NLTK DOCUMENTATION

Introducation

- NLTK, or Natural Language Toolkit, is a Python library that helps computers understand and work with human language, like English. It's really useful for tasks in Natural Language Processing (NLP), which is all about teaching computers to understand and analyze text.

Here's what NLTK can do:

- Breaking down text: It can split up a sentence into individual words or even break large text into sentences.
- Finding the base of words: It helps to simplify words by reducing them to their base form. For example, it can turn "running" into "run."
- Understanding parts of speech: It can label words in a sentence as nouns, verbs, adjectives, etc.
- Understanding grammar: It can analyze how words are structured in a sentence to find out the grammar.
- Identifying important names: It can recognize names of people, places, and organizations in a piece of text.
- Using language data: NLTK also has access to a lot of pre-loaded text (like famous books or reviews) and word databases to help with different language tasks.

01. nltk.tokenize

- The tokenize module use to break down the santence to words and sentences
- word_tokenize ----> break down the paragrph or sentences to words.
- sent_tokenize ----> break down the paragraph basis of sentences.
- blankline_tokenize --> Its heps to count how many paragrpahs in the docoments.
- Whitespace_tokenizer ---> counts the words except the ',' and '.' from the documetation.
- wordpunct_tokenize ---> use for counts the numbers and words

In [1]:
```
paragraph='''
Life is a journey filled with moments of joy, challenge, growth, and discovery.
shaped by the choices we make and the experiences we encounter. Sometimes, life
full of obstacles that test our strength and patience. Other times, it’s like a
Along the way, we learn lessons—about ourselves, about others, and about the wor
it offers countless opportunities to grow, adapt, and evolve. Through every succ
embrace change, and find meaning in both the small and big moments. Ultimately,
and making the most of the time we have.
'''
```

```
In [2]: print(paragraph)
```

```
Life is a journey filled with moments of joy, challenge, growth, and discovery. I
t’s a unique experience for each person,
shaped by the choices we make and the experiences we encounter. Sometimes, life f
eels like an uphill climb,
full of obstacles that test our strength and patience. Other times, it’s like a s
mooth road, bringing happiness, love, and fulfillment.
Along the way, we learn lessons—about ourselves, about others, and about the worl
d around us. The beauty of life lies in its unpredictability;
it offers countless opportunities to grow, adapt, and evolve. Through every succe
ss and setback, life teaches us to keep moving forward,
embrace change, and find meaning in both the small and big moments. Ultimately, l
ife is about finding balance, nurturing relationships,
and making the most of the time we have.
```

```
In [3]: # import nltk
        import os
        import nltk
```

word_tokenize

- Break down the paragraph to list of words

```
In [4]: # import word_tokenizer
        from nltk.tokenize import word_tokenize
        words=word_tokenize(paragraph)
        print(words)
```

```
['Life', 'is', 'a', 'journey', 'filled', 'with', 'moments', 'of', 'joy', ',', 'ch
allenge', ',', 'growth', ',', 'and', 'discovery', '.', 'It', '’', 's', 'a', 'uniq
ue', 'experience', 'for', 'each', 'person', ',', 'shaped', 'by', 'the', 'choice
s', 'we', 'make', 'and', 'the', 'experiences', 'we', 'encounter', '.', 'Sometime
s', ',', 'life', 'feels', 'like', 'an', 'uphill', 'climb', ',', 'full', 'of', 'ob
stacles', 'that', 'test', 'our', 'strength', 'and', 'patience', '.', 'Other', 'ti
mes', ',', 'it', '’', 's', 'like', 'a', 'smooth', 'road', ',', 'bringing', 'happi
ness', ',', 'love', ',', 'and', 'fulfillment', '.', 'Along', 'the', 'way', ',',
'we', 'learn', 'lessons—about', 'ourselves', ',', 'about', 'others', ',', 'and',
'about', 'the', 'world', 'around', 'us', '.', 'The', 'beauty', 'of', 'life', 'lie
s', 'in', 'its', 'unpredictability', ';', 'it', 'offers', 'countless', 'opportuni
ties', 'to', 'grow', ',', 'adapt', ',', 'and', 'evolve', '.', 'Through', 'every',
'success', 'and', 'setback', ',', 'life', 'teaches', 'us', 'to', 'keep', 'movin
g', 'forward', ',', 'embrace', 'change', ',', 'and', 'find', 'meaning', 'in', 'bo
th', 'the', 'small', 'and', 'big', 'moments', '.', 'Ultimately', ',', 'life', 'i
s', 'about', 'finding', 'balance', ',', 'nurturing', 'relationships', ',', 'and',
'making', 'the', 'most', 'of', 'the', 'time', 'we', 'have', '.']
```

In the above code the word_tokenize give all the words are in a paragraph, it consider "," and "." as a single words

```
In [5]: # length of the words
        len(words)

Out[5]: 166
```

sent_tokenize

- Break down the paragraph to noumber of sentences

In [6]:
```
# import sent_tokenize
from nltk.tokenize import sent_tokenize
sentences=sent_tokenize(paragraph)
print(sentences)
```

```
['\nLife is a journey filled with moments of joy, challenge, growth, and discover
y.', 'It’s a unique experience for each person,\nshaped by the choices we make an
d the experiences we encounter.', 'Sometimes, life feels like an uphill climb,\nf
ull of obstacles that test our strength and patience.', 'Other times, it’s like a
smooth road, bringing happiness, love, and fulfillment.', 'Along the way, we lear
n lessons—about ourselves, about others, and about the world around us.', 'The be
auty of life lies in its unpredictability;\nit offers countless opportunities to
grow, adapt, and evolve.', 'Through every success and setback, life teaches us to
keep moving forward,\nembrace change, and find meaning in both the small and big
moments.', 'Ultimately, life is about finding balance, nurturing relationships,\n
and making the most of the time we have.']
```

In [7]:
```
print(len(sentences)) # lenth
```

```
8
```

In [8]:
```
print(sentences[1]) # individual sentences
```

```
It’s a unique experience for each person,
shaped by the choices we make and the experiences we encounter.
```

blankline_tokenize

- Give the number of sentences in a documents

In [9]:
```
# import blankline_tokenize
from nltk.tokenize import blankline_tokenize
n_sentence=blankline_tokenize(paragraph)
n_sentence
```

Out[9]:
```
['\nLife is a journey filled with moments of joy, challenge, growth, and discov
ery. It’s a unique experience for each person,\nshaped by the choices we make a
nd the experiences we encounter. Sometimes, life feels like an uphill climb,\nf
ull of obstacles that test our strength and patience. Other times, it’s like a
smooth road, bringing happiness, love, and fulfillment.\nAlong the way, we lear
n lessons—about ourselves, about others, and about the world around us. The bea
uty of life lies in its unpredictability;\nit offers countless opportunities to
grow, adapt, and evolve. Through every success and setback, life teaches us to
keep moving forward,\nembrace change, and find meaning in both the small and bi
g moments. Ultimately, life is about finding balance, nurturing relationship
s,\nand making the most of the time we have.\n']
```

In [10]:
```
len(n_sentence) # the doc we feed have only one paragraph
```

Out[10]:
```
1
```

Whitespace_tokenizer

- It give the words list of a sentence except the ',' and '.'

```
In [11]: # import Whitespace_tokenizer
         from nltk.tokenize import WhitespaceTokenizer
         word=WhitespaceTokenizer().tokenize(paragraph)
         print(word)
```

```
['Life', 'is', 'a', 'journey', 'filled', 'with', 'moments', 'of', 'joy,', 'challe
nge,', 'growth,', 'and', 'discovery.', 'It’s', 'a', 'unique', 'experience', 'fo
r', 'each', 'person,', 'shaped', 'by', 'the', 'choices', 'we', 'make', 'and', 'th
e', 'experiences', 'we', 'encounter.', 'Sometimes,', 'life', 'feels', 'like', 'a
n', 'uphill', 'climb,', 'full', 'of', 'obstacles', 'that', 'test', 'our', 'streng
th', 'and', 'patience.', 'Other', 'times,', 'it’s', 'like', 'a', 'smooth', 'roa
d,', 'bringing', 'happiness,', 'love,', 'and', 'fulfillment.', 'Along', 'the', 'w
ay,', 'we', 'learn', 'lessons—about', 'ourselves,', 'about', 'others,', 'and', 'a
bout', 'the', 'world', 'around', 'us.', 'The', 'beauty', 'of', 'life', 'lies', 'i
n', 'its', 'unpredictability;', 'it', 'offers', 'countless', 'opportunities', 't
o', 'grow,', 'adapt,', 'and', 'evolve.', 'Through', 'every', 'success', 'and', 's
etback,', 'life', 'teaches', 'us', 'to', 'keep', 'moving', 'forward,', 'embrace',
'change,', 'and', 'find', 'meaning', 'in', 'both', 'the', 'small', 'and', 'big',
'moments.', 'Ultimately,', 'life', 'is', 'about', 'finding', 'balance,', 'nurturi
ng', 'relationships,', 'and', 'making', 'the', 'most', 'of', 'the', 'time', 'we',
'have.']
```

```
In [12]: len(word)

Out[12]: 132
```

```
In [13]: ### In the word_tokenizer we got 166 token where it include ',' and '.' but here
```

wordpunct_tokenize

- If we need token fron numbers like 56.45 --> ['56' , '.' , '45'] as the tokens

```
In [14]: sen= "John bought 12 apples for $5.99 each on 29th October 2024."
```

```
In [15]: # import wordpunct_tokenze
         from nltk.tokenize import wordpunct_tokenize
         wp_sen=wordpunct_tokenize(sen)
         print(wp_sen)
```

```
['John', 'bought', '12', 'apples', 'for', '$', '5', '.', '99', 'each', 'on', '29t
h', 'October', '2024', '.']
```

```
In [16]: len(wp_sen)

Out[16]: 15
```

02. nltk.stem

- stem help to find root word like "holding" => "hold"
- Provides methods for stemming and lemmatization (reducing words to their base forms).

- PorterStemmer() ---> help to find the root word
- LancasterStemmer() --> also use to find the root words of a word
- WordNetLemmatizer()

PorterStemmer

- find the root word of a words like walking --> walk

In [17]:
```
# import PorterStemmer
from nltk.stem import PorterStemmer
st=PorterStemmer()
st.stem("calling") # finding the root words
```

Out[17]:
```
'call'
```

In [18]:
```
st.stem("going")
```

Out[18]:
```
'go'
```

In [19]:
```
st.stem("running")
```

Out[19]:
```
'run'
```

In [20]:
```
# if we have list of word insted we pass one by one leets use for loop
l=['looking','getting','cleaned','works','offers','loving','hates','played']
#use loop
for i in l:
    print(i,':',st.stem(i))
```

```
looking : look
getting : get
cleaned : clean
works : work
offers : offer
loving : love
hates : hate
played : play
```

LancasterStemmer

- also use to find the root words of a word

In [21]:
```
words = ['sincerely','electricity','roughly','ringing','playing','player']

# import LancasterStemmer
from nltk.stem import LancasterStemmer
ls=LancasterStemmer()
for w in words:

    print(w, " : ", ls.stem(w))
```

```
sincerely  :  sint
electricity  :  elect
roughly  :  rough
ringing  :  ring
playing  :  play
player  :  play
```

WordNetLemmatizer

- lemmatize the word to its base forms

In [22]:
```
words = ['holds','played','roughs','rings','groups','mobiles','runs','guns']

# import LancasterStemmer
from nltk.stem import WordNetLemmatizer
wl=WordNetLemmatizer()
for w in words:

    print(w, " : ", wl.lemmatize(w,pos='v')) # pos = part of speach, n= nouns,v=
```

```
holds  :  hold
played  :  play
roughs  :  rough
rings  :  ring
groups  :  group
mobiles  :  mobiles
runs  :  run
guns  :  gun
```

snowballstemmer

- 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 stemmer

In [23]:
```
#we have another stemmer called as snowball stemmer lets see about this snowball

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

```
holds:hold
played:play
roughs:rough
rings:ring
groups:group
mobiles:mobil
runs:run
guns:gun
```

lemmatization

In [24]:
```
# import library
from nltk.stem import wordnet
from nltk.stem import WordNetLemmatizer
word_lem=WordNetLemmatizer()
```

In [25]:
```
# word to apply lemmatization
print(words)
```

```
['holds', 'played', 'roughs', 'rings', 'groups', 'mobiles', 'runs', 'guns']
```

In [26]:
```
for i in words:
    print(i ,":",word_lem.lemmatize(i))
```

```
holds : hold
played : played
roughs : rough
rings : ring
groups : group
mobiles : mobile
runs : run
guns : gun
```

3. nltk.util

- Groups the words based on requrments
- bigrams()- group the words based on 2 words
- trigrams() - group the words base on 3 words
- ngrams() - group the words based on the number you want

bigrams()

In [27]:
```
paragraph
```

Out[27]:
```
'\nLife is a journey filled with moments of joy, challenge, growth, and discove
ry. It’s a unique experience for each person,\nshaped by the choices we make an
d the experiences we encounter. Sometimes, life feels like an uphill climb,\nfu
ll of obstacles that test our strength and patience. Other times, it’s like a s
mooth road, bringing happiness, love, and fulfillment.\nAlong the way, we learn
lessons—about ourselves, about others, and about the world around us. The beaut
y of life lies in its unpredictability;\nit offers countless opportunities to g
row, adapt, and evolve. Through every success and setback, life teaches us to k
eep moving forward,\nembrace change, and find meaning in both the small and big
moments. Ultimately, life is about finding balance, nurturing relationships,\na
nd making the most of the time we have.\n'
```

In [28]:
```
# import bigrams
from nltk.util import bigrams
word_token=word_tokenize(paragraph)
print(word_token)
```

```
['Life', 'is', 'a', 'journey', 'filled', 'with', 'moments', 'of', 'joy', ',', 'ch
allenge', ',', 'growth', ',', 'and', 'discovery', '.', 'It', ''', 's', 'a', 'uniq
ue', 'experience', 'for', 'each', 'person', ',', 'shaped', 'by', 'the', 'choice
s', 'we', 'make', 'and', 'the', 'experiences', 'we', 'encounter', '.', 'Sometime
s', ',', 'life', 'feels', 'like', 'an', 'uphill', 'climb', ',', 'full', 'of', 'ob
stacles', 'that', 'test', 'our', 'strength', 'and', 'patience', '.', 'Other', 'ti
mes', ',', 'it', ''', 's', 'like', 'a', 'smooth', 'road', ',', 'bringing', 'happi
ness', ',', 'love', ',', 'and', 'fulfillment', '.', 'Along', 'the', 'way', ',',
'we', 'learn', 'lessons—about', 'ourselves', ',', 'about', 'others', ',', 'and',
'about', 'the', 'world', 'around', 'us', '.', 'The', 'beauty', 'of', 'life', 'lie
s', 'in', 'its', 'unpredictability', ';', 'it', 'offers', 'countless', 'opportuni
ties', 'to', 'grow', ',', 'adapt', ',', 'and', 'evolve', '.', 'Through', 'every',
'success', 'and', 'setback', ',', 'life', 'teaches', 'us', 'to', 'keep', 'movin
g', 'forward', ',', 'embrace', 'change', ',', 'and', 'find', 'meaning', 'in', 'bo
th', 'the', 'small', 'and', 'big', 'moments', '.', 'Ultimately', ',', 'life', 'i
s', 'about', 'finding', 'balance', ',', 'nurturing', 'relationships', ',', 'and',
'making', 'the', 'most', 'of', 'the', 'time', 'we', 'have', '.']
```

In [29]:
```
# lets make groups based on 2 words
word2=list(nltk.bigrams(word_token))
word2
```

```
Out[29]: [('Life', 'is'),
          ('is', 'a'),
          ('a', 'journey'),
          ('journey', 'filled'),
          ('filled', 'with'),
          ('with', 'moments'),
          ('moments', 'of'),
          ('of', 'joy'),
          ('joy', ','),
          (',', 'challenge'),
          ('challenge', ','),
          (',', 'growth'),
          ('growth', ','),
          (',', 'and'),
          ('and', 'discovery'),
          ('discovery', '.'),
          ('.', 'It'),
          ('It', '’'),
          ('’', 's'),
          ('s', 'a'),
          ('a', 'unique'),
          ('unique', 'experience'),
          ('experience', 'for'),
          ('for', 'each'),
          ('each', 'person'),
          ('person', ','),
          (',', 'shaped'),
          ('shaped', 'by'),
          ('by', 'the'),
          ('the', 'choices'),
          ('choices', 'we'),
          ('we', 'make'),
          ('make', 'and'),
          ('and', 'the'),
          ('the', 'experiences'),
          ('experiences', 'we'),
          ('we', 'encounter'),
          ('encounter', '.'),
          ('.', 'Sometimes'),
          ('Sometimes', ','),
          (',', 'life'),
          ('life', 'feels'),
          ('feels', 'like'),
          ('like', 'an'),
          ('an', 'uphill'),
          ('uphill', 'climb'),
          ('climb', ','),
          (',', 'full'),
          ('full', 'of'),
          ('of', 'obstacles'),
          ('obstacles', 'that'),
          ('that', 'test'),
          ('test', 'our'),
          ('our', 'strength'),
          ('strength', 'and'),
          ('and', 'patience'),
          ('patience', '.'),
          ('.', 'Other'),
          ('Other', 'times'),
          ('times', ','),
(',', 'it'),
('it', '’'),
('’', 's'),
('s', 'like'),
('like', 'a'),
('a', 'smooth'),
('smooth', 'road'),
('road', ','),
(',', 'bringing'),
('bringing', 'happiness'),
('happiness', ','),
(',', 'love'),
('love', ','),
(',', 'and'),
('and', 'fulfillment'),
('fulfillment', '.'),
('.', 'Along'),
('Along', 'the'),
('the', 'way'),
('way', ','),
(',', 'we'),
('we', 'learn'),
('learn', 'lessons—about'),
('lessons—about', 'ourselves'),
('ourselves', ','),
(',', 'about'),
('about', 'others'),
('others', ','),
(',', 'and'),
('and', 'about'),
('about', 'the'),
('the', 'world'),
('world', 'around'),
('around', 'us'),
('us', '.'),
('.', 'The'),
('The', 'beauty'),
('beauty', 'of'),
('of', 'life'),
('life', 'lies'),
('lies', 'in'),
('in', 'its'),
('its', 'unpredictability'),
('unpredictability', ';'),
(';', 'it'),
('it', 'offers'),
('offers', 'countless'),
('countless', 'opportunities'),
('opportunities', 'to'),
('to', 'grow'),
('grow', ','),
(',', 'adapt'),
('adapt', ','),
(',', 'and'),
('and', 'evolve'),
('evolve', '.'),
('.', 'Through'),
('Through', 'every'),
('every', 'success'),
('success', 'and'),
('and', 'setback'),
('setback', ','),
(',', 'life'),
('life', 'teaches'),
('teaches', 'us'),
('us', 'to'),
('to', 'keep'),
('keep', 'moving'),
('moving', 'forward'),
('forward', ','),
(',', 'embrace'),
('embrace', 'change'),
('change', ','),
(',', 'and'),
('and', 'find'),
('find', 'meaning'),
('meaning', 'in'),
('in', 'both'),
('both', 'the'),
('the', 'small'),
('small', 'and'),
('and', 'big'),
('big', 'moments'),
('moments', '.'),
('.', 'Ultimately'),
('Ultimately', ','),
(',', 'life'),
('life', 'is'),
('is', 'about'),
('about', 'finding'),
('finding', 'balance'),
('balance', ','),
(',', 'nurturing'),
('nurturing', 'relationships'),
('relationships', ','),
(',', 'and'),
('and', 'making'),
('making', 'the'),
('the', 'most'),
('most', 'of'),
('of', 'the'),
('the', 'time'),
('time', 'we'),
('we', 'have'),
('have', '.')]
```

In [30]:
```
words
```

Out[30]:
```
['holds', 'played', 'roughs', 'rings', 'groups', 'mobiles', 'runs', 'guns']
```

In [31]:
```
word3=list(nltk.bigrams(words))
word3
```

```
Out[31]: [('holds', 'played'),
          ('played', 'roughs'),
          ('roughs', 'rings'),
          ('rings', 'groups'),
          ('groups', 'mobiles'),
          ('mobiles', 'runs'),
          ('runs', 'guns')]
```

trigrams()

```
In [32]: print(words)

['holds', 'played', 'roughs', 'rings', 'groups', 'mobiles', 'runs', 'guns']
```

```
In [33]: # trigram
         tri=list(nltk.trigrams(words))
```

```
In [34]: tri
```

```
Out[34]: [('holds', 'played', 'roughs'),
          ('played', 'roughs', 'rings'),
          ('roughs', 'rings', 'groups'),
          ('rings', 'groups', 'mobiles'),
          ('groups', 'mobiles', 'runs'),
          ('mobiles', 'runs', 'guns')]
```

ngrams()

```
In [35]: words
```

```
Out[35]: ['holds', 'played', 'roughs', 'rings', 'groups', 'mobiles', 'runs', 'guns']
```

```
In [36]: ngram=list(nltk.ngrams(words))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[36], line 1
----> 1 ngram=list(nltk.ngrams(words))

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

```
In [37]: # here we got error because of it need one argument on integer by which we group
```

```
In [38]: ngram=list(nltk.ngrams(words,4)) # group with 4 words
         ngram
```

```
Out[38]: [('holds', 'played', 'roughs', 'rings'),
          ('played', 'roughs', 'rings', 'groups'),
          ('roughs', 'rings', 'groups', 'mobiles'),
          ('rings', 'groups', 'mobiles', 'runs'),
          ('groups', 'mobiles', 'runs', 'guns')]
```

```
In [39]: ngram=list(nltk.ngrams(words,5)) # group with 4 words
         ngram
```

```
Out[39]: [('holds', 'played', 'roughs', 'rings', 'groups'),
          ('played', 'roughs', 'rings', 'groups', 'mobiles'),
          ('roughs', 'rings', 'groups', 'mobiles', 'runs'),
          ('rings', 'groups', 'mobiles', 'runs', 'guns')]
```

Regexp Tokenizer

- clasify the special symbol and the sentences separated by thesesymbols from the paragraphs

```
In [40]: s = ("Alas, it has not rained today. When, do you think, "  "will it rain again?
```

```
In [41]: s
```

```
Out[41]: 'Alas, it has not rained today. When, do you think, will it rain again?'
```

```
In [42]: # extract all special symbol from the string
         from nltk.tokenize import regexp_tokenize
         regexp_tokenize(s, r'[,\.\?!"]\s*', gaps=False)
```

```
Out[42]: [', ', '. ', ', ', ', ', '?']
```

```
In [43]: # extracts only sentrences separatedd by the symbols from the strings
         regexp_tokenize(s, r'[,\.\?!"]\s*', gaps=True)
```

```
Out[43]: ['Alas',
          'it has not rained today',
          'When',
          'do you think',
          'will it rain again']
```

```
In [44]: # based on our requirments extracts from the strings
         s2 = ("<p>Although this is <b>not</b> the case here, we must " "not relax our vi
         print(s2)

        <p>Although this is <b>not</b> the case here, we must not relax our vigilance!</p
        >
```

```
In [45]: regexp_tokenize(s2, r'</?(?P<named>b|p)>', gaps=False) # we try to extract insia
```

```
Out[45]: ['p', 'b', 'b', 'p']
```

```
In [46]: regexp_tokenize(s2, r'</?(?P<named>b|p)>', gaps=True) # we try to extract
```

```
Out[46]: ['p',
          'Although this is ',
          'b',
          'not',
          'b',
          ' the case here, we must not relax our vigilance!',
          'p']
```

TweetTokenizer

- TweetTokenizer is a tokenizer specifically designed for micro-blogging tokenization tasks.

```
In [47]: # import the tweeetTokenizer
         from nltk.tokenize import TweetTokenizer
         twtk=TweetTokenizer()
```

```
In [ ]:
```

```
In [48]: s3="This is a cooool #dummysmiley: :-) :-P <3 and some arrows < > -> <--"
         twtk.tokenize(s3)
```

```
Out[48]: ['This',
          'is',
          'a',
          'cooool',
          '#dummysmiley',
          ':',
          ':-)',
          ':-P',
          '<3',
          'and',
          'some',
          'arrows',
          '<',
          '>',
          '->',
          '<--']
```

```
In [49]: s
```

```
Out[49]: 'Alas, it has not rained today. When, do you think, will it rain again?'
```

```
In [50]: twtk=TweetTokenizer(reduce_len=True)
         twtk.tokenize(s)
```

```
Out[50]: ['Alas',
          ',',
          'it',
          'has',
          'not',
          'rained',
          'today',
          '.',
          'When',
          ',',
          'do',
          'you',
          'think',
          ',',
          'will',
          'it',
          'rain',
          'again',
          '?']
```

```
In [51]: # TO remove the handel from the text like username
         tknzr = TweetTokenizer(strip_handles=True, reduce_len=True) # adding strip_lengt
```

```
s6 = '@remy: This is waaaaayyyy too much for you!!!!!!'
tknzr.tokenize(s6)
```

Out[51]:
```
[':', 'This', 'is', 'waaayyy', 'too', 'much', 'for', 'you', '!', '!', '!']
```

In [52]:
```
# IF your string have capital words we move to small letters to ading preserve_c
tknzr = TweetTokenizer(preserve_case=False)
s9 = "@jrmy: I'm REALLY HAPPYYY about that! NICEEEE :D :P"
tknzr.tokenize(s9)
```

Out[52]:
```
['@jrmy',
 ':',
 "i'm",
 'really',
 'happyyy',
 'about',
 'that',
 '!',
 'niceeee',
 ':D',
 ':P']
```

In [53]:
```
# IF you string have same panctuans characters
tknzr = TweetTokenizer()
s10 = "Photo: Aujourd'hui sur http://t.co/0gebOFDUzn Projet... http://t.co/bKfIU
tknzr.tokenize(s10)
```

Out[53]:
```
['Photo',
 ':',
 "Aujourd'hui",
 'sur',
 'http://t.co/0gebOFDUzn',
 'Projet',
 '...',
 'http://t.co/bKfIUbydz2',
 '...',
 'http://fb.me/3b6uXpz0L']
```

In [54]:
```
# Tokenize multipllle sentences at once
tknzr = TweetTokenizer()
sentences = [
    "This is a cooool #dummysmiley: :-) :-P <3 and some arrows < > -> <--",
    "@jrmy: I'm REALLY HAPPYYY about that! NICEEEE :D :P",
     "@_willy65: No place for @chuck tonight. Sorry."
]
tknzr.tokenize_sents(sentences)
```

```
Out[54]: [['This',
           'is',
           'a',
           'cooool',
           '#dummysmiley',
           ':',
           ':-)',
           ':-P',
           '<3',
           'and',
           'some',
           'arrows',
           '<',
           '>',
           '->',
           '<--'],
          ['@jrmy',
           ':',
           "I'm",
           'REALLY',
           'HAPPYYY',
           'about',
           'that',
           '!',
           'NICEEEE',
           ':D',
           ':P'],
          ['@_willy65',
           ':',
           'No',
           'place',
           'for',
           '@chuck',
           'tonight',
           '.',
           'Sorry',
           '.']]
```

PunktSentenceTokenizer

- based on white space the sentence split

```
In [55]: from nltk.tokenize import PunktSentenceTokenizer
```

```
In [56]: pst = PunktSentenceTokenizer()
         pst.tokenize('See Section 3).  Or Section 2).  ')
```

```
Out[56]: ['See Section 3).', 'Or Section 2).']
```

```
In [57]: pst.tokenize('See Section 3.)  Or Section 2.)  ', realign_boundaries=False)
```

```
Out[57]: ['See Section 3.', ')  Or Section 2.', ')']
```

4. POS (part of speach) & STOP WORDS

- there is other concept called POS (part of speech) which deals with subject, noun, pronoun but before of this lets go with other concept called STOPWORDS
- STOPWORDS = i, is, as,at, on, about & nltk has their own list of stopewords

In [58]:
```
# to know thw stop word in international language
from nltk.corpus import stopwords # import
```

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

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "y
ou've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'hi
m', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'it
s', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which',
'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'w
as', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'unt
il', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'in
to', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'u
p', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'the
n', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'onl
y', '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', "d
oesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",
'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "sha
n't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "wo
n't", 'wouldn', "wouldn't"]
```

In [60]:
```
len(st)# number of stopwords in english
```

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

In [61]:
```
german=stopwords.words('german')
print(german)
print(f"Total number of Stop Words in German:- ",len(german))
```

```
['aber', 'alle', 'allem', 'allen', 'aller', 'alles', 'als', 'also', 'am', 'an',
'ander', 'andere', 'anderem', 'anderen', 'anderer', 'anderes', 'anderm', 'ander
n', 'anderr', 'anders', 'auch', 'auf', 'aus', 'bei', 'bin', 'bis', 'bist', 'da',
'damit', 'dann', 'der', 'den', 'des', 'dem', 'die', 'das', 'dass', 'daß', 'dersel
be', 'derselben', 'denselben', 'desselben', 'demselben', 'dieselbe', 'dieselben',
'dasselbe', 'dazu', 'dein', 'deine', 'deinem', 'deinen', 'deiner', 'deines', 'den
n', 'derer', 'dessen', 'dich', 'dir', 'du', 'dies', 'diese', 'diesem', 'diesen',
'dieser', 'dieses', 'doch', 'dort', 'durch', 'ein', 'eine', 'einem', 'einen', 'ei
ner', 'eines', 'einig', 'einige', 'einigem', 'einigen', 'einiger', 'einiges', 'ei
nmal', 'er', 'ihn', 'ihm', 'es', 'etwas', 'euer', 'eure', 'eurem', 'euren', 'eure
r', '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', 'je
dem', 'jeden', 'jeder', 'jedes', 'jene', 'jenem', 'jenen', 'jener', 'jenes', 'jet
zt', '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', 's
ein', 'seine', 'seinem', 'seinen', 'seiner', 'seines', 'selbst', 'sich', 'sie',
'ihnen', 'sind', 'so', 'solche', 'solchem', 'solchen', 'solcher', 'solches', 'sol
l', 'sollte', 'sondern', 'sonst', 'über', 'um', 'und', 'uns', 'unsere', 'unsere
m', 'unseren', 'unser', 'unseres', 'unter', 'viel', 'vom', 'von', 'vor', 'währen
d', 'war', 'waren', 'warst', 'was', 'weg', 'weil', 'weiter', 'welche', 'welchem',
'welchen', 'welcher', 'welches', 'wenn', 'werde', 'werden', 'wie', 'wieder', 'wil
l', 'wir', 'wird', 'wirst', 'wo', 'wollen', 'wollte', 'würde', 'würden', 'zu', 'z
um', 'zur', 'zwar', 'zwischen']
Total number of Stop Words in German:-  232
```

In [62]:
```
# similarly we can find the stop word in international languages
```

lets works on pos using nltk library

In [63]:
```
sentence='''I am Darshanikanta, i have completed my graduations from Bhadrak aut
Now i am in Hyderabad taking fullstack data science at  NareshIT technology, und
```

In [64]:
```
# perform word tokenize
print(sentence)
print("\n\n")
from nltk.tokenize import word_tokenize
wt=word_tokenize(sentence)
print(wt)
```

```
I am Darshanikanta, i have completed my graduations from Bhadrak autonomous colle
ge, with 83% aggrigate with 9.01 CGPA.
Now i am in Hyderabad taking fullstack data science at  NareshIT technology, unde
r guidance of Mr. Prakash senapati



['I', 'am', 'Darshanikanta', ',', 'i', 'have', 'completed', 'my', 'graduations',
'from', 'Bhadrak', 'autonomous', 'college', ',', 'with', '83', '%', 'aggrigate',
'with', '9.01', 'CGPA', '.', 'Now', 'i', 'am', 'in', 'Hyderabad', 'taking', 'full
stack', 'data', 'science', 'at', 'NareshIT', 'technology', ',', 'under', 'guidanc
e', 'of', 'Mr.', 'Prakash', 'senapati']
```

In [65]:
```
# lets find out the POS
for i in wt:
    print(nltk.pos_tag([i]))
```

```
[('I', 'PRP')]
[('am', 'VBP')]
[('Darshanikanta', 'NN')]
[(',', ',')]
[('i', 'NN')]
[('have', 'VB')]
[('completed', 'VBN')]
[('my', 'PRP$')]
[('graduations', 'NNS')]
[('from', 'IN')]
[('Bhadrak', 'NN')]
[('autonomous', 'JJ')]
[('college', 'NN')]
[(',', ',')]
[('with', 'IN')]
[('83', 'CD')]
[('%', 'NN')]
[('aggrigate', 'NN')]
[('with', 'IN')]
[('9.01', 'CD')]
[('CGPA', 'NN')]
[('.', '.')]
[('Now', 'RB')]
[('i', 'NN')]
[('am', 'VBP')]
[('in', 'IN')]
[('Hyderabad', 'NN')]
[('taking', 'VBG')]
[('fullstack', 'NN')]
[('data', 'NNS')]
[('science', 'NN')]
[('at', 'IN')]
[('NareshIT', 'NN')]
[('technology', 'NN')]
[(',', ',')]
[('under', 'IN')]
[('guidance', 'NN')]
[('of', 'IN')]
[('Mr.', 'NNP')]
[('Prakash', 'NN')]
[('senapati', 'NN')]
```

5. NER (Named Entity recognization)

- this is the classification where all the extracted nouns & phrase are classified into category such as location,names and much more
- some times entity are misclassification
- so if you are use NER in python then you need to import NER_CHUNK from nltk library

In [66]:
```
# import ner_chunks from nltk library
from nltk import ne_chunk
```

In [67]:
```
sentence
```

```
Out[67]: 'I am Darshanikanta, i have completed my graduations from Bhadrak autonomous co
         llege, with 83% aggrigate with 9.01 CGPA.\nNow i am in Hyderabad taking fullsta
         ck data science at  NareshIT technology, under guidance of Mr. Prakash senapat
         i'
```

```
In [68]: # word tokenize
         wt1=word_tokenize(sentence)
         print(wt)
```

```
['I', 'am', 'Darshanikanta', ',', 'i', 'have', 'completed', 'my', 'graduations',
'from', 'Bhadrak', 'autonomous', 'college', ',', 'with', '83', '%', 'aggrigate',
'with', '9.01', 'CGPA', '.', 'Now', 'i', 'am', 'in', 'Hyderabad', 'taking', 'full
stack', 'data', 'science', 'at', 'NareshIT', 'technology', ',', 'under', 'guidanc
e', 'of', 'Mr.', 'Prakash', 'senapati']
```

```
In [69]: # pos_tag
         sen_chunk=nltk.pos_tag(wt1)
         print(sen_chunk)
```

```
[('I', 'PRP'), ('am', 'VBP'), ('Darshanikanta', 'NNP'), (',', ','), ('i', 'NN'),
('have', 'VBP'), ('completed', 'VBN'), ('my', 'PRP$'), ('graduations', 'NNS'),
('from', 'IN'), ('Bhadrak', 'NNP'), ('autonomous', 'JJ'), ('college', 'NN'),
(',', ','), ('with', 'IN'), ('83', 'CD'), ('%', 'NN'), ('aggrigate', 'NN'), ('wit
h', 'IN'), ('9.01', 'CD'), ('CGPA', 'NNP'), ('.', '.'), ('Now', 'RB'), ('i', 'VB
Z'), ('am', 'VBP'), ('in', 'IN'), ('Hyderabad', 'NNP'), ('taking', 'VBG'), ('full
stack', 'NN'), ('data', 'NNS'), ('science', 'NN'), ('at', 'IN'), ('NareshIT', 'NN
P'), ('technology', 'NN'), (',', ','), ('under', 'IN'), ('guidance', 'NN'), ('o
f', 'IN'), ('Mr.', 'NNP'), ('Prakash', 'NNP'), ('senapati', 'VBD')]
```

```
In [70]: # use ne_chunks
         NE_chunk=ne_chunk(sen_chunk)
         print(NE_chunk)
```

```
(S
  I/PRP
  am/VBP
  (GPE Darshanikanta/NNP)
  ,/,
  i/NN
  have/VBP
  completed/VBN
  my/PRP$
  graduations/NNS
  from/IN
  (GPE Bhadrak/NNP)
  autonomous/JJ
  college/NN
  ,/,
  with/IN
  83/CD
  %/NN
  aggrigate/NN
  with/IN
  9.01/CD
  CGPA/NNP
  ./.
  Now/RB
  i/VBZ
  am/VBP
  in/IN
  (GPE Hyderabad/NNP)
  taking/VBG
  fullstack/NN
  data/NNS
  science/NN
  at/IN
  (ORGANIZATION NareshIT/NNP)
  technology/NN
  ,/,
  under/IN
  guidance/NN
  of/IN
  (PERSON Mr./NNP Prakash/NNP)
  senapati/VBD)
```

6. Word cloud

- generate a image based on frequency of the words

In [71]:
```python
#import Word cloud
from wordcloud import WordCloud
```

In [72]:
```python
text='''
The Mahabharata is an ancient Indian epic filled with tales of Dharma (righteous
At the heart of the story are the Pandavas and Kauravas, two royal families clas
Guided by Krishna, the divine charioteer, Arjuna learns about duty and destiny o
where the Bhagavad Gita was born. Key figures like Yudhishthira embody truth and
unwavering loyalty and the burdens of vows. Draupadi, the Pandavas' queen, stand
Karna, the tragic hero and friend of Duryodhana, faces struggles of loyalty and
weaves a complex narrative of family, betrayal, courage, and sacrifice, while th
```

```
Themes of loyalty, sacrifice, justice, karma, and moksha (liberation) run deep,
both external and internal.
'''
```

In [73]:
```
# word tokenize
from nltk.tokenize import word_tokenize
wt_text=word_tokenize(text)
print(wt_text)
```

```
['The', 'Mahabharata', 'is', 'an', 'ancient', 'Indian', 'epic', 'filled', 'with',
'tales', 'of', 'Dharma', '(', 'righteousness', ')', 'and', 'Adharma', '(', 'injus
tice', ')', '.', 'At', 'the', 'heart', 'of', 'the', 'story', 'are', 'the', 'Panda
vas', 'and', 'Kauravas', ',', 'two', 'royal', 'families', 'clashing', 'for', 'pow
er', 'over', 'Hastinapura', '.', 'Guided', 'by', 'Krishna', ',', 'the', 'divine',
'charioteer', ',', 'Arjuna', 'learns', 'about', 'duty', 'and', 'destiny', 'on',
'the', 'battlefield', 'of', 'Kurukshetra', ',', 'where', 'the', 'Bhagavad', 'Git
a', 'was', 'born', '.', 'Key', 'figures', 'like', 'Yudhishthira', 'embody', 'trut
h', 'and', 'righteousness', ',', 'while', 'Bhishma', 'exemplifies', 'unwavering',
'loyalty', 'and', 'the', 'burdens', 'of', 'vows', '.', 'Draupadi', ',', 'the', 'P
andavas', "'", 'queen', ',', 'stands', 'as', 'a', 'symbol', 'of', 'honor', 'and',
'resilience', '.', 'Karna', ',', 'the', 'tragic', 'hero', 'and', 'friend', 'of',
'Duryodhana', ',', 'faces', 'struggles', 'of', 'loyalty', 'and', 'fate', '.', 'Th
e', 'wise', 'Vyasa', ',', 'author', 'of', 'the', 'Mahabharata', ',', 'weaves',
'a', 'complex', 'narrative', 'of', 'family', ',', 'betrayal', ',', 'courage',
',', 'and', 'sacrifice', ',', 'while', 'the', 'vision', 'of', 'Sanjaya', 'offer
s', 'divine', 'insight', 'into', 'the', 'great', 'war', '.', 'Themes', 'of', 'loy
alty', ',', 'sacrifice', ',', 'justice', ',', 'karma', ',', 'and', 'moksha', '(',
'liberation', ')', 'run', 'deep', ',', 'making', 'the', 'Mahabharata', 'a', 'time
less', 'tale', 'of', 'life', "'s", 'battles', ',', 'both', 'external', 'and', 'in
ternal', '.']
```

In [74]:
```
# creat word cloud image
word_cloud=WordCloud(width=500,height=400,margin=1).generate(text)

#plot thw word_cloud
import matplotlib.pyplot as plt

plt.imshow(word_cloud,interpolation='antialiased')
plt.axis('off')
plt.margins(x=0,y=0)
plt.show()
```

completed

```
In [ ]:
```