Importing nltk and all the contents from book

```
import nltk
from nltk.book import *
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

3. List two things you learned about the tokens() method or Text Object:

1. tokens() returns a list of strings
2. to print tokens you just use '.tokens'. You can't use '.tokens()'. If you want to specify how many tokens you can use '.tokens[]'

```
# Using tokens method from text object to print out the first 20 tokens
tokens = text1.tokens[:20]
print(tokens)
```

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

```
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 '
```

5. Look at the count() method in the API. How does this work, and how is it different or the same as Python's count method?

- With python count(), you can get the count for words, or substrings. With substring count(), you can add additional parameters like start and end points.
- Whereas, in nltk text object count() you can only count for the one parameter, the object or word you are counting. Using text object, you get lower count of the words than when reading the raw text.

```
from nltk.corpus import gutenberg
raw_textmd = gutenberg.raw('melville-moby_dick.txt')
raw_count = raw_textmd.count('sea')
print("Raw text count:", raw_count)
print("Text object count:", text1.count('sea'))
```

```
Raw text count: 702
Text object count: 433
```

```
print(text1.count('sea', 1, 52)) #text object with multiple parameters provides error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_772/1464623362.py in <module>
----> 1 print(text1.count('sea', 1, 52)) #text object with multiple parameters provides error

TypeError: count() takes 2 positional arguments but 4 were given
```

```
#showing python count method allows multiple parameters
raw_text = "This is raw text. This is raw text. This is raw text. This is raw text."
sstring = "This is"
count = raw_text.count(sstring, 0, 100)
print(count)
```

```
4
```

I made up five sentences. Tokenized the raw text using nltk's word tokenizer method, and printed the first 10 tokens

```
from nltk import word_tokenize
raw_text = "I am creating five sentences. This is for an assignment. I am starting to like playing around with nltk. Of course, playing around with already developed code is much easier. Than creating code, to play around with."
tokens = word_tokenize(raw_text)
print(tokens[:10])
```

```
['I', 'am', 'creating', 'five', 'sentences', '.', 'This', 'is', 'for', 'an']
```

Using builtin sentence tokenizer from nltk, here we tokenize sentences from the raw text above

```
from nltk import sent_tokenize
sentences = sent_tokenize(raw_text)
print(sentences)
```

```
['I am creating five sentences.', 'This is for an assignment.', 'I am starting to like playing around with nltk.', 'Of cours
e, playing around with already developed code is much easier.', 'Than creating code, to play around with.']
```

Stemming the raw text from text1 (Moby Dick) using nltk's PorterStemmer()

```
from nltk.stem.porter import *
stemmer = PorterStemmer()
stemmed = [stemmer.stem(t) for t in text1]
print(stemmed)
```

```
'flag', 'of', 'all', 'the', 'known', 'nation', 'of', 'the', 'world', '.', 'he', 'love', 'to', 'dust', 'hi', 'old', 'gramm
ar', ';', 'it', 'somehow', 'mildli', 'remind', 'him', 'of', 'hi', 'mortal', '.', '"', 'while', 'you', 'take', 'in', 'han
d', 'to', 'school', 'other', ',', 'and', 'to', 'teach', 'them', 'by', 'what', 'name', 'a', 'whale', '-', 'fish', 'is', 't
o', 'be', 'call', 'in', 'our', 'tongu', 'leav', 'out', ',', 'through', 'ignor', ',', 'the', 'letter', 'h', ',', 'which',
'almost', 'alon', 'maketh', 'the', 'signif', 'of', 'the', 'word', ',', 'you', 'deliv', 'that', 'which', 'is', 'not', 'tru
e', '."', '--', 'hackluyt', '"', 'whale', '.', '...', 'sw', '.', 'and', 'dan', '.', 'hval', '.', 'thi', 'anim', 'is', 'na
me', 'from', 'round', 'or', 'roll', ';', 'for', 'in', 'dan', '.', 'hvalt', 'is', 'arch', 'or', 'vault', '."', '--', 'webs
ter', "'", 's', 'dictionari', '"', 'whale', '.', '...', 'it', 'is', 'more', 'immedi', 'from', 'the', 'dut', '.', 'and',
'ger', '.', 'wallen', ';', 'a', '.', 's', '.', 'walw', '-', 'ian', ',', 'to', 'roll', ',', 'to', 'wallow', '."', '--', 'r
ichardson', "'", 's', 'dictionari', 'keto', '.', 'greek', '.', 'cetu', ',', 'latin', '.', 'whoel', ',', 'anglo', '-', 'sa
om', 'allus', 'to', 'whale', 'he', 'could', 'anyway', 'find', 'in', 'ani', 'book', 'whatsoev', ',', 'sacr', 'or', 'profa
n', '.', 'therefor', 'you', 'must', 'not', ',', 'in', 'everi', 'case', 'at', 'least', ',', 'take', 'the', 'higgledi', '-
', 'piggledi', 'whale', 'statement', ',', 'howev', 'authent', ',', 'in', 'these', 'extract', ',', 'for', 'verit', 'gospe
l', 'cetolog', '.', 'far', 'from', 'it', '.', 'as', 'touch', 'the', 'ancient', 'author', 'gener', '.', 'as', 'well', 'a
```

9. List at least 5 differences you see in the stems verses the lemmas:

1. stem converts words to lower case-lemma does not
2. stem took off y from etymology-lemma kept the ending y
3. stem took off ed from supplied-lemma kept the ending ed
4. stem took off e from threadbare-lemma does not
5. stem took off s in his-lemma kept the word as his

- Stemming seems to take off last one or two characters from words, whereas lemmas will keep the end. But not every word is altered to a more base form. Sometimes in stemming the y will be turned into an i. Also, the lemma words make more sense than stemmed words.

```
from nltk.stem import WordNetLemmatizer
wnl = WordNetLemmatizer()
nltk.download('wordnet')
lemmatized = [wnl.lemmatize(t) for t in text1]
print(lemmatized)
```

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

```
['[', 'Moby', 'Dick', 'by', 'Herman', 'Melville', '1851', ']', 'ETYMOLOGY', '.', '(', 'Supplied', 'by', 'a', 'Late', 'Con
sumptive', 'Usher', 'to', 'a', 'Grammar', 'School', ')', 'The', 'pale', 'Usher', '--', 'threadbare', 'in', 'coat', ',',
'heart', ',', 'body', ',', 'and', 'brain', ';', 'I', 'see', 'him', 'now', '.', 'He', 'wa', 'ever', 'dusting', 'his', 'ol
d', 'lexicon', 'and', 'grammar', ',', 'with', 'a', 'queer', 'handkerchief', ',', 'mockingly', 'embellished', 'with', 'al
l', 'the', 'gay', 'flag', 'of', 'all', 'the', 'known', 'nation', 'of', 'the', 'world', '.', 'He', 'loved', 'to', 'dust',
'his', 'old', 'grammar', ';', 'it', 'somehow', 'mildly', 'reminded', 'him', 'of', 'his', 'mortality', '.', '"', 'While',
'you', 'take', 'in', 'hand', 'to', 'school', 'others', ',', 'and', 'to', 'teach', 'them', 'by', 'what', 'name', 'a', 'wha
le', '-', 'fish', 'is', 'to', 'be', 'called', 'in', 'our', 'tongue', 'leaving', 'out', ',', 'through', 'ignorance', ',',
'the', 'letter', 'H', ',', 'which', 'almost', 'alone', 'maketh', 'the', 'signification', 'of', 'the', 'word', ',', 'you',
'deliver', 'that', 'which', 'is', 'not', 'true', '."', '--', 'HACKLUYT', '"', 'WHALE', '.', '...', 'Sw', '.', 'and', 'Da
n', '.', 'HVAL', '.', 'This', 'animal', 'is', 'named', 'from', 'roundness', 'or', 'rolling', ';', 'for', 'in', 'Dan',
'.', 'HVALT', 'is', 'arched', 'or', 'vaulted', '."', '--', 'WEBSTER', "'", 'S', 'DICTIONARY', '"', 'WHALE', '.', '...',
'It', 'is', 'more', 'immediately', 'from', 'the', 'Dut', '.', 'and', 'Ger', '.', 'WALLEN', ';', 'A', '.', 'S', '.', 'WAL
W', '-', 'IAN', ',', 'to', 'roll', ',', 'to', 'wallow', '."', '--', 'RICHARDSON', "'", 'S', 'DICTIONARY', 'KETOS', ',',
'GREEK', '.', 'CETUS', ',', 'LATIN', '.', 'WHOEL', ',', 'ANGLO', '-', 'SAXON', '.', 'HVALT', ',', 'DANISH', '.', 'WAL',
' ', 'DUTCH', ' ', 'HWAL', ' ', 'SWEDISH', ' ', 'WHALE', ' ', 'ICELANDIC', ' ', 'WHALE', ' ', 'ENGLISH', ' ', 'BALEINE'
```

Comments:

a. your opinion of the functionality of the NLTK library

I don't have much experience using python or libraries like nltk, but from this assignment i think it functions very intuitively. It didn't take long for someone as inexperienced as myself to be able to figure out and use its methods to display information. Once small observation though was that when stemming and lemmatizing text i expected more words to be truncated and bigger differences between the two operations.

b. your opinion of the code quality of the NLTK library

I did find it somewhat easy to read and follow along, there couldve been more/better comments but thats due to my inexperience in this area. I did have some issue when trying to process the raw text of the different books, but that may have been due to my machine as opposed to the code.

c. a list of ways you may use NLTK in future projects

I do like the premise behind stemming and lemmatizing, as well as all the builtin methods to remove certain characters to process text. Like strip(), split() etc.