```
Reg. No. : 19MAI0032
Name : Harshal Patel
```

# nltk.corpus Demo

In [36]:

```
import nltk
```

In [37]:

```
#nltk.download()
```

# 1. Brown Corpus

In [38]:

```
# Import brown CORPUS and Access Data

from nltk.corpus import brown
brown.categories()
```

Out[38]:

```
['adventure',
 'belles_lettres',
 'editorial',
 'fiction',
 'government',
 'hobbies',
 'humor',
 'learned',
 'lore',
 'mystery',
 'news',
 'religion',
 'reviews',
 'romance',
 'science_fiction']
```

In [39]:

```
# Access the corpus as a list of words
brown.words(categories='adventure')[:100]
```

Out[39]:

```
['Dan', 'Morgan', 'told', 'himself', 'he', 'would', ...]
```

In [40]:

```
# Access the corpus as a list of sentences
# where each sentence is itself just a list of words
brown.sents(categories='adventure')
```

Out[40]:

```
[['Dan', 'Morgan', 'told', 'himself', 'he', 'would', 'forget', 'Ann', 'Turne
r', '.'], ['He', 'was', 'well', 'rid', 'of', 'her', '.'], ...]
```

# 2. Inaugural Corpus

In [41]:

```
# Import inaugural CORPUS and Access Data
from nltk.corpus import inaugural

# List of Presidential Address Files
inaugural.fileids()
```

Out[41]:

```
['1789-Washington.txt',
 '1793-Washington.txt',
 '1797-Adams.txt',
 '1801-Jefferson.txt',
 '1805-Jefferson.txt',
 '1809-Madison.txt',
 '1813-Madison.txt',
 '1817-Monroe.txt',
 '1821-Monroe.txt',
 '1825-Adams.txt',
 '1829-Jackson.txt',
 '1833-Jackson.txt',
 '1837-VanBuren.txt',
 '1841-Harrison.txt',
 '1845-Polk.txt',
 '1849-Taylor.txt',
 '1853-Pierce.txt',
 '1857-Buchanan.txt',
 '1861-Lincoln.txt',
 '1865-Lincoln.txt',
 '1869-Grant.txt',
 '1873-Grant.txt',
 '1877-Hayes.txt',
 '1881-Garfield.txt',
 '1885-Cleveland.txt',
 '1889-Harrison.txt',
 '1893-Cleveland.txt',
 '1897-McKinley.txt',
 '1901-McKinley.txt',
 '1905-Roosevelt.txt',
 '1909-Taft.txt',
 '1913-Wilson.txt',
 '1917-Wilson.txt',
 '1921-Harding.txt',
 '1925-Coolidge.txt',
 '1929-Hoover.txt',
 '1933-Roosevelt.txt',
 '1937-Roosevelt.txt',
 '1941-Roosevelt.txt',
 '1945-Roosevelt.txt',
 '1949-Truman.txt',
 '1953-Eisenhower.txt',
 '1957-Eisenhower.txt',
 '1961-Kennedy.txt',
 '1965-Johnson.txt',
 '1969-Nixon.txt',
 '1973-Nixon.txt',
 '1977-Carter.txt',
 '1981-Reagan.txt',
 '1985-Reagan.txt',
 '1989-Bush.txt',
```

```
 '1993-Clinton.txt',
 '1997-Clinton.txt',
 '2001-Bush.txt',
 '2005-Bush.txt',
 '2009-Obama.txt',
 '2013-Obama.txt',
 '2017-Trump.txt']
```

In [42]:

```
# Access the corpus as a list of words
inaugural.words(fileids = '2005-Bush.txt')
```

Out[42]:

```
['Vice', 'President', 'Cheney', ',', 'Mr', '.', ...]
```

# 3. Conditional Frequency Distribution in Presidential Address

In [43]:

```
# Take a Presidential Address '2005-Bush.txt' as a raw text
president_bush = inaugural.raw('2005-Bush.txt')
```

In [44]:

```
# Conditional Frequency Distribution on Presidential Address of '2005-Bush.txt'
from nltk.probability import ConditionalFreqDist

# Frequency Distribution of Words of Each Length
cfd = ConditionalFreqDist((len(word), word) for word in president_bush.split())
```

In [45]:

```
#Words of length 5
cfd[5]
```

Out[45]:

```
FreqDist({'every': 10, 'their': 10, 'human': 6, 'which': 5, 'time.': 4, 'gre
at': 4, 'those': 4, 'know:': 4, 'honor': 3, 'years': 3, ...})
```

In [46]:

```
#Words of length 6
cfd[6]
```

Out[46]:

```
FreqDist({'United': 5, 'States': 5, 'nation': 5, 'choice': 4, 'fellow': 3,
'excuse': 3, 'world.': 3, 'people': 3, 'always': 3, 'rights': 3, ...})
```

# 4. Webtext Corpus

In [47]:

```
# Import Webtext CORPUS and Access Data
from nltk.corpus import webtext
webtext.fileids()
```

Out[47]:

```
['firefox.txt',
 'grail.txt',
 'overheard.txt',
 'pirates.txt',
 'singles.txt',
 'wine.txt']
```

In [48]:

```
#Prints first 50 char of all webtext fileids
for fileids in webtext.fileids():
    print(fileids," : ", webtext.raw(fileids)[:50])
```

```
firefox.txt  :  Cookie Manager: "Don't allow sites that set remove
grail.txt  :  SCENE 1: [wind] [clop clop clop]
KING ARTHUR: Who
overheard.txt  :  White guy: So, do you have any plans for this even
pirates.txt  :  PIRATES OF THE CARRIBEAN: DEAD MAN'S CHEST, by Ted
singles.txt  :  25 SEXY MALE, seeks attrac older single lady, for
wine.txt  :  Lovely delicate, fragrant Rhone wine. Polished lea
```

In [49]:

```
# Frequency Distribution of words in a webtext file 'firefox.txt'

from nltk.probability import FreqDist
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Take raw firefox.txt file in text
text = webtext.raw('firefox.txt')

# frequency distribution on firefox.txt file
fd_words = nltk.FreqDist(text.split())
fd_words
```

Out[49]:

```
FreqDist({'in': 2144, 'to': 2118, 'the': 1758, 'not': 1459, 'when': 1246, 'o
n': 1175, 'a': 1150, 'is': 1011, 'of': 866, 'and': 863, ...})
```

In [50]:

```
# lneght of words greater than 3 are cosidered in filter_words
filter_words = dict([(m, n) for m, n in fd_words.items() if len(m) > 3])
wcloud = WordCloud().generate_from_frequencies(filter_words)
```

In [51]:

```
# Plotting the wordcloud
plt.imshow(wcloud, interpolation="bicubic")
plt.axis("off")
plt.show()
```


In [52]:

```
# Conditional Frequency Distribution of words in a text

from nltk.probability import ConditionalFreqDist

text = '''Cricket is a bat-and-ball game played between two teams of eleven players on a fi
There are various formats ranging from Twenty20, played over a few hours with each team bat

cfd = ConditionalFreqDist((len(word), word) for word in text.split())
```

In [53]:

```
#Words of length 4
cfd[4]
```

Out[53]:

```
FreqDist({'with': 7, 'each': 5, 'side': 3, 'ball': 3, 'game': 2, 'bat,': 2,
'they': 2, 'hits': 2, 'over': 2, 'team': 2, ...})
```

In [54]:

```
#Words of length 5
cfd[5]
```

Out[54]:

```
FreqDist({'teams': 3, 'which': 3, 'overs': 2, 'field': 1, 'pitch': 1, 'bail
s': 1, 'three': 1, 'while': 1, 'tries': 1, 'Means': 1, ...})
```

# Stemmers

# 5. Porter Stemmer

In [55]:

```
from nltk.stem import PorterStemmer
stemmerporter = PorterStemmer()
stemmerporter.stem('happiness')
```

Out[55]:

```
'happi'
```

In [56]:

```
stemmerporter.stem('terribly')
```

Out[56]:

```
'terribl'
```

In [57]:

```
stemmerporter.stem('mercifull')
```

Out[57]:

```
'merciful'
```

# 6. Lancaster Stemmer

In [58]:

```
from nltk.stem import LancasterStemmer
stemmerLan = LancasterStemmer()
stemmerLan.stem('happiness')
```

Out[58]:

```
'happy'
```

In [59]:

```
stemmerLan.stem('terribly')
```

Out[59]:

```
'terr'
```

In [60]:

```
stemmerLan.stem('mercifull')
```

Out[60]:

```
'merciful'
```

# 7. Snowball Stemmer

In [61]:

```
from nltk.stem import SnowballStemmer
SnowballStemmer.languages
```

Out[61]:

```
('arabic',
 'danish',
 'dutch',
 'english',
 'finnish',
 'french',
 'german',
 'hungarian',
 'italian',
 'norwegian',
 'porter',
 'portuguese',
 'romanian',
 'russian',
 'spanish',
 'swedish')
```

In [62]:

```
engStemmer=SnowballStemmer('english').stem("generously")
print(engStemmer)
```

```
generous
```

In [63]:

```
# Snowball stemmer is not properly work for words which is having stopwords
engStemmer=SnowballStemmer('english').stem("happiness")
print(engStemmer)
```

```
happi
```

# 8. Lemmatization

In [64]:

```
# Lemmatization reduces words to their base word
# which is linguistically correct lemmas

from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
```

In [65]:

```
lem = WordNetLemmatizer()
stem = PorterStemmer()

# Difference of Lemmatization and Stemmer
word = "flying"
print("Lemmatized Word:",lem.lemmatize(word,"v"))
print("Stemmed Word:",stem.stem(word))

# Difference of Lemmatization and Stemmer
word = "better"
print("\nLemmatized Word:",lem.lemmatize(word,"a"))
print("Stemmed Word:",stem.stem(word))
```

```
Lemmatized Word: fly
Stemmed Word: fli

Lemmatized Word: good
Stemmed Word: better
```

In [ ]:

```

```