```
In [103…  import nltk
          from nltk.book import *
          from nltk import word_tokenize
          from nltk import sent_tokenize
          from nltk import PorterStemmer
          from nltk import WordNetLemmatizer
```

The above code block imports nltk, nltk.book, word_tokenize, sent_tokenize

```
In [104…  text1.tokens[:20]

Out[104…  ['[',
           'Moby',
           'Dick',
           'by',
           'Herman',
           'Melville',
           '1851',
           ']',
           'ETYMOLOGY',
           '.',
           '(',
           'Supplied',
           'by',
           'a',
           'Late',
           'Consumptive',
           'Usher',
           'to',
           'a',
           'Grammar']
```

The above code cell gets the first 20 tokens of text1, as described in question 3.

```
In [105…  text1.concordance('sea', 79, 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 '
```

The above code uses the built in concordance method to get the first 5 instances of the word 'sea' in text1, as described in question 4.

Question 5 - The count() method in the API accesses the list of tokens and counts the number of times the token occurs in the list. This is like Python’s count method as Python’s does basically the same thing, counting the amount of times a specified item is in a list.

```
In [106…  print(text1.count('sea'))
          countList = ['sea','test', 'dog', 'sea']
          print(countList.count('sea'))

          433
          2
```

The above code demos the API's count() method first, and the built in Python count() method second.

```
In [107…  raw_text = 'Voldemort himself created his worst enemy, just as tyrants everywhere do! Have you any idea how muc
          tokens = word_tokenize(raw_text)
          print(tokens[:10])

          ['Voldemort', 'himself', 'created', 'his', 'worst', 'enemy', ',', 'just', 'as', 'tyrants']
```

Using the same text from Quiz 2, found in Harry Potter and the Half-Blood Prince, I used the word_tokenize() method to create a list of all the words as tokens, then printed the first 10 tokens from the list.

```
In [108…  sent_tokenize(raw_text)

Out[108…  ['Voldemort himself created his worst enemy, just as tyrants everywhere do!',
           'Have you any idea how much tyrants fear the people they oppress?',
           'All of them realize that, one day, amongst their many victims, there is sure to be one who rises against them
          and strikes back!']
```

The above code uses the same raw_text variable, and the sent_tokenize() method to create tokens of sentences from the text.

```
In [109…  ps = PorterStemmer()
          tokens = nltk.word_tokenize(raw_text)
          stem = [ps.stem(t) for t in tokens]
          print(stem)

          ['voldemort', 'himself', 'creat', 'hi', 'worst', 'enemi', ',', 'just', 'as', 'tyrant', 'everywher', 'do', '!',
          'have', 'you', 'ani', 'idea', 'how', 'much', 'tyrant', 'fear', 'the', 'peopl', 'they', 'oppress', '?', 'all',
          'of', 'them', 'realiz', 'that', ',', 'one', 'day', ',', 'amongst', 'their', 'mani', 'victim', ',', 'there', 'i
          s', 'sure', 'to', 'be', 'one', 'who', 'rise', 'against', 'them', 'and', 'strike', 'back', '!']
```

The above code uses the built in PorterStemmer() class to stem the same raw_text variable. The stemmed list is then printed.

Differences -
stem lowercases, lemma does not
stem trims, lemma does not
not all stem tokens are real words, lemma’s are
stem goes for stem of word, lemma doesn’t always
stemming is more crude, lemma keeps its natural

```
In [110…  lem = WordNetLemmatizer()
          lemList = [lem.lemmatize(t) for t in tokens]
          print(lemList)

          ['Voldemort', 'himself', 'created', 'his', 'worst', 'enemy', ',', 'just', 'a', 'tyrant', 'everywhere', 'do',
          '!', 'Have', 'you', 'any', 'idea', 'how', 'much', 'tyrant', 'fear', 'the', 'people', 'they', 'oppress', '?', 'A
          ll', 'of', 'them', 'realize', 'that', ',', 'one', 'day', ',', 'amongst', 'their', 'many', 'victim', ',', 'ther
          e', 'is', 'sure', 'to', 'be', 'one', 'who', 'rise', 'against', 'them', 'and', 'strike', 'back', '!']
```

The above code uses the built in WordNetLemmatizer() class to lemmatize the tokens list and then the list is printed.

Comment Cell -
a. Your opinion of the functionality of the NLTK library - I think the NLTK has great functionality, it processes the words accurately and the tools (that I have used) seem to work very well.
b. Your opinion of the code quality of the NLTK library - The code quality seems to be high, all the tools I used seem to work very well, and the API is well documented.
c. A list of ways you may use NLTK in future projects - Any time I need basic tokenization, analysis, or stemming/lemmatization, I can use NLTK, as I am comfortbale with these tools.