標註執行環境：Visual Studio Code

程式語言：Python 3.11

執行方式：

1. 準備文件：將待處理的txt檔案放入名為data的資料夾。
2. 安裝 NLTK：打開命令行輸入pip install nltk (用於import porter stemming)。
3. 執行程式：在命令行中輸入python pa2.py。
4. 查看輸出：詞典結果將會儲存到 dictionary.txt 文件中。每個文件的tf-idf vector 結果將會儲存到 output 資料夾中。

作業處理邏輯說明：

1. 讀取文件：使用 os.listdir() 函數讀取 data 資料夾中的所有 .txt 檔案，並透過內建的 open() 函數依次打開每個文件。
2. Tokenization：對txt檔進行 Tokenization，包括split、lowercaing、stopword removal、punctuation removal、stemming。同時將token儲存於vocabulary set作為dictionary。
3. 計算每個文件中的詞頻 (TF)：
    - 使用一個字典 term_frequency 儲存每個詞的詞頻（TF），鍵為詞，值為該詞在文件中的出現次數。
    - 每個文件的 term_frequency 字典都會被 append document_tokens 中。這個列表將包含所有文件的term和tf資訊。
4. 計算文件頻率 (DF)：
    - 透過遍歷 vocabulary 集合中的每個詞，並檢查該詞是否存在於每個文件的 term_frequency 中，如果存在則將該詞的 df 值加 1。
    - 將結果儲存於df_dict 字典。
5. 輸出詞典 (dictionary)：透過迴圈將詞彙的索引、詞彙本身和文件頻率 (DF) 存入名為 dictionary 的列表中，並輸出結果至 dictionary.txt 文件。
6. 將文件轉換為 TF-IDF 向量：
    - 遍歷所有 document_tokens 和 dictionary，若term存在於 document_tokens，則根據公式計算該詞的 TF-IDF 值。(TF-IDF=TF×log(N/DF)
    - 將每個詞的索引與計算出的 TF-IDF 值組成一個向量，儲存於 matrix 列表中。

    - 每個文件的 matrix 最後會 append 到 document_matrices 字典中，這個字典將包含所有文件的 TF-IDF 向量。

7. 輸出文件的 TF-IDF 向量：將每個文件的 TF-IDF 向量輸出至 output 資料夾下的對應 .txt 文件中，每個文件會有對應的 matrix.txt 檔案。

8. 計算 Cosine Similarity：使用 cosine_similarity 函數計算兩個文件的 TF-IDF 向量之間的餘弦相似度，公式如下：

    - Cosine Similarity=dot product of two vectors / (magnitude of vector1 *magnitude of vector2)

    - 函數先計算內積，再計算兩個向量的長度，最後使用公式得到相似度值。