EX.NO.: 02 Stemming, Lemmatization, Stopwords
DATE: 20.12.2024

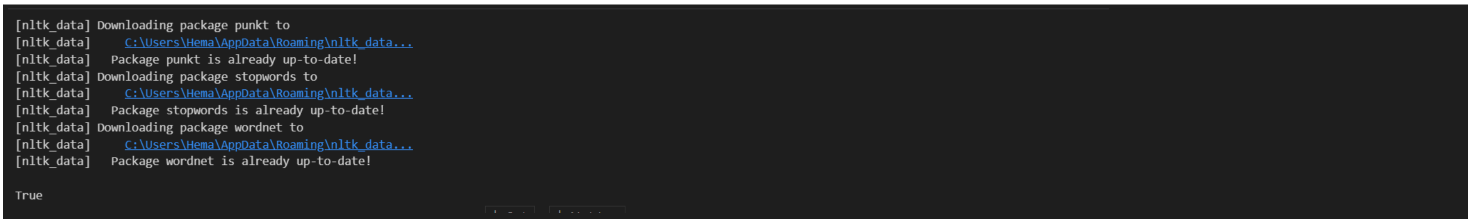
To perform text processing on a given document containing at least 200 words. Specific tasks that include tokenization, stemming, lemmatization and stopwords.

PROCEDURE:

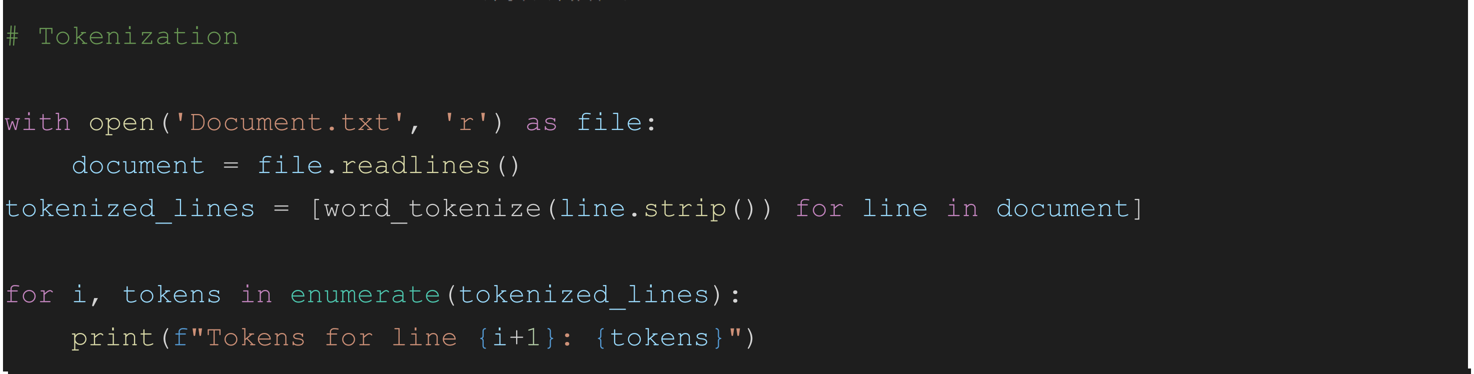
1. Reading and tokenizing
   a. Read document line by line
      > readline ()
   b. Tokenize each line into words
      > word_tokenize (line.strip())
2. Stemming
   a. Reduce the words to their root form
      > PorterStemmer()
   b. Apply stemming to each token in tokenize_lines
      > stemmer.stem(token)
   c. Print and save stemmed word to a file
      > open (‘stemmed_words.txt’, ‘w’) as output file:
      > output_file.write(...)
3. Lemmatization
   a. Tokenize a sample sentence
   b. Tag each token with its POS to help lemmatize based on the context of each word
      > pos_tag(tokens)
   c. Initialize WordNetLemmatizer()
   d. Convert treebank POS tags to WordNet POS tags for better accuracy
      > treebank_tag.startswith(‘’):
   e. Apply lemmatization to each token using POS tag
      > get_wordnet_pos(tag)
   f. Print and save lemmatized words to file
4. Stopwords removal
   a. Load list of stopwords
      > set(stopwords.words(‘english’))
   b. Tokenize each line of document
      > word_tokenize(line.strip())
   c. Filter out stopwords from each tokenized line
      > … if token.lower() not in stop_words
   d. Print and display filtered lines

CODE AND OUTPUT:

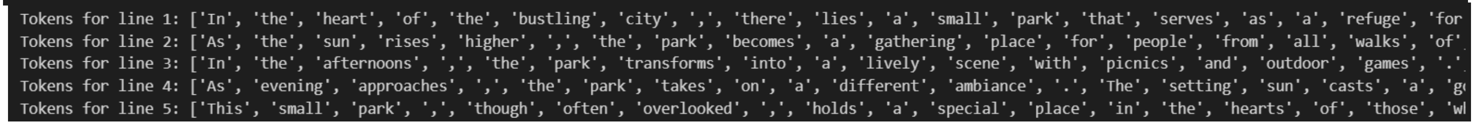
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```



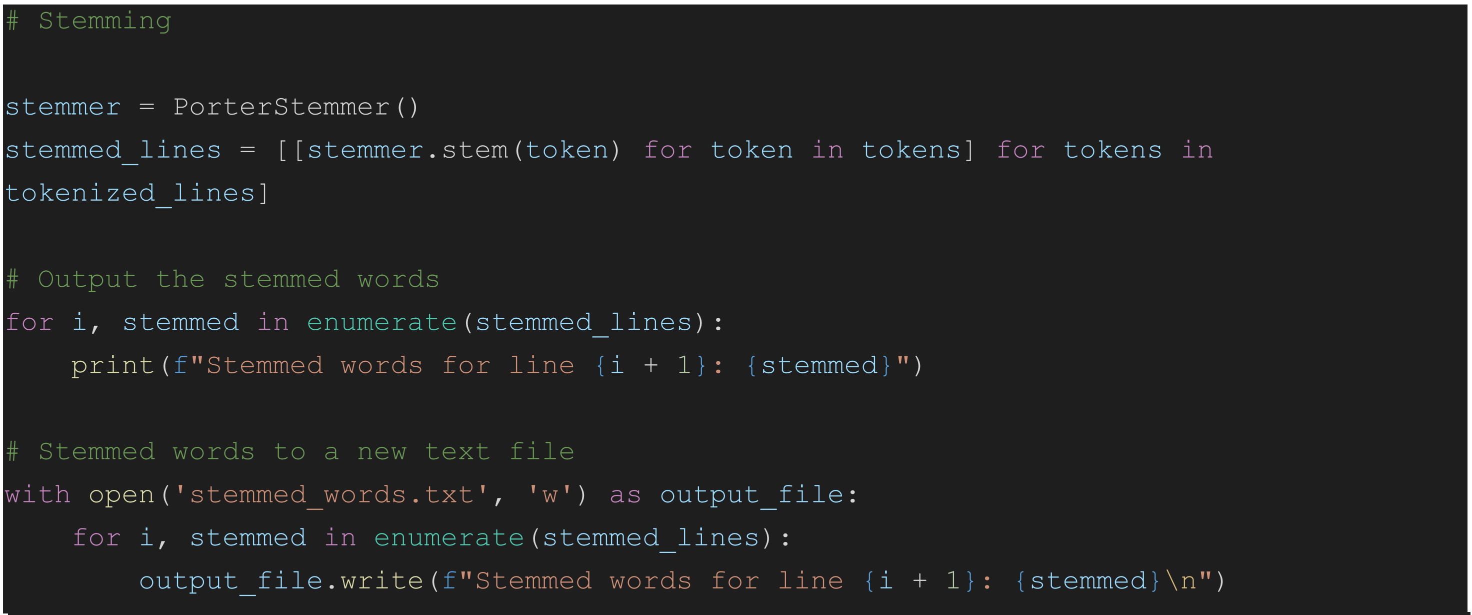
```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Hema\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Hema\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already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

True
```



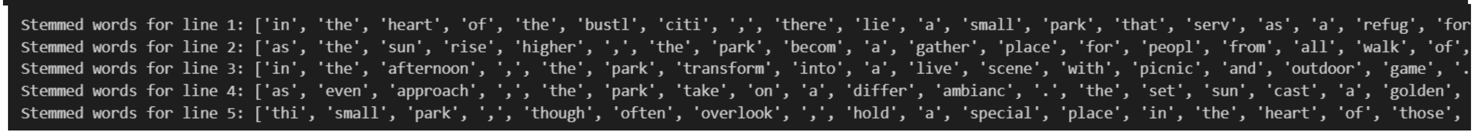
```
# Tokenization

with open('Document.txt', 'r') as file:
    document = file.readlines()
tokenized_lines = [word_tokenize(line.strip()) for line in document]

for i, tokens in enumerate(tokenized_lines):
    print(f"Tokens for line {i+1}: {tokens}")
```



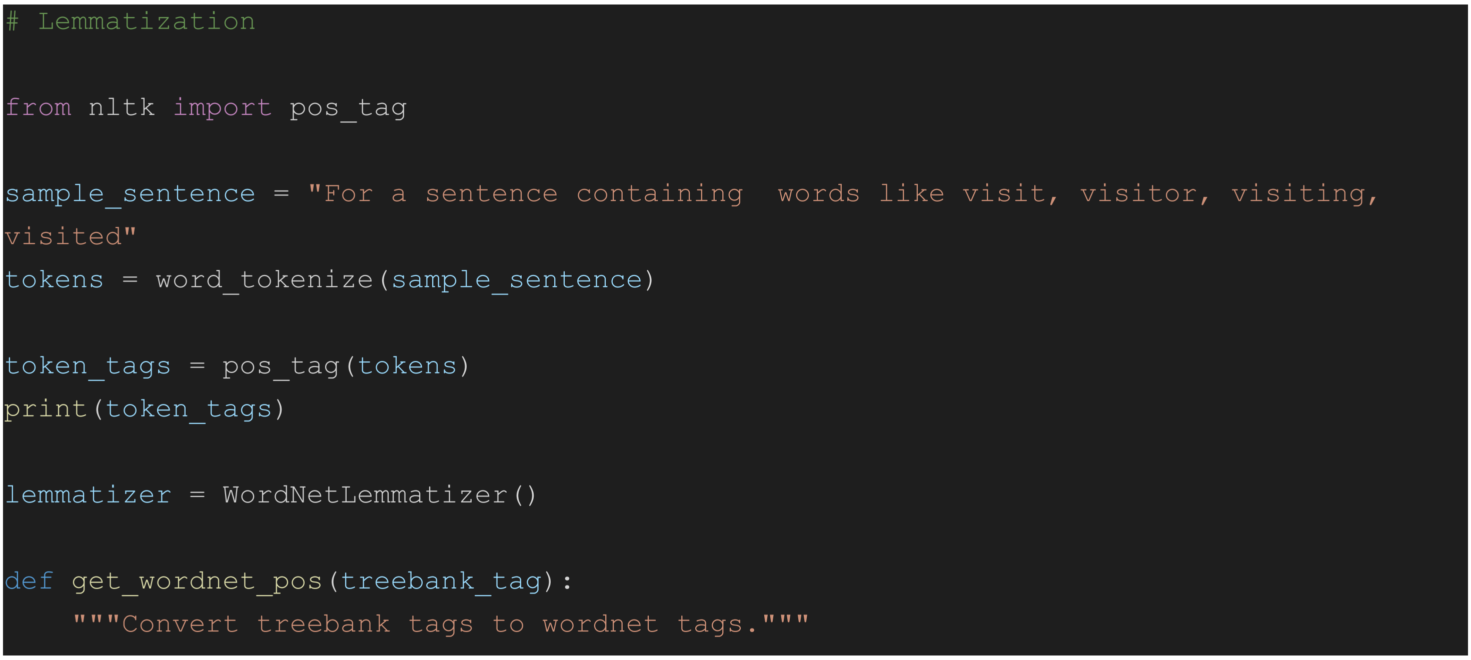
```
Tokens for line 1: ['In', 'the', 'heart', 'of', 'the', 'bustling', 'city', ',', 'there', 'lies', 'a', 'small', 'park', 'that', 'serves', 'as', 'a', 'refuge', 'for
Tokens for line 2: ['As', 'the', 'sun', 'rises', 'higher', ',', 'the', 'park', 'becomes', 'a', 'gathering', 'place', 'for', 'people', 'from', 'all', 'walks', 'of'
Tokens for line 3: ['In', 'the', 'afternoons', ',', 'the', 'park', 'transforms', 'into', 'a', 'lively', 'scene', 'with', 'picnics', 'and', 'outdoor', 'games', '.'
Tokens for line 4: ['As', 'evening', 'approaches', ',', 'the', 'park', 'takes', 'on', 'a', 'different', 'ambiance', '.', 'The', 'setting', 'sun', 'casts', 'a', 'g
Tokens for line 5: ['This', 'small', 'park', ',', 'though', 'often', 'overlooked', ',', 'holds', 'a', 'special', 'place', 'in', 'the', 'hearts', 'of', 'those', 'w
```



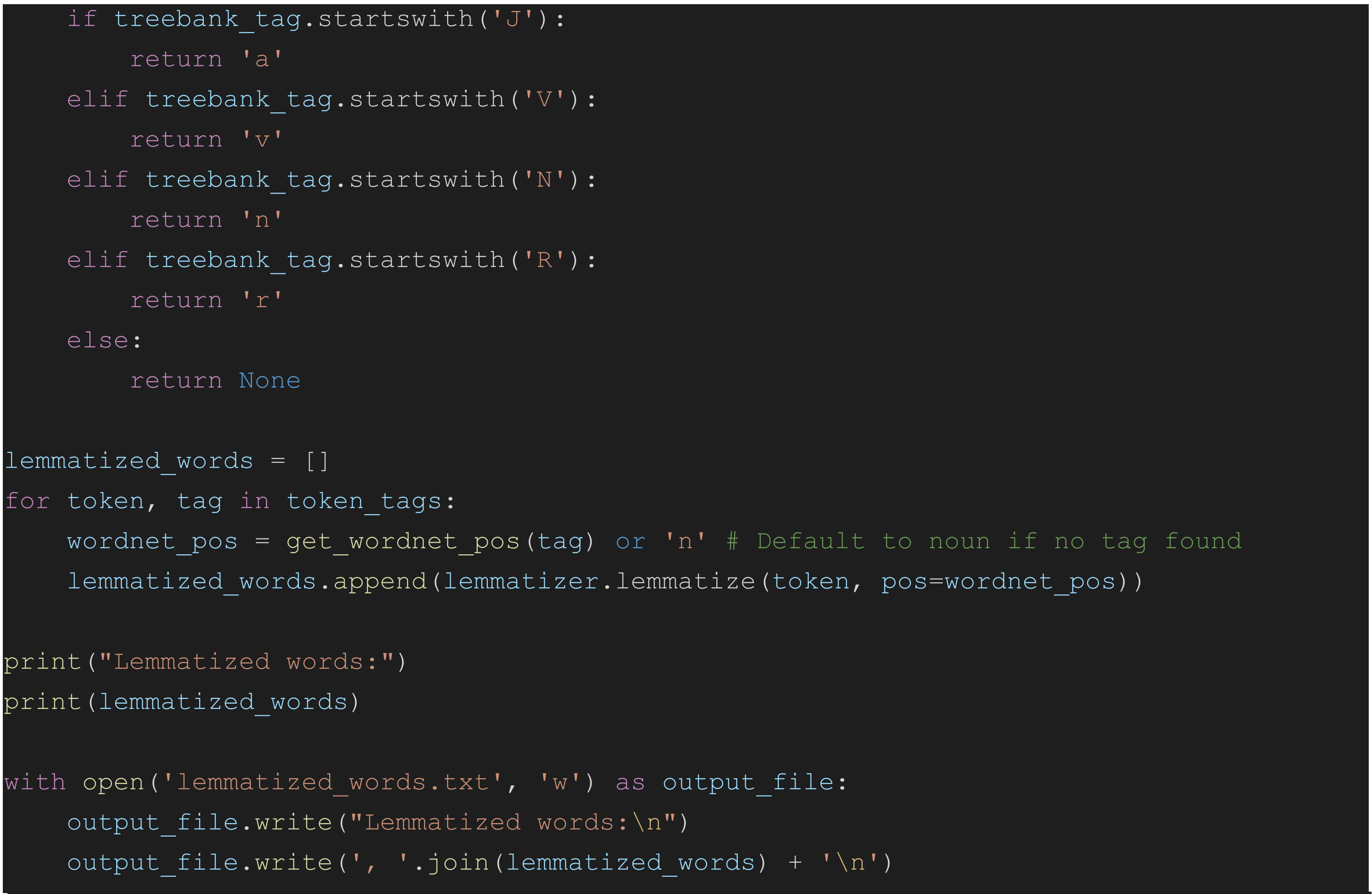
```
# Stemming

stemmer = PorterStemmer()
stemmed_lines = [[stemmer.stem(token) for token in tokens] for tokens in
tokenized_lines]

# Output the stemmed words
for i, stemmed in enumerate(stemmed_lines):
    print(f"Stemmed words for line {i + 1}: {stemmed}")

# Stemmed words to a new text file
with open('stemmed_words.txt', 'w') as output_file:
    for i, stemmed in enumerate(stemmed_lines):
        output_file.write(f"Stemmed words for line {i + 1}: {stemmed}\n")
```



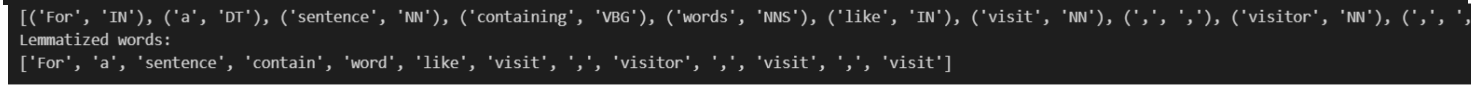
```
Stemmed words for line 1: ['in', 'the', 'heart', 'of', 'the', 'bustl', 'citi', ',', 'there', 'lie', 'a', 'small', 'park', 'that', 'serv', 'as', 'a', 'refug', 'for
Stemmed words for line 2: ['as', 'the', 'sun', 'rise', 'higher', ',', 'the', 'park', 'becom', 'a', 'gather', 'place', 'for', 'peopl', 'from', 'all', 'walk', 'of',
Stemmed words for line 3: ['in', 'the', 'afternoon', ',', 'the', 'park', 'transform', 'into', 'a', 'live', 'scene', 'with', 'picnic', 'and', 'outdoor', 'game', '.
Stemmed words for line 4: ['as', 'even', 'approach', ',', 'the', 'park', 'take', 'on', 'a', 'differ', 'ambianc', '.', 'the', 'set', 'sun', 'cast', 'a', 'golden',
Stemmed words for line 5: ['thi', 'small', 'park', ',', 'though', 'often', 'overlook', ',', 'hold', 'a', 'special', 'place', 'in', 'the', 'heart', 'of', 'those',
```



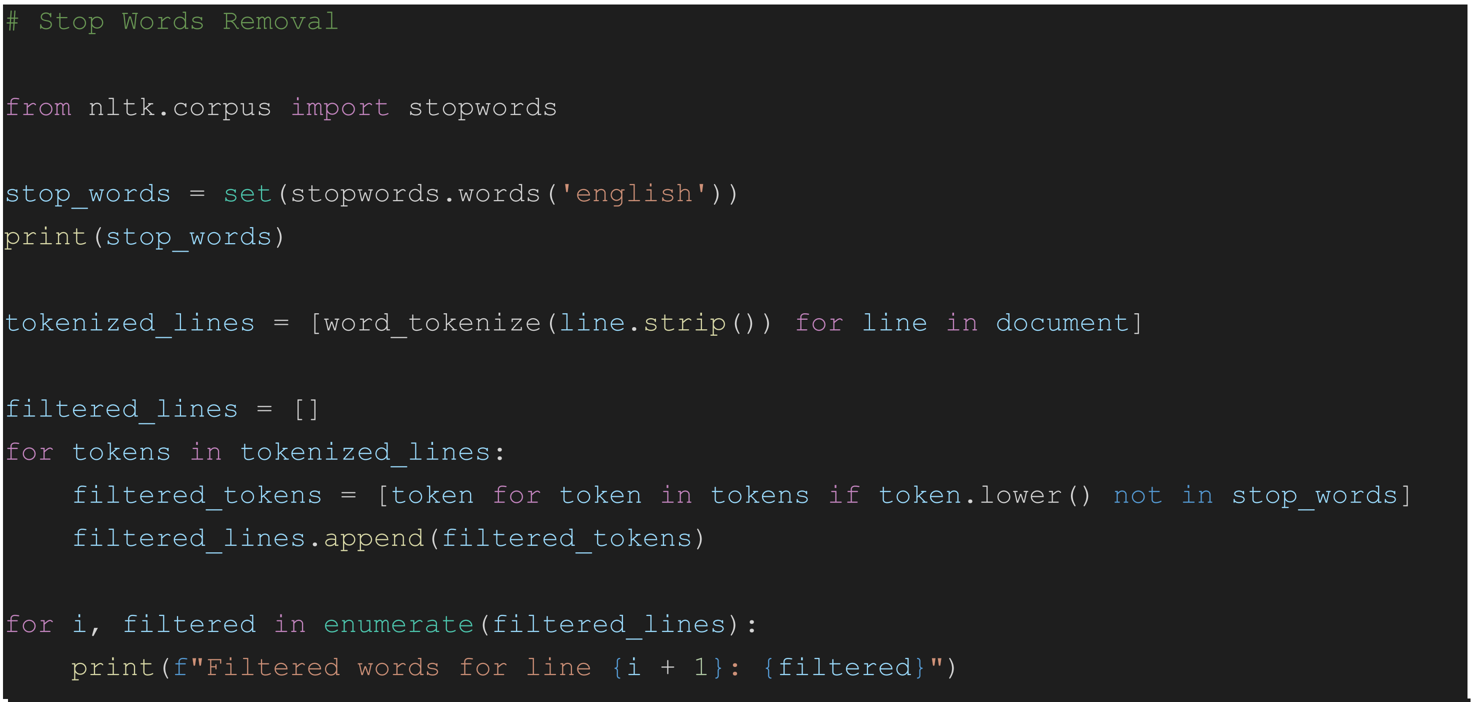
```
# Lemmatization

from nltk import pos_tag

sample_sentence = "For a sentence containing  words like visit, visitor, visiting,
visited"
tokens = word_tokenize(sample_sentence)

token_tags = pos_tag(tokens)
print(token_tags)

lemmatizer = WordNetLemmatizer()

def get_wordnet_pos(treebank_tag):
    """Convert treebank tags to wordnet tags."""
```



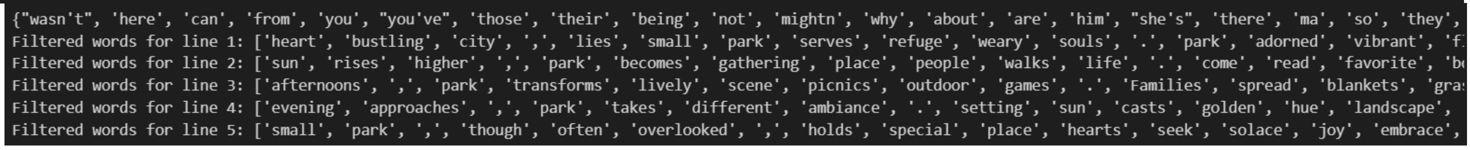
```
    if treebank_tag.startswith('J'):
        return 'a'
    elif treebank_tag.startswith('V'):
        return 'v'
    elif treebank_tag.startswith('N'):
        return 'n'
    elif treebank_tag.startswith('R'):
        return 'r'
    else:
        return None

lemmatized_words = []
for token, tag in token_tags:
    wordnet_pos = get_wordnet_pos(tag) or 'n' # Default to noun if no tag found
    lemmatized_words.append(lemmatizer.lemmatize(token, pos=wordnet_pos))

print("Lemmatized words:")
print(lemmatized_words)

with open('lemmatized_words.txt', 'w') as output_file:
    output_file.write("Lemmatized words:\n")
    output_file.write(', '.join(lemmatized_words) + '\n')
```



```
[('For', 'IN'), ('a', 'DT'), ('sentence', 'NN'), ('containing', 'VBG'), ('words', 'NNS'), ('like', 'IN'), ('visit', 'NN'), (',', ','), ('visitor', 'NN'), (',', ',
Lemmatized words:
['For', 'a', 'sentence', 'contain', 'word', 'like', 'visit', ',', 'visitor', ',', 'visit', ',', 'visit']
```



```
# Stop Words Removal

from nltk.corpus import stopwords

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

tokenized_lines = [word_tokenize(line.strip()) for line in document]

filtered_lines = []
for tokens in tokenized_lines:
    filtered_tokens = [token for token in tokens if token.lower() not in stop_words]
    filtered_lines.append(filtered_tokens)

for i, filtered in enumerate(filtered_lines):
    print(f"Filtered words for line {i + 1}: {filtered}")
```



```
{"wasn't", 'here', 'can', 'from', 'you', "you've", 'those', 'their', 'being', 'not', 'mightn', 'why', 'about', 'are', 'him', "she's", 'there', 'ma', 'so', 'they',
Filtered words for line 1: ['heart', 'bustling', 'city', ',', 'lies', 'small', 'park', 'serves', 'refuge', 'weary', 'souls', '.', 'park', 'adorned', 'vibrant', 'f
Filtered words for line 2: ['sun', 'rises', 'higher', ',', 'park', 'becomes', 'gathering', 'place', 'people', 'walks', 'life', '.', 'come', 'read', 'favorite', 'b
Filtered words for line 3: ['afternoons', ',', 'park', 'transforms', 'lively', 'scene', 'picnics', 'outdoor', 'games', '.', 'Families', 'spread', 'blankets', 'gra
Filtered words for line 4: ['evening', 'approaches', ',', 'park', 'takes', 'different', 'ambiance', '.', 'setting', 'sun', 'casts', 'golden', 'hue', 'landscape',
Filtered words for line 5: ['small', 'park', ',', 'though', 'often', 'overlooked', ',', 'holds', 'special', 'place', 'hearts', 'seek', 'solace', 'joy', 'embrace',
```

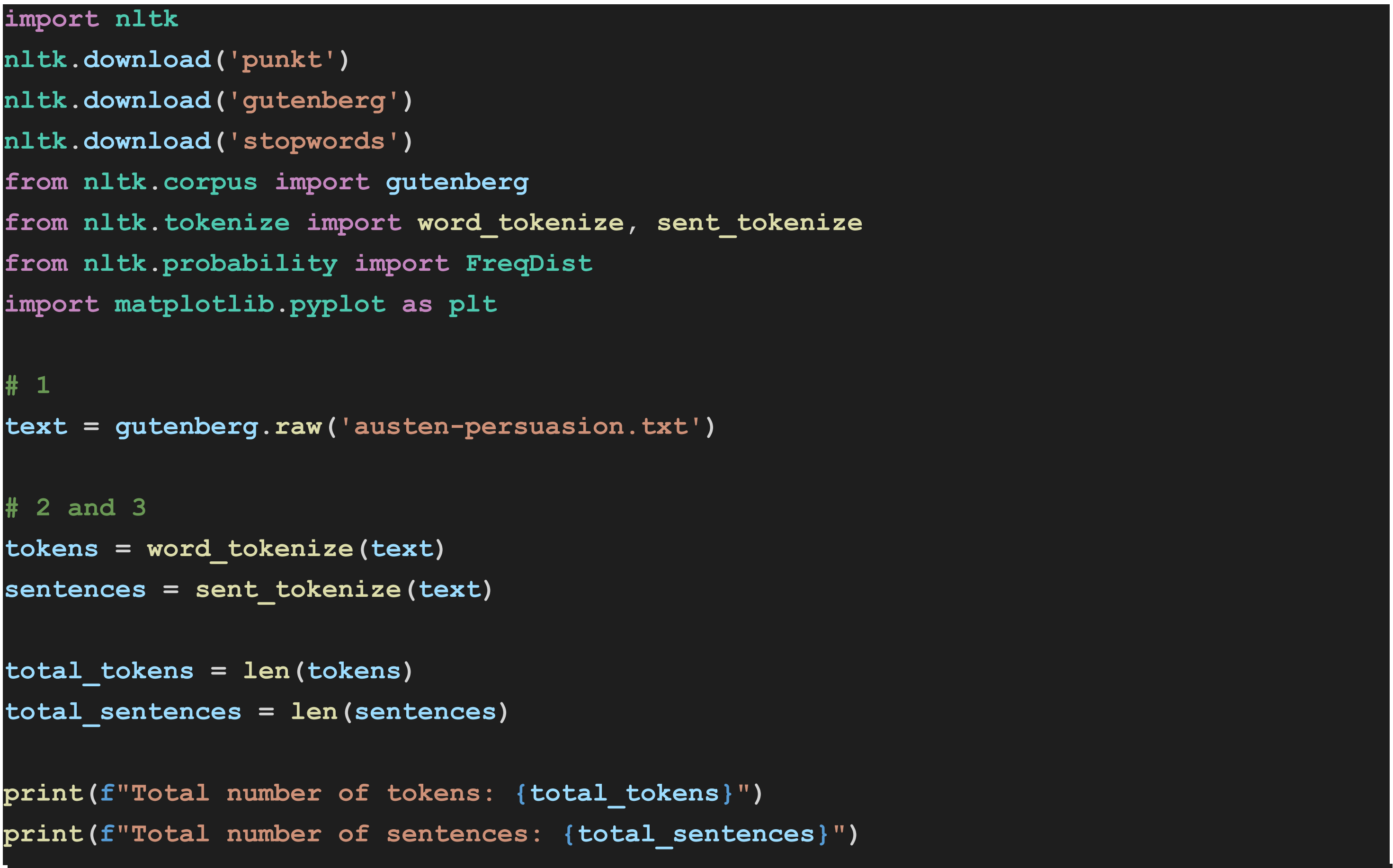
EX.NO.: 03 Word Tokenization and Analysis
DATE: 20.12.2024

To Implement a program to analyze a small corpus of text using word tokenization. Tasks like tokenizing text, counting total no. of tokens and sentences, finding the frequency of words, identifying the frequency of a specific word, display top 5 words with highest frequency and creating a dispersion plot for these words.

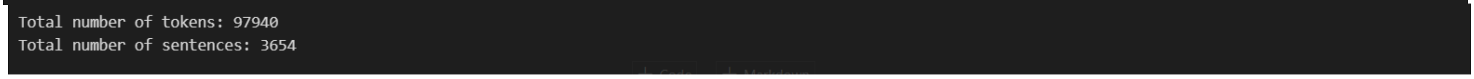
PROCEDURE:

1. Download necessary NLTK data: punkt, gutenberg, stopwords
2. Load a small corpus of text from NLTK library
3. Tokenization
   a. Text into words: word_tokenize(text)
   b. Text into sentence: sent_tokenize(text)
4. Counting tokens and sentences
   a. Count total tokens: len(tokens)
   b. Count total sentence: len(sentences)
5. Frequency distribution
   a. Calculate frequency distribution: FreqDist(tokens)
6. Specify word frequency 'Emma'
   > freq_dist[specific_word]
7. Occurrences of particular word
   a. Total occurrences of particular word 'love'
      > tokens.count(word_to_find)
8. Top 5 words
   a. Top 5 words with high frequency: freq_dist.most_common(5)
9. Dispersion Plot
   a. Plot for top 5 words using matplotlib
      > nltk.draw.dispersion_plot(tokens, words_to_plot)

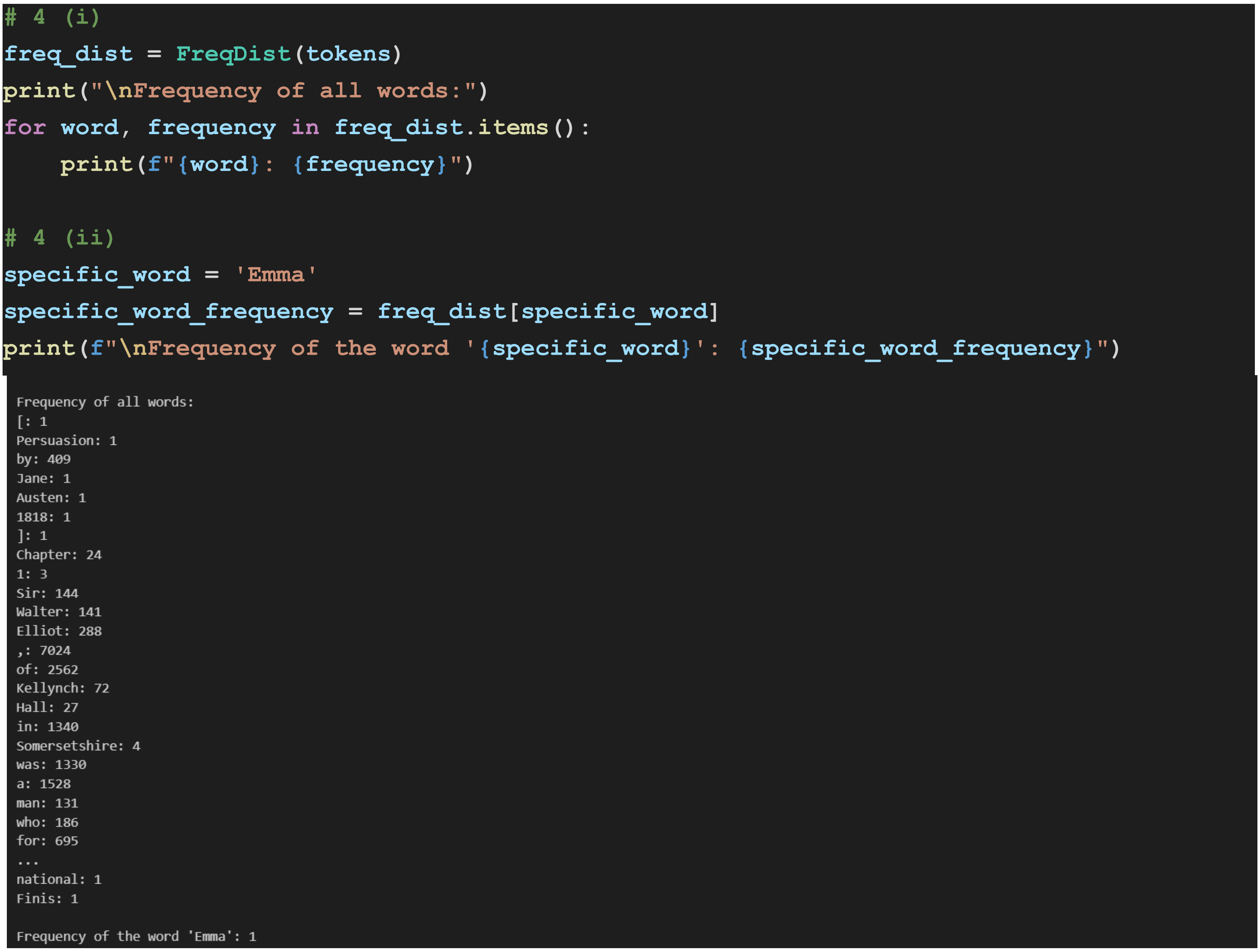
CODE AND OUTPUT:



```
import nltk
nltk.download('punkt')
nltk.download('gutenberg')
nltk.download('stopwords')
from nltk.corpus import gutenberg
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt

# 1
text = gutenberg.raw('austen-persuasion.txt')

# 2 and 3
tokens = word_tokenize(text)
sentences = sent_tokenize(text)

total_tokens = len(tokens)
total_sentences = len(sentences)

print(f"Total number of tokens: {total_tokens}")
print(f"Total number of sentences: {total_sentences}")
```

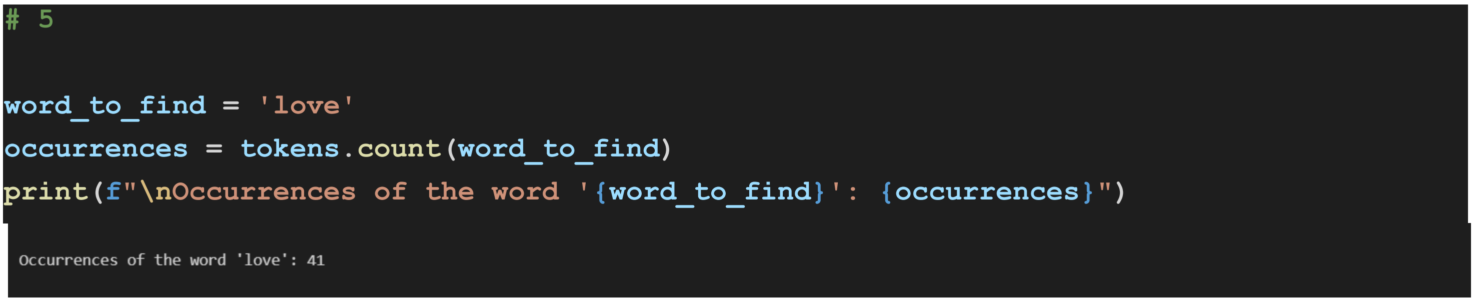


```
Total number of tokens: 97940
Total number of sentences: 3654
```



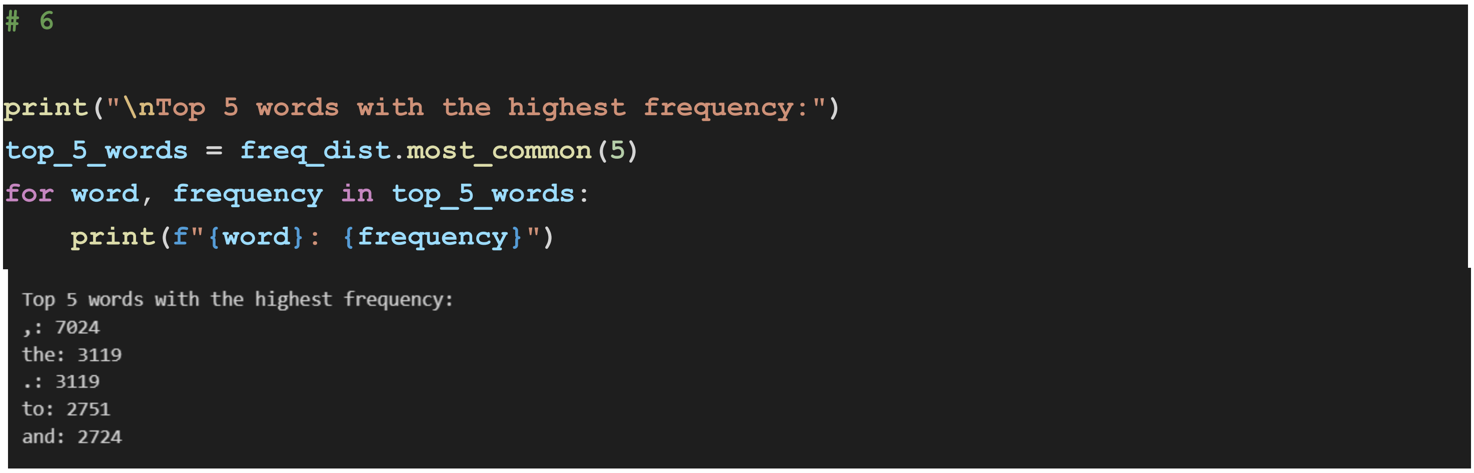
```
# 4 (i)
freq_dist = FreqDist(tokens)
print("\nFrequency of all words:")
for word, frequency in freq_dist.items():
    print(f"{word}: {frequency}")

# 4 (ii)
specific_word = 'Emma'
specific_word_frequency = freq_dist[specific_word]
print(f"\nFrequency of the word '{specific_word}': {specific_word_frequency}")
```

```
Frequency of all words:
[: 1
Persuasion: 1
by: 409
Jane: 1
Austen: 1
1818: 1
]: 1
Chapter: 24
1: 3
Sir: 144
Walter: 141
Elliot: 288
,: 7024
of: 2562
Kellynch: 72
Hall: 27
in: 1340
Somersetshire: 4
was: 1330
a: 1528
man: 131
who: 186
for: 695
...
national: 1
Finis: 1

Frequency of the word 'Emma': 1
```



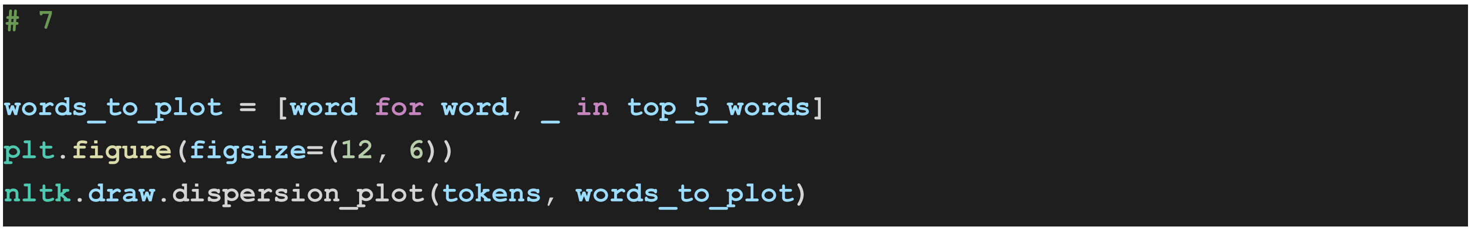
```
# 5

word_to_find = 'love'
occurrences = tokens.count(word_to_find)
print(f"\nOccurrences of the word '{word_to_find}': {occurrences}")
```

```
Occurrences of the word 'love': 41
```

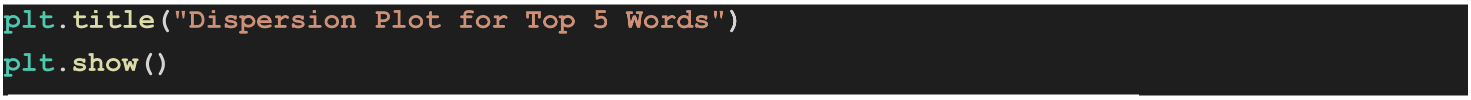


```
# 6

print("\nTop 5 words with the highest frequency:")
top_5_words = freq_dist.most_common(5)
for word, frequency in top_5_words:
    print(f"{word}: {frequency}")
```

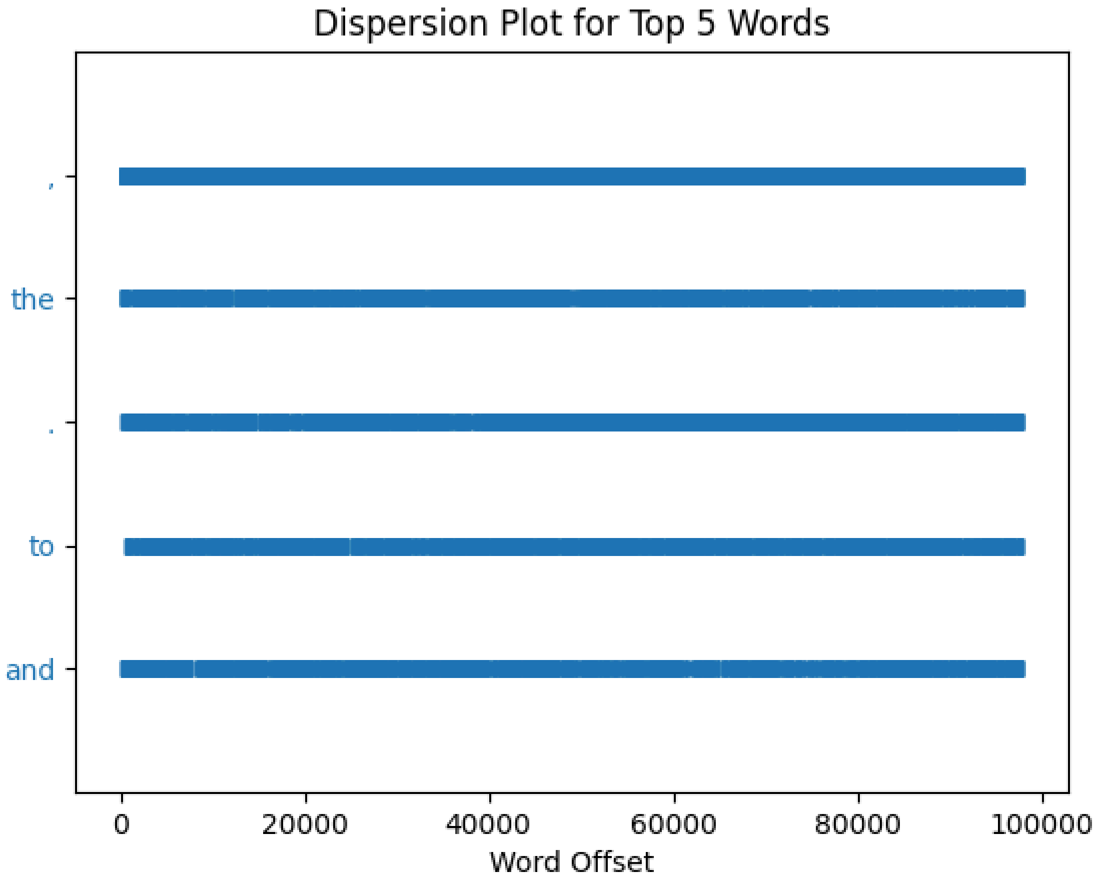
```
Top 5 words with the highest frequency:
,: 7024
the: 3119
.: 3119
to: 2751
and: 2724
```



```
# 7

words_to_plot = [word for word, _ in top_5_words]
plt.figure(figsize=(12, 6))
nltk.draw.dispersion_plot(tokens, words_to_plot)
```



```
plt.title("Dispersion Plot for Top 5 Words")
plt.show()
```



EX.NO.: 04

DATE: 20.12.2024

Regular Expressions in NLTK

To demonstrate the functionality of various regular expressions (regex) operators using a sample corpus, and then understand each operators functioning and type of match it produces.

PROCEDURE:

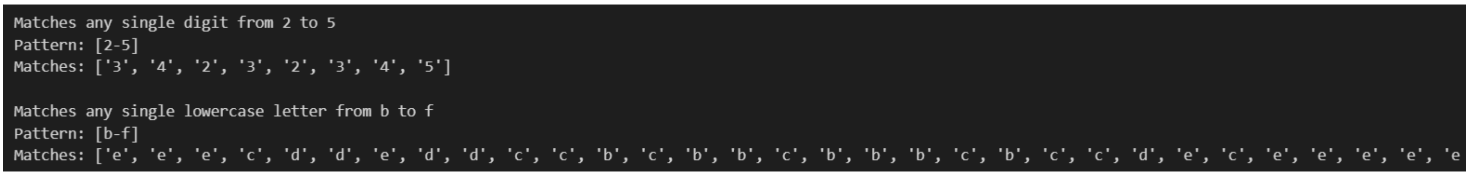
1. Prepare a sample text document (corpus.txt) containing various sentences and phrases.
2. Write python script to read the file and apply different regex patterns to it.

Operators:

1. Hyphen ([2-5],[b-f])
    a. Matches digits from 2-5
    b. Matches lowercase letters from b-f
2. Caret(^)
    a. Matches the start line (^H)
3. Question mark (?)
    a. Colou?r matches ‘color’ or ‘colour’
4. Kleene start (*)
    a. ab* matches ‘a’, ‘ab’, ‘abb’, ‘abbb’, etc (zero or more occurrences)
5. Kleene plus (+)
    a. Ab+ matches ‘ab’, ‘abb’, ‘abbb’, etc (one or more occurrences)
6. Dot (.)
    a. Matches single character except a newline
    b. .1 matches ‘a’, ‘b1’, ‘c1’, ‘d1’, etc
7. Pipe (|)
    a. OR operator
    b. Cat|dog matches either of them
8. Word boundary (\b)
    a. Matches position between a word and a non-word character
    b. \bthe\b matches the word ‘the’
9. Non-word boundary (\B)
    a. Match a position that is not a word boundary
    b. \Bthe\B match ‘the’ in the middle of a word
10. Pattern [^a-zA-Z][tT]he[^a-zA-Z]
    a. Matches ‘the’ or ‘The’ when surrounded by non-letter characters

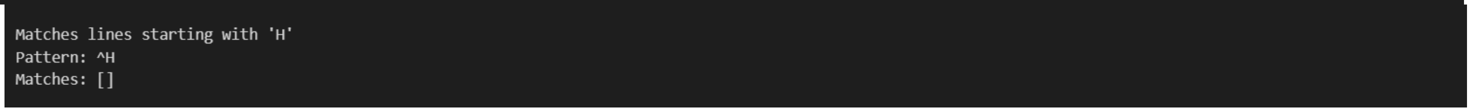
CODE AND OUTPUT:

```
import re

# Read corpus from external file
with open('corpus.txt', 'r') as file:
    corpus = file.read()

# Function to demonstrate regex matching
def demonstrate_regex(pattern, description):
    matches = re.findall(pattern, corpus)
    print(f"\n{description}\nPattern: {pattern}\nMatches: {matches}")

# 1. Hyphen [2-5] and [b-f]
demonstrate_regex(r'[2-5]', "Matches any single digit from 2 to 5")
demonstrate_regex(r'[b-f]', "Matches any single lowercase letter from b to f")
```



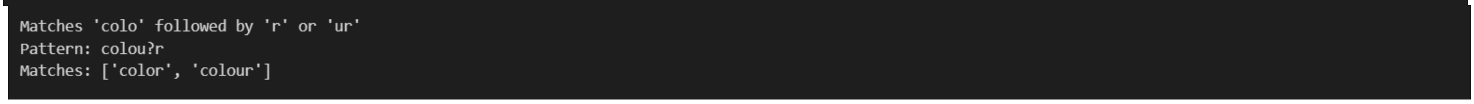
```
Matches any single digit from 2 to 5
Pattern: [2-5]
Matches: ['3', '4', '2', '3', '2', '3', '4', '5']

Matches any single lowercase letter from b to f
Pattern: [b-f]
Matches: ['e', 'e', 'e', 'c', 'd', 'd', 'e', 'd', 'd', 'c', 'c', 'b', 'c', 'b', 'b', 'c', 'b', 'b', 'b', 'c', 'b', 'c', 'c', 'd', 'e', 'c', 'e', 'e', 'e', 'e', 'e
```

```
# 2. Caret symbol (^)
demonstrate_regex(r'^H', "Matches lines starting with 'H'")
```



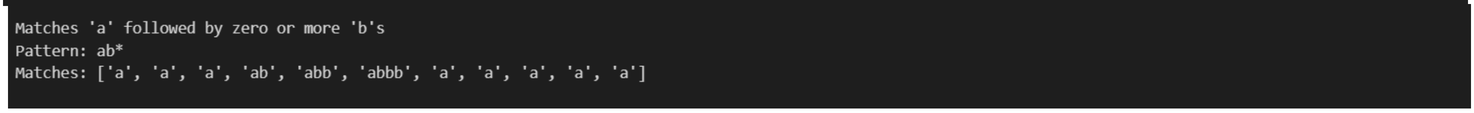
```
Matches lines starting with 'H'
Pattern: ^H
Matches: []
```

```
# 3. Question mark (?) operator
demonstrate_regex(r'colou?r', "Matches 'colo' followed by 'r' or 'ur'")
```



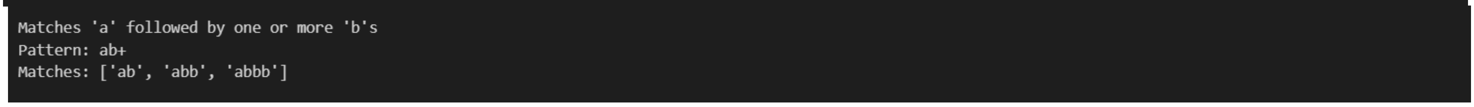
```
Matches 'colo' followed by 'r' or 'ur'
Pattern: colou?r
Matches: ['color', 'colour']
```

```
# 4. Kleene star (*) operator
demonstrate_regex(r'ab*', "Matches 'a' followed by zero or more 'b's")
```



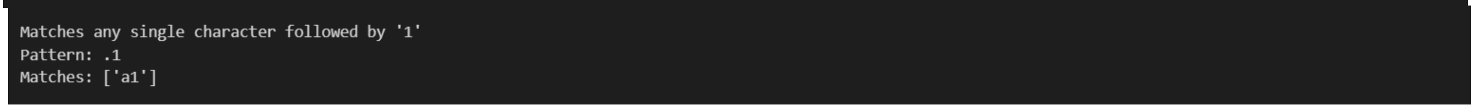
```
Matches 'a' followed by zero or more 'b's
Pattern: ab*
Matches: ['a', 'a', 'a', 'ab', 'abb', 'abbb', 'a', 'a', 'a', 'a', 'a']
```

```
# 5. Kleene plus (+) operator
demonstrate_regex(r'ab+', "Matches 'a' followed by one or more 'b's")
```



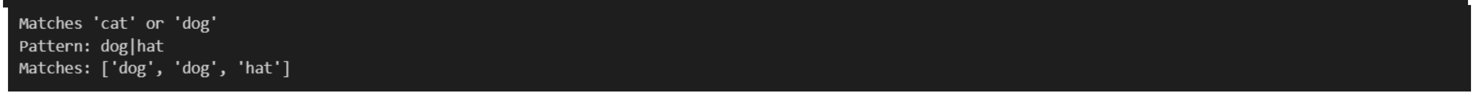
```
Matches 'a' followed by one or more 'b's
Pattern: ab+
Matches: ['ab', 'abb', 'abbb']
```

```
# 6. Dot (.) operator
demonstrate_regex(r'.1', "Matches any single character followed by '1'")
```



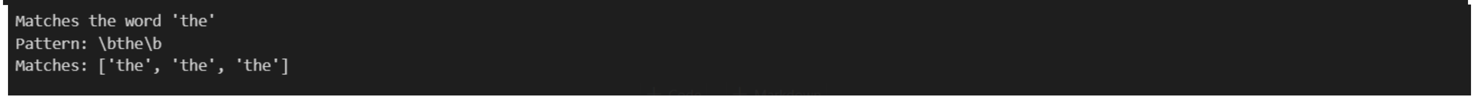
```
Matches any single character followed by '1'
Pattern: .1
Matches: ['a1']
```

```
# 7. Pipe (|) symbol
demonstrate_regex(r'dog|hat', "Matches 'cat' or 'dog'")
```



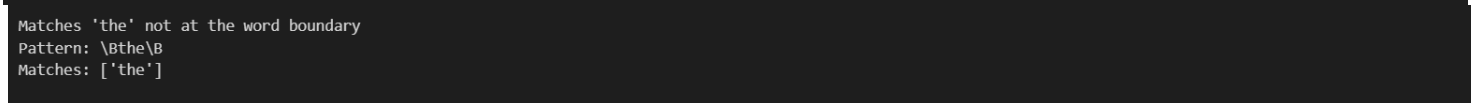
```
Matches 'cat' or 'dog'
Pattern: dog|hat
Matches: ['dog', 'dog', 'hat']
```

```
# 8. Word boundary (\b)
demonstrate_regex(r'\bthe\b', "Matches the word 'the'")
```



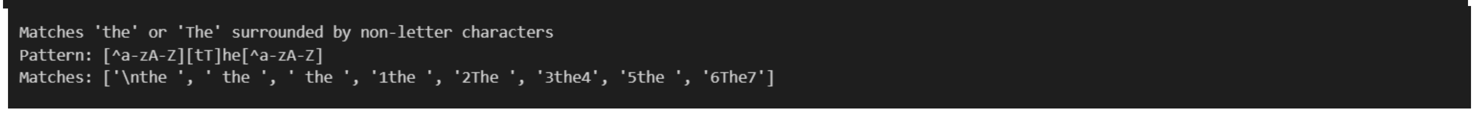
```
Matches the word 'the'
Pattern: \bthe\b
Matches: ['the', 'the', 'the']
```

```
# 9. Non-word boundary (\B)
demonstrate_regex(r'\Bthe\B', "Matches 'the' not at the word boundary")
```



```
Matches 'the' not at the word boundary
Pattern: \Bthe\B
Matches: ['the']
```

```
# 10. Regex /[^a-zA-Z][tT]he[^a-zA-Z]/
demonstrate_regex(r'[^a-zA-Z][tT]he[^a-zA-Z]', "Matches 'the' or 'The' surrounded by non-letter characters")
```



```
Matches 'the' or 'The' surrounded by non-letter characters
Pattern: [^a-zA-Z][tT]he[^a-zA-Z]
Matches: ['\nthe ', ' the ', ' the ', '1the ', '2The ', '3the4', '5the ', '6The7']
```