In [ ]:
```
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')
```

Two things I learned about Text objects:

- Text objects support slices
- The name attribute of a Text object is either the first 8 words or, if there is a square bracket in the first 20 words, the words contained in the square brackets

In [ ]:
```
from nltk.book import text1
print(text1.tokens[:20])
```

```
['[', 'Moby', 'Dick', 'by', 'Herman', 'Melville', '1851', ']', 'ETYMOLOGY', '.',
'(', 'Supplied', 'by', 'a', 'Late', 'Consumptive', 'Usher', 'to', 'a', 'Grammar']
```

The above code imports one of NLTK's built in texts and prints the first 20 tokens of that text.

In [ ]:
```
print(text1.concordance('sea', lines=5))
```

```
Displaying 5 of 455 matches:
 shall slay the dragon that is in the sea ." -- ISAIAH " And what thing soever
 S PLUTARCH ' S MORALS . " The Indian Sea breedeth the most and the biggest fis
cely had we proceeded two days on the sea , when about sunrise a great many Wha
many Whales and other monsters of the sea , appeared . Among the former , one w
 waves on all sides , and beating the sea before him into a foam ." -- TOOKE '
None
```

The above code takes the imported Text object and prints 5 lines that include the word 'sea' along with the surrounding text.

The count() function for Text objects calls count() on the instance token list. This is the same as Python's built in count() for lists.

In [ ]:
```
print(f'text1.count():        {text1.count("sea")}')
print(f'text1.tokens.count(): {text1.tokens.count("sea")}')
```

```
text1.count():        433
text1.tokens.count(): 433
```

The above code prints the result of using the Text object count() function and the built in count() function for lists. The results of both are the same.

The following uses an excerpt found on natethesnake.com

In [ ]:
```
raw_text = "By the end of the day, he starts getting worried. He figures he's been
tokens = nltk.word_tokenize(raw_text)
print(tokens[:10])
```

```
['By', 'the', 'end', 'of', 'the', 'day', ',', 'he', 'starts', 'getting']
```

The above code tokenizes the provided text and prints the first 10 tokens of the provided text.

In [ ]:
```
sentences = nltk.sent_tokenize(raw_text)
print(sentences)
```

```
['By the end of the day, he starts getting worried.', "He figures he's been walking
at least three miles an hour, according to his watch for over ten hours.", 'That me
ans that if his estimate was right, he should be close to the town.', "Unfortunatel
y, he doesn't recognize any of this.", "He had to cross a dry creek bed a mile or t
wo back, and he doesn't remember coming through it in the SUV."]
```

The above code splits the text into its component sentences.

In [ ]:
```
stemmer = nltk.stem.PorterStemmer()
print([stemmer.stem(token) for token in tokens])
```

```
['by', 'the', 'end', 'of', 'the', 'day', ',', 'he', 'start', 'get', 'worri', '.', '
he', 'figur', 'he', "'s", 'been', 'walk', 'at', 'least', 'three', 'mile', 'an', 'ho
ur', ',', 'accord', 'to', 'hi', 'watch', 'for', 'over', 'ten', 'hour', '.', 'that',
'mean', 'that', 'if', 'hi', 'estim', 'wa', 'right', ',', 'he', 'should', 'be', 'clo
se', 'to', 'the', 'town', '.', 'unfortun', ',', 'he', 'doe', "n't", 'recogn', 'ani
', 'of', 'thi', '.', 'he', 'had', 'to', 'cross', 'a', 'dri', 'creek', 'bed', 'a', '
mile', 'or', 'two', 'back', ',', 'and', 'he', 'doe', "n't", 'rememb', 'come', 'thro
ugh', 'it', 'in', 'the', 'suv', '.']
```

The above code uses NLTK's PorterStemmer to print the tokens in their stemmed forms.

Differences between stemming and lemmatizing (stem - lemma):

- by - By
- get - getting
- he - He
- figur - figure
- walk - walking
- accord - according
- hi - his

In [ ]:
```
lemmatizer = nltk.stem.WordNetLemmatizer()
print([lemmatizer.lemmatize(token) for token in tokens])
```

```
['By', 'the', 'end', 'of', 'the', 'day', ',', 'he', 'start', 'getting', 'worried',
'.', 'He', 'figure', 'he', "'s", 'been', 'walking', 'at', 'least', 'three', 'mile',
'an', 'hour', ',', 'according', 'to', 'his', 'watch', 'for', 'over', 'ten', 'hour',
'.', 'That', 'mean', 'that', 'if', 'his', 'estimate', 'wa', 'right', ',', 'he', 'sh
ould', 'be', 'close', 'to', 'the', 'town', '.', 'Unfortunately', ',', 'he', 'doe',
"n't", 'recognize', 'any', 'of', 'this', '.', 'He', 'had', 'to', 'cross', 'a', 'dry
', 'creek', 'bed', 'a', 'mile', 'or', 'two', 'back', ',', 'and', 'he', 'doe', "n'
t", 'remember', 'coming', 'through', 'it', 'in', 'the', 'SUV', '.']
```

The above code uses NLTK's WordNetLemmatizer to print the tokens in lemmatized forms.

I think the NLTK library provides a lot of useful functionalities for processing bodies of text. It is far easier to use functions from NLTK than it is to try to handwrite functions that achieve similar goals. I think the code quality of NLTK is very good. It is easy to read and understand, and it is well-documented. In future projects, I can use NLTK to process a body of text before feeding it to some NLP application, as it is easier to process tokenized and lemmatized text. NLTK can also provide metrics and statistical information about a body of text, which is also useful for text processing.