In [14]:
```
from textblob import TextBlob
import nltk
```

In [4]:
```
b = TextBlob("I havv good spelling")
```

In [5]:
```
b.correct()
```

Out[5]:
```
TextBlob("I have good spelling")
```

In [22]:
```
import nltk
nltk.download('punkt')
b1 = TextBlob("Beautiful is better than ugly."
              "Explicit is better than implicit."
              "Simple is better than complex.")
b1.words
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[22]:
```
WordList(['Beautiful', 'is', 'better', 'than', 'ugly.Explicit', 'is', 'better',
'than', 'implicit.Simple', 'is', 'better', 'than', 'complex'])
```

In [23]:
```
b1.sentences
```

Out[23]:
```
[Sentence("Beautiful is better than ugly.Explicit is better than implicit.Simpl
e is better than complex.")]
```

In [24]:
```
sentence = TextBlob("Use 4 spaces per indentation level")
sentence.words
```

Out[24]:
```
WordList(['Use', '4', 'spaces', 'per', 'indentation', 'level'])
```

In [27]:
```
sentence.words[2].singularize()
```

Out[27]:
```
'space'
```

In [28]:
```
sentence.words[5].pluralize()
```

Out[28]:
```
'levels'
```

In [30]:
```
animals = TextBlob("cat dog octopus")
animals.words
```

Out[30]:
```
WordList(['cat', 'dog', 'octopus'])
```

In [31]:
```
animals.words.pluralize()
```

Out[31]:
```
WordList(['cats', 'dogs', 'octopodes'])
```

In [32]:
```
sen = TextBlob("We are no longer the knights who say Ni." "We are now the knights
sen.word_counts['ekki']
```

Out[32]: 3

In [33]:
```
sen.words.count('ekki')
```

Out[33]: 3

In [34]:
```
sen.words.count('ekki', case_sensitive=True)
```

Out[34]: 2

In [35]:
```
b = TextBlob("And now for something completely different.")
print(b.parse())
```

```
And/CC/O/O now/RB/B-ADVP/O for/IN/B-PP/B-PNP something/NN/B-NP/I-PNP completel
y/RB/B-ADJP/O different/JJ/I-ADJP/O ././O/O
```

In [36]:
```
b1[0:19]
TextBlob("Beautiful is better")
```

Out[36]: TextBlob("Beautiful is better")

In [37]:
```
b1.upper()
```

Out[37]:
```
TextBlob("BEAUTIFUL IS BETTER THAN UGLY.EXPLICIT IS BETTER THAN IMPLICIT.SIMPLE
IS BETTER THAN COMPLEX.")
```

In [39]:
```
b1.find("Simple")
```

Out[39]: 63

In [40]:
```
apple_blob = TextBlob('apples')
banana_blob = TextBlob('bananas')
apple_blob < banana_blob
```

Out[40]: True

In [41]:
```
blob = TextBlob("Now is better then never.")
blob.ngrams(n=3)
```

Out[41]:
```
[WordList(['Now', 'is', 'better']),
 WordList(['is', 'better', 'then']),
 WordList(['better', 'then', 'never'])]
```

In [42]:
```
import nltk
from nltk import tokenize
from nltk.tokenize import sent_tokenize
text = """ Good Day everyone, how are you all today? Its fun learning data analys
text
```

Out[42]:
```
' Good Day everyone, how are you all today? Its fun learning data analysis. Hop
e you all are practicing well.'
```

In [43]:
```
tokenized_text = sent_tokenize(text)
```

In [45]:
```
print(tokenized_text)
```

```
[' Good Day everyone, how are you all today?', 'Its fun learning data analysi
s.', 'Hope you all are practicing well.']
```

In [47]:
```
from nltk.tokenize import word_tokenize
tokenizer_word = word_tokenize(text)
print(tokenizer_word)
```

```
['Good', 'Day', 'everyone', ',', 'how', 'are', 'you', 'all', 'today', '?', 'It
s', 'fun', 'learning', 'data', 'analysis', '.', 'Hope', 'you', 'all', 'are', 'p
racticing', 'well', '.']
```

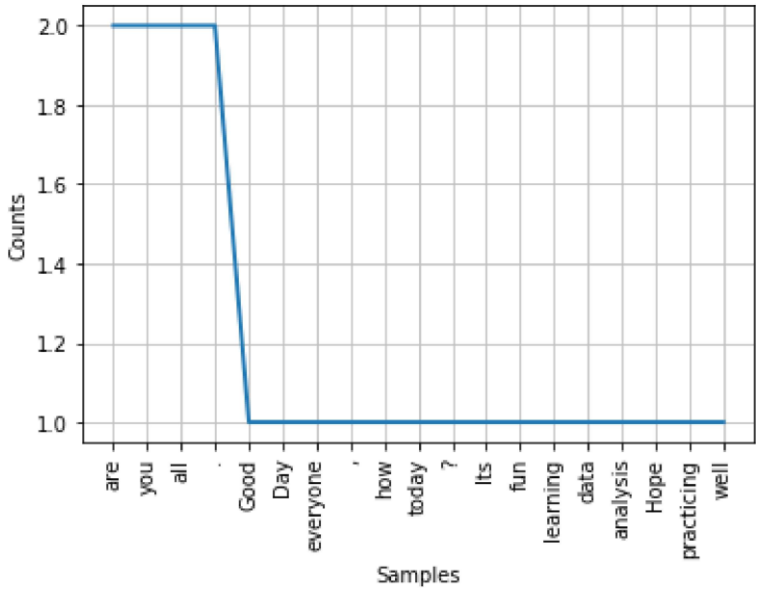
In [48]:
```
from nltk.probability import FreqDist
fdist = FreqDist(tokenizer_word)
print(fdist)
```

```
<FreqDist with 19 samples and 23 outcomes>
```

In [49]:
```
fdist.most_common(4)
```

Out[49]:
```
[('are', 2), ('you', 2), ('all', 2), ('.', 2)]
```

In [50]:
```
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```



In [52]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[52]: True

In [57]:
```
from nltk.corpus import stopwords
stop_words = set(stopwords.words("english"))
print(stop_words)
```

```
{'wasn', 'how', 'herself', 'couldn', 'up', 'during', 'itself', 'than', 't', 'ca
n', 'was', 'mustn', 'yours', 'did', 'didn', 'my', 'at', 'he', 'again', "would
n't", 'myself', "isn't", 'it', "you'll", 'whom', 'any', "you're", "couldn't",
'shouldn', 'all', 'its', 'just', "hasn't", 'won', 'o', "mightn't", 'which', 'ha
sn', 'that', 'theirs', 'over', 'i', 'such', 'am', 'until', 'on', 'through',
's', 'about', 'being', "mustn't", 'under', 'same', 'has', "should've", 'each',
'we', 'them', 'were', 'nor', 'here', "shan't", 'does', 'below', 'why', 'her',
'mightn', 'be', "it's", 'of', 'above', 'aren', 'doesn', 'by', 'wouldn', 'then',
'him', "she's", 'while', 'with', 'for', 'their', 'to', 'out', 'if', 'don', 'our
selves', 'in', 'themselves', 'ours', 'his', 'off', "wasn't", 'the', 'very', 's
o', 'are', 'what', 'from', 'there', 'will', 'yourselves', "that'll", 'she', 'so
me', 'between', "aren't", 'shan', 'who', 'few', 'before', 'weren', 'this', 'bec
ause', 'both', 'd', 'hers', 'll', 'but', "won't", "don't", 'is', 'and', 'wher
e', 'too', "shouldn't", 'ma', 'these', "hadn't", 'you', "you've", 'down', "need
n't", 'or', 'have', 'after', 'only', 'ain', 'own', 'once', "you'd", 've', 'agai
nst', 'y', 'as', 'had', 'haven', 'more', "haven't", 'further', 'no', 'those',
"weren't", 'should', 'been', 'when', 'your', 'me', 'an', 'having', 'isn', 'm',
'doing', "doesn't", 'our', 'do', "didn't", 're', 'hadn', 'they', 'into', 'not',
'himself', 'needn', 'most', 'other', 'now', 'a', 'yourself'}
```

In [60]:
```
filtered_sent=[]
for w in tokenizer_word:
    if w not in stop_words:
        filtered_sent.append(w)

print("Tokenized sentence : ",tokenizer_word)
print("Filtered sentence : ", filtered_sent)
```

```
Tokenized sentence :  ['Good', 'Day', 'everyone', ',', 'how', 'are', 'you', 'al
l', 'today', '?', 'Its', 'fun', 'learning', 'data', 'analysis', '.', 'Hope', 'y
ou', 'all', 'are', 'practicing', 'well', '.']
Filtered sentence :  ['Good', 'Day', 'everyone', ',', 'today', '?', 'Its', 'fu
n', 'learning', 'data', 'analysis', '.', 'Hope', 'practicing', 'well', '.']
```

In [61]:
```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
ps = PorterStemmer()
stemmed_words = []
for w in filtered_sent:
    stemmed_words.append(ps.stem(w))
print("Filtered Sentence : ", filtered_sent)
print("Stemmed Sentence : ", stemmed_words)
```

```
Filtered Sentence :  ['Good', 'Day', 'everyone', ',', 'today', '?', 'Its', 'fu
n', 'learning', 'data', 'analysis', '.', 'Hope', 'practicing', 'well', '.']
Stemmed Sentence :  ['good', 'day', 'everyon', ',', 'today', '?', 'it', 'fun',
'learn', 'data', 'analysi', '.', 'hope', 'practic', 'well', '.']
```

In [62]:
```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\wordnet.zip.
```

Out[62]:
```
True
```

In [64]:
```
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()
from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()
word = "flying"
print("Lemmatizer Word : ", lem.lemmatize(word, "v"))
print("Stemmed Word : ", stem.stem(word))
```

```
Lemmatizer Word
               :  fly
Stemmed Word :  fli
```

In [65]:
```
sent = "Albert Einstien was born in Ulm, Germany in 1879."
tokens = nltk.word_tokenize(sent)
print(tokens)
```

```
['Albert', 'Einstien', 'was', 'born', 'in', 'Ulm', ' '
9' '
                                                , , 'Germany', 'in', '187
  ,  . ]
```

In [66]:
```
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
```

Out[66]:
```
True
```

In [67]:
```
nltk.pos_tag(tokens)
```

Out[67]:
```
[('Albert', 'NNP'),
 ('Einstien', 'NNP'),
 ('was', 'VBD'),
 ('born', 'VBN'),
 ('in', 'IN'),
 ('Ulm', 'NNP'),
 (',', ','),
 ('Germany', 'NNP'),
 ('in', 'IN'),
 ('1879', 'CD'),
 ('.', '.')]
```

In [68]:
```
from collections import Counter
sentence = "Texas A&M University is located in Texas"
term_frequency = Counter(sentence.split())
```

In [69]:
```
term_frequency
```

Out[69]:
```
Counter({'Texas': 2,
         'A&M': 1,
         'University': 1,
         'is': 1,
         'located': 1,
         'in': 1})
```