The code below imports nltk and downloads all the packages that we will be using henceforth.

```
import nltk
import re
from nltk.corpus import stopwords
import re
from nltk.book import *
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.text import *
from nltk.stem import PorterStemmer
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\pecan\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\pecan\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\pecan\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\pecan\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
True
```

I learned about how the tokens method tokenizes based on word and sentence. I learned how text objects have many methods wrapped within them that enable you to use them in novel and innovative ways.

```
print(text1[:20])
```

```
text1.concordance("sea",80,5)
```

This works by counting every instance of the specified token within the text and giving you the count. This works the same as pythons count method.

```
presidents = ['obama','obama','biden', 'trump']
print(presidents.count('obama'))

tokener = [t.lower() for t in text1]
```

```
setter = set(tokener)


wordLemmat = nltk.WordNetLemmatizer()

lemmatLister = [wordLemmat.lemmatize(i) for i in tokener]


lemmaSetter = set(lemmatLister)  # creating a new list that removes duplicates

removePunct = [",", "?", ".", ":", "'", "[", "]",
               "{", "}", "--", "#", "...", "!", "-", "'!", "...", ";", ","]
removeStop = stopwords.words("english") + ["wa"]

removedList = [
    storedvalue for storedvalue in lemmatLister if storedvalue not in removePunct+removeStop]

variable = {x: removedList.count(x) for x in lemmaSetter}

sorted_counts = sorted(variable.items(), key=lambda x: x[1], reverse=True)
print(sorted_counts)
```

These are the first 5 sentences of Moby Dick by Herman Melville

```
raw_text = "Call me Ishmael. Some years ago - never mind how long precisely having little or

tokens = word_tokenize(raw_text)
print(tokens[:10])
```

```
['Call', 'me', 'Ishmael', '.', 'Some', 'years', 'ago', '-', 'never', 'mind']
```

```
sent_tokens = sent_tokenize(raw_text)
print(sent_tokens)
```

```
['Call me Ishmael.', 'Some years ago - never mind how long precisely having little or no
```

```
portStem = PorterStemmer()
stem_list = [portStem.stem(x) for x in tokens]
print(stem_list)
```

```
['call', 'me', 'ishmael', '.', 'some', 'year', 'ago', '-', 'never', 'mind', 'how', 'long
```

precis-precisely hav-having littl-little purs-purse noth-nothing

```
netLemmatizer = nltk.WordNetLemmatizer()
lemmatizedList = [netLemmatizer.lemmatize(i) for i in tokens]
print(lemmatizedList)
```

```
['Call', 'me', 'Ishmael', '.', 'Some', 'year', 'ago', '-', 'never', 'mind', 'how', 'long
```

a. The NLTK library has quite extensive functionality that gives a researcher or developer the tools necessary to parse and understand text. These tools can be used to build tools that can perform many applications related to text comprehension, generation, etc. I think the code in the NLTK library is quite readable and effecient , it is suitable for building large applications. I may use the NLTK library to build text comprehension software that could summarize text without losing the inherent meaning within it. This would be quite useful for many applications such as news summarizing so that you don't have to spend as much time in the morning reading the news.