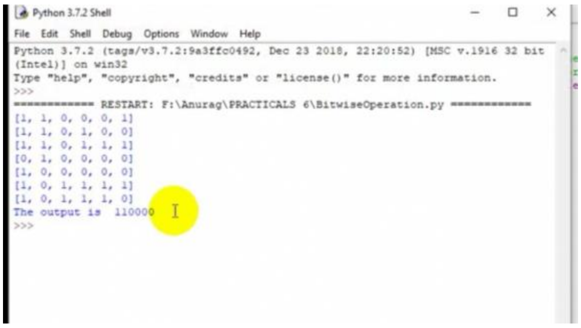
PRACTICAL NO:-01

AIM: WRITE A PROGRAM TO DEMONSTRATE BITWISE OPERATION.

CODE:

```
plays={"Anthony and Cleopatra":"Anthony is there, Brutus is Caeser is with Cleopatra mercy worser.",
    "Julius Ceaser":"Anthony is there, Brutus is Caeser is but Calpurnia is.",
    "The Tempest":"mercy worser","Hamlet":"Caeser and Brutus are present with mercy and worser",
    "Othello":"Caeser is present with mercy and worser","Macbeth":"Anthony is there, Caeser, mercy."}
words=["Anthony","Brutus","Caeser","Calpurnia","Cleopatra","mercy","worser"]
vector_matrix=[[0 for i in range(len(plays))] for j in range(len(words))]

text_list=list((plays.values()))

for i in range(len(words)):
  for j in range(len(text_list)):
    if words[i] in text_list[j]:
      vector_matrix[i][j]=1
    else:
      vector_matrix[i][j]=0

for i in vector_matrix:
  print(i)
result=[]

string_list=[]
for vector in vector_matrix:
  mystring = ""
  for digit in vector:
    mystring += str(digit)
  string_list.append(int(mystring,2))
#print(string_list)



print("The output is ",bin(string_list[0]&string_list[1]&(string_list[2])).replace("0b",""))
```

OUTPUT :



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: F:\Anurag\PRACTICALS 6\BitwiseOperation.py ============
[1, 1, 0, 0, 0, 1]
[1, 1, 0, 1, 0, 0]
[1, 1, 0, 1, 1, 1]
[0, 1, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0]
[1, 0, 1, 1, 1, 1]
[1, 0, 1, 1, 1, 0]
The output is  110000
>>>
```

PRACTICAL NO:-02

AIM: IMPLEMENT PAGE RANK ALGORITHM.

CODE:

```
import numpy as np

import scipy as sc

import pandas as pd

from fractions import Fraction

   def display_format(my_vector, my_decimal):

      return np.round((my_vector).astype(np.float),
decimals=my_decimal)

      my_dp = Fraction(1,1)

      Mat = np.matrix([[0,0,1],

      [Fraction(1,2),0,0],
      [Fraction(1,2),1,0]])

      Ex = np.zeros((3,3))

      Ex[:] = my_dp

      Damp = 0.7

      Al = Damp * Mat + ((1-Damp) * Ex)

      r = np.matrix([my_dp, my_dp, my_dp])

      r = np.transpose(r)

      previous_r = r

   for i in range(1,10):

      r = Al * r

      print (display_format(r,3))

if (previous_r==r).all():

   break

previous_r = r

print ("Final:\n", display_format(r,3))

print ("sum", np.sum(r))
```

Output

```
[[0.333]

[0.217]

[0.45 ]]

[[0.415]

[0.217]

[0.368]]

[[0.358]

[0.245]
[0.397]]
[[0.378]
[0.225]
[0.397]]
[[0.378]
[0.232]
[0.39 ]]
[[0.373]
[0.232]
[0.395]]
[[0.376]
[0.231]
[0.393]]
[[0.375]
[0.232]
[0.393]]
[[0.375]
[0.231]
[0.394]]
[[0.375]
[0.231]
[0.393]]
[[0.375]
[0.231]
[0.393]]
```

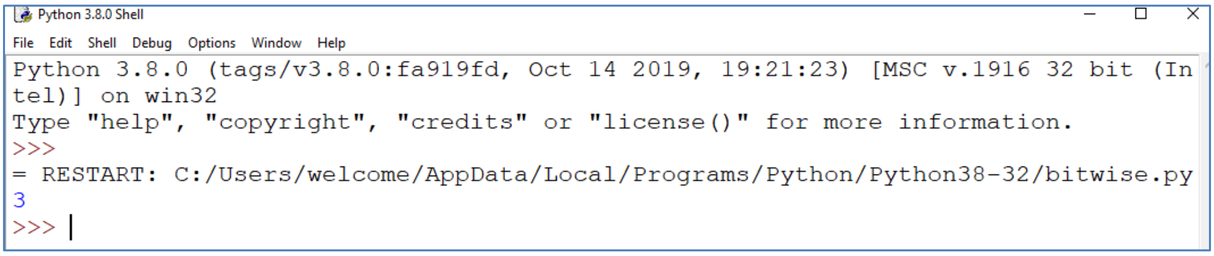
PRACTICAL NO:-03

AIM: IMPLEMENT DYNAMIC PROGRAMMING ALGORITHM FOR COMPUTING THE EDIT DISTANCE BETWEEN STRINGS S1 AND S2. (HINT. LEVENSHTEIN DISTANCE)

CODE:

```
def editDistance(str1, str2, m, n):
        if m == 0:
                return n
        if n == 0:
                return m
        if str1[m-1]== str2[n-1]:
                return editDistance(str1, str2, m-1, n-1)
        return 1 + min(editDistance(str1, str2, m, n-1), # Insert
                                editDistance(str1, str2, m-1, n), # Remove
                                editDistance(str1, str2, m-1, n-1) # Replace   )
str1 = "sunday"
str2 = "saturday"
print (editDistance(str1, str2, len(str1), len(str2)) )
```

OUTPUT:



```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/welcome/AppData/Local/Programs/Python/Python38-32/bitwise.py
3
>>> |
```

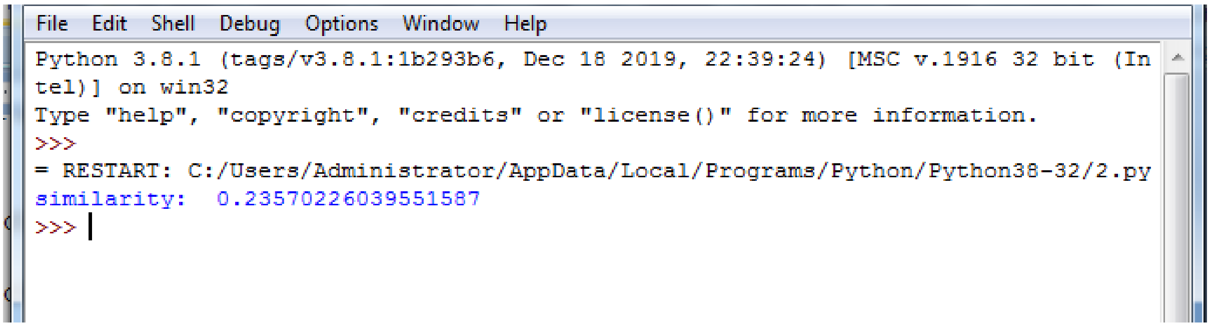
PRACTICAL NO:-04

AIM: WRITE A PROGRAM TO COMPUTE SIMILARITY BETWEEN TWO TEXT DOCUMENTS.

CODE:

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
# X = input("Enter first string: ").lower()
# Y = input("Enter second string: ").lower()
X =open('file1.txt','r').read()
Y =open('file2.txt','r').read()
# tokenization
X_list = word_tokenize(X)
Y_list = word_tokenize(Y)
# sw contains the list of stopwords
sw = stopwords.words('english')
l1 =[];l2 =[]
# remove stop words from string
X_set = {w for w in X_list if not w in sw}
Y_set = {w for w in Y_list if not w in sw}
# form a set containing keywords of both strings
rvector = X_set.union(Y_set)
for w in rvector:
        if w in X_set: l1.append(1) # create a vector
        else: l1.append(0)
        if w in Y_set: l2.append(1)
        else: l2.append(0)
c = 0
# cosine formula
for i in range(len(rvector)):
                c+= l1[i]*l2[i]
cosine = c / float((sum(l1)*sum(l2))**0.5)
print("similarity: ", cosine)
```

OUTPUT:



```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Administrator/AppData/Local/Programs/Python/Python38-32/2.py
similarity:  0.23570226039551587
>>> 
```

PRACTICAL NO:-05

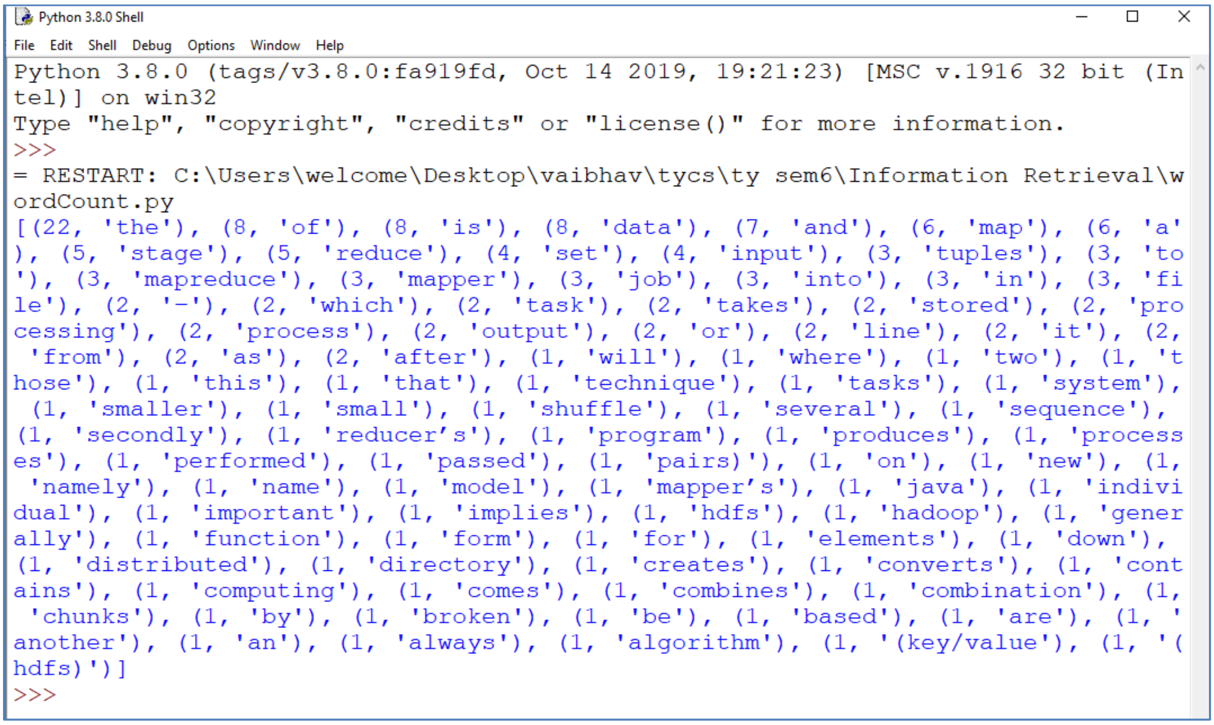
AIM: WRITE A MAP-REDUCE PROGRAM TO COUNT THE NUMBER OF OCCURRENCES OF EACH ALPHABETIC CHARACTER IN THE GIVEN DATASET. THE COUNT FOR EACH LETTER SHOULD BE CASE-INSENSITIVE (I.E., INCLUDE BOTH UPPER-CASE AND LOWER-CASE VERSIONS OF THE LETTER; IGNORE NON-ALPHABETIC CHARACTERS).

CODE:

```
Text="""MapReduce is a processing technique and a program model for distributed
computing based on java. The MapReduce algorithm contains two important tasks, namely
Map and Reduce. Map takes a set of data and converts it into another set of data, where
individual elements are broken down into tuples (key/value pairs). Secondly, reduce task,
which takes the output from a map as an input and combines those data tuples into a smaller
set of tuples. As the sequence of the name MapReduce implies, the reduce task is always
performed after the map job.
   Map stage − The map or mapper’s job is to process the input
data. Generally the input data is in the form of file or directory
and is stored in the Hadoop file system (HDFS). The input file is
passed to the mapper function line by line. The mapper processes
the data and creates several small chunks of data.
   Reduce stage − This stage is the combination of the Shuffle
stage and the Reduce stage. The Reducer’s job is to process the
data that comes from the mapper. After processing, it produces a
new set of output, which will be stored in the HDFS.
"""
# Cleaning text and lower casing all words
for char in '-.,\n':
   Text=Text.replace(char,' ')
Text = Text.lower()# split returns a list of words delimited by sequences of whitespace
(including tabs, newlines, etc, like re's \s)
```

```
word_list = Text.split()
from collections import Counter
Counter(word_list).most_common()
# Initializing Dictionary
d = {}
# counting number of times each word comes up in list of words (in dictionary)
for word in word_list:
    d[word] = d.get(word, 0) + 1
#reverse the key and values so they can be sorted using tuples.
word_freq = []
for key, value in d.items():
    word_freq.append((value, key))
word_freq.sort(reverse=True)
print(word_freq)
```

OUTPUT:



```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
= RESTART: C:\Users\welcome\Desktop\vaibhav\tycs\ty sem6\Information Retrieval\w
ordCount.py
[(22, 'the'), (8, 'of'), (8, 'is'), (8, 'data'), (7, 'and'), (6, 'map'), (6, 'a'
), (5, 'stage'), (5, 'reduce'), (4, 'set'), (4, 'input'), (3, 'tuples'), (3, 'to
'), (3, 'mapreduce'), (3, 'mapper'), (3, 'job'), (3, 'into'), (3, 'in'), (3, 'fi
le'), (2, '-'), (2, 'which'), (2, 'task'), (2, 'takes'), (2, 'stored'), (2, 'pro
cessing'), (2, 'process'), (2, 'output'), (2, 'or'), (2, 'line'), (2, 'it'), (2,
 'from'), (2, 'as'), (2, 'after'), (1, 'will'), (1, 'where'), (1, 'two'), (1, 't
hose'), (1, 'this'), (1, 'that'), (1, 'technique'), (1, 'tasks'), (1, 'system'),
 (1, 'smaller'), (1, 'small'), (1, 'shuffle'), (1, 'several'), (1, 'sequence'),
(1, 'secondly'), (1, 'reducer's'), (1, 'program'), (1, 'produces'), (1, 'process
es'), (1, 'performed'), (1, 'passed'), (1, 'pairs)'), (1, 'on'), (1, 'new'), (1,
 'namely'), (1, 'name'), (1, 'model'), (1, 'mapper's'), (1, 'java'), (1, 'indivi
dual'), (1, 'important'), (1, 'implies'), (1, 'hdfs'), (1, 'hadoop'), (1, 'gener
ally'), (1, 'function'), (1, 'form'), (1, 'for'), (1, 'elements'), (1, 'down'),
(1, 'distributed'), (1, 'directory'), (1, 'creates'), (1, 'converts'), (1, 'cont
ains'), (1, 'computing'), (1, 'comes'), (1, 'combines'), (1, 'combination'), (1,
 'chunks'), (1, 'by'), (1, 'broken'), (1, 'be'), (1, 'based'), (1, 'are'), (1, '
another'), (1, 'an'), (1, 'always'), (1, 'algorithm'), (1, '(key/value'), (1, '(
hdfs)')]
>>> 
```

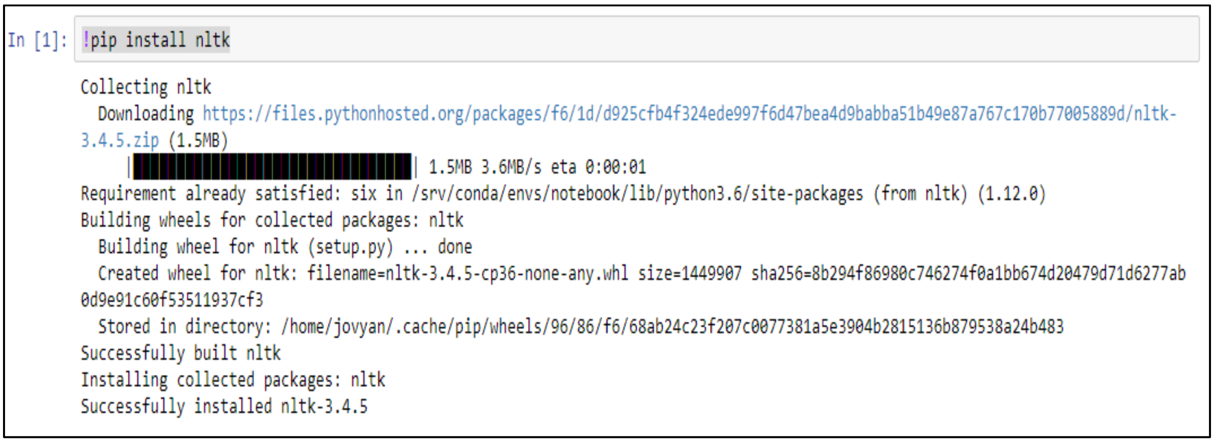
PRACTICAL NO:-06

AIM: WRITE A PROGRAM FOR PRE-PROCESSING OF A TEXT DOCUMENT: STOP WORD REMOVAL.

CODE:

1.Install nltk

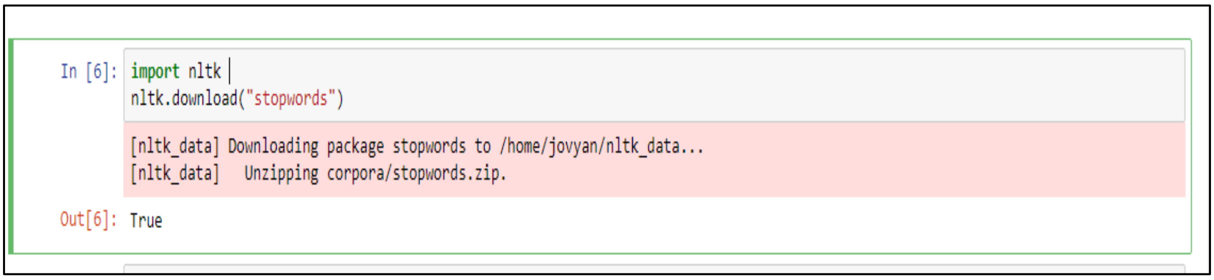
```
!pip install nltk
```



```
In [1]: !pip install nltk

        Collecting nltk
          Downloading https://files.pythonhosted.org/packages/f6/1d/d925cfb4f324ede997f6d47bea4d9babba51b49e87a767c170b77005889d/nltk-
        3.4.5.zip (1.5MB)
             |████████████████████████████████| 1.5MB 3.6MB/s eta 0:00:01
        Requirement already satisfied: six in /srv/conda/envs/notebook/lib/python3.6/site-packages (from nltk) (1.12.0)
        Building wheels for collected packages: nltk
          Building wheel for nltk (setup.py) ... done
          Created wheel for nltk: filename=nltk-3.4.5-cp36-none-any.whl size=1449907 sha256=8b294f86980c746274f0a1bb674d20479d71d6277ab
        0d9e91c60f53511937cf3
          Stored in directory: /home/jovyan/.cache/pip/wheels/96/86/f6/68ab24c23f207c0077381a5e3904b2815136b879538a24b483
        Successfully built nltk
        Installing collected packages: nltk
        Successfully installed nltk-3.4.5
```

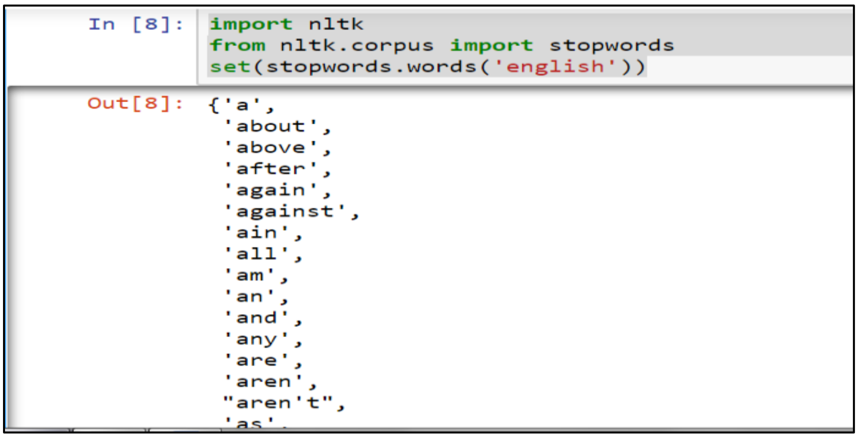
2. download stopwords in nltk

```
import nltk
nltk.download("stopwords")
```



```
In [6]: import nltk
        nltk.download("stopwords")

        [nltk_data] Downloading package stopwords to /home/jovyan/nltk_data...
        [nltk_data]   Unzipping corpora/stopwords.zip.

Out[6]: True
```

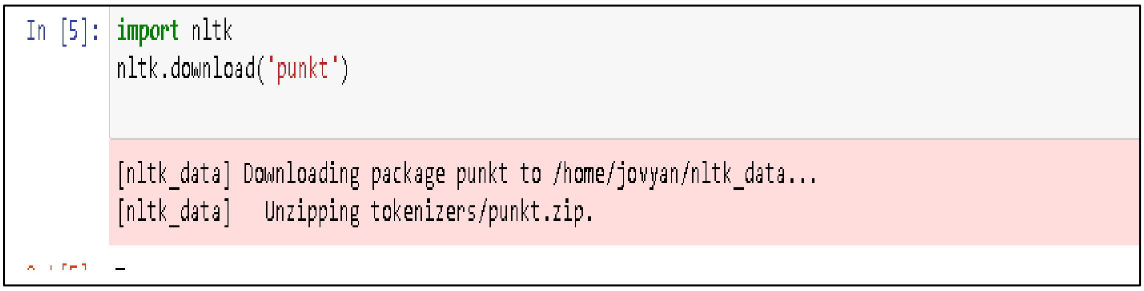
import nltk
from nltk.corpus import stopwords
set(stopwords.words('english'))



```
In [8]: import nltk
        from nltk.corpus import stopwords
        set(stopwords.words('english'))

Out[8]: {'a',
          'about',
          'above',
          'after',
          'again',
          'against',
          'ain',
          'all',
          'am',
          'an',
          'and',
          'any',
          'are',
          'aren',
          "aren't",
          'as',
```

now download punkt in nltk

import nltk
nltk.download('punkt')



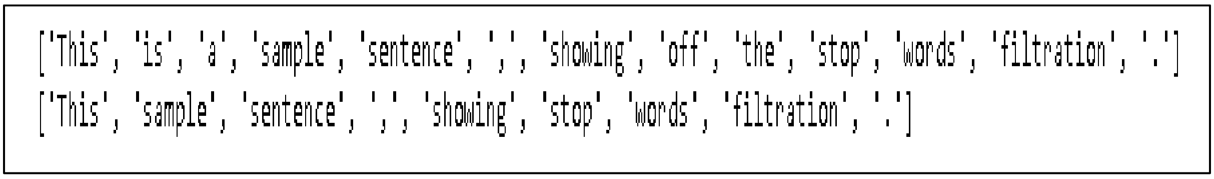
```
In [5]: import nltk
        nltk.download('punkt')

        [nltk_data] Downloading package punkt to /home/jovyan/nltk_data...
        [nltk_data]   Unzipping tokenizers/punkt.zip.
```

4. Stopwords coding

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
example_sent="This is a sample sentence,showing off the stop words filtration."
stop_words=set(stopwords.words('english'))
word_tokens=word_tokenize(example_sent)
filtered_sentence=[w for w in word_tokens if not w in stop_words]
filtered_sentence=[]
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)
```

```
print(word_tokens)
print(filtered_sentence)
```

OUTPUT:



```
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']
```

PRACTICAL NO:-07

AIM: WRITE A PROGRAM TO IMPLEMENT SIMPLE WEB CRAWLER.
A Web Crawler is a program that navigates the Web and finds new or updated pages for indexing. The Crawler starts with seed websites or a wide range of popular URLs (also known as the frontier) and searches in depth and width for hyperlinks to extract.

A Web Crawler must be kind and robust. Kindness for a Crawler means that it respects the rules set by the robots.txt and avoids visiting a website too often. Robustness refers to the ability to avoid spider traps and other malicious behavior. Other good attributes for a Web Crawler is distributivity amongst multiple distributed machines, expandability, continuity and ability to prioritize based on page quality.
Steps to create web crawler
The basic steps to write a Web Crawler are:

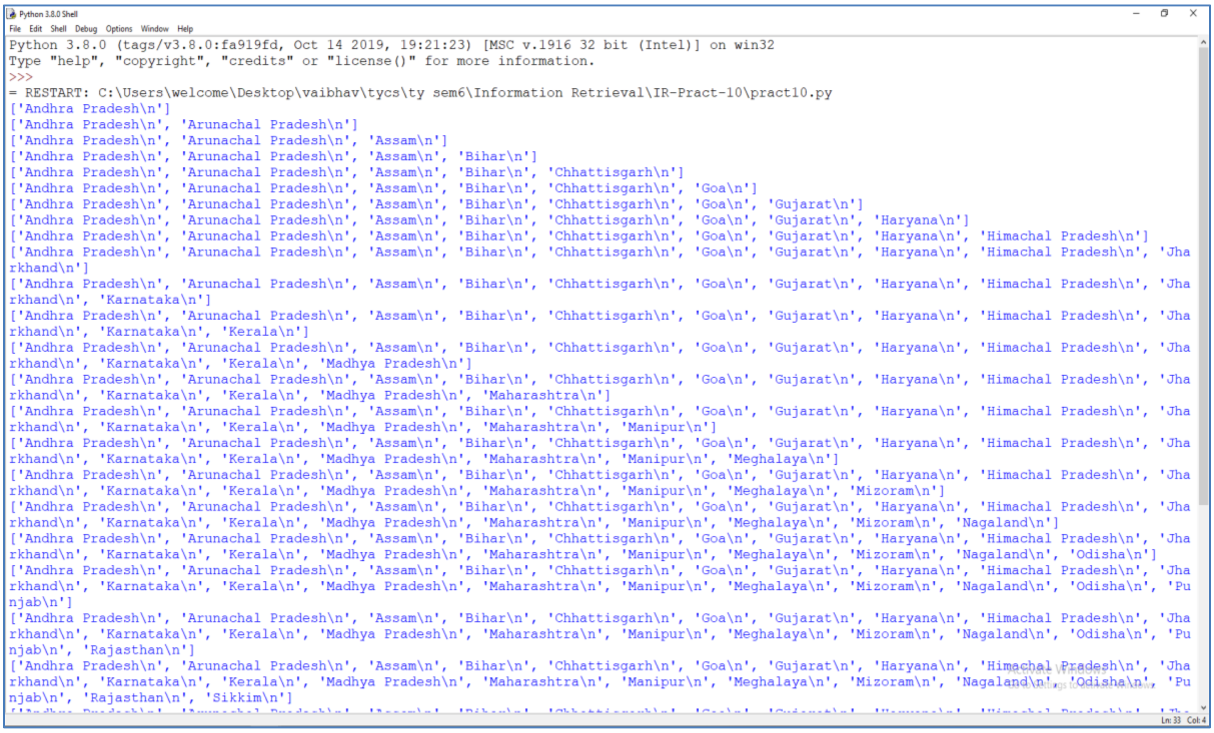
- Pick a URL from the frontier
- Fetch the HTML code
- Parse the HTML to extract links to other URLs
- Check if you have already crawled the URLs and/or if you have seen the same content before

If not add it to the index
For each extracted URL
Confirm that it agrees to be checked (robots.txt, crawling frequency)

CODE:

```
import requests
from bs4 import BeautifulSoup
URL = "https://en.wikipedia.org/wiki/States_and_union_territories_of_India"
res = requests.get(URL).text
soup = BeautifulSoup(res,'lxml')
states=[]
for items in soup.find('table', class_='wikitable').find_all('tr')[1::1]:
    data = items.find_all(['th','td'])
    #print(data[0].text)
    states.append(data[0].text)
    print(states)
```

OUTPUT:



```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\welcome\Desktop\vaibhav\tycs\ty sem6\Information Retrieval\IR-Pract-10\pract10.py
['Andhra Pradesh\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n', 'Karnataka\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n', 'Karnataka\n', 'Kerala\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n', 'Karnataka\n', 'Kerala\n', 'Madhya Pradesh\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n', 'Karnataka\n', 'Kerala\n', 'Madhya Pradesh\n', 'Maharashtra\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n', 'Karnataka\n', 'Kerala\n', 'Madhya Pradesh\n', 'Maharashtra\n', 'Manipur\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n', 'Karnataka\n', 'Kerala\n', 'Madhya Pradesh\n', 'Maharashtra\n', 'Manipur\n', 'Meghalaya\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n', 'Karnataka\n', 'Kerala\n', 'Madhya Pradesh\n', 'Maharashtra\n', 'Manipur\n', 'Meghalaya\n', 'Mizoram\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n', 'Karnataka\n', 'Kerala\n', 'Madhya Pradesh\n', 'Maharashtra\n', 'Manipur\n', 'Meghalaya\n', 'Mizoram\n', 'Nagaland\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n', 'Karnataka\n', 'Kerala\n', 'Madhya Pradesh\n', 'Maharashtra\n', 'Manipur\n', 'Meghalaya\n', 'Mizoram\n', 'Nagaland\n', 'Odisha\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n', 'Karnataka\n', 'Kerala\n', 'Madhya Pradesh\n', 'Maharashtra\n', 'Manipur\n', 'Meghalaya\n', 'Mizoram\n', 'Nagaland\n', 'Odisha\n', 'Pu
njab\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n', 'Karnataka\n', 'Kerala\n', 'Madhya Pradesh\n', 'Maharashtra\n', 'Manipur\n', 'Meghalaya\n', 'Mizoram\n', 'Nagaland\n', 'Odisha\n', 'Pu
njab\n', 'Rajasthan\n']
['Andhra Pradesh\n', 'Arunachal Pradesh\n', 'Assam\n', 'Bihar\n', 'Chhattisgarh\n', 'Goa\n', 'Gujarat\n', 'Haryana\n', 'Himachal Pradesh\n', 'Jha
rkhand\n', 'Karnataka\n', 'Kerala\n', 'Madhya Pradesh\n', 'Maharashtra\n', 'Manipur\n', 'Meghalaya\n', 'Mizoram\n', 'Nagaland\n', 'Odisha\n', 'Pu
njab\n', 'Rajasthan\n', 'Sikkim\n']
```

PRACTICAL NO:-08

AIM: WRITE A PROGRAM TO PARSE XML TEXT, GENERATE WEB GRAPH AND COMPUTE TOPIC SPECIFIC PAGE RANK.

CODE:

Xml file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<root testAttr="testValue">
The Tree
<children>
<child name="Jack">First</child>
```

```
<child name="Rose">Second</child>
<child name="Blue Ivy">
Third
<grandchildren>
<data>One</data>
<data>Two</data>
<unique>Twins</unique>
</grandchildren>
</child>
<child name="Jane">Fourth</child>
</children>
</root>
```

```
import xml.etree.ElementTree as ET
tree = ET.parse('items.xml')
root = tree.getroot()
# all items data
print('Expertise Data:')
for elem in root:
   for subelem in elem:
      print(subelem.text)
```

OUTPUT:

```
Expertise Data:

Expertise Data:

First

Second

Third
```