In [1]:
```
import os
import nltk
#nltk.download()
```

In [2]:
```
#import nltk.corpus
```

In [3]:
```
# we will see what is mean by corpora and what all are availabel in nltk python
#print(os.listdir(nltk.data.find('corpora')))

#you get a lot of file , some of have some textual document, different function
#for our example i will lets take consideration as brown & we will understand wh
```

In [4]:
```
#from nltk.corpus import brown
#brown.words()
```

In [5]:
```
#nltk.corpus.gutenberg.fileids()
```

In [6]:
```
# you can also create your own words

AI = '''Artificial Intelligence refers to the intelligence of machines. This is
humans and animals. With Artificial Intelligence, machines perform functions suc
problem-solving. Most noteworthy, Artificial Intelligence is the simulation of h
It is probably the fastest-growing development in the World of technology and in
AI could solve major challenges and crisis situations.'''
```

In [7]:
```
AI
```

Out[7]:
```
'Artificial Intelligence refers to the intelligence of machines. This is in con
trast to the natural intelligence of \nhumans and animals. With Artificial Inte
lligence, machines perform functions such as learning, planning, reasoning and
\nproblem-solving. Most noteworthy, Artificial Intelligence is the simulation o
f human intelligence by machines. \nIt is probably the fastest-growing developm
ent in the World of technology and innovation. Furthermore, many experts believ
e\nAI could solve major challenges and crisis situations.'
```

In [8]:
```
type(AI)
```

Out[8]:
```
str
```

In [9]:
```
from nltk.tokenize import word_tokenize
```

In [10]:
```
AI_tokens = word_tokenize(AI)
AI_tokens
```

Out[10]:
```
['Artificial',
 'Intelligence',
 'refers',
 'to',
 'the',
 'intelligence',
 'of',
 'machines',
 '.',
 'This',
 'is',
 'in',
 'contrast',
 'to',
 'the',
 'natural',
 'intelligence',
 'of',
 'humans',
 'and',
 'animals',
 '.',
 'With',
 'Artificial',
 'Intelligence',
 ',',
 'machines',
 'perform',
 'functions',
 'such',
 'as',
 'learning',
 ',',
 'planning',
 ',',
 'reasoning',
 'and',
 'problem-solving',
 '.',
 'Most',
 'noteworthy',
 ',',
 'Artificial',
 'Intelligence',
 'is',
 'the',
 'simulation',
 'of',
 'human',
 'intelligence',
 'by',
 'machines',
 '.',
 'It',
 'is',
 'probably',
 'the',
 'fastest-growing',
 'development',
 'in',
```

```
 'the',
 'World',
 'of',
 'technology',
 'and',
 'innovation',
 '.',
 'Furthermore',
 ',',
 'many',
 'experts',
 'believe',
 'AI',
 'could',
 'solve',
 'major',
 'challenges',
 'and',
 'crisis',
 'situations',
 '.']
```

In [11]:
```
len(AI_tokens)
```

Out[11]: 81

In [12]:
```
from nltk.tokenize import sent_tokenize
```

In [13]:
```
AI_sent = sent_tokenize(AI)
AI_sent
```

Out[13]:
```
['Artificial Intelligence refers to the intelligence of machines.',
 'This is in contrast to the natural intelligence of \nhumans and animals.',
 'With Artificial Intelligence, machines perform functions such as learning, pl
anning, reasoning and \nproblem-solving.',
 'Most noteworthy, Artificial Intelligence is the simulation of human intellige
nce by machines.',
 'It is probably the fastest-growing development in the World of technology and
innovation.',
 'Furthermore, many experts believe\nAI could solve major challenges and crisis
situations.']
```

In [14]:
```
len(AI_sent)
```

Out[14]: 6

In [15]:
```
AI
```

Out[15]:
```
'Artificial Intelligence refers to the intelligence of machines. This is in con
trast to the natural intelligence of \nhumans and animals. With Artificial Inte
lligence, machines perform functions such as learning, planning, reasoning and
\nproblem-solving. Most noteworthy, Artificial Intelligence is the simulation o
f human intelligence by machines. \nIt is probably the fastest-growing developm
ent in the World of technology and innovation. Furthermore, many experts believ
e\nAI could solve major challenges and crisis situations.'
```

In [16]:
```
from nltk.tokenize import blankline_tokenize # GiVE YOU HOW MANY PARAGRAPH
AI_blank = blankline_tokenize(AI)
```

```
AI_blank
#AI_blank
```

Out[16]:
```
['Artificial Intelligence refers to the intelligence of machines. This is in co
ntrast to the natural intelligence of \nhumans and animals. With Artificial Int
elligence, machines perform functions such as learning, planning, reasoning and
\nproblem-solving. Most noteworthy, Artificial Intelligence is the simulation o
f human intelligence by machines. \nIt is probably the fastest-growing developm
ent in the World of technology and innovation. Furthermore, many experts believ
e\nAI could solve major challenges and crisis situations.']
```

In [17]:
```
len(AI_blank)
```

Out[17]:
```
1
```

In [18]:
```
# NEXT WE WILL SEE HOW WE WILL USE UNI-GRAM,BI-GRAM,TRI-GRAM USING NLTK

from nltk.util import bigrams,trigrams,ngrams
```

In [19]:
```
string = 'the best and most beautifull thing in the world cannot be seen or even
quotes_tokens = nltk.word_tokenize(string)
```

In [20]:
```
quotes_tokens
```

Out[20]:
```
['the',
 'best',
 'and',
 'most',
 'beautifull',
 'thing',
 'in',
 'the',
 'world',
 'can',
 'not',
 'be',
 'seen',
 'or',
 'even',
 'touched',
 ',',
 'they',
 'must',
 'be',
 'felt',
 'with',
 'heart']
```

In [21]:
```
len(quotes_tokens)
```

Out[21]:
```
23
```

In [22]:
```
quotes_bigrams = list(nltk.bigrams(quotes_tokens))
quotes_bigrams
```

Out[22]:
```
[('the', 'best'),
 ('best', 'and'),
 ('and', 'most'),
 ('most', 'beautifull'),
 ('beautifull', 'thing'),
 ('thing', 'in'),
 ('in', 'the'),
 ('the', 'world'),
 ('world', 'can'),
 ('can', 'not'),
 ('not', 'be'),
 ('be', 'seen'),
 ('seen', 'or'),
 ('or', 'even'),
 ('even', 'touched'),
 ('touched', ','),
 (',', 'they'),
 ('they', 'must'),
 ('must', 'be'),
 ('be', 'felt'),
 ('felt', 'with'),
 ('with', 'heart')]
```

In [23]:
```
quotes_tokens
```

Out[23]:
```
['the',
 'best',
 'and',
 'most',
 'beautifull',
 'thing',
 'in',
 'the',
 'world',
 'can',
 'not',
 'be',
 'seen',
 'or',
 'even',
 'touched',
 ',',
 'they',
 'must',
 'be',
 'felt',
 'with',
 'heart']
```

In [24]:
```
quotes_trigrams = list(nltk.trigrams(quotes_tokens))
quotes_trigrams
```

Out[24]:
```
[('the', 'best', 'and'),
 ('best', 'and', 'most'),
 ('and', 'most', 'beautifull'),
 ('most', 'beautifull', 'thing'),
 ('beautifull', 'thing', 'in'),
 ('thing', 'in', 'the'),
 ('in', 'the', 'world'),
 ('the', 'world', 'can'),
 ('world', 'can', 'not'),
 ('can', 'not', 'be'),
 ('not', 'be', 'seen'),
 ('be', 'seen', 'or'),
 ('seen', 'or', 'even'),
 ('or', 'even', 'touched'),
 ('even', 'touched', ','),
 ('touched', ',', 'they'),
 (',', 'they', 'must'),
 ('they', 'must', 'be'),
 ('must', 'be', 'felt'),
 ('be', 'felt', 'with'),
 ('felt', 'with', 'heart')]
```

In [25]:
```
quotes_trigrams = list(nltk.ngrams(quotes_tokens))
quotes_trigrams
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[25], line 1
----> 1 quotes_trigrams = list(nltk.ngrams(quotes_tokens))
      2 quotes_trigrams

TypeError: ngrams() missing 1 required positional argument: 'n'
```

In [26]:
```
quotes_ngrams = list(nltk.ngrams(quotes_tokens, 4))
quotes_ngrams

#it has given n-gram of length 4
```

Out[26]:
```
[('the', 'best', 'and', 'most'),
 ('best', 'and', 'most', 'beautifull'),
 ('and', 'most', 'beautifull', 'thing'),
 ('most', 'beautifull', 'thing', 'in'),
 ('beautifull', 'thing', 'in', 'the'),
 ('thing', 'in', 'the', 'world'),
 ('in', 'the', 'world', 'can'),
 ('the', 'world', 'can', 'not'),
 ('world', 'can', 'not', 'be'),
 ('can', 'not', 'be', 'seen'),
 ('not', 'be', 'seen', 'or'),
 ('be', 'seen', 'or', 'even'),
 ('seen', 'or', 'even', 'touched'),
 ('or', 'even', 'touched', ','),
 ('even', 'touched', ',', 'they'),
 ('touched', ',', 'they', 'must'),
 (',', 'they', 'must', 'be'),
 ('they', 'must', 'be', 'felt'),
 ('must', 'be', 'felt', 'with'),
 ('be', 'felt', 'with', 'heart')]
```

In [27]:
```
len(quotes_tokens)
```

Out[27]: 23

In [28]:
```
quotes_ngrams_1 = list(nltk.ngrams(quotes_tokens, 5))
quotes_ngrams_1
```

Out[28]:
```
[('the', 'best', 'and', 'most', 'beautifull'),
 ('best', 'and', 'most', 'beautifull', 'thing'),
 ('and', 'most', 'beautifull', 'thing', 'in'),
 ('most', 'beautifull', 'thing', 'in', 'the'),
 ('beautifull', 'thing', 'in', 'the', 'world'),
 ('thing', 'in', 'the', 'world', 'can'),
 ('in', 'the', 'world', 'can', 'not'),
 ('the', 'world', 'can', 'not', 'be'),
 ('world', 'can', 'not', 'be', 'seen'),
 ('can', 'not', 'be', 'seen', 'or'),
 ('not', 'be', 'seen', 'or', 'even'),
 ('be', 'seen', 'or', 'even', 'touched'),
 ('seen', 'or', 'even', 'touched', ','),
 ('or', 'even', 'touched', ',', 'they'),
 ('even', 'touched', ',', 'they', 'must'),
 ('touched', ',', 'they', 'must', 'be'),
 (',', 'they', 'must', 'be', 'felt'),
 ('they', 'must', 'be', 'felt', 'with'),
 ('must', 'be', 'felt', 'with', 'heart')]
```

In [29]:
```
quotes_ngrams = list(nltk.ngrams(quotes_tokens, 9))
quotes_ngrams
```

Out[29]:
```
[('the', 'best', 'and', 'most', 'beautifull', 'thing', 'in', 'the', 'world'),
 ('best', 'and', 'most', 'beautifull', 'thing', 'in', 'the', 'world', 'can'),
 ('and', 'most', 'beautifull', 'thing', 'in', 'the', 'world', 'can', 'not'),
 ('most', 'beautifull', 'thing', 'in', 'the', 'world', 'can', 'not', 'be'),
 ('beautifull', 'thing', 'in', 'the', 'world', 'can', 'not', 'be', 'seen'),
 ('thing', 'in', 'the', 'world', 'can', 'not', 'be', 'seen', 'or'),
 ('in', 'the', 'world', 'can', 'not', 'be', 'seen', 'or', 'even'),
 ('the', 'world', 'can', 'not', 'be', 'seen', 'or', 'even', 'touched'),
 ('world', 'can', 'not', 'be', 'seen', 'or', 'even', 'touched', ','),
 ('can', 'not', 'be', 'seen', 'or', 'even', 'touched', ',', 'they'),
 ('not', 'be', 'seen', 'or', 'even', 'touched', ',', 'they', 'must'),
 ('be', 'seen', 'or', 'even', 'touched', ',', 'they', 'must', 'be'),
 ('seen', 'or', 'even', 'touched', ',', 'they', 'must', 'be', 'felt'),
 ('or', 'even', 'touched', ',', 'they', 'must', 'be', 'felt', 'with'),
 ('even', 'touched', ',', 'they', 'must', 'be', 'felt', 'with', 'heart')]
```

In [30]:
```
# Next we need to make some changes in tokens and that is called as stemming, st
# also we will see some root form of the word & limitation of the word

#porter-stemmer
from nltk.stem import PorterStemmer
pst = PorterStemmer()
```

In [31]:
```
pst.stem('having') #stem will gives you the root form of the word
```

Out[31]: 'have'

In [32]:
```
pst.stem('affection')
```

Out[32]: 'affect'

In [33]:
```
pst.stem('playing')
```

Out[33]: 'play'

In [34]:
```
pst.stem('give')
```

Out[34]: 'give'

In [35]:
```
words_to_stem=['give','giving','given','gave']
for words in words_to_stem:
    print(words+  ':' + pst.stem(words))
```

```
give:give
giving:give
given:given
gave:gave
```

In [36]:
```
pst.stem('playing')
```

Out[36]: 'play'

In [37]:
```
words_to_stem=['give','giving','given','gave','thinking', 'loving', 'final', 'fi
# i am giving these different words to stem, using porter stemmer we get the out

for words in words_to_stem:
    print(words+ ':' +pst.stem(words))

#in porterstemmer removes ing and replaces with e
```

```
give:give
giving:give
given:given
gave:gave
thinking:think
loving:love
final:final
finalized:final
finally:final
```

In [38]:
```
#another stemmer known as lencastemmer stemmer and lets see what the different w
#stem the same thing using lencastemmer

from nltk.stem import LancasterStemmer
lst = LancasterStemmer()
for words in words_to_stem:
    print(words + ':' + lst.stem(words))

# lancasterstemmer is more aggresive then the porterstemmer
```

```
give:giv
giving:giv
given:giv
gave:gav
thinking:think
loving:lov
final:fin
finalized:fin
finally:fin
```

In [39]:
```
words_to_stem=['give','giving','given','gave','thinking', 'loving', 'final', 'fi
# i am giving these different words to stem, using porter stemmer we get the out

for words in words_to_stem:
    print(words+ ':' +pst.stem(words))
```

```
give:give
giving:give
given:given
gave:gave
thinking:think
loving:love
final:final
finalized:final
finally:final
```

In [40]:
```
#we have another stemmer called as snowball stemmer lets see about this snowball

from nltk.stem import SnowballStemmer
sbst = SnowballStemmer('english')
for words in words_to_stem:
    print(words+ ':' +sbst.stem(words))

#snowball stemmer is same as portstemmer
#different type of stemmer used based on different type of task
#if you want to see how many type of giv has occured then we will see the lancas
```

```
give:give
giving:give
given:given
gave:gave
thinking:think
loving:love
final:final
finalized:final
finally:final
```

In [41]:
```
#sometime stemming does not work & lets say e.g - fish,fishes & fishing all of t
#one hand stemming will cut the end & lemmatization will take into the morpholog

from nltk.stem import wordnet
from nltk.stem import WordNetLemmatizer
word_lem = WordNetLemmatizer()

#Hear we are going to wordnet dictionary & we are going to import the wordnet le
```

In [42]:
```
words_to_stem
```

Out[42]:
```
['give',
 'giving',
 'given',
 'gave',
 'thinking',
 'loving',
 'final',
 'finalized',
 'finally']
```

In [43]:
```
#word_lem.lemmatize('corpora') #we get output as corpus

#refers to a collection of texts. Such collections may be formed of a single lan

for words in words_to_stem:
    print(words+ ':' +word_lem.lemmatize(words))
```

```
give:give
giving:giving
given:given
gave:gave
thinking:thinking
loving:loving
final:final
finalized:finalized
finally:finally
```

In [44]:
```
pst.stem('final')
```

Out[44]:
```
'final'
```

In [45]:
```
lst.stem('finally')
```

Out[45]:
```
'fin'
```

In [46]:
```
sbst.stem('finalized')
```

Out[46]:
```
'final'
```

In [47]:
```
lst.stem('final')
```

Out[47]:
```
'fin'
```

In [48]:
```
lst.stem('finalized')
```

Out[48]:
```
'fin'
```

In [49]:
```
# there is other concept called POS (part of speech) which deals with subject, n
# STOPWORDS = i, is, as,at, on, about & nltk has their own list of stopewords

from nltk.corpus import stopwords
```

In [50]:
```
stopwords.words('english')
```

Out[50]:
```
['a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'ain',
 'all',
 'am',
 'an',
 'and',
 'any',
 'are',
 'aren',
 "aren't",
 'as',
 'at',
 'be',
 'because',
 'been',
 'before',
 'being',
 'below',
 'between',
 'both',
 'but',
 'by',
 'can',
 'couldn',
 "couldn't",
 'd',
 'did',
 'didn',
 "didn't",
 'do',
 'does',
 'doesn',
 "doesn't",
 'doing',
 'don',
 "don't",
 'down',
 'during',
 'each',
 'few',
 'for',
 'from',
 'further',
 'had',
 'hadn',
 "hadn't",
 'has',
 'hasn',
 "hasn't",
 'have',
 'haven',
 "haven't",
 'having',
 'he',
 "he'd",
```

```
"he'll",
'her',
'here',
'hers',
'herself',
"he's",
'him',
'himself',
'his',
'how',
'i',
"i'd",
'if',
"i'll",
"i'm",
'in',
'into',
'is',
'isn',
"isn't",
'it',
"it'd",
"it'll",
"it's",
'its',
'itself',
"i've",
'just',
'll',
'm',
'ma',
'me',
'mightn',
"mightn't",
'more',
'most',
'mustn',
"mustn't",
'my',
'myself',
'needn',
"needn't",
'no',
'nor',
'not',
'now',
'o',
'of',
'off',
'on',
'once',
'only',
'or',
'other',
'our',
'ours',
'ourselves',
'out',
'over',
'own',
```

```
're',
's',
'same',
'shan',
"shan't",
'she',
"she'd",
"she'll",
"she's",
'should',
'shouldn',
"shouldn't",
"should've",
'so',
'some',
'such',
't',
'than',
'that',
"that'll",
'the',
'their',
'theirs',
'them',
'themselves',
'then',
'there',
'these',
'they',
"they'd",
"they'll",
"they're",
"they've",
'this',
'those',
'through',
'to',
'too',
'under',
'until',
'up',
've',
'very',
'was',
'wasn',
"wasn't",
'we',
"we'd",
"we'll",
"we're",
'were',
'weren',
"weren't",
"we've",
'what',
'when',
'where',
'which',
'while',
'who',
```

```
 'whom',
 'why',
 'will',
 'with',
 'won',
 "won't",
 'wouldn',
 "wouldn't",
 'y',
 'you',
 "you'd",
 "you'll",
 'your',
 "you're",
 'yours',
 'yourself',
 'yourselves',
 "you've"]
```

In [51]:
```
len(stopwords.words('english'))
```

Out[51]: 198

In [52]:
```
stopwords.words('spanish')
```

Out[52]:
```
['de',
 'la',
 'que',
 'el',
 'en',
 'y',
 'a',
 'los',
 'del',
 'se',
 'las',
 'por',
 'un',
 'para',
 'con',
 'no',
 'una',
 'su',
 'al',
 'lo',
 'como',
 'más',
 'pero',
 'sus',
 'le',
 'ya',
 'o',
 'este',
 'sí',
 'porque',
 'esta',
 'entre',
 'cuando',
 'muy',
 'sin',
 'sobre',
 'también',
 'me',
 'hasta',
 'hay',
 'donde',
 'quien',
 'desde',
 'todo',
 'nos',
 'durante',
 'todos',
 'uno',
 'les',
 'ni',
 'contra',
 'otros',
 'ese',
 'eso',
 'ante',
 'ellos',
 'e',
 'esto',
 'mí',
 'antes',
```

```
'algunos',
'qué',
'unos',
'yo',
'otro',
'otras',
'otra',
'él',
'tanto',
'esa',
'estos',
'mucho',
'quienes',
'nada',
'muchos',
'cual',
'poco',
'ella',
'estar',
'estas',
'algunas',
'algo',
'nosotros',
'mi',
'mis',
'tú',
'te',
'ti',
'tu',
'tus',
'ellas',
'nosotras',
'vosotros',
'vosotras',
'os',
'mío',
'mía',
'míos',
'mías',
'tuyo',
'tuya',
'tuyos',
'tuyas',
'suyo',
'suya',
'suyos',
'suyas',
'nuestro',
'nuestra',
'nuestros',
'nuestras',
'vuestro',
'vuestra',
'vuestros',
'vuestras',
'esos',
'esas',
'estoy',
'estás',
'está',
```

```
'estamos',
'estáis',
'están',
'esté',
'estés',
'estemos',
'estéis',
'estén',
'estaré',
'estarás',
'estará',
'estaremos',
'estaréis',
'estarán',
'estaría',
'estarías',
'estaríamos',
'estaríais',
'estarían',
'estaba',
'estabas',
'estábamos',
'estabais',
'estaban',
'estuve',
'estuviste',
'estuvo',
'estuvimos',
'estuvisteis',
'estuvieron',
'estuviera',
'estuvieras',
'estuviéramos',
'estuvierais',
'estuvieran',
'estuviese',
'estuvieses',
'estuviésemos',
'estuvieseis',
'estuviesen',
'estando',
'estado',
'estada',
'estados',
'estadas',
'estad',
'he',
'has',
'ha',
'hemos',
'habéis',
'han',
'haya',
'hayas',
'hayamos',
'hayáis',
'hayan',
'habré',
'habrás',
'habrá',
```

```
                'habremos',
                'habréis',
                'habrán',
                'habría',
                'habrías',
                'habríamos',
                'habríais',
                'habrían',
                'había',
                'habías',
                'habíamos',
                'habíais',
                'habían',
                'hube',
                'hubiste',
                'hubo',
                'hubimos',
                'hubisteis',
                'hubieron',
                'hubiera',
                'hubieras',
                'hubiéramos',
                'hubierais',
                'hubieran',
                'hubiese',
                'hubieses',
                'hubiésemos',
                'hubieseis',
                'hubiesen',
                'habiendo',
                'habido',
                'habida',
                'habidos',
                'habidas',
                'soy',
                'eres',
                'es',
                'somos',
                'sois',
                'son',
                'sea',
                'seas',
                'seamos',
                'seáis',
                'sean',
                'seré',
                'serás',
                'será',
                'seremos',
                'seréis',
                'serán',
                'sería',
                'serías',
                'seríamos',
                'seríais',
                'serían',
                'era',
                'eras',
                'éramos',
                'erais',
```

```
'eran',
'fui',
'fuiste',
'fue',
'fuimos',
'fuisteis',
'fueron',
'fuera',
'fueras',
'fuéramos',
'fuerais',
'fueran',
'fuese',
'fueses',
'fuésemos',
'fueseis',
'fuesen',
'sintiendo',
'sentido',
'sentida',
'sentidos',
'sentidas',
'siente',
'sentid',
'tengo',
'tienes',
'tiene',
'tenemos',
'tenéis',
'tienen',
'tenga',
'tengas',
'tengamos',
'tengáis',
'tengan',
'tendré',
'tendrás',
'tendrá',
'tendremos',
'tendréis',
'tendrán',
'tendría',
'tendrías',
'tendríamos',
'tendríais',
'tendrían',
'tenía',
'tenías',
'teníamos',
'teníais',
'tenían',
'tuve',
'tuviste',
'tuvo',
'tuvimos',
'tuvisteis',
'tuvieron',
'tuviera',
'tuvieras',
'tuviéramos',
```

```
 'tuvierais',
 'tuvieran',
 'tuviese',
 'tuvieses',
 'tuviésemos',
 'tuvieseis',
 'tuviesen',
 'teniendo',
 'tenido',
 'tenida',
 'tenidos',
 'tenidas',
 'tened']
```

In [53]:
```
len(stopwords.words('spanish'))
```

Out[53]: 313

In [54]:
```
stopwords.words('french')
```

Out[54]:
```
['au',
 'aux',
 'avec',
 'ce',
 'ces',
 'dans',
 'de',
 'des',
 'du',
 'elle',
 'en',
 'et',
 'eux',
 'il',
 'ils',
 'je',
 'la',
 'le',
 'les',
 'leur',
 'lui',
 'ma',
 'mais',
 'me',
 'même',
 'mes',
 'moi',
 'mon',
 'ne',
 'nos',
 'notre',
 'nous',
 'on',
 'ou',
 'par',
 'pas',
 'pour',
 'qu',
 'que',
 'qui',
 'sa',
 'se',
 'ses',
 'son',
 'sur',
 'ta',
 'te',
 'tes',
 'toi',
 'ton',
 'tu',
 'un',
 'une',
 'vos',
 'votre',
 'vous',
 'c',
 'd',
 'j',
 'l',
```

```
'à',
'm',
'n',
's',
't',
'y',
'été',
'étée',
'étées',
'étés',
'étant',
'étante',
'étants',
'étantes',
'suis',
'es',
'est',
'sommes',
'êtes',
'sont',
'serai',
'seras',
'sera',
'serons',
'serez',
'seront',
'serais',
'serait',
'serions',
'seriez',
'seraient',
'étais',
'était',
'étions',
'étiez',
'étaient',
'fus',
'fut',
'fûmes',
'fûtes',
'furent',
'sois',
'soit',
'soyons',
'soyez',
'soient',
'fusse',
'fusses',
'fût',
'fussions',
'fussiez',
'fussent',
'ayant',
'ayante',
'ayantes',
'ayants',
'eu',
'eue',
'eues',
'eus',
```

```
 'ai',
 'as',
 'avons',
 'avez',
 'ont',
 'aurai',
 'auras',
 'aura',
 'aurons',
 'aurez',
 'auront',
 'aurais',
 'aurait',
 'aurions',
 'auriez',
 'auraient',
 'avais',
 'avait',
 'avions',
 'aviez',
 'avaient',
 'eut',
 'eûmes',
 'eûtes',
 'eurent',
 'aie',
 'aies',
 'ait',
 'ayons',
 'ayez',
 'aient',
 'eusse',
 'eusses',
 'eût',
 'eussions',
 'eussiez',
 'eussent']
```

In [55]:
```
len(stopwords.words('french'))
```

Out[55]: 157

In [56]:
```
stopwords.words('german')
```

Out[56]:
```
['aber',
 'alle',
 'allem',
 'allen',
 'aller',
 'alles',
 'als',
 'also',
 'am',
 'an',
 'ander',
 'andere',
 'anderem',
 'anderen',
 'anderer',
 'anderes',
 'anderm',
 'andern',
 'anderr',
 'anders',
 'auch',
 'auf',
 'aus',
 'bei',
 'bin',
 'bis',
 'bist',
 'da',
 'damit',
 'dann',
 'der',
 'den',
 'des',
 'dem',
 'die',
 'das',
 'dass',
 'daß',
 'derselbe',
 'derselben',
 'denselben',
 'desselben',
 'demselben',
 'dieselbe',
 'dieselben',
 'dasselbe',
 'dazu',
 'dein',
 'deine',
 'deinem',
 'deinen',
 'deiner',
 'deines',
 'denn',
 'derer',
 'dessen',
 'dich',
 'dir',
 'du',
 'dies',
```

```
'diese',
'diesem',
'diesen',
'dieser',
'dieses',
'doch',
'dort',
'durch',
'ein',
'eine',
'einem',
'einen',
'einer',
'eines',
'einig',
'einige',
'einigem',
'einigen',
'einiger',
'einiges',
'einmal',
'er',
'ihn',
'ihm',
'es',
'etwas',
'euer',
'eure',
'eurem',
'euren',
'eurer',
'eures',
'für',
'gegen',
'gewesen',
'hab',
'habe',
'haben',
'hat',
'hatte',
'hatten',
'hier',
'hin',
'hinter',
'ich',
'mich',
'mir',
'ihr',
'ihre',
'ihrem',
'ihren',
'ihrer',
'ihres',
'euch',
'im',
'in',
'indem',
'ins',
'ist',
'jede',
```

```
                'jedem',
                'jeden',
                'jeder',
                'jedes',
                'jene',
                'jenem',
                'jenen',
                'jener',
                'jenes',
                'jetzt',
                'kann',
                'kein',
                'keine',
                'keinem',
                'keinen',
                'keiner',
                'keines',
                'können',
                'könnte',
                'machen',
                'man',
                'manche',
                'manchem',
                'manchen',
                'mancher',
                'manches',
                'mein',
                'meine',
                'meinem',
                'meinen',
                'meiner',
                'meines',
                'mit',
                'muss',
                'musste',
                'nach',
                'nicht',
                'nichts',
                'noch',
                'nun',
                'nur',
                'ob',
                'oder',
                'ohne',
                'sehr',
                'sein',
                'seine',
                'seinem',
                'seinen',
                'seiner',
                'seines',
                'selbst',
                'sich',
                'sie',
                'ihnen',
                'sind',
                'so',
                'solche',
                'solchem',
                'solchen',
```

```
 'solcher',
 'solches',
 'soll',
 'sollte',
 'sondern',
 'sonst',
 'über',
 'um',
 'und',
 'uns',
 'unsere',
 'unserem',
 'unseren',
 'unser',
 'unseres',
 'unter',
 'viel',
 'vom',
 'von',
 'vor',
 'während',
 'war',
 'waren',
 'warst',
 'was',
 'weg',
 'weil',
 'weiter',
 'welche',
 'welchem',
 'welchen',
 'welcher',
 'welches',
 'wenn',
 'werde',
 'werden',
 'wie',
 'wieder',
 'will',
 'wir',
 'wird',
 'wirst',
 'wo',
 'wollen',
 'wollte',
 'würde',
 'würden',
 'zu',
 'zum',
 'zur',
 'zwar',
 'zwischen']
```

In [57]:
```
len(stopwords.words('german'))
```

Out[57]: 232

In [58]:
```
stopwords.words('hindi') # research phase
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
Cell In[58], line 1
----> 1 stopwords.words('hindi')

File ~\anaconda3\Lib\site-packages\nltk\corpus\reader\wordlist.py:21, in WordList
CorpusReader.words(self, fileids, ignore_lines_startswith)
     18 def words(self, fileids=None, ignore_lines_startswith="\n"):
     19     return [
     20         line
---> 21         for line in line_tokenize(self.raw(fileids))
     22         if not line.startswith(ignore_lines_startswith)
     23     ]

File ~\anaconda3\Lib\site-packages\nltk\corpus\reader\api.py:218, in CorpusReade
r.raw(self, fileids)
    216 contents = []
    217 for f in fileids:
--> 218     with self.open(f) as fp:
    219         contents.append(fp.read())
    220 return concat(contents)

File ~\anaconda3\Lib\site-packages\nltk\corpus\reader\api.py:231, in CorpusReade
r.open(self, file)
    223 """
    224 Return an open stream that can be used to read the given file.
    225 If the file's encoding is not None, then the stream will
   (...)
    228 :param file: The file identifier of the file to read.
    229 """
    230 encoding = self.encoding(file)
--> 231 stream = self._root.join(file).open(encoding)
    232 return stream

File ~\anaconda3\Lib\site-packages\nltk\data.py:333, in FileSystemPathPointer.joi
n(self, fileid)
    331 def join(self, fileid):
    332     _path = os.path.join(self._path, fileid)
--> 333     return FileSystemPathPointer(_path)

File ~\anaconda3\Lib\site-packages\nltk\data.py:311, in FileSystemPathPointer.__i
nit__(self, _path)
    309 _path = os.path.abspath(_path)
    310 if not os.path.exists(_path):
--> 311     raise OSError("No such file or directory: %r" % _path)
    312 self._path = _path

OSError: No such file or directory: 'C:\\Users\\shaik\\AppData\\Roaming\\nltk_dat
a\\corpora\\stopwords\\hindi'
```

In [59]:
```
stopwords.words('marathi')
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
Cell In[59], line 1
----> 1 stopwords.words('marathi')

File ~\anaconda3\Lib\site-packages\nltk\corpus\reader\wordlist.py:21, in WordList
CorpusReader.words(self, fileids, ignore_lines_startswith)
     18 def words(self, fileids=None, ignore_lines_startswith="\n"):
     19     return [
     20         line
---> 21         for line in line_tokenize(self.raw(fileids))
     22         if not line.startswith(ignore_lines_startswith)
     23     ]

File ~\anaconda3\Lib\site-packages\nltk\corpus\reader\api.py:218, in CorpusReade
r.raw(self, fileids)
    216 contents = []
    217 for f in fileids:
--> 218     with self.open(f) as fp:
    219         contents.append(fp.read())
    220 return concat(contents)

File ~\anaconda3\Lib\site-packages\nltk\corpus\reader\api.py:231, in CorpusReade
r.open(self, file)
    223 """
    224 Return an open stream that can be used to read the given file.
    225 If the file's encoding is not None, then the stream will
   (...)
    228 :param file: The file identifier of the file to read.
    229 """
    230 encoding = self.encoding(file)
--> 231 stream = self._root.join(file).open(encoding)
    232 return stream

File ~\anaconda3\Lib\site-packages\nltk\data.py:333, in FileSystemPathPointer.joi
n(self, fileid)
    331 def join(self, fileid):
    332     _path = os.path.join(self._path, fileid)
--> 333     return FileSystemPathPointer(_path)

File ~\anaconda3\Lib\site-packages\nltk\data.py:311, in FileSystemPathPointer.__i
nit__(self, _path)
    309 _path = os.path.abspath(_path)
    310 if not os.path.exists(_path):
--> 311     raise OSError("No such file or directory: %r" % _path)
    312 self._path = _path

OSError: No such file or directory: 'C:\\Users\\shaik\\AppData\\Roaming\\nltk_dat
a\\corpora\\stopwords\\marathi'
```

In [60]:
```
# first we need to compile from re module to create string that matched any digi
import re
punctuation = re.compile(r'[-.?!,:;()|0-9]')

#now i am going to create to empty list and append the word without any punctuat
```

In [61]:
```
punctuation
```

Out[61]:
```
re.compile(r'[-.?!,:;()|0-9]', re.UNICODE)
```

In [62]:
```
AI
```

Out[62]:
```
'Artificial Intelligence refers to the intelligence of machines. This is in con
trast to the natural intelligence of \nhumans and animals. With Artificial Inte
lligence, machines perform functions such as learning, planning, reasoning and
\nproblem-solving. Most noteworthy, Artificial Intelligence is the simulation o
f human intelligence by machines. \nIt is probably the fastest-growing developm
ent in the World of technology and innovation. Furthermore, many experts believ
e\nAI could solve major challenges and crisis situations.'
```

In [63]:
```
AI_tokens
```

Out[63]:
```
['Artificial',
 'Intelligence',
 'refers',
 'to',
 'the',
 'intelligence',
 'of',
 'machines',
 '.',
 'This',
 'is',
 'in',
 'contrast',
 'to',
 'the',
 'natural',
 'intelligence',
 'of',
 'humans',
 'and',
 'animals',
 '.',
 'With',
 'Artificial',
 'Intelligence',
 ',',
 'machines',
 'perform',
 'functions',
 'such',
 'as',
 'learning',
 ',',
 'planning',
 ',',
 'reasoning',
 'and',
 'problem-solving',
 '.',
 'Most',
 'noteworthy',
 ',',
 'Artificial',
 'Intelligence',
 'is',
 'the',
 'simulation',
 'of',
 'human',
 'intelligence',
 'by',
 'machines',
 '.',
 'It',
 'is',
 'probably',
 'the',
 'fastest-growing',
 'development',
 'in',
```

```
 'the',
 'World',
 'of',
 'technology',
 'and',
 'innovation',
 '.',
 'Furthermore',
 ',',
 'many',
 'experts',
 'believe',
 'AI',
 'could',
 'solve',
 'major',
 'challenges',
 'and',
 'crisis',
 'situations',
 '.']
```

In [64]:
```
len(AI_tokens)
```

Out[64]: 81

In [65]:
```
    #POS [part of sppech] is always talking about grammaticaly type of the word

#how the word will function in grammatically within the sentence, a word can hav

#so lets see some pos tags & description, so pos tags are usualy used to descrie

#in this slide we have so many tags along with their description with different

#this tags are beginning from coordinating conjunction to whadverb & lets unders

#next we will see how we will implement this POS in our text
```

In [66]:
```
# we will see how to work in POS using NLTK library

sent = 'kathy is a natural when it comes to drawing'
sent_tokens = word_tokenize(sent)
sent_tokens

# first we will tokenize usning word_tokenize & then we will use pos_tag on all
```

Out[66]: ['kathy', 'is', 'a', 'natural', 'when', 'it', 'comes', 'to', 'drawing']

In [67]:
```
for token in sent_tokens:
    print(nltk.pos_tag([token]))
```

```
[('kathy', 'NN')]
[('is', 'VBZ')]
[('a', 'DT')]
[('natural', 'JJ')]
[('when', 'WRB')]
[('it', 'PRP')]
[('comes', 'VBZ')]
[('to', 'TO')]
[('drawing', 'VBG')]
```

In [68]:
```
sent2 = 'john is eating a delicious cake'
sent2_tokens = word_tokenize(sent2)

for token in sent2_tokens:
    print(nltk.pos_tag([token]))
```

```
[('john', 'NN')]
[('is', 'VBZ')]
[('eating', 'VBG')]
[('a', 'DT')]
[('delicious', 'JJ')]
[('cake', 'NN')]
```

In [69]:
```
# Another concept of POS is called NER ( NAMED ENTITIY RECOGNITION ), NER is the
# there are 3 phases of NER - ( 1ST PHASE IS - NOUN PHRASE EXTRACTION OR NOUN PH
# 2nd step we have phrase classification - this is the classification where all
# some times entity are misclassification
# so if you are use NER in python then you need to import NER_CHUNK from nltk li
```

In [70]:
```
from nltk import ne_chunk
```

In [71]:
```
NE_sent = 'The US president stays in the WHITEHOUSE '
```

In [72]:
```
# IN NLTK also we have syntax- set of rules,principals & process
# lets understand set of rules & that will indicates the syntax tree & in the re

# now lets understand the important concept called CHUNKING using the sentence s
# chunking means grouping of words into chunks & lets understand the example of
# chunking will help to easy process the data
```

In [73]:
```
NE_tokens = word_tokenize(NE_sent)

#after tokenize need to add the pos tags
NE_tokens
```

Out[73]:
```
['The', 'US', 'president', 'stays', 'in', 'the', 'WHITEHOUSE']
```

In [74]:
```
NE_tags = nltk.pos_tag(NE_tokens)
NE_tags
```

Out[74]:
```
[('The', 'DT'),
 ('US', 'NNP'),
 ('president', 'NN'),
 ('stays', 'NNS'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('WHITEHOUSE', 'NNP')]
```

In [75]:
```
#we are passin the NE_NER into ne_chunks function and lets see the outputs

NE_NER = ne_chunk(NE_tags)
print(NE_NER)
```

```
(S
  The/DT
  (GSP US/NNP)
  president/NN
  stays/NNS
  in/IN
  the/DT
  (ORGANIZATION WHITEHOUSE/NNP))
```

In [76]:
```
new = 'the big cat ate the little mouse who was after fresh cheese'
new_tokens = nltk.pos_tag(word_tokenize(new))
new_tokens

# tokenize done and lets add the pos tags also
```

Out[76]:
```
[('the', 'DT'),
 ('big', 'JJ'),
 ('cat', 'NN'),
 ('ate', 'VBD'),
 ('the', 'DT'),
 ('little', 'JJ'),
 ('mouse', 'NN'),
 ('who', 'WP'),
 ('was', 'VBD'),
 ('after', 'IN'),
 ('fresh', 'JJ'),
 ('cheese', 'NN')]
```

In [77]:
```
# Libraries
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

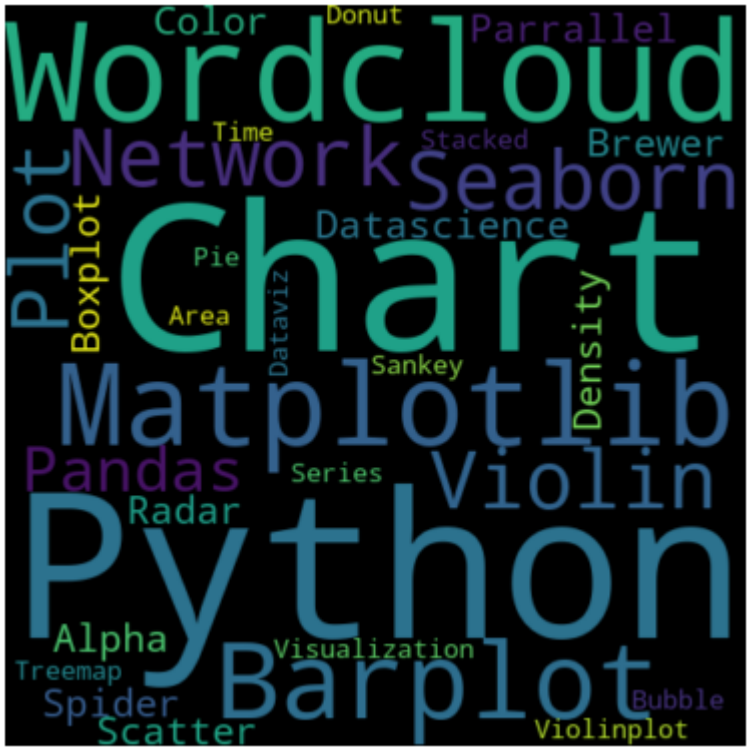
In [78]:
```
# Create a list of word
text=("Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin Ch
```

In [79]:
```
text
```

Out[79]:
```
'Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin Chart P
andas Datascience Wordcloud Spider Radar Parrallel Alpha Color Brewer Density S
catter Barplot Barplot Boxplot Violinplot Treemap Stacked Area Chart Chart Visu
alization Dataviz Donut Pie Time-Series Wordcloud Wordcloud Sankey Bubble'
```

In [80]:
```
# Create the wordcloud object
wordcloud = WordCloud(width=480, height=480, margin=0).generate(text)
```

In [84]:
```
# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()
```


Wordcloud
Chart
Python
Matplotlib
Barplot
Network
Seaborn
Plot
Violin
Pandas
Color
Donut
Parrallel
Time
Stacked
Brewer
Boxplot
Datascience
Pie
Area
Dataviz
Density
Sankey
Series
Radar
Alpha
Visualization
Treemap
Spider
Scatter
Bubble
Violinplot

In [ ]: