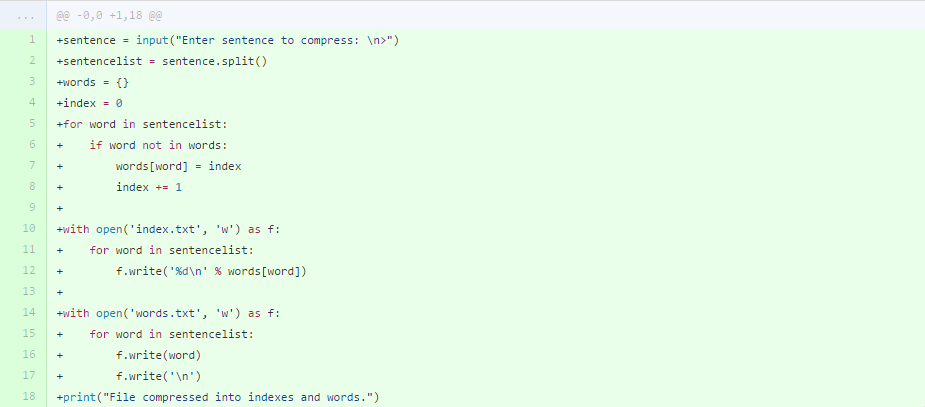


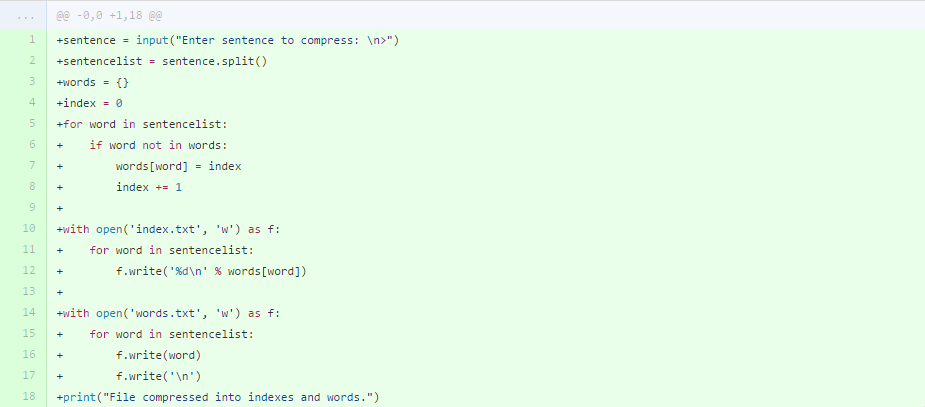
For task 2 there are clear stages for the code. The first of which you can see above, is to input the sentence and then store each word in a separate list. This was easily done as previously done.

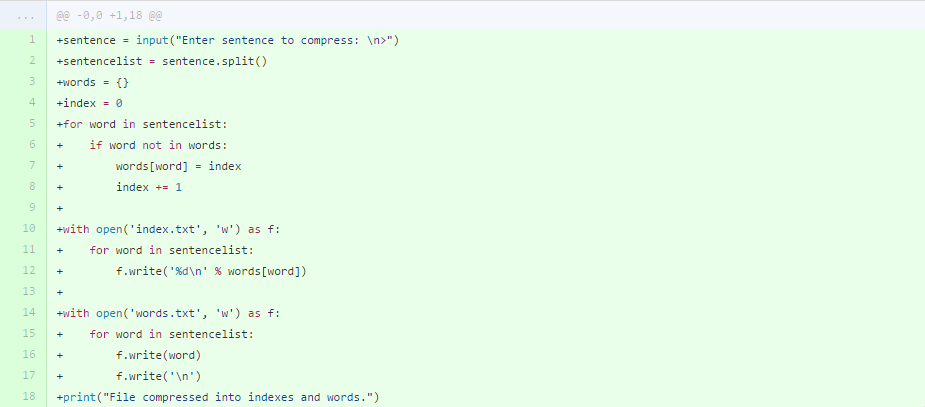


The second job to do was to take every word from the sentencelist and put them into another list called words but each word would have the value of its index. Now, I know that probably doesn’t really make sense so here’s an example: Let’s say that sentencelist has the words (‘hello’, ‘my’, ‘name’, ‘is’, ‘cameron’,’name’). From here, the word ‘hello’ will be added to the words list but along with its index which is 0, this would look like {‘hello’: 0}. So, my final words list would look like this:

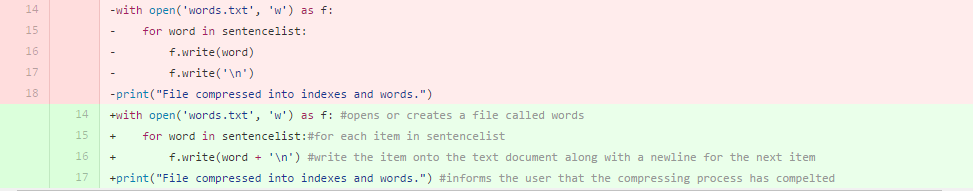
{'hello': 0, 'my': 1 'name': 2, 'is': 3, 'cameron': 4}

For what purpose is this for you ask? It’s so later on, I can write these indexes to a file in the correct order. However I must confess, I think that it’s actually pointless. I’ll tell you why later on as it will make more sense. Now onto outputting information into files. Two files need to be made, an index file which contains the indexes of the words and a words list which, obviously, contains the words from the sentence. Writing the indexes to a text document was simple. First create and then open index.txt. Second, using a simple for loop, for each word in the sentence list write its index value (which is found in the words list) into the text file. You can see that for each thing written, a new line is written with it so each index has its own line.

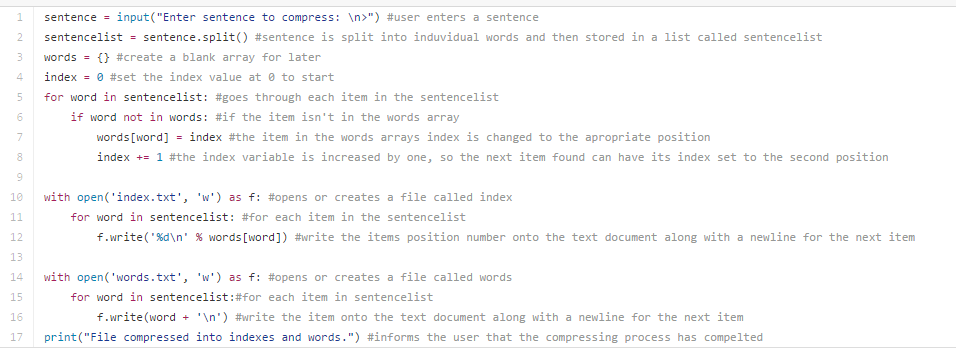


As for writing the words to another text document, it was really easy. Again, create and open a words.txt file and, with the help of a really simple for loop, take each word from the sentence list and write it to the text document. Again, each word has its own line so it can easily be recognised as an individual word later on. Then of course a nice message to say all the hard work has been done. Oh and in case you’re wondering, the text documents are stored in the same directory as the program was run, so I would recommend running the program in its own folder.

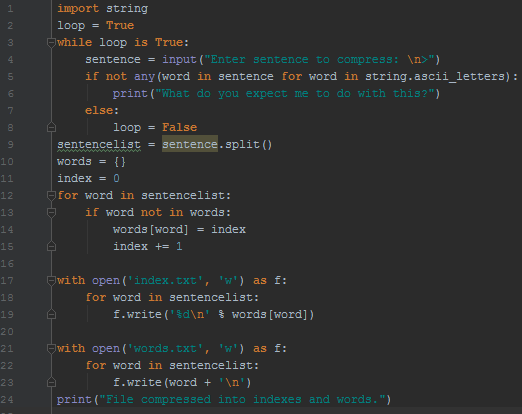
Later on I noticed two lines which could be reduced into one through string formatting. I simply made it so it writes the new line and the word at the same time instead of separately, hopefully making the program run faster.



And here ~~is~~ was the final version of task two!



Update on task 2:

After making some changes near the input section of the code on task 1 I decided to do that same for task 2. Simply adding a while statement to the input with a condition that must be met in order to continue on with the program. That conditions is the same as the one in task 1 which basically checks the variable that you input to see if it contains any letter from the alphabet which is a more efficient way of checking to see if they actually wrote something relevant than just checking if the variable is empty as the user could just type a space which would be recognised as a character and the program would continue giving logical errors. So here’s the new version of task 2: