1. Here's a Python program to reverse a string without using any built-in string reversal functions:

```python

def reverse\_string(input\_string):

reversed\_string = ''

for char in input\_string:

reversed\_string = char + reversed\_string

return reversed\_string

# Example usage:

input\_string = "Hello, World!"

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

print(reversed\_string)

```

2. Here's an implementation of a function to check if a given string is a palindrome:

```python

def is\_palindrome(input\_string):

input\_string = input\_string.lower() # Convert to lowercase for case-insensitive comparison

left = 0

right = len(input\_string) - 1

while left < right:

if input\_string[left] != input\_string[right]:

return False

left += 1

right -= 1

return True

# Example usage:

input\_string = "level"

if is\_palindrome(input\_string):

print("Palindrome")

else:

print("Not a palindrome")

```

3. Here's a program to find the largest element in a given list:

```python

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

largest = input\_list[0]

for num in input\_list:

if num > largest:

largest = num

return largest

# Example usage:

numbers = [5, 9, 3, 7, 2]

largest\_number = find\_largest\_element(numbers)

print(largest\_number)

```

4. Here's an implementation of a function to count the occurrence of each element in a list:

```python

def count\_occurrences(input\_list):

occurrences = {}

for element in input\_list:

if element in occurrences:

occurrences[element] += 1

else:

occurrences[element] = 1

return occurrences

# Example usage:

numbers = [1, 2, 3, 2, 1, 3, 1, 1, 4]

occurrences = count\_occurrences(numbers)

print(occurrences)

```

5. Here's a Python program to find the second largest number in a list:

```python

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

if len(input\_list) < 2:

return None

largest = max(input\_list[0], input\_list[1])

second\_largest = min(input\_list[0], input\_list[1])

for i in range(2, len(input\_list)):

if input\_list[i] > largest:

second\_largest = largest

largest = input\_list[i]

elif input\_list[i] > second\_largest:

second\_largest = input\_list[i]

return second\_largest

# Example usage:

numbers = [5, 9, 3, 7, 2]

second\_largest = find\_second\_largest(numbers)

print(second\_largest)

```

6. Here's an implementation of a function to remove duplicate elements from a list:

```python

def remove\_duplicates(input\_list):

unique\_elements = []

for element in input\_list:

if element not in unique\_elements:

unique\_elements.append(element)

return unique\_elements

# Example usage:

numbers = [1, 2, 3, 2, 1, 3, 1, 1, 4]

unique\_numbers = remove\_duplicates(numbers)

print(unique\_numbers)

```

(Note: The order of elements may change after removing duplicates.)

7. Here's a program to calculate the factorial of a given number:

```python

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

# Example usage:

number = 5

factorial\_result = factorial(number)

print(factorial\_result)

```

8. Here's an implementation of a function to check if a given number is prime:

```python

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

# Example usage:

number = 17

if is\_prime(number):

print("Prime")

else:

print("Not prime")

```

9. Here's a Python program to sort a list of integers in ascending order:

```python

def sort\_list(input\_list):

sorted\_list = input\_list.copy()

for i in range(len(sorted\_list)):

for j in range(i+1, len(sorted\_list)):

if sorted\_list[i] > sorted\_list[j]:

sorted\_list[i], sorted\_list[j] = sorted\_list[j], sorted\_list[i]

return sorted\_list

# Example usage:

numbers = [5, 9, 3, 7, 2]

sorted\_numbers = sort\_list(numbers)

print(sorted\_numbers)

```

10. Here's an implementation of a function to find the sum of all numbers in a list:

```python

def find\_sum(input\_list):

total = 0

for num in input\_list:

total += num

return total

# Example usage:

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

sum\_of\_numbers = find\_sum(numbers)

print(sum\_of\_numbers)

```

11. Here's a program to find the common elements between two lists:

```python

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

common\_elements = []

for element in list1:

if element in list2 and element not in common\_elements:

common\_elements.append(element)

return common\_elements

# Example usage:

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

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

common\_elements = find\_common\_elements(list1, list2)

print(common\_elements)

```

12. Here's an implementation of a function to check if a given string is an anagram of another string:

```python

def is\_anagram(str1, str2):

str1 = str1.lower()

str2 = str2.lower()

return sorted(str1) == sorted(str2)

# Example usage:

string1 = "listen"

string2 = "silent"

if is\_anagram(string1, string2):

print("Anagram")

else:

print("Not an anagram")

```

13. Here's a Python program to generate all permutations of a given string:

```python

def permute\_string(input\_string):

if len(input\_string) == 0:

return ['']

permutations = []

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

char = input\_string[i]

remaining\_chars = input\_string[:i] + input\_string[i+1:]

for permutation in permute\_string(remaining\_chars):

permutations.append(char + permutation)

return permutations

# Example usage:

string = "abc"

permutations = permute\_string(string)

print(permutations)

```

14. Here's an implementation of a function to calculate the Fibonacci sequence up to a given number of terms:

```python

def fibonacci(n):

sequence = []

if n >= 1:

sequence.append(0)

if n >= 2:

sequence.append(1)

for i in range(2, n):

sequence.append(sequence[i-1] + sequence[i-2])

return sequence

# Example usage:

num\_terms = 10

fib\_sequence = fibonacci(num\_terms)

print(fib\_sequence)

```

15. Here's a program to find the median of a list of numbers:

```python

def find\_median(input\_list):

sorted\_list = sorted(input\_list)

n = len(sorted\_list)

if n % 2 == 1:

return sorted\_list[n // 2]

else:

mid\_right = n // 2

mid\_left = mid\_right - 1

return (sorted\_list[mid\_left] + sorted\_list[mid\_right]) / 2

# Example usage:

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

median = find\_median(numbers)

print(median)

```

16. Here's an implementation of a function to check if a given list is sorted in non-decreasing order:

```python

def is\_sorted(input\_list):

for i in range(len(input\_list) - 1):

if input\_list[i] > input\_list[i+1]:

return False

return True

# Example usage:

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

if is\_sorted(numbers):

print("Sorted")

else:

print("Not sorted")

```

17. Here's a Python program to find the intersection of two lists:

```python

def find\_intersection(list1, list2):

intersection = []

for element in list1:

if element in list2 and element not in intersection:

intersection.append(element)

return intersection

# Example usage:

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

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

intersection = find\_intersection(list1, list2)

print(intersection)

```

18. Here's an implementation of a function to find the maximum subarray sum in a given list:

```python

def find\_maximum\_subarray\_sum(input\_list):

current\_sum = 0

max\_sum = float('-inf')

for num in input\_list:

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

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

return max\_sum

# Example usage:

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

max\_subarray\_sum = find\_maximum\_subarray\_sum(numbers)

print(max\_subarray\_sum)

```

19. Here's a program to remove all vowels from a given string:

```python

def remove\_vowels(input\_string):

vowels = ['a', 'e', 'i', 'o', 'u']

modified\_string = ''

for char in input\_string:

if char.lower() not in vowels:

modified\_string += char

return modified\_string

# Example usage:

string = "Hello, World!"

modified\_string = remove\_vowels(string)

print(modified\_string)

```

20. Here's an implementation of a function to reverse the order of words in a given sentence:

```python

def reverse\_words(input\_sentence):

words = input\_sentence.split()

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

return reversed\_words

# Example usage:

sentence = "Hello, World!"

reversed\_sentence = reverse\_words(sentence)

print(reversed\_sentence)

```

21. Here's a Python program to check if two strings are anagrams of each other:

```python

def are\_anagrams(str1, str2):

str1 = str1.lower()

str2 = str2.lower()

return sorted(str1) == sorted(str2)

# Example usage:

string1 = "listen"

string2 = "silent"

if are\_anagrams(string1, string2):

print("Anagrams")

else:

print("Not anagrams")

```

22. Here's an implementation of a function to find the first non-repeating character in a string:

```python

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

char\_count = {}

for char in input\_string:

if char in char\_count:

char\_count[char] += 1

else:

char\_count[char] = 1

for char in input\_string:

if char\_count[char] == 1:

return char

return None

# Example usage:

string = "aabbcde"

first\_non\_repeating\_char = find\_first\_non\_repeating\_char(string)

print(first\_non\_repeating\_char)

```

23. Here's a program to find the prime factors of a given number:

```python

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:

number = 36

prime\_factors = find\_prime\_factors(number)

print(prime\_factors)

```

24. Here's an implementation of a function to check if a given number is a power of two:

```python

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

if number <= 0:

return False

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

# Example usage:

number = 16

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

print("Power of two")

else:

print("Not a power of two")

```

25. Here's a Python program to merge two sorted lists into a single sorted list:

```python

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

merged\_list = []

i = j = 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

# Example usage:

list1 = [1, 3, 5, 7]

list2 = [2, 4, 6, 8]

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

print(merged\_list)

```

26. Here's an implementation of a function to find the mode of a list of numbers:

```python

from collections import Counter

def find\_mode(input\_list):

count = Counter(input\_list)

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

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

return mode

# Example usage:

numbers = [1, 2, 3, 2, 1, 3, 1, 1, 4]

mode = find\_mode(numbers)

print(mode)

```

27. Here's a program to find the greatest common divisor (GCD) of two numbers:

```python

def find\_gcd(a, b):

while b:

a, b = b, a % b

return a

# Example usage:

num1 = 24

num2 = 36

gcd = find\_gcd(num1, num2)

print(gcd)

```

28. Here's an implementation of a function to calculate the square root of a given number:

```python

def square\_root(number):

if number < 0:

return None

guess = number

while abs(guess \* guess - number) > 1e-10:

guess = (guess + number / guess) / 2

return guess

# Example usage:

number = 25

sqrt = square\_root(number)

print(sqrt)

```

29. Here's a Python program to check if a given string is a valid palindrome ignoring non-alphanumeric characters:

```python

import re

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

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

cleaned\_string = cleaned\_string.lower()

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

# Example usage:

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

if is\_valid\_palindrome(string):

print("Valid palindrome")

else:

print("Not a valid palindrome")

```

30. Here's an implementation of a function to find the minimum element in a rotated sorted list:

```python

def find\_minimum\_rotated(input\_list):

left = 0

right = len(input\_list) - 1

while left < right:

mid = (left + right) // 2

if input\_list[mid] > input\_list[right]:

left = mid + 1

else:

right = mid

return input\_list[left]

# Example usage:

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

minimum = find\_minimum\_rotated(numbers)

print(minimum)

```

31. Here's a program to find the sum of all even numbers in a list:

```python

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

sum\_even = 0

for num in input\_list:

if num % 2 == 0:

sum\_even += num

return sum\_even

# Example usage:

numbers = [1, 2, 3, 4, 5, 6]

sum\_even = sum\_even\_numbers(numbers)

print(sum\_even)

```

32. Here's an implementation of a function to calculate the power of a number using recursion:

```python

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:

base = 2

exponent = 3

result = power(base, exponent)

print(result)

```

33. Here's a Python program to remove duplicates from a list while preserving the order:

```python

def remove\_duplicates\_preserve\_order(input\_list):

unique\_elements = []

for element in input\_list:

if element not in unique\_elements:

unique\_elements.append(element)

return unique\_elements

# Example usage:

numbers = [1, 2, 3, 2, 1, 3, 1, 1, 4]

unique\_numbers = remove\_duplicates\_preserve\_order(numbers)

print(unique\_numbers)

```

34. Here's an implementation of a function to find the longest common prefix among a list of strings:

```python

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

if not strings:

return ''

prefix = strings[0]

for string in strings[1:]:

while string[:len(prefix)] != prefix:

prefix = prefix[:-1]

if not prefix:

return ''

return prefix

# Example usage:

words = ["flower", "flow", "flight"]

common\_prefix = find\_longest\_common\_prefix(words)

print(common\_prefix)

```

35. Here's a program to check if a given number is a perfect square:

```python

def is\_perfect\_square(number):

if number < 0:

return False

sqrt = int(number \*\* 0.5)

return sqrt \* sqrt == number

# Example usage:

number = 16

if is\_perfect\_square(number):

print("Perfect square")

else:

print("Not a perfect square")

```

36. Here's an implementation of a function to calculate the product of all elements in a list:

```python

def calculate\_product(input\_list):

product = 1

for num in input\_list:

product \*= num

return product

# Example usage:

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

product = calculate\_product(numbers)

print(product)

```

37. Here's a Python program to reverse the order of words in a sentence while preserving the word order:

```python

def reverse\_words\_order(sentence):

words = sentence.split()

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

return reversed\_words

# Example usage:

sentence = "Hello, World!"

reversed\_sentence = reverse\_words\_order(sentence)

print(reversed\_sentence)

```

38. Here's an implementation of a function to find the missing number in a given list of consecutive numbers:

```python

def find\_missing\_number(input\_list):

n = len(input\_list) + 1

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

actual\_sum = sum(input\_list)

missing\_number = total\_sum - actual\_sum

return missing\_number

# Example usage:

numbers = [1, 2, 3, 5, 6, 7, 8, 9]

missing\_number = find\_missing\_number(numbers)

print(missing\_number)

```

39. Here's a program to find the sum of digits of a given number:

```python

def sum\_of\_digits(number):

total\_sum = 0

while number > 0:

digit = number % 10

total\_sum += digit

number //= 10

return total\_sum

# Example usage:

number = 12345

digit\_sum = sum\_of\_digits(number)

print(digit\_sum)

```

40. Here's an implementation of a function to check if a given string is a valid palindrome considering case sensitivity:

```python

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

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

# Example usage:

string = "Racecar"

if is\_valid\_palindrome\_case\_sensitive(string):

print("Valid palindrome")

else:

print("Not a valid palindrome")

```

41. Here's a Python program to find the smallest missing positive integer in a list:

```python

def find\_smallest\_missing\_positive(input\_list):

positive\_nums = set()

for num in input\_list:

if num > 0:

positive\_nums.add(num)

smallest\_missing = 1

while smallest\_missing in positive\_nums:

smallest\_missing += 1

return smallest\_missing

# Example usage:

numbers = [1, 2, 0]

smallest\_missing = find\_smallest\_missing\_positive(numbers)

print(smallest\_missing)

```

42. Here's an implementation of a function to find the longest palindrome substring in a given string:

```python

def find\_longest\_palindrome\_substring(input\_string):

longest\_palindrome = ''

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

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

substring = input\_string[i:j+1]

if substring == substring[::-1] and len(substring) > len(longest\_palindrome):

longest\_palindrome = substring

return longest\_palindrome

# Example usage:

string = "babad"

longest\_pal

indrome = find\_longest\_palindrome\_substring(string)

print(longest\_palindrome)

```

43. Here's a program to find the number of occurrences of a given element in a list:

```python

def count\_occurrences(input\_list, element):

count = 0

for item in input\_list:

if item == element:

count += 1

return count

# Example usage:

numbers = [1, 2, 3, 2, 1, 3, 1, 1, 4]

element = 1

occurrences = count\_occurrences(numbers, element)

print(occurrences)

```

44. Here's an implementation of a function to check if a given number is a perfect number:

```python

def is\_perfect\_number(number):

if number < 2:

return False

divisor\_sum = 1

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

if number % i == 0:

divisor\_sum += i

if i != number // i:

divisor\_sum += number // i

return divisor\_sum == number

# Example usage:

number = 28

if is\_perfect\_number(number):

print("Perfect number")

else:

print("Not a perfect number")

```

45. Here's a Python program to remove all duplicates from a string:

```python

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)

# Example usage:

string = "Hello, World!"

modified\_string = remove\_duplicates\_from\_string(string)

print(modified\_string)

```

46. Here's an implementation of a function to reverse the order of words in a sentence:

```python

def reverse\_words\_in\_sentence(sentence):

words = sentence.split()

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

return reversed\_sentence

# Example usage:

sentence = "Hello, World!"

reversed\_sentence = reverse\_words\_in\_sentence(sentence)

print(reversed\_sentence)

```