How Code Is Structured

The main file coursework.py is run, welcomes the user then enters into a loop “while True:” which is only exited when the user chooses option 6 – Exit on the menu.

Opens shelf “word\_list.dat” which has already been supplemented by the files in Lists.py. It will store any additions or deletions made when last used.

Prints a list of options for the user, an if function is set up depending on which option the user enters.

Prints a list of options for the user, an if function is set up depending on which option the user enters.

If 1: Accesses, the shelf and the scores.dat file where it writes the score and time to using pickles.

If 2: I access my own module option\_2 in jumble.py which prints asks which list to print, then prints it.

If 3: lets user write to the shelf in the same style as in lists.py then updates shelf.

If 4: lets user delete item off the shelf.

If 5: Opens the scores.dat and imports a module which sorts the scores and presents it ordered to user.

If 6: then exit the programme and so doesn’t repeat the while loop.

A Record of Tests Performed

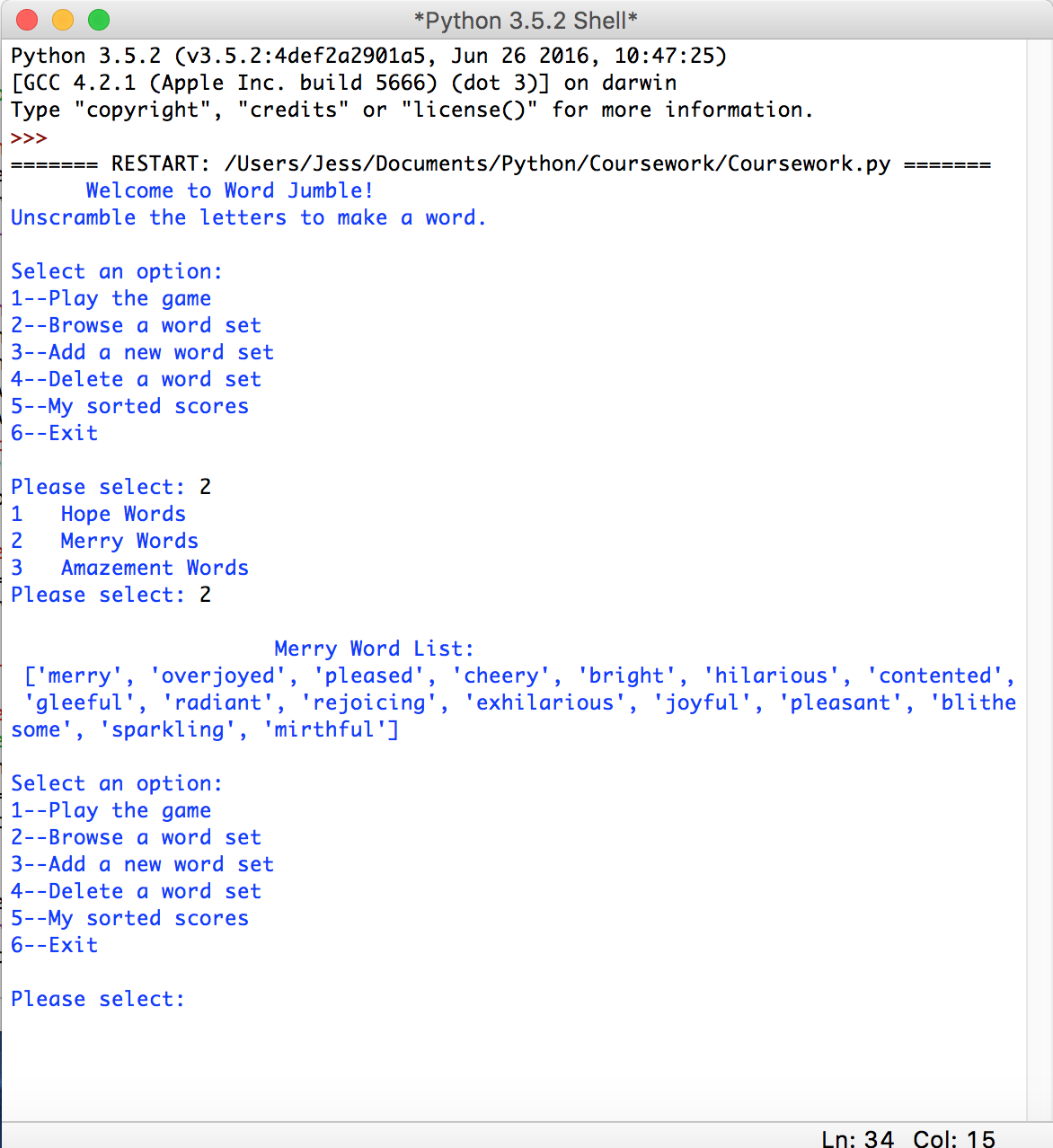


In the first test you can see 10 words being given to the user and then the user being scored.

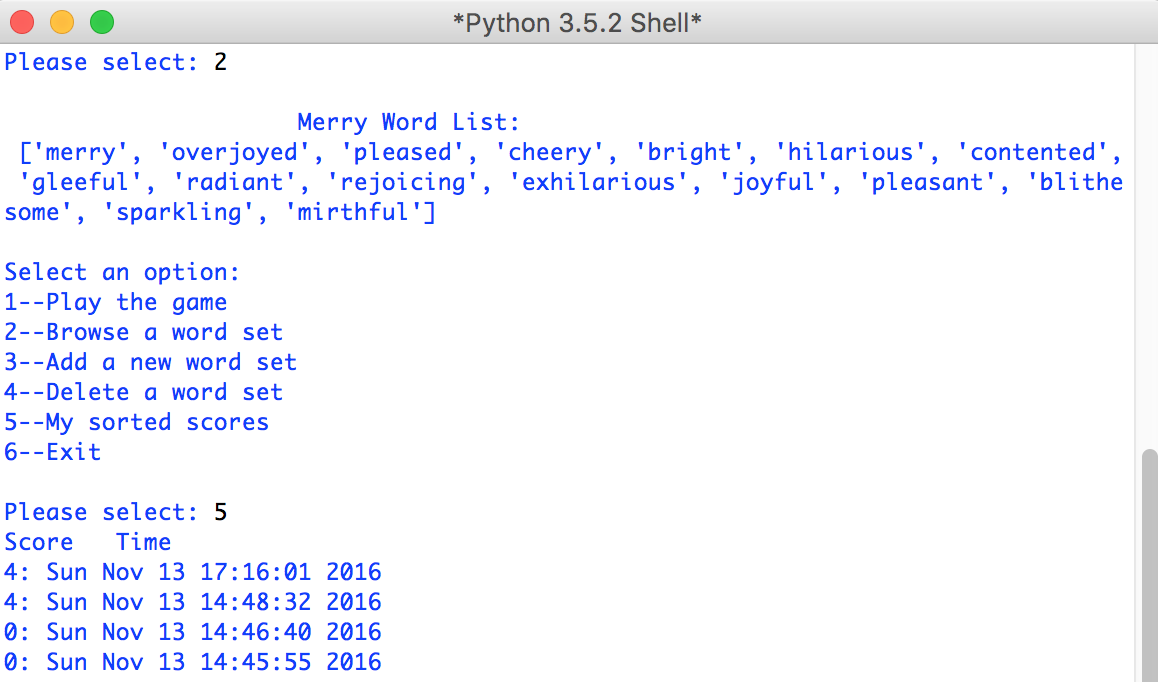


Following on from the last test the program gives a list of words request by the user.

Here we can see the the user adds the word set Merry Words and deletes Sad Words, then using the menu option to exit the program is closed with the changes to the word sets saved – which will be shown in the next images.



Here you can see that deleting Sad Words and adding Merry Words has both worked and are permanent changes.

The final functionality of the jumble game to prove is the ordered score list. As You can see the lists are ordered by how high the user has scored.

While writing the code I often used print(word\_dictionary) to see if changes had been implemented especially when adding and deleting words.

To check that the scores had saved I opened the scores.dat in a text editor file to make sure any additions had been written to there.

I have left the remains of my testing code commented out in some places so that you may see them – for example in lists.py I tested that all the word .txt files have been shelved by printing them before implementing within the main coursework.py. This allowed me to locate any mistakes when I was in the process of coding more quickly.

Conclusion on Correctness of Implementation

As the tests shown I have created a fully functioning jumble word game that fully satisfies all the requirements set out in the coursework brief. You can play the game, browse word sets, add new word sets by shelving and delete them from the shelf. All scores and time of the scores are recorded permanently and can be accessed by the menu, where they are presented in order of score size. The user is taken back to the main menu after each action is worked through and can exit the program through an item on the menu.

The methods to which I created the program also satisfy the criteria in the course work guide. I have read txt files and written to them. Defined variables and processed strings in different ways – such as jumbling them for the game. Words sets are presented to the user through running a module I created and imported from a different .py file. I have also use lots of other native modules in python like random and shelve. I have sorted the player scores from highest score to lowest score.