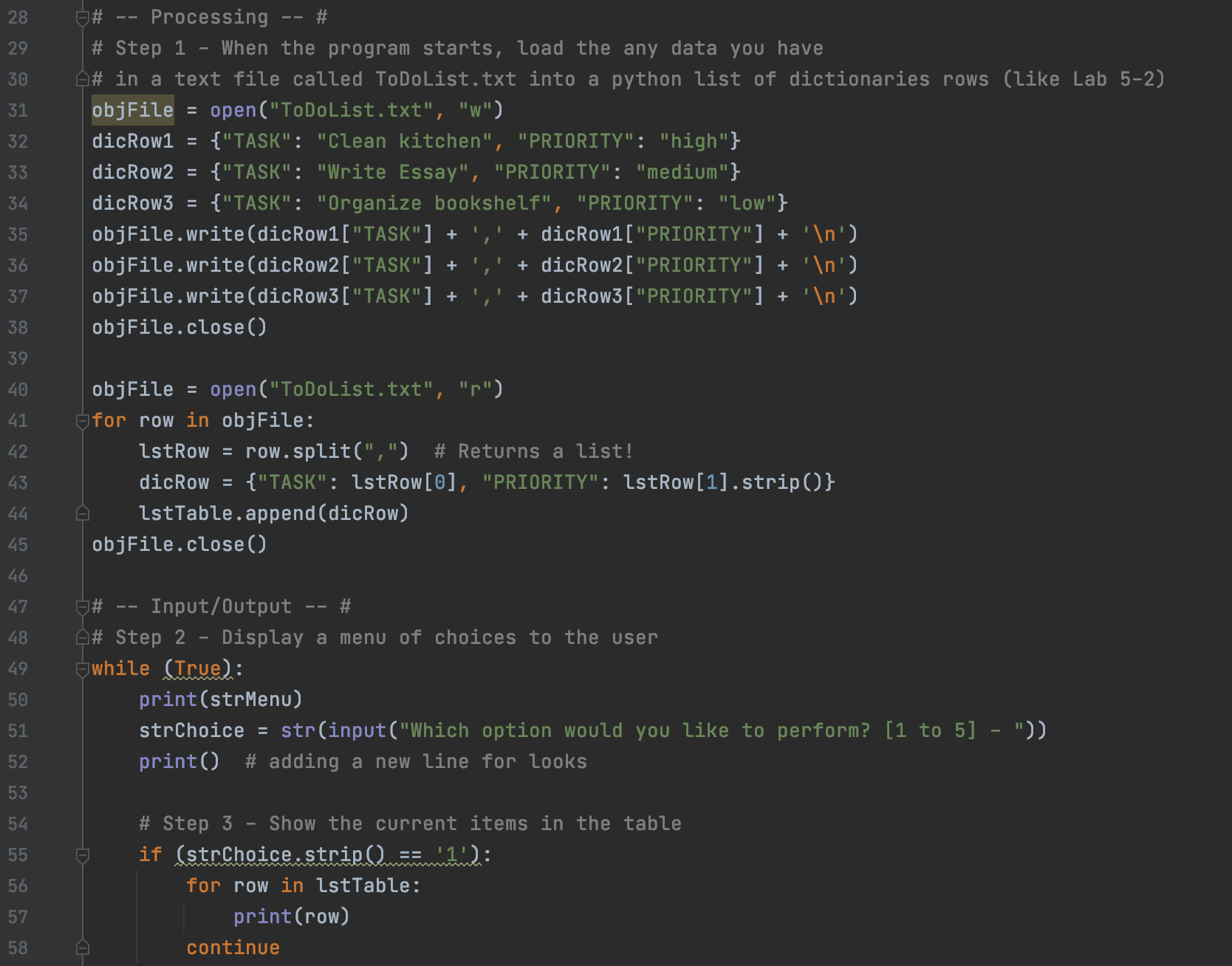
If the user chooses option3, then the third elif statement is considered true and the input() function asks the user for the name of the task they wish to remove as a string variable called edit. Then a for loop reads each row in the lsttable and matches the user`s input to the Task key in the rows using an if statement. Once a match is found, the entire row is removed from the lsttable using the remove() function.

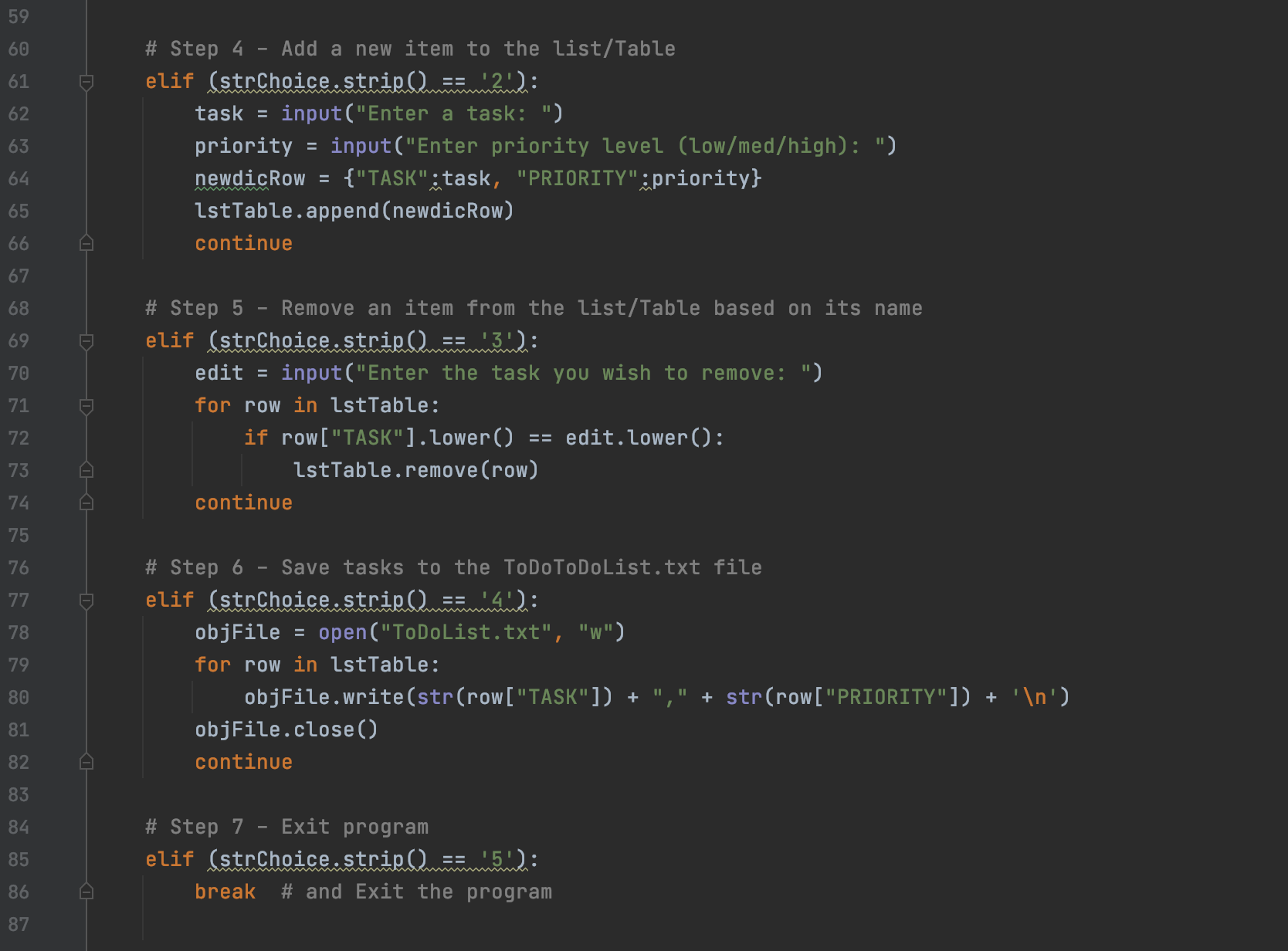
If the user chooses option 4, then the fourth elif statement is considered true and the ToDoList.txt file is opened using the open () function and data in the lsttable is overwritten over the existing data in this file by the use of the “w” argument. The write () function then writes the contents of each row of the lsttable to the ToDoList.txt file using the concatenation operator between the dictionary keys of task & priority and a visible comma character to separate the elements and finally inserting a carriage return (‘\n’) at the end so that the next row contents go to the next line. The close () function closes the file.

If the user chooses option 5, then the while loop gets broken and the program ends.

This script ran successfully in PyCharm as shown in Figure 2 and was able to edit and save the ToDoList.txt file with the user input as shown in Figure 3.



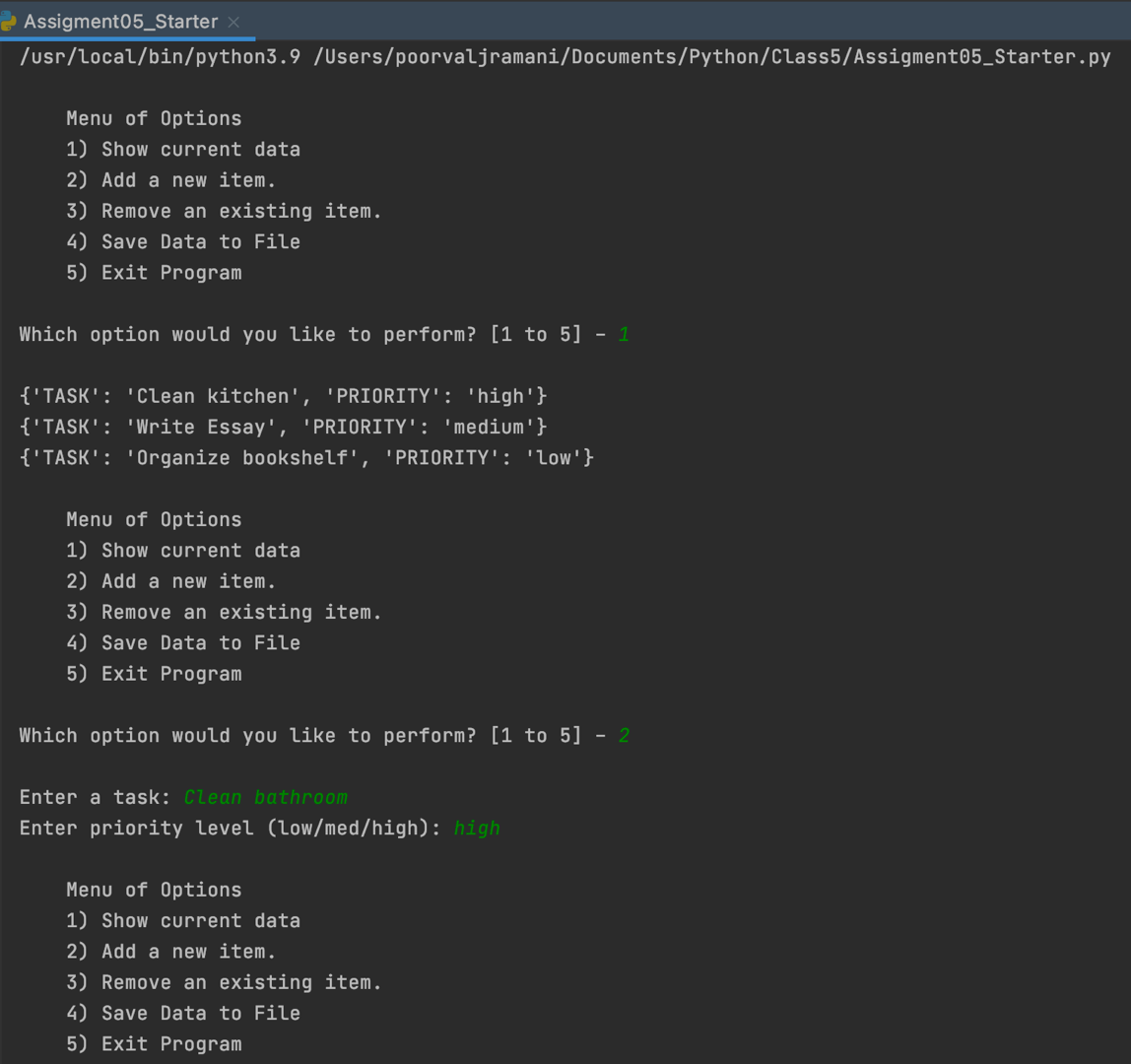
****

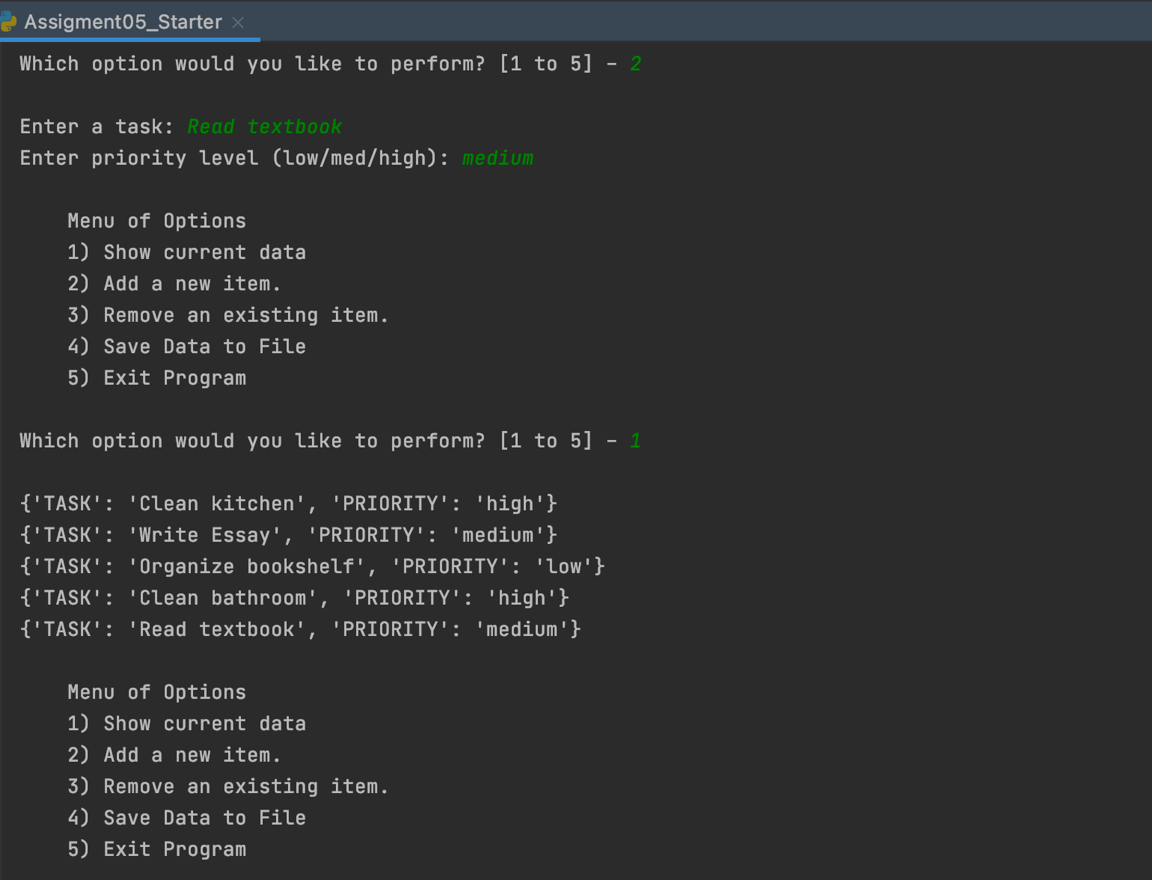
****

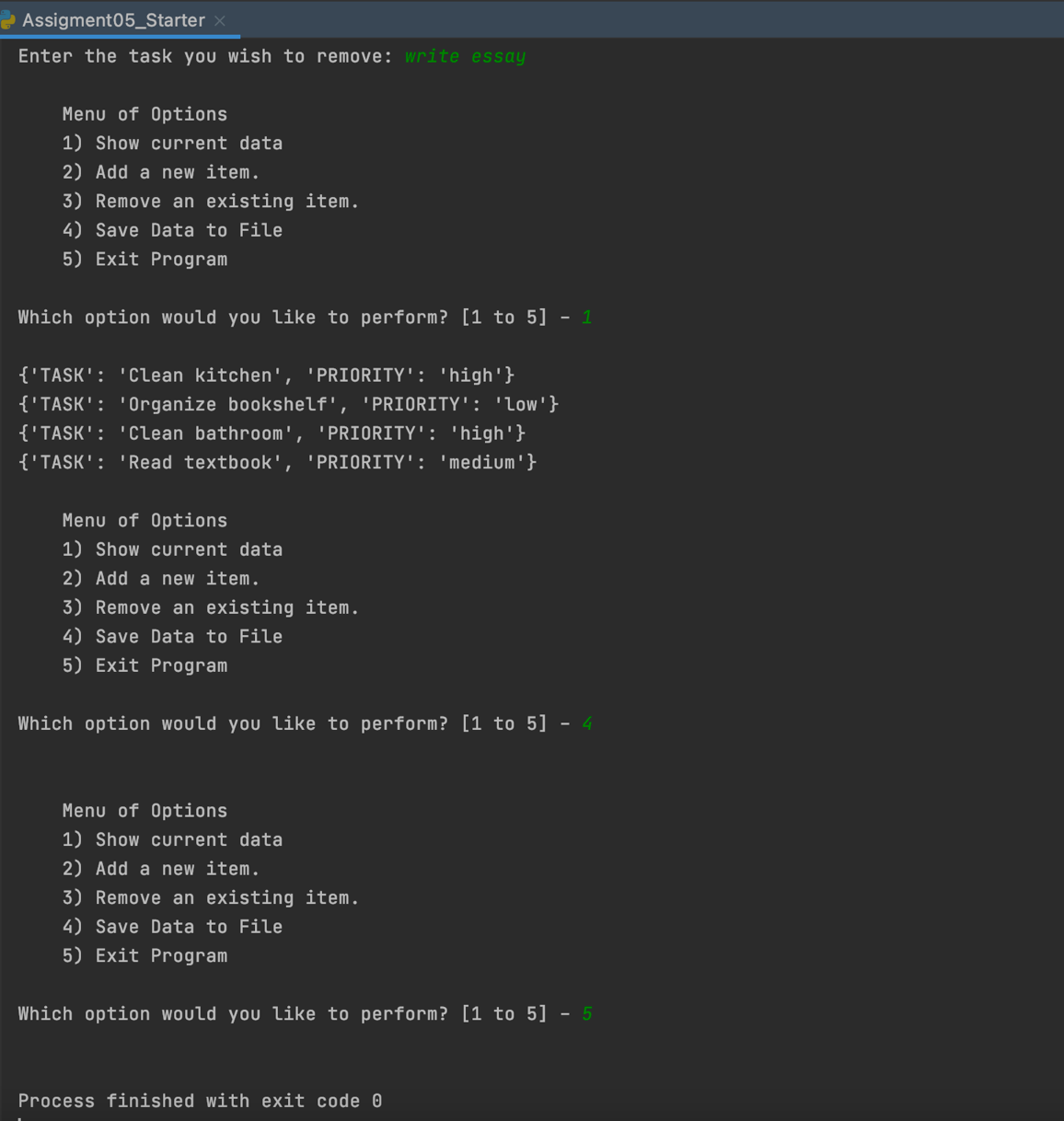
**Figure 1: Assignment5 script in PyCharm** **to create a To Do list.**

# Running the script in PyCharm

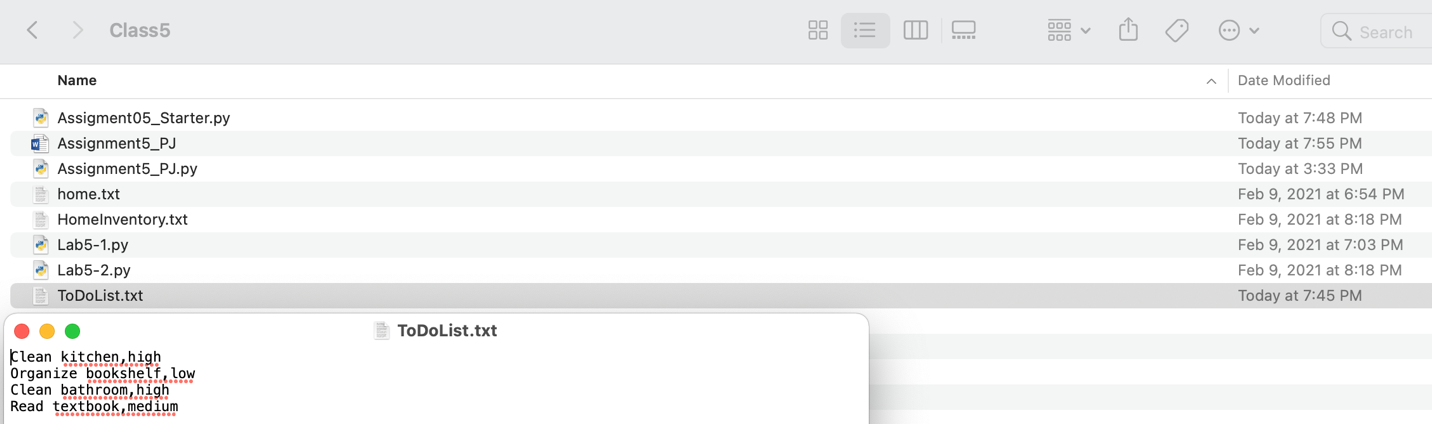
To run this script in PyCharm, I right clicked in the PyCharm open window and selected run Assigment05\_Starter.py and the script ran as expected as shown below in Figure 2.





****

**Figure 2: Running the script in PyCharm**

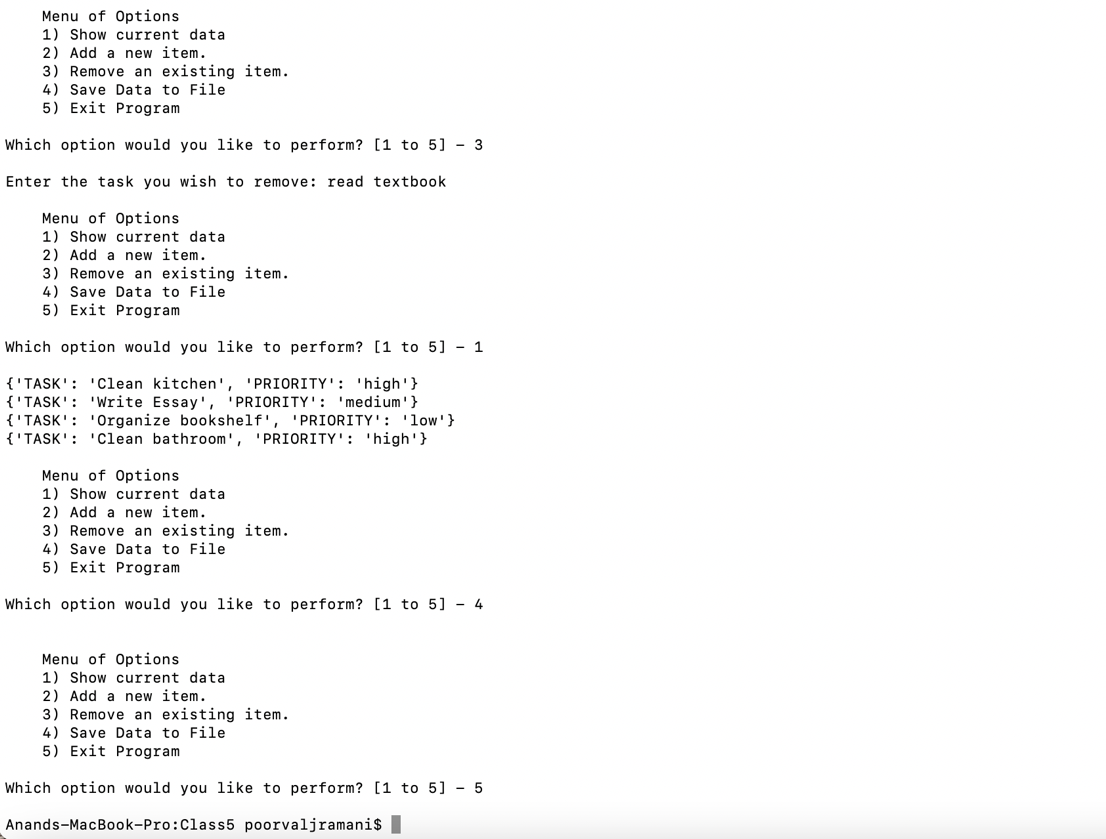
****

**Figure 3: Verifying that the ToDoList.txt file has the saved the data**

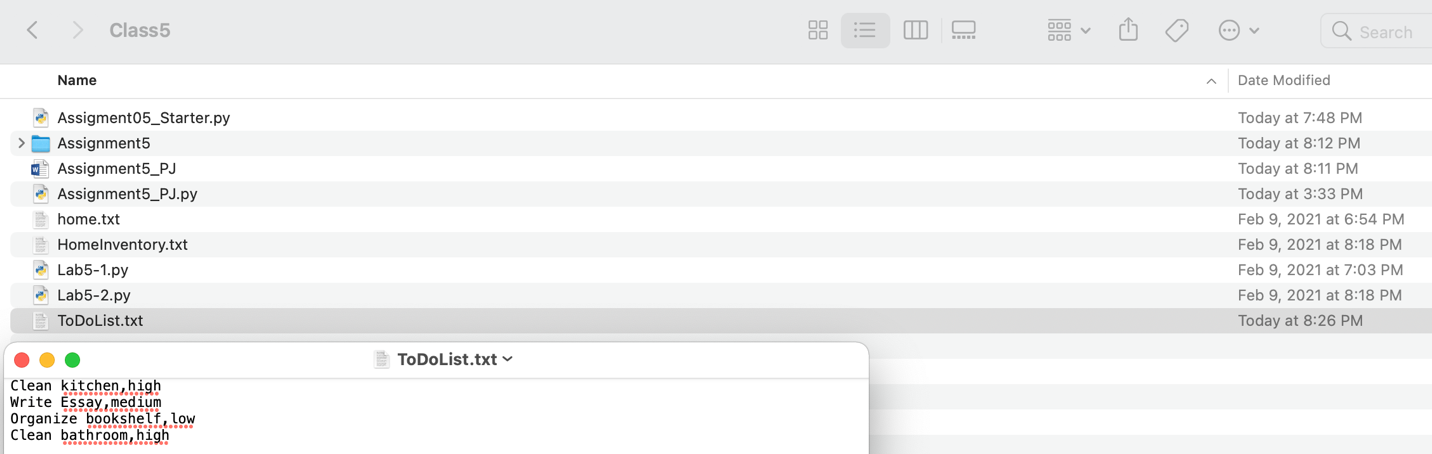
# Running the home inventory script on the terminal

To run the python script on the terminal, I first changed my directory to class4 where I had the Assigment05\_Starter.py file so that the ToDoList.txt file gets created in the same location. Then I used the Python3 command followed by the path of the Assigment05\_Starter.py file which allowed the terminal to create a python environment and run the python script. The script ran as shown in Figure 4 below and created a ToDoList.txt file in the same folder as the script as shown in Figure 5.

# 



**Figure 4: Running the script on the terminal**



**Figure 5: Verifying that running the script on the terminal created a ToDoList.txt file and saved data**

# Summary

In summary, I have become more comfortable using lists and dictionaries in python. I am now able to declare lists and dictionaries, add data to them or collect data from the user to add to them, print their contents, remove their contents and save their contents to a file. Additionally, I have some working knowledge of how to upload my code in GitHub and its explanation in this readme file to make the code easier to understand.