*#how to spot the list inside the list***import** os  
**import** nltk *#for nltk download  
#nltk.download()***from** nltk.tokenize **import** RegexpTokenizer *#for tokenize***from** nltk.tokenize **import** word\_tokenize  
**from** nltk.corpus **import** stopwords *#for stopwords downlaod***from** nltk.stem.porter **import** PorterStemmer *#for stemmer***from** collections **import** Mapping, defaultdict  
  
**def** tokenize(line): *# tokenize the line into words* tokenizer = RegexpTokenizer(**'\w+|\$[\d\.]+|\S+'**)  
 **return** tokenizer.tokenize(line)  
*# ===============================  
#1. read the file == lowercase  
# ===========================  
  
#filename = './debate.txt'*filename = **'./test.txt'**file = open(filename, **"r"**, encoding=**'UTF-8'**)  
doc = file.readlines()  
  
list1 = []  
**for** line **in** doc:  
 **if**(len(line.strip())!= 0): *#checks for the empty line* line = tokenize(line) *#gives the tokenized version of the line* list1.append(line) *#appends the tokenized paragraph to the list1***for** i **in** range(len(list1)): *#prints the list of the each paragraph* print(**"List No: "**,i, list1[i],**" >> "** ,len(list1[i]))  
  
print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
print(**"\n>>> No of Paragraph = "**, len(list1),**">>"**)  
  
*# ===========================  
# 2. Stop Word Removal  
# =============================*stop\_list = []  
  
**for** words **in** stopwords.words(**'english'**): *#converts the downloaded stop words into list of stopwords* stop\_list.append(words)  
*# for i in range(len(stop\_list)): #prints the list of the stopwords  
# print("Stopword No: ",i+1 , stop\_list[i]," >> ")***for** i **in** range(len(list1)):  
 temp = [] *#store the temporary stopwords removed list* temp\_word = **''** *#temp list to store the words for comparison* **for** word **in** list1[i]: *#iterates through words in the given list* temp = list1[i] *#temp list intilized at the index* temp\_word = word *#temp word intilized for comparison* **for** j **in** range(len(stop\_list)):  
 **if**(temp\_word == stop\_list[j]): *#if the word in the list is same as stopword list,* temp.remove(temp\_word) *#then remove from the list* list1[i] = temp *#updates the list no stopwords list = temp*print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
**for** i **in** range(len(list1)): *#prints list with no stop word* print(**"No Stopword List No: "**,i, list1[i],**" >> "** ,len(list1[i]))  
  
*#==========================  
# 3. Stemms the words in the list  
#==========================***for** i **in** range(len(list1)):  
 temp = [] *#temp list to store the stemmed words* stemmer = PorterStemmer()  
 **for** word **in** list1[i]: *#iterates through words in the given list* temp.append(stemmer.stem(word)) *#appends the stemmed words in the temp list* list1[i] = temp *#updates the list with the stemmed words list = temp*print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
**for** i **in** range(len(list1)): *#prints the stemmed list* print(**"Stemmed List No: "**,i, list1[i],**" >> "** ,len(list1[i]))  
  
*#===================================  
# 4. Dictionary of the (key, count)  
#===================================*dic = {} *#introduction of the dictionary variable***for** i **in** range(len(list1)):  
 **for** word **in** list1[i]:  
 dic[word] = dic.get(word, 0) + 1 *#idiom: update the counter very time the word is found, else zero*print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
print(**"List Dictionary: \n "**, dic) *#prints the list of the dictionary  
  
#==================================  
#  
#==================================*file.close()  
  
  
**while**(**True**):  
 user = input(**"Enter the query you want to perform: "**)  
  
*###################################*

++++++++++++++++++++++++++++++++++++++second phase+++++++++++++++

*#how to spot the list inside the list***import** os  
**import** re *#for tokinization***import** math *#for log calc***import** nltk *#for nltk download  
#nltk.download()***from** nltk.tokenize **import** RegexpTokenizer *#for tokenize***from** nltk.tokenize **import** word\_tokenize  
**from** nltk.corpus **import** stopwords *#for stopwords downlaod***from** nltk.stem.porter **import** PorterStemmer *#for stemmer***from** collections **import** Mapping, defaultdict  
  
**def** tokenize(line): *# tokenize the line into words* tokenizer = RegexpTokenizer(**'\w+|\$[\d\.]+|\S+""'**)  
 **return** tokenizer.tokenize(line)  
**def** string\_token(string):  
 **return** string.split()  
  
  
  
**def** swremover(list1):  
 stop\_list = []  
 **for** words **in** stopwords.words(**'english'**): *# converts the downloaded stop words into list of stopwords* stop\_list.append(words)  
 *# for i in range(len(stop\_list)): #prints the list of the stopwords  
 # print("Stopword No: ",i+1 , stop\_list[i]," >> ")* **for** i **in** range(len(list1)):  
 temp = [] *# store the temporary stopwords removed list* temp\_word = **''** *# temp list to store the words for comparison* **for** word **in** list1[i]: *# iterates through words in the given list* temp = list1[i] *# temp list intilized at the index* temp\_word = word *# temp word intilized for comparison* **for** j **in** range(len(stop\_list)):  
 **if** (temp\_word == stop\_list[j]): *# if the word in the list is same as stopword list,* temp.remove(temp\_word) *# then remove from the list* list1[i] = temp *# updates the list no stopwords list = temp* print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
 **for** i **in** range(len(list1)): *# prints list with no stop word* print(**"No Stopword List No: "**, i, list1[i], **" >> "**, len(list1[i]))  
  
 **return** list1  
  
**def** stemmize (list1):  
 **for** i **in** range(len(list1)):  
 temp = [] *# temp list to store the stemmed words* stemmer = PorterStemmer()  
 **for** word **in** list1[i]: *# iterates through words in the given list* temp.append(stemmer.stem(word)) *# appends the stemmed words in the temp list* list1[i] = temp *# updates the list with the stemmed words list = temp* print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
 **for** i **in** range(len(list1)): *# prints the stemmed list* print(**"Stemmed List No: "**, i, list1[i], **" >> "**, len(list1[i]))  
 **return** list1  
  
**def** dictionize(list1):  
 dic = {} *# introduction of the dictionary variable* **for** i **in** range(len(list1)):  
 **for** word **in** list1[i]:  
 dic[word] = dic.get(word, 0) + 1 *# idiom: update the counter very time the word is found, else zero* print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
 *#print("List Dictionary: \n ", dic) # prints the list of the dictionary* **return** dic  
*#===============================***def** dictionize2(list1):  
 dic = {} *# introduction of the dictionary variable* **for** word **in** list1:  
 dic[word] = dic.get(word, 0) + 1 *# idiom: update the counter very time the word is found, else zero* print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
 print(**"List Dictionary: \n "**, dic) *# prints the list of the dictionary* **return** dic  
  
  
*# ===============================  
#1. read the file == lowercase  
# ===========================*filename = **'./debate.txt'***#filename = './test.txt'*file = open(filename, **"r"**, encoding=**'UTF-8'**)  
  
doc = file.readlines()  
list1 = []  
list2 = []  
**for** line **in** doc:  
 **if**(len(line.strip())!= 0): *#checks for the empty line* line = tokenize(line) *#gives the tokenized version of the line* list1.append(line) *#appends the tokenized paragraph to the list1***for** i **in** range(len(list1)): *#prints the list of the each paragraph* print(**"List No: "**,i, list1[i],**" >> "** ,len(list1[i]))  
  
print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
print(**"\n>>> No of Paragraph = "**, len(list1),**">>"**)  
*#=================================*list1 = swremover(list1)  
print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
**for** i **in** range(len(list1)): *# prints list with no stop word* print(**"Func. No StopWord List: "**, i, list1[i], **" >> "**, len(list1[i]))  
*#==================================*list1 = stemmize(list1)  
print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
**for** i **in** range(len(list1)): *# prints the stemmed list* print(**"Func. Stemmed List No: "**, i, list1[i], **" >> "**, len(list1[i]))  
*#=================================*diction = {}  
diction = dictionize(list1)  
print(**"Overall Dictionary: "**, diction)  
*# diction = []  
# for i in range(len(list1)):  
# temp\_dic = {}  
# temp\_dic = dictionize2(list1[i])  
# diction[i].append(temp\_dic)  
  
#print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
#for i in range(len(list1)): # prints the stemmed list  
 # print("Dictionary: ",i, diction[i]) # prints the list of the dictionary  
#================================***for** i **in** range(len(list1)): *#concatnates the individual paragraph into one list for the whole file* list2 += list1[i]  
print(**"Whole text: "**, list2,**" >> "** ,len(list2))  
  
*#variables*N = len(list1)  
print(**"Number of documents(N) = "**, N)  
  
**while**(**True**):  
 command = input(**'Enter command :: '**)  
 *#for the getidf, preliminary calculation* **if**(**'getidf' in** command):  
  
 idf = 0 *#idf variable for the document frequencey* str = **" "** *#temporary string variable to store the command* strg = **" "** slist = []  
 strg = command *#storing the command in the temp variable = strg* slist = tokenize(strg) *#storing the command as tokens in the list = slist* print(slist)  
 str = slist[1] *#Assuming the last token is the word to be evaluated* stemmer = PorterStemmer() *#using stemmer to stem the word* str = stemmer.stem(str)  
 print (**"before stemmed: "**, str, idf)  
 **for** i **in** range(len(list1)): *#for loop to check if the word exist in the list,* **for** word **in** list1[i]:  
 **if** (str == word): *#if exist then increment the idf counter and break out the document list* idf = idf+1  
 **break** print(**"after stemmed: "**, str, idf)  
  
 **if** idf == 0: *#if word not in document, then set defualt idf to 1* idf = 1  
 print(**"No Match = "**,idf)  
 **else**: *#else calculat the idf with formula* idf = math.log(N/idf) *# idf = log (N/idf)* print(**"Idf :: "**, idf)  
  
 **if** (**'getqvec'in** command):  
   
 print(**"qvecc..."**)  
  
 **else**:  
 print(**'enter again'**)  
  
  
*#=================================*file.close()  
  
  
*#string\_token(str)*

*#how to spot the list inside the list***import** os  
**import** re *#for tokinization***import** math *#for log calc***import** nltk *#for nltk download  
#nltk.download()***from** nltk.tokenize **import** RegexpTokenizer *#for tokenize***from** nltk.tokenize **import** word\_tokenize  
**from** nltk.corpus **import** stopwords *#for stopwords downlaod***from** nltk.stem.porter **import** PorterStemmer *#for stemmer***from** collections **import** Mapping, defaultdict  
  
*#tokenize function for line and word***def** tokenize(line): *# tokenize the line into words* tokenizer = RegexpTokenizer(**'\w+|\$[\d\.]+|\S+""'**)  
 **return** tokenizer.tokenize(line)  
**def** string\_token(string):  
 **return** string.split()  
  
  
*# turns the list into lower case***def** lowerize(list1):  
 **for** i **in** range(len(list1)):  
 temp = []  
 **for** word **in** list1[i]:  
 temp.append(word.lower())  
 list1[i]= temp  
 *#print("Lower list: ", i, list1[i])* **return** list1  
  
  
*#Stop word removal function for the list of words***def** swremover(list1):  
 stop\_list = []  
 **for** words **in** stopwords.words(**'english'**): *# converts the downloaded stop words into list of stopwords* stop\_list.append(words)  
 *# for i in range(len(stop\_list)): #prints the list of the stopwords  
 # print("Stopword No: ",i+1 , stop\_list[i]," >> ")* **for** i **in** range(len(list1)):  
 temp = [] *# store the temporary stopwords removed list* temp\_word = **''** *# temp list to store the words for comparison* **for** word **in** list1[i]: *# iterates through words in the given list* temp = list1[i] *# temp list intilized at the index* temp\_word = word *# temp word intilized for comparison* **for** j **in** range(len(stop\_list)):  
 **if** (temp\_word == stop\_list[j]): *# if the word in the list is same as stopword list,* temp.remove(temp\_word) *# then remove from the list* list1[i] = temp *# updates the list no stopwords list = temp* print(**"------------------------------------------------"**)  
 *# for i in range(len(list1)): # prints list with no stop word  
 # print("No Stopword List No: ", i, list1[i], " >> ", len(list1[i]))* **return** list1  
  
*#Stemming function of the list of words***def** stemmize (list1):  
 **for** i **in** range(len(list1)):  
 temp = [] *# temp list to store the stemmed words* stemmer = PorterStemmer()  
 **for** word **in** list1[i]: *# iterates through words in the given list* temp.append(stemmer.stem(word)) *# appends the stemmed words in the temp list* list1[i] = temp *# updates the list with the stemmed words list = temp* print(**"------------------------------------------------"**)  
 *# for i in range(len(list1)): # prints the stemmed list  
 # print("Stemmed List No: ", i, list1[i], " >> ", len(list1[i]))* **return** list1  
  
*#stems the sample list***def** list\_stem(sam\_list):  
 temp = []  
 **for** word **in** sam\_list:  
 stemmer = PorterStemmer()  
 temp.append((stemmer.stem(word)))  
 sam\_list = temp  
  
  
*#dictionary function to creat the dictionary of whole words***def** dictionize(list1):  
 dic = {} *# introduction of the dictionary variable* **for** i **in** range(len(list1)):  
 **for** word **in** list1[i]:  
 dic[word] = dic.get(word, 0) + 1 *# idiom: update the counter very time the word is found, else zero* print(**"------------------------------------------------"**)  
 *#print("List Dictionary: \n ", dic) # prints the list of the dictionary* **return** dic  
*#===============================  
  
#sample dictionary of just a single list***def** dictionize2(list1):  
 dic = {} *# introduction of the dictionary variable* **for** word **in** list1:  
 dic[word] = dic.get(word, 0) + 1 *# idiom: update the counter very time the word is found, else zero* print(**"------------------------------------------------"**)  
 print(**"List Dictionary: \n "**, dic) *# prints the list of the dictionary* **return** dic  
*#====================================  
# Dft\_ finder: finds the document frequency of the given word  
#===================================***def** dft\_finder(str):  
  
 idf = 0  
 stemmer = PorterStemmer() *# using stemmer to stem the word* str = stemmer.stem(str)  
 *#print("Func. before stemmed: ", str, idf)* **for** i **in** range(len(list1)): *# for loop to check if the word exist in the list,* **for** word **in** list1[i]:  
 **if** (str == word): *# if exist then increment the idf counter and break out the document list* idf = idf + 1  
 **break** *#print("Func. after stemmed: ", str, idf)* **if** idf == 0: *# if word not in document, then set defualt idf to 1* idf = 1  
 *#print("Func. No Match %.4f= ", idf)* **else**: *# else calculat the idf with formula* idf = math.log10(N / idf) *# idf = log10 (N/idf)  
 #print("Func. Idf %.4f:: ", idf)* **return** idf  
*#=========================================***def** tf\_finder(str, strlist):  
 tf = 0  
 stemmer = PorterStemmer() *# using stemmer to stem the word* str = stemmer.stem(str)  
 *#print("TF Func. before stemmed: ", str, tf)* **for** word **in** strlist:  
 **if** (str == word):  
 tf = tf + 1  
 *#print("TF Func. After stem: ", str, tf)* tf = (1 + math.log10(tf))  
 **return** tf  
  
*#===========================================  
  
# ===============================  
#1. read the file  
# ===========================*filename = **'./debate.txt'***#filename = './test.txt'*file = open(filename, **"r"**, encoding=**'UTF-8'**)  
  
doc = file.readlines()  
list1 = []  
list2 = []  
**for** line **in** doc:  
 **if**(len(line.strip())!= 0): *#checks for the empty line* line = tokenize(line) *#gives the tokenized version of the line* list1.append(line) *#appends the tokenized paragraph to the list1***for** i **in** range(len(list1)): *#prints the list of the each paragraph* print(**"List No: "**,i, list1[i],**" >> "** ,len(list1[i]))  
print(**"------------------------------------------------"**)  
print(**"\n>>> No of Paragraph = "**, len(list1),**">>"**)  
*#=================================*list1 = swremover(list1)  
print(**"------------------------------------------------"**)  
**for** i **in** range(len(list1)): *# prints list with no stop word* print(**"Func. No StopWord List: "**, i, list1[i], **" >> "**, len(list1[i]))  
  
*#==================================*print(**"------------------------------------------------"**)  
  
list1 = stemmize(list1)  
**for** i **in** range(len(list1)): *# prints the stemmed list* print(**"Func. Stemmed List No: "**, i, list1[i], **" >> "**, len(list1[i]))  
  
*#=================================*print(**"------------------------------------------------"**)  
**for** i **in** range(len(list1)): *#prints the list of the each paragraph* print(**"Sim List No: "**,i, list1[i],**" >> "** ,len(list1[i]))  
print(**"------------------------------------------------"**)  
list1 = lowerize(list1) *#lowerises the whole list1***for** i **in** range(len(list1)): *#prints the list of the each paragraph* print(**"Lower List No: "**,i, list1[i],**" >> "** ,len(list1[i]))  
  
diction = {} *#presents the dictionary for the whole text file: not useful*diction = dictionize(list1)  
*#print("Overall Dictionary: ", diction)  
#==============================***for** i **in** range(len(list1)): *#concatnates the individual paragraph into one list for the whole file* list2 += list1[i]  
print(**"Whole text: "**, list2,**" >> "** ,len(list2))  
  
*#variables*N = len(list1)  
print(**"Number of documents(N) = "**, N)  
  
*#======================================  
#takes the string list and gives the weight vector***def** weight\_vector\_finder (strlist):  
 wt\_vec = {}  
 TF = 0  
 IDF = 0  
  
 **for** word **in** strlist:  
 TF = tf\_finder(word,strlist)  
 IDF = dft\_finder(word)  
 value = TF \* IDF  
 print(word , value)  
 wt\_vec[word] = value  
  
 print(**"The weight vector dictionary: %.4f"**, wt\_vec)  
 **return** wt\_vec  
*#====================================  
#takes the string and the paragraph and gives the desired weight vector  
# not yet working***def** getqvec\_funct(str, strlist):  
 wt\_vec = {}  
 TF = 0  
 IDF = 0  
 str = list\_stem(str)  
 **for** i **in** range(len(strlist)):  
 **for** j **in** range(len(str)): *#needs to be fixed* **if**(str[j]==strlist[i] ):  
 TF = tf\_finder(str[j], strlist)  
 IDF = dft\_finder(str[j])  
 value = TF \* IDF  
 print(str[j], value)  
 wt\_vec[str[j]] = value  
  
 print(**"Weight vector for slected wordlist: "**, wt\_vec)  
 **return** wt\_vec  
*#==========================================***while**(**True**):  
 command = input(**'Enter command :: '**)  
  
 *#for the getidf, preliminary calculation* **if**(**'getidf' in** command):  
  
 idf = 0 *#idf variable for the document frequencey* str = **" "** *#temporary string variable to store the command* strg = **" "** slist = []  
 strg = command *#storing the command in the temp variable = strg* slist = tokenize(strg) *#storing the command as tokens in the list = slist* print(slist)  
 str = slist[len(slist)-1] *#Assuming the last token is the word to be evaluated  
  
 #===============* idf = dft\_finder(str)  
 print(**"Function IDF %.4f : "**, idf)  
*#* **if** (**'getqvec'in** command):  
  
 print(**"qvecc..."**)  
  
 tf = 0  
 idf = 0  
  
 idf = dft\_finder(**"this"**) *#checks the idf for the sample word* print(**"Idf Returned: %.4f "**, idf)  
 print(**"\n"**)  
 tf = tf\_finder( **"this"**, list1[0]) *#prints the tf for the smaple list* print(**"TF Returned: %.4f"**, tf)  
  
 wt\_vector = {}  
 wt\_vector = weight\_vector\_finder(list1[0])  
 print(**"The Wt vector:"**,wt\_vector)  
  
 wt\_vector2 ={}  
 sam\_list = [**'this'**, **'is'**, **'country'**]  
  
 wt\_vector2 = getqvec\_funct(sam\_list,list1[0])  
  
 **if** (command == **'quit'**):  
 print(**"Exiting the Program...."**)  
 exit()  
 **else**:  
 print(**'enter again'**)  
  
  
*#=================================*file.close()  
  
  
*#string\_token(str)*