NLP (Natural Language Processing)

we will start with nltk framework with NLU (Natural Language Understanding)

•We need to install it first

•To install nltk follow the below steps

In [4]:
```
import os
import nltk
#nltk.download()
```

In [5]:
```
import nltk.corpus
```

Tokenization

This process is used to create tokens of separate words, sentences, pararagraphs.

In [6]:
```
AI = '''Artificial Intelligence refers to the intelligence of machines. This is in contrast to the natural inte
humans and animals. With Artificial Intelligence, machines perform functions such as learning, planning, reason
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 exp
AI could solve major challenges and crisis situations.'''
```

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

Out[7]:
```
'Artificial Intelligence refers to the intelligence of machines. This is in contrast to the natural intelligenc
e of\nhumans and animals. With Artificial Intelligence, machines perform functions such as learning, planning,
reasoning and\nproblem-solving. Most noteworthy, Artificial Intelligence is the simulation of human intelligenc
e by machines.\nIt is probably the fastest-growing development in the World of technology and innovation. Furth
ermore, many experts believe\nAI could solve major challenges and crisis situations.'
```

In [8]:
```
#To check type
type(AI)
```

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

In [9]:
```
#To create each word as token use below module
#this module can create token of each word including full stop(.) and commas(,)

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]:
```
#To check length of token
```

```
len(AI_tokens)
```

Out[11]: 81

In [12]:
```
AI
```

Out[12]:
```
'Artificial Intelligence refers to the intelligence of machines. This is in contrast to the natural intelligenc
e of\nhumans and animals. With Artificial Intelligence, machines perform functions such as learning, planning,
reasoning and\nproblem-solving. Most noteworthy, Artificial Intelligence is the simulation of human intelligenc
e by machines.\nIt is probably the fastest-growing development in the World of technology and innovation. Furth
ermore, many experts believe\nAI could solve major challenges and crisis situations.'
```

In [13]:
```
#  This module is used to make each sentence as a separate token

from nltk.tokenize import sent_tokenize
```

In [14]:
```
AI_sent = sent_tokenize(AI)
AI_sent
```

Out[14]:
```
['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, planning, reasoning and\nproblem-s
olving.',
 '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\nAI could solve major challenges and crisis situations.']
```

In [15]:
```
len(AI_sent)
```

Out[15]: 6

In [16]:
```
AI
```

Out[16]:
```
'Artificial Intelligence refers to the intelligence of machines. This is in contrast to the natural intelligenc
e of\nhumans and animals. With Artificial Intelligence, machines perform functions such as learning, planning,
reasoning and\nproblem-solving. Most noteworthy, Artificial Intelligence is the simulation of human intelligenc
e by machines.\nIt is probably the fastest-growing development in the World of technology and innovation. Furth
ermore, many experts believe\nAI could solve major challenges and crisis situations.'
```

In [ ]:
```

```

In [17]:
```
#This module is used to make each paragraph as a new token

from nltk.tokenize import blankline_tokenize    #gives us how many paragraph are there
AI_blank = blankline_tokenize(AI)
AI_blank
```

Out[17]:
```
['Artificial Intelligence refers to the intelligence of machines. This is in contrast to the natural intelligen
ce of\nhumans and animals. With Artificial Intelligence, machines perform functions such as learning, planning,
reasoning and\nproblem-solving. Most noteworthy, Artificial Intelligence is the simulation of human intelligenc
e by machines.\nIt is probably the fastest-growing development in the World of technology and innovation. Furth
ermore, many experts believe\nAI could solve major challenges and crisis situations.']
```

In [18]:
```
len(AI_blank)
```

Out[18]: 1

In [19]:
```
# This module is used to create each word as a token without full stop(.) and commas(,)

from nltk.tokenize import WhitespaceTokenizer
wt = WhitespaceTokenizer().tokenize(AI)
wt
```

Out[19]:
```
['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 [20]:
```
len(wt)
```

Out[20]:
```
70
```

In [ ]:
```

```

In [21]:
```
a1 = 'Good apples cost @5 in hyderabad. please buy two of them. thank you.'
a1
```

Out[21]:
```
'Good apples cost @5 in hyderabad. please buy two of them. thank you.'
```

```

```

In [22]:
```
#this module is used to create each word, number, special character as a token

from nltk.tokenize import wordpunct_tokenize
A1 = wordpunct_tokenize(a1)
A1
```

Out[22]:
```
['Good',
 'apples',
 'cost',
 '@',
 '5',
 'in',
 'hyderabad',
 '.',
 'please',
 'buy',
 'two',
 'of',
 'them',
 '.',
 'thank',
 'you',
 '.']
```

In [23]:
```
len(A1)
```

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

In [ ]:

In [24]:
```
AI
```

Out[24]:
```
'Artificial Intelligence refers to the intelligence of machines. This is in contrast to the natural intelligenc
e of\nhumans and animals. With Artificial Intelligence, machines perform functions such as learning, planning,
reasoning and\nproblem-solving. Most noteworthy, Artificial Intelligence is the simulation of human intelligenc
e by machines.\nIt is probably the fastest-growing development in the World of technology and innovation. Furth
ermore, many experts believe\nAI could solve major challenges and crisis situations.'
```

In [25]:
```
AI1 = wordpunct_tokenize(AI)
AI1
```

Out[25]:
```
['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 [26]:
```
len(AI1)
```

Out[26]:
```
85
```

In [ ]:
```

```

In [27]:
```
#This module is used to create word as tokens in consecutive pairs of two, three and n formate

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

In [28]:
```
string = 'hello my name is nltk use me to manipulate text.'
string
```

Out[28]:
```
'hello my name is nltk use me to manipulate text.'
```

In [29]:
```
str1 = nltk.word_tokenize(string)
str1
```

Out[29]:
```
['hello',
 'my',
 'name',
 'is',
 'nltk',
 'use',
 'me',
 'to',
 'manipulate',
 'text',
 '.']
```

In [30]:
```
string
```

Out[30]:
```
'hello my name is nltk use me to manipulate text.'
```

In [31]:
```
str1
```

Out[31]:
```
['hello',
 'my',
 'name',
 'is',
 'nltk',
 'use',
 'me',
 'to',
 'manipulate',
 'text',
 '.']
```

In [32]:
```
len(str1)
```

Out[32]:
```
11
```

In [ ]:
```

```

In [33]:
```
#If used this module on a string without tokenization

str_bigrams = list(nltk.bigrams(string))
str_bigrams
```

Out[33]:
```
[('h', 'e'),
 ('e', 'l'),
 ('l', 'l'),
 ('l', 'o'),
 ('o', ' '),
 (' ', 'm'),
 ('m', 'y'),
 ('y', ' '),
 (' ', 'n'),
 ('n', 'a'),
 ('a', 'm'),
 ('m', 'e'),
 ('e', ' '),
 (' ', 'i'),
 ('i', 's'),
 ('s', ' '),
 (' ', 'n'),
 ('n', 'l'),
 ('l', 't'),
 ('t', 'k'),
 ('k', ' '),
 (' ', 'u'),
 ('u', 's'),
 ('s', 'e'),
 ('e', ' '),
 (' ', 'm'),
 ('m', 'e'),
 ('e', ' '),
 (' ', 't'),
 ('t', 'o'),
 ('o', ' '),
 (' ', 'm'),
 ('m', 'a'),
 ('a', 'n'),
 ('n', 'i'),
 ('i', 'p'),
 ('p', 'u'),
 ('u', 'l'),
 ('l', 'a'),
 ('a', 't'),
 ('t', 'e'),
 ('e', ' '),
 (' ', 't'),
 ('t', 'e'),
 ('e', 'x'),
 ('x', 't'),
 ('t', '.')]
```

In [34]:
```
str_bigrams1 = list(nltk.bigrams(str1))
str_bigrams1
```

Out[34]:
```
[('hello', 'my'),
 ('my', 'name'),
 ('name', 'is'),
 ('is', 'nltk'),
 ('nltk', 'use'),
 ('use', 'me'),
 ('me', 'to'),
 ('to', 'manipulate'),
 ('manipulate', 'text'),
 ('text', '.')]
```

In [35]:
```
str_trigrams = list(nltk.trigrams(str1))
str_trigrams
```

Out[35]:
```
[('hello', 'my', 'name'),
 ('my', 'name', 'is'),
 ('name', 'is', 'nltk'),
 ('is', 'nltk', 'use'),
 ('nltk', 'use', 'me'),
 ('use', 'me', 'to'),
 ('me', 'to', 'manipulate'),
 ('to', 'manipulate', 'text'),
 ('manipulate', 'text', '.')]
```

In [36]:
```
str_ngrams = list(nltk.ngrams(str1, 5))            #it create token in consecutive pairs as we want
str_ngrams
```

Out[36]:
```
[('hello', 'my', 'name', 'is', 'nltk'),
 ('my', 'name', 'is', 'nltk', 'use'),
 ('name', 'is', 'nltk', 'use', 'me'),
 ('is', 'nltk', 'use', 'me', 'to'),
 ('nltk', 'use', 'me', 'to', 'manipulate'),
 ('use', 'me', 'to', 'manipulate', 'text'),
 ('me', 'to', 'manipulate', 'text', '.')]
```

In [37]:
```
len(str_ngrams)
```

Out[37]:
```
7
```

In [ ]:
```

```

Stemming

Stemming is divided into three parts 1. porterstemmer (pst): it gives us root form of words 2. Lancasterstemmer (lst): it gives us core root form of word 3. snowballstemmer (snbt): it behaves like porterstemmer

In [ ]:
```

```

PorterStemmer

refered by pst

it gives us root form of words

In [39]:
```
#This module is used to give root form of words

from nltk.stem import PorterStemmer
pst = PorterStemmer()
```

In [40]:
```
pst.stem('playing')
```

Out[40]:
```
'play'
```

In [41]:
```
pst.stem('introduction')
```

Out[41]:
```
'introduct'
```

In [42]:
```
pst.stem('nationality')
```

Out[42]:
```
'nation'
```

In [43]:
```
# To print stem of more than one words we use loops

word_to_stem = ['give', 'giving', 'given', 'gave', 'pricing', 'maximum']

for i in word_to_stem:
```

```
    print(i+ ' : ' +pst.stem(i))
```

```
give : give
giving : give
given : given
gave : gave
pricing : price
maximum : maximum
```

In [44]:
```
words = ['pricing', 'stemming', 'cashing', 'maximum']

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

```
pricing : price
stemming : stem
cashing : cash
maximum : maximum
```

In [ ]:
```
```

Lancasterstemmer

refered by lst

it gives us core root form of words

In [45]:
```
from nltk.stem import LancasterStemmer
lst = LancasterStemmer()

for i in word_to_stem:    #used above loop of words
    print(i+ ' : ' +lst.stem(i))
```

```
give : giv
giving : giv
given : giv
gave : gav
pricing : pric
maximum : maxim
```

In [ ]:
```
```

SnowballStemmer

refered by sbst

it behaves like porterstemmert

In [46]:
```
from nltk.stem import SnowballStemmer
sbst = SnowballStemmer('english')

for i in word_to_stem:
    print(i+ ' : ' +sbst.stem(i))
```

```
give : give
giving : give
given : given
gave : gave
pricing : price
maximum : maximum
```

In [47]:
```
words = ['roaming']
for words in words:
    print(sbst.stem(words))
```

```
roam
```

In [48]:
```
sbst.stem('roaming')
```

Out[48]:
```
'roam'
```

In [ ]:
```
```

Lemmatization

In [49]:
```
from nltk.stem import wordnet
from nltk.stem import WordNetLemmatizer
```

```
word_lem = WordNetLemmatizer()
```

In [50]:
```
word_to_stem
```

Out[50]:
```
['give', 'giving', 'given', 'gave', 'pricing', 'maximum']
```

In [51]:
```
for words in word_to_stem:
    print(words+ ' : ' +word_lem.lemmatize(words))
```

```
give : give
giving : giving
given : given
gave : gave
pricing : pricing
maximum : maximum
```

In [52]:
```
word_lem.lemmatize('passionate')
```

Out[52]:
```
'passionate'
```

In [53]:
```
pst.stem('passionate')
```

Out[53]:
```
'passion'
```

In [54]:
```
lst.stem('passionate')
```

Out[54]:
```
'pass'
```

In [55]:
```
sbst.stem('Passionate')
```

Out[55]:
```
'passion'
```

In [56]:
```
word_to_stem
```

Out[56]:
```
['give', 'giving', 'given', 'gave', 'pricing', 'maximum']
```

In [ ]:
```

```

all in single loop

In [57]:
```
print()
print('PorterStemmer')
for i in word_to_stem:
    print(i+ ' : ' +pst.stem(i))

print()
print('LancasterStemming')
for i in word_to_stem:
    print(i+ (':') +lst.stem(i))

print()
print('SnowballStemmer')
for i in word_to_stem:
    print(i+ (':') +sbst.stem(i))

print()
print('Lammatizer')
for i in word_to_stem:
    print(i+ (':') +word_lem.lemmatize(i))
```

```
PorterStemmer
give : give
giving : give
given : given
gave : gave
pricing : price
maximum : maximum

LancasterStemming
give:giv
giving:giv
given:giv
gave:gav
pricing:pric
maximum:maxim

SnowballStemmer
give:give
giving:give
given:given
gave:gave
pricing:price
maximum:maximum

Lammatizer
give:give
giving:giving
given:given
gave:gave
pricing:pricing
maximum:maximum
```

In [ ]:

Stopwords

Stop words are commonly used words that are removed from natural language processing (NLP) tasks because they don't add much meaning to the text. They are often excluded from text processing tasks to search engine algorithms as well.

In [58]:
```
from nltk.corpus import stopwords
```

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

Out[59]:
```
['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 [60]:
```
len(stopwords.words('english'))
```

Out[60]:
```
179
```

In [61]:
```
stopwords.words('french')
```

Out[61]:
```
['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 [62]:
```
len(stopwords.words('french'))
```

Out[62]: 157

In [64]:
```
len(stopwords.words('chinese'))
```

Out[64]: 841

In [66]:
```
len(stopwords.words('spanish'))
```

Out[66]: 313

In [ ]:
```

```

Part of speech

In [67]:
```
sent = 'sam is a natural when it comes to drawing'
sent_token = word_tokenize(sent)
sent_token
```

Out[67]: ['sam', 'is', 'a', 'natural', 'when', 'it', 'comes', 'to', 'drawing']

In [68]:
```
for token in sent_token:
    print(nltk.pos_tag([token]))
```

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

In [ ]:

In [69]:
```
sent2 = 'my name is pratik and i am learning data science'
sent_t = word_tokenize(sent2)
sent_t
```

Out[69]:
```
['my', 'name', 'is', 'pratik', 'and', 'i', 'am', 'learning', 'data', 'science']
```

In [70]:
```
for i in sent_t:
    print(nltk.pos_tag([i]))
```

```
[('my', 'PRP$')]
[('name', 'NN')]
[('is', 'VBZ')]
[('pratik', 'NN')]
[('and', 'CC')]
[('i', 'NN')]
[('am', 'VBP')]
[('learning', 'VBG')]
[('data', 'NNS')]
[('science', 'NN')]
```

In [71]:
```
for i in str1:
    print(nltk.pos_tag([i]))
```

```
[('hello', 'NN')]
[('my', 'PRP$')]
[('name', 'NN')]
[('is', 'VBZ')]
[('nltk', 'NN')]
[('use', 'NN')]
[('me', 'PRP')]
[('to', 'TO')]
[('manipulate', 'NN')]
[('text', 'NN')]
[('.', '.')]
```

In [ ]:

NER : Name Entity Recognition

In [72]:
```
from nltk import ne_chunk
```

In [73]:
```
ne_sent = 'The US president stays in the WHITEHOUSE'
ne_sent
```

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

In [74]:
```
ne_token = word_tokenize(ne_sent)
ne_token
```

Out[74]:
```
['The', 'US', 'president', 'stays', 'in', 'the', 'WHITEHOUSE']
```

In [75]:
```
ne_tags = nltk.pos_tag(ne_token)
ne_tags
```

Out[75]:
```
[('The', 'DT'),
 ('US', 'NNP'),
 ('president', 'NN'),
 ('stays', 'NNS'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('WHITEHOUSE', 'NNP')]
```

In [76]:
```
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 [ ]:

In [ ]:

In [77]:
```
sent5 = 'india is a greate nation in the world and the PM of india is mr. Narendra modi'
sent5
```

Out[77]:
```
'india is a greate nation in the world and the PM of india is mr. Narendra modi'
```

In [78]:
```
sent5_token = word_tokenize(sent5)
sent5_token
```

Out[78]:
```
['india',
 'is',
 'a',
 'greate',
 'nation',
 'in',
 'the',
 'world',
 'and',
 'the',
 'PM',
 'of',
 'india',
 'is',
 'mr.',
 'Narendra',
 'modi']
```

In [79]:
```
sent5_tags = nltk.pos_tag(sent5_token)
sent5_tags
```

Out[79]:
```
[('india', 'NN'),
 ('is', 'VBZ'),
 ('a', 'DT'),
 ('greate', 'JJ'),
 ('nation', 'NN'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('world', 'NN'),
 ('and', 'CC'),
 ('the', 'DT'),
 ('PM', 'NNP'),
 ('of', 'IN'),
 ('india', 'NN'),
 ('is', 'VBZ'),
 ('mr.', 'JJ'),
 ('Narendra', 'NNP'),
 ('modi', 'NN')]
```

In [80]:
```
sent5_ner = ne_chunk(sent5_tags)
print(sent5_ner)
```

```
(S
  india/NN
  is/VBZ
  a/DT
  greate/JJ
  nation/NN
  in/IN
  the/DT
  world/NN
  and/CC
  the/DT
  (ORGANIZATION PM/NNP)
  of/IN
  india/NN
  is/VBZ
  (PERSON mr./JJ Narendra/NNP)
  modi/NN)
```

In [ ]:

In [ ]:

NLG (Natural Language Generation)

In [84]:
```
#import libraries

# !pip install WordCloud

from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

In [86]:
```
text = ("Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin Chart Pandas Datascience Wordcl
```

In [87]:
```
text_token = word_tokenize(text)
```

In [88]:
```
text_token
```

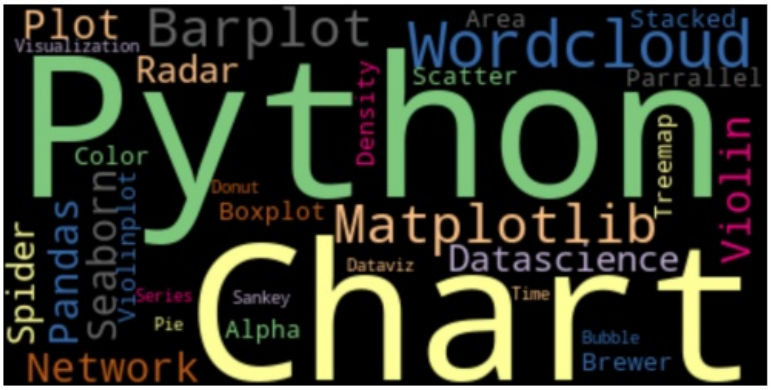
Out[88]:
```
['Python',
 'Python',
 'Python',
 'Matplotlib',
 'Matplotlib',
 'Seaborn',
 'Network',
 'Plot',
 'Violin',
 'Chart',
 'Pandas',
 'Datascience',
 'Wordcloud',
 'Spider',
 'Radar',
 'Parrallel',
 'Alpha',
 'Color',
 'Brewer',
 'Density',
 'Scatter',
 'Barplot',
 'Barplot',
 'Boxplot',
 'Violinplot',
 'Treemap',
 'Stacked',
 'Area',
 'Chart',
 'Chart',
 'Visualization',
 'Dataviz',
 'Donut',
 'Pie',
 'Time-Series',
 'Wordcloud',
 'Wordcloud',
 'Sankey',
 'Bubble']
```

In [89]:
```
text
```

Out[89]:
```
'Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin Chart Pandas Datascience Wordcloud Spid
er Radar Parrallel Alpha Color Brewer Density Scatter Barplot Barplot Boxplot Violinplot Treemap Stacked Area C
hart Chart Visualization Dataviz Donut Pie Time-Series Wordcloud Wordcloud Sankey Bubble'
```

In [133…
```
wordcloud = WordCloud(width=400, height=200, margin=2, background_color='black', colormap='Accent', mode = 'RGB
```

In [135…
```
plt.imshow(wordcloud, interpolation='quadric')
plt.axis('off')
plt.margins(x=0, y=0)
plt.show()
```



In [231…
```
text1= 'Pratik, Pratik,Pratik,Abhijeet,Abhijeet,Ashok,Ashok,dosti,dosti,Bhau,Abhijeet,maitri,Ashok,mitra,YZ,YZ,
text1
```

Out[231…
```
'Pratik, Pratik,Pratik,Abhijeet,Abhijeet,Ashok,Ashok,dosti,dosti,Bhau,Abhijeet,maitri,Ashok,mitra,YZ,YZ, Bhai,B
hau,kartik,Kartik,jigri,jigri Pratik, kartik, Prathamesh, Prathamesh, Friends,Frieds, Group, Group, Yaari,Yaari
'
```

In [ ]:

In [232…
```
cloud = WordCloud( width=350, height=200, margin=2,  background_color='black', min_font_size=5, colormap='Accen
```

In [233…
```
plt.imshow(cloud, interpolation='quadric')
plt.axis('off')
plt.show()
```

In [ ]:

In [ ]: