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

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

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

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

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

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

In [ ]:
```
#nltk.corpus.gutenberg
```

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

In [2]:
```
# you can also create your own words

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

In [3]:
```
AI
```

Out[3]:
```
'Artificial Intelligence refers to the intelligence of machines. This is in contra
st to the natural intelligence of \nhumans and animals. With Artificial Intelligen
ce, machines perform functions such as learning, planning, reasoning and \nproblem
-solving. Most noteworthy, Artificial Intelligence is the simulation of human inte
lligence by machines. \nIt is probably the fastest-growing development in the Worl
d of technology and innovation. Furthermore, many experts believe\nAI could solve
major challenges and crisis situations.'
```

In [4]:
```
type(AI)
```

Out[4]:
```
str
```

In [5]:
```
from nltk.tokenize import word_tokenize
```

In [6]:
```
AI_tokens = word_tokenize(AI)
AI_tokens
```

Out[6]:
```
['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 [7]:
```
len(AI_tokens)
```

Out[7]:
```
81
```

In [8]:
```
from nltk.tokenize import sent_tokenize
```

In [9]:
```
AI_sent = sent_tokenize(AI)
AI_sent
```

Out[9]:
```
['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, plann
ing, reasoning and \nproblem-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 in
novation.',
 'Furthermore, many experts believe\nAI could solve major challenges and crisis si
tuations.']
```

In [10]:
```
len(AI_sent)
```

Out[10]:
```
6
```

In [11]:
```
AI
```

Out[11]:
```
'Artificial Intelligence refers to the intelligence of machines. This is in contra
st to the natural intelligence of \nhumans and animals. With Artificial Intelligen
ce, machines perform functions such as learning, planning, reasoning and \nproblem
-solving. Most noteworthy, Artificial Intelligence is the simulation of human inte
lligence by machines. \nIt is probably the fastest-growing development in the Worl
d of technology and innovation. Furthermore, many experts believe\nAI could solve
major challenges and crisis situations.'
```

In [12]:
```
from nltk.tokenize import blankline_tokenize # GiVE YOU HOW MANY PARAGRAPH
AI_blank = blankline_tokenize(AI)
AI_blank
#AI_blank
```

Out[12]:
```
['Artificial Intelligence refers to the intelligence of machines. This is in contr
ast to the natural intelligence of \nhumans and animals. With Artificial Intellige
nce, machines perform functions such as learning, planning, reasoning and \nproble
m-solving. Most noteworthy, Artificial Intelligence is the simulation of human int
elligence by machines. \nIt is probably the fastest-growing development in the Wor
ld of technology and innovation. Furthermore, many experts believe\nAI could solve
major challenges and crisis situations.']
```

In [13]:
```
len(AI_blank)
```

Out[13]:
```
1
```

In [14]:
```
# NEXT WE WILL SEE HOW WE WILL USE UNI-GRAM,BI-GRAM,TRI-GRAM USING NLTK

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

In [15]:
```
string = 'the best and most beautifull thing in the world cannot be seen or even to
quotes_tokens = nltk.word_tokenize(string)
```

In [16]:
```
quotes_tokens
```

Out[16]:
```
['the',
 'best',
 'and',
 'most',
 'beautifull',
 'thing',
 'in',
 'the',
 'world',
 'can',
 'not',
 'be',
 'seen',
 'or',
 'even',
 'touched',
 ',',
 'they',
 'must',
 'be',
 'felt',
 'with',
 'heart']
```

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

Out[17]:
```
23
```

In [18]:
```
quotes_bigrams = list(nltk.bigrams(quotes_tokens))
quotes_bigrams
```

Out[18]:
```
[('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 [22]:
```
quotes_tokens
```

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

In [23]:
```
quotes_trigrams = list(nltk.trigrams(quotes_tokens))
quotes_trigrams
```

Out[23]:
```
[('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 [19]:
```
quotes_trigrams = list(nltk.ngrams(quotes_tokens))
quotes_trigrams
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-a46038d19d4e> in <module>
----> 1 quotes_trigrams = list(nltk.ngrams(quotes_tokens))
      2 quotes_trigrams

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

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

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

In [20]:
```
len(quotes_tokens)
```

Out[20]: 23

In [21]:
```
quotes_ngrams_1 = list(nltk.ngrams(quotes_tokens, 5))
quotes_ngrams_1
```

Out[21]:
```
[('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 [22]:
```
quotes_ngrams = list(nltk.ngrams(quotes_tokens, 9))
quotes_ngrams
```

Out[22]:
```
[('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 [27]:
```
# Next we need to make some changes in tokens and that is called as stemming, stemm
# also we will see some root form of the word & limitation of the word

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

In [28]:
```
pst.stem('having') #stem will gives you the root form of the word
```

Out[28]:
```
'have'
```

In [29]:
```
pst.stem('affection')
```

Out[29]:
```
'affect'
```

In [30]:
```
pst.stem('playing')
```

Out[30]:
```
'play'
```

In [31]:
```
pst.stem('give')
```

Out[31]:
```
'give'
```

In [32]:
```
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 [33]:
```
pst.stem('playing')
```

Out[33]:
```
'play'
```

In [34]:
```
words_to_stem=['give','giving','given','gave','thinking', 'loving', 'final', 'final
# i am giving these different words to stem, using porter stemmer we get the output
```

```
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 [35]:
```
#another stemmer known as lencastemmer stemmer and lets see what the different we 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 [36]:
```
words_to_stem=['give','giving','given','gave','thinking', 'loving', 'final', 'final
# i am giving these different words to stem, using porter stemmer we get the output

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 [37]:
```
#we have another stemmer called as snowball stemmer lets see about this snowball st

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 lancaster
```

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

In [40]:
```
#sometime stemming does not work & lets say e.g - fish,fishes & fishing all of them
#one hand stemming will cut the end & lemmatization will take into the morphologica

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 lemat
```

In [39]:
```
words_to_stem
```

Out[39]:
```
['give',
 'giving',
 'given',
 'gave',
 'thinking',
 'loving',
 'final',
 'finalized',
 'finally']
```

In [41]:
```
#word_lem.lemmatize('corpora') #we get output as corpus

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

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 [42]:
```
pst.stem('final')
```

Out[42]: 'final'

In [43]:
```
lst.stem('finally')
```

Out[43]: 'fin'

In [44]:
```
sbst.stem('finalized')
```

Out[44]: 'final'

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

Out[45]: 'fin'

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

Out[46]: 'fin'

In [47]:
```
# there is other concept called POS (part of speech) which deals with subject, noun
# STOPWORDS = i, is, as,at, on, about & nltk has their own list of stopewords

from nltk.corpus import stopwords
```

In [48]:
```
stopwords.words('english')
```

Out[48]:
```
['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',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
 'who',
 'whom',
 'this',
 'that',
 "that'll",
 'these',
 'those',
 'am',
 'is',
 'are',
 'was',
 'were',
 'be',
 'been',
 'being',
 'have',
 'has',
 'had',
 'having',
```

```
'do',
'does',
'did',
'doing',
'a',
'an',
'the',
'and',
'but',
'if',
'or',
'because',
'as',
'until',
'while',
'of',
'at',
'by',
'for',
'with',
'about',
'against',
'between',
'into',
'through',
'during',
'before',
'after',
'above',
'below',
'to',
'from',
'up',
'down',
'in',
'out',
'on',
'off',
'over',
'under',
'again',
'further',
'then',
'once',
'here',
'there',
'when',
'where',
'why',
'how',
'all',
'any',
'both',
'each',
'few',
'more',
```

```
'most',
'other',
'some',
'such',
'no',
'nor',
'not',
'only',
'own',
'same',
'so',
'than',
'too',
'very',
's',
't',
'can',
'will',
'just',
'don',
"don't",
'should',
"should've",
'now',
'd',
'll',
'm',
'o',
're',
've',
'y',
'ain',
'aren',
"aren't",
'couldn',
"couldn't",
'didn',
"didn't",
'doesn',
"doesn't",
'hadn',
"hadn't",
'hasn',
"hasn't",
'haven',
"haven't",
'isn',
"isn't",
'ma',
'mightn',
"mightn't",
'mustn',
"mustn't",
'needn',
"needn't",
'shan',
```

```
 "shan't",
 'shouldn',
 "shouldn't",
 'wasn',
 "wasn't",
 'weren',
 "weren't",
 'won',
 "won't",
 'wouldn',
 "wouldn't"]
```

In [49]:
```
len(stopwords.words('english'))
```

Out[49]: 179

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

Out[50]:
```
['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 [51]:
```
len(stopwords.words('spanish'))
```

Out[51]:
```
313
```

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

Out[52]:
```
['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 [53]:
```
len(stopwords.words('french'))
```

Out[53]: 157

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

Out[54]:
```
['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 [55]:
```
len(stopwords.words('german'))
```

Out[55]:
```
232
```

In [56]:
```
stopwords.words('hindi') # research phase
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
<ipython-input-56-df50742b7e1b> in <module>
----> 1 stopwords.words('hindi') # research phase

~\anaconda3\lib\site-packages\nltk\corpus\reader\wordlist.py in words(self, fileids,
ignore_lines_startswith)
     23         return [
     24             line
---> 25             for line in line_tokenize(self.raw(fileids))
     26             if not line.startswith(ignore_lines_startswith)
     27         ]

~\anaconda3\lib\site-packages\nltk\corpus\reader\wordlist.py in raw(self, fileids)
     32         elif isinstance(fileids, string_types):
     33             fileids = [fileids]
---> 34         return concat([self.open(f).read() for f in fileids])
     35
     36

~\anaconda3\lib\site-packages\nltk\corpus\reader\wordlist.py in <listcomp>(.0)
     32         elif isinstance(fileids, string_types):
     33             fileids = [fileids]
---> 34         return concat([self.open(f).read() for f in fileids])
     35
     36

~\anaconda3\lib\site-packages\nltk\corpus\reader\api.py in open(self, file)
    211         """
    212         encoding = self.encoding(file)
--> 213         stream = self._root.join(file).open(encoding)
    214         return stream
    215

~\anaconda3\lib\site-packages\nltk\data.py in join(self, fileid)
    353     def join(self, fileid):
    354         _path = os.path.join(self._path, fileid)
--> 355         return FileSystemPathPointer(_path)
    356
    357     def __repr__(self):

~\anaconda3\lib\site-packages\nltk\compat.py in _decorator(*args, **kwargs)
    226     def _decorator(*args, **kwargs):
    227         args = (args[0], add_py3_data(args[1])) + args[2:]
--> 228         return init_func(*args, **kwargs)
    229
    230     return wraps(init_func)(_decorator)

~\anaconda3\lib\site-packages\nltk\data.py in __init__(self, _path)
    331         _path = os.path.abspath(_path)
    332         if not os.path.exists(_path):
--> 333             raise IOError('No such file or directory: %r' % _path)
    334         self._path = _path
    335
```

```
OSError: No such file or directory: 'C:\\Users\\kdata\\AppData\\Roaming\\nltk_data
\\corpora\\stopwords\\hindi'
```

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

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
<ipython-input-57-4f60cd586bb5> in <module>
----> 1 stopwords.words('marathi')

~\anaconda3\lib\site-packages\nltk\corpus\reader\wordlist.py in words(self, fileids,
ignore_lines_startswith)
     23         return [
     24             line
---> 25             for line in line_tokenize(self.raw(fileids))
     26             if not line.startswith(ignore_lines_startswith)
     27         ]

~\anaconda3\lib\site-packages\nltk\corpus\reader\wordlist.py in raw(self, fileids)
     32         elif isinstance(fileids, string_types):
     33             fileids = [fileids]
---> 34         return concat([self.open(f).read() for f in fileids])
     35
     36

~\anaconda3\lib\site-packages\nltk\corpus\reader\wordlist.py in <listcomp>(.0)
     32         elif isinstance(fileids, string_types):
     33             fileids = [fileids]
---> 34         return concat([self.open(f).read() for f in fileids])
     35
     36

~\anaconda3\lib\site-packages\nltk\corpus\reader\api.py in open(self, file)
    211         """
    212         encoding = self.encoding(file)
--> 213         stream = self._root.join(file).open(encoding)
    214         return stream
    215

~\anaconda3\lib\site-packages\nltk\data.py in join(self, fileid)
    353     def join(self, fileid):
    354         _path = os.path.join(self._path, fileid)
--> 355         return FileSystemPathPointer(_path)
    356
    357     def __repr__(self):

~\anaconda3\lib\site-packages\nltk\compat.py in _decorator(*args, **kwargs)
    226     def _decorator(*args, **kwargs):
    227         args = (args[0], add_py3_data(args[1])) + args[2:]
--> 228         return init_func(*args, **kwargs)
    229
    230     return wraps(init_func)(_decorator)

~\anaconda3\lib\site-packages\nltk\data.py in __init__(self, _path)
    331         _path = os.path.abspath(_path)
    332         if not os.path.exists(_path):
--> 333             raise IOError('No such file or directory: %r' % _path)
    334         self._path = _path
    335
```

```
OSError: No such file or directory: 'C:\\Users\\kdata\\AppData\\Roaming\\nltk_data
\\corpora\\stopwords\\marathi'
```

In [58]:
```
stopwords.words('telugu')
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
<ipython-input-58-d1b3db3d8785> in <module>
----> 1 stopwords.words('telugu')

~\anaconda3\lib\site-packages\nltk\corpus\reader\wordlist.py in words(self, fileids,
ignore_lines_startswith)
     23         return [
     24             line
---> 25             for line in line_tokenize(self.raw(fileids))
     26             if not line.startswith(ignore_lines_startswith)
     27         ]

~\anaconda3\lib\site-packages\nltk\corpus\reader\wordlist.py in raw(self, fileids)
     32         elif isinstance(fileids, string_types):
     33             fileids = [fileids]
---> 34         return concat([self.open(f).read() for f in fileids])
     35
     36

~\anaconda3\lib\site-packages\nltk\corpus\reader\wordlist.py in <listcomp>(.0)
     32         elif isinstance(fileids, string_types):
     33             fileids = [fileids]
---> 34         return concat([self.open(f).read() for f in fileids])
     35
     36

~\anaconda3\lib\site-packages\nltk\corpus\reader\api.py in open(self, file)
    211         """
    212         encoding = self.encoding(file)
--> 213         stream = self._root.join(file).open(encoding)
    214         return stream
    215

~\anaconda3\lib\site-packages\nltk\data.py in join(self, fileid)
    353     def join(self, fileid):
    354         _path = os.path.join(self._path, fileid)
--> 355         return FileSystemPathPointer(_path)
    356
    357     def __repr__(self):

~\anaconda3\lib\site-packages\nltk\compat.py in _decorator(*args, **kwargs)
    226     def _decorator(*args, **kwargs):
    227         args = (args[0], add_py3_data(args[1])) + args[2:]
--> 228         return init_func(*args, **kwargs)
    229
    230     return wraps(init_func)(_decorator)

~\anaconda3\lib\site-packages\nltk\data.py in __init__(self, _path)
    331         _path = os.path.abspath(_path)
    332         if not os.path.exists(_path):
--> 333             raise IOError('No such file or directory: %r' % _path)
    334         self._path = _path
    335
```

```
OSError: No such file or directory: 'C:\\Users\\kdata\\AppData\\Roaming\\nltk_data
\\corpora\\stopwords\\telugu'
```

In [59]:
```
# first we need to compile from re module to create string that matched any digits
import re
punctuation = re.compile(r'[-.?!,:;()|0-9]')

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

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

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

In [ ]:
```
AI
```

In [ ]:
```
AI_tokens
```

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

#POS [part of sppech] is always talking about grammaticaly type of the word called verbs, noun, adjective, proverb, #how the word will function in grammatically within the sentence, a word can have more then one pos based on context in which it will use #so lets see some pos tags & description, so pos tags are usualy used to descrie weather te word is used for noun,adjective,pronoun, propernoun, singular, plural, is it symbol or is it adverb #in this slide we have so many tags along with their description with different tags #this tags are beginning from coordinating conjunction to whadverb & lets understand about one of the example #next we will see how we will implement this POS in our text

In [62]:
```
# 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 of
```

Out[62]:
```
['kathy', 'is', 'a', 'natural', 'when', 'it', 'comes', 'to', 'drawing']
```

In [63]:
```
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 [64]:
```
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 [ ]:
```
# Another concept of POS is called NER ( NAMED ENTITIY RECOGNITION ), NER is the pr
# there are 3 phases of NER - ( 1ST PHASE IS - NOUN PHRASE EXTRACTION OR NOUN PHASE
# 2nd step we have phrase classification - this is the classification where all the
# some times entity are misclassification
# so if you are use NER in python then you need to import NER_CHUNK from nltk libra
```

In [65]:
```
from nltk import ne_chunk
```

In [66]:
```
NE_sent = 'The US president stays in the WHITEHOUSE '
```

IN NLTK also we have syntax- set of rules,principals & process # lets understand set of rules & that will indicates the syntax tree & in the real time also you have build this type of tree from the sentenses # now lets understand the important concept called CHUNKING using the sentence structure # chunking means grouping of words into chunks & lets understand the example of chunking # chunking will help to easy process the data

In [67]:
```
NE_tokens = word_tokenize(NE_sent)

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

Out[67]:
```
['The', 'US', 'president', 'stays', 'in', 'the', 'WHITEHOUSE']
```

In [68]:
```
NE_tags = nltk.pos_tag(NE_tokens)
NE_tags
```

Out[68]:
```
[('The', 'DT'),
 ('US', 'NNP'),
 ('president', 'NN'),
 ('stays', 'NNS'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('WHITEHOUSE', 'NNP')]
```

In [69]:
```
#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 [70]:
```
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[70]:
```
[('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 [71]:
```
# Libraries
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

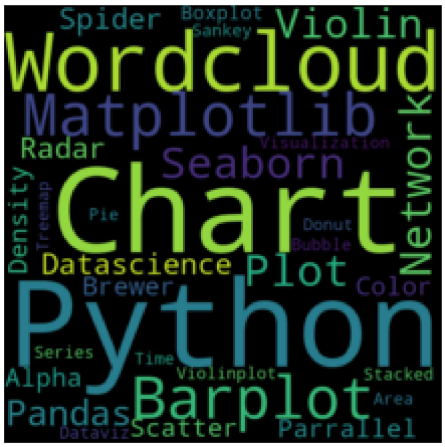
In [72]:
```
# Create a list of word
text=("Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin Chart
```

In [73]:
```
text
```

Out[73]:
```
'Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin Chart Pand
as Datascience Wordcloud Spider Radar Parrallel Alpha Color Brewer Density Scatter
Barplot Barplot Boxplot Violinplot Treemap Stacked Area Chart Chart Visualization
Dataviz Donut Pie Time-Series Wordcloud Wordcloud Sankey Bubble'
```

In [75]:
```
# Create the wordcloud object
wordcloud = WordCloud(width=480, height=480, margin=0).generate(text)
```

In [76]:
```
# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()
```



In [ ]:
```
```