Name:- Ujwal Sahu
Roll No:- 34
Branch:- BVOC TY AIDS B
Subject:- Natural Language Processing

ASSIGNMENT 01

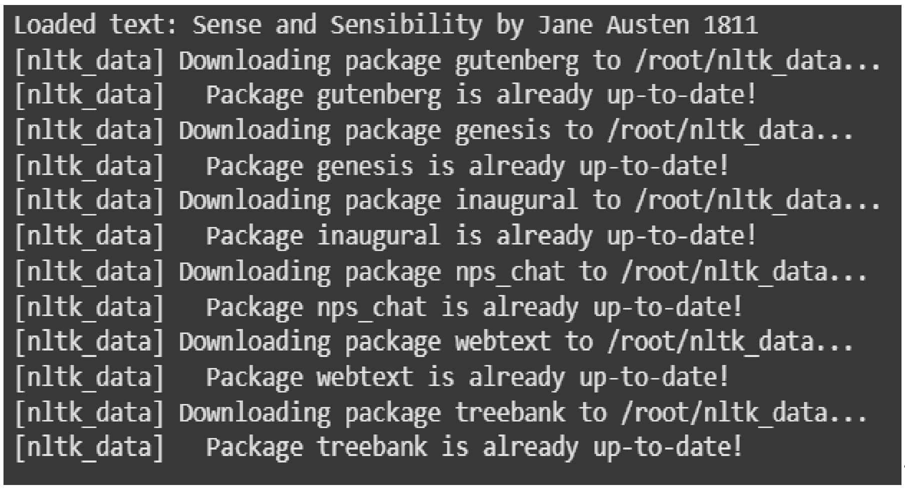
Title: Text Exploration

Theory: Text Exploration is the process of analysing a text to understand its content and structure. It includes tasks like concordance (finding a word in context), similar words, common contexts, and dispersion plots (visualizing word occurrence). The goal is to identify word usage, patterns, and themes. It is a key first step in Natural Language Processing for understanding the text before deeper analysis.

Code:

1. **Load text2 (Sense and Sensibility) from nltk.book.**

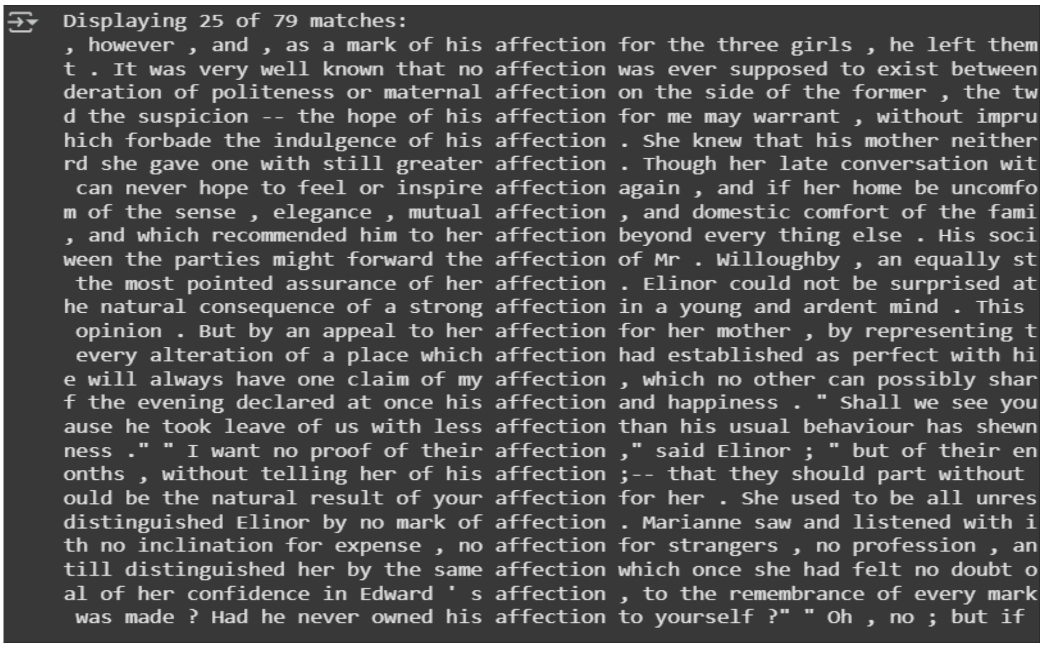
```
import nltk

nltk.download('gutenberg')

nltk.download('genesis')

nltk.download('inaugural')

nltk.download('nps_chat')

nltk.download('webtext')

nltk.download('treebank')

from nltk.book import *

print("Loaded text:", text2.name)
```



```
Loaded text: Sense and Sensibility by Jane Austen 1811
[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!
```
,

2. **Find the concordance for the word "affection".**

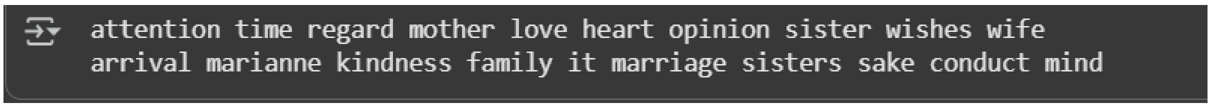
text2.concordance("affection")



```
Displaying 25 of 79 matches:
, however , and , as a mark of his affection for the three girls , he left them
t . It was very well known that no affection was ever supposed to exist between
deration of politeness or maternal affection on the side of the former , the tw
d the suspicion -- the hope of his affection for me may warrant , without impru
hich forbade the indulgence of his affection . She knew that his mother neither
rd she gave one with still greater affection . Though her late conversation wit
 can never hope to feel or inspire affection again , and if her home be uncomfo
m of the sense , elegance , mutual affection , and domestic comfort of the fami
, and which recommended him to her affection beyond every thing else . His soci
ween the parties might forward the affection of Mr . Willoughby , an equally st
 the most pointed assurance of her affection . Elinor could not be surprised at
he natural consequence of a strong affection in a young and ardent mind . This
 opinion . But by an appeal to her affection for her mother , by representing t
 every alteration of a place which affection had established as perfect with hi
e will always have one claim of my affection , which no other can possibly shar
f the evening declared at once his affection and happiness . " Shall we see you
ause he took leave of us with less affection than his usual behaviour has shewn
ness ." " I want no proof of their affection ," said Elinor ; " but of their en
onths , without telling her of his affection ;-- that they should part without
ould be the natural result of your affection for her . She used to be all unres
distinguished Elinor by no mark of affection . Marianne saw and listened with i
th no inclination for expense , no affection for strangers , no profession , an
till distinguished her by the same affection which once she had felt no doubt o
al of her confidence in Edward ' s affection , to the remembrance of every mark
 was made ? Had he never owned his affection to yourself ?" " Oh , no ; but if
```

3. **Find words similar to "affection" in text2.**

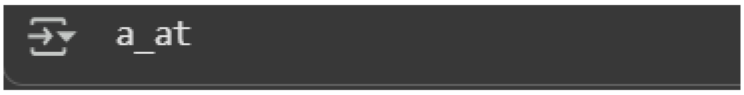
text2.similar("affection")



```
attention time regard mother love heart opinion sister wishes wife
arrival marianne kindness family it marriage sisters sake conduct mind
```

4. **Find the common_contexts for the words "lady" and "gentleman" in text2.**

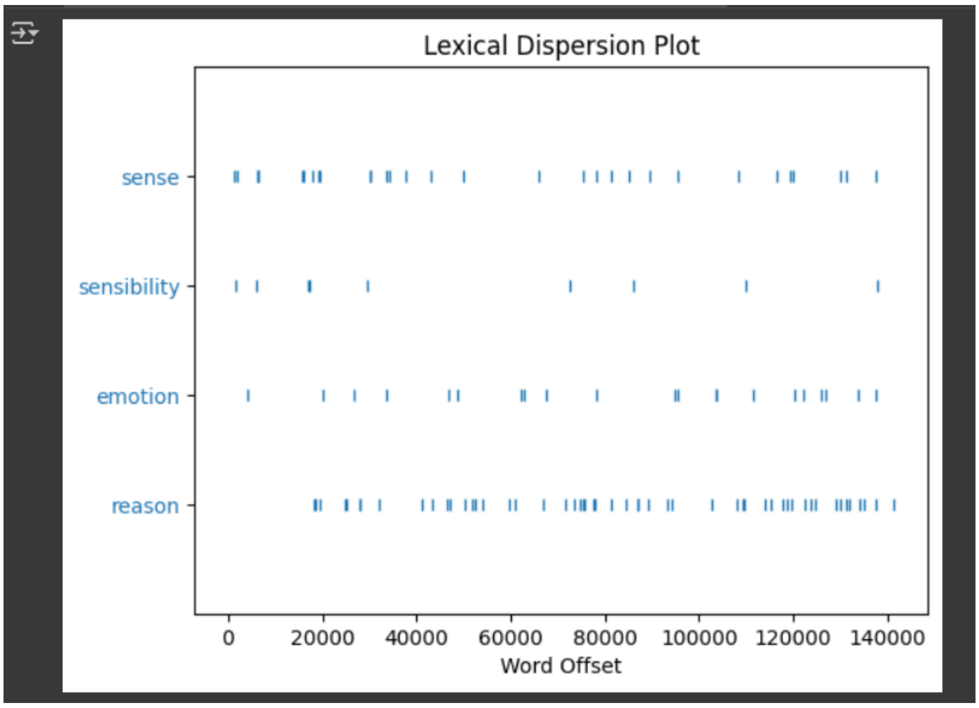
text2.common_contexts(["lady", "gentleman"])



```
a_at
```

5. **Generate a dispersion_plot for the words "sense", "sensibility", "emotion", and "reason" in text2.**

text2.dispersion_plot(["sense", "sensibility", "emotion", "reason"])



Conclusion: Text exploration provides a clear understanding of how words are used in a text. By analyzing concordances, similar words, common contexts, and dispersion plots, we can identify patterns, themes, and relationships between words. These insights help in understanding the style and structure of the text, forming a strong foundation for deeper NLP analysis.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

Name:- Ujwal Sahu
Roll No:- 34
Branch:- BVOC TY AIDS B
Subject:- Natural Language Processing

ASSIGNMENT 02

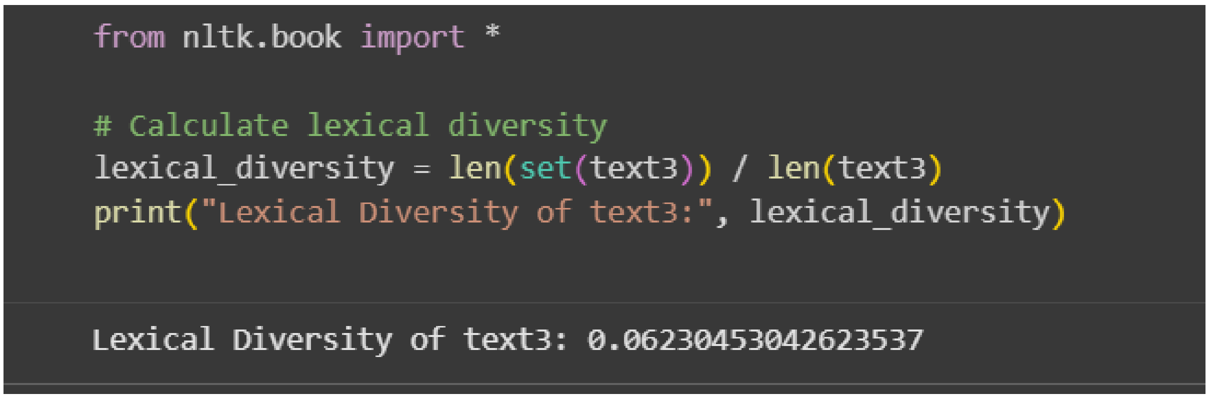
Title: Lexical Analysis

Theory: Lexical Analysis is the process of examining and analyzing the words (lexical items) that make up a text.
In NLP, it means studying the vocabulary, frequency, and diversity of words in a corpus.

Code:

1. **Calculate the lexical diversity of text3 (The Book of Genesis).**

```
from nltk.book import *

lexical_diversity = len(set(text3)) / len(text3)

print("Lexical Diversity of text3:", lexical_diversity)
```

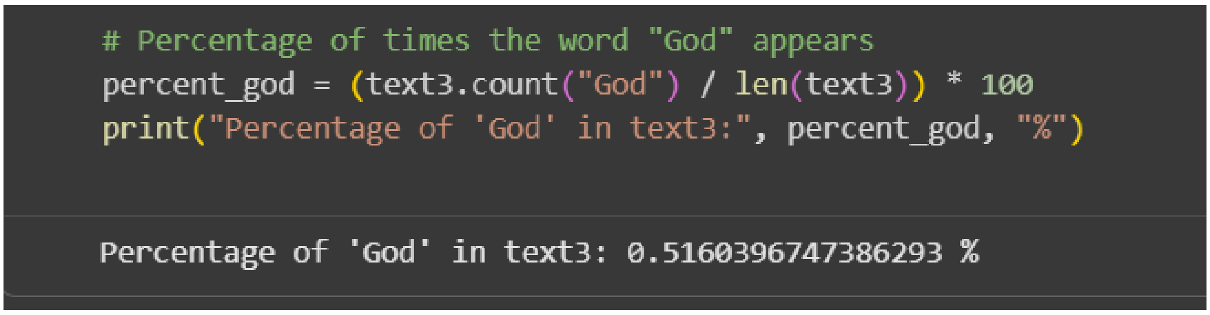


```
from nltk.book import *

# Calculate lexical diversity
lexical_diversity = len(set(text3)) / len(text3)
print("Lexical Diversity of text3:", lexical_diversity)
```

```
Lexical Diversity of text3: 0.06230453042623537
```

2. **Calculate the percentage of times the word "God" appears in text3.**

```
percent_god = (text3.count("God") / len(text3)) * 100

print("Percentage of 'God' in text3:", percent_god, "%")
```



```
# Percentage of times the word "God" appears
percent_god = (text3.count("God") / len(text3)) * 100
print("Percentage of 'God' in text3:", percent_god, "%")
```

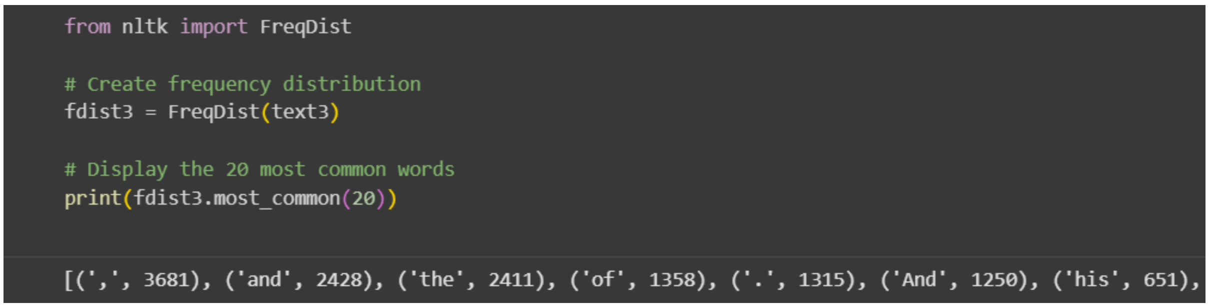
```
Percentage of 'God' in text3: 0.5160396747386293 %
```

3. **Find the 20 most common words in text3.**

from nltk import FreqDist

fdist3 = FreqDist(text3)

print(fdist3.most_common(20))



```
from nltk import FreqDist

# Create frequency distribution
fdist3 = FreqDist(text3)

# Display the 20 most common words
print(fdist3.most_common(20))
```

```
[(',', 3681), ('and', 2428), ('the', 2411), ('of', 1358), ('.', 1315), ('And', 1250), ('his', 651),
```

Conclusion: Lexical analysis helps in understanding the vocabulary richness and word usage in a text. By calculating lexical diversity, word frequencies, and percentages, we can identify common terms and assess how words like "God" are emphasized in *The Book of Genesis*. This analysis provides insights into the text's style, structure, and thematic focus

For Faculty Use

Correction Parameters	Formative Assessmen t [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

Name:- Ujwal Sahu
Roll No:- 34
Branch:- BVOC TY AIDS B
Subject:- Natural Language Processing

ASSIGNMENT 03

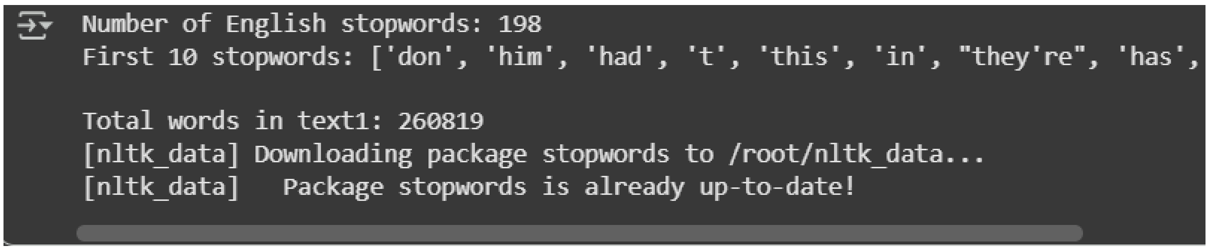
Title: Working with Stopwords

Theory: In Natural Language Processing (NLP), stopwords are common words in a language that usually carry less meaning for text analysis or model training.

Code:

1. **Load the English stopwords from nltk.corpus.stopwords.**

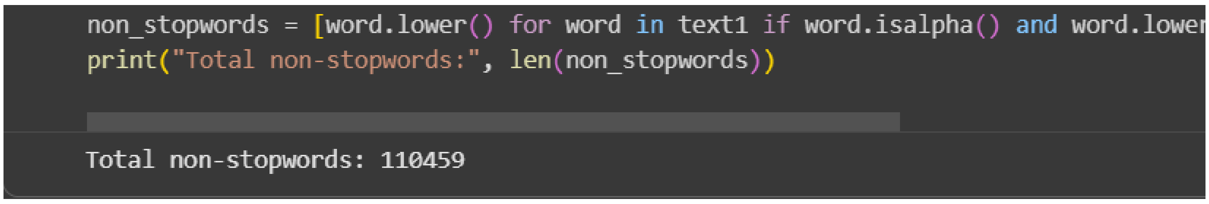
```
import nltk

from nltk.corpus import stopwords

from nltk.book import text1

nltk.download('stopwords')

stop_words = set(stopwords.words('english'))

print("Number of English stopwords:", len(stop_words))

print("First 10 stopwords:", list(stop_words)[:10])
```



```
Number of English stopwords: 198
First 10 stopwords: ['don', 'him', 'had', 't', 'this', 'in', "they're", 'has',

Total words in text1: 260819
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

2. **Create a list of words from text1 (Moby Dick) that are not stopwords and are purely alphabetic.**

```
non_stopwords = [word.lower() for word in text1 if word.isalpha() and word.lower() not in stop_words]

print("Total non-stopwords:", len(non_stopwords))
```

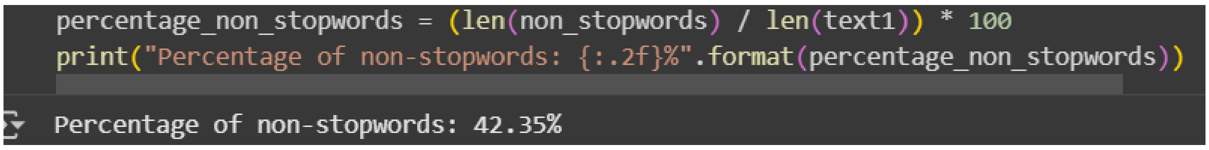


```
non_stopwords = [word.lower() for word in text1 if word.isalpha() and word.lowe
print("Total non-stopwords:", len(non_stopwords))

Total non-stopwords: 110459
```

3. Calculate the percentage of non-stopwords in text1.

percentage_non_stopwords = (len(non_stopwords) / len(text1)) * 100

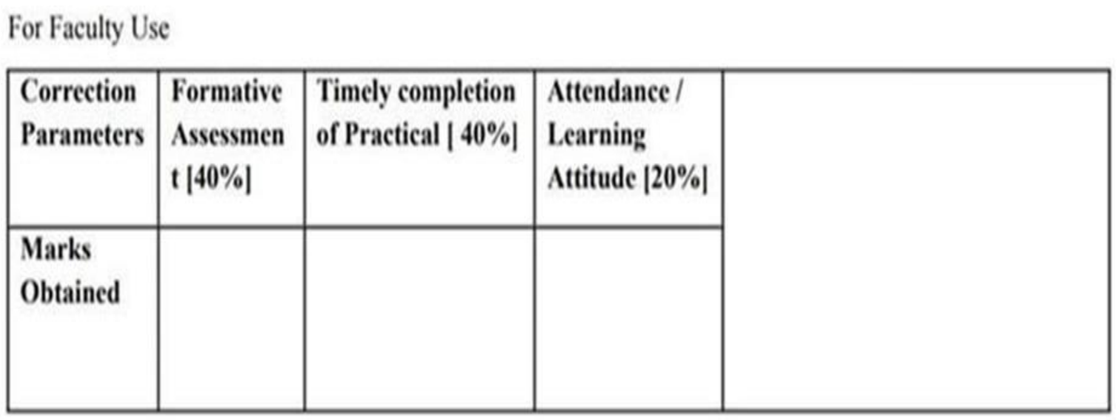
print("Percentage of non-stopwords: {:.2f}%".format(percentage_non_stopwords))



```
percentage_non_stopwords = (len(non_stopwords) / len(text1)) * 100
print("Percentage of non-stopwords: {:.2f}%".format(percentage_non_stopwords))
```

```
Percentage of non-stopwords: 42.35%
```

Conclusion: Removing stopwords helps focus on the **meaningful words** in a text. By filtering out common words and analyzing the remaining alphabetic tokens, we gain clearer insights into the important vocabulary and thematic content of the text.



For Faculty Use

Correction Parameters	Formative Assessmen t [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

Name:- Ujwal Sahu
Roll No:- 34
Branch:- BVOC TY AIDS B
Subject:- Natural Language Processing

ASSIGNMENT 04

Title: Stemming

Theory: Stemming is the process of reducing words to their base or root form (called a *stem*). It helps group different forms of a word so they can be treated as the same during text analysis.

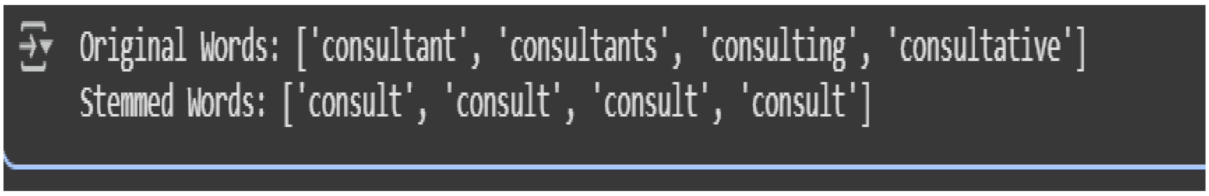
Code:

1. **Import a stemmer (e.g., PorterStemmer) from nltk.stem.**

```
from nltk.stem import PorterStemmer

from nltk.book import text1

from nltk.corpus import stopwords
```

2. **Stem a list of words (e.g., "consultant", "consultants", "consulting", "consultative").**

```
ps = PorterStemmer()

example_words = ["consultant", "consultants", "consulting", "consultative"]

print("Original Words:", example_words)

print("Stemmed Words:", [ps.stem(word) for word in example_words])
```

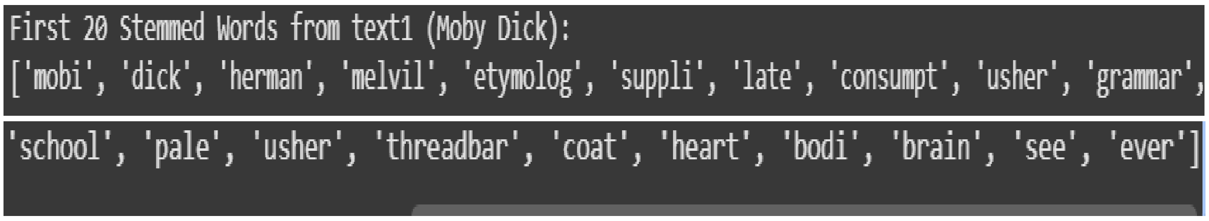


```
Original Words: ['consultant', 'consultants', 'consulting', 'consultative']
Stemmed Words: ['consult', 'consult', 'consult', 'consult']
```

3. **Apply the stemmer to the non-stopwords list created in Assignment 5 and display the first 20 stemmed words.**

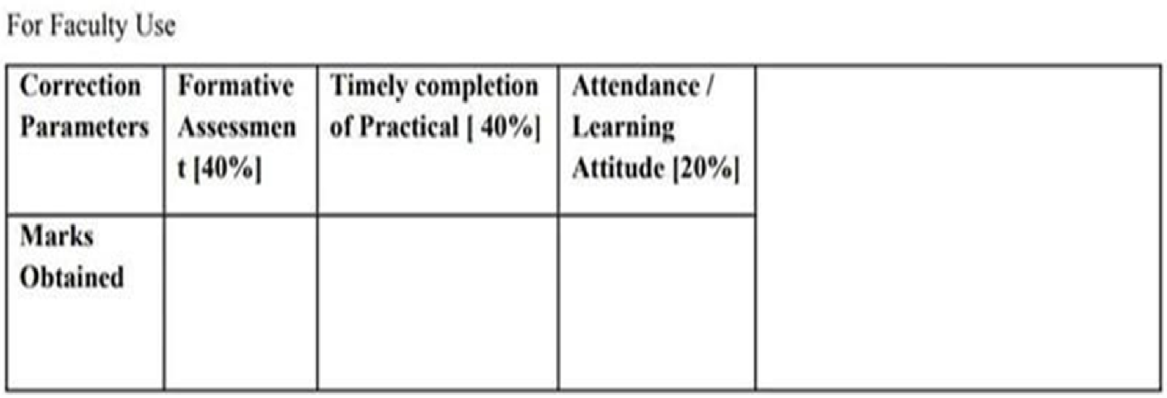
```
stop_words = set(stopwords.words('english'))

non_stopwords = [word.lower() for word in text1 if word.isalpha() and word.lower() not
in stop_words]

stemmed_words = [ps.stem(word) for word in non_stopwords]

print("\nFirst 20 Stemmed Words from text1 (Moby Dick):")
```

print(stemmed_words[:20])



```
First 20 Stemmed Words from text1 (Moby Dick):
['mobi', 'dick', 'herman', 'melvil', 'etymolog', 'suppli', 'late', 'consumpt', 'usher', 'grammar',
'school', 'pale', 'usher', 'threadbar', 'coat', 'heart', 'bodi', 'brain', 'see', 'ever']
```

Conclusion: Stemming reduces words to their **root forms**, allowing us to group similar words like “consultant” and “consulting.” Applying stemming to a text simplifies analysis, reduces redundancy, and improves efficiency in NLP tasks like search, indexing, and text classification.



For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

Name:- Ujwal Sahu
Roll No:- 34
Branch:- BVOC TY AIDS B
Subject:- Natural Language Processing

ASSIGNMENT 05

Title: Parts of speech tagging

Theory: Parts of Speech (POS) Tagging is the process of labeling each word in a sentence with its grammatical category —like noun, verb, adjective, adverb, etc.

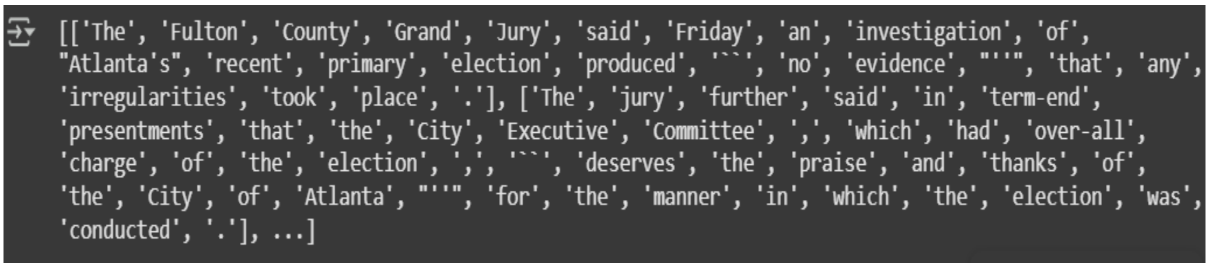
Code:

1. **Import the pos_tag function from nltk.**

```
import nltk

from nltk import pos_tag
```

2. **Import a corpus for tagging (e.g., brown corpus, specifically the 'news' category).**

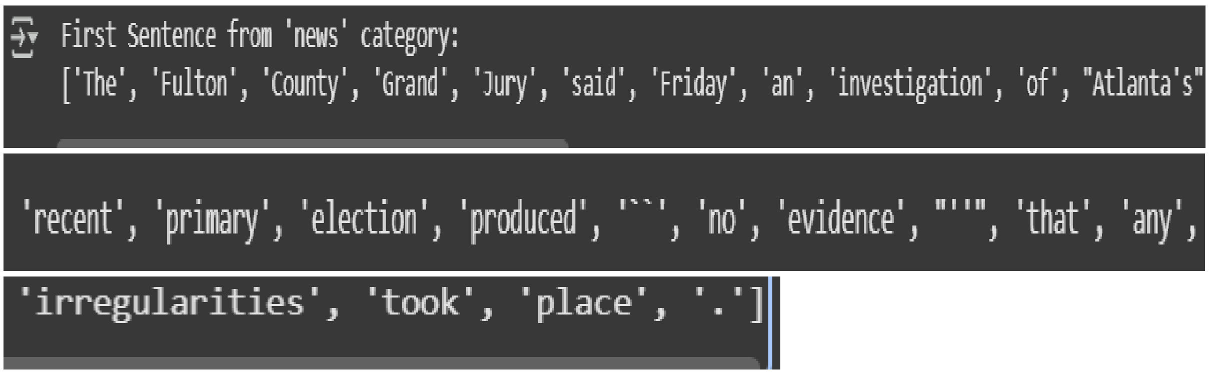
```
from nltk.corpus import brown
```



```
[['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investigation', 'of',
"Atlanta's", 'recent', 'primary', 'election', 'produced', '``', 'no', 'evidence', "''", 'that', 'any',
'irregularities', 'took', 'place', '.'], ['The', 'jury', 'further', 'said', 'in', 'term-end',
'presentments', 'that', 'the', 'City', 'Executive', 'Committee', ',', 'which', 'had', 'over-all',
'charge', 'of', 'the', 'election', ',', '``', 'deserves', 'the', 'praise', 'and', 'thanks', 'of',
'the', 'City', 'of', 'Atlanta', "''", 'for', 'the', 'manner', 'in', 'which', 'the', 'election', 'was',
'conducted', '.'], ...]
```

3. **Get the first sentence from the 'news' category of the brown corpus.**

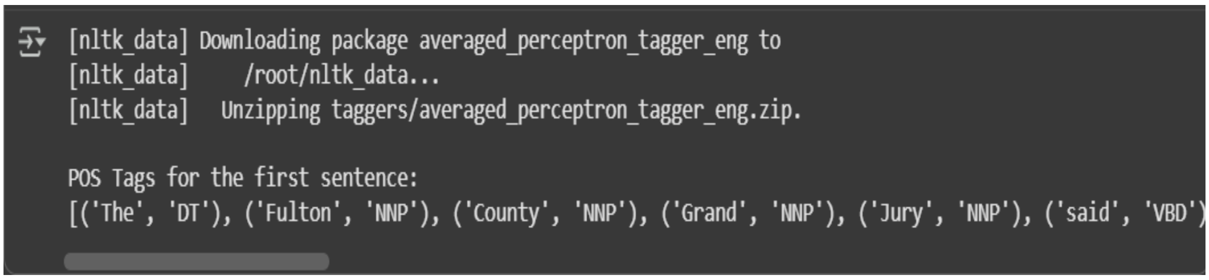
```
sentences = brown.sents(categories='news')

first_sentence = sentences[0]

print("First Sentence from 'news' category:")

print(first_sentence)
```



```
First Sentence from 'news' category:
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investigation', 'of', "Atlanta's"
'recent', 'primary', 'election', 'produced', '``', 'no', 'evidence', "''", 'that', 'any',
'irregularities', 'took', 'place', '.']
```

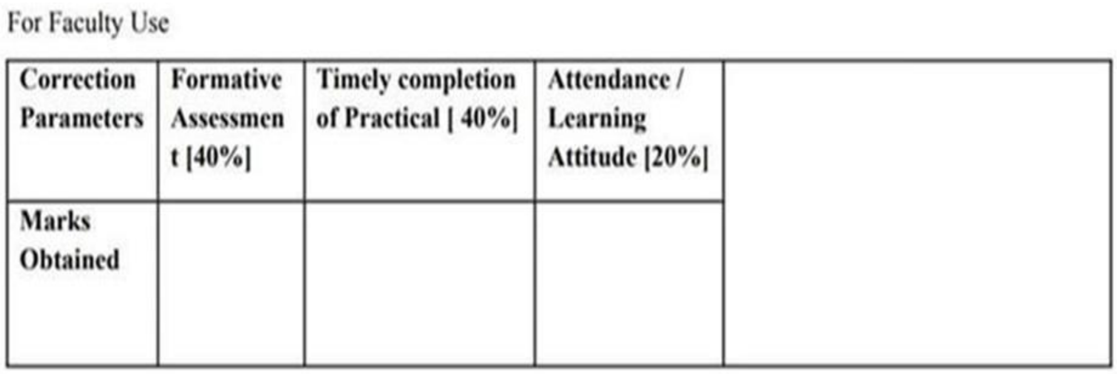
4. **Perform parts of speech tagging on this sentence and display the results.**

```
        nltk.download('averaged_perceptron_tagger_eng')
pos_tags = pos_tag(first_sentence)
        print("\nPOS Tags for the first sentence:")
        print(pos_tags)
```



```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.

POS Tags for the first sentence:
[('The', 'DT'), ('Fulton', 'NNP'), ('County', 'NNP'), ('Grand', 'NNP'), ('Jury', 'NNP'), ('said', 'VBD'
```

Conclusion: POS tagging identifies the grammatical roles of words in a sentence. This helps understand the structure, syntax, and meaning of the text, forming a foundation for advanced NLP tasks like parsing, named entity recognition, and semantic analysis.



For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

Name:- Ujwal Sahu
Roll No:- 34
Branch:- BVOC TY AIDS B
Subject:- Natural Language Processing

ASSIGNMENT 06

Title: Collocations

Theory: Collocations are pairs or groups of words that occur together unusually often in natural language.

In NLP, detecting collocations helps in:

- Language modeling
- Machine translation
- Text generation
- Keyword extraction

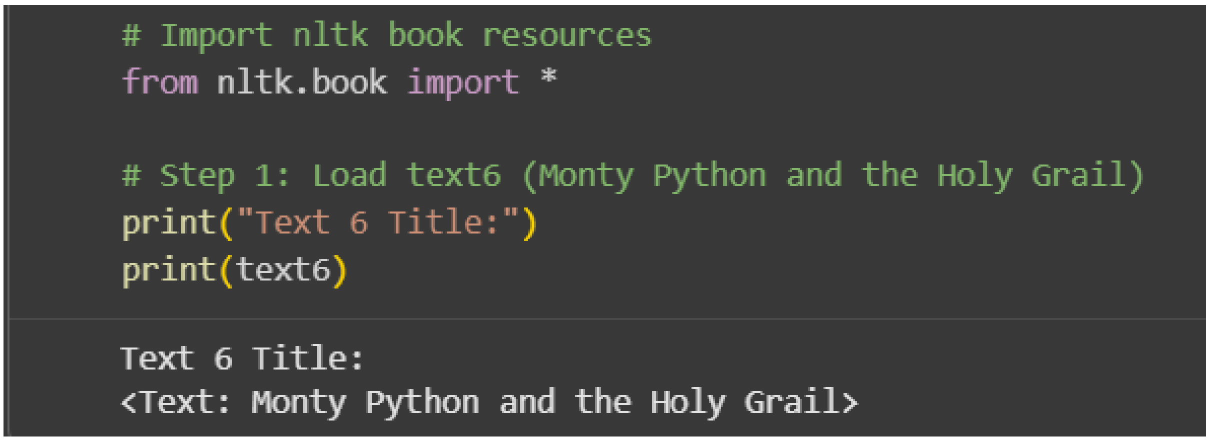
Code:

1. **Load text6 (Monty Python and the Holy Grail) from nltk.book.**

from nltk.book import *

print("Text 6 Title:")

print(text6)



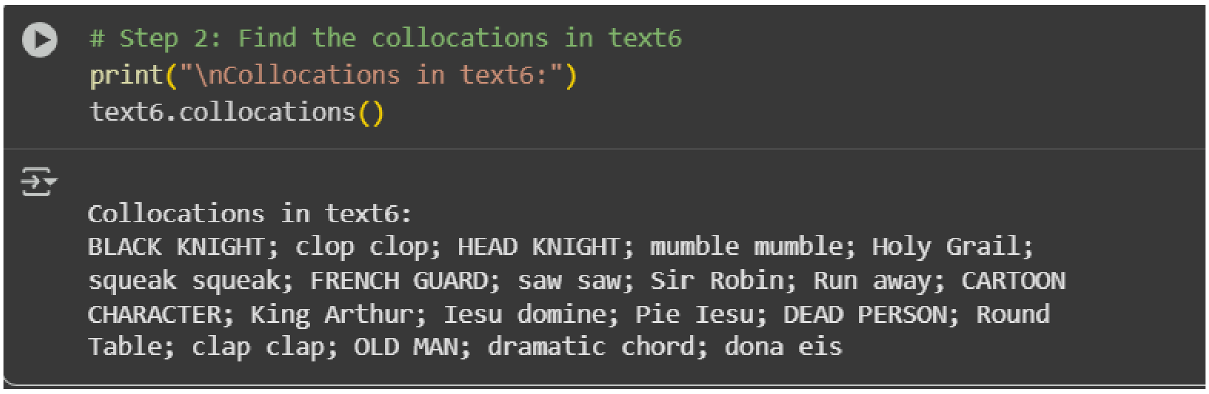
```
# Import nltk book resources
from nltk.book import *

# Step 1: Load text6 (Monty Python and the Holy Grail)
print("Text 6 Title:")
print(text6)
```

```
Text 6 Title:
<Text: Monty Python and the Holy Grail>
```

2. **Find the collocations in text6.**

print("\nCollocations in text6:")

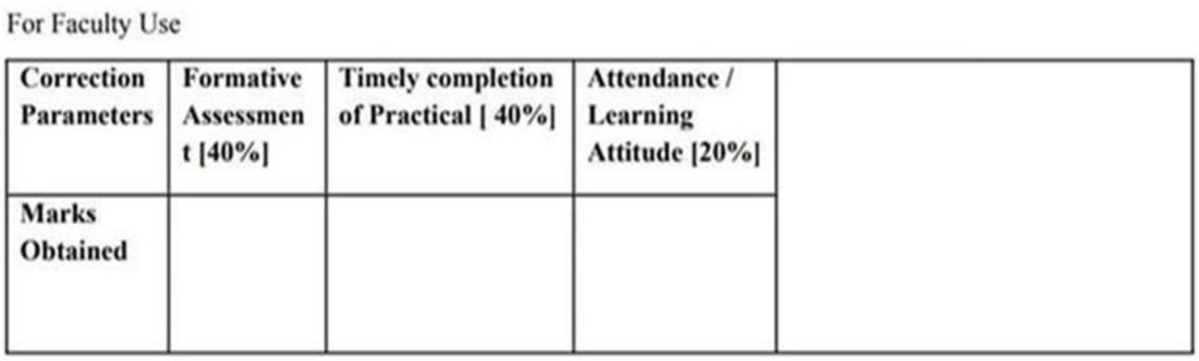
text6.collocations()



```
# Step 2: Find the collocations in text6
print("\nCollocations in text6:")
text6.collocations()
```

```

Collocations in text6:
BLACK KNIGHT; clop clop; HEAD KNIGHT; mumble mumble; Holy Grail;
squeak squeak; FRENCH GUARD; saw saw; Sir Robin; Run away; CARTOON
CHARACTER; King Arthur; Iesu domine; Pie Iesu; DEAD PERSON; Round
Table; clap clap; OLD MAN; dramatic chord; dona eis
```

Conclusion: Finding collocations reveals frequently co-occurring word pairs in a text. This helps identify common expressions, themes, and relationships between words, providing insights into natural language patterns and improving tasks like text generation and keyword extraction.



For Faculty Use

Correction Parameters	Formative Assessmen t [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

Name:- Ujwal Sahu
Roll No:- 34
Branch:- BVOC TY AIDS B
Subject:- Natural Language Processing

ASSIGNMENT 07

Title: Basic Frequency Distribution

Theory: A Frequency Distribution counts how many times each word appears in a text.

- Purpose in NLP:
    - Identify most common words
    - Understand text structure and themes
    - Preprocess for tasks like stopword removal, keyword extraction, or topic modeling

Code:

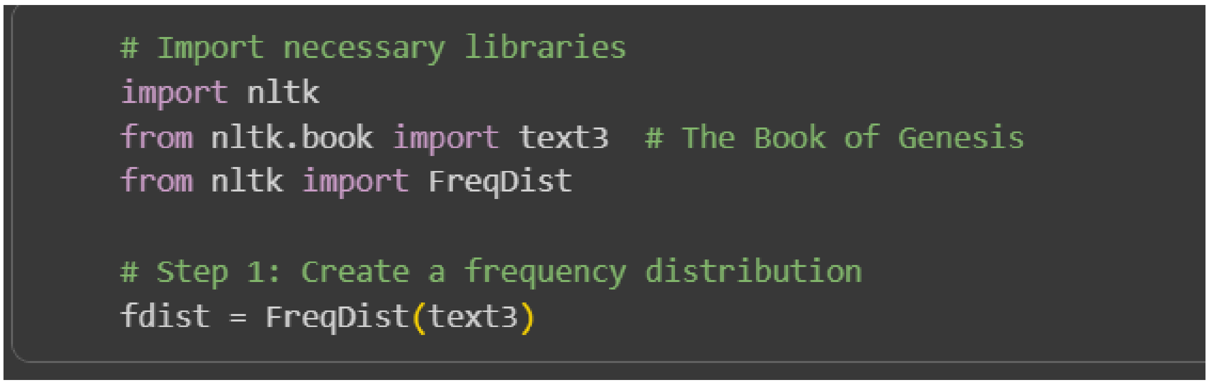
1. **Create a frequency distribution for text3 (The Book of Genesis).**

import nltk

from nltk.book import text3 # The Book of Genesis

from nltk import FreqDist

fdist = FreqDist(text3)

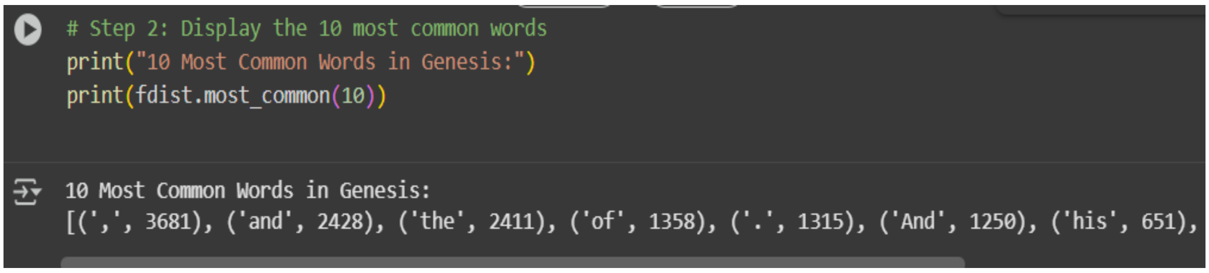


```
# Import necessary libraries
import nltk
from nltk.book import text3  # The Book of Genesis
from nltk import FreqDist

# Step 1: Create a frequency distribution
fdist = FreqDist(text3)
```

2. **Display the 10 most common words in text3.**

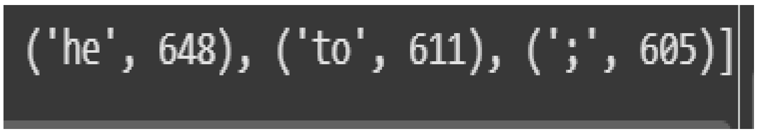
print("10 Most Common Words in Genesis:")

print(fdist.most_common(10))



```
# Step 2: Display the 10 most common words
print("10 Most Common Words in Genesis:")
print(fdist.most_common(10))
```

```
10 Most Common Words in Genesis:
[(',', 3681), ('and', 2428), ('the', 2411), ('of', 1358), ('.', 1315), ('And', 1250), ('his', 651),
```



```
('he', 648), ('to', 611), (';', 605)]
```

Conclusion: Frequency distribution highlights the most common words in a text. It helps in understanding word usage patterns, identifying key terms, and serves as a basis for further analysis like stopword removal, lexical richness, and thematic exploration.

For Faculty Use

Correction Parameters	Formative Assessmen t [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				

Name:- Ujwal Sahu
Roll No:- 34
Branch:- BVOC TY AIDS B
Subject:- Natural Language Processing

ASSIGNMENT 08

Title: Counting Specific Words

Theory: Counting the occurrences of a specific word is a basic frequency analysis task in NLP. It is useful for Understand the importance of key terms in a text, Compare word usage across texts and Useful in text mining, sentiment analysis, and keyword extraction.

Methods:

1. Using .count() on the text (fast and simple).
2. Using FreqDist for more advanced analysis.

Code:

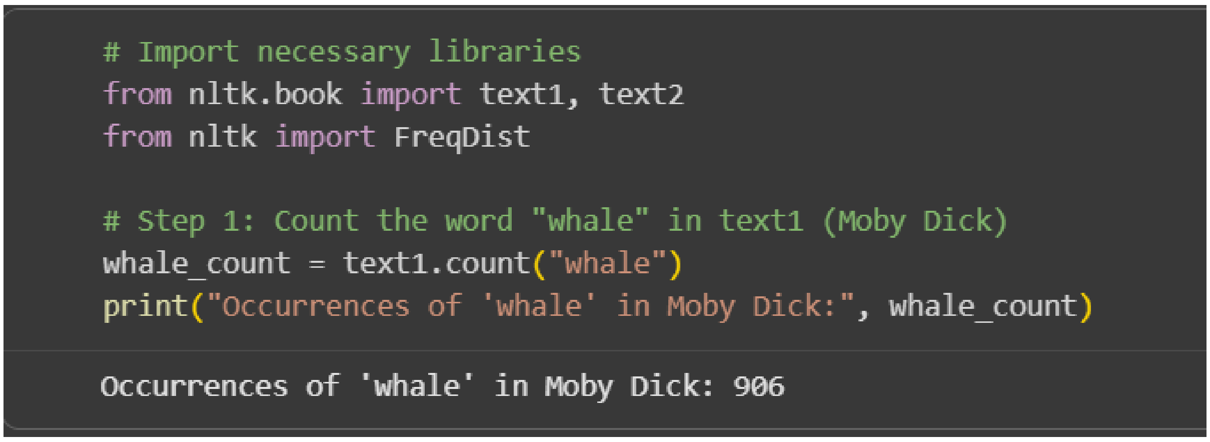
1. **Count the occurrences of the word "whale" in text1 (Moby Dick).**

   from nltk.book import text1, text2

   from nltk import FreqDist

   whale_count = text1.count("whale")

   print("Occurrences of 'whale' in Moby Dick:", whale_count)



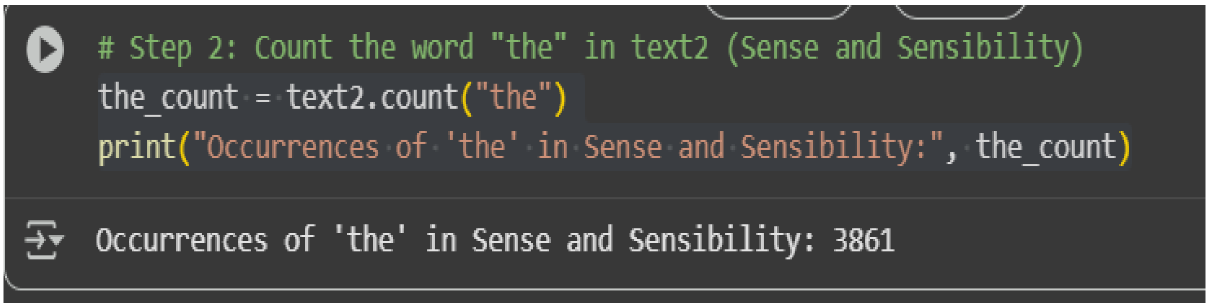
```
# Import necessary libraries
from nltk.book import text1, text2
from nltk import FreqDist

# Step 1: Count the word "whale" in text1 (Moby Dick)
whale_count = text1.count("whale")
print("Occurrences of 'whale' in Moby Dick:", whale_count)
```

```
Occurrences of 'whale' in Moby Dick: 906
```

2. **Count the occurrences of the word "the" in text2 (Sense and Sensibility).**

   the_count = text2.count("the")

   print("Occurrences of 'the' in Sense and Sensibility:", the_count)



```
# Step 2: Count the word "the" in text2 (Sense and Sensibility)
the_count = text2.count("the")
print("Occurrences of 'the' in Sense and Sensibility:", the_count)
```

```
Occurrences of 'the' in Sense and Sensibility: 3861
```

Conclusion: Counting specific words quantifies their importance and emphasis in a text. By analyzing word occurrences, we can understand themes, narrative focus, and stylistic patterns, which is useful in text analysis and content summarization.

For Faculty Use

Correction Parameters	Formative Assessmen t [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				