Natural Language Processing (NLP)

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

In [3]:
```
AI = '''Artificial Intelligence refers to the intelligence of machines. This is in contrast to the natural intelliger
humans and animals. With Artificial Intelligence, machines perform functions such as learning, planning, reasoning ar
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 b
AI could solve major challenges and crisis situations.'''
```

In [4]:
```
AI
```

Out[4]:
```
'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-solving. Most noteworthy, Artificial Intelligence is the simulation of human intelligence by machine
s.\nIt is probably the fastest-growing development in the World of technology and innovation. Furthermore, many expe
rts believe\nAI could solve major challenges and crisis situations.'
```

In [5]:
```
type(AI)
```

Out[5]:
```
str
```

In [6]:
```
from nltk.tokenize import word_tokenize
```

In [7]:
```
AI_tokens = word_tokenize(AI)
AI_tokens
```

Out[7]:
```
['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 [8]:
```
len(AI_tokens)
```

Out[8]: 81

In [9]:
```
AI
```

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, planning, reasoning
and\nproblem-solving. Most noteworthy, Artificial Intelligence is the simulation of human intelligence by machine
s.\nIt is probably the fastest-growing development in the World of technology and innovation. Furthermore, many expe
rts believe\nAI could solve major challenges and crisis situations.'
```

In [10]:
```
from nltk.tokenize import sent_tokenize
```

In [11]:
```
AI_sent = sent_tokenize(AI)
AI_sent
```

Out[11]:
```
['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-solvin
g.',
 '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 [12]:
```
len(AI_sent)
```

Out[12]: 6

In [13]:
```
AI
```

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, planning, reasoning
and\nproblem-solving. Most noteworthy, Artificial Intelligence is the simulation of human intelligence by machine
s.\nIt is probably the fastest-growing development in the World of technology and innovation. Furthermore, many expe
rts believe\nAI could solve major challenges and crisis situations.'
```

In [14]:
```
from nltk.tokenize import blankline_tokenize
AI_blank = blankline_tokenize(AI)
AI_blank
```

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-solving. Most noteworthy, Artificial Intelligence is the simulation of human intelligence by machine
s.\nIt is probably the fastest-growing development in the World of technology and innovation. Furthermore, many expe
rts believe\nAI could solve major challenges and crisis situations.']
```

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

Out[15]:
```
1
```

In [16]:
```
from nltk.tokenize import WhitespaceTokenizer
wt = WhitespaceTokenizer().tokenize(AI)
wt
```

Out[16]:
```
['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 [17]:
```
print(len(wt))
```

```
70
```

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

Out[18]:
```
81
```

In [19]:
```
s = 'Good apple cost $3.88 in hyderabad. Please buy two of them. Thanks.'
s
```

Out[19]:
```
'Good apple cost $3.88 in hyderabad. Please buy two of them. Thanks.'
```

In [20]:
```
from nltk.tokenize import wordpunct_tokenize
wordpunct_tokenize(s)
```

Out[20]:
```
['Good',
 'apple',
 'cost',
 '$',
 '3',
 '.',
 '88',
 'in',
 'hyderabad',
 '.',
 'Please',
 'buy',
 'two',
 'of',
 'them',
 '.',
 'Thanks',
 '.']
```

In [21]:
```
w_p = wordpunct_tokenize(AI)
w_p
```

Out[21]:
```
['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 [22]:
```
len(w_p)
```

Out[22]:
```
85
```

In [23]:
```
import nltk
```

In [24]:
```
from nltk.util import bigrams,trigrams,ngrams
```

In [25]:
```
string="The quick brown fox jumps over the lazy dog. Natural Language Processing with NLTK is quite interesting!"
quotes_tokens = nltk.word_tokenize(string)
quotes_tokens
```

Out[25]:
```
['The',
 'quick',
 'brown',
 'fox',
 'jumps',
 'over',
 'the',
 'lazy',
 'dog',
 '.',
 'Natural',
 'Language',
 'Processing',
 'with',
 'NLTK',
 'is',
 'quite',
 'interesting',
 '!']
```

In [26]:
```
len(quotes_tokens)
```

Out[26]:
```
19
```

In [27]:
```
quotes_bigrams = list(nltk.bigrams(quotes_tokens))
quotes_bigrams
```

Out[27]:
```
[('The', 'quick'),
 ('quick', 'brown'),
 ('brown', 'fox'),
 ('fox', 'jumps'),
 ('jumps', 'over'),
 ('over', 'the'),
 ('the', 'lazy'),
 ('lazy', 'dog'),
 ('dog', '.'),
 ('.', 'Natural'),
 ('Natural', 'Language'),
 ('Language', 'Processing'),
 ('Processing', 'with'),
 ('with', 'NLTK'),
 ('NLTK', 'is'),
 ('is', 'quite'),
 ('quite', 'interesting'),
 ('interesting', '!')]
```

In [28]:
```
quotes_trigrams = list(nltk.trigrams(quotes_tokens))
quotes_trigrams
```

Out[28]:
```
[('The', 'quick', 'brown'),
 ('quick', 'brown', 'fox'),
 ('brown', 'fox', 'jumps'),
 ('fox', 'jumps', 'over'),
 ('jumps', 'over', 'the'),
 ('over', 'the', 'lazy'),
 ('the', 'lazy', 'dog'),
 ('lazy', 'dog', '.'),
 ('dog', '.', 'Natural'),
 ('.', 'Natural', 'Language'),
 ('Natural', 'Language', 'Processing'),
 ('Language', 'Processing', 'with'),
 ('Processing', 'with', 'NLTK'),
 ('with', 'NLTK', 'is'),
 ('NLTK', 'is', 'quite'),
 ('is', 'quite', 'interesting'),
 ('quite', 'interesting', '!')]
```

In [29]:
```
quotes_ngrams = list(nltk.ngrams(quotes_tokens,4))
quotes_ngrams
```

Out[29]:
```
[('The', 'quick', 'brown', 'fox'),
 ('quick', 'brown', 'fox', 'jumps'),
 ('brown', 'fox', 'jumps', 'over'),
 ('fox', 'jumps', 'over', 'the'),
 ('jumps', 'over', 'the', 'lazy'),
 ('over', 'the', 'lazy', 'dog'),
 ('the', 'lazy', 'dog', '.'),
 ('lazy', 'dog', '.', 'Natural'),
 ('dog', '.', 'Natural', 'Language'),
 ('.', 'Natural', 'Language', 'Processing'),
 ('Natural', 'Language', 'Processing', 'with'),
 ('Language', 'Processing', 'with', 'NLTK'),
 ('Processing', 'with', 'NLTK', 'is'),
 ('with', 'NLTK', 'is', 'quite'),
 ('NLTK', 'is', 'quite', 'interesting'),
 ('is', 'quite', 'interesting', '!')]
```

In [30]:
```
quotes_ngrams = list(nltk.ngrams(quotes_tokens,9))
quotes_ngrams
```

Out[30]:
```
[('The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog'),
 ('quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog', '.'),
 ('brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog', '.', 'Natural'),
 ('fox', 'jumps', 'over', 'the', 'lazy', 'dog', '.', 'Natural', 'Language'),
 ('jumps',
  'over',
  'the',
  'lazy',
  'dog',
  '.',
  'Natural',
  'Language',
  'Processing'),
 ('over',
  'the',
  'lazy',
  'dog',
  '.',
  'Natural',
  'Language',
  'Processing',
  'with'),
 ('the',
  'lazy',
  'dog',
  '.',
  'Natural',
  'Language',
  'Processing',
  'with',
  'NLTK'),
 ('lazy',
  'dog',
  '.',
  'Natural',
  'Language',
  'Processing',
  'with',
  'NLTK',
  'is'),
 ('dog',
  '.',
```

```
 'Natural',
 'Language',
 'Processing',
 'with',
 'NLTK',
 'is',
 'quite'),
('.',
 'Natural',
 'Language',
 'Processing',
 'with',
 'NLTK',
 'is',
 'quite',
 'interesting'),
('Natural',
 'Language',
 'Processing',
 'with',
 'NLTK',
 'is',
 'quite',
 'interesting',
 '!')]
```

Stemming concept

Normalise words into its base form or root form. Stemming is 3 types:

In [32]:
```
# PorterStemmer gives core form
from nltk.stem import PorterStemmer
pst = PorterStemmer()
```

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

Out[33]:
```
'affect'
```

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

Out[34]: 'play'

In [35]:
```
pst.stem('maximum')
```

Out[35]: 'maximum'

In [36]:
```
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 [37]:
```
words_to_stem = ['give','giving','given','gaved','thinking','loving','maximum']

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

```
give : give
giving : give
given : given
gaved : gave
thinking : think
loving : love
maximum : maximum
```

In [38]:
```
#LancasterStemmer gives core root form
from nltk.stem import LancasterStemmer
lst = LancasterStemmer()

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

```
give : giv
giving : giv
given : giv
gaved : gav
thinking : think
loving : lov
maximum : maxim
```

In [39]:
```
#SnowBallStemmer gives root word behaves like porter stemmer only
from nltk.stem import SnowballStemmer
sbst = SnowballStemmer('english')

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

```
give : give
giving : give
given : given
gaved : gave
thinking : think
loving : love
maximum : maximum
```

Lemmatizations

In [41]:
```
from nltk.stem import wordnet
from nltk.stem import WordNetLemmatizer
word_lem = WordNetLemmatizer()
```

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

Out[42]:
```
['give', 'giving', 'given', 'gaved', 'thinking', 'loving', 'maximum']
```

In [43]:
```
for words in words_to_stem:
    print(words+ ' : ' + word_lem.lemmatize(words))
```

```
give : give
giving : giving
given : given
gaved : gaved
thinking : thinking
loving : loving
maximum : maximum
```

Stopwords

In [45]:
```
from nltk.corpus import stopwords
```

In [46]:
```
stopwords.words('english')
```

Out[46]:
```
['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 [47]:
```
len(stopwords.words('english'))
```

Out[47]:
```
179
```

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

Out[48]:
```
['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 [49]:
```
len(stopwords.words('french'))
```

Out[49]: 157

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

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

Out[51]:
```
232
```

Machine understanding grammer

In [53]:
```
# we will see how to work in POS uwing NLTK Library
sent = 'sam is a natural when it comes to drawing'
sent_tokens = word_tokenize(sent)
sent_tokens

# First we will tokenize using word_tokenize and then we will use pos_tag on all of the tokens
```

Out[53]:
```
['sam', 'is', 'a', 'natural', 'when', 'it', 'comes', 'to', 'drawing']
```

In [54]:
```
for token in sent_tokens:
    print(nltk.pos_tag([token]))
```

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

In [55]:
```
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')]
```

NER - Named Entity Recognition

In [57]:
```
from nltk import ne_chunk
```

In [58]:
```
NE_sent = 'The US president stays in the WHITEHOUSE'
```

In [59]:
```
NE_tokens = word_tokenize(NE_sent)
NE_tokens
```

Out[59]:
```
['The', 'US', 'president', 'stays', 'in', 'the', 'WHITEHOUSE']
```

In [60]:
```
NE_tags = nltk.pos_tag(NE_tokens)
NE_tags
```

Out[60]:
```
[('The', 'DT'),
 ('US', 'NNP'),
 ('president', 'NN'),
 ('stays', 'NNS'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('WHITEHOUSE', 'NNP')]
```

In [61]:
```
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 [62]:
```
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[62]:
```
[('the', 'DT'),
 ('big', 'JJ'),
 ('cat', 'NN'),
 ('ate', 'VBD'),
 ('the', 'DT'),
 ('little', 'JJ'),
 ('mouse', 'NN'),
 ('who', 'WP'),
 ('was', 'VBD'),
 ('after', 'IN'),
 ('fresh', 'JJ'),
 ('cheese', 'NN')]
```

NLU completed

NLG - Natural Language Generations

WordCloud

In [65]:
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

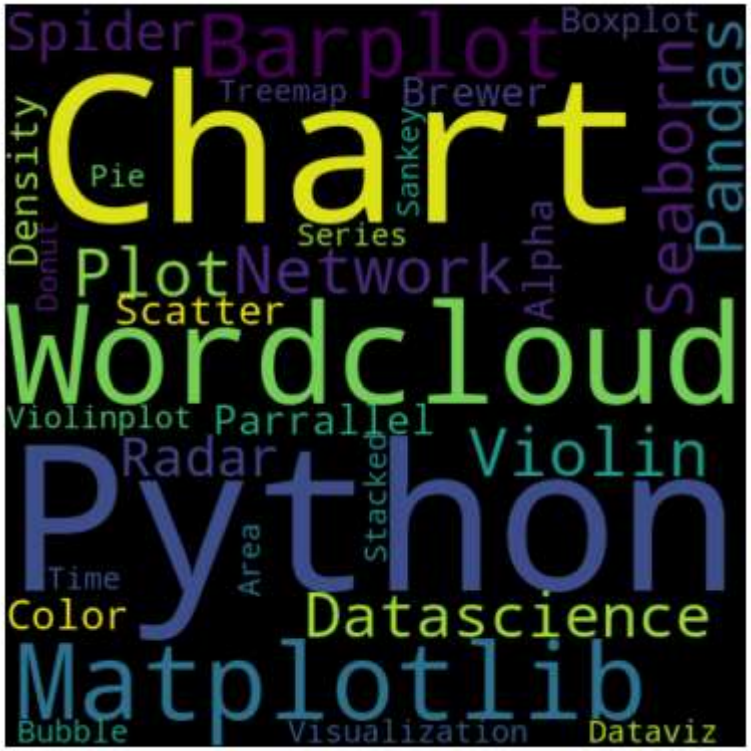
In [128…
```
# Create a list of word
text=("Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin Chart Pandas Datascience Wordcloud Spic
```

In [130…
```
text
```

Out[130…
```
'Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin Chart Pandas Datascience Wordcloud Spider Ra
dar 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 [132…
```
# Create the wordcloud object
wordcloud = WordCloud(width=480, height=480, margin=0).generate(text)
```

In [134…
```
# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()
```


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

In [ ]: