```
import nltk
nltk.download("stopwords")
nltk.download("wordnet")
nltk.download("punkt")

nltk.download("omw-1.4")
nltk.download('gutenberg')
nltk.download('genesis')
nltk.download('inaugural')
nltk.download('nps_chat')
nltk.download('webtext')
nltk.download('treebank')
from nltk.book import *
#I learned that the tokens method splits the text into a token list and that a text object ca
#code prints first 20 tokens in text1
for i in range (0,20):
       print(text1.tokens[i])

#prints a concordance list for the word sea with 5 lines
print  (text1.concordance_list("sea", 79, 5))
text2 = "CS 4395"

#Count() in nltk counts the number of times a word appears in the text. It works by counting
#appears in the token list. Python’s count method can also count how many times a word occurs
print(text2.count('C'))
print(text1.tokens.count("Moby"))

#Source : QUiz 2
raw_text = 'Voldemort himself created his worst enemy, just as tyrants everywhere do! Have yo

#code imports word tokenize, tokenize the text into tokens, and prints the first 10 tokens
from nltk import word_tokenize
token = word_tokenize(raw_text)
for i in range (0, 10):
  print(token[i])

#code imports word tokenize, tokenize the text into sentences, and prints the sentence list
from nltk import sent_tokenize
token = sent_tokenize(raw_text)
print(token)

#code imports PorterStemmer, tokenizes the text, and implements a stemmer to every token
from nltk import PorterStemmer
token = word_tokenize(raw_text)
stemmer = PorterStemmer()
stemmed = [stemmer.stem(t) for t in token]
print(stemmed)
```

```
#The stems have certain afffixes removed, like ‘ed’. The lemmas are the roots of each word.
#Some of the stemmed words lose their meaning after the affix is removed, but the lemmas are
#The list of lemmas is longer than the list of stems.
#The lemmas have some words with capital letters while the stems don’t.
#All the lemma words look meaningful whereas some of the stems are not actual words and thus

#code imports WordNetLemmatizer, tokenizes the text, and implements a lemmatizer to every tok
from nltk import WordNetLemmatizer
wnl = WordNetLemmatizer()
lemmatized = [wnl.lemmatize(t) for t in token]
print(lemmatized)
```

```
[
Moby
Dick
by
Herman
Melville
1851
]
ETYMOLOGY
.
(
Supplied
by
a
Late
Consumptive
Usher
to
a
Grammar
[ConcordanceLine(left=['piercing', 'serpent', ',', 'even', 'Leviathan', 'that', 'crooked
1
84
Voldemort
himself
created
his
worst
enemy
,
just
as
tyrants
['Voldemort himself created his worst enemy, just as tyrants everywhere do!', 'Have you
['voldemort', 'himself', 'creat', 'hi', 'worst', 'enemi', ',', 'just', 'as', 'tyrant',
['Voldemort', 'himself', 'created', 'his', 'worst', 'enemy', ',', 'just', 'a', 'tyrant',
[nltk_data] Downloading package stopwords to /root/nltk_data...
```

```
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package genesis to /root/nltk_data...
[nltk_data]   Package genesis is already up-to-date!
[nltk_data] Downloading package inaugural to /root/nltk_data...
[nltk_data]   Package inaugural is already up-to-date!
[nltk_data] Downloading package nps_chat to /root/nltk_data...
[nltk_data]   Package nps_chat is already up-to-date!
[nltk_data] Downloading package webtext to /root/nltk_data...
[nltk_data]   Package webtext is already up-to-date!
[nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data]   Package treebank is already up-to-date!
```

Output:

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

4. [ConcordanceLine(left=['piercing', 'serpent', ',', 'even', 'Leviathan', 'that', 'crooked', 'serpent', ';', 'and', 'he', 'shall', 'slay', 'the', 'dragon', 'that', 'is', 'in', 'the'], query='sea', right=['."', '--', 'ISAIAH', '"', 'And', 'what', 'thing', 'soever', 'besides', 'cometh', 'within', 'the', 'chaos', 'of', 'this', 'monster', '"', 's'], offset=759, left_print=' shall slay the dragon that is in the', right_print='." -- ISAIAH " And what thing soever ', line=' shall slay the dragon that is in the sea ." -- ISAIAH " And what thing soever '), ConcordanceLine(left=['the', 'bottomless', 'gulf', 'of', 'his', 'paunch', '."', '--', 'HOLLAND', '"', 'S', 'PLUTARCH', '"', 'S', 'MORALS', '.', '"', 'The', 'Indian'], query='Sea', right=['breedeth', 'the', 'most', 'and', 'the', 'biggest', 'fishes', 'that', 'are', ':', 'among', 'which', 'the', 'Whales', 'and', 'Whirlpooles', 'called', 'Balaene'], offset=823, left_print=' S PLUTARCH ' S MORALS . " The Indian', right_print='breedeth the most and the biggest fis', line=' S PLUTARCH ' S MORALS . " The Indian Sea breedeth the most and the biggest fis'), ConcordanceLine(left=['arpens', 'of', 'land', '."', '--', 'HOLLAND', '"', 'S', 'PLINY', '.', '"', 'Scarcely', 'had', 'we', 'proceeded', 'two', 'days', 'on', 'the'], query='sea', right=[',', 'when', 'about', 'sunrise', 'a', 'great', 'many', 'Whales', 'and', 'other', 'monsters', 'of', 'the', 'sea', ',', 'appeared', '.', 'Among'], offset=872, left_print='cely had we proceeded two days on the', right_print=', when about sunrise a great many Wha', line='cely had we proceeded two days on the sea , when about sunrise a great many Wha'), ConcordanceLine(left=['proceeded', 'two', 'days', 'on', 'the', 'sea', ',', 'when', 'about', 'sunrise', 'a', 'great', 'many', 'Whales', 'and', 'other', 'monsters', 'of', 'the'], query='sea', right=[',', 'appeared', '.', 'Among', 'the', 'former', ',', 'one', 'was', 'of', 'a', 'most', 'monstrous', 'size', '.', '...', 'This', 'came'], offset=886, left_print='many Whales and other

monsters of the', right_print=', appeared . Among the former , one w', line='many Whales and other monsters of the sea , appeared . Among the former , one w'), ConcordanceLine(left=['This', 'came', 'towards', 'us', ',', 'open', '-', 'mouthed', ',', 'raising', 'the', 'waves', 'on', 'all', 'sides', ',', 'and', 'beating', 'the'], query='sea', right=['before', 'him', 'into', 'a', 'foam', '."', '--', 'TOOKE', '"', 'S', 'LUCIAN', '.', "", 'THE', 'TRUE', 'HISTORY', '."', ""], offset=922, left_print=' waves on all sides , and beating the', right_print='before him into a foam ." -- TOOKE ' ', line=' waves on all sides , and beating the sea before him into a foam ." -- TOOKE ' ')]

5. 1
6. 84
7. Voldemort himself created his worst enemy , just as tyrants
8. ['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!'] ['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', 'is', 'sure', 'to', 'be', 'one', 'who', 'rise', 'against', 'them', 'and', 'strike', 'back', '!']
9. ['Voldemort', 'himself', 'created', 'his', 'worst', 'enemy', ',', 'just', 'a', 'tyrant', 'everywhere', 'do', '!', 'Have', 'you', 'any', 'idea', 'how', 'much', 'tyrant', 'fear', 'the',

My opinion is that the NLTK library is very useful. It has functions to stem text, lemmatize text, tokenize text and much more. These functions can be used in small and larger-scale NLTK projects. I think that the code quality in the NLTK is library is good. I think that they could give a few more examples outlining what each function does, but the documentation provided is enough to understand each function. I would use NLTK to lemmatize text in future projects and also use functions such as work tokenize and sentence tokenize to analyze words and sentences and apply learning concepts to them.

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-5e1b015e4046> in <module>
----> 1 print  (text1.concordance_list("sea", 79, 5))

NameError: name 'text1' is not defined
```

SEARCH STACK OVERFLOW