Hyo Park

August 9, 2020

Foundations of Programming: Python

Assignment 05

To Do List Script

# Introduction

For module 5’s assignment, we were asked to expand on the Assignment05\_Starter.py file. The program is a simple database program in which the user gets to read, write, delete tasks and their priorities. Because it is a menu-driven program, we were asked to complete 7 steps outlined in the file while fulfilling the necessary requirements. I felt this assignment was the toughest one so far due to the multiple required steps the program used as well as the freedom we had in filling the requirements.

Code Setup

As seen on figure 1, I wrote a pseudocode for this assignment. Looking at it after finishing the assignment I realized how bare bones the algorithm is. The fact that we were provided with starter file probably didn’t make pseudocode too useful, but I still feel it provided me a sense of direction in the right way.

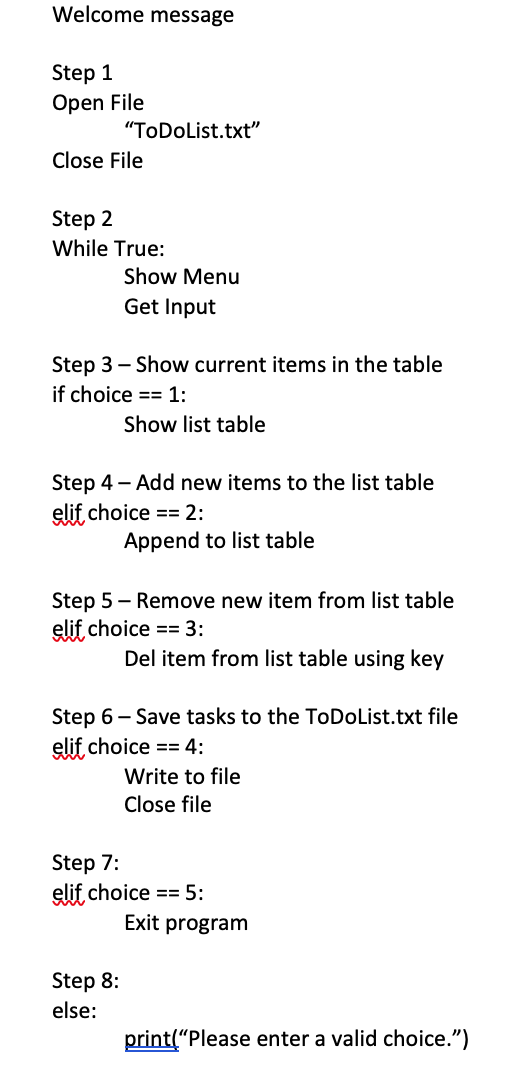


Figure 1: Pseudocode of Assignment05\_Starter.py

Main Code

I will go over the assignment’s requires steps one by one.

Step 1 – In the first step of the program, the text file “ToDoList.txt” file is read using the “r” method. Because the file doesn’t necessary hold the data in list format, the information is converted into list using the split() method. It is then converted into dictionary using the key “Task” and value “Priority”. The rows are then appended into list called lstTable. I chose not to print the list since it is already one of the menu options (1). What is important to note is that this is the only time the file is read from the computer as throughout the whole program, the information is held in random access memory.

A screenshot of a cell phone

Description automatically generated

Figure 2: Step 1 of Assignment05\_Starter.py showing reading ToDoList.txt into memory.

Step 2 – Probably the simplest part of the assignment, step 2 is simply a list of menus the user gets to choose from. Nothing was altered from the starter file except I renamed few of the options so that it better fits the purpose of the program.

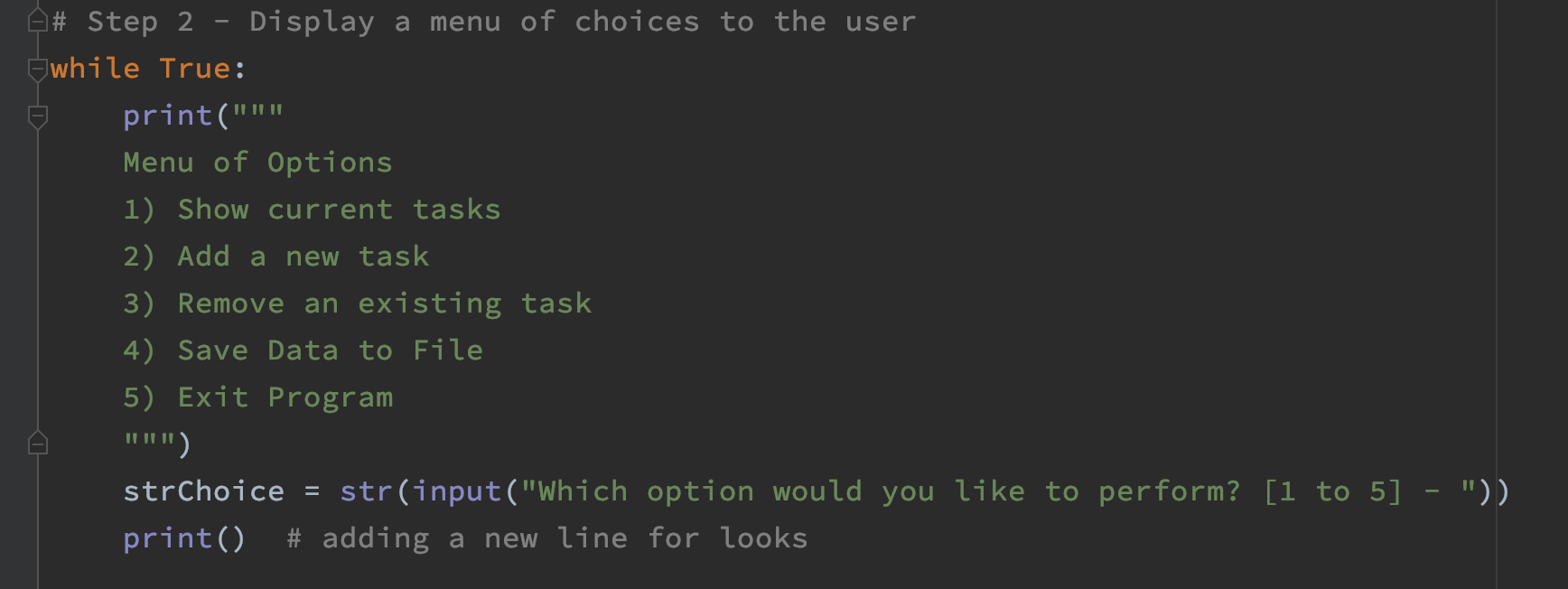


Figure 3: Step 2 of Assignment05\_Starter.py showing list of menu options.

Step 3 – In this step, the user gets to see their current list of tasks and their priorities. The list is in real time, meaning the user get to see the list after it has been modified. I added a print statement to relay to the user what the program will be printing to the screen. I also added tab escape characters to make the list more appealing.

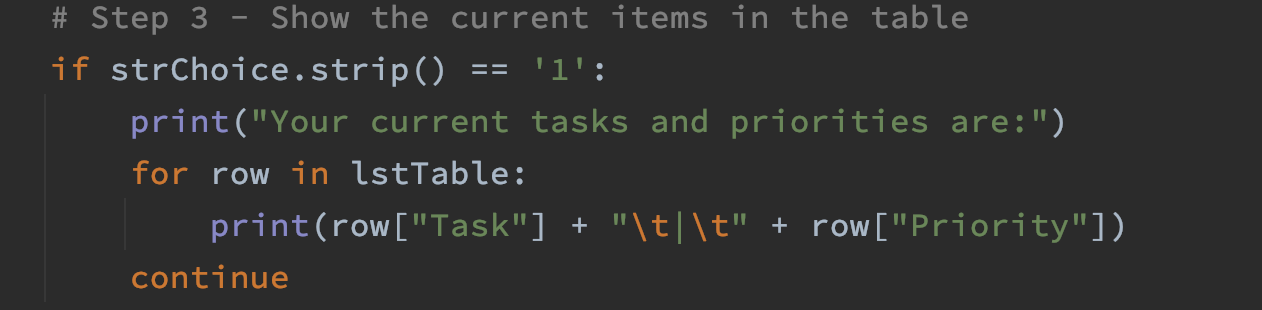
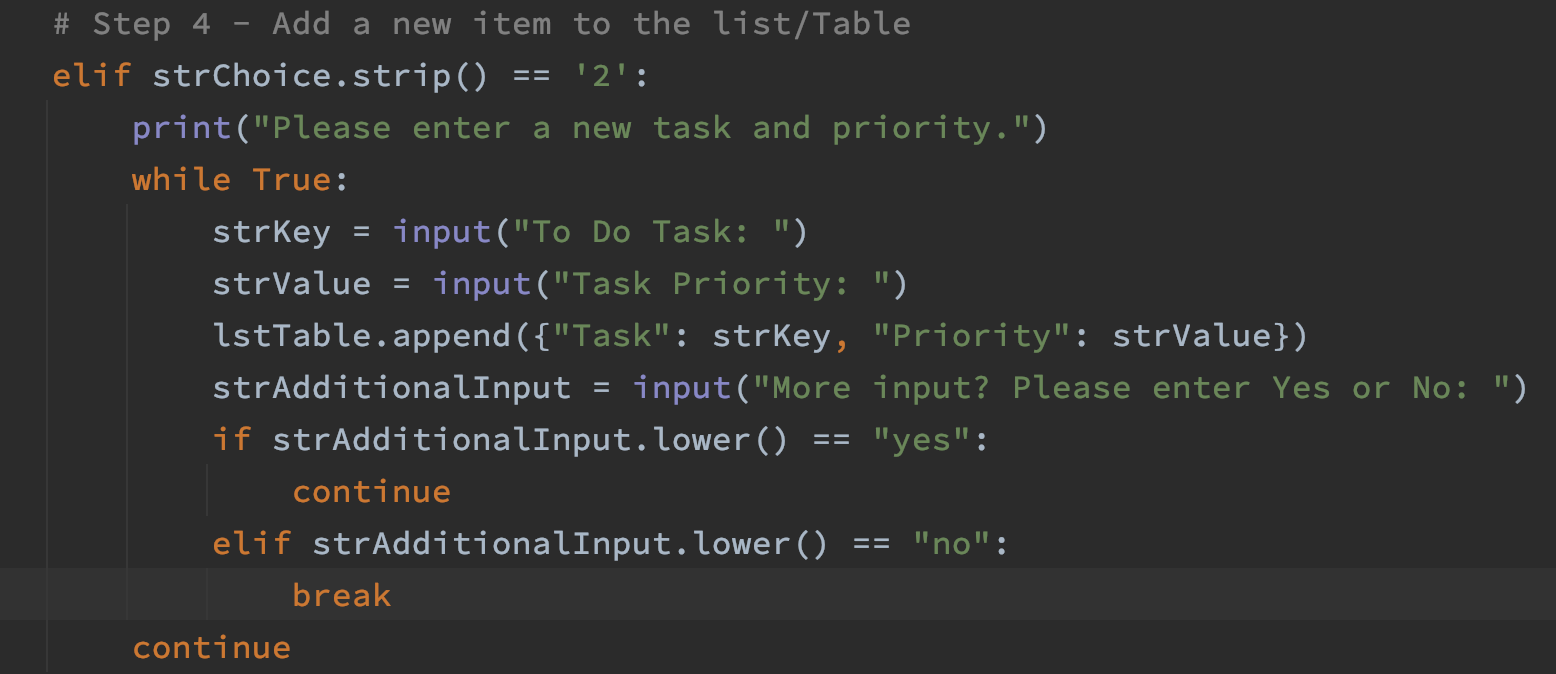


Figure 4: Step 3 of Assignment05\_Starter.py.

Step 4 – In step 4, the user gets to add to the current list of tasks. Inside a while True loop, there are two input statements for task and priorities. They are appended to the lstTable list as a dictionary. There is a user prompt for additional task and priority inputs. Additional if statement allows the user to enter more if they choose to. Lower() method is used here in case the user enters lower case “yes” or “no”.

  
Figure 5: Step 4 of Assignment05\_Starter.py.

Step 5 – The most difficult part of the assignment, in this step the user gets to delete a row from lstTable by entering the key or Task of the dictionary list. For loop is used so we can compare a row of the lstTable and compare if it is same as what the user entered. If the row’s Task is same, the entire row gets removed via the remove method. It should be noted that the entire row gets removed as we wouldn’t only want the task to be removed. But what if the user enters a task that is currently no in lstTable? I initially had hard time making this work as a for loop is being used to compare the input. For example, if I enter a task that is at the end of the list, it would still remove the item but only after going through the entire list. Because I am using print statements to communicate that either a task is removed or not valid, it would trigger both statements in this situation. To get around this I used a len method to get the number of rows in the lstTable. I used an un-indented if statement so that if would compare the length of the lstTable to the current number of tasks. The reasoning behind is that if there was no match to what the user entered and what is in the lstTable, the length of lstTable and length of current tasks would be the same. The program goes back to top of the menu regardless there is match or not.

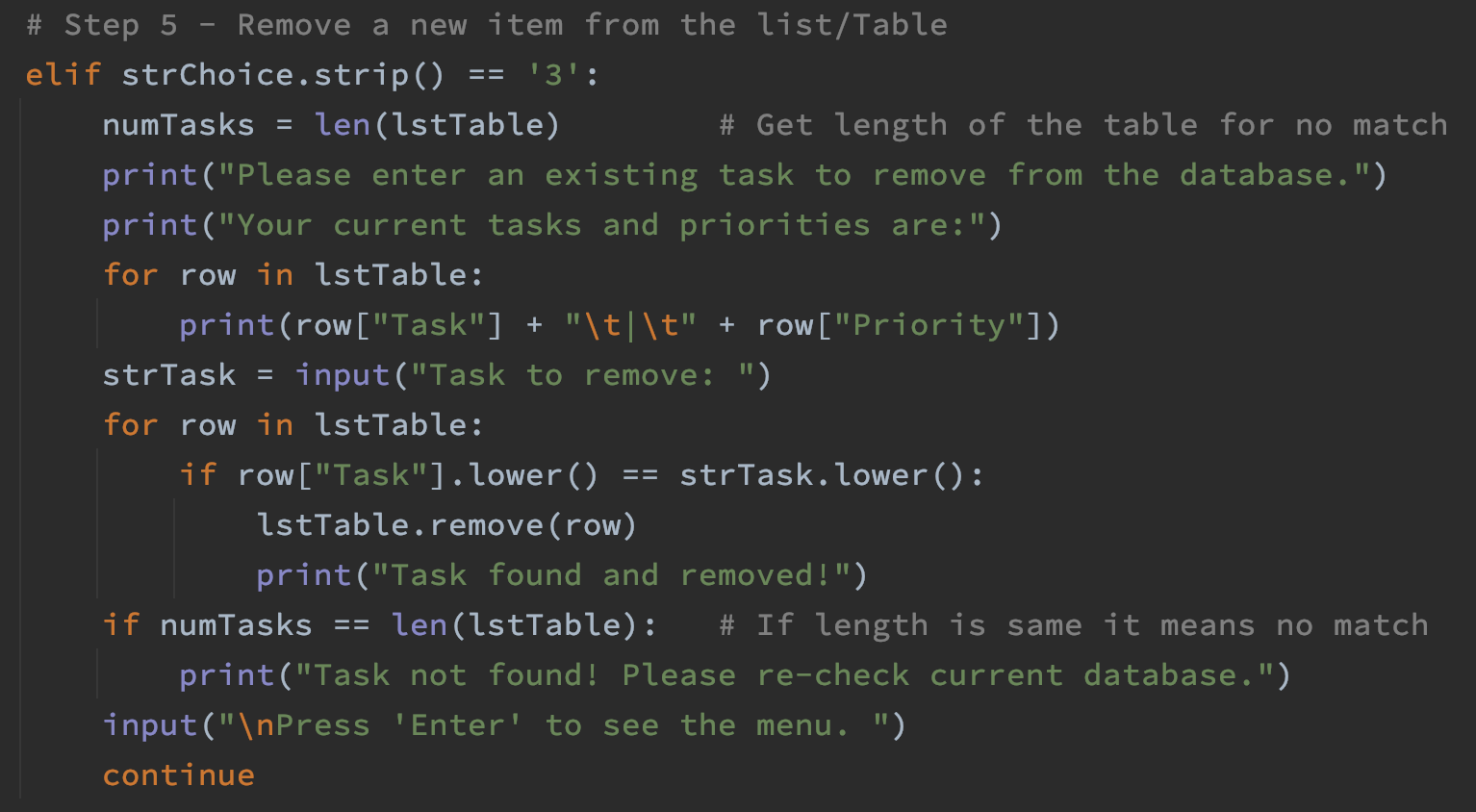


Figure 6: Step 5 of Assignment05\_Starter.py.

Step 6 – This is similar to module 4’s assignment. We are opening the TodoList.txt file using the “w” method. Each row of the lstTable is saved to the file using a for loop.

Step 7 – Exiting the program. Only a print statement was added to communicate to the user that the program is exiting.

else – I added a catchall else statement in case the user enters a menu choice other than 1, 2, 3, 4, or 5.

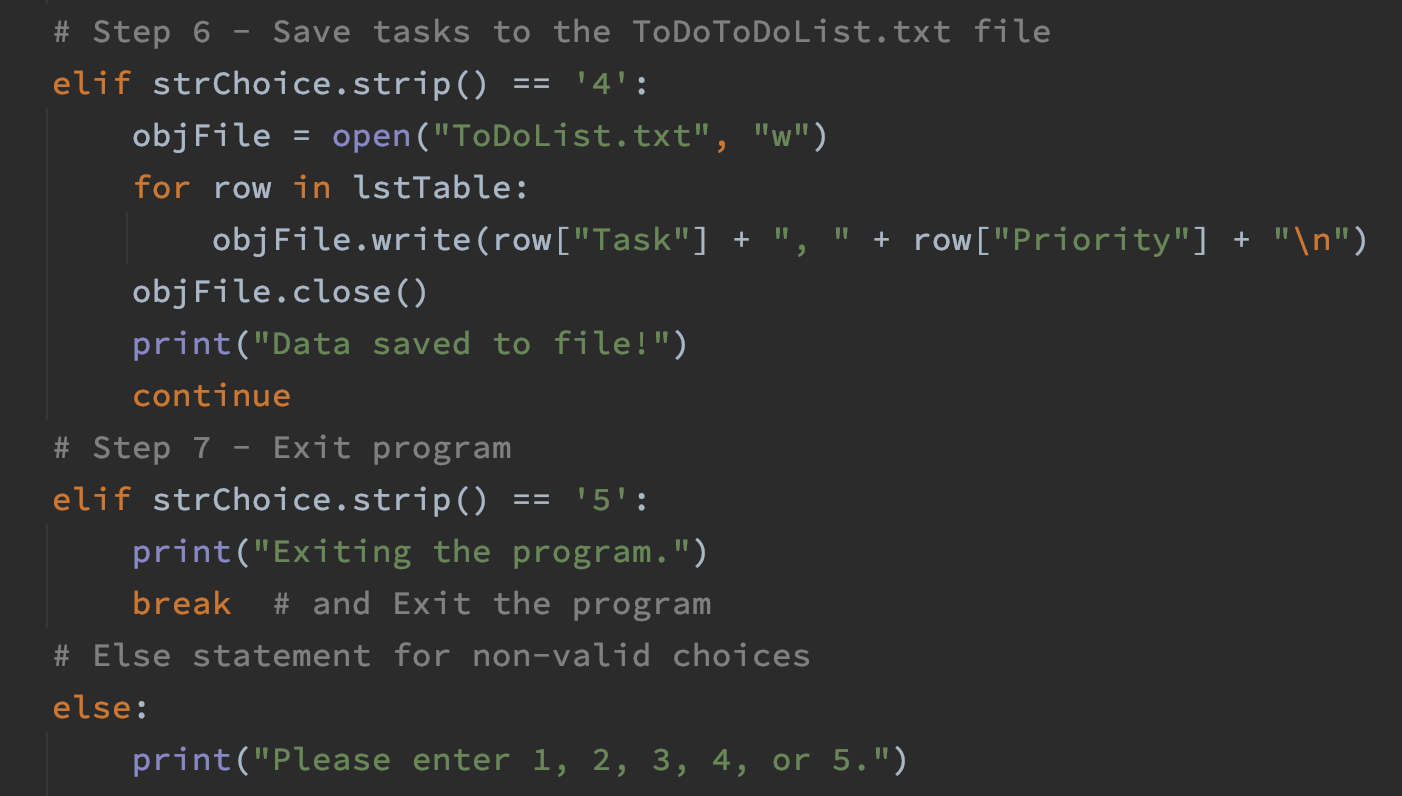


Figure 7: Step 6, 7, and else statement of Assignment05\_Starter.py.

Code Output

I am pleased the final program is working as I want and that I was able to add some personal touches. As expected, using the Mac OS command line to test the code created the ToDoList.txt in my user home directory folder as my current working Python directory is located there. Testing on PyCharm, however, the text file was created in the same folder as my .py file.

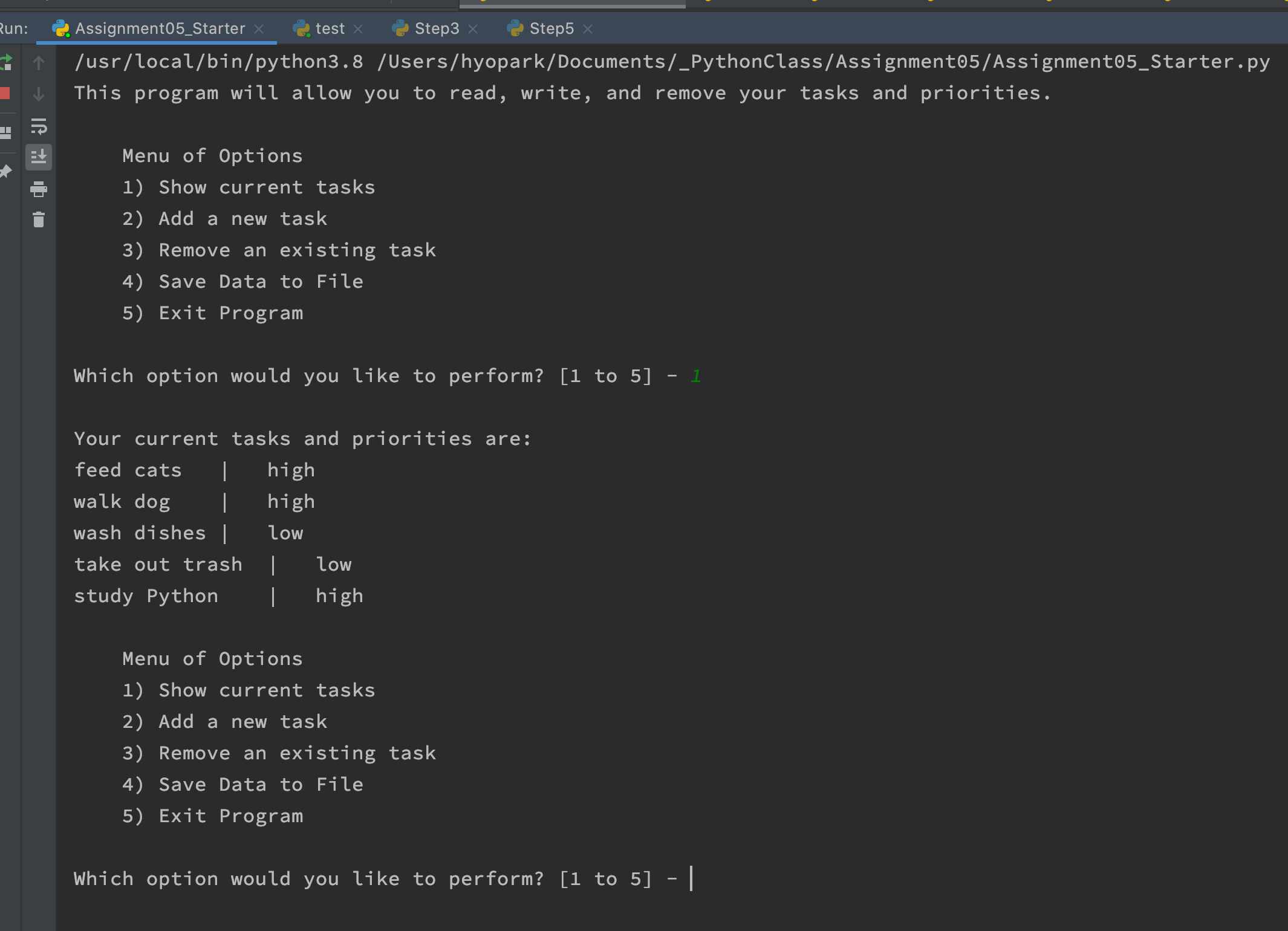


Figure 8: Assignment05\_Starter.py running on PyCharm.

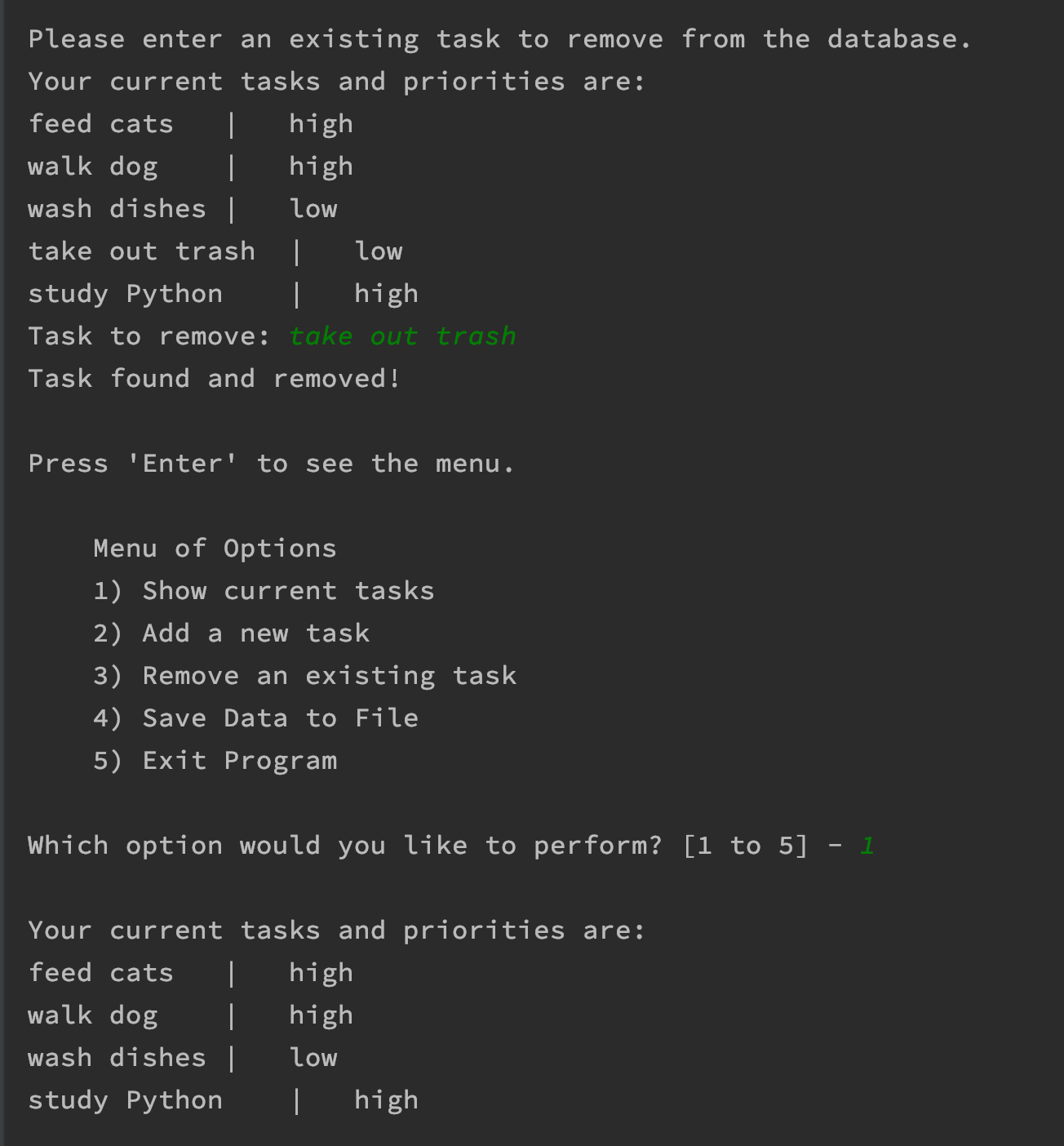


Figure 9: Assignment05\_Starter.py running on PyCharm.

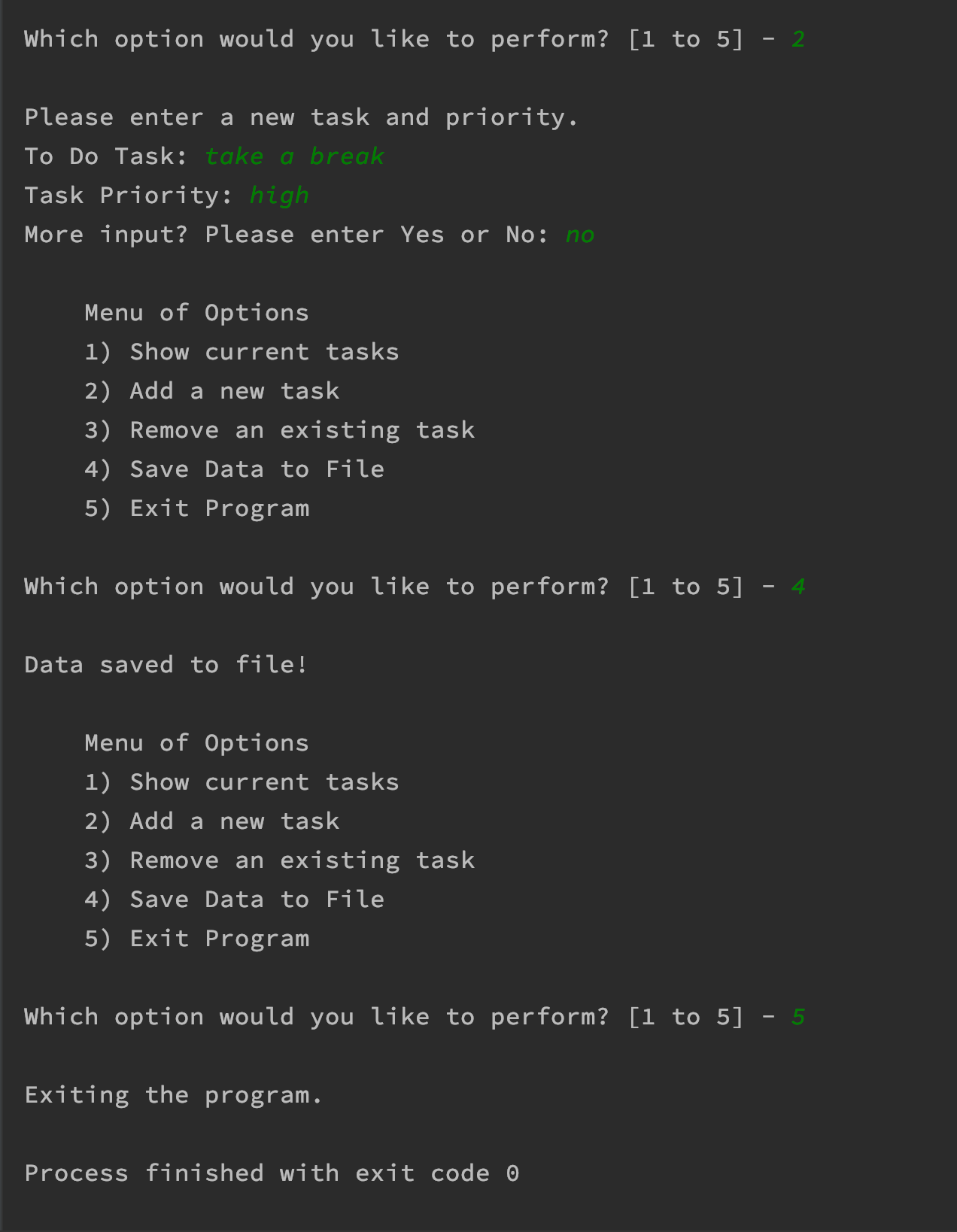


Figure 10: Assignment05\_Starter.py running on PyCharm.

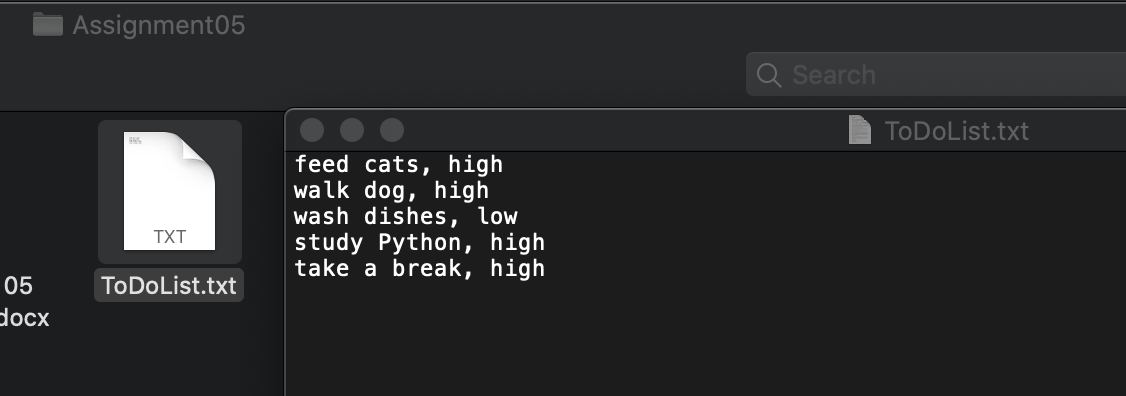


Figure 11: ToDoList.txt in directory with saved results.

# 

Figure 12: Assignment05\_Starter.py running on MacOS command line.

# Summary

To me, the fact that we were provided with a starter file made this assignment easier but more difficult at the same time. It guided me in the right direction right off the bat but felt constrained at the same time since the structure of the program was already provided. I imagine this is a common situation in real life where programmers working as a team. First and foremost, I am pleased the program worked. But I am also glad that although simple as it may be, the program resembles what we may use in real life situation. I am definitely looking forward to rest of course’s assignments.