Meredith Shapiro

August 10, 2022

IT FDN 110 B - Foundations of Programming: Python

Assignment 05

Working with Dictionaries and Files

**INTRODUCTION**

In this paper, I will explain how to write a script that continues to build on the last two exercises where we wrote data from a list to a file. In this exercise, we are working with a to-do list featuring tasks and their priorities, and we will be using dictionaries instead of just lists. Dictionaries are similar to lists, but they allow us to name the indexes (like a column header in a spreadsheet) rather than always using the numeric index.

This script access a text file with a to-do list and load data from it, then give the user a menu of options for what to do next. They will be able to print the existing data or enter additional tasks (similar to homework 4). They will also be able to delete data from the to-do list, which is new functionality. And of course, the user will be able to save and/or exit the program.

**WRITING THE SCRIPT**

**Declare Variables**

To write this program, we started with a skeleton program that listed the menu options we would be presenting the user, and had blanks to fill in code to perform those actions.

First, variables were declared. I used the variables presented in the starter code, and added some as well.

**Read Data**

The program begins by reading in the data from ToDoList.txt. I populated this in advance to see what would happen when I read the data I supplied.

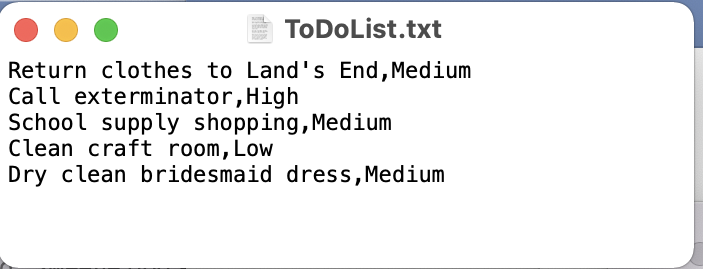


Figure 1. Contents of ToDoList.txt

As you can see in Figure 1., I started with a list of five items. To read these into the program, I used a for loop. For each row in the file, I split the row into two items using the comma. Then I turned the tow into a dictionary, and named my indexes for the row. I named them Task and Priority, and then appended each new dictionary row to the list table.

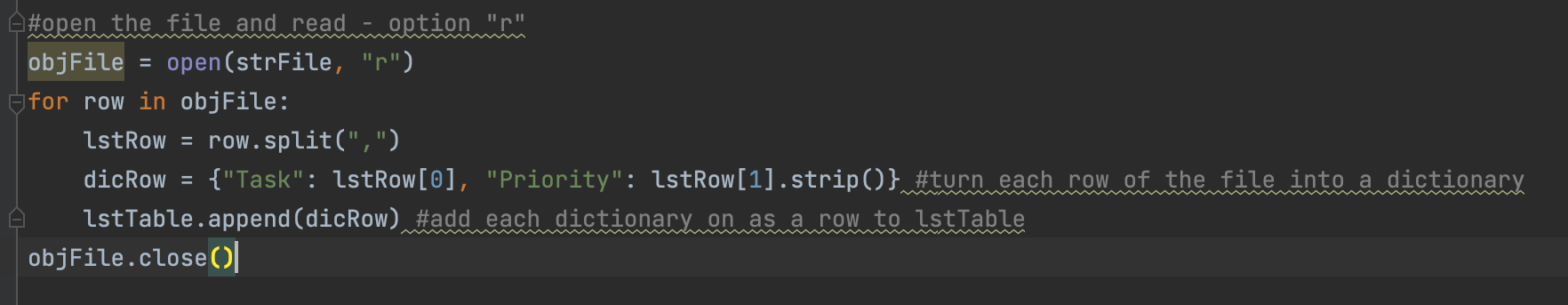


Figure 2. Loop to read in data

In Figure 2, you see that I created dictionaries for each row in the file. I called position 0 “Task” and position 1 “Priority.” I had to strip Priority, since this is reading a row from the text file there is a carriage return after each one and we don’t want that to be part of our dictionary.

**Loop Through Menu Options**

I won’t dwell on how we create the menu as this was discussed in Homework 4. This menu is very similar, but it has more options.

**Option 1 – Print data**

If the user chooses to print the data, we loop through each row of the table and print it. But this time our table is made of dictionary rows.

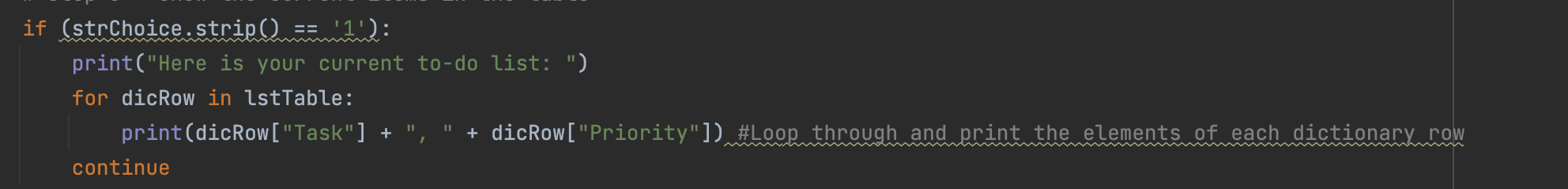


Figure 3. Loop to print data

As seen in Figure 3, this is similar to our print loop from homework 4, but we are pointing to dicRow[“Task”] and dicRow[“Priority”] instead of row[0] and row[1]. This is useful for us as programmers to be sure we are pointing to the right data – especially as we begin to work with wider datasets.

**Option 2 – Add a new to-do list item**

Option 2 allows the user to enter a new task and corresponding priority. This is another process that is familiar – getting user input for a task and a priority, storing those values, then putting them into a dictionary this time instead of a list, and appending that to our table.

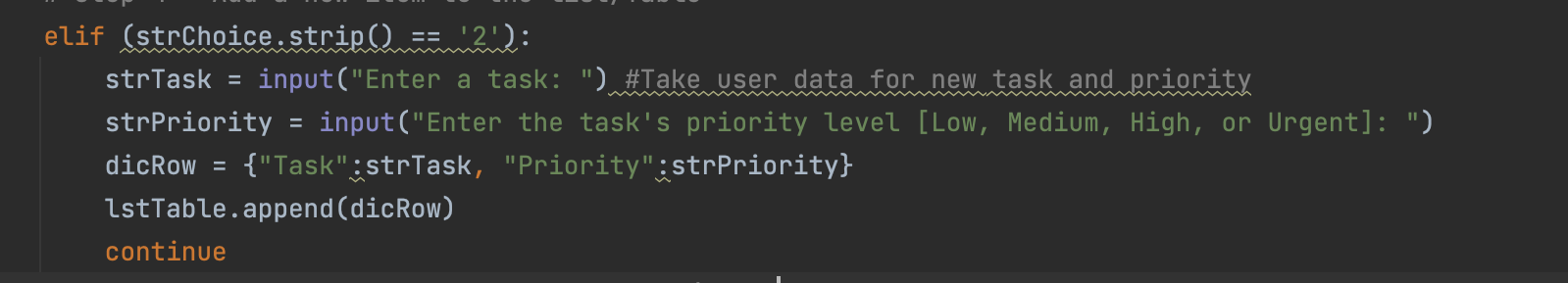


Figure 4. Code to take user input for task and priority, create new dictionary row

In Figure 4, you see again that we are using these index names Task and Priority to show where we are putting data.

**Option 3 – Remove a task from to-do list**

We would also like the user be able to remove a row from the to-do list – whether due to entering it in error, or because they completed the task and get to cross it off.

To accomplish this, we take user input for which task they want to remove (using the name of the task). This could be prone to user error, although since they can display their data at any time, hopefully that would help them identify the name of the task to delete.

When we have the task to delete value stored, we loop through all the rows and compare this against the first element of each row, and if they are equal to each other, we remove the whole row where we found that match.

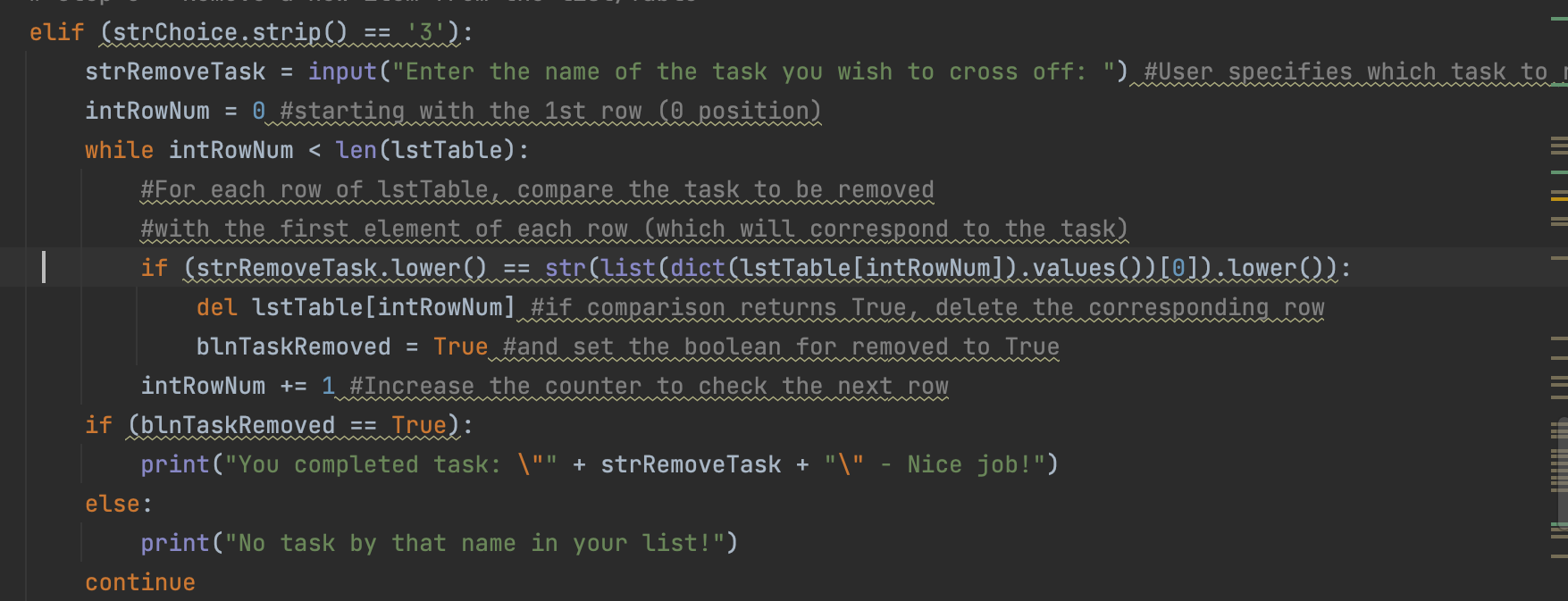


Figure 5. Loop to look for user-specified task to delete

In figure 5 you can see that I tried to add some code to handle errors – I convert the strRemoveTask and the compared value from the list to lower case, in case the user has the right name but inconsistent capitalization. If there is not a match, a message prints to say no task with that name was found, and the user could select option 3 again and try it a second time.

**Option 4 – Saving data**

In the previous homework I got this wrong – it is better, if the user choses this option, to verify that they want to save the data. If they have been writing practice data then they may not want to save it to their real to-do list. So, we start by taking user input (y or n) for whether they want to save. If they select y, we loop through each dictionary row and print the Task and Priority, followed by a carriage return, then close the file. Make sure to use the “w” option in open() – I initially thought we should use “a” for append, since we start with a text file of data don’t we want to only append the new rows we’ve added? But since we store ALL the data in a table of dictionary rows, including what was initially in the file, if we loop through that whole table and append it to what was already in the file, we will end up with duplicates for all the rows that weren’t added during the run of the program.

If the user doesn’t want to save after all, we let them hit enter to return to the menu.

**Option 5 – Exit**

If the user decides to exit, print a message to confirm exiting, and then break so the program ends.

**RUNNING THE SCRIPT**

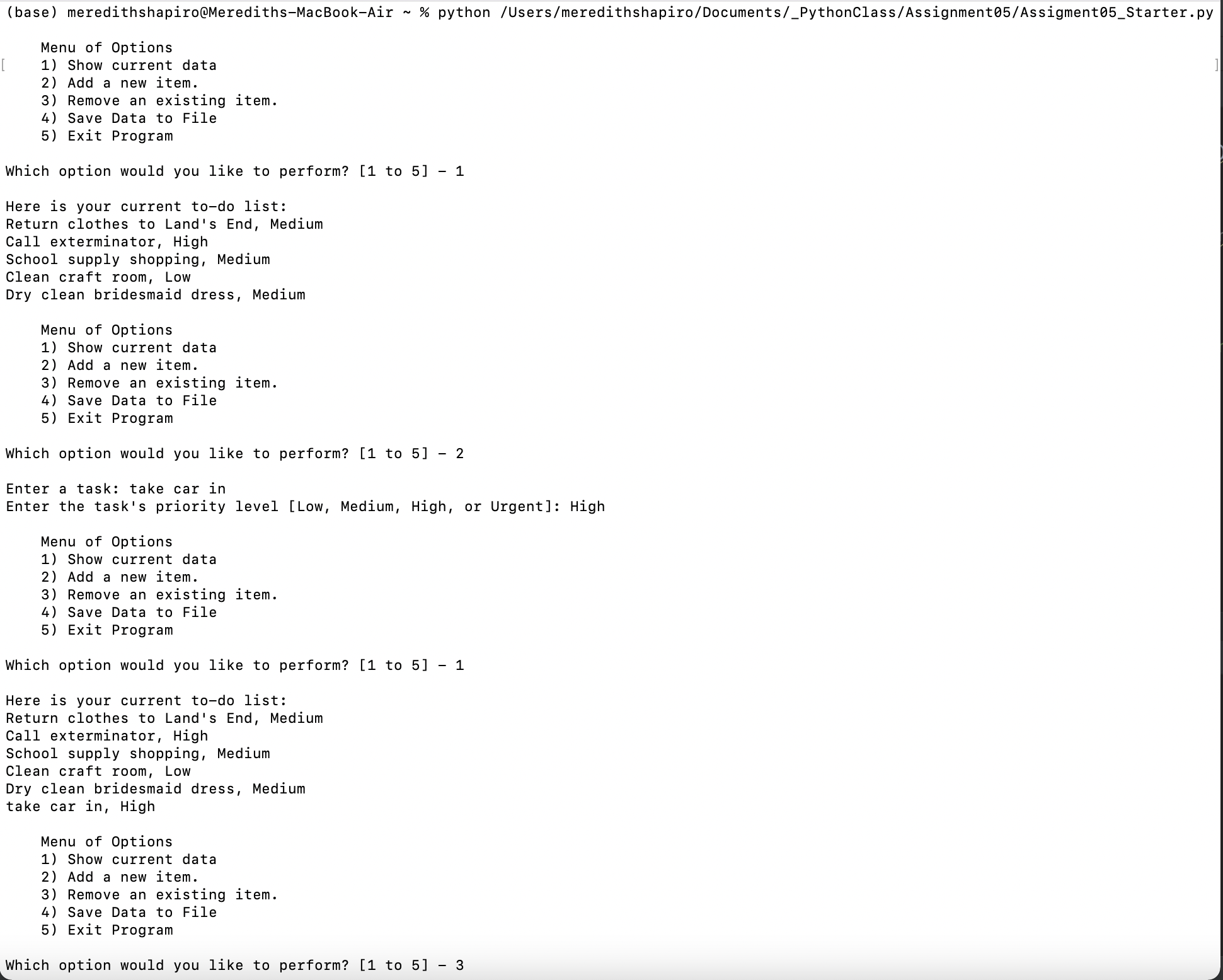
Here is how the program looks running in Terminal 

Figure 6. Program running in Terminal

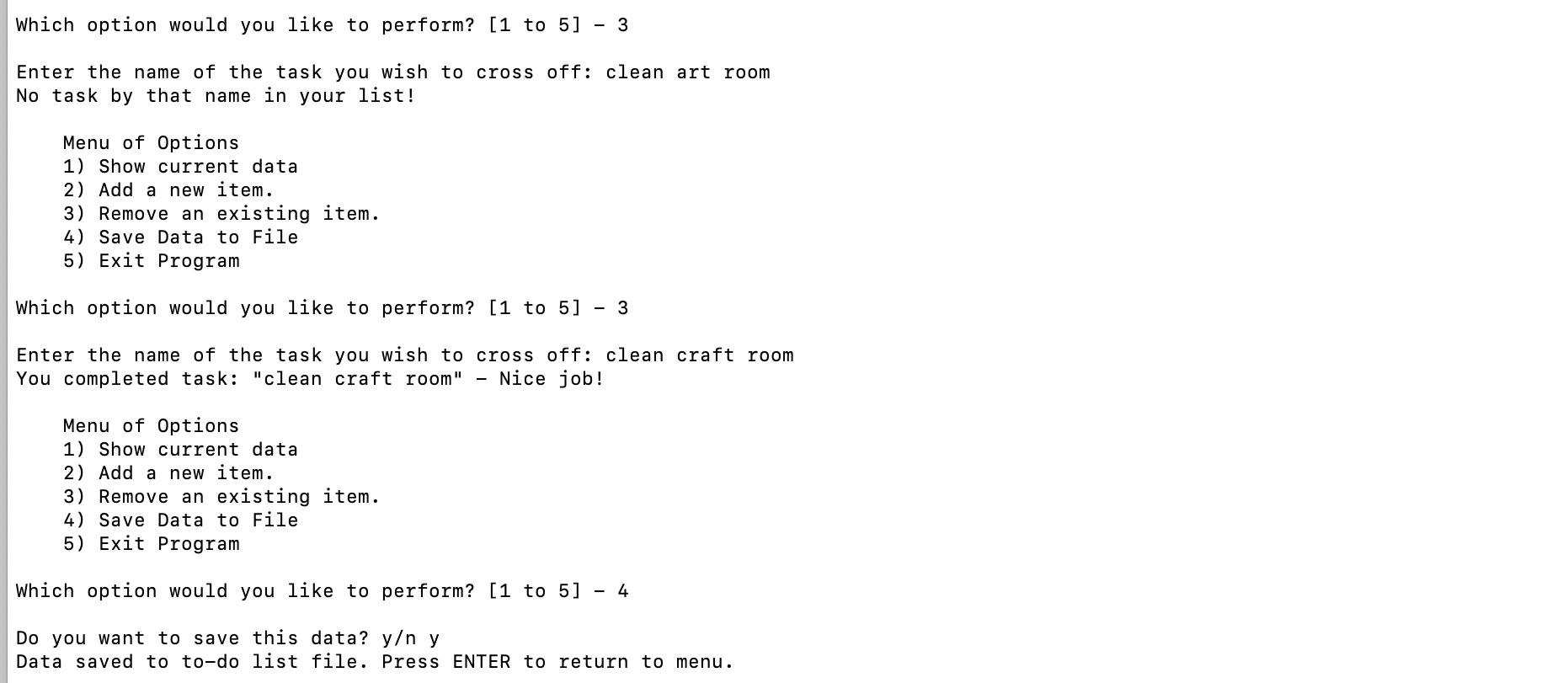


Figure 7. Continuation of Figure 6 – program running in Terminal

In Figures 6-7 you can see how the program works for selecting all the options. I experimented with entering something wrong (say I referred to my craft room as an “art room” instead) and got the error message. Next I successfully removed that task.

I then opened the text file to verify my changes:

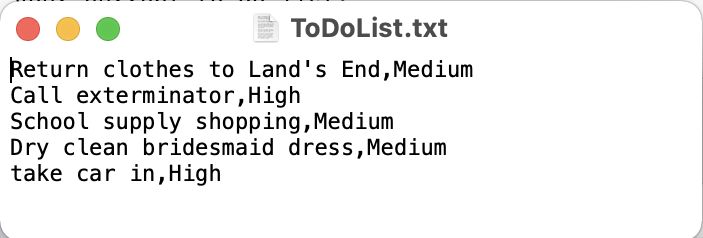
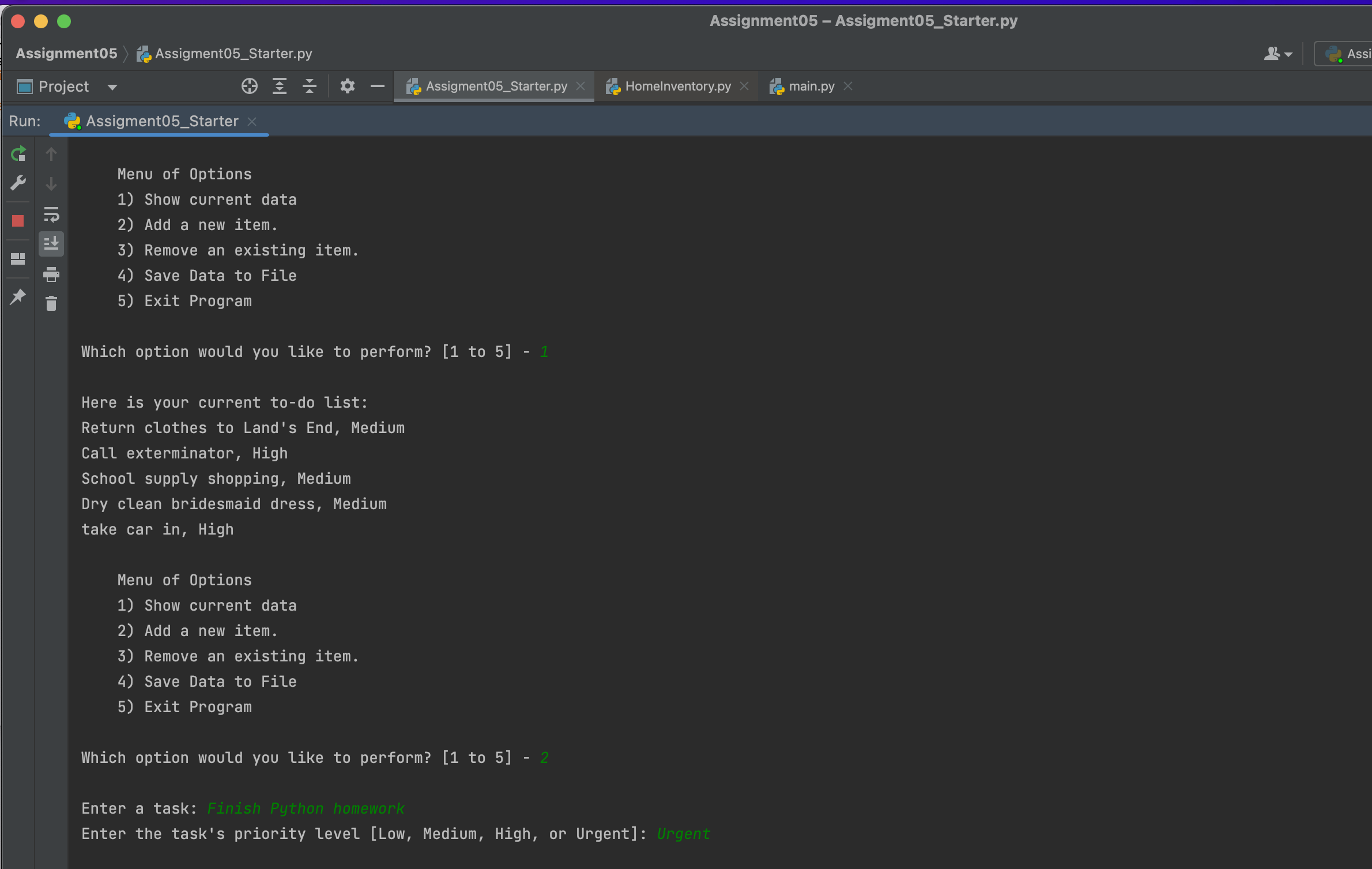


Figure 8. Updated text file

Figure 8 shows that my changes were successfully saved – I added a task for taking the car in, and removed clean craft room.

We can also run the program in Pycharm.



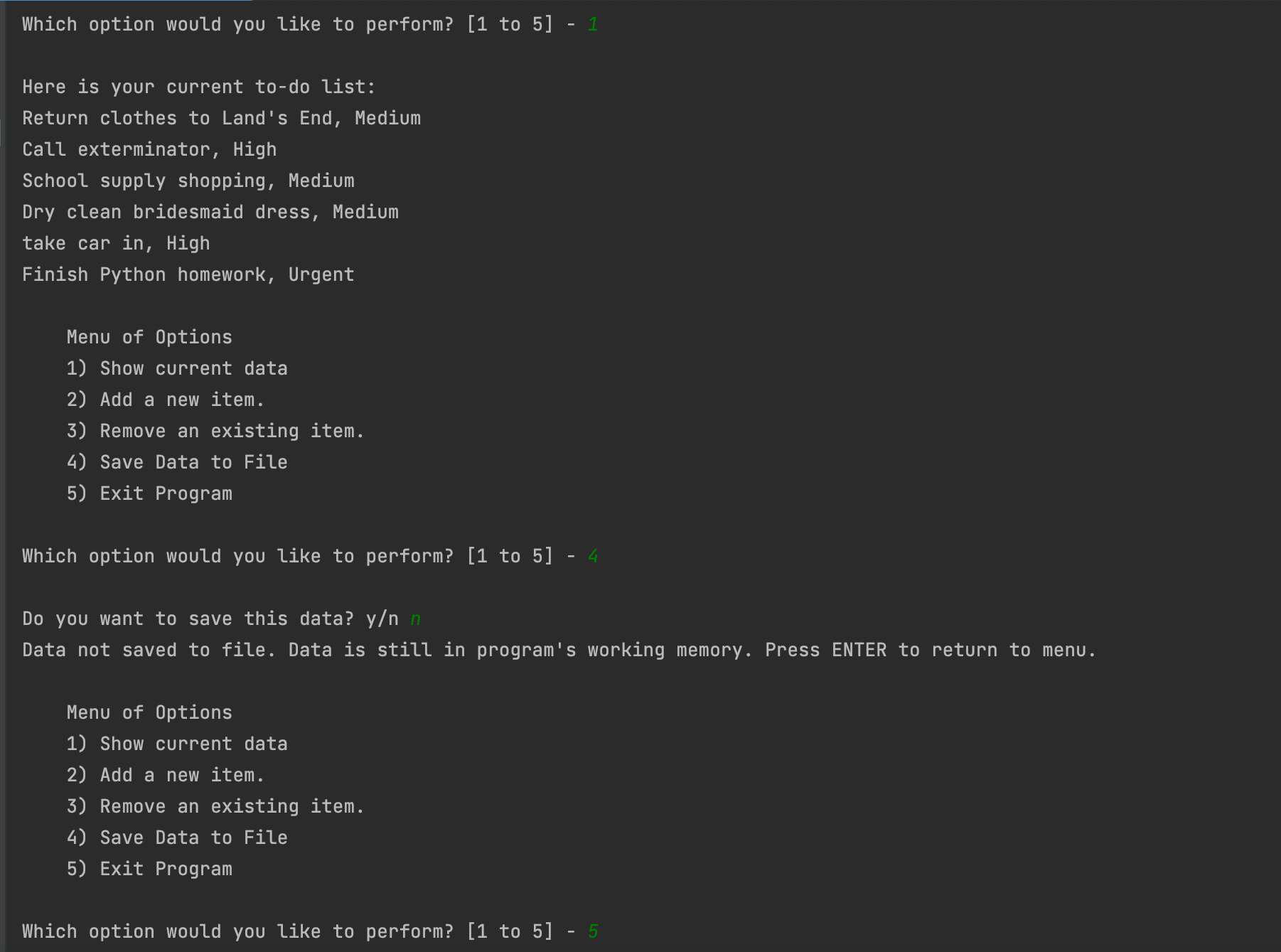


Figure 8. Program running in Pycharm

I added a task to the list called “Finish Python homework”, but did not save it to the text file.

**CONCLUSION**

From this assignment I learned how to read data into my program from a text file. I also learned how to use dictionary objects. In some ways, using dictionaries were more intuitive because the indexes are labeled. But in a way it made things more complicated, because it felt I was having to move between dictionaries and lists/convert between the two – especially for the step for deleting rows, which I didn’t understand at all until I watched the solution video.