In [1]:
```
#Loading NLTK
#TEXT MINING ANALYSIS
#1.NLTK IS A POWERFUL PACKAGE THAT PROVIDES A SET OF DIVERSE NATURAL LANGUAGES ALGORITHM.
#2.IT IS FREE,OPENSOURCE EASY TO USE AND WEEL DOCUMENTED.
#3.NLTK CONSISTS OF THE MOST COMMON ALGORITHMS SUCH AS TOKENZING,PART OO SPEECH TAGGING,STEMMING,SENTIMENT ANALYSIS,
# TOPIC SEGMENTATION,AND NAMED ENTITY RECOGNITION NLTK HELPS THE COMPUTER TO ANALYSIS,PREPROCESS,AND UNDERSTAND THE WRITTEN TEXT.
import nltk
```

In [2]:
```
#SENTENCE TOKENIZATION
from nltk.tokenize import sent_tokenize
text="""Hello Miss.Madhuri,How Have you been?"""
tokenized_sent=sent_tokenize(text)
print(tokenized_sent)
```

```
['Hello Miss.Madhuri,How Have you been?"']
```

In [3]:
```
# WORD TOKENIZATION
from nltk.tokenize import word_tokenize
text="""Hello Miss.Madhuri,How Have you been?"""
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Hello', 'Miss.Madhuri', ',', 'How', 'Have', 'you', 'been', '?']
```

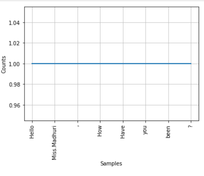
In [4]:
```
#FREQUENCY DISTRIBUTION
from nltk.probability import FreqDist
fdist=FreqDist(tokenized_word)
print(fdist)
```

```
<FreqDist with 8 samples and 8 outcomes>
```

In [5]:
```
fdist.most_common(2)
```

Out[5]:
```
[('Hello', 1), ('Miss.Madhuri', 1)]
```

In [6]:
```
#FREQUENCY DISTRIBUTION PLOT
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```



In [7]:
```
nltk.word_tokenize("Hi, How's it Going")
```

Out[7]:
```
['Hi', ',', 'How', "'s", 'it', 'Going']
```

In [8]:
```
#STOPWORDS
from nltk.corpus import stopwords
stop_words=stopwords.words("english")
print(stop_words)
```

```
['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 [10]:
```
# Removing Words
from nltk.tokenize import sent_tokenize,word_tokenize
from nltk.corpus import stopwords

data="There are Several Type of Programming language."
stopwords=set(stopwords.words('english'))
words=word_tokenize(data)
wordsFiltered=[]

for w in words:
    if w not in stopwords:
        wordsFiltered.append(w)

print(wordsFiltered)
```

```
['There', 'Several', 'Type', 'Programming', 'language', '.']
```

In [11]:
```
#Stemming
import nltk
from nltk.stem import PorterStemmer
#from nltk.tokenize import word_tokenize
stemmer= PorterStemmer()
Input_str="There are several types of stemming Algorithms."
Input_str=nltk.word_tokenize(Input_str)
for word in Input_str:
    print(stemmer.stem(word))
```

```
there
are
sever
type
of
stem
algorithm
.
```

In [13]:
```
#Lemmatization
import nltk
wn=nltk.WordNetLemmatizer()
ps=nltk.PorterStemmer()
dir(wn)
```

Out[13]:
```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 'lemmatize']
```

In [14]:
```
print(ps.stem('study'))
print(ps.stem('studing'))
```

```
studi
stude
```

In [16]:
```
#Stemming Code
import nltk
from nltk.stem.porter import PorterStemmer
porter_stemmer=PorterStemmer()
text="There are Several type of Stemming Algorithm"
tokenization=nltk.word_tokenize(text)
for w in tokenization:
    print("Stemming for {} is {}".format(w,porter_stemmer.stem(w)))
```

```
Stemming for There is there
Stemming for are is are
Stemming for Several is sever
Stemming for type is type
Stemming for of is of
Stemming for Stemming is stem
Stemming for Algorithm is algorithm
```

In [17]:
```
# Lemmatization Code
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer=WordNetLemmatizer()
text="There are Several type of Stemming Algorithm"
tokenization=nltk.word_tokenize(text)
for w in tokenization:
    print("lemma for {} is {}".format(w,wordnet_lemmatizer.lemmatize(w)))
```

```
lemma for There is There
lemma for are is are
lemma for Several is Several
lemma for type is type
lemma for of is of
lemma for Stemming is Stemming
lemma for Algorithm is Algorithm
```

In [18]:
```
#Pos Tagging
import nltk
text=nltk.word_tokenize("python is a Type of Programming language")
nltk.pos_tag(text)
```

Out[18]:
```
[('python', 'NN'),
 ('is', 'VBZ'),
 ('a', 'DT'),
 ('Type', 'NN'),
 ('of', 'IN'),
 ('Programming', 'NNP'),
 ('language', 'NN')]
```

In [19]:
```
nltk.help.upenn_tagset('NN')
```

```
NN: noun, common, singular or mass
    common-carrier cabbage knuckle-duster Casino afghan shed thermostat
    investment slide humour falloff slick wind hyena override subhumanity
    machinist ...
```

In [20]:
```
nltk.help.upenn_tagset('VB,*')
```

```
VB: verb, base form
    ask assemble assess assign assume atone attention avoid bake balkanize
    bank begin behold believe bend benefit bevel beware bless boil bomb
    boost brace break bring broil brush build ...
VBD: verb, past tense
    dipped pleaded swiped regummed soaked tidied convened halted registered
    cushioned exacted snubbed strode aimed adopted belied figgered
    speculated wore appreciated contemplated ...
VBG: verb, present participle or gerund
    telegraphing stirring focusing angering judging stalling lactating
    hankerin' alleging veering capping approaching traveling besieging
    encrypting interrupting erasing wincing ...
VBN: verb, past participle
    multihulled dilapidated aerosolized chaired languished panelized used
    experimented flourished imitated reunifed factored condensed sheared
    unsettled primed dubbed desired ...
VBP: verb, present tense, not 3rd person singular
    predominate wrap resort sue twist spill cure lengthen brush terminate
    appear tend stray glisten obtain comprise detest tease attract
    emphasize mold postpone sever return wag ...
VBZ: verb, present tense, 3rd person singular
    bases reconstructs marks mixes displeases seals carps weaves snatches
    slumps stretches authorizes smolders pictures emerges stockpiles
    seduces fizzes uses bolsters slaps speaks pleads ...
```

In [22]:
```
#Bag of Words
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
```

In [23]:
```
phrases=["python is a type of Programming Language"]
```

In [24]:
```
vect = CountVectorizer()
vect.fit(phrases)
```

Out[24]:
```
CountVectorizer()
```

In [25]:
```
print("Vocabulary size: {}".format(len(vect.vocabulary_)))
print("Vocabulary content:\n {}".format(vect.vocabulary_))
```

```
Vocabulary size: 6
Vocabulary content:
 {'python': 4, 'is': 0, 'type': 5, 'of': 2, 'programming': 3, 'language': 1}
```

In [26]:
```
bag_of_words = vect.transform(phrases)
```

In [27]:
```
print(bag_of_words)
```

```
  (0, 0)        1
  (0, 1)        1
  (0, 2)        1
  (0, 3)        1
  (0, 4)        1
  (0, 5)        1
```

In [28]:
```
print("bag_of_words as an array:\n{}".format(bag_of_words.toarray()))
```

```
bag_of_words as an array:
[[1 1 1 1 1 1]]
```

In [29]:
```
 vect.get_feature_names()
```

Out[29]:
```
['is', 'language', 'of', 'programming', 'python', 'type']
```

In [ ]:
```

```