EXPERIMENT NO. 1

Student Name and Roll Number: RAHUL DHAWAN (17CSU146)
Semester /Section: VIII-C
Date: 30 January 2021
Faculty Signature:
Grade:

Objectives:

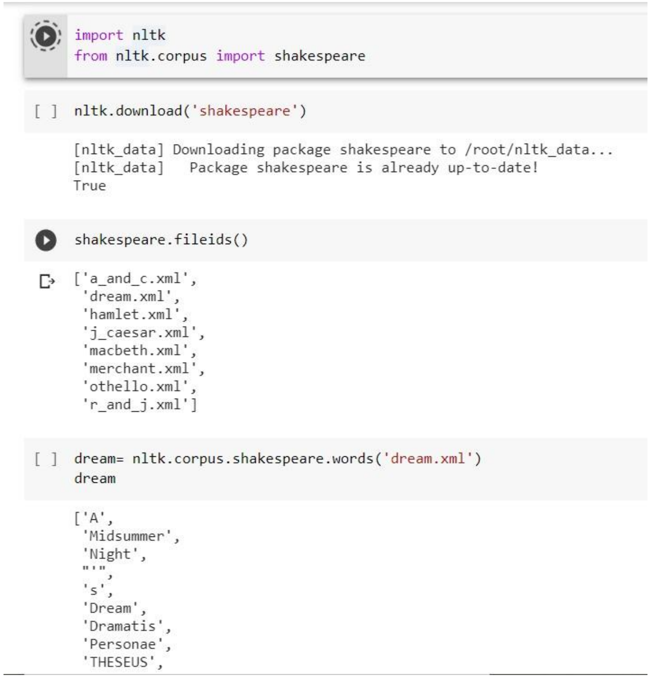
1. Import the corpus Shakespeare and find the frequency of each word in the file dream.xml.
2. Find 5 most frequently occurring words from the file dream.xml.
3. Import wordnet corpus from the available nltk corpus list and find out the sysnset of word bank. Also find the definition and example of first sysnset in the list.

Outcome: Students will be able to learn the concepts of nltk.freqdist(), collections in python and sysnets in wordnet library.

Problem Statement:

1. Import the corpus Shakespeare and find the frequency of each word in the file dream.xml.

CODE AND OUTPUT:



```
import nltk
from nltk.corpus import shakespeare
```

```
[ ]  nltk.download('shakespeare')

     [nltk_data] Downloading package shakespeare to /root/nltk_data...
     [nltk_data]   Package shakespeare is already up-to-date!
     True
```

```
shakespeare.fileids()

['a_and_c.xml',
 'dream.xml',
 'hamlet.xml',
 'j_caesar.xml',
 'macbeth.xml',
 'merchant.xml',
 'othello.xml',
 'r_and_j.xml']
```

```
[ ]  dream= nltk.corpus.shakespeare.words('dream.xml')
     dream

     ['A',
      'Midsummer',
      'Night',
      "'"
         ,
      's',
      'Dream',
      'Dramatis',
      'Personae',
      'THESEUS',
[ ]  len(dream)
```

```
21538
```

```
[ ] #cleaning data of punctuations
     import string
     l=string.punctuation.split()
     no_punct_dream=[words *or words in dream if words not in string.punctuations
     no_punct_dream
```

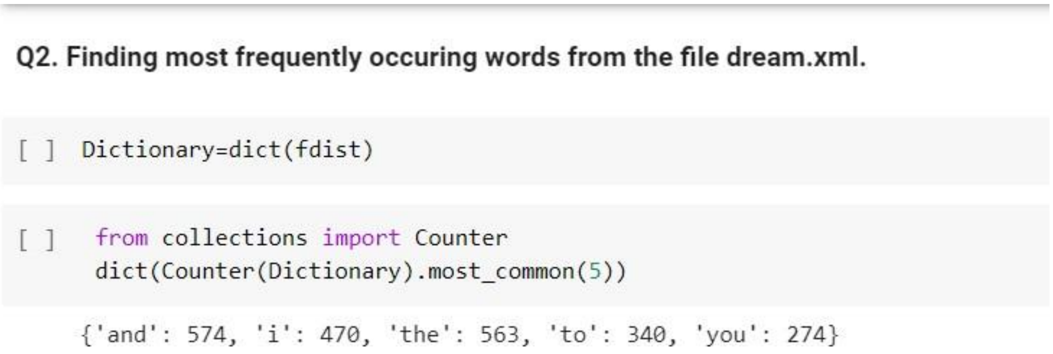
```
 'Midsummer',
 'Night',

 'Dreaxi',
 'Dr amatis,'
 ' Personae,'
 'THESEUR,'
 'Duke,'
 'of',
 'Athens',
 'EGEUS',
 '*ather',

 'Hermia',
 'LYSANDER',
 'DEMETRIUS',
 'in',
 'love',
 'with',
 'Hermia',
 'PHILOSTRATE',
 'master',
 'of',
```

```
[ ]  #Finding frequency
     fdist=nltk.FreqDist(w.lower() for w in no punct_dream)
     fdist
```

```
     FreqDist(('a' : 273,
              'midsummer': 2,
              'night': 52,
              's': 133,
              'dream': 16,
              'dramatis': 1,
              'personae': 1,
              'theseus': 67,
              'duke': 14,
              'of': 272,
              'athens' : 27,
              'egeus': 17,
              'father': 14,
              'to': 3d0,
              'hermia' : 103,
              'lysander': 103,
              'demetrius': 101,
              'in': 2d3,
              'love': 117,
              'with' : 177,
              'philostrate': ld,
              'master': 8,
              'the': 563,
              'revels': 5,
              'quince': 55,
              'carpenter': 1,
              'snug': 10,
              'joiner': d,
              'bottom': 69,
              'weaver': 3,
              'flute': 19,
              'bellows': 3,
```

2. Find 5 most frequently occurring words from the file dream.xml.

CODE AND OUTPUT:



Q2. Finding most frequently occuring words from the file dream.xml.

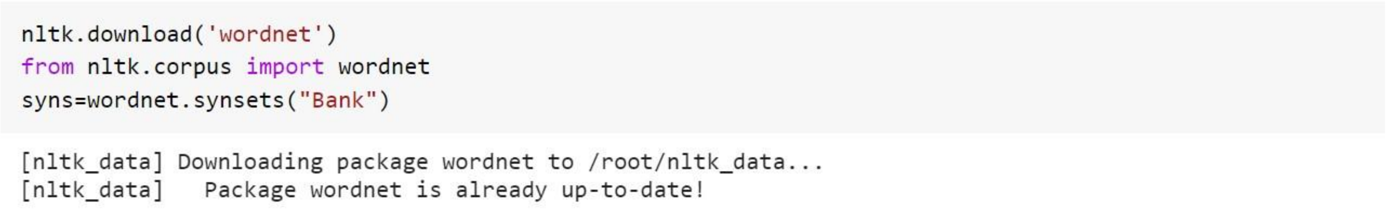
```
[ ]  Dictionary=dict(fdist)
```

```
[ ]   from collections import Counter
      dict(Counter(Dictionary).most_common(5))
```

```
{'and': 574, 'i': 470, 'the': 563, 'to': 340, 'you': 274}
```

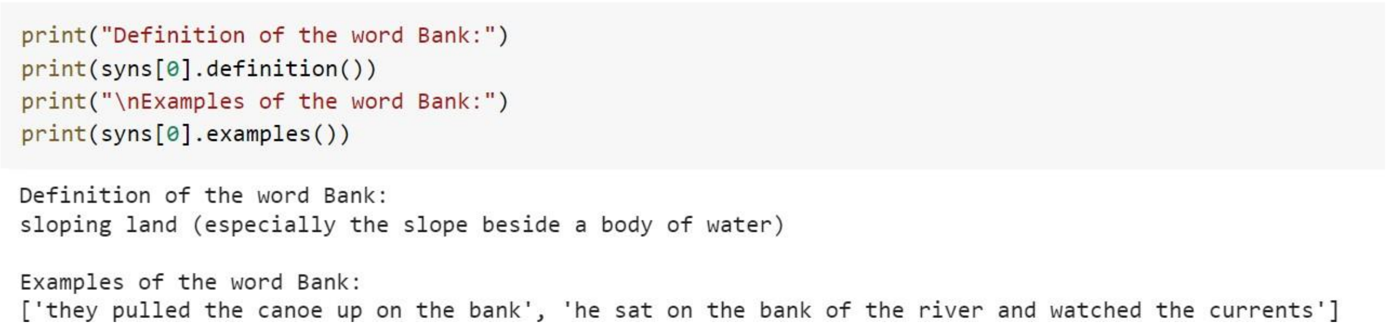
3. Import wordnet corpus from the available nltk corpus list and find out the sysnset of word bank. Also find the definition and example of first sysnset in the list.

CODE AND OUTPUT:



```
nltk.download('wordnet')
from nltk.corpus import wordnet
syns=wordnet.synsets("Bank")
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```



```
print("Definition of the word Bank:")
print(syns[0].definition())
print("\nExamples of the word Bank:")
print(syns[0].examples())
```

```
Definition of the word Bank:
sloping land (especially the slope beside a body of water)

Examples of the word Bank:
['they pulled the canoe up on the bank', 'he sat on the bank of the river and watched the currents']
```

EXPERIMENT NO. 2

Student Name and Roll Number: RAHUL DHAWAN (17CSU146)
Semester /Section: VIII-C
Date: 6 February 2021
Faculty Signature:
Grade:

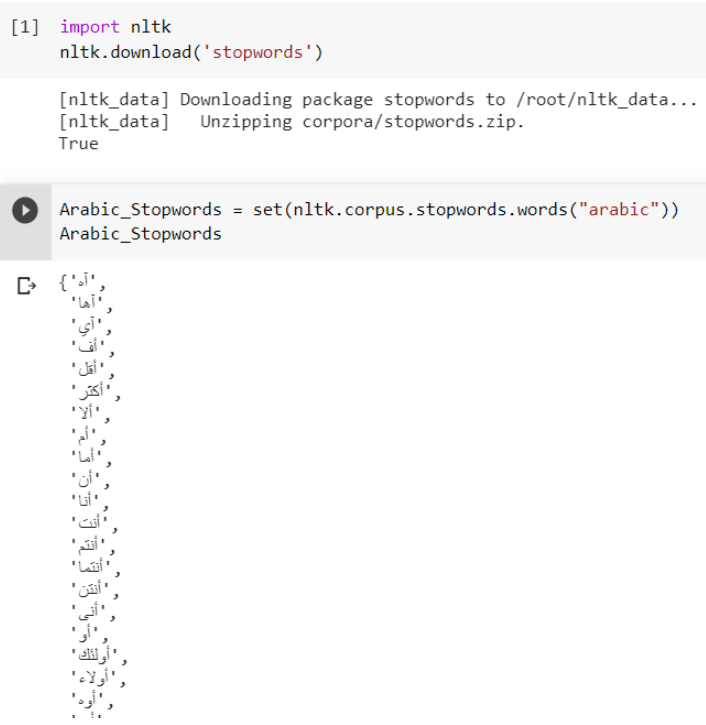
Objective:
1. Print all the Arabic Stopwords.
2. Omit a given list of stop words from the total stopwords list of English language.

Outcome: Students will be able to understand the concept of stopwords in nltk and list comprehensions in python.

Problem Statement:

1. Print all the Arabic Stopwords.

CODE AND OUTPUT:



```
[1] import nltk
    nltk.download('stopwords')
```

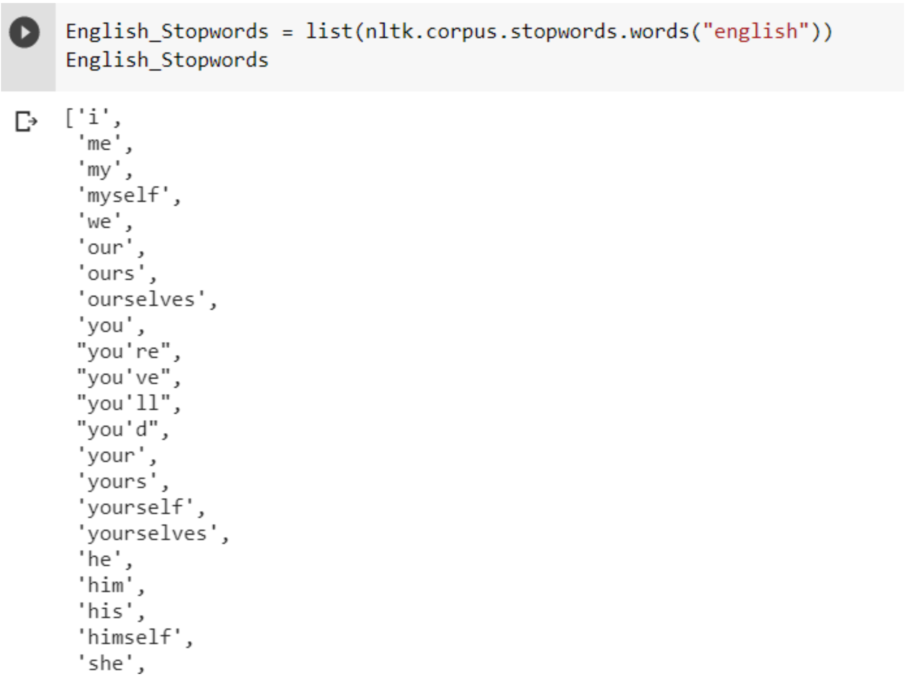
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
Arabic_Stopwords = set(nltk.corpus.stopwords.words("arabic"))
Arabic_Stopwords
```

```
{'آه',
 'آها',
 'آي',
 'أف',
 'أقل',
 'أكثر',
 'ألا',
 'أم',
 'أما',
 'أن',
 'أنا',
 'أنت',
 'أنتم',
 'أنتما',
 'أنتن',
 'أنى',
 'أو',
 'أولئك',
 'أولاء',
 'أوه',
```

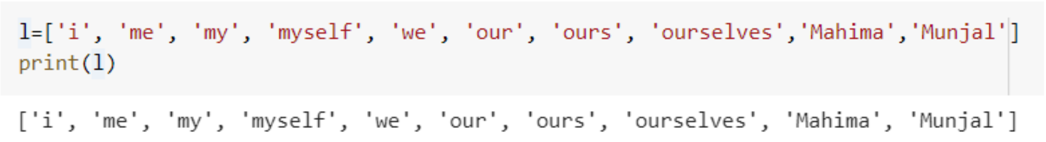
2. Omit a given list of stop words from the total stopwords list of English language.

CODE AND OUTPUT:



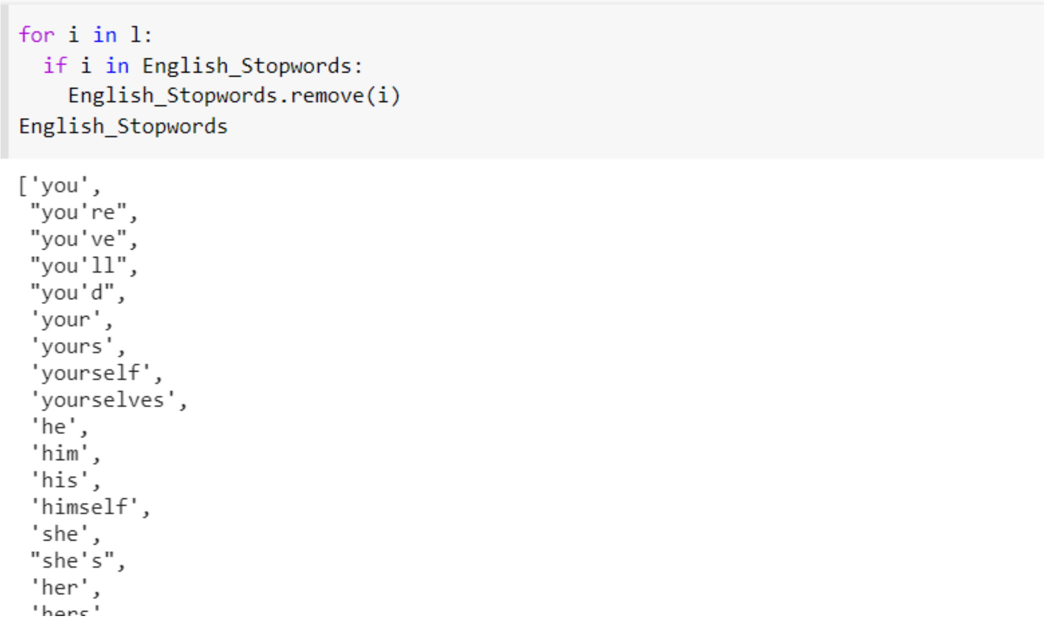
```
English_Stopwords = list(nltk.corpus.stopwords.words("english"))
English_Stopwords
```

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
```



```
l=['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves','Mahima','Munjal']
print(l)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'Mahima', 'Munjal']
```



```
for i in l:
  if i in English_Stopwords:
    English_Stopwords.remove(i)
English_Stopwords
```

```
['you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers'
```

EXPERIMENT NO. 3

Student Name and Roll Number: RAHUL DHAWAN (17CSU146)
Semester /Section: VIII-C
Date: 13 February 2021
Faculty Signature:
Grade:

Objectives:

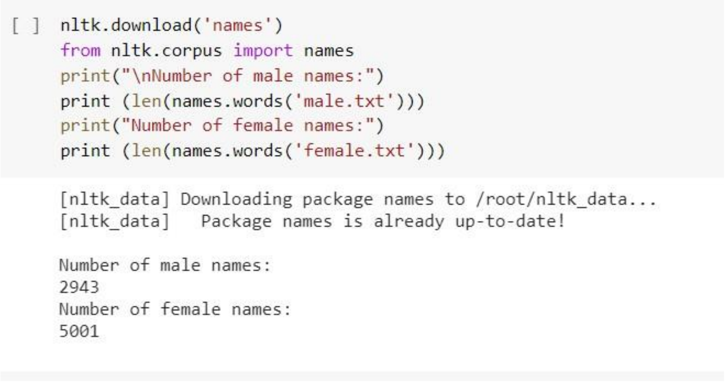
1. Print the total number of male and female names in the names corpus. Then, Print the first 15 male and female names.
2. From the names corpus, combine all the labelled male and female names and print any 20.
3. Print the definition and examples of any one English language word using WordNet corpus.

Outcome: Students will be able to explore names corpus, understand the concept of labelling the data and learn about sysnets in Wordnet corpus.

Problem Statement:

1. Print the total number of male and female names in the names corpus. Then, Print the first 15 male and female names.

CODE AND OUTPUT:

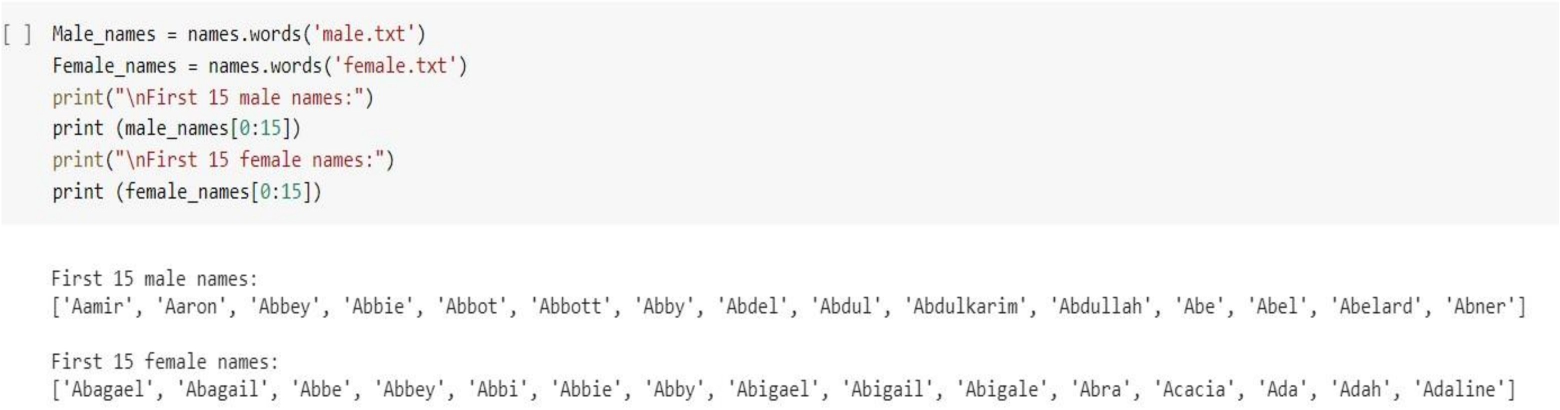


```
[ ]  nltk.download('names')
     from nltk.corpus import names
     print("\nNumber of male names:")
     print (len(names.words('male.txt')))
     print("Number of female names:")
     print (len(names.words('female.txt')))
```

```
[nltk_data] Downloading package names to /root/nltk_data...
[nltk_data]   Package names is already up-to-date!

Number of male names:
2943
Number of female names:
5001
```

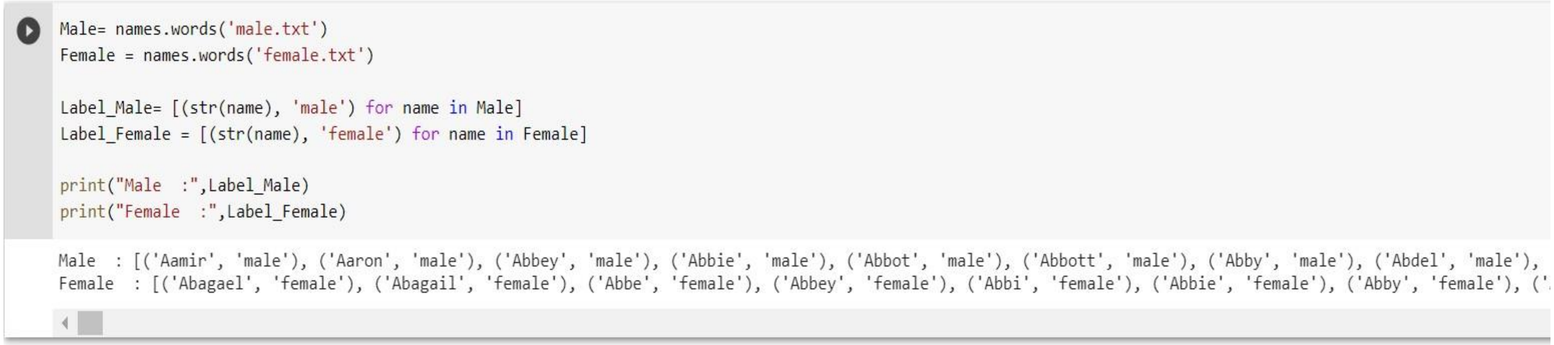
2. From the names corpus, combine all the labelled male and female names and print any 20.

CODE AND OUTPUT:



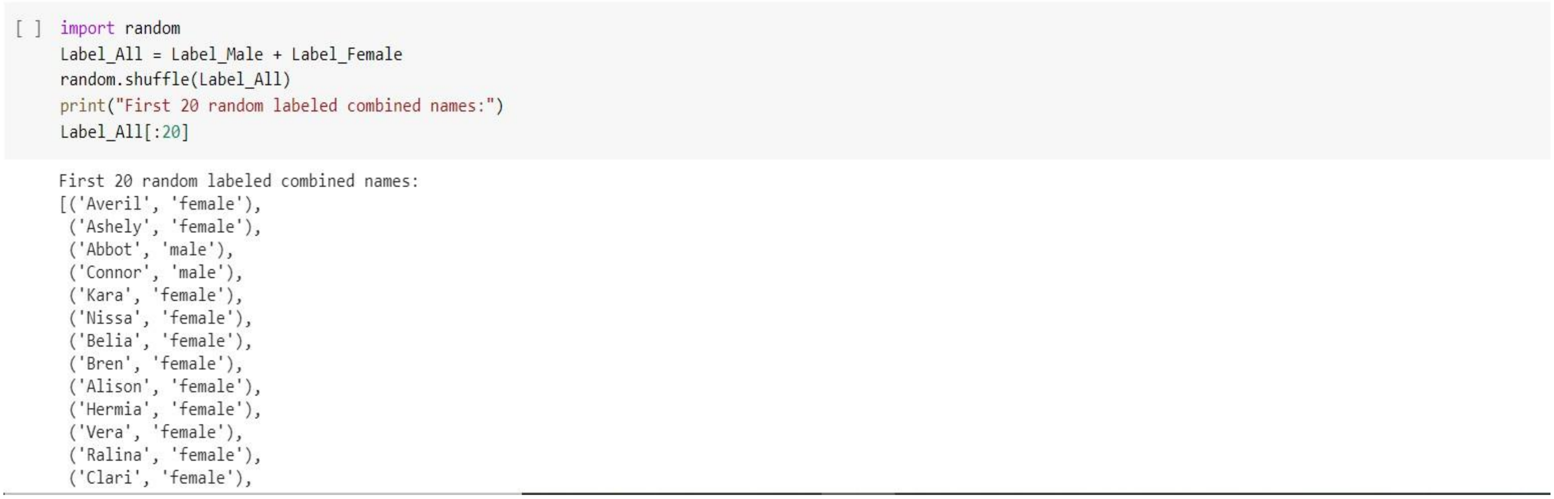
```
[ ]  Male_names = names.words('male.txt')
     Female_names = names.words('female.txt')
     print("\nFirst 15 male names:")
     print (male_names[0:15])
     print("\nFirst 15 female names:")
     print (female_names[0:15])
```

```
First 15 male names:
['Aamir', 'Aaron', 'Abbey', 'Abbie', 'Abbot', 'Abbott', 'Abby', 'Abdel', 'Abdul', 'Abdulkarim', 'Abdullah', 'Abe', 'Abel', 'Abelard', 'Abner']

First 15 female names:
['Abagael', 'Abagail', 'Abbe', 'Abbey', 'Abbi', 'Abbie', 'Abby', 'Abigael', 'Abigail', 'Abigale', 'Abra', 'Acacia', 'Ada', 'Adah', 'Adaline']
```



```
Male= names.words('male.txt')
Female = names.words('female.txt')

Label_Male= [(str(name), 'male') for name in Male]
Label_Female = [(str(name), 'female') for name in Female]

print("Male  :",Label_Male)
print("Female  :",Label_Female)
```

```
Male  : [('Aamir', 'male'), ('Aaron', 'male'), ('Abbey', 'male'), ('Abbie', 'male'), ('Abbot', 'male'), ('Abbott', 'male'), ('Abby', 'male'), ('Abdel', 'male'),
Female  : [('Abagael', 'female'), ('Abagail', 'female'), ('Abbe', 'female'), ('Abbey', 'female'), ('Abbi', 'female'), ('Abbie', 'female'), ('Abby', 'female'), ('
```

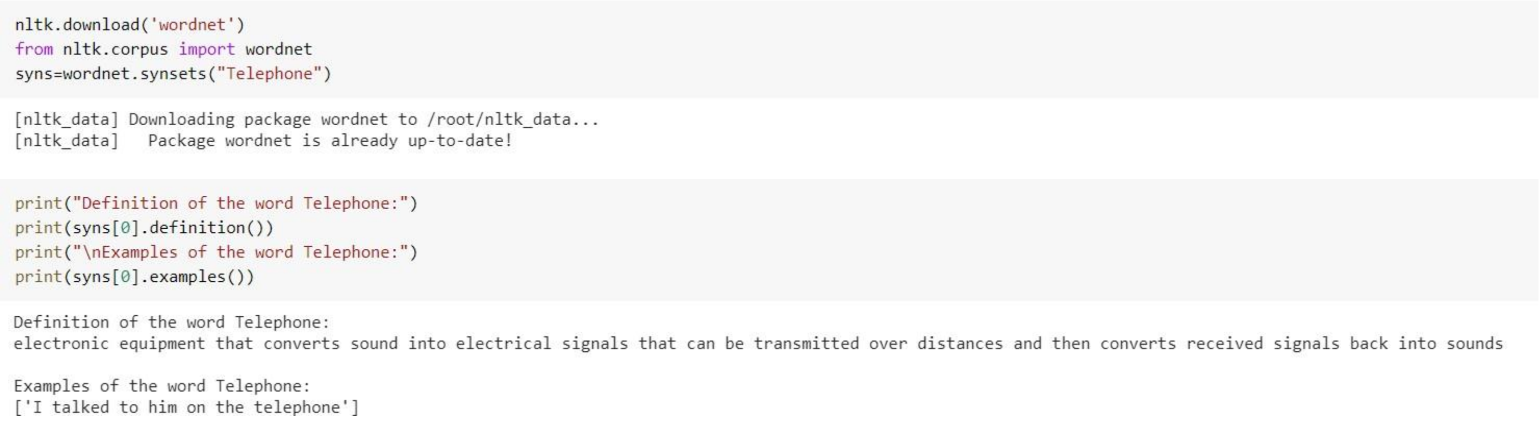


```
[ ]  import random
     Label_All = Label_Male + Label_Female
     random.shuffle(Label_All)
     print("First 20 random labeled combined names:")
     Label_All[:20]
```

```
First 20 random labeled combined names:
[('Averil', 'female'),
 ('Ashely', 'female'),
 ('Abbot', 'male'),
 ('Connor', 'male'),
 ('Kara', 'female'),
 ('Nissa', 'female'),
 ('Belia', 'female'),
 ('Bren', 'female'),
 ('Alison', 'female'),
 ('Hermia', 'female'),
 ('Vera', 'female'),
 ('Ralina', 'female'),
 ('Clari', 'female'),
```

3. Print the definition and examples of any one English language word using WordNet corpus.

CODE AND OUTPUT:



```
nltk.download('wordnet')
from nltk.corpus import wordnet
syns=wordnet.synsets("Telephone")
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
print("Definition of the word Telephone:")
print(syns[0].definition())
print("\nExamples of the word Telephone:")
print(syns[0].examples())
```

```
Definition of the word Telephone:
electronic equipment that converts sound into electrical signals that can be transmitted over distances and then converts received signals back into sounds

Examples of the word Telephone:
['I talked to him on the telephone']
```

EXPERIMENT NO. 4

Student Name and Roll Number: RAHUL DHAWAN (17CSU146)
Semester /Section: VIII-C
Date: 27 February 2021
Faculty Signature:
Grade:

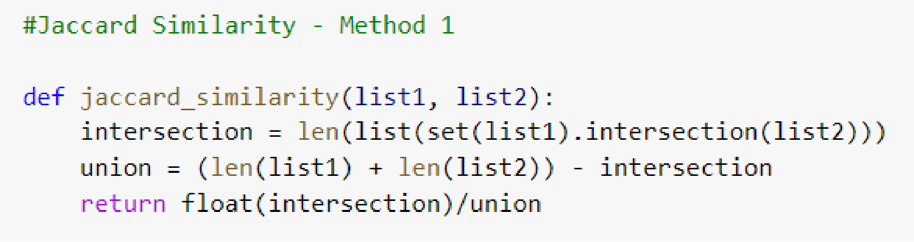
Objective: To implement Levenshtein Edit Distance , Jaccard similarity , Cosine Similarity using both TF-IDF and count vectorizer

Outcome: Students will be able to demonstrate Levenshtein Edit Distance , Jaccard similarity , Cosine Similarity using both TF-IDF and count vectorizer

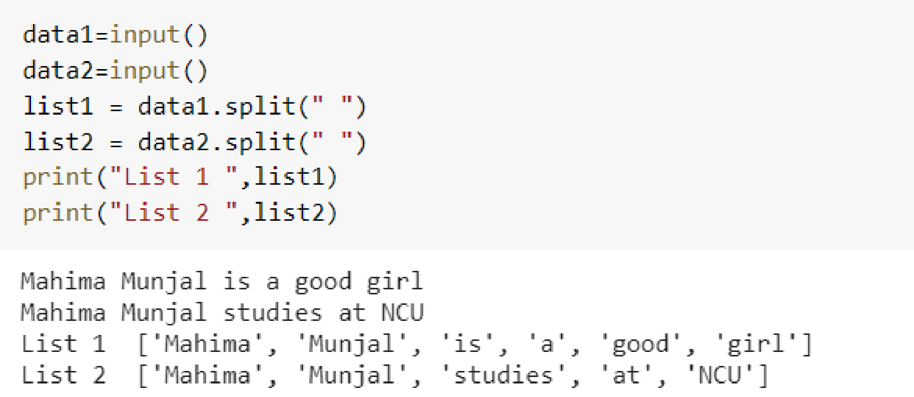
Problem Statement:

1. Demonstrate the computation of Similarity Metrics such as Jaccard, Levenshtein and Cosine.

CODE AND OUTPUT :

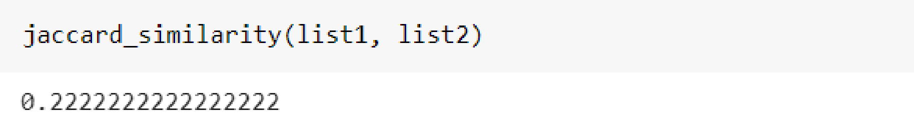


```
#Jaccard Similarity - Method 1

def jaccard_similarity(list1, list2):
    intersection = len(list(set(list1).intersection(list2)))
    union = (len(list1) + len(list2)) - intersection
    return float(intersection)/union
```



```
data1=input()
data2=input()
list1 = data1.split(" ")
list2 = data2.split(" ")
print("List 1 ",list1)
print("List 2 ",list2)
```

```
Mahima Munjal is a good girl
Mahima Munjal studies at NCU
List 1  ['Mahima', 'Munjal', 'is', 'a', 'good', 'girl']
List 2  ['Mahima', 'Munjal', 'studies', 'at', 'NCU']
```



```
jaccard_similarity(list1, list2)
```

```
0.2222222222222222
```

```
#Jaccard Similarity - Method  Z

def jaccard_similarities(1ist1, list2):
    s1  =set(list1)
    s2  =  set(list2)
    return float(len(s1.intersection(s2)) / len(s1.union(sZ)))


3accard_simi1arities(1ist1, list2)

0.222Z222Z222Z222Z
```

LEVENSHTEIN DISTANCE

```
[     ttLe vensht e1n D1stance

      impo l enc hant

      datal - “Mahima Munjal is a good girl.”
      data2 = “Mahima Munjal”

     enchant.uti1s.levenshtein(datal,data2)

     16
```

COSINE SIMILARITY

```
[     1np a °t nltk
      nltk.down load ( ' punkt ' )
      nltk.download('stopwords')

     [nltk_data) Downloading package punkt to /root/nltk_data...
     [nltk_data)   Package punkt is already up-to-date!
      nltk_data) Downloading package stopwords to /root/nltk_data...
     [nltk_data)   Package stopwords is already uo-to-date!
     True


I rom nltk . corpus in paat st opwords
from nltk.tokenzze import word_tokenize


I = input(”Enter first string: ”).lower()
II = input(”Enter second string: ”).lower()

Enter first string: Mahima Munjal is a good girl
Enter second string: Mahima Munjal studies at NCU and works at Nagarro


# tokenization
I_list = uord_tokenize(I)
II_list = word_tokenize(II)
print('First List', I_list)
print('Second Lzst', II_list)

First List ['mahima', 'munjal', 'is', 'a', 'good', 'girl']
Second List ['mahima', 'munjal', 'studies', 'at', 'ncu', 'and', 'works', 'at', 'uagarro']


#remov1ng st opwords

sw  = stopuords.words('english')
11  =[];12 =[l
I_set =  (w for w in I_list if not w in sw}
II_set = {w for w in II_list if not w in su}
pr1nt ( ' F1rst Set, '    I set)
pr1nt('5econd Set,'   II_set )

First Set {'good', 'mahima', 'girl', 'munjal'}
Second  Set  {'nagaono',  'ncu'   'stud1es', 'mu nj aJ ,'    'monks ', 'mahima '}
```

```
# Forming a set containing keywords of both strings

rvector = I_set.union(II_set}
for w in rvector:
    if w in I set: 11.append(1)
    else: 11.append(0)
    if w in II_set: 12.append(l)
    else: 12.append(0)
c = 0


# cosine formula
for i in range(len(rvector)):
        c+= 11[i]’l2[i]
cosine = c / float((sum(l1)*sum(lZ))**0.5)

print('Cosine Similarity Value :',cosine}

Cosine Similarity Value’ 0.A882A8290A638631
```

COSINE SIMILARITY MATRIX

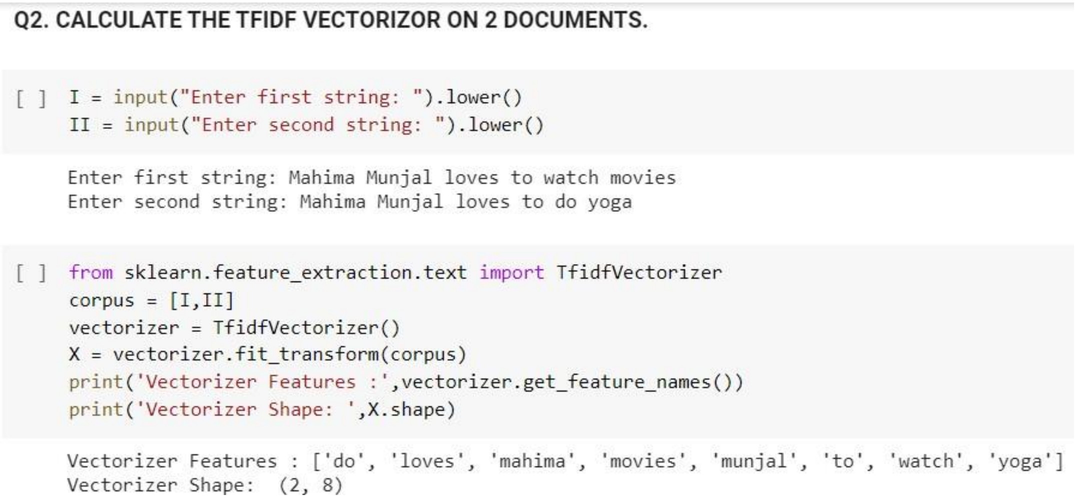
```
[ ] I = input("Enter first string: ").lower()
     II = input(”Enter second string: ”).lower()

     Enter first string: Mahima Munjal is a student Df Text and Web Analytics
     Enter second string: Mahima Munjal is a student of The Northcap University


[ ] documents = [I,II]


[ ] from sklearn.feature extraction.text import CountVectorizer
     import pandas as pd


[ ] count vectorizer = CountVectorizer(stop_words=’english')
     count_vectorizer = CountVectorizer()
     sparse_matrix = count vectorZzer.fit transform(documents)


     doc term matrix = sparse matrix.tDdense()
     df = pd.DataFrame(doc term_matrix,
                       columns=count vectorizer.get feature names(),
                       index=['I', 'II'])
     df
```

	ana1yt1cs	and	is	Inah1ma	munjal	northcap	a-I-	student	text	I he	univers1ty	web
I	1	1	1	1	1	0	1	1	1	0	0	1
ii	0	0	1	1	1	1	1	1	0	1	1	0

```
[ ]  from sklearn.metrics.pairwise import cosine_similarity
     print(cosine similarity(df, df))

     [[1.          0.58925565]
      [&.&g 2s&6s 1.         ]]
```

2. **Calculate the TF-IDF vectorizer on 2 documents.**

CODE AND OUTPUT:



Q2. CALCULATE THE TFIDF VECTORIZOR ON 2 DOCUMENTS.

```
[ ]  I = input("Enter first string: ").lower()
     II = input("Enter second string: ").lower()
```

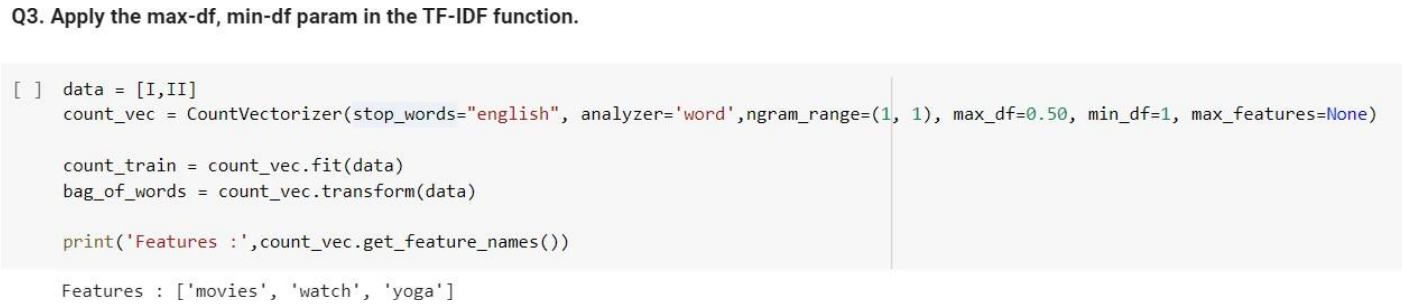
```
Enter first string: Mahima Munjal loves to watch movies
Enter second string: Mahima Munjal loves to do yoga
```

```
[ ]  from sklearn.feature_extraction.text import TfidfVectorizer
     corpus = [I,II]
     vectorizer = TfidfVectorizer()
     X = vectorizer.fit_transform(corpus)
     print('Vectorizer Features :',vectorizer.get_feature_names())
     print('Vectorizer Shape: ',X.shape)
```

```
Vectorizer Features : ['do', 'loves', 'mahima', 'movies', 'munjal', 'to', 'watch', 'yoga']
Vectorizer Shape:  (2, 8)
```

3. **Apply the max-df, min-df param in the TF-IDF function.**

CODE AND OUTPUT:



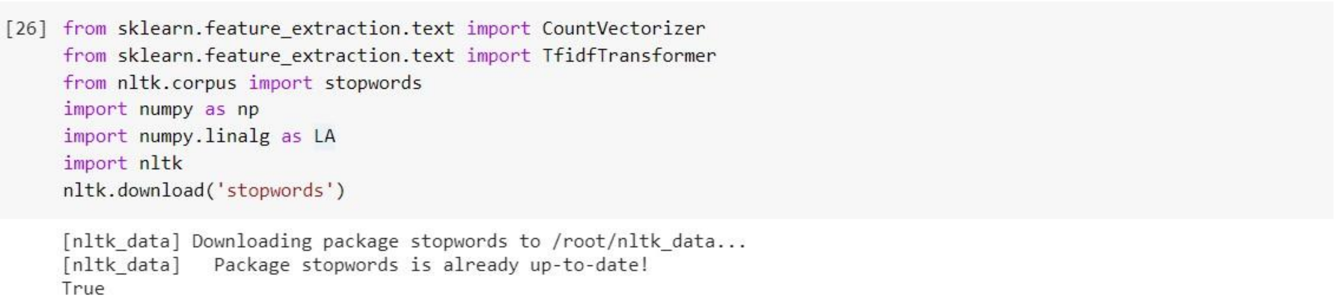
Q3. Apply the max-df, min-df param in the TF-IDF function.

```
[ ]  data = [I,II]
     count_vec = CountVectorizer(stop_words="english", analyzer='word',ngram_range=(1, 1), max_df=0.50, min_df=1, max_features=None)

     count_train = count_vec.fit(data)
     bag_of_words = count_vec.transform(data)

     print('Features :',count_vec.get_feature_names())
```

```
Features : ['movies', 'watch', 'yoga']
```

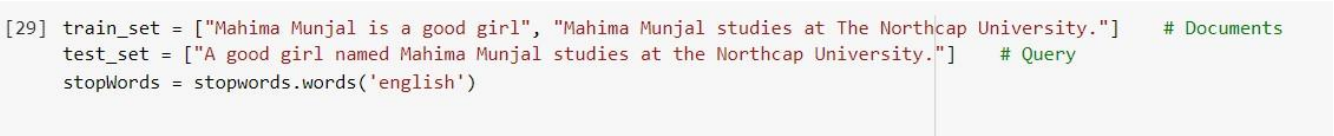
4. **Compute Cosine Similarity using both TF-IDF and Count vectorizer**

CODE AND OUTPUT:



```
[26] from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.feature_extraction.text import TfidfTransformer
     from nltk.corpus import stopwords
     import numpy as np
     import numpy.linalg as LA
     import nltk
     nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```



```
[29] train_set = ["Mahima Munjal is a good girl", "Mahima Munjal studies at The Northcap University."]    # Documents
     test_set = ["A good girl named Mahima Munjal studies at the Northcap University."]    # Query
     stopWords = stopwords.words('english')
```

```
vectorizer = CountVectorizer(stop_words = stopWords)
print('Vectorizer : ',vectorizer)
transformer = TfidfTransformer()
print('\n\nTF-IDF Transformer :',transformer)
```

```
Vectorizer    CountVectorizer(analyzer='wo d', bina y=False, decode_error='strict',
                dtype=<class 'numpy.int64'», encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max features=None, min_df=1,
                ngram range=(1, 1), preprocessor=None,
                stop words=['i', 'me', 'my', 'myself', 'we', 'our', 'ours
                            'ourselves', 'you', "you're", "you've", "you'11",
                            "you'd", 'your', 'yours', 'yourself', 'yourselves',
                            'he', 'him', 'his', 'himself', 'she', "she's",
                            'her', 'hers', 'herself', 'it', "it's", 'its',
                            'itself', ...],
                strip accents=None, token pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabu]ary=None)


TF-IDF Transformer    TfidfTransformer(norm='l2', smooth_idf=True, sublinear_t*=False, use_idf=True)
```

```
#Using Count Vectorizer

trainVectorlzerArray = vectorizer.fit_transform(train set).toarray()
testVectorizerArray   vectorizer.transform(test_set).toarray()
print ('Fit Vectorizer to train set \n', trainVectorizerArray)
print ('\n\nTransform Vectorizer to test set\n', testUectorizerArray)
```

```
Fit vectorizer to train set
 [[1 1 1 1 0 0 0]
 [0 0 1 1 1 1 1]]


Transform Vectorizer to test set
 [[1 1 1 1 1 1 1]]
```

```
#Using

transformer.fit(trainVectorizerArray)
print('Fit transformer to train set \n',transformer.transform(trainVectorizerArray).toarray())
transformer.fit(test4ectorizerArray)
tfidf = transformer.transform(testVectorizerArray)
print('\n\nFit transformer to test set\n',tfidf.todense())
```

```
Fit transformer to train set
 [[0.57615236 0.57615236 0.40993715 0.40993715 0.         0.
  0.        ]
 [0.         0.         0.35520009 0.35520009 0.49922133 0.ñ9922133
  0.49922133]]


Fit transformer to test set
 [[0.37796447 0.37796447 0.37796447 0.37796447 0.37796447 0.37796447
  0.37796447]]
```

EXPERIMENT NO. 5

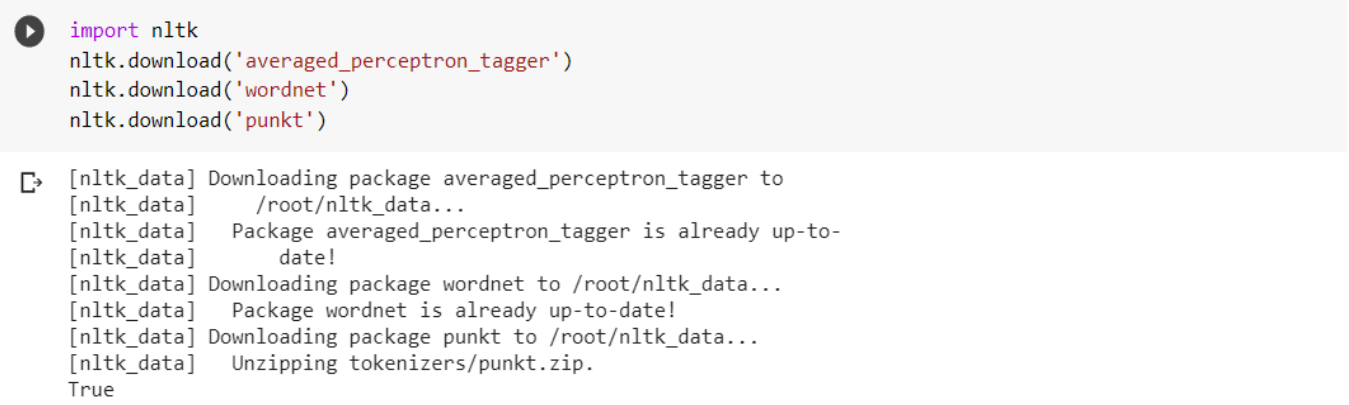
Student Name and Roll Number: RAHUL DHAWAN (17CSU146)
Semester /Section: VIII-C
Date: 6 March 2021
Faculty Signature:
Grade:

Objective: Implementation of the Lesk algorithm for Word Sense Disambiguation.

Outcome: Students will be able to demonstrate how to Lesk algorithm works.

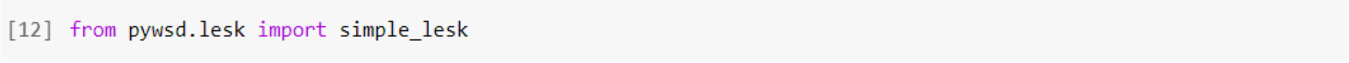
Problem Statement: Implement Lesk algorithm for Word Sense Disambiguation.

CODE AND OUTPUT:

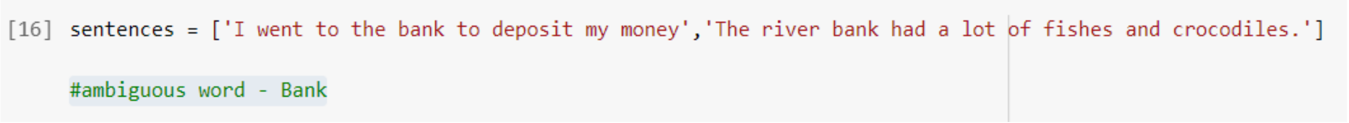


```
import nltk
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('punkt')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```



```
[12] from pywsd.lesk import simple_lesk
```



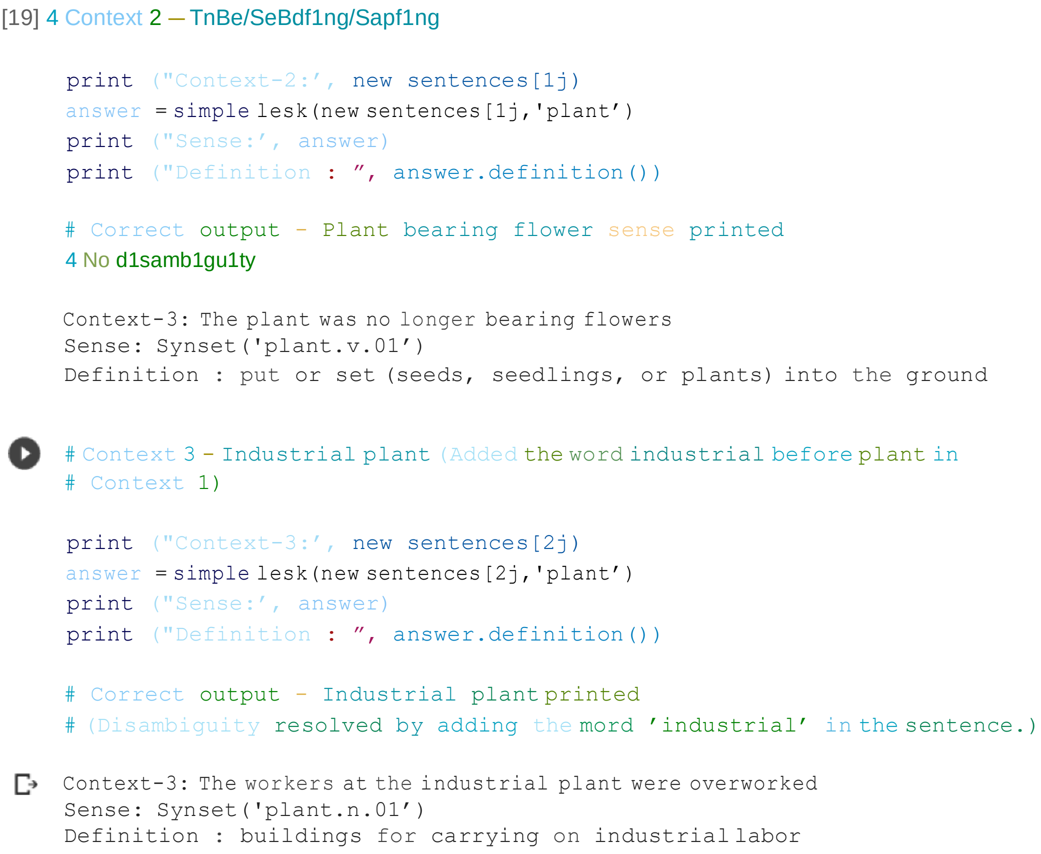
```
[16] sentences = ['I went to the bank to deposit my money','The river bank had a lot of fishes and crocodiles.']

     #ambiguous word - Bank
```

• LESK WORKS CORRECTLY

```
}13] # Context 1 - Financial institution

      print ("Context-1:", sentences[0])
      answer = simple lesk(sentences[0],'bank')
      print ("Sense:", answer)
      print ("Definition    ", answer.definition())

      # Correct Output - Financial Institution printed
      # No disambiguity

      Context-1: I went to the bank to deposit my money
      Sense: Synset('depository_financial_institution.n.01')
      Definition : a financial institution that accepts deposits and channels the money into lending activities


      # Context 2 - River Bank

      print ("Context-2:", sentences[1])
      answer = simple_lesk(sentences[1],'bank')
      print ("Sense:", answer)
      print ("Definition    ", answer.definition())


      # Correct Output - River Bank or sloping land printed
      # No disambiguity

      Context-2: The river bank had a lot of fishes and crocodiles.
      Sense: Syn set(' bank. n. 01' )
      Definition'  sloping land (especially the slope beside a body of water)
```

LESK WORKS INCORRECTLY

```
[17] neu_sentenc es   ['the «ac' ers at the p ant were overworked', 'The Dlant uas no longer Dearing +louers','the worker s at the i ndustria l ptant were over\vor kea']

     #ambiguous word °laut

[18] # Context 1 - I ndu strial pLant

     orint ("Context-1:", new sentences[0])
     answer = simple lesk(new sentences[0),
     oriut ("Sense:", answer)
     Drint              ", answer definition())

     # I ncorr ect output - I ndust ria 1 pla nt not printed
     # Disanhiguity riceured

     Context -1: The no Akers at the plant were ove ruorked
     Sense : Synset ( ' plant. v.06 ' )
     De+1n1tion     put firmly in the cdna
```



```
[19] 4 Context 2 — TnBe/SeBdf1ng/Sapf1ng

     print ("Context-2:’, new sentences[1j)
     answer = simple lesk(new sentences[1j,'plant’)
     print ("Sense:’, answer)
     print ("Definition : ”, answer.definition())

     # Correct output - Plant bearing flower sense printed
     4 No d1samb1gu1ty

     Context-3: The plant was no longer bearing flowers
     Sense: Synset('plant.v.01’)
     Definition : put or set (seeds, seedlings, or plants) into the ground
```

```
     # Context 3 - Industrial plant (Added the word industrial before plant in
     # Context 1)

     print ("Context-3:’, new sentences[2j)
     answer = simple lesk(new sentences[2j,'plant’)
     print ("Sense:’, answer)
     print ("Definition : ”, answer.definition())

     # Correct output - Industrial plant printed
     # (Disambiguity resolved by adding the mord ’industrial’ in the sentence.)

     Context-3: The workers at the industrial plant were overworked
     Sense: Synset('plant.n.01’)
     Definition : buildings for carrying on industrial labor
```