```
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')
nltk.download('book')
from nltk.book import text1
from nltk.tokenize import word_tokenize
from nltk.text import Text
from nltk.tokenize import sent_tokenize
from nltk.stem import *
from nltk.stem.porter import *
from nltk.stem import WordNetLemmatizer
```

```
[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 collection 'book'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Package abc is already up-to-date!
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Package brown is already up-to-date!
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Package chat80 is already up-to-date!
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Package conll2000 is already up-to-date!
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Package conll2002 is already up-to-date!
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package dependency_treebank 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 gutenberg to /root/nltk_data...
[nltk_data]    |   Package gutenberg is already up-to-date!
[nltk_data]    | Downloading package ieer to /root/nltk_data...
[nltk_data]    |   Package ieer 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 movie_reviews to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package nps_chat to /root/nltk_data...
```

✓ 0s completed at 7:11 PM

```
[nltk_data]    |   Package names is already up-to-date!
[nltk_data]    | Downloading package ppattach to /root/nltk_data...
[nltk_data]    |   Package ppattach is already up-to-date!
[nltk_data]    | Downloading package reuters to /root/nltk_data...
[nltk_data]    |   Package reuters is already up-to-date!
[nltk_data]    | Downloading package senseval to /root/nltk_data...
[nltk_data]    |   Package senseval is already up-to-date!
[nltk_data]    | Downloading package state_union to /root/nltk_data...
[nltk_data]    |   Package state_union is already up-to-date!
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package swadesh to /root/nltk_data...
[nltk_data]    |   Package swadesh is already up-to-date!
[nltk_data]    | Downloading package timit to /root/nltk_data...
[nltk_data]    |   Package timit is already up-to-date!
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package toolbox to /root/nltk_data...
[nltk_data]    |   Package toolbox is already up-to-date!
```

3.) Extract first 20 tokens from text1.

Code should output the first 20 tokens from text1.

Two things I learned about text objects are that the tokens() method can either return all tokens from a Text object or a specific number of tokens given an integer parameter, and that tokens() can store words, punctuation, and even sentences.

```
text1.tokens[:20]
```

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

```
 'Grammar']
```

4.) Concordance for the 'sea' in text1 for only 5 lines

Code should print the first 5 lines in the text with the word 'sea'.

```
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 '
```

5.) The count() method works by comparing the string passed as a parameter against every token in the Text object, and returning the number of tokens that match the given string exactly. The built in count() Python method is used with either Strings or Lists; for Strings, it counts the amount of times the subtring passed in is found, for Lists it counts how many of the String passed in appears in the List.

6.) Using raw text of at least 5 sentences of your choice from any source (cite the source), save the text into a variable called raw_text. Using NLTK's word tokenizer, tokenize the text into variable 'tokens'. Print the first 10 tokens.

Text from "The Neverending Story" by Michael Ende.

Code should output the first 10 tokens in raw_text.

```
raw_text = "Once someone dreams a dream, it can't just drop out of existence. But if the dr
rt_tokens = Text(word_tokenize(raw_text))
rt_tokens.tokens[:10]
```

```
['Once', 'someone', 'dreams', 'a', 'dream', ',', 'it', 'ca', "n't", 'just']
```

7.) Using the same raw text, and NLTK's sentence tokenizer sent_tokenize(), perform sentence segmentation and display the sentences.

Code should output each sentence in the raw_text.

```
rt_sent = Text(sent_tokenize(raw_text))
rt_sent.tokens
```

```
["Once someone dreams a dream, it can't just drop out of existence.",
 "But if the dreamer can't remember it, what becomes of it?",
 'It lives on in Fantastica, deep under earth.',
 'There are forgotten dreams stored in many layers.',
 'The deeper one digs, the closer they are.',
 'All Fantastica rests on a foundation of forgotten dreams.']
```

8.) Using NLTK's PorterStemmer(), write a list comprehension to stem the text. Display the list.

Code should output the stemmed version of all tokens in raw_text.

```
stemmer = PorterStemmer()
[stemmer.stem(t) for t in rt_tokens.tokens]
```

```
['onc',
 'someon',
 'dream',
 'a',
 'dream',
 ',',
 'it',
 'ca',
 "n't",
 'just',
```

```
 'drop',
 'out',
 'of',
 'exist',
 '.',
 'but',
 'if',
 'the',
 'dreamer',
 'ca',
 "n't",
 'rememb',
 'it',
 ',',
 'what',
 'becom',
 'of',
 'it',
 '?',
 'it',
 'live',
 'on',
 'in',
 'fantastica',
 ',',
 'deep',
 'under',
 'earth',
 '.',
 'there',
 'are',
 'forgotten',
 'dream',
 'store',
 'in',
 'mani',
 'layer',
 '.',
 'the',
 'deeper',
 'one',
 'dig',
 ',',
 'the',
 'closer',
 'they',
 'are',
 '.',
```

9.) Using NLTK's WordNetLemmatizer, write a list comprehension to lemmatize the text. Display the list. In

the text cell above this code cell, list at least 5 differences you see in the stems verses the lemmas. You can just write them each on a line, like this: stem-lemma

Code should print the lemmatized version of all tokens in raw_text.

Stem-Lemma:

onc-Once, someon-someone, exist-existence, becom-becomes, live-life

```
lemmatizer = WordNetLemmatizer()
[lemmatizer.lemmatize(t) for t in rt_tokens.tokens]
```

```
['Once',
 'someone',
 'dream',
 'a',
 'dream',
 ',',
 'it',
 'ca',
 "n't",
 'just',
 'drop',
 'out',
 'of',
 'existence',
 '.',
 'But',
 'if',
 'the',
 'dreamer',
 'ca',
 "n't",
 'remember',
 'it',
 ',',
 'what',
 'becomes',
 'of',
 'it',
 '?',
 'It',
 'life',
 'on',
 'in',
 'Fantastica',
 ',',
 'deep',
 'under'
```

```
 under',
'earth',
'.',
'There',
'are',
'forgotten',
'dream',
'stored',
'in',
'many',
'layer',
'.',
'The',
'deeper',
'one',
'dig',
',',
'the',
'closer',
'they',
'are',
'.',
```

10.) Comment cell: Write a paragraph outlining:

1. your opinion of the functionality of the NLTK library
2. your opinion of the code quality of the NLTK library
3. a list of ways you may use NLTK in future projects

> I think the NLTK library functionality is extremely useful for NLP and processing tokens and sentences. It's also useful for finding a concordance of a text (searching for specific tokens) to see how often a word appears in a text. The code quality of the NLTK library is also fairly high seeing as it's straightforward using the methods for each class and accessing the data values for each class is simple as well. As for my use of NLTK in future projects, I'm actually going to be using NLTK along with NLP concepts in my Senior Design Project this semester (Fall 2022). I hope to be able to apply what I learn to my project and for other future projects revolving around the human-computer interaction.

Colab paid products - Cancel contracts here