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

A 1

def reverse\_string(input\_string):

reversed\_str = ""

for char in input\_string:

reversed\_str = char + reversed\_str

return reversed\_str

input\_str = "Hello, World!"

print(reverse\_string(input\_str)) # Output: "!dlroW ,olleH"

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

def is\_palindrome(input\_string):

input\_string = input\_string.lower().replace(" ", "")

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

input\_str = "radar"

print(is\_palindrome(input\_str)) # Output: True

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

def find\_largest\_element(input\_list):

largest = input\_list[0]

for num in input\_list:

if num > largest:

largest = num

return largest

numbers\_list = [10, 5, 20, 15, 30]

print(find\_largest\_element(numbers\_list)) # Output: 30

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

def count\_occurrences(input\_list):

occurrences = {}

for element in input\_list:

occurrences[element] = occurrences.get(element, 0) + 1

return occurrences

numbers\_list = [1, 2, 1, 3, 4, 2, 5, 1]

print(count\_occurrences(numbers\_list)) # Output: {1: 3, 2: 2, 3: 1, 4: 1, 5: 1}

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

def find\_second\_largest(input\_list):

unique\_numbers = sorted(set(input\_list))

return unique\_numbers[-2] if len(unique\_numbers) >= 2 else None

numbers\_list = [10, 5, 20, 15, 30]

print(find\_second\_largest(numbers\_list)) # Output: 20

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

def remove\_duplicates(input\_list):

return list(set(input\_list))

numbers\_list = [1, 2, 1, 3, 4, 2, 5, 1]

print(remove\_duplicates(numbers\_list)) # Output: [1, 2, 3, 4, 5]

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

def factorial(n):

if n == 0 or n == 1:

return 1

return n \* factorial(n - 1)

num = 5

print(factorial(num)) # Output: 120

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

def is\_prime(number):

if number <= 1:

return False

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

if number % i == 0:

return False

return True

num = 17

print(is\_prime(num)) # Output: True

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

def sort\_list\_ascending(input\_list):

return sorted(input\_list)

numbers\_list = [10, 5, 20, 15, 30]

print(sort\_list\_ascending(numbers\_list)) # Output: [5, 10, 15, 20, 30]

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

def sum\_of\_numbers(input\_list):

return sum(input\_list)

numbers\_list = [1, 2, 3, 4, 5]

print(sum\_of\_numbers(numbers\_list)) # Output: 15

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

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

common\_elements = set(list1) & set(list2)

return list(common\_elements)

list1 = [1, 2, 3, 4, 5]

list2 = [4, 5, 6, 7, 8]

print(find\_common\_elements(list1, list2)) # Output: [4, 5]

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

def is\_anagram(str1, str2):

return sorted(str1) == sorted(str2)

string1 = "listen"

string2 = "silent"

print(is\_anagram(string1, string2)) # Output: True

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

from itertools import permutations

def generate\_permutations(input\_string):

perms = [''.join(p) for p in permutations(input\_string)]

return perms

string = "abc"

print(generate\_permutations(string)) # Output: ['abc', 'acb', 'bac', 'bca', 'cab', 'cba']

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

def fibonacci\_sequence(num\_terms):

sequence = [0, 1]

while len(sequence) < num\_terms:

next\_term = sequence[-1] + sequence[-2]

sequence.append(next\_term)

return sequence

num\_terms = 10

print(fibonacci\_sequence(num\_terms)) # Output: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

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

def find\_median(numbers):

sorted\_numbers = sorted(numbers)

n = len(sorted\_numbers)

if n % 2 == 1:

return sorted\_numbers[n // 2]

else:

middle = n // 2

return (sorted\_numbers[middle - 1] + sorted\_numbers[middle]) / 2

numbers\_list = [10, 5, 8, 3, 7, 2, 9]

print(find\_median(numbers\_list)) # Output: 7

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

def is\_sorted(numbers):

return all(numbers[i] <= numbers[i + 1] for i in range(len(numbers) - 1))

numbers\_list = [1, 2, 3, 4, 5]

print(is\_sorted(numbers\_list)) # Output: True

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

def find\_intersection(list1, list2):

return list(set(list1) & set(list2))

list1 = [1, 2, 3, 4, 5]

list2 = [4, 5, 6, 7, 8]

print(find\_intersection(list1, list2)) # Output: [4, 5]

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

def max\_subarray\_sum(numbers):

max\_sum = current\_sum = numbers[0]

for num in numbers[1:]:

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

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

return max\_sum

numbers\_list = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

print(max\_subarray\_sum(numbers\_list)) # Output: 6

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

def remove\_vowels(input\_string):

vowels = "aeiouAEIOU"

return "".join(char for char in input\_string if char not in vowels)

string = "Hello, World!"

print(remove\_vowels(string)) # Output: "Hll, Wrld!"

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

def reverse\_words(sentence):

words = sentence.split()

return " ".join(reversed(words))

sentence = "Hello, World!"

print(reverse\_words(sentence)) # Output: "World! Hello,"

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

def are\_anagrams(str1, str2):

return sorted(str1) == sorted(str2)

string1 = "listen"

string2 = "silent"

print(are\_anagrams(string1, string2)) # Output: True

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

def find\_first\_non\_repeating\_character(input\_string):

char\_counts = {}

for char in input\_string:

char\_counts[char] = char\_counts.get(char, 0) + 1

for char in input\_string:

if char\_counts[char] == 1:

return char

return None

string = "statistics"

print(find\_first\_non\_repeating\_character(string)) # Output: 'a'

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

def find\_prime\_factors(number):

factors = []

divisor = 2

while divisor <= number:

if number % divisor == 0:

factors.append(divisor)

number //= divisor

else:

divisor += 1

return factors

num = 36

print(find\_prime\_factors(num)) # Output: [2, 2, 3, 3]

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

num = 16

print(is\_power\_of\_two(num)) # Output: True

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

merged\_list.extend(list1[i:])

merged\_list.extend(list2[j:])

return merged\_list

list1 = [1, 3, 5]

list2 = [2, 4, 6]

print(merge\_sorted\_lists(list1, list2)) # Output: [1, 2, 3, 4, 5, 6]

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

from collections import Counter

def find\_mode(numbers):

count\_dict = Counter(numbers)

max\_count = max(count\_dict.values())

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

numbers\_list = [1, 2, 3, 2, 4, 3, 5, 3]

print(find\_mode(numbers\_list)) # Output: [3]

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

def find\_gcd(num1, num2):

while num2:

num1, num2 = num2, num1 % num2

return abs(num1)

a, b = 24, 36

print(find\_gcd(a, b)) # Output: 12

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

def square\_root(num):

if num < 0:

return None

guess = num

while abs(guess \* guess - num) > 1e-6:

guess = (guess + num / guess) / 2

return guess

num = 25

print(square\_root(num)) # Output: 5.0

29. Write a Python program to check if a given string is a valid palindrome ignoring non-alphanumeric characters.

import re

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

cleaned\_string = re.sub(r'[^a-zA-Z0-9]', '', input\_string).lower()

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

string = "A man, a plan, a canal, Panama!"

print(is\_valid\_palindrome(string)) # Output: True

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

def find\_minimum\_rotated\_sorted(nums):

left, right = 0, len(nums) - 1

while left < right:

mid = (left + right) // 2

if nums[mid] > nums[right]:

left = mid + 1

else:

right = mid

return nums[left]

rotated\_list = [4, 5, 6, 7, 0, 1, 2]

print(find\_minimum\_rotated\_sorted(rotated\_list)) # Output: 0

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

def sum\_even\_numbers(numbers):

return sum(num for num in numbers if num % 2 == 0)

numbers\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(sum\_even\_numbers(numbers\_list)) # Output: 30

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 1 / power(base, -exponent)

else:

half\_power = power(base, exponent // 2)

if exponent % 2 == 0:

return half\_power \* half\_power

else:

return half\_power \* half\_power \* base

base = 2

exponent = 5

print(power(base, exponent)) # Output: 32

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

def remove\_duplicates(input\_list):

seen = set()

return [x for x in input\_list if not (x in seen or seen.add(x))]

numbers\_list = [1, 2, 2, 3, 4, 4, 5, 5]

print(remove\_duplicates(numbers\_list)) # Output: [1, 2, 3, 4, 5]

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

def longest\_common\_prefix(strings):

if not strings:

return ""

prefix = strings[0]

for string in strings[1:]:

i = 0

while i < len(prefix) and i < len(string) and prefix[i] == string[i]:

i += 1

prefix = prefix[:i]

return prefix

string\_list = ["flower", "flow", "flight"]

print(longest\_common\_prefix(string\_list)) # Output: "fl"

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

def is\_perfect\_square(num):

if num < 0:

return False

root = int(num \*\* 0.5)

return root \* root == num

num = 16

print(is\_perfect\_square(num)) # Output: True

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

def product\_of\_elements(numbers):

result = 1

for num in numbers:

result \*= num

return result

numbers\_list = [1, 2, 3, 4, 5]

print(product\_of\_elements(numbers\_list)) # Output: 120

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]

return " ".join(reversed\_words)

sentence = "Hello, World!"

print(reverse\_sentence(sentence)) # Output: "World! Hello,"

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

def find\_missing\_number(nums):

n = len(nums)

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

actual\_sum = sum(nums)

return total\_sum - actual\_sum

numbers\_list = [1, 2, 3, 5, 6]

print(find\_missing\_number(numbers\_list)) # Output: 4

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

def sum\_of\_digits(number):

return sum(int(digit) for digit in str(number))

num = 12345

print(sum\_of\_digits(num)) # Output: 15

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

def is\_valid\_palindrome\_case\_sensitive(input\_string):

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

string = "Racecar"

print(is\_valid\_palindrome\_case\_sensitive(string)) # Output: False

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

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

nums\_set = set(nums)

smallest\_positive = 1

while smallest\_positive in nums\_set:

smallest\_positive += 1

return smallest\_positive

numbers\_list = [3, 5, -1, 1]

print(find\_smallest\_missing\_positive(numbers\_list)) # Output: 2

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

def longest\_palindrome\_substring(input\_string):

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

while left >= 0 and right < len(input\_string) and input\_string[left] == input\_string[right]:

left -= 1

right += 1

return input\_string[left + 1: right]

longest\_palindrome = ""

for i in range(len(input\_string)):

odd\_palindrome = expand\_around\_center(i, i)

even\_palindrome = expand\_around\_center(i, i + 1)

if len(odd\_palindrome) > len(longest\_palindrome):

longest\_palindrome = odd\_palindrome

if len(even\_palindrome) > len(longest\_palindrome):

longest\_palindrome = even\_palindrome

return longest\_palindrome

string = "babad"

print(longest\_palindrome\_substring(string)) # Output: "bab" or "aba"

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

def count\_occurrences(nums, target):

return nums.count(target)

numbers\_list = [1, 2, 2, 3, 2, 4, 2, 5]

target\_num = 2

print(count\_occurrences(numbers\_list, target\_num)) # Output: 4

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

def is\_perfect\_number(num):

if num <= 0:

return False

divisors\_sum = sum(i for i in range(1, num) if num % i == 0)

return divisors\_sum == num

num = 28

print(is\_perfect\_number(num)) # Output: True

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

def remove\_duplicates\_from\_string(input\_string):

unique\_chars = []

for char in input\_string:

if char not in unique\_chars:

unique\_chars.append(char)

return "".join(unique\_chars)

string = "abracadabra"

print(remove\_duplicates\_from\_string(string)) # Output: "abrcd"

46. Implement a function to find the first missing positive

def first\_missing\_positive(nums):

n = len(nums)

for i in range(n):

while 1 <= nums[i] <= n and nums[nums[i] - 1] != nums[i]:

nums[nums[i] - 1], nums[i] = nums[i], nums[nums[i] - 1]

for i in range(n):

if nums[i] != i + 1:

return i + 1

return n + 1

numbers\_list = [3, 4, -1, 1]

print(first\_missing\_positive(numbers\_list)) # Output: 2