# PRACTICAL NO : 07

# TEXT ANALYTICS

## CODE :

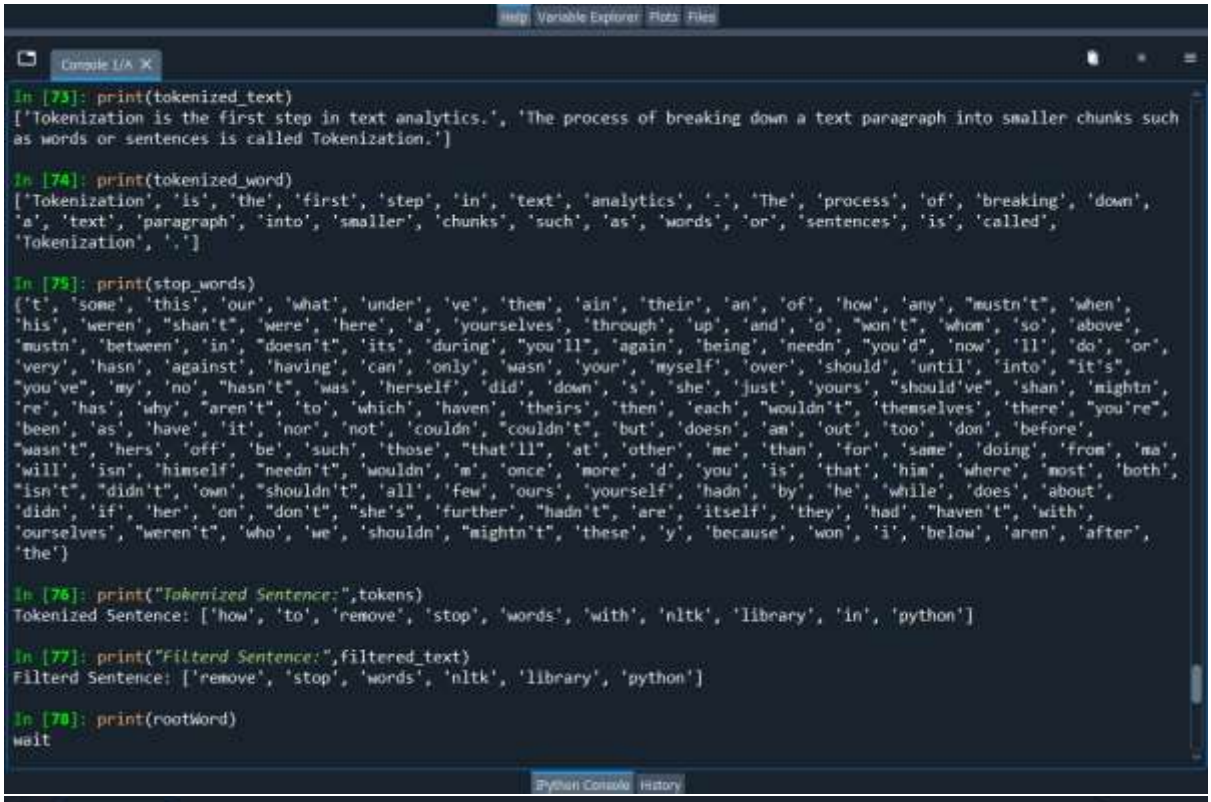
```
text="Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentences is called Tokenization."


from nltk.tokenize import sent_tokenize
tokenized_text=sent_tokenize(text)
print(tokenized_text)


from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)


from nltk import re
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)


text= "How to remove stop words with NLTK library in Python?"
text= re.sub('[^a-zA-Z]', ' ',text)
tokens = word_tokenize(text.lower())
filtered_text=[]
for w in tokens:
```

```
    if w not in stop_words:
        filtered_text.append(w)
print("Tokenized Sentence:",tokens)
print("Filterd Sentence:",filtered_text)

from nltk.stem import PorterStemmer
e_words=["wait","waitng","waited","waits"]
ps=PorterStemmer()
for w in e_words :
    rootWord=ps.stem(w)
    print(rootWord)

import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
for w in tokenization:
    print("Lemma for {} is {}".format(w,wordnet_lemmatizer.lemmatize(w)))

import nltk
from nltk.tokenize import word_tokenize
data="The pink sweater fit her perfectly"
words=word_tokenize(data)
for w in words:
    print(nltk.pos_tag([w]))
```

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'

bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')

uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
print(uniqueWords)

numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
    numOfWordsA[word] += 1
numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsB:
    numOfWordsB[word] += 1

def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count / float(bagOfWordsCount)
        return tfDict
```

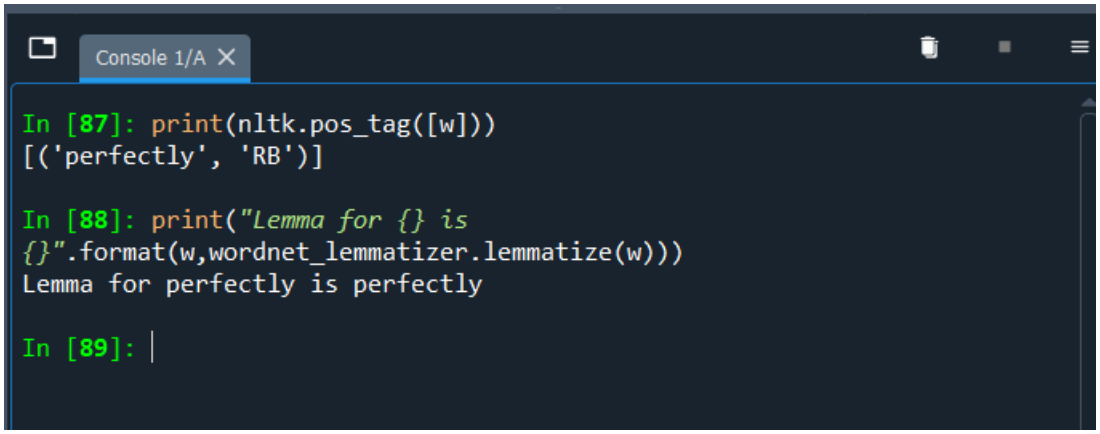
```
tfA = computeTF(numOfWordsA, bagOfWordsA)
tfB = computeTF(numOfWordsB, bagOfWordsB)

def computeIDF(documents):
    import math
    N = len(documents)
    idfDict = dict.fromkeys(documents[0].keys(), 0)
    for document in documents:
        for word, val in document.items():
            if val > 0:
                idfDict[word] += 1
            for word, val in idfDict.items():
                idfDict[word] = math.log(N / float(val))
                return idfDict
idfs = computeIDF([numOfWordsA, numOfWordsB])
```

**OUTPUT :**



```
Help  Variable Explorer  Plots  Files

Console 1/A

In [73]: print(tokenized_text)
['Tokenization is the first step in text analytics.', 'The process of breaking down a text paragraph into smaller chunks such
as words or sentences is called Tokenization.']

In [74]: print(tokenized_word)
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analytics', '.', 'The', 'process', 'of', 'breaking', 'down',
'a', 'text', 'paragraph', 'into', 'smaller', 'chunks', 'such', 'as', 'words', 'or', 'sentences', 'is', 'called',
'Tokenization', '.']

In [75]: print(stop_words)
{'t', 'some', 'this', 'our', 'what', 'under', 've', 'them', 'ain', 'their', 'an', 'of', 'how', 'any', "mustn't", 'when',
'his', 'weren', "shan't", 'were', 'here', 'a', 'yourselves', 'through', 'up', 'and', 'o', "won't", 'whom', 'so', 'above',
'mustn', 'between', 'in', "doesn't", 'its', 'during', "you'll", 'again', 'being', 'needn', "you'd", 'now', 'll', 'do', 'or',
'very', 'hasn', 'against', 'having', 'can', 'only', 'wasn', 'your', 'myself', 'over', 'should', 'until', 'into', "it's",
"you've", 'my', 'no', "hasn't", 'was', 'herself', 'did', 'down', 's', 'she', 'just', 'yours', "should've", 'shan', 'mightn',
're', 'has', 'why', "aren't", 'to', 'which', 'haven', 'theirs', 'then', 'each', "wouldn't", 'themselves', 'there', "you're",
'been', 'as', 'have', 'it', 'nor', 'not', 'couldn', "couldn't", 'but', 'doesn', 'am', 'out', 'too', 'don', 'before',
"wasn't", 'hers', 'off', 'be', 'such', 'those', "that'll", 'at', 'other', 'me', 'than', 'for', 'same', 'doing', 'from', 'ma',
'will', 'isn', 'himself', "needn't", 'wouldn', 'm', 'once', 'more', 'd', 'you', 'is', 'that', 'him', 'where', 'most', 'both',
"isn't", "didn't", 'own', "shouldn't", 'all', 'few', 'ours', 'yourself', 'hadn', 'by', 'he', 'while', 'does', 'about',
'didn', 'if', 'her', 'on', "don't", "she's", 'further', "hadn't", 'are', 'itself', 'they', 'had', "haven't", 'with',
'ourselves', "weren't", 'who', 'we', 'shouldn', "mightn't", 'these', 'y', 'because', 'won', 'i', 'below', 'aren', 'after',
'the'}

In [76]: print("Tokenized Sentence:",tokens)
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'library', 'in', 'python']

In [77]: print("Filterd Sentence:",filtered_text)
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library', 'python']

In [78]: print(rootWord)
wait

IPython Console  History
```

```
Console 1/A

IPython 8.15.0 -- An enhanced Interactive Python.

In [1]: import nltk

In [2]: nltk.download('punkt')
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\vedik\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
Out[2]: True

In [3]: nltk.download('stopwords')
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\vedik\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
Out[3]: True

In [4]: nltk.download('wordnet')
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\vedik\AppData\Roaming\nltk_data...
Out[4]: True

In [5]: nltk.download('average_perceptron_tagger')
[nltk_data] Error loading average_perceptron_tagger: Package
[nltk_data]     'average_perceptron_tagger' not found in index
Out[5]: False

In [6]: nltk.download('averaged_perceptron_tagger')
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\vedik\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
Out[6]: True

In [7]:

IPython Console  History
```


Console 1/A
In [87]: print(nltk.pos_tag([w]))
[('perfectly', 'RB')]
In [88]: print("Lemma for {} is
{}".format(w,wordnet_lemmatizer.lemmatize(w)))
Lemma for perfectly is perfectly
In [89]: