In [1]:

```
pip install requests beautifulsoup4 lxml
```

```
Requirement already satisfied: beautifulsoup4 in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (4.11.1)
Collecting lxml
  Downloading lxml-4.9.1-cp39-cp39-win_amd64.whl (3.6 MB)
     ---------------------------------------- 3.6/3.6 MB 171.4 kB/s eta 0:00:00
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.13-py2.py3-none-any.whl (140 kB)
     ---------------------------------- 140.6/140.6 kB 157.4 kB/s eta 0:00:00
Collecting charset-normalizer<3,>=2
  Downloading charset_normalizer-2.1.1-py3-none-any.whl (39 kB)
Collecting idna<4,>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
     ---------------------------------------- 61.5/61.5 kB 80.2 kB/s eta 0:00:00
Requirement already satisfied: certifi>=2017.4.17 in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (from requ
ests) (2022.6.15)
Requirement already satisfied: soupsieve>1.2 in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (from beautiful
soup4) (2.3.1)
Installing collected packages: urllib3, lxml, idna, charset-normalizer, requests
Successfully installed charset-normalizer-2.1.1 idna-3.4 lxml-4.9.1 requests-2.28.1 urllib3-1.26.13
```

In [2]:

```
import os
```

In [5]:

```
from collections import deque
```

In [3]:

```
#Python program to scrape website
#and save quotes from website
url_set=set()
url_queue=deque()
import requests
from bs4 import BeautifulSoup
import csv

URL = "https://www.javatpoint.com/python-tutorial"
r = requests.get(URL)

soup = BeautifulSoup(r.content, 'html5lib')

quotes=[] # a list to store quotes

table = soup.find('div', attrs = {'id':'menu'})
print(table)
```

```
<div id="menu">
<div class="leftmenu2">
<h2 class="spanh2"><span class="spanh2">Python Tutorial</span></h2>
</div>
<div class="leftmenu">
<a href="python-tutorial">Python Tutorial</a>
<a href="python-features">Python Features</a>
<a href="python-history">Python History</a>
<a href="python-applications">Python Applications</a>
<a href="how-to-install-python">Python Install</a>
<a href="python-example">Python Example</a>
<a href="python-variables">Python Variables</a>
<a href="python-data-types">Python Data Types</a>
<a href="python-keywords">Python Keywords</a>
<a href="python-literals">Python Literals</a>
<a href="python-operators">Python Operators</a>
<a href="python-comments">Python Comments</a>
<a href="python-if-else">Python If else</a>
<a href="python-loops">Python Loops</a>
```

In [4]:

```
for link in table.find_all('a'):
    url_set.add('https://www.javatpoint.com/'+link.get('href'))
    url_queue.append('https://www.javatpoint.com/'+link.get('href'))
print(url_set)
```

```
{'https://www.javatpoint.com/python-front-end-framework', 'https://www.javatpoint.com/data-hiding-in-python', 'https://w
ww.javatpoint.com/python-unit-testing', 'https://www.javatpoint.com/python-audio-modules', 'https://www.javatpoint.com/h
ow-to-add-two-lists-in-python', 'https://www.javatpoint.com/python-graphviz', 'https://www.javatpoint.com/creating-inter
active-pdf-forms-using-python', 'https://www.javatpoint.com/modules-vs-packages-in-python', 'https://www.javatpoint.com/
python-new-line', 'https://www.javatpoint.com/agg-function-in-python', 'https://www.javatpoint.com/digital-clock-using-p
yqt5-in-python', 'https://www.javatpoint.com/python-strings', 'https://www.javatpoint.com/python-read-csv-file', 'http
s://www.javatpoint.com/python-time-module', 'https://www.javatpoint.com/solve-physics-computational-problems-using-pytho
n', 'https://www.javatpoint.com/python-list-comprehension', 'https://www.javatpoint.com/best-resources-to-learn-numpy-an
d-pandas', 'https://www.javatpoint.com/dbscan-algorithm-in-python', 'https://www.javatpoint.com/how-to-get-index-of-elem
ent-in-list-python', 'https://www.javatpoint.com/convert-string-to-dictionary-in-python', 'https://www.javatpoint.com/mo
de-in-python', 'https://www.javatpoint.com/heap-sort-in-python', 'https://www.javatpoint.com/python-program-for-word-gue
ssing-game', 'https://www.javatpoint.com/project-in-python-breast-cancer-classification-with-deep-learning', 'https://ww
w.javatpoint.com/python-project-with-source-code-profile-finder-in-github', 'https://www.javatpoint.com/python-dictionar
y', 'https://www.javatpoint.com/shuffle-in-python', 'https://www.javatpoint.com/web-scraping-using-python', 'https://ww
w.javatpoint.com/understanding-robotics-with-python', 'https://www.javatpoint.com/python-linter', 'https://www.javatpoin
t.com/python-website-blocker-building-python-script', 'https://www.javatpoint.com/python-shelve-module', 'https://www.ja
vatpoint.com/python-set', 'https://www.javatpoint.com/class-method-vs-static-method-vs-instance-method', 'https://www.ja
vatpoint.com/how-to-make-a-python-auto-clicker', 'https://www.javatpoint.com/google-search-packages-using-python', 'http
s://www.javatpoint.com/typeerror-string-indices-must-be-an-integer', 'https://www.javatpoint.com/destructors-in-python',
```

In [5]:

```
print(url_queue)
```

```
ww.javatpoint.com/strip-function-in-python', 'https://www.javatpoint.com/gradient-descent-algorithm', 'https://www.javat
point.com/prettytable-in-python', 'https://www.javatpoint.com/sentiment-analysis-in-python', 'https://www.javatpoint.co
m/convert-python-list-to-numpy-arrays', 'https://www.javatpoint.com/traceback-in-python', 'https://www.javatpoint.com/ti
me-clock-method-in-python', 'https://www.javatpoint.com/deque-in-python', 'https://www.javatpoint.com/dictionary-compreh
ension-in-python', 'https://www.javatpoint.com/python-data-analytics', 'https://www.javatpoint.com/python-seek-method',
'https://www.javatpoint.com/ternary-operator-in-python', 'https://www.javatpoint.com/how-to-calculate-area-of-circle-usi
ng-python', 'https://www.javatpoint.com/how-to-write-in-text-file-using-python', 'https://www.javatpoint.com/python-keye
rror', 'https://www.javatpoint.com/python-super-function', 'https://www.javatpoint.com/max-function-in-python', 'http
s://www.javatpoint.com/fraction-module-in-python', 'https://www.javatpoint.com/popular-python-framework-to-build-api',
'https://www.javatpoint.com/how-to-check-python-version', 'https://www.javatpoint.com/python-s-string-formatting', 'http
s://www.javatpoint.com/python-seaborn-library', 'https://www.javatpoint.com/countplot-in-python', 'https://www.javatpoin
t.com/range-vs-xrange-python', 'https://www.javatpoint.com/wordcloud-package-in-python', 'https://www.javatpoint.com/con
vert-dataframe-into-list', 'https://www.javatpoint.com/anova-test-in-python', 'https://www.javatpoint.com/python-program
-to-find-compound-interest', 'https://www.javatpoint.com/ansible-in-python', 'https://www.javatpoint.com/python-importan
t-tips-and-tricks', 'https://www.javatpoint.com/python-coroutines', 'https://www.javatpoint.com/double-underscores-in-py
thon', 'https://www.javatpoint.com/re-search-vs-re-findall-in-python-regex', 'https://www.javatpoint.com/how-to-install-
statsmodels-in-python', 'https://www.javatpoint.com/cos-in-python', 'https://www.javatpoint.com/vif-in-python', 'http
s://www.javatpoint.com/__add__-in-python', 'https://www.javatpoint.com/ethical-hacking-with-python', 'https://www.javatp
oint.com/class-variable-vs-instance', 'https://www.javatpoint.com/perfect-number-in-python', 'https://www.javatpoint.co
m/eol-in-python', 'https://www.javatpoint.com/python-program-to-convert-hexadecimal-string-to-decimal-string', 'https://
```

In [6]:

```
doc_counter=0
while(doc_counter<1000 and url_queue):

    r = requests.get(url_queue.popleft())

    soup = BeautifulSoup(r.content, 'html5lib')
    print(doc_counter)

    table = soup.find('div', attrs = {'id':'city'})
    if (not table) or(not table.find_all('a')):
        continue
    if not table.get_text():
        continue
    doc_counter+=1
    with open("raw_doc/document"+str(doc_counter)+".txt", "w", encoding="utf-8") as f:
        f.write(table.get_text())
    for link in table.find_all('a'):
        if link or "https" in link.get('href'):
            continue
        if 'https://www.javatpoint.com/'+str(link.get('href')) not in url_set:
            url_set.add('https://www.javatpoint.com/'+link.get('href'))
            url_queue.append('https://www.javatpoint.com/'+link.get('href'))
```

232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251

In [70]:

```
len(url_set)
```

Out[70]:

896

In [3]:

```
dir_path = r'C:\Users\Karthik\OneDrive\Desktop\SEM_7\Information_retrival\Project\raw_doc'
count = 0
# Iterate directory
for path in os.listdir(dir_path):
    # check if current path is a file
    if os.path.isfile(os.path.join(dir_path, path)):
        count += 1
print('File count:', count)
```

File count: 899

In [6]:

```
doc_words_deque=deque()
doc_words_list=[]
for i in range(count):
    document_data=open('C:/Users/Karthik/OneDrive/Desktop/SEM_7/Information_retrival/Project/raw_doc/document'+str(i+1)+'.txt', "r",encodi
    javatpoint_data = document_data.read()

    javatpoint_list1 = javatpoint_data.replace('\n', ' ').split(" ")
    document_data.close()
    doc_words_deque.extendleft(javatpoint_list1)
    doc_words_list.append(javatpoint_list1)
```

In [7]:

```
document_data=open('C:/Users/Karthik/OneDrive/Desktop/SEM_7/Information_retrival/Project/raw_doc/document1.txt', "r",encoding="utf8")
#     with open(r'C:/Users/Karthik/OneDrive/Desktop/SEM_7/Information_retrival/Project/raw_doc/document'+str(i+1)+'.txt', 'w') as fp:
#         # write each item on a new line
#         item=" ".join(url_set[i])
#         print(item)
#         fp.write("%s\n" % item)
javatpoint_data = document_data.read()

javatpoint_list1 = javatpoint_data.replace('\n', ' ').split(" ")

# printing the data
document_data.close()
```

In [8]:

```
print(javatpoint_list1)
```

```
lds', 'data', 'associated', 'with', 'a', 'class', 'and', 'its', 'objects.', '', 'To', 'read', 'the', 'oops', 'concept',
'in', 'detail,', 'visit', 'the', 'following', 'resources.', '', 'Python', 'Oops', 'Concepts', 'Python', 'Object', 'and',
'classes', 'Python', 'Constructor', 'Python', 'Inheritance', 'Python', 'Polymorphism', '', 'Python', 'Advance', 'Topic
s', 'Python', 'includes', 'many', 'advance', 'and', 'useful', 'concepts', 'that', 'help', 'the', 'programmer', 'to', 'so
lve', 'the', 'complex', 'tasks.', 'These', 'concepts', 'are', 'given', 'below.', 'Python', 'Iterator', 'An', 'iterator',
'is', 'simply', 'an', 'object', 'that', 'can', 'be', 'iterated', 'upon.', 'It', 'returns', 'one', 'object', 'at', 'a',
'time.', 'It', 'can', 'be', 'implemented', 'using', 'the', 'two', 'special', 'methods,', '__iter__()', 'and', '__next__
().', 'To', 'learn', 'more', 'about', 'the', 'iterators', 'visit', 'our', 'Python', 'Iterators', 'tutorial.', 'Python',
'Generators', 'The', 'Generators', 'are', 'an', 'easiest', 'way', 'of', 'creating', 'Iterators.', 'To', 'learn', 'more',
'about,', 'visit', 'our', 'Python', 'Generators', 'tutorial.', 'Python', 'Decorators', 'These', 'are', 'used', 'to', 'mo
dify', 'the', 'behavior', 'of', 'the', 'function.', 'Decorators', 'provide', 'the', 'flexibility', 'to', 'wrap', 'anothe
r', 'function', 'to', 'expand', 'the', 'working', 'of', 'wrapped', 'function,', 'without', 'permanently', 'modifying',
'it.', 'To', 'learn', 'more', 'about,', 'visit', 'the', 'Python', 'Decorators', 'tutorial.', 'Python', 'Database', 'Conn
ections', 'We', 'can', 'use', 'various', 'databases', 'along', 'with', 'Python.', 'You', 'can', 'learn', 'the', 'full',
'tutorial', 'to', 'visit', 'below', 'resources.', 'Python', 'DBI-API', 'acclaims', 'standard', 'sets', 'of', 'functional
ity', 'to', 'be', 'included', 'in', 'the', 'database', 'connectivity', 'modules', 'for', 'respective', 'RDBMS', 'product
s.', 'We', 'explain', 'all', 'important', 'database', 'connectivity', 'using', 'Python', 'DBI-API.', 'Python', 'MySQL',
'Environment', 'Setup', 'Database', 'Connection', 'Creating', 'New', 'Database', 'Creating', 'Tables', 'Insert', 'Operat
ion', 'Read', 'Operation', 'Update', 'Operation', 'Join', 'Operation', 'Performing', 'Transactions', 'Python', 'MongoD
```

In [9]:

```
document_data=open('C:/Users/Karthik/OneDrive/Desktop/SEM_7/Information_retrival/Project/raw_doc/document1.txt', "r",encoding="utf8")
asdf=document_data.read()
print(asdf)
document_data.close()
```

```
Python 2 uses print as a statement and used as print "something" to print some string on the console. On the other hand,
Python 3 uses print as a function and used as print("something") to print something on the console.
Python 2 uses the function raw_input() to accept the user's input. It returns the string representing the value, which i
s typed by the user. To convert it into the integer, we need to use the int() function in Python. On the other hand, Pyt
hon 3 uses input() function which automatically interpreted the type of input entered by the user. However, we can cast
this value to any type by using primitive functions (int(), str(), etc.).
In Python 2, the implicit string type is ASCII, whereas, in Python 3, the implicit string type is Unicode.
Python 3 doesn't contain the xrange() function of Python 2. The xrange() is the variant of range() function which return
s a xrange object that works similar to Java iterator. The range() returns a list for example the function range(0,3) co
ntains 0, 1, 2.
There is also a small change made in Exception handling in Python 3. It defines a keyword as which is necessary to be us
ed. We will discuss it in Exception handling section of Python programming tutorial.


Java vs Python Program
Unlike the other programming languages, Python provides the facility to execute the code using few lines. For example -
Suppose we want to print the "Hello World" program in Java; it will take three lines to print it.
Java Program

public class HelloWorld {
```

In [10]:

```
print(len(doc_words_deque))
```

```
1650270
```

In [11]:

```
print(len(doc_words_list))
```

```
899
```

In [12]:

```
import operator
dic={}
while(doc_words_deque):
    i=doc_words_deque.popleft()
    if ((i in dic) and len(i)<4):
        dic[i]=dic[i]+1
    else:
        dic[i]=1
dic=dict( sorted(dic.items(), key=operator.itemgetter(1),reverse=True))
```

In [13]:

```
stopwords_top_3500=list(dic)[:3500]
print(stopwords_top_3500)
```

```
'SE', '9', '7', '=', '143', 'V6', 'V5', 'V4', 'V1', '491', '(36', 'LED', 'Oil', 'WT1', '(V5', 'SF1', 'P1g', '4)]',
'="', 'Q1,', 'TD', 'Q(S', '~x', 'TIt', '{b}', '56]', 's4:', 'FOr', '350', 'TK', 'HST', 'BST', '-03', 'tz', 'ico', '729',
'((3', '(3', '<=,', '(16', 'dwn', '-5.', 'c3)', 'c3', 'c2)', 'c1)', 'Txb', '[[3', "'R'", '5->', 'ws', '187', '746', '11
7', '118', '186', '184', 'cp', '"to', 'DK', 'VB', 'LS', 'is"', 'PNL', 'ND', '((', '-3.', 't):', '$")', 'y'", 'tts', 'b_
1', 'm_q', 'p)', '?0', 'yi', '….,', 'N:', "'!", '21:', '17:', "p'", "'q", 'id:', 'n2,', '(n1', '[[_', "C's", '-t', 'GT
X', 'D:\\', 'C:\\', '%")', '360', '874', 'pb2', '<h3', 'sp', '"P"', '2.2', '207', '"B"', '<39', '113', '725', 'gsl', '#I
n', 'O/P', '4x', "'?'", 'REs', 'ab+', 'foo', '"Is', 'TWO', '14', '-X', '`--', '-3]', "to'", 'DML', '901', '891', '.ps',
'q]', '[p,', 's_0', 'io,', 'Ex.', 'ep:', '(f"', 's_r', '/.', '5.2', 'buf', 'n*n', 'Pop', 're.', 'i,k', 'b>a', '56.',
'5!', '4!', '3!', '2!', '1]`', '31:', 'D3', '3/1', 'CLS', '676', '*)', '")"', '}}', '4.x', '5.x', ':-', 'ke', "'no", 'DB
M', 'z0,', 'XY', '8})', '(w,', '"G"', '%u"', '%i,', '\\w', '\\a', 'W2', '(vi', '78)', '4.2', '+:', '#)', 'N', 'µ', 'X', '
f…', 'H,', 'Pt', 'ch', "'W'", "'IP", 'yx', '(g', 'x1:', 'x_2', '(q', '((p', '+31', '+15', '+9', '[[1', 'x),', 'l:', '52
4', 'pgr', 'SQS', 'w_n', 'EHR', 'TI,', 'CT,', 'IN,', 'EED', 'b2,', 'Won', '2#', '1#', '340', '339', '338', '337', '567',
'"Up', '"R"', '()"', 'ret', '"&"', 'VPN', 'ts,', '`👍`', '"An', 'MIT', '6a', '52,', 'hsv', '"y', '6)}', 'AS', 'd1:', '5
0}', 'sst', 'Y_1', 'ys,', ';\t', 'fib', '&A[', 'A[', 'a[]', '225', '254', 'MG', 'ray', 'JPG', 'PT', 'q1', 'q3', '0.7',
'abs', 'so;', 'ids', '210', '765', '540', 'Sr', 'ar', 'it)', '[+]', '270', '|,', '=[', '(By', '7th', 'n<2', '15%', '7%',
'71)', 'CS3', 'CS1', 'dic', '@,', '1,6', '@@', 'CA', '49:', '320', 'ZZ', 'AA', 'ab', 'MIB', 'pow', 'x+n', '182', '523',
'f.', 'f))', 'e))', '(d', 'B)', "'".", 'M_1', 'MPG', '-6,', 'ESP', "'7'", 'b=', 'dll', '323', 'X_2', 'LR', 'KV', 'C2',
'14}', 'atr', '%"', 'by,', 'fl', 'J:', 'O(K', '[2.', '1/6', 'H_1', 'N_1', '[20', '4kb', 'CBV', '"I"', '64]', '124', '2
5}', 'co', '44]', 't1,', 'sr', "'it", "'],", "'6'", 'ii)', 'vif', 'ing', 'V,', '(>', 'typ', '159', "-V'", '22:', '{q:',
'74]', '0g', "'$'", '$$$', 'lt', 'p*q', 'p+q', 'R):', '"%Y', '09', 'NO', '[]}', 'com', '%Y"', '"%H', 'O),', 'Tie', 'mc',
```

In [14]:

```
print(len(dic))
```

```
102673
```

In [15]:

```
nostop_word_doc_words_list=[]
for i in doc_words_list:
    temp=[]
    for j in i:
        if j not in stopwords_top_3500:
            temp.append(j)
    nostop_word_doc_words_list.append(temp)
```

In [16]:

```
for i,j in enumerate(nostop_word_doc_words_list):
    with open("nostop_doc/document"+str(i)+".txt", "w", encoding="utf-8") as f:
        f.write(" ".join(j))
```

In [19]:

```
def stemming(content):

    stem = PorterStemmer()

    content = re.sub('[^A-Za-z]+', ' ', content)
    tokens = nltk.word_tokenize(content)

    stem_tokens = [stem.stem(word) for word in tokens]
    clean_stem_tokens = ' '.join(map(str,  stem_tokens))

    shortword = re.compile(r'\W*\b\w{1,2}\b')
    clean_stem_tokens = shortword.sub('', clean_stem_tokens)


    return clean_stem_tokens
```

In [23]:

```
from nltk.stem import PorterStemmer
import re
import nltk
for i in range(count):
    with open('nostop_doc/document'+str(i)+'.txt' , 'r', encoding="utf-8") as f:
        content = f.readlines()
        content=" ".join(content)
        new_content = stemming(content)
        with open('stemmed_doc/document'+str(i)+'.txt', 'w' ,encoding="utf-8") as f1:
            f1.write(new_content)
        f1.close()
    f.close()
```

In [24]:

```
words = []
for i in range(count):
    with open('stemmed_doc/document'+str(i)+'.txt' , 'r',encoding="utf-8") as f:
        content = f.read()
        word_tokens = nltk.word_tokenize(content)
        fdist_mis = nltk.FreqDist(word_tokens)
    for i in fdist_mis:
        words.append(i)
```

In [25]:

```
print(words)
```

```
e', 'harvard', 'biggest', 'corpor', 'were', 'shown', 'astronom', 'contemporari', 'telescop', 'ideal', 'close', 'earth',
'upcom', 'done', 'ellipt', 'apollo', 'centaur', 'jupyt', 'journal', 'ipython', 'kdnugget', 'contrast', 'presidenti', 'ca
mpaign', 'seri', 'figur', 'notebook', 'regular', 'sometim', 'opinion', 'ascertain', 'feel', 'celebr', 'polit', 'parti',
'categor', 'unfavor', 'neutral', 'sure', 'perfect', 'talent', 'social', 'media', 'polar', 'keyword', 'emot', 'apach', 'n
ifi', 'gettwitt', 'ingest', 'queue', 'hashtag', 'azur', 'term', 'audienc', 'plu', 'junki', 'alexandr', 'bhole', 'playsto
r', 'medium', 'profil', 'miss', 'textblob', 'and', 'worldwid', 'prime', 'candid', 'challeng', 'incorrect', 'split', 'us
a', 'greatest', 'million', 'full', 'indic', 'nlu', 'gram', 'embed', 'vector', 'microarray', 'svm', 'implement', 'sciki
t', 'showcas', 'get', 'inaccur', 'damag', 'insuffici', 'unreli', 'messi', 'vacat', 'compani', 'stay', 'throughout', 'cit
i', 'territori', 'distribut', 'metric', 'draw', 'window', 'cultur', 'zeitgeist', 'worth', 'titl', 'channel', 'publish',
'tag', 'synopsi', 'could', 'clip', 'categori', 'influenc', 'potenti', 'feder', 'disabl', 'taken', 'analys', 'assemblag',
'dept', 'demograph', 'suppos', 'career', 'preprocess', 'deepspeech', 'transpar', 'spoken', 'synthesi', 'apiai', 'speechr
ecognit', 'anderson', 'crowdsourc', 'summari', 'toolbox', 'nltk', 'anyon', 'dialogu', 'enter', 'increas', 'obtain', 'res
olv', 'masteri', 'total', 'calor', 'expenditur', 'distanc', 'walk', 'run', 'heart', 'pace', 'roll', 'averag', 'count',
'role', 'senior', 'explicit', 'output', 'fresh', 'danger', 'deploy', 'movi', 'movielen', 'consist', 'phase', 'chemic',
'viscos', 'acid', 'below', 'random', 'forest', 'descent', 'stochast', 'svc', 'numpi', 'array', 'wherea', 'kind', 'acquis
it', 'final', 'matplotlib', 'collabor', 'commod', 'establish', 'actor', 'factor', 'subset', 'firm', 'advis', 'devic', 't
ype', 'all', 'tailor', 'page', 'think', 'piqu', 'audio', 'engag', 'leav', 'alon', 'context', 'leader', 'failur', 'noti
c', 'horror', 'ceremoni', 'apostl', 'consid', 'can', 'not', 'expressli', 'comedi', 'seem', 'highli', 'ambigu', 'bought',
'widget', 'transform', 'journey', 'offici', 'began', 'tremend', 'titan', 'commerc', 'account', 'sale', 'likelihood', 'be
com', 'repeat', 'purchas', 'uniqu', 'past', 'impress', 'stride', 'app', 'convert', 'abubakar', 'newspap', 'virtual', 'st
```

In [26]:

```
print(len(words))
```

```
218392
```

In [27]:

```
def create_vector_index(path):
    inverted_index = {}
    dir = os.listdir(path)
    no_docs = len(dir)
    for d in dir:
        id = d.split('.')[0][8:]
        print(id)
        f = open(path + d, 'r')
        content = f.read()
        f.close()
        print(content)
        tokens = nltk.word_tokenize(content)
        for token in tokens:
            if token not in inverted_index:
                inverted_index[token] = []
            if id not in inverted_index[token]:
                inverted_index[token].append(int(id))
    return inverted_index
```

In [28]:

```
index = create_vector_index('stemmed_doc/')
for i in sorted(index):
    index[i].sort()
#    print(i, index[i])
#print(index['spheric'])
print(sorted(index))
index={k : index[k] for k in sorted(index)}
f = open('inverted_index.txt', 'w')
f.write(json.dumps(index))
f.close()
```

```
r while count print python loop loop control statement will discuss loop control statement detail will exampl each contr
ol statement continu statement return control begin loop code python program show continu statement work initi loop stri
ng python loop string string string continu print current letter string output current letter current letter current let
ter current letter current letter current letter current letter break statement stop execut loop when break statement re
ach code python program show break statement work initi loop string python loop string break print current letter string
output current letter current letter current letter current letter current letter current letter current letter pass sta
tement pass statement use creat empti loop pass statement also employ class function empti control statement code python
program show pass statement work string python loop pass print last letter string output last letter next topicpython lo
op prev next
130
next prev heap sort python heap sort quit same select sort where find maximum element end base comparison sort algorithm
which work binari heap data structur best exampl effici sort algorithm what heap sort heap sort effici popular sort algo
rithm heap sort concept elimin element from heap part list one one insert them sort part list befor learn more about hea
p sort algorithm let discuss heap data structur place algorithm which mean fix amount memori use store sort list memori
size doesn reli size preliminari list exampl don need addit memori stack store sort array neither recurs call stack heap
sort algorithm usual implement use second array sort fix valu thi process quick simpl natur easi implement other hand he
ap sort unstabl which mean doesn maintain compar order element with equal valu quickli sort primit type such integ chara
ct problem with complex type object let understand follow exampl have custom class student with properti name sever obje
ct that class array includ student call thoma age also peter have age appear same order sort array peopl age then there
```

In [30]:

```
import json
f = open('inverted_index.txt', 'w')
f.write(json.dumps(index))
f.close()
```

In [31]:

```
file1 = open('inverted_index.txt', "r")

# reading the file
data = file1.read()

print(data)
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

In [32]:

```
from nltk.stem import PorterStemmer
ps = PorterStemmer()
corpus=[]
for i,j in enumerate(nostop_word_doc_words_list):
    # using map to stem words in word list
    words = map(lambda w: ps.stem(str(w)), j)
    # joining words into string and adding to corpus list
    corpus.append(' '.join(words))
```

In [33]:

```
print(len(corpus))
```

```
899
```

In [37]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity


vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(corpus)
```

In [38]:

```
print(tfidf_matrix)
```

```
  (0, 31120)    0.021949518263430425
  (0, 14521)    0.006969068412335304
  (0, 10031)    0.014277468126271875
  (0, 24167)    0.011016947629450666
  (0, 23910)    0.011107761826455034
  (0, 20490)    0.015342149369077514
  (0, 14118)    0.0062994787807782015
  (0, 6900)     0.01516567504688197
  (0, 24576)    0.023571825998836192
  (0, 6991)     0.01516567504688197
  (0, 18365)    0.01088554651799765
  (0, 24370)    0.012549676708854911
  (0, 8807)     0.019809453998988265
  (0, 27686)    0.014030339459914252
  (0, 16495)    0.011448457050241327
  (0, 13536)    0.01220240542272846
  (0, 17431)    0.01744823132477639
  (0, 27347)    0.013913817703049901
  (0, 10745)    0.009182356506454457
  (0, 33341)    0.0180816625815713
  (0, 13316)    0.014277468126271875
  (0, 17432)    0.01088554651799765
  (0, 14954)    0.053008169203638315
  (0, 28805)    0.03145717115825129
  (0, 20662)    0.03396088584591543
  :     :
  (898, 7528)   0.005566631238113155
  (898, 7568)   0.02006857991654252
  (898, 19031)  0.21916732563794117
  (898, 12452)  0.005531126005819735
  (898, 14686)  0.05163163081727603
  (898, 32716)  0.01428862023631287
  (898, 32789)  0.006367711422028706
  (898, 32455)  0.006908923555761128
  (898, 6635)   0.010852416772577181
  (898, 14811)  0.008226161632835578
  (898, 20901)  0.007067073462759756
  (898, 21415)  0.012004827233573353
  (898, 19616)  0.005399484997200918
  (898, 33438)  0.003922862904049398
  (898, 10906)  0.005032595025144993
  (898, 28160)  0.0068940879001404415
  (898, 15413)  0.02716818473875534
  (898, 30553)  0.005195062437699316
  (898, 30986)  0.011170204647259095
  (898, 24697)  0.011170204647259095
  (898, 15017)  0.013260960575283311
  (898, 7367)   0.014569209112971456
  (898, 24673)  0.49788275953492483
  (898, 25123)  0.42050875012590216
  (898, 21730)  0.010226118818686146
```

In [39]:

```
cos_sim_matrix = cosine_similarity(tfidf_matrix)
#print('\nPrinting cosine similarity matrix: ')
#print('\t Doc1 Doc2 Doc3 Doc4')
csres = []
for idx, row in enumerate(cos_sim_matrix):
    csres.append(str(row))
```

In [42]:

```
import csv
with open('cosine_simi/csres.txt', 'w') as f:
    for line in csres:
        f.write(f"{line}\n")
```

In [ ]:

```

```

In [ ]:

```

```

In [36]:

```
pip install scikit-learn
```

```
Collecting scikit-learn
  Downloading scikit_learn-1.2.0-cp39-cp39-win_amd64.whl (8.3 MB)
     ---------------------------------------- 8.3/8.3 MB 8.2 MB/s eta 0:00:00
Requirement already satisfied: joblib>=1.1.1 in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (from scikit-lear
n) (1.2.0)
Collecting scipy>=1.3.2
  Using cached scipy-1.9.3-cp39-cp39-win_amd64.whl (40.2 MB)
Requirement already satisfied: numpy>=1.17.3 in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (from scikit-lear
n) (1.23.4)
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-3.1.0-py3-none-any.whl (14 kB)
Installing collected packages: threadpoolctl, scipy, scikit-learn
Successfully installed scikit-learn-1.2.0 scipy-1.9.3 threadpoolctl-3.1.0
Note: you may need to restart the kernel to use updated packages.
```

In [81]:

```
pip install sklearn
```

```
Collecting sklearn
  Downloading sklearn-0.0.post1.tar.gz (3.6 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Building wheels for collected packages: sklearn
  Building wheel for sklearn (setup.py): started
  Building wheel for sklearn (setup.py): finished with status 'done'
  Created wheel for sklearn: filename=sklearn-0.0.post1-py3-none-any.whl size=2936 sha256=540849fdc291df417b903fe356f6f14fe
2f90626c0c72cad4184d36b6c5e9cb1
  Stored in directory: c:\users\karthik\appdata\local\pip\cache\wheels\f8\e0\3d\9d0c2020c44a519b9f02ab4fa6d2a4a996c98d79ab2
f569fa1
Successfully built sklearn
Installing collected packages: sklearn
Successfully installed sklearn-0.0.post1
Note: you may need to restart the kernel to use updated packages.
```

In [82]:

```
pip install tkinter
```

```
Note: you may need to restart the kernel to use updated packages.

ERROR: Could not find a version that satisfies the requirement tkinter (from versions: none)
ERROR: No matching distribution found for tkinter
```

In [83]:

```
from tkinter import *
parent = Tk()
name = Label(parent,text = "Name").grid(row = 0, column = 0)
e1 = Entry(parent).grid(row = 0, column = 1)
password = Label(parent,text = "Password").grid(row = 1, column = 0)
e2 = Entry(parent).grid(row = 1, column = 1)
submit = Button(parent, text = "Submit").grid(row = 4, column = 0)
parent.mainloop()
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Input In [83], in <cell line: 8>()
      6 e2 = Entry(parent).grid(row = 1, column = 1)
      7 submit = Button(parent, text = "Submit").grid(row = 4, column = 0)
----> 8 parent.mainloop()

File ~\anaconda3\envs\IR_LAB\lib\tkinter\__init__.py:1429, in Misc.mainloop(self, n)
   1427 def mainloop(self, n=0):
   1428     """Call the mainloop of Tk."""
-> 1429     self.tk.mainloop(n)

KeyboardInterrupt:
```

In [ ]:

```
print(e1,e2)
```

In [ ]:

```
import tkinter as tk
from functools import partial


def call_result(label_result, n1, n2):
    num1 = (n1.get())
    num2 = (n2.get())
    result = int(num1)+int(num2)
    label_result.config(text="Result = %d" % result)
    return

root = tk.Tk()
root.geometry('400x200+100+200')

root.title('Calculator')

number1 = tk.StringVar()
number2 = tk.StringVar()

labelNum1 = tk.Label(root, text="A").grid(row=1, column=0)

labelNum2 = tk.Label(root, text="B").grid(row=2, column=0)

labelResult = tk.Label(root)

labelResult.grid(row=7, column=2)

entryNum1 = tk.Entry(root, textvariable=number1).grid(row=1, column=2)

entryNum2 = tk.Entry(root, textvariable=number2).grid(row=2, column=2)

call_result = partial(call_result, labelResult, number1, number2)

buttonCal = tk.Button(root, text="Calculate", command=call_result).grid(row=3, column=0)

root.mainloop()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [27]:

```
r = requests.get(url_queue.popleft())

soup = BeautifulSoup(r.content, 'html5lib')

quotes=[] # a list to store quotes

table = soup.find('div', attrs = {'id':'city'})
print(table.get_text())
```

```
The built-in classes define many magic methods. The dir() function can be used to see the number of magic methods inherit
ted by a class. It has two prefixes and suffix underscores in the method name.

Python Magic Methods

Python Oops Concepts
Everything in Python is treated as an object including integer values, floats, functions, classes, and none. Apart from
that, Python supports all oriented concepts. Below is the brief introduction of oops concepts of Python.

Classes and Objects - Python classes are the blueprint of the object. An object is a collection of data and method that
act on the data.
Inheritance - An inheritance is a technique where one class inherits the properties of other classes.
Constructor - Python provides a special method __init__() which is known as a constructor. This method is automatically
called when an object is instantiated.
Data Member - A variable that holds data associated with a class and its objects.

To read the oops concept in detail, visit the following resources.

Python Oops Concepts
Python Object and classes
```

In [32]:

```
with open('trail1', "w", encoding="utf-8") as f:
    f.write(table.get_text())
```

In [38]:

```
for link in table.find_all('a'):
    print(link.get('href'))
```

```
python-lists
python-numbers
python-string-capitalize-method
python-string-casefold-method
python-string-center-method
python-string-count-method
python-string-encode-method
python-string-endswith-method
python-string-expandtabs-method
python-string-find-method
python-string-format-method
python-string-index-method
python-string-isalnum-method
python-string-isalpha-method
python-string-isdecimal-method
python-string-isdigit-method
python-string-isidentifier-method
None
python-string-islower-method
python-string-isnumeric-method
python-string-isprintable-method
python-string-isupper-method
python-string-isspace-method
python-string-istitle-method
python-string-isupper-method
python-string-join-method
python-string-ljust-method
python-string-lower-method
python-string-lstrip-method
python-string-partition-method
python-string-replace-method
python-string-rfind-method
python-string-rindex-method
python-string-rjust-method
python-string-rstrip-method
python-string-rsplit-method
python-string-split-method
python-string-splitlines-method
python-string-startswith-method
python-string-swapcase-method
python-string-translate-method
python-string-upper-method
python-string-zfill-method
python-string-rpartition-method
python-lists
python-numbers
python-lists
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [26]:

```
pip install requests
pip install html5lib
pip install bs4
```

```
  Input In [26]
    pip install requests
        ^
SyntaxError: invalid syntax
```

In [27]:

```
pip install requests
```

```
Requirement already satisfied: requests in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (2.28.1)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (from r
equests) (2.1.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (from request
s) (2022.6.15)
Requirement already satisfied: idna<4,>=2.5 in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (from requests) (3.
4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (from requ
ests) (1.26.13)
Note: you may need to restart the kernel to use updated packages.
```

In [28]:

```
pip install html5lib
```

```
Collecting html5lib
  Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
     ---------------------------------- 112.2/112.2 kB 815.5 kB/s eta 0:00:00
Requirement already satisfied: webencodings in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (from html5lib) (0.
5.1)
Requirement already satisfied: six>=1.9 in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (from html5lib) (1.16.
0)
Installing collected packages: html5lib
Successfully installed html5lib-1.1
Note: you may need to restart the kernel to use updated packages.
```

In [29]:

```
pip install bs4
```

```
Collecting bs4
  Downloading bs4-0.0.1.tar.gz (1.1 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: beautifulsoup4 in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (from bs4) (4.11.
1)
Requirement already satisfied: soupsieve>1.2 in c:\users\karthik\anaconda3\envs\ir_lab\lib\site-packages (from beautifulsou
p4->bs4) (2.3.1)
Building wheels for collected packages: bs4
  Building wheel for bs4 (setup.py): started
  Building wheel for bs4 (setup.py): finished with status 'done'
  Created wheel for bs4: filename=bs4-0.0.1-py3-none-any.whl size=1257 sha256=f61380aa81562e22c10b6cacdb146ba6f2aa9a8d26aa8
fb5e05b16a583d4a440
  Stored in directory: c:\users\karthik\appdata\local\pip\cache\wheels\73\2b\cb\099980278a0c9a3e57ff1a89875ec07bfa0b6fcbebb
9a8cad3
Successfully built bs4
Installing collected packages: bs4
Successfully installed bs4-0.0.1
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [7]:

```
import os
import numpy as np
import re
import nltk
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
import json
```

In [ ]:

```
data = open("", "r")

# reading the file
cran_data = cran_all_1400.read()

cran_list1 = cran_data.replace('\n', ' ').split(" ")

# printing the data
cran_all_1400.close()

cran_list2=cran_list1.copy()
stop_char_list = ['.', '.', '.A', '.B', '.T', '(', ')', 'j','.','.W','','j.']
for i in cran_list2:
    if i in stop_char_list:
        cran_list1.remove(i)
print(cran_list1)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: