# String Operations (Continued)

# 7. Remove leading, trailing, and extra spaces in a string

def remove\_extra\_spaces(s):

s = s.strip()

s = " ".join(s.split())

return s

s = "hello bhaskar sai ram"

print(f"String after removing extra spaces: \"{remove\_extra\_spaces(s)}\"")

# 8. Count the number of words in a given string

def count\_words(s):

s = s.strip()

words = s.split()

return len(words)

s = "bhaskar sai ram"

print(f"Number of words in the given string: {count\_words(s)}")

# 9. Count the number of letters in a given string

def count\_letters(s):

return sum(1 for char in s if char.isalpha())

s = "bhaskar"

print(f"Number of letters in the given string: {count\_letters(s)}")

# 10. Print the min and max frequency elements in an array

def min\_max\_frequency(arr):

from collections import Counter

counts = Counter(arr)

max\_freq = max(counts.values())

min\_freq = min(counts.values())

print(f"Max frequency: {max\_freq}")

print(f"Min frequency: {min\_freq}")

arr = [1, 2, 3, 2, 3, 2, 2, 2, 2]

min\_max\_frequency(arr)

# 11. Check if there are two or three consecutive identical characters in a string

def check\_consecutive(s):

if not s:

return "String can't be defined"

for i in range(len(s) - 1):

if s[i] == s[i + 1]:

return f"Two consecutive characters are found: {s[i]}"

return "No two consecutive characters are found"

print(check\_consecutive("abaab"))

print(check\_consecutive("acdba"))

# 12. Find the first occurrence of a substring (similar to LeetCode 28)

def find\_first\_occurrence(haystack, needle):

if not needle:

return 0

for i in range(len(haystack) - len(needle) + 1):

if haystack[i:i + len(needle)] == needle:

return i

return -1

haystack = "sadbutsad"

needle = "sad"

print(find\_first\_occurrence(haystack, needle))

# 13. Find the first and last index of occurrence for each character in a string

def first\_and\_last\_indices(s):

if not s:

return "String can't be defined"

index\_map = {}

for i, char in enumerate(s):

if char in index\_map:

index\_map[char][1] = i

else:

index\_map[char] = [i, i]

return index\_map

print(first\_and\_last\_indices("bhaskar"))

# 14. Check if a string contains all letters from 'a' to 'z'

def contains\_all\_alphabet(s):

return len(set(filter(str.isalpha, s.lower()))) == 26

print(contains\_all\_alphabet("bhaskar"))

print(contains\_all\_alphabet("abcdefghijklmnopqrstuvwxyz"))

# 15. Insert an element at a specific position in a string

def insert\_at\_position(s, char, position):

if position < 0 or position > len(s):

return "Invalid position"

return s[:position] + char + s[position:]

print(insert\_at\_position("bhaskar", "s", 4))

# 16. Insert an element at predefined positions in a string

def insert\_at\_predefined\_positions(s, char, pos1, pos2, pos3):

if not all(0 <= pos <= len(s) for pos in [pos1, pos2, pos3]):

return "Invalid position"

result = s

for pos in sorted([pos1, pos2, pos3], reverse=True):

result = result[:pos] + char + result[pos:]

return result

print(insert\_at\_predefined\_positions("bhaskar", "s", 0, 6, 3))

# 17. Insert at the first, last, and kth position in a string

def insert\_at\_first\_last\_k(s, char, k):

if k < 0 or k >= len(s):

return "Invalid Kth position."

result = char + s + char

return result[:k + 1] + char + result[k + 1:]

print(insert\_at\_first\_last\_k("bhaskar", "s", 3))

# 18. Remove the first, last, and kth character from a string

def remove\_first\_last\_k(s, k):

if not s or len(s) < 3 or not (1 <= k < len(s) - 1):

return "String too short or invalid position for K!"

return s[1:k] + s[k + 1:-1]

print(remove\_first\_last\_k("bhaskar", 2))

# 19. Find a specific substring within a string

def specific\_substring(s, k):

if k < 0 or k >= len(s):

return "Substring is not possible"

return s[:k + 1]

print(specific\_substring("bhaskar", 2))