1. Write a Python program to count the total number of words in a given sentence.

Answer:

```
text = "  I    am    a     Man   "  # Input string has multiple spaces
for worst case scenario
words = text.split()
"""Here, str.split() return a list of words by splitting the string at
whitespace
and split() without any arguments count single space and ignore
multiple spaces as well as before and after spaces."""
total_words = len(words) # Count the number of words in the list as
element
print(total_words)
```

2. Write a Python program to count the frequency of each character in a given string.

Answer:

```
text = "I am a man"
fre_cnt = dict()  # Initialize an empty dictionary to store frequency
counts
for char in text:
    if char == " ":
        continue  # Skip spaces in the text
    elif char in fre_cnt:  # check if the character is already in the
dictionary as key
        fre_cnt[char] += 1  # Increment the value of key by 1 if it
exists
    else:
        fre_cnt[char] = 1  # Initialize the value of key to 1 if it
does not exist as well as it automatically adding to the dictionary as
key
print(fre_cnt)
```

3. Write a Python program to count how many vowels and consonants are present in a Sentence.

Answer:

```
text = "Hello, how are you doing today? I hope you're having a great
day!"
cnt_vow = 0
cnt_con = 0
vowels = "aeiouAEIOU"  # define list of vowels
for char in text:
    if char in vowels:  # check if the character is a vowel
        cnt_vow += 1
    elif char.isalpha():  # check if it's a consonant, not a special
character
        cnt_con += 1
print("Number of vowels:", cnt_vow)
print("Number of consonants", cnt_con)
```

4. Write a Python program to check if a word or sentence is a palindrome, ignoring punctuation and spaces.

Answer:

```
import string
text = "I am a man"
cln_text = ''.join(char.lower() for char in text if char.isalnum()) #
remove spaces and punctuation
rev_text = cln_text[::-1]   # reverse the clean text
if cln_text == rev_text:   # check if the clean text is the same as the
reversed text
    print("The text is a palindrome.")
else:
    print("The text is not a palindrome.")
```

5. Write a Python program to count the frequency of each word in a paragraph.

Answer:

```
import re
text = "  Hello, how are you doing today? I hope you are doing well. I
am doing fine, thank you!  "
cln_text = re.findall(r'\b\w+\b', text)  # Find all complete words
without punctuation and return a list of words
fre_words = dict() # Initialize an empty dictionary to store word
frequencies
for word in cln_text:
    word = word.lower()  # convert to lowercase to ensure case
insensitivity counting
    if word in fre_words:
        fre_words[word] += 1  # Increment the value for the word if it
exists
    else:
        fre_words[word] = 1  # Initialize the word with a value of 1
if it does not exist
print(fre_words)
```

6. Write a Python program to remove common English stopwords (like "is," "the," "a," and "an") from a given string.

Answer:

```
import re
text = "I am a Man.This is a test.It is an example."
stopwords = ["is", "the", "a", "an"]
for word in stopwords:
        text = re.sub(r'\b' + re.escape(word) + r'\b', '', text, flags
= re.IGNORECASE)  # remove stopwords
text = re.sub(r'\s+', ' ',text).strip()  # remove extra spaces
print(text)
```

7. Write a Python program to count the number of unique words in a sentence.

Answer:

```
import re
text = "  Hello, how are you doing today? I hope you are doing well. I am
doing fine, thank you!  "
cln_text = re.findall(r'\b\w+\b', text)  # Find all complete words without
punctuation
unique_words = set(word.lower() for word in cln_text)
cnt_unique = len(unique_words)
print(f"Number of unique words: {cnt_unique}")
```

8. Write a Python program to replace a specific word in a sentence with another word.
Input: A sentence, target word, and replacement word.
Output: Modified sentence with the replacement.

Answer:

```
import re  # Importing the regular expression module
text = "I am a student.I am studying Python.I love programming."
word = "I"  # Specific word to replace
new_text = re.sub(r'\b' + re.escape(word)  + r'\b', 'You', text, flags
= re.IGNORECASE)  # Replacing the word "I" with "You" and ignoring
case sensitivity
print(new_text)
```

9. Write a Python program to find the most frequent word in a sentence or paragraph.

Answer:

```
import re  # Importing the regular expression module
from collections import Counter  # Importing counter as a dict()
subclass for counting frequency of words
text = "I am a student.I am studying Python.I love programming."
words = re.findall(r'\b\w+\b', text)  # Finding all words in the text
while ignoring punctuation
word_count = Counter(words)  # Counting the frequency of each word
from words list
sorted_word_count = sorted(word_count.items(), key=lambda item:
item[1])  # Sorting the word_count dictionary by frequency
print(sorted_word_count[-1] [0])  # Printing the most frequent word
```

10. Write a Python program to convert a sentence into title case format (each word's first letter capitalized).

Answer:

```
text = "this is a test string for testing purposes"
title_text = text.title()  # Use title() to capitalize the first of each
word
print(title_text)
```

Theory Question:

1. What is a String in Programming?

Answer:

A string in Programming is a sequence of Characters. These characters can be:

Letter: (A-Z,a-z)
Digits: (0-9)
Special characters :(such as , . , \, / , ?, !, #, _ , - spaces etc.)
Strings are usually enclosed in quotation marks (“ or ‘) .Strings are commonly used to store text data in most programming languages. Example:”He is 34 years old.”

2. Difference Between String, Word, and Sentence in Text Processing

Answer:

Aspect	String	Word	Sentence
Definition	A sequence of characters (letters, digits, symbols). It may or may not form a meaningful word or sentence.	A meaningful sequence of characters separated by spaces or punctuation	A set of words arranged in a meaningful way, usually end with punctuation (. , ? , ! etc)
Example	"Python is a Programming language" entire string is a sequence of characters	"Python" , "is', "a", "Programming", "language" each represent a word	"Python is a Programming language." meaningfully arranged words, represent a sentence

3. What is String Immutability, and Why Does It Matter?

Answer:
String immutability means that once a string is created, it cannot be changed or modified.This is a special feature of strings in Python.A specific character of a string

cannot be changed directly.Instead, creating a new string if any modification is needed.
Example:

```
text = "I am a Man"
text[0] = "i"  # This will raise an error because strings are
immutable in Python.
```

Matters:
Safety:Immutable strings prevent accidental changes and make the code more predictable as well as reliable.
Performance: Since strings are immutable that's why it optimize memory , speeds up the code and increases reusability.

4. What Are Common String Operations Used in NLP?

Answer:
String operations commonly used in NLP are:
Tokenization: "I Love NLP" => "I", "Love", "NLP" (Breaking text into smaller parts)
Matching/Searching: Used functions like find(), regex etc.Find specific words or patterns.
Lowercasing: "Python" => "python".
Removing Punctuation: . / , ? ! etc.
Remove stopwords: "is", "an", "a", "are", "the" , "and" etc.
Stemming: "Running" => "run" (convert words to their roots)
Vectorization: Converting character into number.
Regular expression
Text normalization
etc.

5. What Is Tokenization, and How Is It Related to Strings?

Answer:
Tokenization is a string operation used in natural language processing (nlp), used for breaking text (string) into smaller parts, called tokens.These tokens can be words,

sentences, or subwords.
Example:
“I Love NLP” = > “I” , “Love”, “NLP”
Relation:
Tokenization directly works on string, which is a sequence of characters. It splits the string into tokens so that a machine can easily process and understand a string very fast.