Install all the required NLTK libraries and book.

```
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')
nltk.download('book')
```

Extract the first 20 tokens from text1.

I learned that tokens is an attribute and not a method.

I also learned that it splits contractions.

```
text1 = nltk.book.text1
print(text1.tokens[:20])
```

```
['[', 'Moby', 'Dick', 'by', 'Herman', 'Melville', '1851', ']', 'ETYMOLOGY', '.', '(',
```

Print a concordance for text1 word 'sea', selecting only 5 lines.

```
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 count method on a concordance works by counting the number of times a word appears in a text.

The method is just a call to Python's List count method on the tokens, so both lines of code below do exactly the same thing.

```
print(text1.count('sea'))
print(text1.tokens.count('sea'))
```

```
433
433
```

Using 5 sentences of raw text made by me and printing the first 10 tokens of that text.

```
raw_text = "Hello, how are you doing on this fine Monday evening? I am doing great, thank y
tokens = nltk.word_tokenize(raw_text)
print(tokens[:10])
```

```
['Hello', ',', 'how', 'are', 'you', 'doing', 'on', 'this', 'fine', 'Monday']
```

Perform sentence segmentation and display the sentences.

```
sents = nltk.sent_tokenize(raw_text)
print(sents)
```

```
['Hello, how are you doing on this fine Monday evening?', 'I am doing great, thank yc
```

Write a list comprehension to stem the text and display it.

```
stemmer = nltk.PorterStemmer()
print([stemmer.stem(x) for x in nltk.word_tokenize(raw_text)])
```

```
['hello', ',', 'how', 'are', 'you', 'do', 'on', 'thi', 'fine', 'monday', 'even', '?',
```

Write a list comprehension to lemmatize the text and display it.

- The stemmer makes everything lowercase, the lemmatizer does not.
- The stemmer chops off much more-- "thi" vs. "this" from the lemmatizer.
- The stemmer does not take into account the semantics, only syntax--"evening" is interpreted as a verb and stemmed to "even", compared to the full "evening" from the lemmatizer.
- The stemmer converts words ending in "y" to "i", like "realli" vs. "really".
- The lemmatizer makes no change to the text, while the stemmer makes any.

```
lemmatizer = nltk.WordNetLemmatizer()
print([lemmatizer.lemmatize(x) for x in nltk.word_tokenize(raw_text)])
```

```
['Hello', ',', 'how', 'are', 'you', 'doing', 'on', 'this', 'fine', 'Monday', 'evening
```

-

Though we didn't explore much of it for this exercise, I feel like the NLTK library has a lot to offer in terms of functionality. The code is very Pythonic, which is not what I'm used to reading, but it seems to be written succinctly. In the future I might use NLTK as a convenient way to process linguistic text and make use of it, such as improving user searches by simplifying their conjugations of words to make it easier for a search algorithm or more easily analyzing the counts of words in some speech or writing to determine tone or overall uniqueness of speech (based on number of unique words).