In [2]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Lenovo\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[2]:
```
True
```

In [4]:
```
from nltk.tokenize import sent_tokenize
text="""Hello Mr. Smith, how are you doing today? The weather is
great, and city is awesome. The sky is pinkish-blue. You shouldn't
eat cardboard"""
tokenized_text=sent_tokenize(text)
print(tokenized_text)
```

```
['Hello Mr. Smith, how are you doing today?', 'The weather is\ngreat, and city is aw
esome.', 'The sky is pinkish-blue.', "You shouldn't\neat cardboard"]
```

In [5]:
```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '?', 'The', 'w
eather', 'is', 'great', ',', 'and', 'city', 'is', 'awesome', '.', 'The', 'sky', 'is
', 'pinkish-blue', '.', 'You', "shouldn't", 'eat', 'cardboard']
```

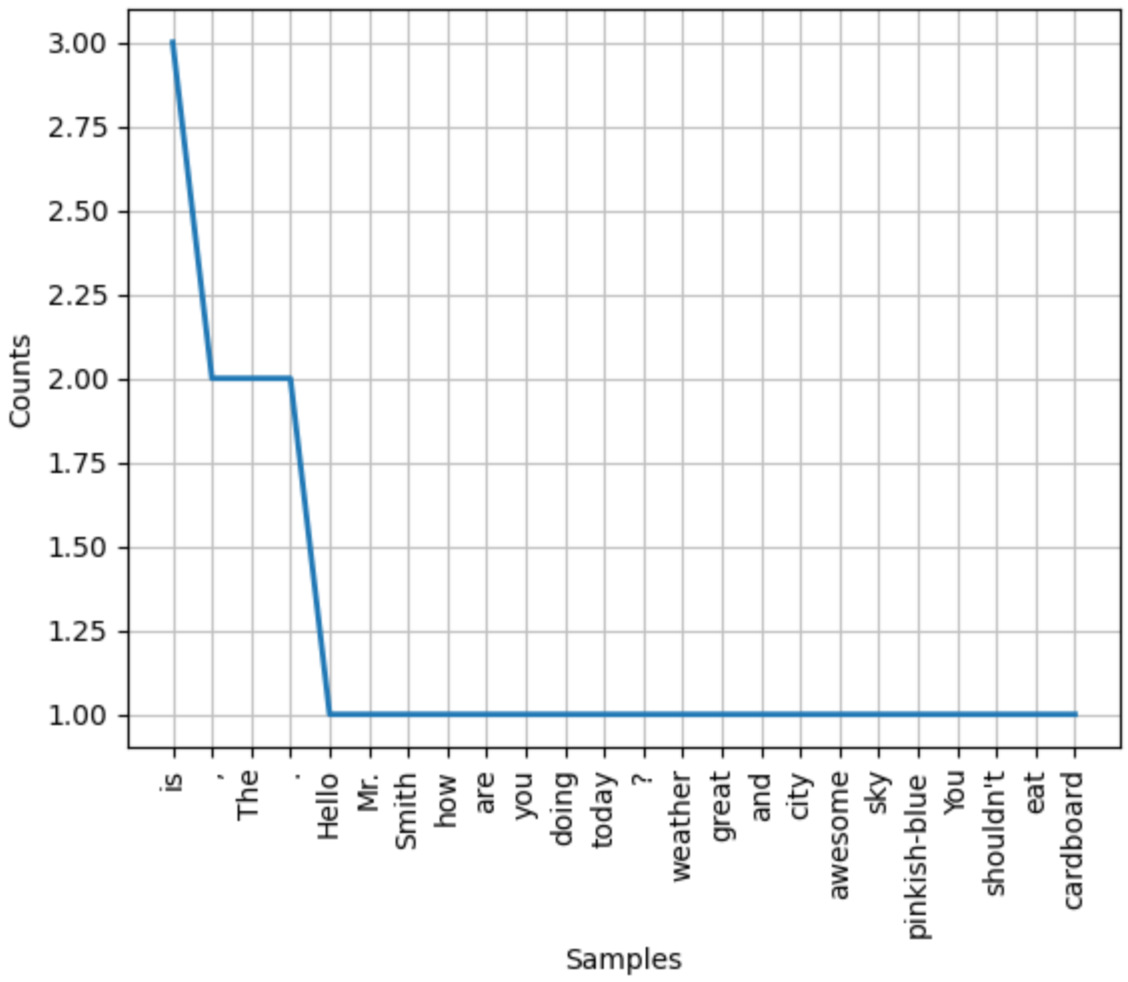
In [6]:
```
from nltk.probability import FreqDist
fdist = FreqDist(tokenized_word)
print(fdist)
```

```
<FreqDist with 24 samples and 29 outcomes>
```

In [7]:
```
fdist.most_common(2)
```

Out[7]:
```
[('is', 3), (',', 2)]
```

In [8]:
```
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```



In [9]:
```
sent = "Albert Einstein was born in Ulm, Germany in 1879."
tokens=nltk.word_tokenize(sent)
print(tokens)
```

```
['Albert', 'Einstein', 'was', 'born', 'in', 'Ulm', ',', 'Germany', 'in', '1879',
'.']
```

In [10]:
```
nltk.download('averaged_perceptron_tagger')
nltk.pos_tag(tokens)
```

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

Out[10]:
```
[('Albert', 'NNP'),
 ('Einstein', 'NNP'),
 ('was', 'VBD'),
 ('born', 'VBN'),
 ('in', 'IN'),
 ('Ulm', 'NNP'),
 (',', ','),
 ('Germany', 'NNP'),
 ('in', 'IN'),
 ('1879', 'CD'),
 ('.', '.')]
```

In [13]:
```
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Lenovo\AppData\Roaming\nltk_data...
{"wasn't", 'than', 'why', 'has', 'any', 'd', "you'd", 'if', 'no', 'does', 'yours', "
doesn't", 'my', 'haven', 'he', 'up', 'we', 'ours', 'under', 'for', 'wasn', 'until',
"you'll", 'myself', "you've", 'can', 'not', 's', "shan't", "she's", 'its', 'hasn', '
couldn', 'this', "isn't", 've', 'there', 'just', "couldn't", 'after', 'about', "don'
t", "won't", 'these', 'they', 'y', 'ma', 'only', 'very', 'during', 'in', 'then', 'su
ch', 'down', 'between', 'needn', 'so', 'both', "that'll", 'mustn', 'hadn', 'your', '
were', 'ourselves', 'm', 'do', 'other', "hadn't", 'don', 'itself', 'out', 'shouldn',
'will', 'you', 'him', 'himself', 'had', 'when', 'nor', 'it', 'here', 'through', "hav
en't", 'his', 'own', 'aren', 'am', 'herself', 't', 'now', 'of', 'mightn', 'our', 'll
', 'be', 'what', 'a', 'above', 'are', 'being', 'didn', 'i', 'ain', 'because', 'few',
'whom', 'more', 'some', 'theirs', 'or', 'doing', 'hers', 'each', 'and', 'isn', 'how
', 'weren', 'wouldn', "mightn't", 'off', 'yourself', 'while', "shouldn't", "wouldn'
t", 'once', "aren't", 'at', 'which', 'that', 'too', 'having', "didn't", 'from', 'bel
ow', 'those', 'further', 'their', 'but', 'into', 'who', 'was', 'shan', 'most', 'by',
'won', 'them', 'doesn', 'is', 'on', "should've", "you're", 'should', 'to', 'the', 'm
e', 'yourselves', 'where', 'before', 'all', 'against', 'she', 'over', "hasn't", 'o',
'with', "it's", 'her', 'did', 'same', "needn't", 're', 'themselves', "mustn't", "wer
en't", 'again', 'have', 'been', 'an', 'as'}
[nltk_data]   Unzipping corpora\stopwords.zip.
```

In [19]:
```
filtered_sent=[]
for w in tokenized_word:
    if w not in stop_words:
        filtered_sent.append(w)

print("Filterd Sentence:",filtered_sent)
print("Tokenized Sentence:",tokenized_word)
```

```
Filterd Sentence: ['Hello', 'Mr.', 'Smith', ',', 'today', '?', 'The', 'weather', 'gr
eat', ',', 'city', 'awesome', '.', 'The', 'sky', 'pinkish-blue', '.', 'You', 'eat',
'cardboard']
Tokenized Sentence: ['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'to
day', '?', 'The', 'weather', 'is', 'great', ',', 'and', 'city', 'is', 'awesome',
'.', 'The', 'sky', 'is', 'pinkish-blue', '.', 'You', "shouldn't", 'eat', 'cardboard
']
```

In [21]:
```
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: ['Hello', 'Mr.', 'Smith', ',', 'today', '?', 'The', 'weather', 'g
reat', ',', 'city', 'awesome', '.', 'The', 'sky', 'pinkish-blue', '.', 'You', 'eat',
'cardboard']
Stemmed Sentence: ['hello', 'mr.', 'smith', ',', 'today', '?', 'the', 'weather', 'gr
eat', ',', 'citi', 'awesom', '.', 'the', 'sky', 'pinkish-blu', '.', 'you', 'eat', 'c
ardboard']
```

In [24]:
```
nltk.download('wordnet')
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()
from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()
word = "flying"
print("Lemmatized Word:",lem.lemmatize(word,"v"))
print("Stemmed Word:",stem.stem(word))
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Lenovo\AppData\Roaming\nltk_data...
Lemmatized Word: fly
Stemmed Word: fli
```

In [25]:
```
import pandas as pd
import sklearn as sk
import math
```

In [26]:
```
first_sentence = "Data Science is the sexiest job of the 21st century"
second_sentence = "machine learning is the key for data science"#split so each word
first_sentence= first_sentence.split(" ")
second_sentence = second_sentence.split(" ")#join them to remove common duplicate w
total=set(first_sentence).union(set(second_sentence))
print(total)
```

```
{'sexiest', 'job', 'machine', 'Data', 'Science', 'the', 'century', 'science', 'of',
'key', 'is', 'for', '21st', 'data', 'learning'}
```

In [27]:
```
wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)
for word in first_sentence:
    wordDictA[word]+=1

for word in second_sentence:
    wordDictB[word]+=1
```

In [28]:
```
pd.DataFrame([wordDictA, wordDictB])
```

Out[28]:

	sexiest	job	machine	Data	Science	the	century	science	of	key	is	for	21st	dat
0	1	1	0	1	1	2	1	0	1	0	1	0	1	
1	0	0	1	0	0	1	0	1	0	1	1	1	0	

In [31]:
```
def computeTF(wordDict, doc):
    tfDict = {}
    corpusCount = len(doc)
    for word, count in wordDict.items():
        tfDict[word] = count/float(corpusCount)
    return(tfDict)#running our sentences through the tffunction:
tfFirst = computeTF(wordDictA, first_sentence)
tfSecond = computeTF(wordDictB, second_sentence)#Converting todataframe for
visualizationtf = pd.DataFrame([tfFirst,tfSecond])
visualizationtf
```

Out[31]:

	sexiest	job	machine	Data	Science	the	century	science	of	key	is	for
0	0.1	0.1	0.000	0.1	0.1	0.200	0.1	0.000	0.1	0.000	0.100	0.000
1	0.0	0.0	0.125	0.0	0.0	0.125	0.0	0.125	0.0	0.125	0.125	0.125

In [33]:
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words =set(stopwords.words('english'))
filtered_sentence = [w for w in wordDictA if not w in stop_words]
print(filtered_sentence)
```

```
['sexiest', 'job', 'machine', 'Data', 'Science', 'century', 'science', 'key', '21st
', 'data', 'learning']
```

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

In [36]:
```
def computeIDF(docList):
    idfDict = {}
    N = len(docList)

    idfDict = dict.fromkeys(docList[0].keys(), 0)
    for word, val in idfDict.items():
        idfDict[word] = math.log10(N / (float(val) + 1))

    return(idfDict)#inputing our sentences in the log file
idfs = computeIDF([wordDictA, wordDictB])
```

In [37]:
```
def computeTFIDF(tfBow, idfs):
    tfidf = {}
    for word, val in tfBow.items():
        tfidf[word] = val*idfs[word]
    return(tfidf)
#running our two sentences through the IDF:
idfFirst =computeTFIDF(tfFirst, idfs)
idfSecond = computeTFIDF(tfSecond, idfs)
#putting it in a dataframe
idf= pd.DataFrame([idfFirst, idfSecond])
print(idf)
```

```
    sexiest       job   machine      Data   Science       the   century  \
0  0.030103  0.030103  0.000000  0.030103  0.030103  0.060206  0.030103   
1  0.000000  0.000000  0.037629  0.000000  0.000000  0.037629  0.000000   

    science        of       key        is       for      21st      data  \
0  0.000000  0.030103  0.000000  0.030103  0.000000  0.030103  0.000000   
1  0.037629  0.000000  0.037629  0.037629  0.037629  0.000000  0.037629   

   learning  
0  0.000000  
1  0.037629  
```

In [43]:

```
#first step is to import the library
from sklearn.feature_extraction.text import TfidfVectorizer
#for the sentence, make sure all words are lowercase or you will run
#into error. for simplicity, I just made the same sentence all
#lowercase
firstV= "Data Science is the sexiest job of the 21st century"
secondV= "machine learning is the key for data science"
#calling the TfidfVectorizer
vectorize= TfidfVectorizer()
#fitting the model and passing our sentences right away:
response= vectorize.fit_transform([firstV, secondV])
print(response)
```

```
  (0, 1)	0.34211869506421816
  (0, 0)	0.34211869506421816
  (0, 9)	0.34211869506421816
  (0, 5)	0.34211869506421816
  (0, 11)	0.34211869506421816
  (0, 12)	0.48684053853849035
  (0, 4)	0.24342026926924518
  (0, 10)	0.24342026926924518
  (0, 2)	0.24342026926924518
  (1, 3)	0.40740123733358447
  (1, 6)	0.40740123733358447
  (1, 7)	0.40740123733358447
  (1, 8)	0.40740123733358447
  (1, 12)	0.28986933576883284
  (1, 4)	0.28986933576883284
  (1, 10)	0.28986933576883284
  (1, 2)	0.28986933576883284
```

In [ ]: