# Lab Test 2

Data structures used: string, list, dictionary and set. Note: variable names and function names are coloured. My code consists of the following functions:

## main

First of all, main calls read\_files (explained later?), which returns the word dictionary(word\_dict) and documents dictionary (doc\_dict). A menu is created through the input method, this prompts the user to input 1, 2 or 3 corresponding to the menu. Using an if statement the method executes the appropriate follow up steps:

* Input is equal to 1: function calls search\_docs, with word\_dict as parameter
* Input is equal to 2: prompt user through input method for a document number, then the display\_doc function is called, with the input and doc\_dict as parameters
* Input is equal to 3: exit() the program

The code for the menu is in a while True loop and tried(?) for ValueError and KeyError. This is done to have the user be prompted indefinitely until they quit the program and to make sure no incorrect inputs slip through the cracks (?). If these exceptions are raised the program prints: Invalid input, please try again:. Since a while True loop is used, the user will automatically be prompted again after.

## read\_files

After opening and reading the file, list of the documents (docs\_list) was created using the .split method. After which, the elements in docs\_list were stripped of leading and trailing spaces. Then, doc\_dict (key: document number, value: document text) was created by iterating through docs\_list and assigning the elements as values and the index+1 as keys.

The next step was to create a set of all words present in the documents (word\_set), a set was used to make sure no duplicates were present. This was done by iterating through docs\_list, for each it was stripped of spaces and punctuation, transformed into lowercase and split into a list of words (words\_list). To add each word to the word\_set, iterate through words\_list and use .add method.

To create the word\_dict (key: word, value: document numbers the word is present in), a list of stripped of spaces/punctuation, lowercase documents is made using list comprehension (stripped\_doc). This is done to make sure the document search is not case-sensitive or influences by punctation and spaces. Using a nested for loop, iterate through the stripped\_doc and word\_set (respectively?). If the word in word\_set is present in the doc of stripped\_doc, the word needs to be added to word\_dict. Initially, I had the word\_dict values as a string and not a set. I had issues changing it from string to set, assigning a set as a dictionary value and making sure all the correct document numbers were present. Eventually, I figured out to use assignment (=) if the word wasn’t present in word\_dict and the .add method if the word was present in word\_dict.

## search\_docs

the code in this function is tried (?) for TypeError and ValueError (when result is empty?), if these exceptions are raised ‘No relevant documents are found’ is printed. The user is prompted for search words. These search words are stripped of spaces/punctuation and made lowercase, after which they are split into a list (search\_words). Using list comprehension, a list of the document number sets for the search\_words (doc\_sets\_list) is created. The relevant documents list (result) is the intersection of the doc\_sets\_list. To print out result, result is sorted and the elements are converted to strings. Using the ‘ ’.join(), a string of result (res) is created. ‘Documents fitting search’ and res are printed.

## display\_doc

This function takes the document number(doc\_number) and doc\_dict and prints out the specific document. Document # doc\_number, a separation line, document text and another separation line is printed using print() with .format and sep=’\n’. .format() was used to print the corresponding document number and sep was used to make sure there was a linebreak(?) after document number, separation lines and document text.