資訊檢索與文字探勘導論：作業二 Report

R06725027 資管碩一 林均泰

1. 執行環境：此次作業是以**Pycharm**作為IDE來開發
2. 程式語言：**Python 3.6.0**
3. 執行方式：將上傳至作業區的zip檔解壓縮後，會有一個檔名為PA-2的資料夾，而PA-2的資料夾中總共有4份python file，分別是**porter\_stemmer.py**、**dictionary.py**、**tfidf.py**和**vector\_space.py**。

須先透過pip安裝**pandas**以及**numpy**。

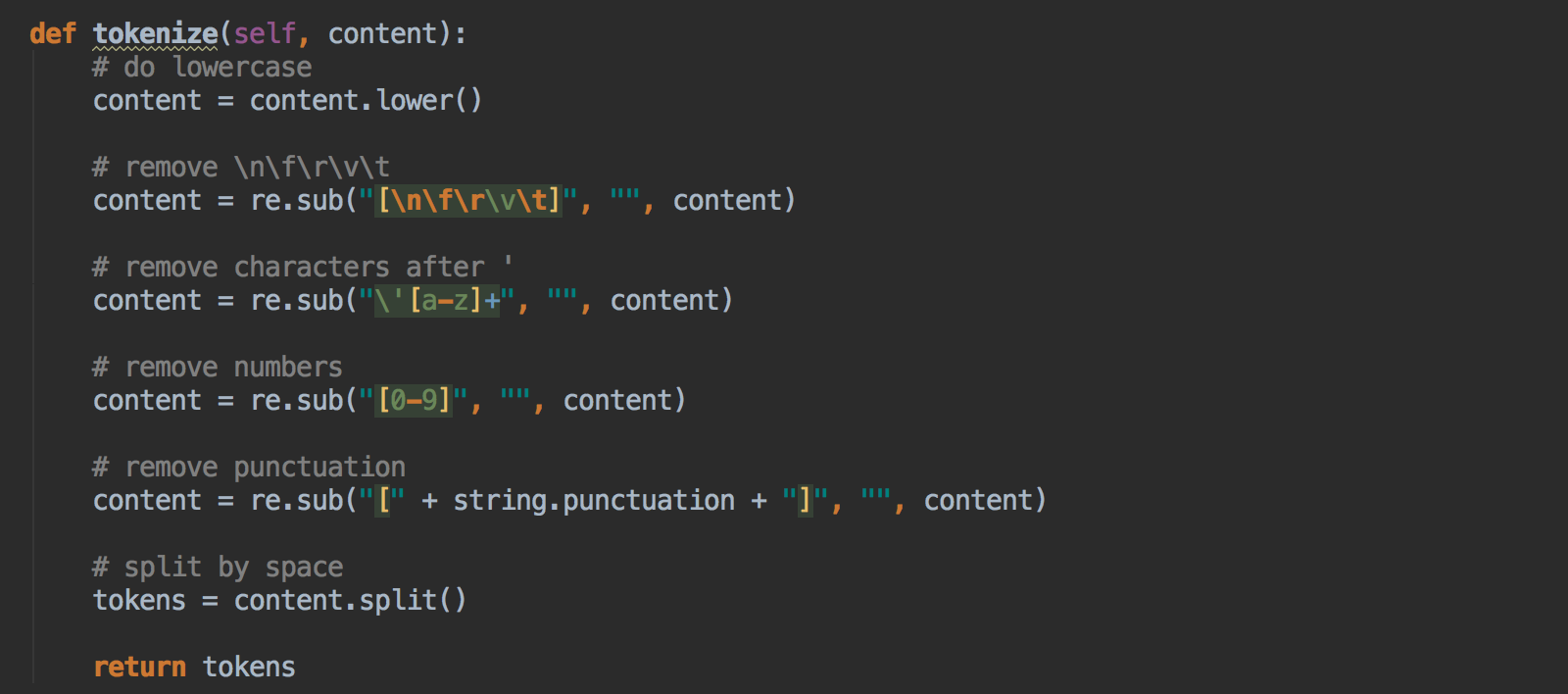
1. 開啟終端機，將路徑設置到PA-2資料夾所在的位置，輸入**python dictionary.py，**待程式執行一段時間結束後，會在PA-2的資料夾產生一個新的**dictionary.txt**的檔案，而這個檔案就是所有1095篇document中經過處理後所建立的dictionary，內容包含**t\_index**、**term**以及**term frequency**。
2. 接著，在終端機輸入**python tfidf.py**，待程式執行一段時間結束後，會在PA-2底下的TF-IDF資料夾中，產生**1.txt**、**2.txt** … **1095.txt**，而這每一個text檔案則是從原先的1095篇document中，算出對應document id的每個term的**TF-IDF**值，內容包含**t\_index**和**tf-idf**。
3. 最後，一樣在終端機輸入**python vector\_space.py**，待程式執行一段時間結束後，會在終端機顯示**cosine similarity**的值，目前是預設算**1.txt**和**2.txt**的**cosine similarity**。
4. 作業處理邏輯：
5. **dictionary.py**

在dictionary.py中定義了**Class Dictionary**，而在Class Dictionary中定義了一些function。

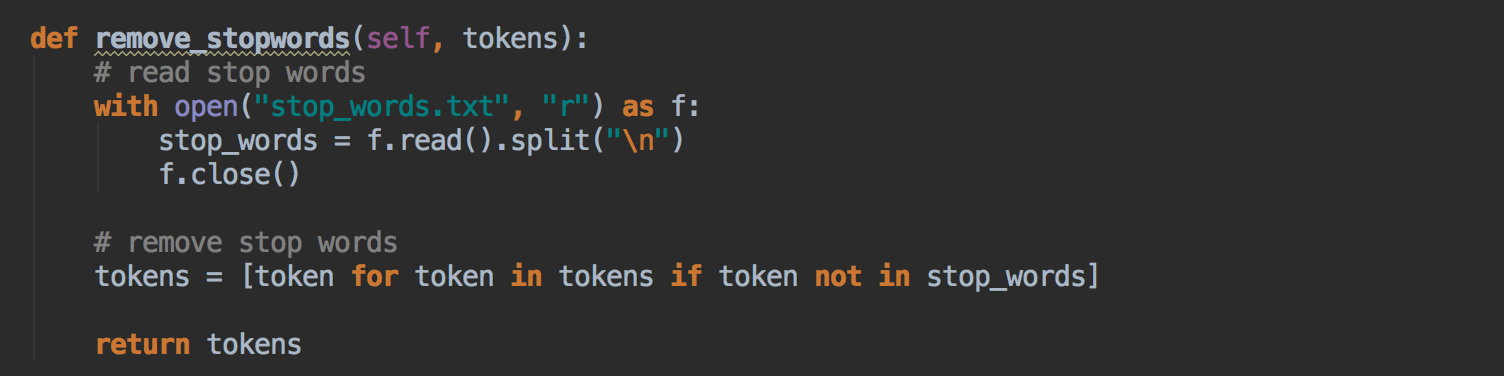
第一個function是**read\_files**，read\_files主要是要讀取所有1095篇的document，將每一篇document的content相加並回傳。



第二個function是**tokenize**，tokenize主要是會將readFiles所回傳的content利用regular expression過濾，首先將content進行lowercase，再過濾掉\n、\f、\r、\v、\t、數字以及標點符號等，最後以空白做為區隔進行split。



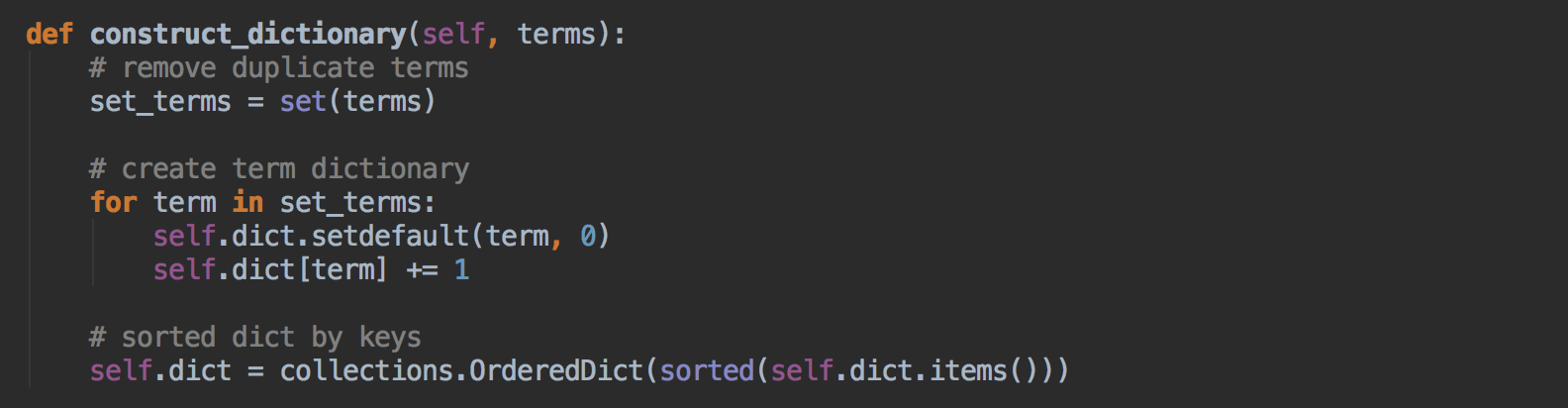
第三個function是**remove\_stopwords，remove\_stopwords**主要會讀取PA-2資料夾中的stop\_words.txt，並以\n進行split，回傳stop words的list。



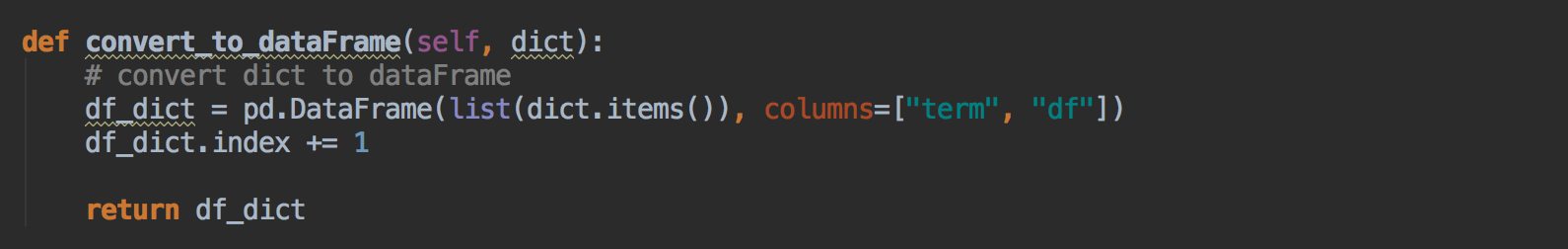
第四個function是**stemming**，stemming主要會建立**PorterStemmer**的instance，將remove\_stopwords所回傳的list，透過PorterStemmer進行stem，並回傳list。



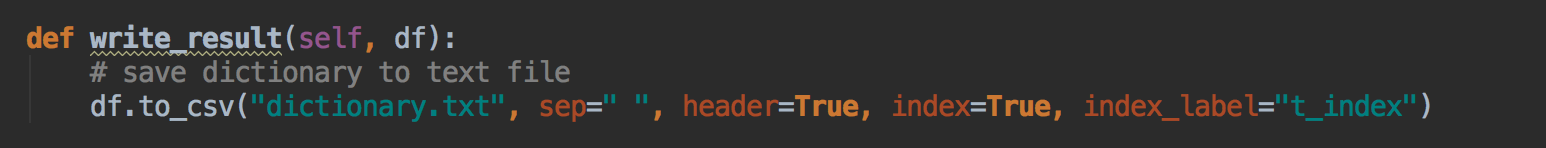
第五個function是**construct\_dictionary**，construct\_dictionary會將stemming所回傳的terms，先過濾掉重複的term，接著建立一個dictionary，把term當作key，出現次數當作value，記錄每個term出現在所有document中的次數，並回傳dictionary。



第六個function是**convert\_to\_dataFrame**，convert\_to\_dataFrame會將construct\_dictionary所回傳的dictionary轉型成dataFrame，並新增t\_index的column，最後回傳dataFrame。



第七個function是**write\_result**，wirte\_result會將covert\_to\_dataFrame所回傳的dataFrame存成text檔案。



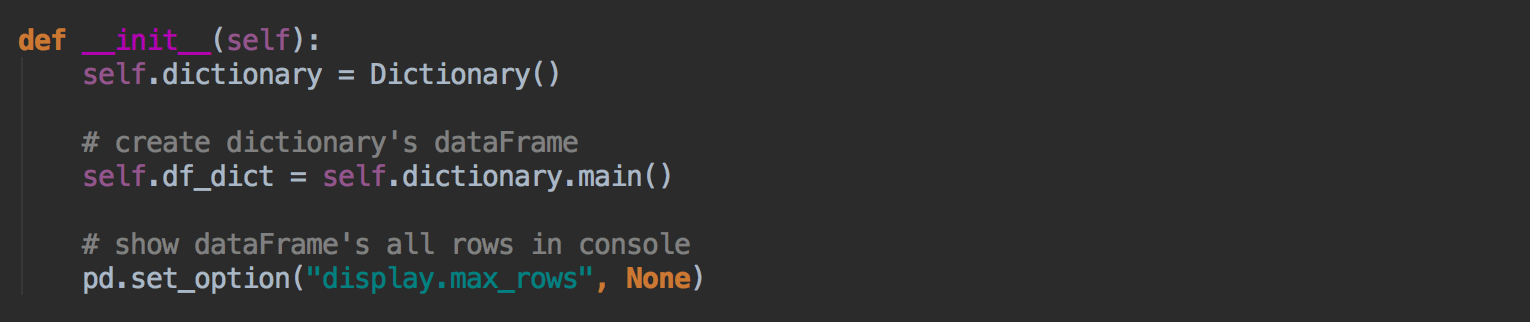
第八個function是**main**，main會先取得所有document的path，以一個list儲存起來。接著，對list做迴圈，進行read\_files、tokenize、remove\_stopwords、stemming、construct\_dictionary，最後把所有document所建立dictionary進行convert\_to\_dataFrame，再以write\_result儲存起來。



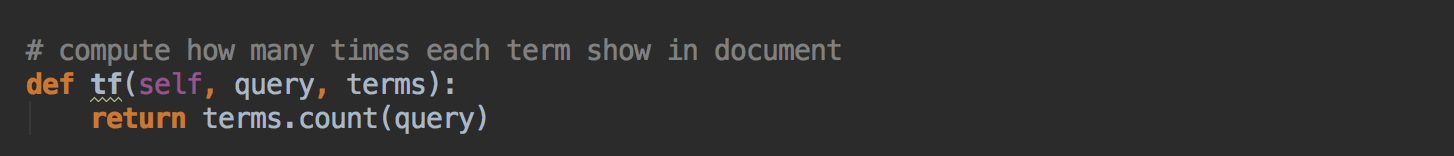
1. **tfidf.py**

在tfidf.py中定義了**Class TFIDF**，而在Class TFIDF中定義了一些function。

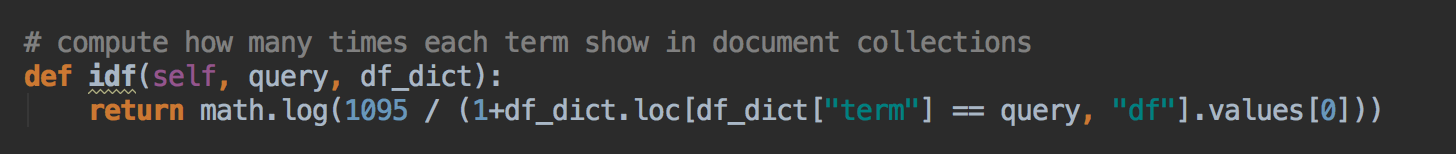
首先，在**\_\_init\_\_**的constructor中，會呼叫Dictionary Instance的main function，建立一個dictionary dataFrame，以df\_dict作為變數儲存起來。



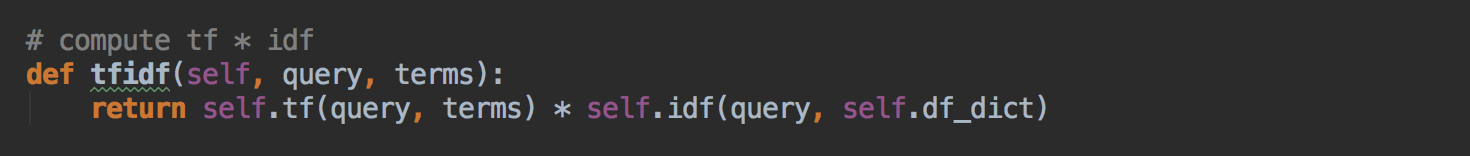
第一個function是**tf**，tf會計算input的query在特定terms的出現次數，計算query的出現次數 / 所有term的個數，並回傳結果。



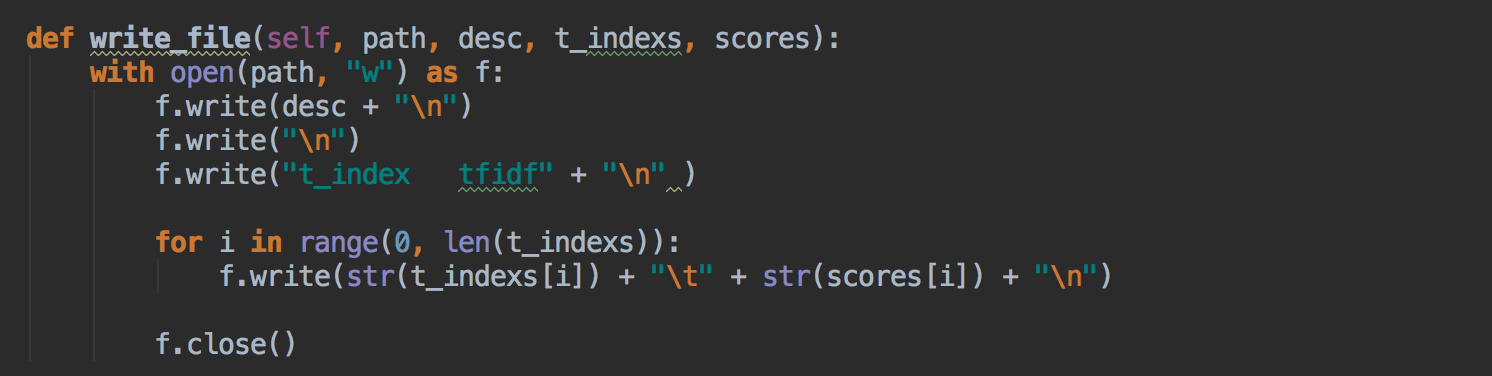
第二個function是**idf**，idf會計算input的query在所有的document中的出現次數，計算log值並回傳結果。



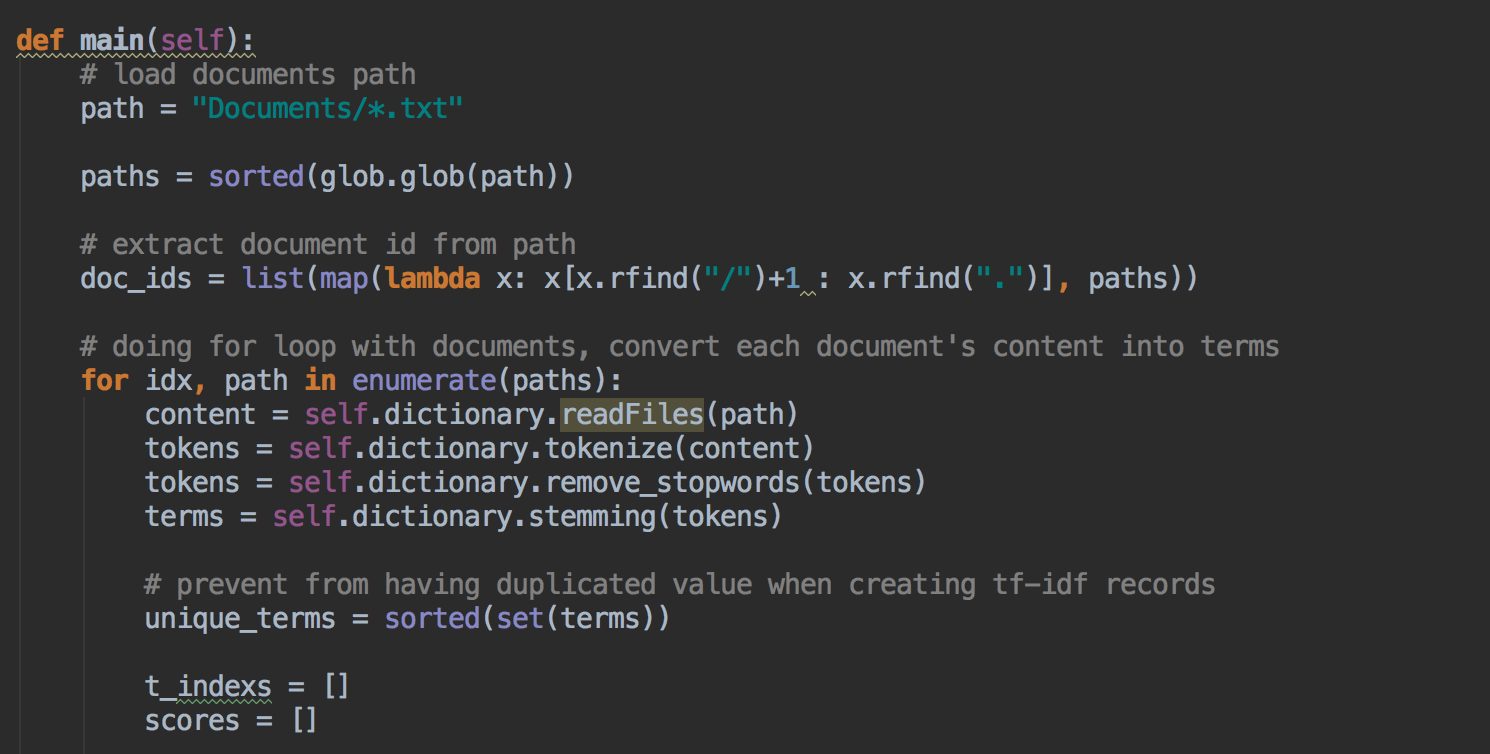
第三個function是**tfidf**，tfidf會計算tf的值乘上idf的值，並回傳結果。

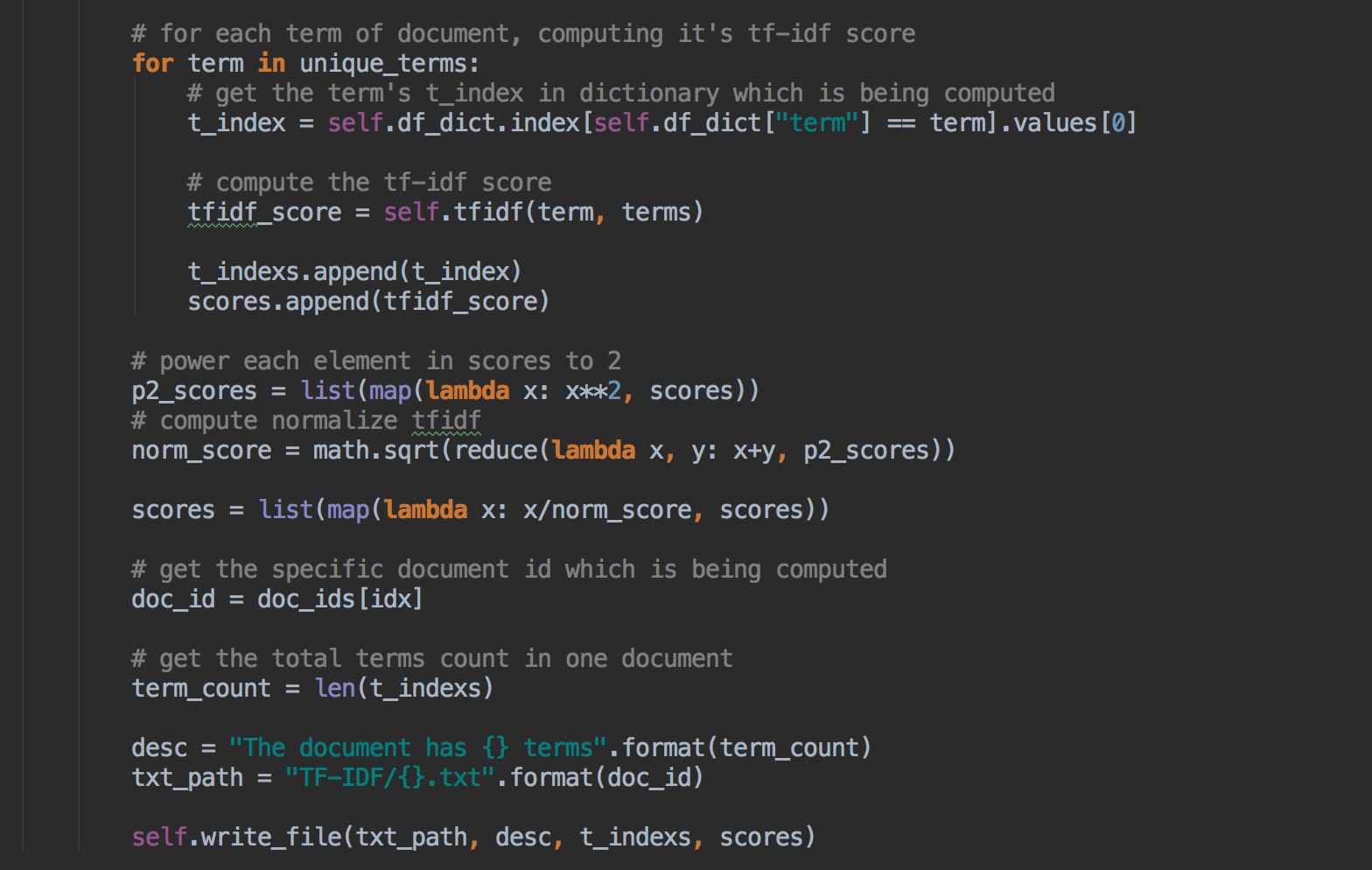


第四個function是**write\_file**，write\_file會將input的t\_indexs和scores儲存成text檔案。



第四個function是**main**，main會先list儲存所有document的path，接著再對list做迴圈，進行read\_files、tokenize、remove\_stopwords以及stemming後會得到terms，再對terms做迴圈，找出term在df\_dict中的t\_index，並計算每個term的tf-idf值，再對每個tf-idf值做normalize，最後透過write\_file儲存成text檔案。

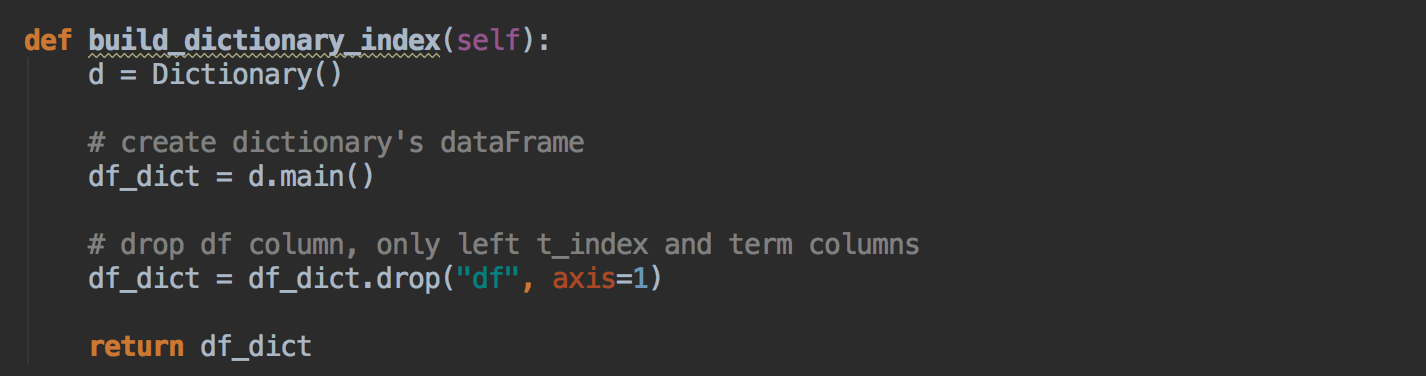




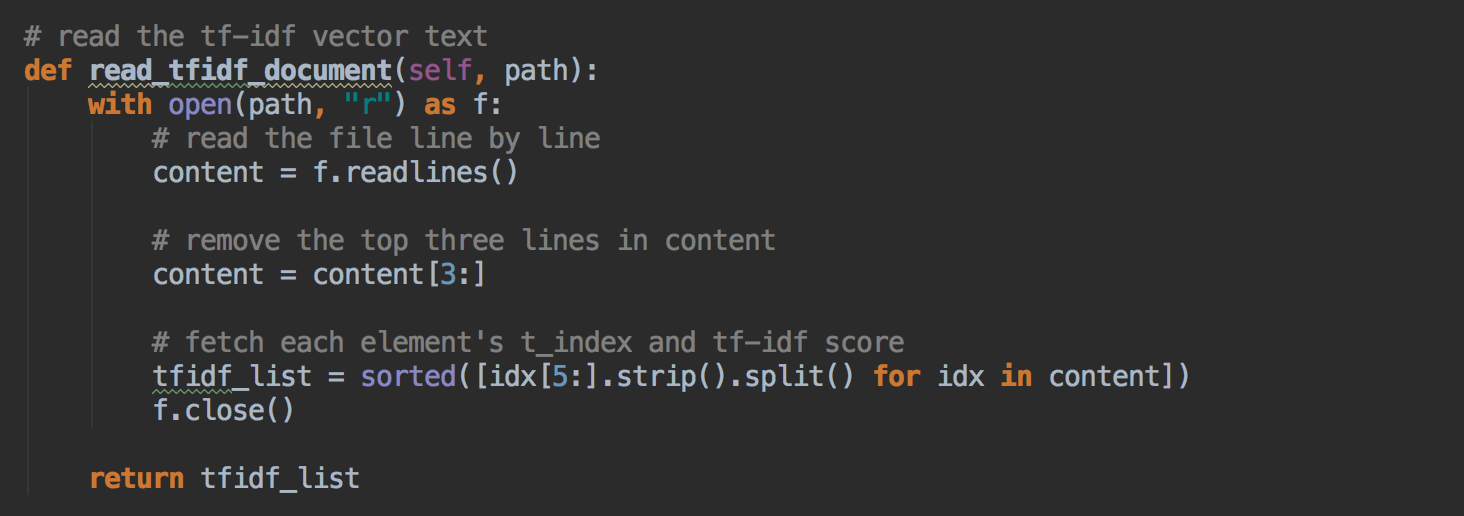
1. **vector\_space.py**

在vecotr\_space.py中定義了**Class Vector\_Space**，而在Class Vector\_Space中定義了一些function。

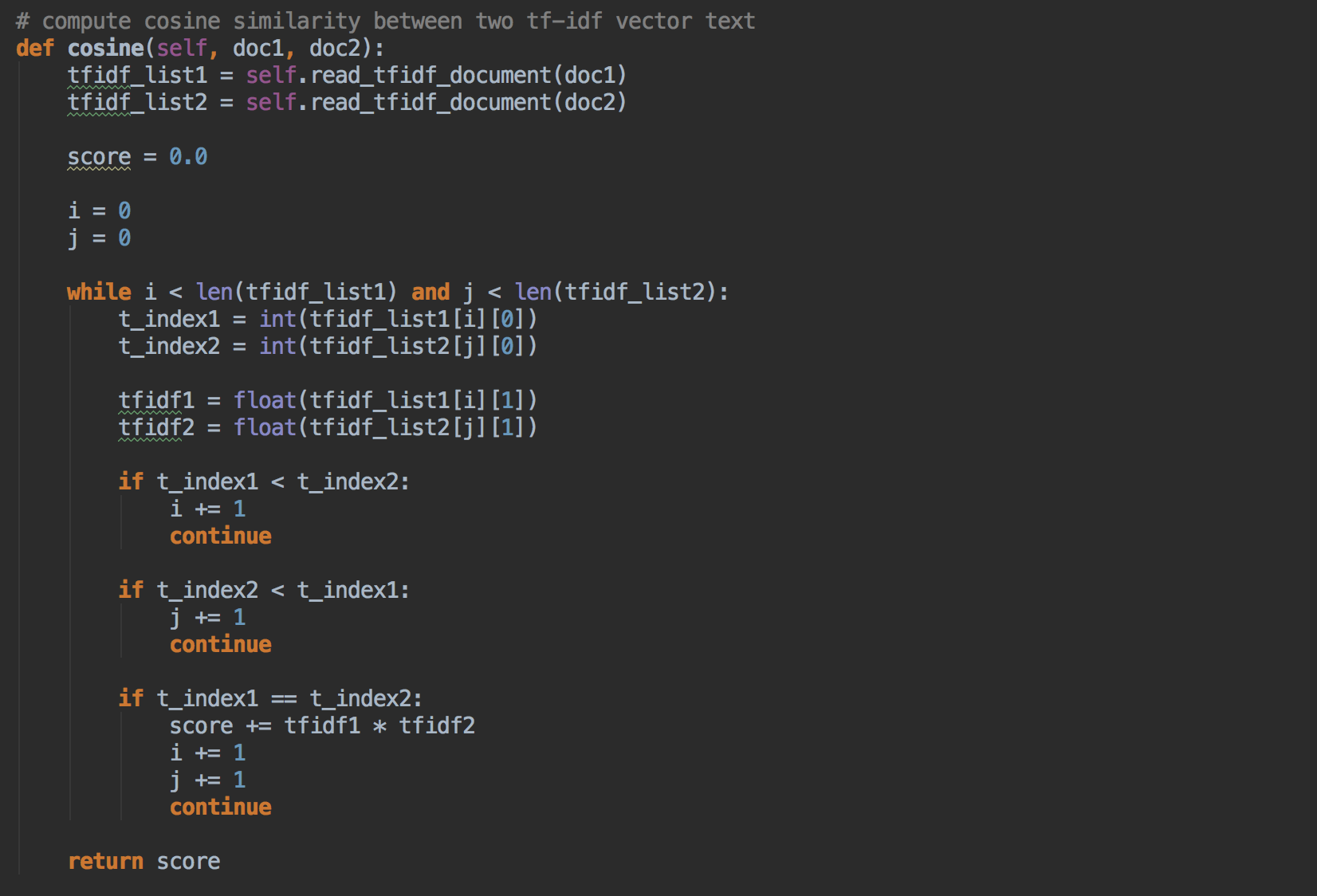
第一個function是**build\_dictionary\_index**，build\_dictionary\_index會先建立Dictionary的instance，透過Dictionary的main取得dataFrame，drop掉df欄位，並回傳dataFrame



第二個function是**read\_tfidf\_document**，read\_tfidf\_document會先讀取由tfidf.py的main函數所建立的text檔，取出t\_index以及tf-idf的值，儲存在tfidf\_list中。



第三個function是**cosine**，cosine會對input的doc1和doc2透過read\_tfidf\_document分別建立tfidf\_list1和tfidf\_list2。接著進行while迴圈，如果tfidf\_list1中的t\_index與tfidf\_list2的t\_index，則將兩邊的tfidf值相乘後加到score



1. 運行結果：可參考PA-2資料夾中的dictionary.txt檔案以及TF-IDF資料夾中的1.txt、2.txt … 1095.txt，而cosine similarity則可透過執行vector\_space.py看到運算結果(預設是計算1.txt和2.txt)。