In [9]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\rakhe\anaconda3\lib\site-packages (3.5)
Requirement already satisfied: tqdm in c:\users\rakhe\anaconda3\lib\site-packages (from nltk) (4.50.2)
Requirement already satisfied: regex in c:\users\rakhe\anaconda3\lib\site-packages (from nltk) (2020.10.15)
Requirement already satisfied: click in c:\users\rakhe\anaconda3\lib\site-packages (from nltk) (7.1.2)
Requirement already satisfied: joblib in c:\users\rakhe\anaconda3\lib\site-packages (from nltk) (0.17.0)
```

In [10]:
```
import nltk
nltk.download("punkt")
```

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

Out[10]:
```
True
```

In [11]:
```
text = """The voice that navigated was definitely that of a machine, and yet you could tell that the machine was a woman,
which hurt my mind a little. How can machines have genders? The machine also had an American accent.
How can machines have nationalities? This can't be a good idea, making machines talk like real people, can it?
Giving machines humanoid identities?"""
text
```

Out[11]:
```
"The voice that navigated was definitely that of a machine, and yet you could tell that the machine was a woman, \nwhich hurt my m
ind a little. How can machines have genders? The machine also had an American accent.\nHow can machines have nationalities? This c
an't be a good idea, making machines talk like real people, can it? \nGiving machines humanoid identities?"
```

In [12]:
```
from nltk.tokenize import sent_tokenize
text_to_sentence = sent_tokenize(text)
for str in text_to_sentence:
    print(str)
```

```
The voice that navigated was definitely that of a machine, and yet you could tell that the machine was a woman,
which hurt my mind a little.
How can machines have genders?
The machine also had an American accent.
How can machines have nationalities?
This can't be a good idea, making machines talk like real people, can it?
Giving machines humanoid identities?
```

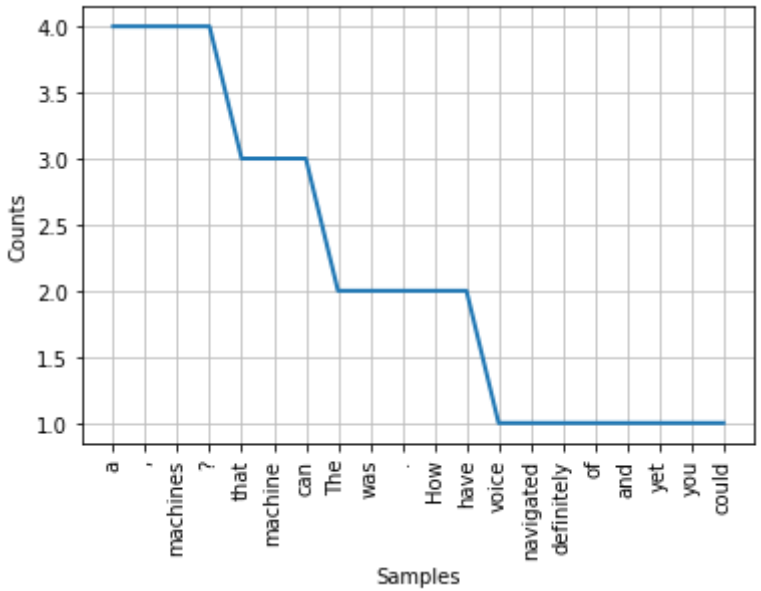
In [13]:
```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['The', 'voice', 'that', 'navigated', 'was', 'definitely', 'that', 'of', 'a', 'machine', ',', 'and', 'yet', 'you', 'could', 'tel
```

```
l', 'that', 'the', 'machine', 'was', 'a', 'woman', ',', 'which', 'hurt', 'my', 'mind', 'a', 'little', '.', 'How', 'can', 'machine
s', 'have', 'genders', '?', 'The', 'machine', 'also', 'had', 'an', 'American', 'accent', '.', 'How', 'can', 'machines', 'have', 'n
ationalities', '?', 'This', 'ca', "n't", 'be', 'a', 'good', 'idea', ',', 'making', 'machines', 'talk', 'like', 'real', 'people',
',', 'can', 'it', '?', 'Giving', 'machines', 'humanoid', 'identities', '?']
```

In [ ]:
```
from nltk.probability import FreqDist
freq_dist_of_words = FreqDist(tokenized_word)
print(freq_dist_of_words)
freq_dist_of_words.most_common(73)
```

In [15]:
```
import matplotlib.pyplot as plt
freq_dist_of_words.plot(20,cumulative=False)
plt.show()
```



Filtering Stop Words - Stop words are used to filter some words which are repetitive and don’t hold any information. For example, words like – {that these, below, is, are, etc.} don’t provide any information, so they need to be removed from the text. Stop Words are considered as Noise. NLTK provides a huge list of stop words.

In [17]:
```
nltk.download('stopwords')
```

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

Out[17]: True

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

printing the stop words from a list

In [20]:
```
text1 = 'Learn to lose your destiny to find where it leads you'
filtered_text = []
tokenized_word = word_tokenize(text1)
for each_word in tokenized_word:
    if each_word not in stop_words:
        filtered_text.append(each_word)
print('Toxenized list with stop words: {}'.format(tokenized_word))
print('Toxenized list with out stop words: {}'.format(filtered_text))
```

```
Toxenized list with stop words: ['Learn', 'to', 'lose', 'your', 'destiny', 'to', 'find', 'where', 'it', 'leads', 'you']
Toxenized list with out stop words: ['Learn', 'lose', 'destiny', 'find', 'leads']
```

print all the stop words from your tokenized word list