1. 執行環境

Visual Studio Code

1. 程式語言

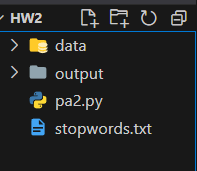
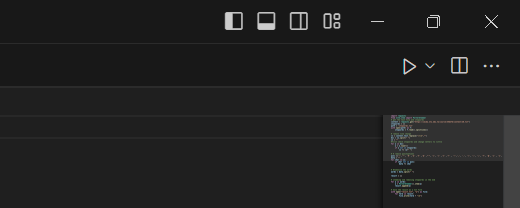
Python 3.10.6

1. 執行方式

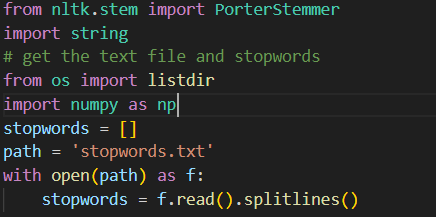
確認python已經安裝並可由vscode執行

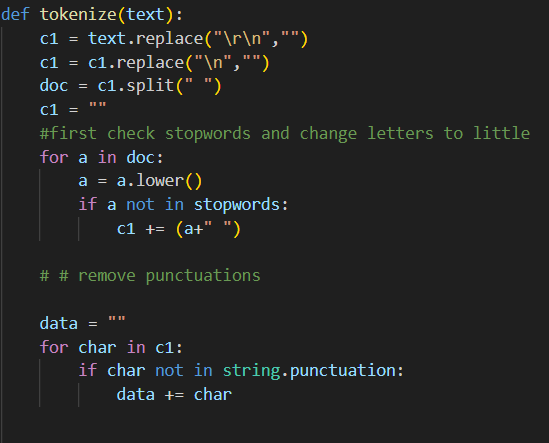
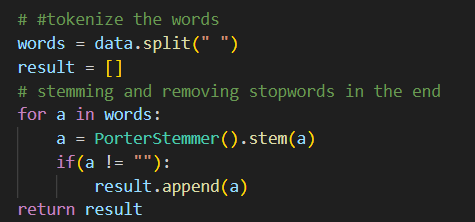
在cmd中使用pip install numpy, pip install nltk來安裝套件，額外import的套件為os, math, string(應為python原生，不須額外install)

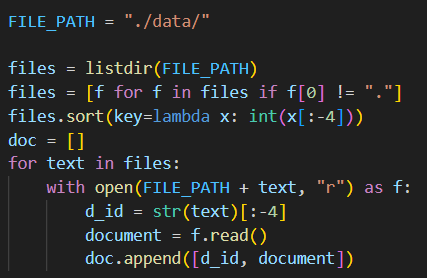
確認data資料夾(存解壓縮後的文字檔)，output資料夾(開始前應為空)，stopwords.txt與pa2.py在同個資料夾中，在vscode打開該資料夾點擊run即可執行(必須打開整個資料夾避免找不到stopwords.txt)

  執行完應會發現原本為空的output資料夾中有vector file和dictionary，點開即可看見結果

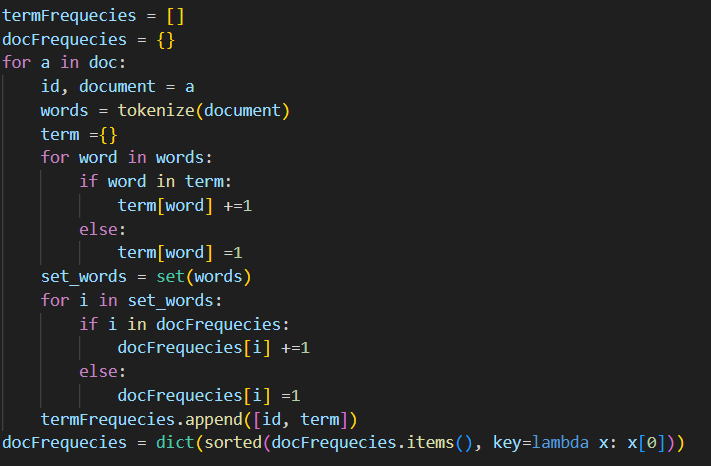
1. 邏輯說明
2. Package import: import string, listdir, numpy and PorterStemmer from nltk.stem, and read stopwords.txt to construct stopwords.



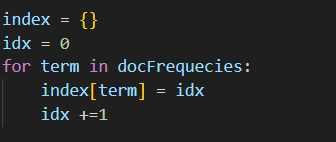
1. Function of tokenize(): Based on pa1, I now define the function tokenize, which will transfer the input text, which is unprocessed to a list of tokens.  
2. Read data: use listdir() to read documents in data file, and remove the ‘.txt’ file name, put the mapping of the document number and document text in a list.



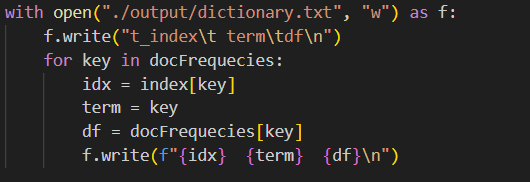
1. Compute term frequencies and doc frequencies for each term. For every document, pass its text into the tokenize() function, I will get a list of words. For term frequencies, when we see a term in the lists, I push it into its corresponding dictionary, and count it and store the result in a list. For document frequencies, I use set() method to make the list unique then compute each word occurs in how many documents.



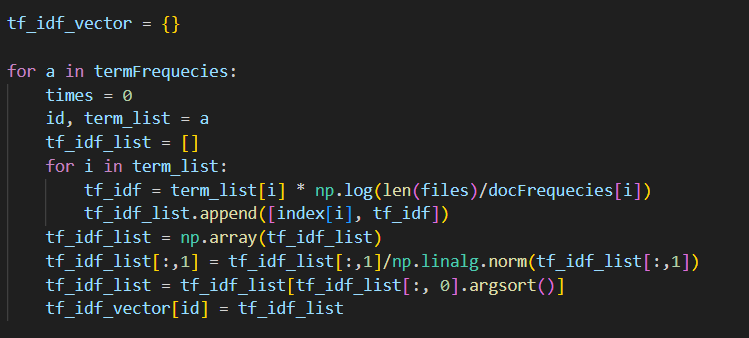
1. Give each word an index: construct a 1 to 1 mapping of term and its index



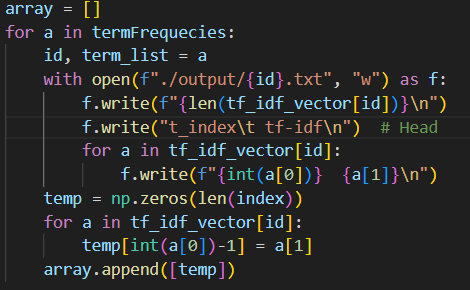
1. Write each word’s index, term and document frequency in the dictionary.txt file.



1. Compute tf-idf: I now can compute each document’s tf-idf based on the term frequency and document frequency. First, we access every term’s term frequency in each document, multiply it with the term’s idf value, and store the result in tf\_idf\_list, then change the list to a unit vector and rearrange it based on the term’s index. Store the list in a big dictionary.



1. Put the result for every document in the output file, and since what I do in g only have the term which is not zero in the document. To compute cosine similarity, the term which does not occur in the document must be added and set to zero in tf-idf vector.



1. Compute cosine similarity: since now I have each document’s tf-idf vector, I can simply compute the similarity based on the product of tf-idf vector of docA and tf-idf vector of docB. I have also computed the similarity of 1.txt and 2.txt as an example.

