**1. Write a Python program to reverse a string without using any built-in string reversal functions.**

def reverse\_string(string):

reversed\_string = ''

for i in range(len(string)-1, -1, -1):

reversed\_string += string[i]

return reversed\_string

input\_string = input("Enter a string: ")

reversed\_string = reverse\_string(input\_string)

print("Reversed string:", reversed\_string)

**2. Implement a function to check if a given string is a palindrome.**

def is\_palindrome(string):

string = string.lower()

left = 0

right = len(string) - 1

while left < right:

if string[left] != string[right]:

return False

left += 1

right -= 1

return True

input\_string = input("Enter a string: ")

if is\_palindrome(input\_string):

print("The string is a palindrome.")

else:

print("The string is not a palindrome.")

**3. Write a program to find the largest element in a given list.**

def find\_largest\_element(lst):

if len(lst) == 0:

raise ValueError("The list is empty.")

largest = lst[0]

for i in range(1, len(lst)):

if lst[i] > largest:

largest = lst[i]

return largest

input\_list = input("Enter a list of numbers separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

largest\_element = find\_largest\_element(input\_list)

print("The largest element in the list is:", largest\_element)

**4. Implement a function to count the occurrence of each element in a list.**

def count\_occurrences(lst):

occurrence\_count = {}

for element in lst:

if element in occurrence\_count:

occurrence\_count[element] += 1

else:

occurrence\_count[element] = 1

return occurrence\_count

input\_list = input("Enter a list of elements separated by spaces: ").split()

occurrence\_dict = count\_occurrences(input\_list)

print("Element Occurrence Count:")

for element, count in occurrence\_dict.items():

print(element, ":", count)

**5. Write a Python program to find the second largest number in a list.**

def find\_second\_largest(lst):

if len(lst) < 2:

raise ValueError("The list does not have enough elements.")

largest = float('-inf')

second\_largest = float('-inf')

for num in lst:

if num > largest:

second\_largest = largest

largest = num

elif num > second\_largest and num != largest:

second\_largest = num

if second\_largest == float('-inf'):

raise ValueError("There is no second largest element in the list.")

return second\_largest

input\_list = input("Enter a list of numbers separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

second\_largest = find\_second\_largest(input\_list)

print("The second largest number in the list is:", second\_largest)

**6. Implement a function to remove duplicate elements from a list.**

def remove\_duplicates(lst):

return list(set(lst))

input\_list = input("Enter a list of elements separated by spaces: ").split()

unique\_list = remove\_duplicates(input\_list)

print("List with duplicates removed:", unique\_list)

**7. Write a program to calculate the factorial of a given number.**

def factorial(n):

if n < 0:

raise ValueError("Factorial is not defined for negative numbers.")

elif n == 0 or n == 1:

return 1

else:

result = 1

for i in range(2, n + 1):

result \*= i

return result

input\_number = int(input("Enter a number: "))

factorial\_result = factorial(input\_number)

print("Factorial of", input\_number, "is:", factorial\_result)

**8. Implement a function to check if a given number is prime.**

def is\_prime(number):

if number < 2:

return False

for i in range(2, int(number\*\*0.5) + 1):

if number % i == 0:

return False

return True

input\_number = int(input("Enter a number: "))

if is\_prime(input\_number):

print(input\_number, "is a prime number.")

else:

print(input\_number, "is not a prime number.")

**9. Write a Python program to sort a list of integers in ascending order.**

def ascending\_sort(lst):

sorted\_lst = sorted(lst)

return sorted\_lst

input\_list = input("Enter a list of integers separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

sorted\_list = ascending\_sort(input\_list)

print("Sorted list in ascending order:", sorted\_list)

**10. Implement a function to find the sum of all numbers in a list.**

def calculate\_sum(lst):

total = 0

for num in lst:

total += num

return total

input\_list = input("Enter a list of numbers separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

sum\_result = calculate\_sum(input\_list)

print("The sum of the numbers in the list is:", sum\_result)

**11. Write a program to find the common elements between two lists.**

def find\_common\_elements(list1, list2):

common\_elements = []

for element in list1:

if element in list2:

common\_elements.append(element)

return common\_elements

input\_list1 = input("Enter the elements of the first list separated by spaces: ").split()

input\_list2 = input("Enter the elements of the second list separated by spaces: ").split()

common\_elements = find\_common\_elements(input\_list1, input\_list2)

print("Common elements:", common\_elements)

**12. Implement a function to check if a given string is an anagram of another string.**

def is\_anagram(string1, string2):

string1 = string1.replace(" ", "").lower()

string2 = string2.replace(" ", "").lower()

return sorted(string1) == sorted(string2)

input\_string1 = input("Enter the first string: ")

input\_string2 = input("Enter the second string: ")

if is\_anagram(input\_string1, input\_string2):

print("The strings are anagrams.")

else:

print("The strings are not anagrams.")

**13. Write a Python program to generate all permutations of a given string.**

def generate\_permutations(string):

if len(string) == 1:

return [string]

permutations = []

for i in range(len(string)):

first\_char = string[i]

remaining\_chars = string[:i] + string[i+1:]

sub\_permutations = generate\_permutations(remaining\_chars)

for sub\_permutation in sub\_permutations:

permutations.append(first\_char + sub\_permutation)

return permutations

input\_string = input("Enter a string: ")

permutations = generate\_permutations(input\_string)

print("Permutations:")

for permutation in permutations:

print(permutation)

**14. Implement a function to calculate the Fibonacci sequence up to a given number of terms.**

def fibonacci\_sequence(n):

sequence = []

a, b = 0, 1

while len(sequence) < n:

sequence.append(a)

a, b = b, a + b

return sequence

num\_terms = int(input("Enter the number of terms for the Fibonacci sequence: "))

fibonacci\_seq = fibonacci\_sequence(num\_terms)

print("Fibonacci sequence:")

print(fibonacci\_seq)

**15. Write a program to find the median of a list of numbers.**

def find\_median(lst):

sorted\_lst = sorted(lst)

length = len(sorted\_lst)

middle\_index = length // 2

if length % 2 == 0:

# If the length is even, average the middle two numbers

median = (sorted\_lst[middle\_index - 1] + sorted\_lst[middle\_index]) / 2

else:

# If the length is odd, take the middle number

median = sorted\_lst[middle\_index]

return median

# Example usage

input\_list = input("Enter a list of numbers separated by spaces: ").split()

input\_list = [float(num) for num in input\_list]

median = find\_median(input\_list)

print("Median:", median)

**16. Implement a function to check if a given list is sorted in non-decreasing order.**

def is\_sorted(lst):

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

if lst[i] > lst[i + 1]:

return False

return True

# Example usage

input\_list = input("Enter a list of numbers separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

if is\_sorted(input\_list):

print("The list is sorted in non-decreasing order.")

else:

print("The list is not sorted in non-decreasing order.")

**17. Write a Python program to find the intersection of two lists.**

def find\_intersection(list1, list2):

intersection = []

for element in list1:

if element in list2:

intersection.append(element)

return intersection

# Example usage

input\_list1 = input("Enter the elements of the first list separated by spaces: ").split()

input\_list2 = input("Enter the elements of the second list separated by spaces: ").split()

intersection = find\_intersection(input\_list1, input\_list2)

print("Intersection:", intersection)

**18. Implement a function to find the maximum subarray sum in a given list.**

def max\_subarray\_sum(lst):

if len(lst) == 0:

return 0

current\_sum = lst[0]

max\_sum = lst[0]

for i in range(1, len(lst)):

current\_sum = max(lst[i], current\_sum + lst[i])

max\_sum = max(max\_sum, current\_sum)

return max\_sum

# Example usage

input\_list = input("Enter a list of numbers separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

max\_sum = max\_subarray\_sum(input\_list)

print("Maximum subarray sum:", max\_sum)

**19. Write a program to remove all vowels from a given string.**

def remove\_vowels(string):

vowels = 'aeiouAEIOU'

without\_vowels = ''.join([char for char in string if char not in vowels])

return without\_vowels

# Example usage

input\_string = input("Enter a string: ")

string\_without\_vowels = remove\_vowels(input\_string)

print("String without vowels:", string\_without\_vowels)

**20. Implement a function to reverse the order of words in a given sentence.**

def reverse\_sentence(sentence):

words = sentence.split()

reversed\_sentence = ' '.join(words[::-1])

return reversed\_sentence

# Example usage

input\_sentence = input("Enter a sentence: ")

reversed\_sentence = reverse\_sentence(input\_sentence)

print("Reversed sentence:", reversed\_sentence)

**21. Write a Python program to check if two strings are anagrams of each other.**

def are\_anagrams(str1, str2):

str1 = str1.replace(" ", "").lower()

str2 = str2.replace(" ", "").lower()

if len(str1) != len(str2):

return False

char\_count = {}

# Count the characters in str1

for char in str1:

if char in char\_count:

char\_count[char] += 1

else:

char\_count[char] = 1

# Decrement the character count for each character in str2

for char in str2:

if char in char\_count:

char\_count[char] -= 1

else:

return False

# Check if all character counts are zero

for count in char\_count.values():

if count != 0:

return False

return True

# Example usage

input\_str1 = input("Enter the first string: ")

input\_str2 = input("Enter the second string: ")

if are\_anagrams(input\_str1, input\_str2):

print("The strings are anagrams.")

else:

print("The strings are not anagrams.")

**22. Implement a function to find the first non-repeating character in a string.**

def find\_first\_non\_repeating\_char(string):

char\_count = {}

for char in string:

if char in char\_count:

char\_count[char] += 1

else:

char\_count[char] = 1

for char in string:

if char\_count[char] == 1:

return char

return None

# Example usage

input\_string = input("Enter a string: ")

first\_non\_repeating\_char = find\_first\_non\_repeating\_char(input\_string)

if first\_non\_repeating\_char:

print("The first non-repeating character is:", first\_non\_repeating\_char)

else:

print("There are no non-repeating characters in the string.")

**23. Write a program to find the prime factors of a given number.**

def find\_prime\_factors(number):

prime\_factors = []

divisor = 2

while divisor <= number:

if number % divisor == 0:

prime\_factors.append(divisor)

number = number // divisor

else:

divisor += 1

return prime\_factors

# Example usage

input\_number = int(input("Enter a number: "))

prime\_factors = find\_prime\_factors(input\_number)

print("Prime factors:", prime\_factors)

**24. Implement a function to check if a given number is a power of two.**

def is\_power\_of\_two(number):

if number <= 0:

return False

return (number & (number - 1)) == 0

# Example usage

input\_number = int(input("Enter a number: "))

if is\_power\_of\_two(input\_number):

print(input\_number, "is a power of two.")

else:

print(input\_number, "is not a power of two.")

**25. Write a Python program to merge two sorted lists into a single sorted list.**

def merge\_sorted\_lists(list1, list2):

merged\_list = []

i, j = 0, 0

while i < len(list1) and j < len(list2):

if list1[i] <= list2[j]:

merged\_list.append(list1[i])

i += 1

else:

merged\_list.append(list2[j])

j += 1

# Append any remaining elements from list1

while i < len(list1):

merged\_list.append(list1[i])

i += 1

# Append any remaining elements from list2

while j < len(list2):

merged\_list.append(list2[j])

j += 1

return merged\_list

# Example usage

input\_list1 = input("Enter the elements of the first sorted list separated by spaces: ").split()

input\_list2 = input("Enter the elements of the second sorted list separated by spaces: ").split()

list1 = [int(num) for num in input\_list1]

list2 = [int(num) for num in input\_list2]

merged\_list = merge\_sorted\_lists(list1, list2)

print("Merged list:", merged\_list)

**26. Implement a function to find the mode of a list of numbers.**

from collections import Counter

def find\_mode(lst):

counts = Counter(lst)

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

mode = [num for num, count in counts.items() if count == max\_count]

return mode

# Example usage

input\_list = input("Enter a list of numbers separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

mode = find\_mode(input\_list)

print("Mode:", mode)

**27. Write a program to find the greatest common divisor (GCD) of two numbers.**

def find\_gcd(a, b):

while b != 0:

a, b = b, a % b

return a

# Example usage

input\_a = int(input("Enter the first number: "))

input\_b = int(input("Enter the second number: "))

gcd = find\_gcd(input\_a, input\_b)

print("GCD:", gcd)

**28. Implement a function to calculate the square root of a given number.**

def square\_root(number):

if number < 0:

return None

if number == 0:

return 0

guess = number / 2

while True:

new\_guess = (guess + number / guess) / 2

if abs(new\_guess - guess) < 1e-9:

break

guess = new\_guess

return new\_guess

# Example usage

input\_number = float(input("Enter a number: "))

sqrt = square\_root(input\_number)

if sqrt is not None:

print("Square root:", sqrt)

else:

print("Cannot calculate square root of a negative number.")

**29. Write a Python program to check if a given string is a valid palindrome ignoring**

**non-alphanumeric characters.**

def is\_valid\_palindrome(string):

# Remove non-alphanumeric characters and convert to lowercase

alphanumeric\_string = ''.join(char.lower() for char in string if char.isalnum())

# Check if the alphanumeric string is a palindrome

return alphanumeric\_string == alphanumeric\_string[::-1]

# Example usage

input\_string = input("Enter a string: ")

if is\_valid\_palindrome(input\_string):

print("The string is a valid palindrome.")

else:

print("The string is not a valid palindrome.")

**30. Implement a function to find the minimum element in a rotated sorted list.**

def find\_min\_in\_rotated\_sorted\_list(lst):

left = 0

right = len(lst) - 1

while left < right:

mid = left + (right - left) // 2

if lst[mid] > lst[right]:

# Minimum element is in the right half

left = mid + 1

else:

# Minimum element is in the left half or is mid itself

right = mid

return lst[left]

# Example usage

input\_list = input("Enter a rotated sorted list of numbers separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

min\_element = find\_min\_in\_rotated\_sorted\_list(input\_list)

print("Minimum element:", min\_element)

**31. Write a program to find the sum of all even numbers in a list.**

def sum\_of\_even\_numbers(lst):

sum\_even = 0

for num in lst:

if num % 2 == 0:

sum\_even += num

return sum\_even

# Example usage

input\_list = input("Enter a list of numbers separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

sum\_even = sum\_of\_even\_numbers(input\_list)

print("Sum of even numbers:", sum\_even)

**32. Implement a function to calculate the power of a number using recursion.**

def power(base, exponent):

if exponent == 0:

return 1

elif exponent > 0:

return base \* power(base, exponent - 1)

else:

return 1 / power(base, -exponent)

# Example usage

input\_base = float(input("Enter the base: "))

input\_exponent = int(input("Enter the exponent: "))

result = power(input\_base, input\_exponent)

print("Result:", result)

**33. Write a Python program to remove duplicates from a list while preserving the order.**

def remove\_duplicates(lst):

seen = set()

result = []

for item in lst:

if item not in seen:

seen.add(item)

result.append(item)

return result

# Example usage

input\_list = input("Enter a list of elements separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

result\_list = remove\_duplicates(input\_list)

print("List without duplicates:", result\_list)

**34. Implement a function to find the longest common prefix among a list of strings.**

def find\_longest\_common\_prefix(strs):

if not strs:

return ""

common\_prefix = ""

min\_length = min(len(s) for s in strs)

for i in range(min\_length):

current\_char = strs[0][i]

if all(s[i] == current\_char for s in strs):

common\_prefix += current\_char

else:

break

return common\_prefix

# Example usage

input\_list = input("Enter a list of strings separated by spaces: ").split()

longest\_common\_prefix = find\_longest\_common\_prefix(input\_list)

print("Longest common prefix:", longest\_common\_prefix)

**35. Write a program to check if a given number is a perfect square.**

def is\_perfect\_square(number):

if number < 0:

return False

elif number == 0:

return True

# Using binary search

left = 1

right = number

while left <= right:

mid = (left + right) // 2

square = mid \* mid

if square == number:

return True

elif square < number:

left = mid + 1

else:

right = mid - 1

return False

# Example usage

input\_number = int(input("Enter a number: "))

if is\_perfect\_square(input\_number):

print("The number is a perfect square.")

else:

print("The number is not a perfect square.")

**36. Implement a function to calculate the product of all elements in a list.**

def calculate\_product(lst):

product = 1

for num in lst:

product \*= num

return product

# Example usage

input\_list = input("Enter a list of numbers separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

product = calculate\_product(input\_list)

print("Product:", product)

**37. Write a Python program to reverse the order of words in a sentence while preserving the word order.**

def reverse\_sentence(sentence):

words = sentence.split()

reversed\_words = words[::-1]

reversed\_sentence = ' '.join(reversed\_words)

return reversed\_sentence

# Example usage

input\_sentence = input("Enter a sentence: ")

reversed\_sentence = reverse\_sentence(input\_sentence)

print("Reversed sentence:", reversed\_sentence)

**38. Implement a function to find the missing number in a given list of consecutive numbers.**

def find\_missing\_number(lst):

n = len(lst) + 1

expected\_sum = (n \* (n + 1)) // 2

actual\_sum = sum(lst)

missing\_number = expected\_sum - actual\_sum

return missing\_number

# Example usage

input\_list = input("Enter a list of consecutive numbers separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

missing\_number = find\_missing\_number(input\_list)

print("Missing number:", missing\_number)

**39. Write a program to find the sum of digits of a given number.**

def sum\_of\_digits(number):

sum\_digits = 0

while number > 0:

digit = number % 10

sum\_digits += digit

number //= 10

return sum\_digits

# Example usage

input\_number = int(input("Enter a number: "))

sum\_digits = sum\_of\_digits(input\_number)

print("Sum of digits:", sum\_digits)

**40. Implement a function to check if a given string is a valid palindrome considering case sensitivity.**

def is\_valid\_palindrome(string):

reversed\_string = string[::-1]

return string == reversed\_string

# Example usage

input\_string = input("Enter a string: ")

if is\_valid\_palindrome(input\_string):

print("The string is a valid palindrome.")

else:

print("The string is not a valid palindrome.")

**41. Write a Python program to find the smallest missing positive integer in a list.**

def find\_smallest\_missing\_positive(lst):

n = len(lst)

# Move positive numbers to their correct positions

i = 0

while i < n:

if 1 <= lst[i] <= n and lst[i] != lst[lst[i] - 1]:

# Swap lst[i] with lst[lst[i] - 1]

lst[lst[i] - 1], lst[i] = lst[i], lst[lst[i] - 1]

else:

i += 1

# Find the first index where the number is not in the correct position

for i in range(n):

if lst[i] != i + 1:

return i + 1

# If all numbers are in their correct positions, the missing number is n + 1

return n + 1

# Example usage

input\_list = input("Enter a list of numbers separated by spaces: ").split()

input\_list = [int(num) for num in input\_list]

missing\_number = find\_smallest\_missing\_positive(input\_list)

print("Smallest missing positive integer:", missing\_number)

**42. Implement a function to find the longest palindrome substring in a given string.**

def longest\_palindrome\_substring(string):

if not string:

return ""

n = len(string)

start, end = 0, 0

def expand\_around\_center(left, right):

while left >= 0 and right < n and string[left] == string[right]:

left -= 1

right += 1

return left + 1, right - 1

for i in range(n):

# Expand around the center for odd-length palindromes

left1, right1 = expand\_around\_center(i, i)

if right1 - left1 > end - start:

start, end = left1, right1

# Expand around the center for even-length palindromes

left2, right2 = expand\_around\_center(i, i + 1)

if right2 - left2 > end - start:

start, end = left2, right2

return string[start: end + 1]

# Example usage

input\_string = input("Enter a string: ")

longest\_palindrome = longest\_palindrome\_substring(input\_string)

print("Longest palindrome substring:", longest\_palindrome)

**43. Write a program to find the number of occurrences of a given element in a list.**

def count\_occurrences(lst, element):

count = 0

for item in lst:

if item == element:

count += 1

return count

# Example usage

input\_list = input("Enter a list of elements separated by spaces: ").split()

element = input("Enter the element to count: ")

occurrences = count\_occurrences(input\_list, element)

print("Number of occurrences:", occurrences)

**44. Implement a function to check if a given number is a perfect number.**

def is\_perfect\_number(number):

if number <= 0:

return False

divisor\_sum = 0

for i in range(1, number):

if number % i == 0:

divisor\_sum += i

return divisor\_sum == number

# Example usage

input\_number = int(input("Enter a number: "))

if is\_perfect\_number(input\_number):

print("The number is a perfect number.")

else:

print("The number is not a perfect number.")

**45. Write a Python program to remove all duplicates from a string.**

def remove\_duplicates(string):

unique\_chars = []

for char in string:

if char not in unique\_chars:

unique\_chars.append(char)

return ''.join(unique\_chars)

# Example usage

input\_string = input("Enter a string: ")

result\_string = remove\_duplicates(input\_string)

print("String without duplicates:", result\_string)

**46. Implement a function to find the first missing positive**

def find\_first\_missing\_positive(nums):

n = len(nums)

# Step 1: Move all positive numbers to their correct positions

i = 0

while i < n:

current\_num = nums[i]

if 1 <= current\_num <= n and nums[current\_num - 1] != current\_num:

nums[current\_num - 1], nums[i] = nums[i], nums[current\_num - 1]

else:

i += 1

# Step 2: Find the first index where the number is not in the correct position

for i in range(n):

if nums[i] != i + 1:

return i + 1

# If all numbers are in their correct positions, the missing number is n + 1

return n + 1

# Example usage

input\_nums = input("Enter a list of numbers separated by spaces: ").split()

input\_nums = [int(num) for num in input\_nums]

missing\_number = find\_first\_missing\_positive(input\_nums)

print("First missing positive:", missing\_number)