```
# Aditya Guin
# CS 4395.001
# Portfolio Assignment 2

# 1, 2
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 *
from nltk import word_tokenize, sent_tokenize, PorterStemmer, 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 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!
```

List two things you learned about the tokens() method or Text objects in the text cell above this code cell.

- Text objects have two attributes; a name and tokens list.
- Text objects are used in the TextCollections class
- Tokens method splits a text into tokens and stores it in a text tokens object.

```
# 3. Extract the first 20 tokens from text1.
for i, token in enumerate(text1.tokens):
  if i >= 20:
    break
  print(token)
```

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

```
# 4. Look at the concordance() method in the API. Using the documentation to guide you, in co
# print a concordance for text1 word 'sea', selecting only 5 lines.

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. NLTK Count method vs Python Count method

For a given text, the count() method returns the number of occurrences of a word within that text. It is similar to pythons count method in that it returns the count of a phrase in a sentence. However, it is slightly different compared python's count method since the API returns the count from the tokens. If the nltk count method were used for the phrase "world.", or any word appended with any punctuation, it would always return 0. However pythons count function could return a positive number. This is because the tokenization in nltk splits text into tokens based of punctuation as well, whereas python's inbuilt count function doesn't do this.

```
# 5. Experimenting with NLTK count method vs Python count method

raw_text = "This is a ball. The ball says hi."

nltk_tokens = word_tokenize(raw_text)

# Same outputs for says
print(f'Python count for "says": {raw_text.count("says")}')
print(f'NLTK count for "says": {nltk_tokens.count("says")}')

print()
# Different outputs for ball.
print(f'Python count for "ball.": {raw_text.count("ball.")}')
print(f'NLTK count for "ball.": {nltk_tokens.count("ball.")}')
```

```
Python count for "says": 1
NLTK count for "says": 1

Python count for "ball.": 1
NLTK count for "ball.": 0
```

```
# 6. Using 5 sentences from Hunger Games (opening sentences)
# LINK: https://docs.google.com/viewer?a=v&pid=sites&srcid=c21jc3R1ZGVudHMuY2F8bXItbGFsb25kZS

raw_text = '''
When I wake up, the other side of the bed is cold. My fingers stretch out, seeking Prim’s war
'''

# Word tokenize raw_text and printing first 10 tokens
for i, token in enumerate(word_tokenize(raw_text)):
  if i >= 10:
    break
  print(token)
```

```
When
I
wake
up
,
the
other
side
of
the
```

```
# 7. Sentence tokenize raw_text using NLTK's sent_tokenize and printing the sentence
sentence_tokens = sent_tokenize(raw_text)
for sentence in sentence_tokens:
```

```
print(sentence)
```

```
When I wake up, the other side of the bed is cold.
My fingers stretch out, seeking Prim’s warmth but finding only the rough canvas cover of
She must have had bad dreams and climbed in with our mother.
Of course, she did.
This is the day of the reaping.
```

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

stemmer = PorterStemmer()
stemmed = [stemmer.stem(token) for token in word_tokenize(raw_text)]

print(stemmed)
```

```
['when', 'i', 'wake', 'up', ',', 'the', 'other', 'side', 'of', 'the', 'bed', 'is', 'colc
```

```
# 9. Using NLTK's WordNetLemmatizer, write a list comprehension to lemmatize the text. Displa

lemma = WordNetLemmatizer()
lemmed = [lemma.lemmatize(token) for token in word_tokenize(raw_text)]
print(lemmed)

# Differences (stem-lemma)
# when-When
# i-I
# seek-seeking
# find-finding
# my-My
```

```
['When', 'I', 'wake', 'up', ',', 'the', 'other', 'side', 'of', 'the', 'bed', 'is', 'colc
```

10. Opinion on functionality, code quality and project applications of NLTK

I believe that NLTK is broad in terms of tasks it can do. For example, tokenize words, sentences, and also being able to lemmatize words are examples of the vast functions the library has. The code quality is very good, and class methods and variables are done following good coding practices. Some projects one can complete using nltk is creating a sentiment analyzer, and part of speech tagging.