Python Lab 3B -

Q - In what way did you "clean up" or divide up the text into words (in the program; the text files should be left unaffected)? This does not have to be perfect in any sense, but it should at least avoid counting "lord", "Lord" and "lord." as different words.

A – I read the text file into a variable and converted it into lower case. I am then iterating over each character and I keep concatenating characters into a word until a non-alphabetic character occurs. So, the words I get just keep the alphabetic part and eliminate all other characters.

Q - Which data structures have you used (such as lists, tuples, dictionaries, sets, ...)? Why does that choice make sense? You do not have to do any extensive research on the topics, or try to find exotic modern data structures, but you should reflect on which of the standard data types (or variants thereof) make sense. If you have tried some other solution and updated your code later on, feel free to discuss the effects!

A – I have used Dictionaries. I have used a dictionary for storing the character frequency and unique words. The dictionary containing the unique words has the words that occur next to them in the text and the number of times they occur next to the word in a dictionary. So, it’s a dictionary within a dictionary. I chose a dictionary because this gives me access to the corresponding value for a word or a letter in approximate constant time (key, value pair). And for a large file this would make a huge difference in the access time while counting the unique words or occurrences of each letter.

I used a list of tuples while sorting for getting the alphabets in order according to the count in the text and the top 5 most popular words. I just converted the dictionary to a list of tuples while sorting using dict\_name.items().