In [1]:

```
text = """The Moon is a barren, rocky world without air and water.
It has dark lava plain on its surface. The Moon is filled wit craters.
It has no light of its own. It gets its light from the Sun. The Moo keeps
changing its shape as it moves round the Earth. It spins on its axis in 27.3
days stars were named after the Edwin Aldrin were the first ones to set their
foot on the Moon on 21 July 1969 They reached the Moon in their space craft named Apollo II
```

Preprocessing

1. Tokenization

In [2]:

```
# Sentence Tokenization
from nltk.tokenize import sent_tokenize
sentences = sent_tokenize(text)
sentences
```

Out[2]:

```
['The Moon is a barren, rocky world without air and water.',
 'It has dark lava plain on its surface.',
 'The Moon is filled wit craters.',
 'It has no light of its own.',
 'It gets its light from the Sun.',
 'The Moo keeps \nchanging its shape as it moves round the Earth.',
 'It spins on its axis in 27.3 \ndays stars were named after the Edwin Aldri
n were the first ones to set their \nfoot on the Moon on 21 July 1969 They r
eached the Moon in their space craft named Apollo II.']
```

In [ ]:

```
# conjunction ,punctuation,syntax
```

In [3]:

```
# Word tokenization
from nltk.tokenize import word_tokenize
tokens = word_tokenize(text)
tokens
```

Out[3]:

```
['The',
 'Moon',
 'is',
 'a',
 'barren',
 ',',
 'rocky',
 'world',
 'without',
 'air',
 'and',
 'water',
 '.',
 'It',
 'has',
 'dark',
 'lava',
 'plain',
 'on',
 'its',
 'surface',
 '.',
 'The',
 'Moon',
 'is',
 'filled',
 'wit',
 'craters',
 '.',
 'It',
 'has',
 'no',
 'light',
 'of',
 'its',
 'own',
 '.',
 'It',
 'gets',
 'its',
 'light',
 'from',
 'the',
 'Sun',
 '.',
 'The',
 'Moo',
 'keeps',
 'changing',
 'its',
 'shape',
 'as',
```

```
 'it',
 'moves',
 'round',
 'the',
 'Earth',
 '.',
 'It',
 'spins',
 'on',
 'its',
 'axis',
 'in',
 '27.3',
 'days',
 'stars',
 'were',
 'named',
 'after',
 'the',
 'Edwin',
 'Aldrin',
 'were',
 'the',
 'first',
 'ones',
 'to',
 'set',
 'their',
 'foot',
 'on',
 'the',
 'Moon',
 'on',
 '21',
 'July',
 '1969',
 'They',
 'reached',
 'the',
 'Moon',
 'in',
 'their',
 'space',
 'craft',
 'named',
 'Apollo',
 'II',
 '.']
```

In [4]:

```
# white space tokenizer
from nltk.tokenize import WhitespaceTokenizer
tokens1 = WhitespaceTokenizer().tokenize(text)
tokens1
```

Out[4]:

```
['The',
 'Moon',
 'is',
 'a',
 'barren,',
 'rocky',
 'world',
 'without',
 'air',
 'and',
 'water.',
 'It',
 'has',
 'dark',
 'lava',
 'plain',
 'on',
 'its',
 'surface.',
 'The',
 'Moon',
 'is',
 'filled',
 'wit',
 'craters.',
 'It',
 'has',
 'no',
 'light',
 'of',
 'its',
 'own.',
 'It',
 'gets',
 'its',
 'light',
 'from',
 'the',
 'Sun.',
 'The',
 'Moo',
 'keeps',
 'changing',
 'its',
 'shape',
 'as',
 'it',
 'moves',
 'round',
 'the',
 'Earth.',
 'It',
```

```
 'spins',
 'on',
 'its',
 'axis',
 'in',
 '27.3',
 'days',
 'stars',
 'were',
 'named',
 'after',
 'the',
 'Edwin',
 'Aldrin',
 'were',
 'the',
 'first',
 'ones',
 'to',
 'set',
 'their',
 'foot',
 'on',
 'the',
 'Moon',
 'on',
 '21',
 'July',
 '1969',
 'They',
 'reached',
 'the',
 'Moon',
 'in',
 'their',
 'space',
 'craft',
 'named',
 'Apollo',
 'II.']
```

2. Normalization

In [5]:

```
# Lowercase
lowercase = [ word.lower() for word in tokens ]
lowercase
```

Out[5]:

```
['the',
 'moon',
 'is',
 'a',
 'barren',
 ',',
 'rocky',
 'world',
 'without',
 'air',
 'and',
 'water',
 '.',
 'it',
 'has',
 'dark',
 'lava',
 'plain',
 'on',
 'its',
 'surface',
 '.',
 'the',
 'moon',
 'is',
 'filled',
 'wit',
 'craters',
 '.',
 'it',
 'has',
 'no',
 'light',
 'of',
 'its',
 'own',
 '.',
 'it',
 'gets',
 'its',
 'light',
 'from',
 'the',
 'sun',
 '.',
 'the',
 'moo',
 'keeps',
 'changing',
 'its',
 'shape',
 'as',
 'it',
```

```
 'moves',
 'round',
 'the',
 'earth',
 '.',
 'it',
 'spins',
 'on',
 'its',
 'axis',
 'in',
 '27.3',
 'days',
 'stars',
 'were',
 'named',
 'after',
 'the',
 'edwin',
 'aldrin',
 'were',
 'the',
 'first',
 'ones',
 'to',
 'set',
 'their',
 'foot',
 'on',
 'the',
 'moon',
 'on',
 '21',
 'july',
 '1969',
 'they',
 'reached',
 'the',
 'moon',
 'in',
 'their',
 'space',
 'craft',
 'named',
 'apollo',
 'ii',
 '.']
```

3. Stopwords Removal

In [6]:

```
from nltk.corpus import stopwords
stopword_list = stopwords.words("english")
stopword_list
```

```
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
 'who',
 'whom',
 'this',
 'that',
 "that'll",
 'these',
 'those',
 'am',
 'is',
 'are',
 'was',
 'were',
 'be',
 'been',
```

In [8]:

```
text_without_stop = [ word for word in lowercase if word not in stopword_list]
text_without_stop[:10]
```

Out[8]:

```
['moon',
 'barren',
 ',',
 'rocky',
 'world',
 'without',
 'air',
 'water',
 '.',
 'dark']
```

In [9]:

```
lowercase[:10]
```

Out[9]:

```
['the', 'moon', 'is', 'a', 'barren', ',', 'rocky', 'world', 'without', 'ai
r']
```

4. Stemming and Lemmatization

In [12]:

```
from nltk.stem import LancasterStemmer,WordNetLemmatizer
stemmer = LancasterStemmer()
lemmatizer = WordNetLemmatizer()
for word in text_without_stop:
    stemmed_word = stemmer.stem(word)
    lemmatized_word = lemmatizer.lemmatize(word)
    print(f"Original Word  : {word}")
    print(f"stemmed Word : {stemmed_word}")
    print(f"lemmatized Word  : {lemmatized_word}")
    print("*"*100)
```

```
Original Word  : moon
stemmed Word : moon
lemmatized Word  : moon
****************************************************************************************************
Original Word  : barren
stemmed Word : bar
lemmatized Word  : barren
****************************************************************************************************
Original Word  : ,
stemmed Word : ,
lemmatized Word  : ,
****************************************************************************************************
Original Word  : rocky
stemmed Word : rocky
lemmatized Word  : rocky
****************************************************************************************************
```

In [ ]:

```
# porter stemmer,snowball stemmer
```

5. Contraction Mapping

In [15]:

```
# {'doesn't : does not ,didn't : did not }
import contractions
text = " I doesn't like the product"
expanded_text = contractions.fix(text)
expanded_text
```

Out[15]:

```
' I does not like the product'
```

In [16]:

```
from unidecode import unidecode
text = "â, ê, î, ô, û, Â, Ê, Î, Ô, Û I doesn't like the product"
fixed_text = unidecode(text)
fixed_text
```

Out[16]:

```
"a, e, i, o, u, A, E, I, O, U I doesn't like the product"
```

In [ ]:

```
# â, ê, î, ô, û, Â, Ê, Î, Ô, Û = accented characters
```

Remove punctuations

In [17]:

```
from string import punctuation
punctuation
```

Out[17]:

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [18]:

```
text_without_punct = [ word for word in text_without_stop if word not in punctuation]
text_without_punct
```

Out[18]:

```
['moon',
 'barren',
 'rocky',
 'world',
 'without',
 'air',
 'water',
 'dark',
 'lava',
 'plain',
 'surface',
 'moon',
 'filled',
 'wit',
 'craters',
 'light',
 'gets',
 'light',
 'sun',
 'moo',
 'keeps',
 'changing',
 'shape',
 'moves',
 'round',
 'earth',
 'spins',
 'axis',
 '27.3',
 'days',
 'stars',
 'named',
 'edwin',
 'aldrin',
 'first',
 'ones',
 'set',
 'foot',
 'moon',
 '21',
 'july',
 '1969',
 'reached',
 'moon',
 'space',
 'craft',
 'named',
 'apollo',
 'ii']
```

In [ ]:

```
```