Tao Ye

May 14, 2020

Foundations of Programming: Python

Assignment 05

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

Python Script Using Dictionary

Introduction

In this assignment, I will add codes to complete a Python script to manage a "ToDo list." The "ToDo list" file contains two columns of data, "Task" and "Priority” that are read into a Python Dictionary object. Each dictionary object represents one row of data, and these rows are added to a Python List object to create a table of data. The script will be tested and run in both PyCharm and a Windows command shell as a console application.

Add Codes to the Python Template in PyCharm

For this assignment, I created a new folder **C:\\_PythonClass \Assignment05**. Then in PyCharm, I created a new project that uses the **C:\\_PythonClass\Assignment05** folder as its location,and added the Python template file **Assigment05\_Starter.py** under this project. For this assignment, I will add codes to this Python file according to the instructions in the comments.

* **# Step 1 - When the program starts, load any data you have  
  # in a text file called ToDoList.txt into a python list of dictionaries rows**

At this point, the file may or may not exist before the program attempt to read it; so I use a **try: except** structure to capture the runtime error in case the file does not exist yet. If the file exists, it reads the file; otherwise, it prints a message before continuing.

try:  
 fileHandle = open(objFile, **"r"**)  
 for row in fileHandle:  
 lstRow = row.split(**","**)  
 dicRow = {**"Task"**:lstRow[0], **"Priority"**:lstRow[1].strip()}  
 lstTable.append(dicRow)  
 fileHandle.close()  
except FileNotFoundError:  
 print(objFile,**" does not exist yet."**)

In the file “**ToDoList.txt”,** each line has a task name and a priority number separated by a comma. So the code reads each line into a list **lstRow** using **split()** function to separate task name and priority number in the list based on commas in each line. The list elements in **lstRow** are values of the dictionary key-value pair. So they are combined with keys **“Task”** and **“Priority”** to form a dictionary row which is then added to a list table **lstTable**, as shown below.

lstRow = row.split(**","**)  
 dicRow = {**"Task"**:lstRow[0], **"Priority"**:lstRow[1].strip()}  
 lstTable.append(dicRow)

* **# Step 3 - Show the current items in the table**

To list all “Task” and “Priority” rows in the **lstTable**, a column header “ Task Priority” is prinited first; followed by a for loop to run through all rows in the table to print the dictionary item values of each row **dicRow**, as shown below. Each **dicRow** of the **lstTable** is a dictionary object. The values of “Task” and “Priority” keys for each dictionary object are accessed through indexing using the keys.

print(**"Task"** + **" "** + **"Priority"**)  
for dicRow in lstTable:  
 print(dicRow[**"Task"**] + **" "** + dicRow[**"Priority"**])

* **# Step 4 - Add a new item to the list/Table**

To add a new item (row) to the Table **lstTable**, the program first asks the user to enter a task and checks whether the task is already in the list. A for loop checks each row of the list and compares the “Task” value to the user input. The string method **lower()** is used to do case-insensitive comparison. A boolean variable **taskExist** is used to record the outcome: True means a match is found and vice versa. If the task user entered is already in the list, it will not be added in the list and a message is shown stating that the task is already in the list. If it is a new task, the program then asks the user to enter the priority value. The user inputs of task and priority become the values of keys “Task” and “Priority” keys in the dictionary row, **dicRow**. The **dicRow** is then added to the end of the table **lstTable** using **append()** method of the list.

strTask = input(**"Enter a Task: "**)  
taskExist = False  
for dicRow in lstTable:  
 if (dicRow[**"Task"**].lower() == strTask.lower()):  
 taskExist = True  
 break  
if (not taskExist):  
 strPriority = input(**"Enter a Priority: "**)  
 dicRow = {**"Task"**:strTask, **"Priority"**:strPriority}  
 lstTable.append(dicRow)  
else:  
 print(**"The task"**, strTask, **"already exists in the list."**)

* **# Step 5 - Remove an existing item from the list/Table**

To remove an existing item (row) from the table **lstTable**, the program first asks the user to enter a task name, **taskChoice,** to be removed. A Boolean variable, **taskExist**, is initialized to **False** to aid in recording whether the task the user wants to remove exists in the table. The a for loop checks each row, **dicRow**, of the table and compares the “Task” value of the dictionary object in **dicRow** to the task name the user enters. If a match is found, the row, **dicRow**, is removed from the table and the Boolean variable, **taskExist,** is updated to “True” to indicate that at least a match is found. At the end, if the value of **taskExist** is false, the program prints a message stating that the task the user entered is not in the table.

taskChoice = input(**"What task would you like to delete?: "**)  
taskExist = False  
for dicRow in lstTable:  
 if (dicRow[**"Task"**].lower() == taskChoice.lower()):  
 lstTable.remove(dicRow)  
 taskExist = True  
if (not taskExist): print(**"The task"**, taskChoice, **"is not in the list."**)

* **# Step 6 - Save tasks to the ToDoList.txt file**

The "ToDoList.txt" file contains two columns of data, "Task" and "Priority”. So to save the table to the file, only the values of each row (dictionary) are saved to the file. Here, **dicRow[“Task”]** accesses the “Task” value and **dicRow["Priority"]** accesses the “Priority” value of the dictionary object (row). The two strings are written to the file on one line separated by a comma, and a new line carriage is added at the end of the line.

fileHandle = open(objFile, **"w"**)  
for dicRow in lstTable:  
 fileHandle.write(dicRow[**"Task"**] + **","** + dicRow[**"Priority"**] + **"**\n**"**)  
fileHandle.close()

* **# Step 7 - Exit program**

Before it exits, the program first saves the current table to the file "ToDoList.txt", and then asks the user to hit return before it exits. The file save has the same statement as in the step #6 above.

fileHandle = open(objFile, **"w"**)  
for dicRow in lstTable:  
 fileHandle.write(dicRow[**"Task"**] + **","** + dicRow[**"Priority"**] + **"**\n**"**)  
fileHandle.close()  
input(**"Please hit Enter to exit"**)

Run the Script

1. Run the script in a Windows command shell.

In the Windows shell, first change the current folder to **C:\\_PythonClass\Assignment05** so the data file “ToDoList.txt” will be created in the desired folder. Then type in the python command to run the script as shown in Figure 1. Since this is the very first time this script is run, the file “ToDoList.txt” doesn’t exist yet. To add some rows in the table, option “2” is selected several times to add the following tasks and priorities:

aaa, 1

bbb, 2

ccc, 3

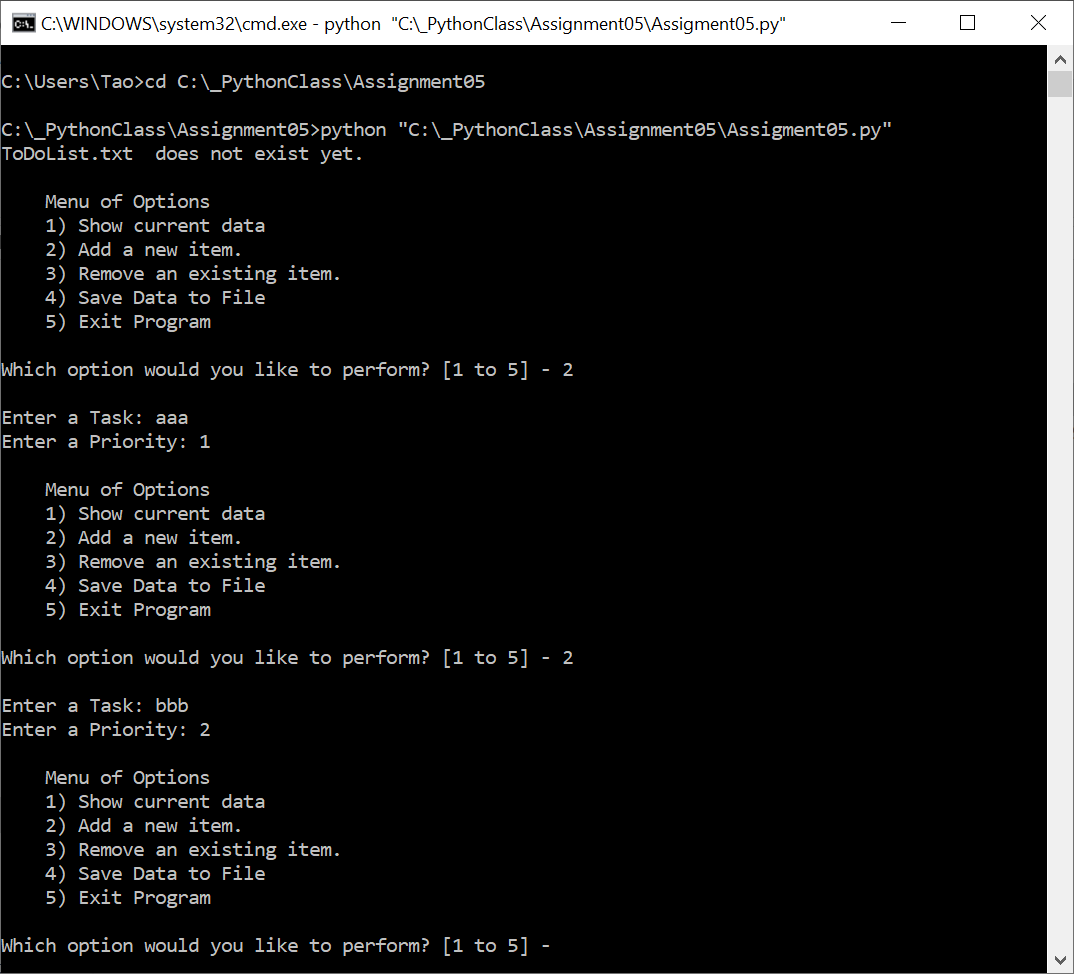


Figure 1. Script Run in Windows Shell and Option “2” Selection

Then, option “1” is selected to show the current table, as shown in Figure 2.

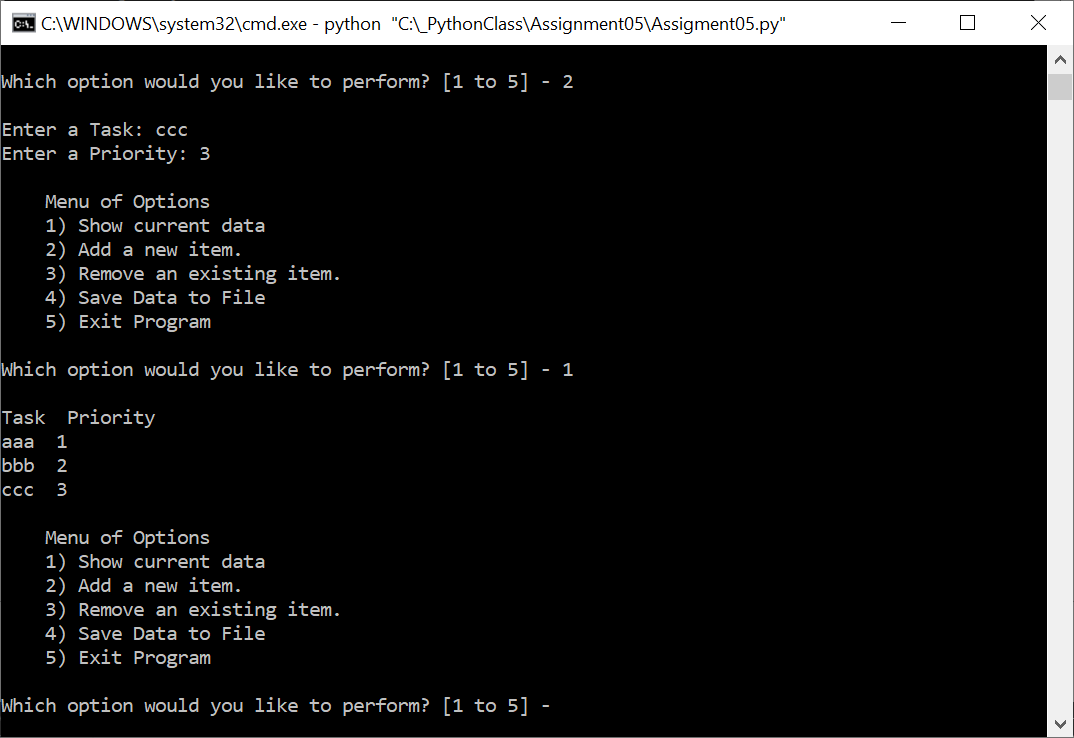


Figure 2. Script Run in Windows Shell and Option “1” Selection

At this point, option “3” is tested to remove task “bbb”. At first, I intentionally input task “bbbb” which is not in the list. The program correctly printed out a message stating the task is not in the list. Then I enter the task name “bBB”. Since the task name is not case-sensitive, the task “bbb” should be removed from the list. Subsequent Option “1” selection showed the removal worked correctly with task “aaa” and “ccc” left in the list. The operation sequence is shown in Figure 3.

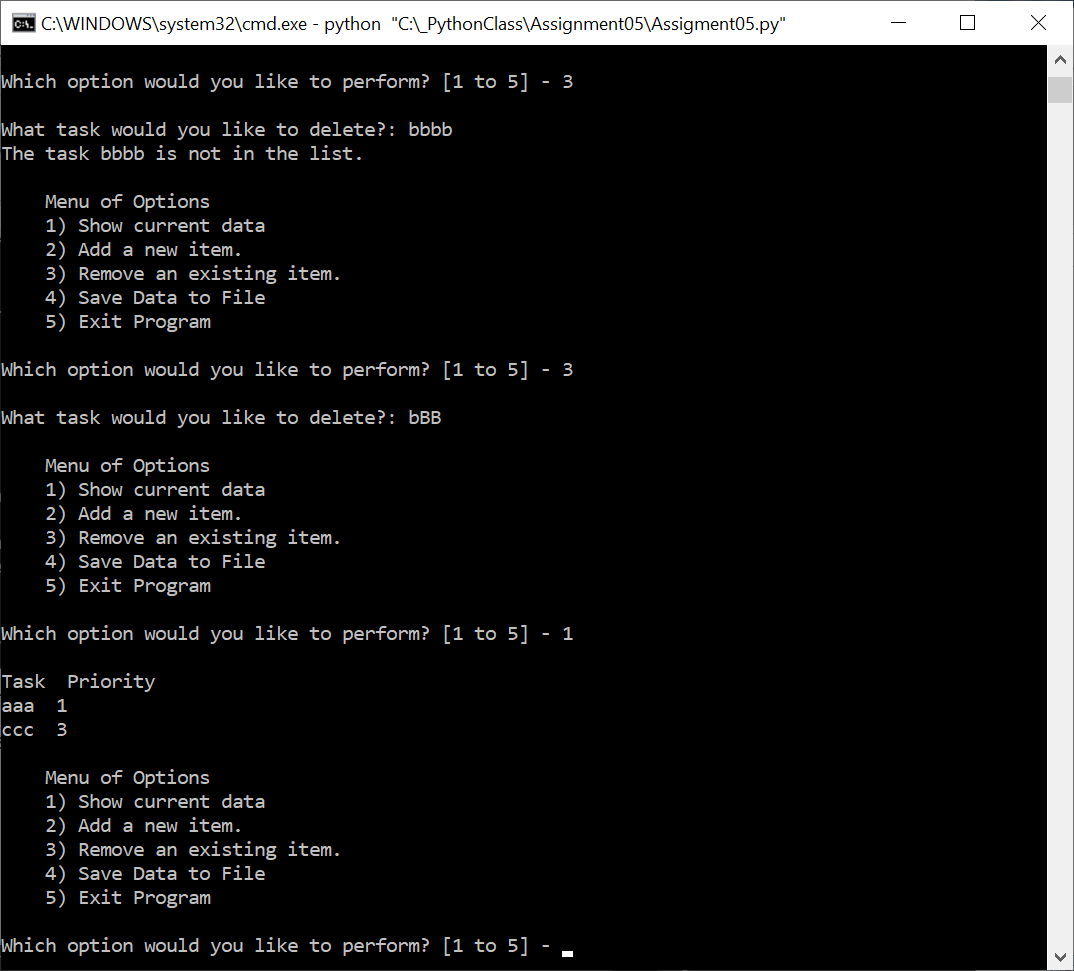


Figure 3. Option “3” and “1” Execution

Now, option “4” is selected to save the list to the file. The file “ToDoList.txt” is created in the current folder and the content of the file is shown in Figure 4 that matched the screen list in Figure 3.

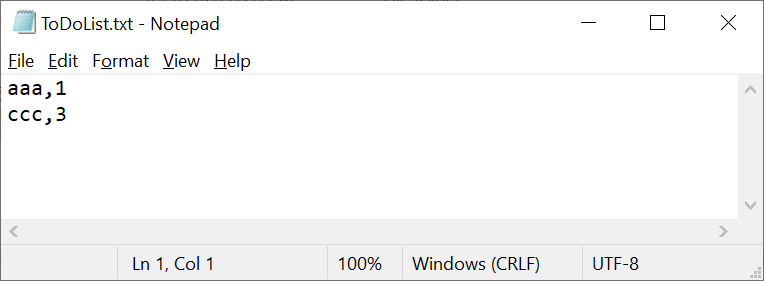


Figure 4. Content of the File “ToDoList.txt”

2. Run the script in PyCharm

After the Windows shell run of the script, the script is run in PyCharm. Now the file “ToDoList.txt” is present in the folder **C:\\_PythonClass\Assignment05**. So the program read the file content into the list in the memory and the option “1” showed the current list as in Figure 5, same as in Figure 4 where it was last saved in the file.

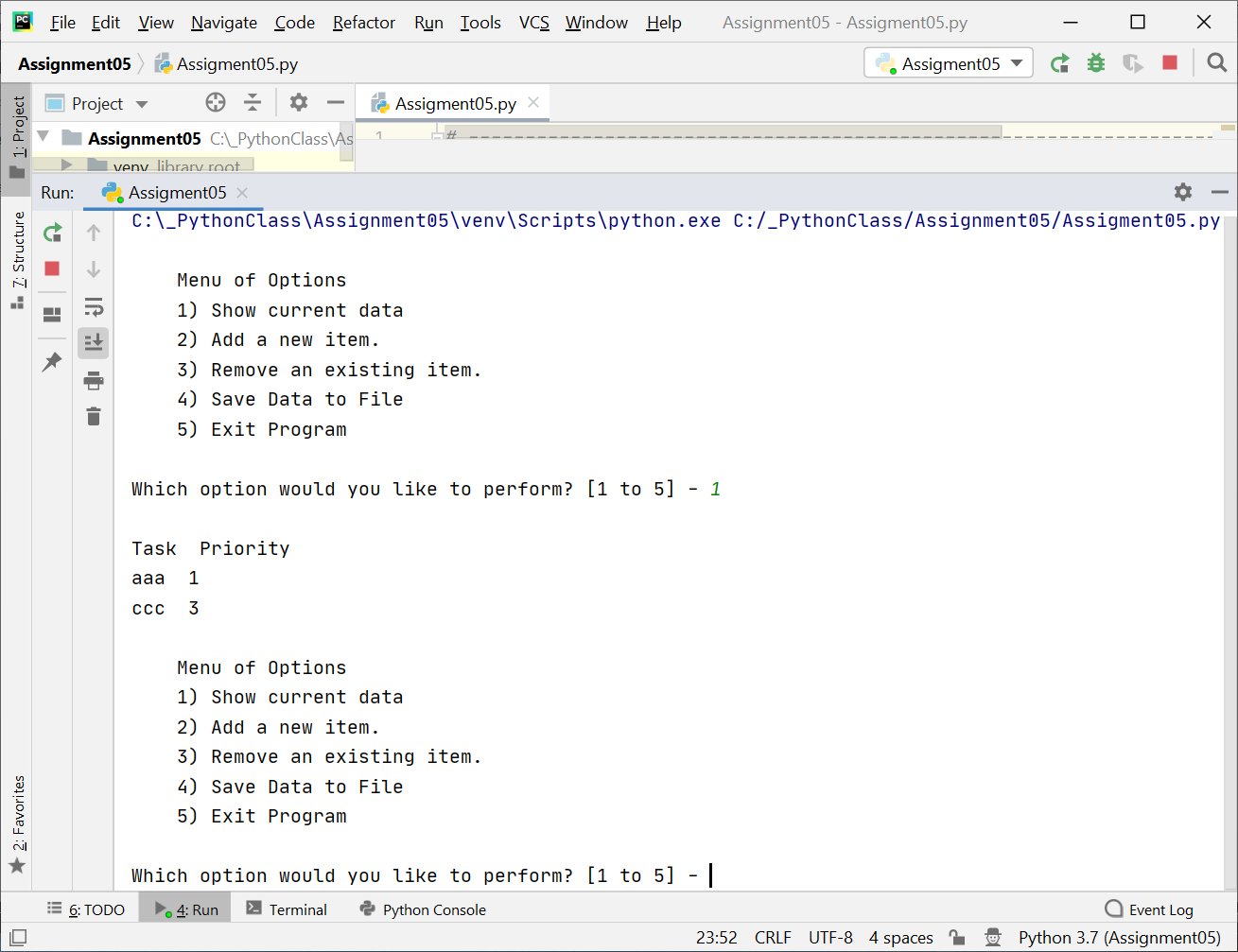


Figure 5. Script Execution in PyCharm and Option “1” Selection

Now, option “2” is selected to add task “dddd”, followed by option “1” to show the current list as

Task Priority

aaa 1

ccc 3

dddd 4

These are shown in Figure 6.

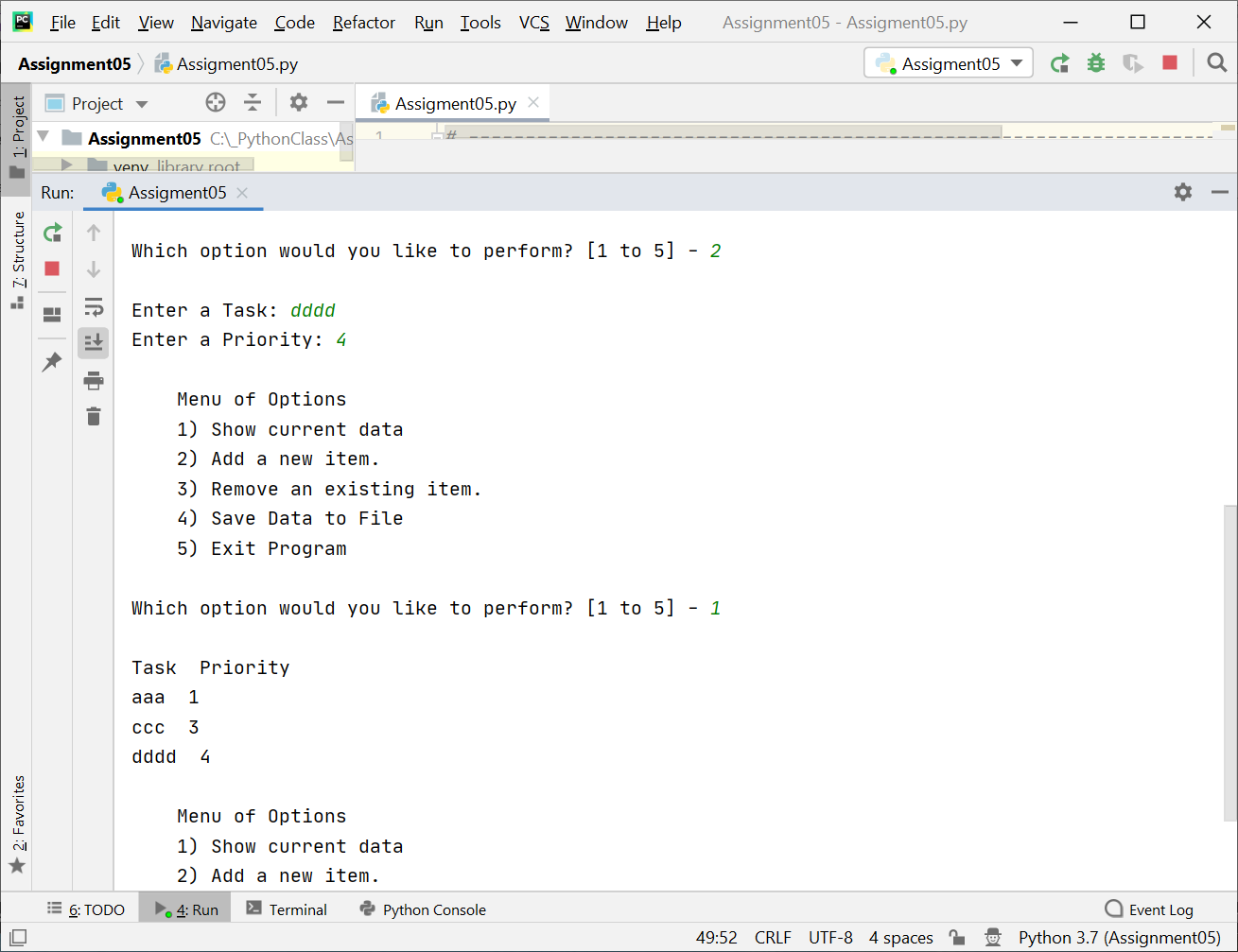


Figure 6. Option “2” and “1” Execution in PyCharm

Finally, option “5” is selected to save the list to the file and exit.

Open the file “ToDoList.txt” as shown in Figure 7, it shows the content that matches the listing of the data on the screen, as shown in Figure 6. Therefore, the current list is indeed saved to the file.

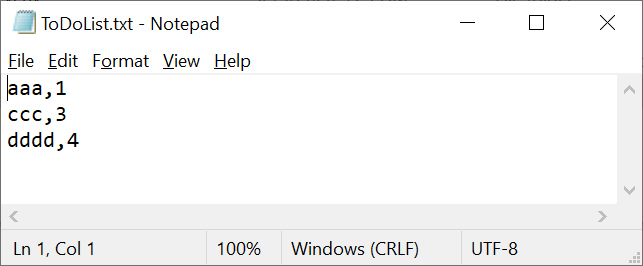


Figure 7. Content of the File “ToDoList.txt”

Conclusion

The assignment is to add codes to a Python script template to build a complete script to use dictionary object and list to manipulate the data table between the memory and a file. The list is composed of dictionary objects. The functions implemented include displaying the current list in the memory, adding new data items as dictionary objects in the list, removing selected dictionary object in the list, saving the current list in a file, etc. The program is tested in both PyCharm and Windows shell. The results show that the program implemented intended functions.