Part 1

Create the Notebook: Done!

Part 2

```
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

Part 3

Purpose

Each of the built-in 9 texts is an NLTK Text object. Look at the code for the Text object at this link: https://www.nltk.org/_modules/nltk/text.html. Look at the tokens() method. Extract the first 20 tokens from text1. List two things you learned about the tokens() method or Text objects in the text cell above this code cell.

What I Learned

1. The tokens() method is merely a getter method, as the tokens attribute is set beforehand for the Text object.
2. The self.tokens attribute is set in the **init** method.

```
nltk.download('book')
from nltk.book import *
temp = text1.tokens[:20]
temp
```

```
[nltk_data]    |   Unzipping corpora/words.zip.
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
```

```
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/maxent_treebank_pos_tagger.zip.
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/universal_tagset.zip.
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/book_grammars.zip.
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/city_database.zip.
[nltk_data]    | Downloading package tagsets to /root/nltk_data...
[nltk_data]    |   Unzipping help/tagsets.zip.
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    |
[nltk_data]  Done downloading collection book
*** 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
['[',
 'Moby',
 'Dick',
 'by',
 'Herman',
 'Melville',
 '1851',
 ']',
 'ETYMOLOGY',
 '.',
 '(',
 'Supplied',
 'by',
 'a',
 'Late',
 'Consumptive',
 'Usher',
 'to',
        'a',
```

Part 4

Purpose

Look at the concordance() method in the API. Using the documentation to guide you, in code, 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 '
```

Part 5

Purpose

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? Write your commentary above the code cell. In the code cells, experiment with both count() methods.

The count() method in the NLTK API actually uses Python's built-in count method. In essence the API simplifies the method for counting. Without the class method, we would half to call Python's count method on the tokens attribute of an instance of the Text class. Instead, the class method helps hide the tokens attribute from the user.

```
print(text1.count('sea')) # Using the NLTK count() method
print(text1.tokens.count('sea')) # Using the Python built-in count method

     433
     433
```

Part 6

Purpose

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.

```
raw_text = """LONDON — Swiftly taking on the mantle of Britain’s monarch, King Charles III re

The king’s speech capped a day of mourning across Britain, but it was also a vivid demonstrat

“Queen Elizabeth was a life well lived, a promise with destiny kept,” Charles said in a telev

Recalling Elizabeth’s vow, on her 21st birthday, to serve her people for the remainder of her

# SOURCE FOR TEXT: https://www.nytimes.com/live/2022/09/09/world/queen-elizabeth-king-charles

from nltk.tokenize import word_tokenize
tokens = word_tokenize(raw_text)
print(tokens[:10])
```

```
['LONDON', '—', 'Swiftly', 'taking', 'on', 'the', 'mantle', 'of', 'Britain', '’']
```

Part 7

Purpose

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

```
from nltk.tokenize import sent_tokenize
tokens = sent_tokenize(raw_text)

for sent in tokens:
  print("'" + sent + "'")
```

```
'LONDON — Swiftly taking on the mantle of Britain’s monarch, King Charles III returned t

The king’s speech capped a day of mourning across Britain, but it was also a vivid demor
'He met with the new prime minister, Liz Truss, just four days after the queen anointed 
'“Queen Elizabeth was a life well lived, a promise with destiny kept,” Charles said in a
'Recalling Elizabeth’s vow, on her 21st birthday, to serve her people for the remainder 
```

Part 8

Purpose

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

```
from nltk.stem import *
stem = PorterStemmer()
tokens = word_tokenize(raw_text)
stemmed = [stem.stem(token) for token in tokens]
full = ' '.join(stemmed)
print(full)
```

```
london — swiftli take on the mantl of britain ’ s monarch , king charl iii return to lon
```

Part 9

Purpose

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

Differences I Found

1. Stem has no capital letters (normalized) - Lemma keeps capital letters.
2. Stem replaces all ‘y’ characters with ‘i’ (‘many’ becomes ‘mani’) - Lemma does not naively replace ‘y’ with ‘i’.
3. Stem leaves ‘as’ as it is – Lemma for some reason strips the ‘s’ from ‘as.’
4. Stem strips all trailing ‘e’ regardless of validity or not – Lemma is more intelligent and leaves the ‘e’ characters as last characters.
5. Stem strips ‘ed’ from ‘created’ - Lemma cannot lemmatize ‘created’ to ‘create’ properly.

```
from nltk.stem import WordNetLemmatizer
tokens = word_tokenize(raw_text)
lemmatizer = WordNetLemmatizer()
processed = [lemmatizer.lemmatize(token) for token in tokens]
full = ' '.join(processed)
print(full)
```

```
LONDON — Swiftly taking on the mantle of Britain ’ s monarch , King Charles III returne
```

Part 10

Purpose

Comment cell: Write a paragraph outlining:

a. your opinion of the functionality of the NLTK library

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

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

My Views on NLTK

I think that the functionality of the NLTK library is adequate for an educational setting but is probably not the best in all applications of NLP. For example it has some flaws in its stemming and lemmatizing that I might not be present in other libraries (though depending on the number of errors the flaws may be acceptable). However, I do think its tokenizer is fairly sophisticated and I would be comfortable using it in my own machine learning projects. My opinion of the code quality of the NLTK library is good. It seems intuitive to use, the code is clean and easy to read, and the documentation is concise. In the future I will definitely be using the tokenizer. Furthermore, I imagine that if I can load my own texts into the 'text' class, the concordance() and count() methods both seem very useful.