Lab Observation Report: NLP Lab - L1

Date: 20/06/2025

Lab No: R8001\_L1

**Q1) R8001\_L1\_Installing\_NLTK.py**  
Question: Install and use NLTK package to analyze texts from built-in corpora.

**Draft Plan**

**Program Description:**

This program demonstrates how to install and use the NLTK library to analyze various English texts. It includes computing concordances, finding similar words, analyzing lexical richness, and visualizing word dispersion.

**Program Logic:**

Libraries: nltk, matplotlib.pyplot  
Concepts: Tokenization, concordance, similarity, lexical diversity, indexing  
Uses corpora: text1 to text6 from nltk.book  
Displays plots and statistics

# Code and Outputs

Code:

import nltk  
nltk.download('punkt\_tab')  
from nltk.book import \*  
import matplotlib  
matplotlib.use('TkAgg')  
import matplotlib.pyplot as plt

Output:

[nltk\_data] Downloading package punkt\_tab to  
[nltk\_data] C:\Users\joans\AppData\Roaming\nltk\_data...  
[nltk\_data] Package punkt\_tab is already up-to-date!

Output:

\*\*\* 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

Code:

# 1.Computing with Language: Texts and Words  
# Concordances  
print("\n--- text1: concordance for 'monstrous' ---")  
text1.concordance("monstrous")

Output:

--- text1: concordance for 'monstrous' ---  
Displaying 11 of 11 matches:  
ong the former , one was of a most monstrous size . ... This came towards us ,   
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r  
ll over with a heathenish array of monstrous clubs and spears . Some were thick  
d as you gazed , and wondered what monstrous cannibal and savage could ever hav  
that has survived the flood ; most monstrous and most mountainous ! That Himmal  
they might scout at Moby Dick as a monstrous fable , or still worse and more de  
th of Radney .'" CHAPTER 55 Of the Monstrous Pictures of Whales . I shall ere l  
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly  
ere to enter upon those still more monstrous stories of them which are to be fo  
ght have been rummaged out of this monstrous cabinet there is no telling . But   
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u

Code:

print("\n--- text2: concordance for 'sensibilty' (misspelled) ---")  
text2.concordance("sensibilty")

Output:

--- text2: concordance for 'sensibilty' (misspelled) ---  
no matches

Code:

print("\n--- text2: concordance for 'sensibility' ---")  
text2.concordance("sensibility")

Output:

--- text2: concordance for 'sensibility' ---  
Displaying 10 of 10 matches:  
 [ Sense and Sensibility by Jane Austen 1811 ] CHAPTER 1 T  
rn , the excess of her sister ' s sensibility ; but by Mrs . Dashwood it was va  
, to hear him read with so little sensibility . Mama , the more I know of the w  
erable ; and he read with all the sensibility and spirit which Edward had unfor  
the ridicule so justly annexed to sensibility . Elinor was obliged , though unw  
 at consolation from either . Her sensibility was potent enough ! When breakfas  
her on the delicacies of a strong sensibility , and the graces of a polished ma  
 a strong impulse of affectionate sensibility , she moved after a moment , to h  
proof of her ' s , but by his own sensibility . " We may treat it as a joke ,"   
oor Fanny had suffered agonies of sensibility -- and he considered the existenc

Code:

print("\n--- text3: concordance for 'lived' ---")  
text3.concordance("lived")

Output:

--- text3: concordance for 'lived' ---  
Displaying 25 of 38 matches:  
ay when they were created . And Adam lived an hundred and thirty years , and be  
ughters : And all the days that Adam lived were nine hundred and thirty yea and  
y years , and begat Peleg : And Eber lived after he begat Peleg four hundred an

Code:

print("\n--- text4: concordance for 'nation' ---")  
text4.concordance("nation")

Output:

--- text4: concordance for 'nation' ---  
Displaying 25 of 348 matches:  
 to the character of an independent nation seems to have been distinguished by  
f Heaven can never be expected on a nation that disregards the eternal rules o  
 great agricultural interest of the nation prospers under its protection . Loc  
ak our Union , and demolish us as a nation . Our distance from Europe and the

Code:

print("\n--- text4: concordance for 'terror' ---")  
text4.concordance("terror")

Output:

--- text4: concordance for 'terror' ---  
Displaying 8 of 8 matches:  
menaces , by fraud or violence , by terror , intrigue , or venality , the Gove  
ameless , unreasoning , unjustified terror which paralyzes needed efforts to c  
d maintain a strong defense against terror and destruction . Our children will  
k to advance their aims by inducing terror and slaughtering innocents , we say

Code:

print("\n--- text4: concordance for 'god' ---")  
text4.concordance("god")

Output:

--- text4: concordance for 'god' ---  
Displaying 25 of 116 matches:  
eliance on the protection of Almighty God , I shall forthwith commence the duti  
 citizens and the aid of the Almighty God in the discharge of my responsible du  
our heartstrings like some air out of God ' s own presence , where justice and   
 forward - looking men , to my side . God helping me , I will not fail them , i

Code:

print("\n--- text5: concordance for 'im' ---")  
text5.concordance("im")

Output:

--- text5: concordance for 'im' ---  
Displaying 25 of 149 matches:  
 now im left with this gay name :P PART hey e  
what did you but on e-bay i feel like im in the wrong room yeee haw U30 im con  
ike im in the wrong room yeee haw U30 im considering changing my nickname to "  
oeer is sum1 gonna ghet fuked up ? :) im always hungry yeah U45 .. i believe i  
without first asking permission . U35 im sorry U35 i tried to refrain me too U

Code:

print("\n--- text5: concordance for 'ur' ---")  
text5.concordance("ur")

Output:

--- text5: concordance for 'ur' ---  
Displaying 21 of 21 matches:  
k up PART no i dont want daughters !! ur annoying . ACTION Now Playing - Cradl  
II Men scorpions rock ... lol what is ur job me too U11 hehe went to manhattan  
JOIN PART JOIN JOIN yea guitar rocker ur kool u lil guitar rocker PART whats e

Code:

print("\n--- text5: concordance for 'lol' ---")  
text5.concordance("lol")

Output:

--- text5: concordance for 'lol' ---  
Displaying 25 of 822 matches:  
ast PART 24 / m boo . 26 / m and sexy lol U115 boo . JOIN PART he drew a girl w  
 is 1.99 / min . lol @ innocent hahah lol .... yeah LOLOLOLLL U12 thats not nic  
s . lmao no U115 Check my record . :) Lol lick em U7 U23 how old r u lol Way to

Code:

# Similar words  
print("\n--- text1: similar to 'monstrous' ---")  
text1.similar("monstrous")  
  
print("\n--- text2: similar to 'monstrous' ---")  
text2.similar("monstrous")  
  
print("\n--- text4: similar to 'god' ---")  
text4.similar("god")

Output:

--- text1: similar to 'monstrous' ---  
true contemptible christian abundant few part mean careful puzzled  
mystifying passing curious loving wise doleful gamesome singular  
delightfully perilous fearless  
  
--- text2: similar to 'monstrous' ---  
very so exceedingly heartily a as good great extremely remarkably  
sweet vast amazingly  
  
--- text4: similar to 'god' ---  
america freedom us war which power it all liberty justice congress  
peace law man life people government you opinion me

Code:

# Common contexts  
print("\n--- text2: common contexts for 'monstrous' and 'very' ---")  
text2.common\_contexts(["monstrous", "very"])  
print("\n--- text4: common contexts for 'god' and 'war' ---")  
text4.common\_contexts(["god", "war"])

Output:

--- text2: common contexts for 'monstrous' and 'very' ---  
am\_glad a\_pretty a\_lucky is\_pretty be\_glad

--- text4: common contexts for 'god' and 'war' ---  
of\_and the\_of of\_we of\_s of\_in of\_may of\_is the\_who of\_they of\_but

Code:

# Dispersion plot  
print("\n--- text4: dispersion plot for political words ---")  
text4.dispersion\_plot(["citizens", "democracy", "freedom", "duties", "America"])  
plt.show()

Output:

--- text4: dispersion plot for political words ---

A graph with blue and white lines

AI-generated content may be incorrect.

Code:

print("\n--- Generating random text from text3 ---")  
text3.generate()  
  
print("\n--- number of characters in text3: ---\n", len(text3))  
  
print("\n--- number of vocabulary items of text3: ---\n",sorted(set(text3)),"\n\n --- and length of it is ---\n",len(sorted(set(text3))))  
  
print("\n --- Lexical Richness of Text3 --- \n",len(set(text3)) / len(text3))  
  
print("\n --- how many time these words come in their respective texts ---\n","the word 'smote' comes in text3 :",text3.count("smote"),  
 "times","\n","with the percentage of",100 \* text3.count('smote') / len(text3),"\n\n",  
 "the word 'a' comes in text4 :",text4.count("a"),"times","\n","with the percentage of",100 \* text4.count('a') / len(text4),"\n\n",  
 "the word 'lol comes in text5 :",text5.count("lol"),"times","\n","with the percentage of",100 \* text5.count('lol') / len(text5))

Output:

--- Generating random text from text3 ---

Building ngram index...

laid by her , and said unto Cain , Where art thou , and said , Go to ,  
for as a prince hast thou found of all the cattle in the valley , and  
the wo The  
  
--- number of characters in text3: ---  
 44764  
  
--- number of vocabulary items of text3: ---  
 ['!', "'", '(', ')', ',', ',)', '.', '.)', ':', ';', ';)', '?', '?)', 'A', 'Abel', 'Abelmizraim', 'Abidah', 'Abide', 'Abimael', 'Abimelech', 'Abr', 'Abrah', 'Abraham', 'Abram', 'Accad', 'Achbor', 'Adah', 'Adam', 'younge', 'younger', 'youngest', 'your', 'yourselves', 'youth']   
  
 --- and length of it is ---  
 2789  
  
 --- Lexical Richness of Text3 ---   
 0.06230453042623537

--- how many time these words come in their respective texts ---  
 the word 'smote' comes in text3 : 5 times   
 with the percentage of 0.01116968992940756   
  
 the word 'a' comes in text4 : 2277 times   
 with the percentage of 1.4569256756756757   
  
 the word 'lol comes in text5 : 704 times   
 with the percentage of 1.5640968673628082

Code:

def lexical\_diversity(text):  
 return len(set(text)) / len(text)  
  
def percentage(count,total):  
 return (count / total) \* 100  
  
print("\n --- lexical diversity of text2 ---\n",lexical\_diversity(text2))  
print("\n --- lexical diversity of text6 ---\n",lexical\_diversity(text6))  
  
print("\n percntage example :",percentage(3,27))  
print(" percentage of 'money' in text3 :",percentage(text3.count('money'),len(text3)))

Output:

--- lexical diversity of text2 ---  
 0.04826383002768831  
  
 --- lexical diversity of text6 ---  
 0.1276595744680851  
  
 percntage example : 11.11111111111111  
 percentage of 'money' in text3 : 0.07148601554820838

Code:

# 2) A Closer Look at Python: Texts as Lists of Words  
#Lists  
sent1 = ['Call', 'me', 'Ishmael', '.']  
print("\n sent1",sent1)  
print("length of sent1",len(sent1))  
print("lexical diversity of sent1",lexical\_diversity(sent1))  
  
persons = ['albin', 'shawn', 'bennison', 'antony','albin']  
print("\n sent1",persons)  
print("length of sent1",len(persons))  
print("lexical diversity of sent1",lexical\_diversity(persons))  
  
print("\naddition in lists",sent1 + persons)  
print("\naddition in lists",['Monty', 'Python'] + ['and', 'the', 'Holy', 'Grail'])  
  
persons.append("joan")  
print("\nappend in list of persons",persons)

Output:

sent1 ['Call', 'me', 'Ishmael', '.']  
length of sent1 4  
lexical diversity of sent1 1.0  
  
 sent1 ['albin', 'shawn', 'bennison', 'antony', 'albin']  
length of sent1 5  
lexical diversity of sent1 0.8  
  
addition in lists ['Call', 'me', 'Ishmael', '.', 'albin', 'shawn', 'bennison', 'antony', 'albin']  
  
addition in lists ['Monty', 'Python', 'and', 'the', 'Holy', 'Grail']  
  
append in list of persons ['albin', 'shawn', 'bennison', 'antony', 'albin', 'joan']

Code:

#Indexing List  
print("\n173rd word in text4: ",text4[173])  
print("\n100th word in text5: ",text5[100])  
  
print("\n the word Egyptian's index number in text3 is",text3.index('Egyptian'))  
  
sent = ['word1', 'word2', 'word3', 'word4', 'word5','word6', 'word7', 'word8', 'word9', 'word10']  
print(sent)  
print("indexing will always start from 0 so sent[0] will be ",sent[0])  
  
print("\n the words between 50-55 index numbers in text2",text2[50:56])  
  
sent[0] = 'First'  
sent[9] = 'Last'  
print("\n length of sent",len(sent),"\n sent:",sent)  
sent[1:9] = ['Second', 'Third']  
print("\n updated sent",sent)

Output:

173rd word in text4: awaken  
  
100th word in text5: my  
 the word Egyptian's index number in text3 is 10245  
['word1', 'word2', 'word3', 'word4', 'word5', 'word6', 'word7', 'word8', 'word9', 'word10']  
indexing will always start from 0 so sent[0] will be word1  
 the words between 50-55 index numbers in text2 ['lived', 'in', 'so', 'respectable', 'a', 'manner']  
 length of sent 10   
 sent: ['First', 'word2', 'word3', 'word4', 'word5', 'word6', 'word7', 'word8', 'word9', 'Last']  
 updated sent ['First', 'Second', 'Third', 'Last']

Code:

#variables  
my\_sent = ['Bravely', 'bold', 'Sir', 'Robin', ',',  
 'rode','forth', 'from', 'Camelot', '.']  
noun\_phrase = my\_sent[:4]  
print("\n noun phrase",noun\_phrase)  
print("\n sorted noun phrase",sorted(noun\_phrase))  
  
vocab = set(text1)  
vocab\_size = len(vocab)  
print("vocabulary size of text1",vocab\_size)

Output:

noun phrase ['Bravely', 'bold', 'Sir', 'Robin']  
 sorted noun phrase ['Bravely', 'Robin', 'Sir', 'bold']  
vocabulary size of text1 19317

Code:

#strings  
name = "joker"  
print("\n name",name)  
print("\n name[0] is",name[0])  
print("\n name[:4] is",name[:4])  
print("\n after adding ! :\n",name + '!')  
print("\n Multiplicating itself : \n",name \* 3)  
  
my\_name=['joan','sandeep','larson']  
full\_name=' '.join(my\_name)  
print("\n full name:\n",full\_name)  
  
splitted\_name=full\_name.split()  
print("\n splitted name:\n",splitted\_name)

Output:

name joker  
 name[0] is j  
 name[:4] is joke  
 after adding ! :  
 joker!  
 Multiplicating itself :   
 jokerjokerjoker  
 full name:  
 joan sandeep larson  
 splitted name:  
 ['joan', 'sandeep', 'larson']

**Q2) R8001\_L1\_TextProcessing.py**  
  
Question: Preprocess and analyze a paragraph using Python.

**Draft Plan**

**Program Description:**

Processes a hardcoded paragraph by cleaning, tokenizing, and analyzing its word statistics including frequency and longest word.

**Program Logic:**

Libraries: string, collections. Counter  
Steps: Lowercase conversion, Remove punctuation, Tokenize, Count stats and frequency analysis

Code and Outputs:

Code:

import string  
from collections import Counter  
# 1) Define a string containing a paragraph as the value  
paragraph = """  
Natural Language Processing (NLP) is a subfield of artificial intelligence that deals with the interaction   
between computers and humans through natural language. It enables machines to understand, interpret, and   
generate human language in a valuable way.  
"""  
cleaned\_text = paragraph.lower().translate(str.maketrans('', '', string.punctuation))  
# Tokenize the paragraph into words  
words = cleaned\_text.split()

# 2) Count total words and total unique words  
total\_words = len(words)  
unique\_words = len(set(words))

# 3) Calculate word frequency  
word\_freq = Counter(words)  
# Get the most frequent word  
most\_frequent = word\_freq.most\_common(1)[0]  
# Get the least frequent word  
least\_frequent = min(word\_freq.items(), key=lambda item: item[1])  
# 4) Find the longest word  
longest\_word = max(words, key=len)print("Total number of words:", total\_words)  
print("Total number of unique words:", unique\_words)  
print("\nWord Frequencies:")  
for word, freq in word\_freq.items():  
 print(f"{word}: {freq}")  
print("\nMost Frequent Word:", most\_frequent)  
print("Least Frequent Word:", least\_frequent)  
print("Longest Word:", longest\_word)

Output:

Total number of words: 36  
Total number of unique words: 31  
Word Frequencies:  
natural: 2  
language: 3  
processing: 1  
nlp: 1  
is: 1  
a: 2  
subfield: 1  
of: 1  
artificial: 1  
intelligence: 1  
that: 1  
deals: 1  
with: 1  
the: 1  
interaction: 1  
between: 1enables: 1  
machines: 1  
to: 1  
understand: 1  
interpret: 1  
generate: 1  
human: 1  
in: 1  
valuable: 1  
way: 1  
Most Frequent Word: ('language', 3)  
Least Frequent Word: ('processing', 1)  
Longest Word: intelligence

**Q3) R8001\_L1\_RegularExpression.py**

Question: Explore regular expression features in Python.

**Draft Plan**

**Program Description:**

This program demonstrates the usage of regular expressions to search and extract patterns from text, including character classes, grouping, disjunction, anchors, and quantifiers.

**Program Logic:**

Library: re  
Functions: re.findall(), re.search()  
Covers: Character matching, Grouping, Anchors, Quantifiers

Code and Outputs

Code:

# Basic Regulsr Expressions  
import re  
text="The simplest kind of regular expression is a sequence of Simple characters, then 1 by 1 we will learn"  
match=re.findall(r"simple",text)  
print("\n")  
print(match)

Output:

['simple']

Code:

match=re.findall("[Ss]imple",text)  
print("\n")  
print(match)  
print("\n")

Output:

['simple', 'Simple']

Code:

match=re.findall("[A-Z]",text)  
print(match)  
print("\n")

Output:

['T', 'S']

Code:

match=re.findall("[0-9]",text)  
print(match)  
print("\n")

Output:

['1', '1']

Code:

match=re.findall("[^a-z]",text)  
print(match)  
print("\n")

Output:

['T', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'S', ' ', ',', ' ', ' ', ' ', '1', ' ', ' ', '1', ' ', ' ', ' ']

Code:

match=re.findall("[Tt]hen?",text)  
print(match)  
print("\n")

Output:

['The', 'then']

Code:

match=re.findall("a\*",text)  
print(match)  
print("\n")

Output:

['', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', 'a', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', 'a', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', 'a', '', 'a', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', 'a', '', '', '']

Code:

match=re.findall("then+",text)  
print(match)  
print("\n")

Output:

['then']

Code:

match=re.findall("aa\*",text)  
print(match)  
print("\n")

Output:

['a', 'a', 'a', 'a', 'a']

Code:

match=re.findall("w.",text)  
print(match)  
print("\n")

Output:

['we', 'wi']

Code:

match=re.findall("^n",text)  
print(match)  
print("\n")

Output:

[]

Code:

match=re.findall("[Tt]$",text)  
print(match)  
print("\n")

Output:

[]

Code:

#Disjunction, Grouping, and Precedence  
text= "in this we will see the use of disjunction, grouping and precedence"  
match=re.findall("in|this",text)  
print(match)  
print("\n")

Output:

['in', 'this', 'in']

Code:

match=re.findall("th(is|e)",text)  
print(match)  
print("\n")

Output:

['is', 'e']

Code:

#A Simple Example  
text="In The jungle a quick brown fox jumps into the bush"  
match=re.findall("in",text)  
print(match)  
print("\n")

Output:

['in']

Code:

match=re.findall("[Ii]n",text)  
print(match)  
print("\n")

Output:

['In', 'in']

Code:

match=re.findall("\b[Ii]n\b",text)  
print(match)  
print("\n")

Output:

[]

Code:

match=re.findall(r"\b[Ii]n\b",text)  
print(match)  
print("\n")

Output:

['In']

Code:

match=re.findall("(?:^|[^a-zA-Z])([Ii]n)(?=[^a-zA-Z]|$)",text)  
print(match)  
print("\n")

Output:

['In']

Code:

#A More Complex Example  
text="any PC with more than 500 MHz and 32 Gb of disk space for less than $1000.5. soso sososo"  
match=re.findall(r"\$[0-9]+",text)  
print(match)  
print("\n")

Output:

['$1000']

Code:

match=re.findall(r"\$[0-9]+(?:\.[0-9]+)?",text)  
print(match)  
print("\n")

Output:

['$1000.5']

Code:

match=re.findall(r"\b[0-9]+ \*(MHz|[Mm]egahertz|GHz|[Gg]igahertz)\b",text)  
print(match)  
print("\n")

Output:

['MHz']

Code:

match=re.findall("\W",text)  
print(match)  
print("\n")

Output:

[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '$', '.', '.', ' ', ' ']

Code:

match=re.findall("(?:so){3}",text)  
print(match)  
print("\n")

Output:

['sososo']

Code:

text1 = "The bigger they were, the bigger they will be"  
text2 = "The bigger they were, the faster they will be"  
pattern = r"the (.\*)er they were, the \1er they will be"  
# Using re.IGNORECASE so "The" matches "the"  
match1 = re.search(pattern, text1, re.IGNORECASE)  
match2 = re.search(pattern, text2, re.IGNORECASE)  
print("Match 1:", bool(match1))  
print("Match 2:", bool(match2))

Output:

Match 1: True  
Match 2: False

**R8001\_L1\_Practice\_Q1.py**  
Question: Tokenize and analyze a chapter file using NLTK.

**Draft Plan**

**Program Description:**

Tokenizes text from a file, converts it to lowercase, calculates word frequencies, and visualizes the most frequent words.

**Program Logic:**

Libraries: nltk, FreqDist, matplotlib.pyplot  
Reads file chapter.txt  
Tokenizes and normalizes text  
Calculates and plots frequency distribution

Code and Outputs:

Code:

import nltk  
from nltk.tokenize import word\_tokenize  
from nltk import FreqDist  
import matplotlib.pyplot as plt

Code:

nltk.download('punkt')

Output:

[nltk\_data] Downloading package punkt to  
[nltk\_data] C:\Users\joans\AppData\Roaming\nltk\_data...  
[nltk\_data] Package punkt is already up-to-date!

Code:

# Read text file  
with open('chapter.txt', 'r', encoding='utf-8') as f:  
 text = f.read()  
# Normalize: lowercase and tokenize  
tokens = word\_tokenize(text.lower())  
# Frequency distribution using nltk  
freq\_dist = FreqDist(tokens)  
# Print top 10 frequent words  
print(freq\_dist.most\_common(10))

Output:

[(',', 19360), ('the', 14608), ('.', 6988), ('of', 6715), ('and', 6447), ('a', 4700), ('to', 4662), ('in', 4212), (';', 4182), ('that', 3017)]

Code:

# Plot frequency distribution  
freq\_dist.plot(30, cumulative=False)  
plt.show()

Output:

A graph with a line

AI-generated content may be incorrect.

**R8001\_L1\_Practice\_Q2.py**  
Question: Analyze word patterns and statistics using NLTK's webtext corpus.

**Draft Plan**

**Program Description:**

The program examines specific patterns like suffixes, infixes, and capitalizations from webtext corpus. It also calculates average word length, vocabulary size, and percentage frequency.

**Program Logic:**

Corpus: webtext.words('firefox.txt')  
Functions: .endswith(), .startswith(), string operations  
Calculations: average word length, vocab size, frequency percentage

Code and Outputs:

Code:

from nltk.corpus import webtext

text6 = webtext.words('firefox.txt')  # Example file from webtext corpus

Output:

[nltk\_data] Downloading package webtext to

[nltk\_data] [C:\Users\joans\AppData\Roaming\nltk\_data...](file:///C:\Users\joans\AppData\Roaming\nltk_data...)

[nltk\_data] Package webtext is already up-to-date!

Code:

#24

# a. Words ending in 'ize'

words\_ending\_ize = [w for w in text6 if w.endswith('ize')]

print(words\_ending\_ize[:10])

# b. Words containing 'z'

words\_contain\_z = [w for w in text6 if 'z' in w]

print(words\_contain\_z[:10])

# c. Words containing 'pt'

words\_contain\_pt = [w for w in text6 if 'pt' in w]

print(words\_contain\_pt[:10])

# d. Words with all lowercase except initial capital (titlecase)

titlecase\_words = [w for w in text6 if w.istitle()]

print(titlecase\_words[:10])

Output:

['customize', 'Size', 'customize', 'customize', 'resize', 'Customize', 'minimize', 'minimize', 'size', 'customize']

['customization', 'customize', 'mozbrowser', 'Size', 'customize', 'Mozilla', 'Mozilla', 'Customizing', 'Mozilla', 'mozilla']

['script', 'attempting', 'ptoolbar', 'options', 'consumption', 'Options', 'pt', 'Sept', 'exception', 'empty']

['Cookie', 'Manager', 'Don', 'When', 'Pressing', 'Ctrl', 'N', 'So', 'Implement', 'Cocoa']

Code:

#25

sent = ['she', 'sells', 'sea', 'shells', 'by', 'the', 'sea', 'shore']

# a. Words beginning with 'sh'

print([w for w in sent if w.startswith('sh')])

# b. Words longer than 4 characters

print([w for w in sent if len(w) > 4])

Output:

['she', 'shells', 'shore']

['sells', 'shells', 'shore']

Code:

#26

import nltk

from nltk.book import text1  # 'Moby Dick' text

total\_chars = sum(len(w) for w in text1)

total\_words = len(text1)

average\_word\_length = total\_chars / total\_words

print(f"Average word length: {average\_word\_length:.2f}")

Output:

Average word length: 3.83

Code:

#27

def vocab\_size(text):

    return len(set(text))

print("Vocabulary size of text1:", vocab\_size(text1))

Output:

Vocabulary size of text1: 19317

Code:

#28

def percent(word, text):

    count = text.count(word)

    total = len(text)

    return 100 \* count / total if total > 0 else 0

print(f"Percentage of 'whale' in text1: {percent('whale', text1):.2f}%")

Output:

Percentage of 'whale' in text1: 0.35%