85 PYTHON BASIC PROGRAMS

BY - SHUBHAM WADEKAR

87 BASIC PYTHON PROGRAMS

1. Write a Python program to print "Hello Python".

Explanation:

This is a simple program to print a string.

Logic:

Use the print() function to display the string.

Program:

```python
print("Hello Python")
```

2. Write a Python program to do arithmetical operations addition and division.

Explanation:

This program performs addition and division on two numbers.

Logic:

- Take two numbers as input.
- Perform addition using the + operator.
- Perform division using the / operator.

Program:

```python
a = 10
b = 5

# Addition
addition = a + b
print("Addition:", addition)

# Division
division = a / b
print("Division:", division)
```

3. Write a Python program to find the area of a triangle.

Explanation:

This program calculates the area of a triangle using the formula:

Area=12×base×heightArea=1/2×base×height

Logic:

- Take base and height as input.
- Use the formula to calculate the area.

Program:

```
base = 10
height = 5

# Calculate area
area = 0.5 * base * height
print("Area of the triangle:", area)
```

4. Write a Python program to swap two variables.

Explanation:

This program swaps the values of two variables.

Logic:

- Use a temporary variable to hold one value during the swap.

Program:

```
a = 5
b = 10

# Swap using a temporary variable
temp = a
a = b
b = temp

print("After swapping:")
print("a =", a)
print("b =", b)
```

5. Write a Python program to generate a random number.

Explanation:

This program generates a random number within a specified range.

Logic:

- Use the random module to generate a random number.

Program:

```
import random

# Generate a random number between 1 and 100
random_number = random.randint(1, 100)
print("Random number:", random_number)
```

6. Write a Python program to convert kilometers to miles.

Explanation:

This program converts a distance from kilometers to miles using the formula:

Miles=Kilometers×0.621371Miles=Kilometers×0.621371

Logic:

- Take kilometers as input.
- Use the formula to convert to miles.

```
kilometers = 10

# Convert to miles
miles = kilometers * 0.621371
print(f"{kilometers} kilometers = {miles} miles")
```

7. Write a Python program to convert Celsius to Fahrenheit.

Explanation:

This program converts a temperature from Celsius to Fahrenheit using the formula:

Fahrenheit=(Celsius×95)+32Fahrenheit=(Celsius×9/5)+32

Logic:

- Take Celsius as input.
- Use the formula to convert to Fahrenheit.

Program:

```
celsius = 25

# Convert to Fahrenheit
fahrenheit = (celsius * 9/5) + 32
print(f"{celsius}°C = {fahrenheit}°F")
```

8. Write a Python program to display a calendar.

Explanation:

This program displays the calendar for a specific month and year.

Logic:

- Use the calendar module to display the calendar.

Program:

```
import calendar

year = 2023
month = 10

# Display calendar
print(calendar.month(year, month))
```

9. Write a Python program to solve a quadratic equation.

Explanation:

This program solves a quadratic equation of the form:

$$ax^2 + bx + c = 0$$

using the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Logic:

- Take coefficients a*a*, b*b*, and c*c* as input.
- Calculate the discriminant (D=b2−4ac*D*=*b*2−4*ac*).
- Use the quadratic formula to find the roots

Program:

```python
import math

a = 1
b = -3
c = 2

# Calculate discriminant
discriminant = b**2 - 4*a*c

if discriminant > 0:
    root1 = (-b + math.sqrt(discriminant)) / (2*a)
    root2 = (-b - math.sqrt(discriminant)) / (2*a)
    print("Roots are real and different:", root1, root2)
elif discriminant == 0:
    root = -b / (2*a)
    print("Roots are real and same:", root)
else:
    print("No real roots")
```

10. Write a Python program to swap two variables without a temporary variable.

Explanation:

This program swaps the values of two variables without using a temporary variable.

Logic:

- Use arithmetic operations or tuple unpacking to swap values.

Program:

```python
a = 5
b = 10

# Swap without a temporary variable
a, b = b, a

print("After swapping:")
print("a =", a)
print("b =", b)
```

11. Write a Python Program to Check if a Number is Positive, Negative or Zero.

Explanation:

This program checks whether a given number is positive, negative, or zero.

Logic:

- Use conditional statements (if, elif, else) to compare the number with zero.

Program:

```python
num = float(input("Enter a number: "))

if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

12. Write a Python Program to Check if a Number is Odd or Even.

Explanation:

This program checks whether a given number is odd or even.

Logic:

- Use the modulus operator % to check if the number is divisible by 2.

Program:

```python
num = int(input("Enter a number: "))

if num % 2 == 0:
    print("Even number")
else:
    print("Odd number")
```

13. Write a Python Program to Check Leap Year.

Explanation:

This program checks whether a given year is a leap year.

Logic:

- A year is a leap year if:
    - It is divisible by 4 but not by 100, or
    - It is divisible by 400.

```python
year = int(input("Enter a year: "))

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print("Leap year")
else:
    print("Not a leap year")
```

14. Write a Python Program to Check Prime Number.

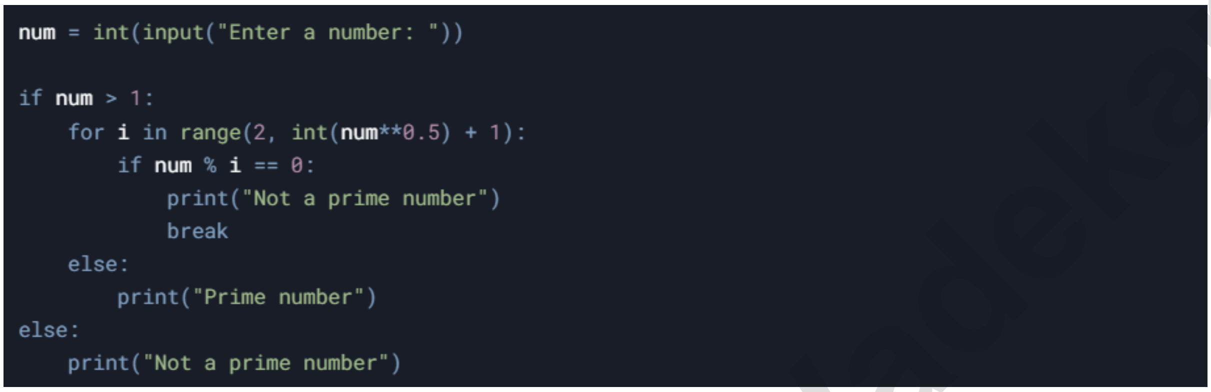
Explanation:

This program checks whether a given number is a prime number.

Logic:

- A prime number is only divisible by 1 and itself.
- Check divisibility from 2 to the square root of the number.

Program:



```python
num = int(input("Enter a number: "))

if num > 1:
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            print("Not a prime number")
            break
    else:
        print("Prime number")
else:
    print("Not a prime number")
```

15. Write a Python Program to Print all Prime Numbers in an Interval of 1-10.

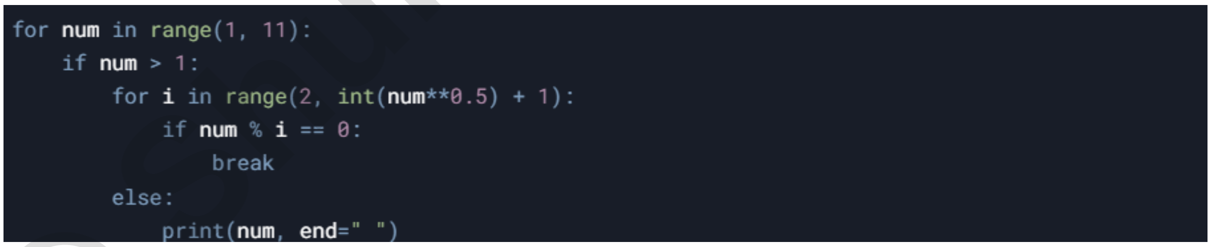
Explanation:

This program prints all prime numbers between 1 and 10.

Logic:

- Use a loop to iterate through numbers from 1 to 10.
- Check if each number is prime using the logic from the previous program.

Program:



```python
for num in range(1, 11):
    if num > 1:
        for i in range(2, int(num**0.5) + 1):
            if num % i == 0:
                break
        else:
            print(num, end=" ")
```

16. Write a Python Program to Find the Factorial of a Number.

Explanation:

This program calculates the factorial of a given number.

Logic:

- Factorial of a number n*n* is the product of all positive integers less than or equal to n*n*.
- Use a loop to multiply numbers from 1 to n*n*.

Program:

```
num = int(input("Enter a number: "))
factorial = 1

for i in range(1, num + 1):
    factorial *= i

print("Factorial:", factorial)
```

17. Write a Python Program to Display the Multiplication Table.

Explanation:

This program displays the multiplication table for a given number.

Logic:

- Use a loop to multiply the number by integers from 1 to 10.

Program:

```
num = int(input("Enter a number: "))

for i in range(1, 11):
    print(f"{num} x {i} = {num * i}")
```

18. Write a Python Program to Print the Fibonacci Sequence.

Explanation:

This program prints the Fibonacci sequence up to a specified number of terms.

Logic:

- The Fibonacci sequence starts with 0 and 1, and each subsequent number is the sum of the previous two.
- Use a loop to generate the sequence.

Program:

```
n = int(input("Enter the number of terms: "))
a, b = 0, 1

for _ in range(n):
    print(a, end=" ")
    a, b = b, a + b
```

19. Write a Python Program to Check Armstrong Number.

Explanation:

This program checks whether a given number is an Armstrong number.

Logic:

- An Armstrong number is a number that is equal to the sum of its own digits raised to the power of the number of digits.

For example, $153 = 1^3 + 5^3 + 3^3$.

Program:

```
num = int(input("Enter a number: "))
order = len(str(num))
sum = 0

temp = num
while temp > 0:
    digit = temp % 10
    sum += digit ** order
    temp //= 10

if num == sum:
    print("Armstrong number")
else:
    print("Not an Armstrong number")
```

20. Write a Python Program to Find Armstrong Numbers in an Interval.

Explanation:

This program finds all Armstrong numbers within a given interval.

Logic:

- Use a loop to iterate through the interval.
- Check if each number is an Armstrong number using the logic from the previous program.

Program:

```
lower = int(input("Enter lower range: "))
upper = int(input("Enter upper range: "))

for num in range(lower, upper + 1):
    order = len(str(num))
    sum = 0
    temp = num
    while temp > 0:
        digit = temp % 10
        sum += digit ** order
        temp //= 10
    if num == sum:
        print(num, end=" ")
```

21. Write a Python Program to Find the Sum of Natural Numbers.

Explanation:

This program calculates the sum of the first nn natural numbers.

Logic:

- Use the formula for the sum of the first nn natural numbers:

$$\text{Sum} = \frac{n(n+1)}{2}$$

Program:

```python
n = int(input("Enter a number: "))
sum = n * (n + 1) // 2
print("Sum of natural numbers up to", n, "is", sum)
```

22. Write a Python Program to Find LCM.

Explanation:

This program calculates the Least Common Multiple (LCM) of two numbers.

Logic:

- Use the relationship between LCM and HCF:

$$\text{LCM}(a, b) = \frac{a \times b}{\text{HCF}(a, b)}$$

Program:

```python
def hcf(a, b):
    while b:
        a, b = b, a % b
    return a

def lcm(a, b):
    return (a * b) // hcf(a, b)

a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
print("LCM of", a, "and", b, "is", lcm(a, b))
```

23. Write a Python Program to Find HCF.

Explanation:

This program calculates the Highest Common Factor (HCF) of two numbers.

Logic:

- Use the Euclidean algorithm to find the HCF:

    Repeatedly replace the larger number with the remainder of the division of the two numbers until one of them becomes zero.

Program:

```
def hcf(a, b):
    while b:
        a, b = b, a % b
    return a

a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
print("HCF of", a, "and", b, "is", hcf(a, b))
```

24. Write a Python Program to Convert Decimal to Binary, Octal and Hexadecimal.

Explanation:

This program converts a decimal number to binary, octal, and hexadecimal.

Logic:

- Use built-in functions bin(), oct(), and hex() for conversion.

Program:

```
decimal = int(input("Enter a decimal number: "))

print("Binary:", bin(decimal))
print("Octal:", oct(decimal))
print("Hexadecimal:", hex(decimal))
```

25. How can we manually convert a decimal number to binary, octal and hexadecimal?

Explanation:

This program manually converts a decimal number to binary, octal, and hexadecimal.

Logic:

- For binary: Repeatedly divide the number by 2 and record the remainders.
- For octal: Repeatedly divide the number by 8 and record the remainders.
- For hexadecimal: Repeatedly divide the number by 16 and record the remainders.

Program:

```python
def decimal_to_binary(n):
    binary = ""
    while n > 0:
        binary = str(n % 2) + binary
        n //= 2
    return binary

def decimal_to_octal(n):
    octal = ""
    while n > 0:
        octal = str(n % 8) + octal
        n //= 8
    return octal

def decimal_to_hex(n):
    hex_digits = "0123456789ABCDEF"
    hex_num = ""
    while n > 0:
        hex_num = hex_digits[n % 16] + hex_num
        n //= 16
    return hex_num

decimal = int(input("Enter a decimal number: "))
print("Binary:", decimal_to_binary(decimal))
print("Octal:", decimal_to_octal(decimal))
print("Hexadecimal:", decimal_to_hex(decimal))
```

26. Write a Python Program To Find ASCII Value of a Character.

Explanation:

This program finds the ASCII value of a given character.

Logic:

- Use the ord() function to get the ASCII value.

Program:

```python
char = input("Enter a character: ")
print("ASCII value of", char, "is", ord(char))
```

27. Write a Python Program to Make a Simple Calculator with 4 Basic Mathematical Operations.

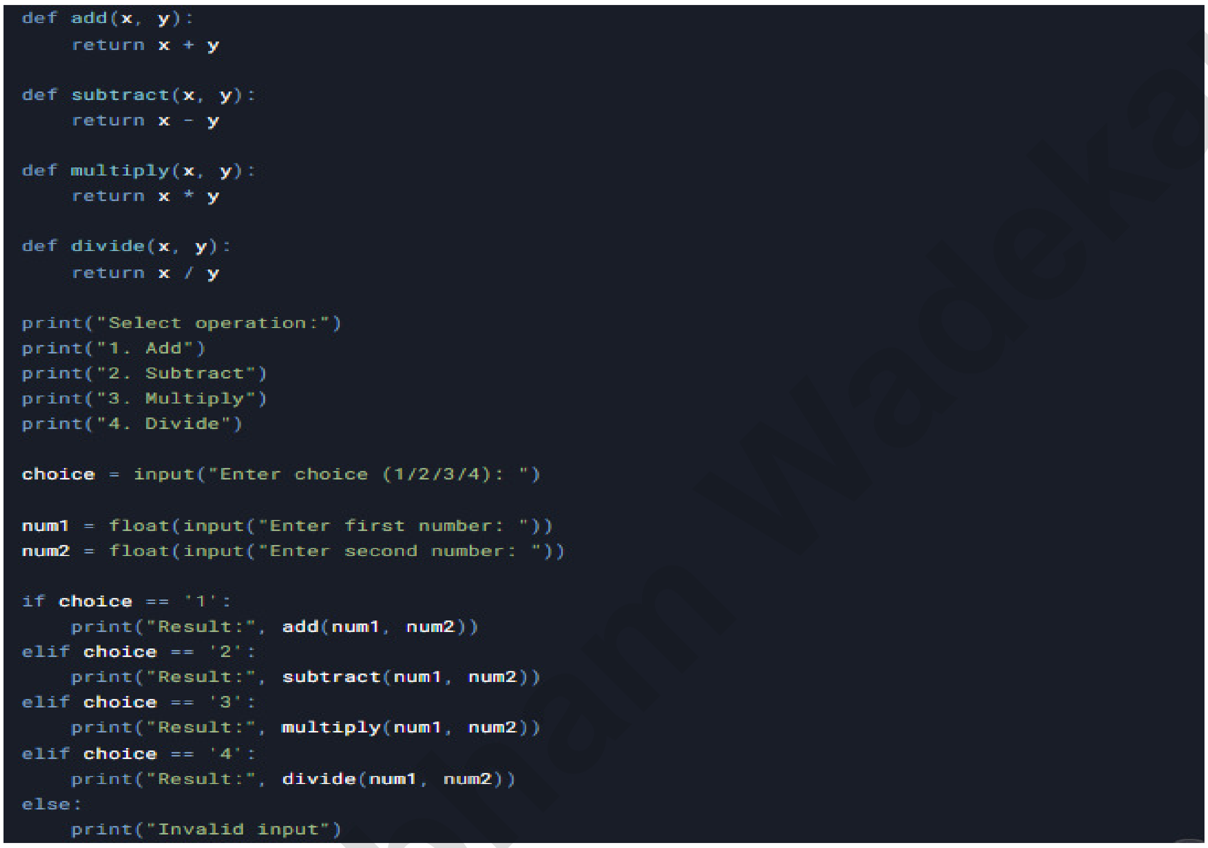
Explanation:

This program performs addition, subtraction, multiplication, and division.

Logic:

- Use conditional statements to perform the selected operation.

Program:



```python
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    return x / y

print("Select operation:")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")

choice = input("Enter choice (1/2/3/4): ")

num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

if choice == '1':
    print("Result:", add(num1, num2))
elif choice == '2':
    print("Result:", subtract(num1, num2))
elif choice == '3':
    print("Result:", multiply(num1, num2))
elif choice == '4':
    print("Result:", divide(num1, num2))
else:
    print("Invalid input")
```

28. Write a Python Program to Display Fibonacci Sequence Using Recursion.

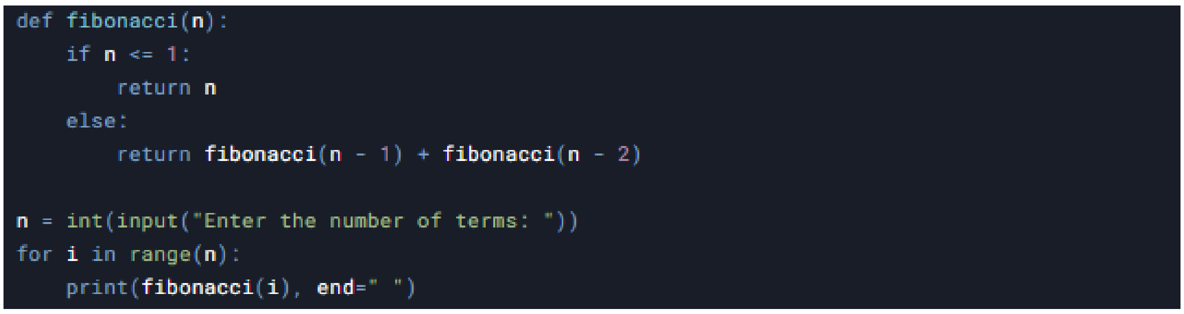
Explanation:

This program prints the Fibonacci sequence using recursion.

Logic:

- Define a recursive function to calculate the Fibonacci number at a given position.

Program:



```python
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

n = int(input("Enter the number of terms: "))
for i in range(n):
    print(fibonacci(i), end=" ")
```

29. Write a Python Program to Find Factorial of Number Using Recursion.

Explanation:

This program calculates the factorial of a number using recursion.

Logic:

- Define a recursive function to calculate the factorial.

Program:

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

num = int(input("Enter a number: "))
print("Factorial of", num, "is", factorial(num))
```

30. Write a Python Program to Calculate Your Body Mass Index (BMI).

Explanation:

This program calculates BMI using the formula:

$$\text{BMI} = \frac{\text{weight (kg)}}{\text{height (m)}^2}$$

Logic:

- Take weight and height as input.
- Use the formula to calculate BMI.

Program:

```
weight = float(input("Enter your weight in kg: "))
height = float(input("Enter your height in meters: "))

bmi = weight / (height ** 2)
print("Your BMI is:", bmi)
```

31. Write a Python Program to Calculate the Natural Logarithm of Any Number.

Explanation:

This program calculates the natural logarithm of a number using the math module.

Logic:

- Use the math.log() function to calculate the natural logarithm.

Program:

```
import math

num = float(input("Enter a number: "))
print("Natural logarithm of", num, "is", math.log(num))
```

32. Write a Python Program for Cube Sum of First n*n* Natural Numbers.

Explanation:

This program calculates the sum of cubes of the first n*n* natural numbers.

Logic:

- Use the formula for the sum of cubes of the first n*n* natural numbers:

Sum=(n(n+1)2)2Sum=(2*n*(*n*+1))2

Program:

```
n = int(input("Enter a number: "))
sum_of_cubes = (n * (n + 1) // 2) ** 2
print("Sum of cubes of first", n, "natural numbers is", sum_of_cubes)
```

33. Write a Python Program to Find the Sum of an Array.

Explanation:

This program calculates the sum of all elements in an array.

Logic:

- Use a loop to iterate through the array and accumulate the sum.

Program:

```
arr = [1, 2, 3, 4, 5]
sum_arr = sum(arr)
print("Sum of array:", sum_arr)
```

34. Write a Python Program to Find the Largest Element in an Array.

Explanation:

This program finds the largest element in an array.

Logic:

- Use the max() function or iterate through the array to find the largest element.

Program:

```
arr = [10, 20, 4, 45, 99]
max_element = max(arr)
print("Largest element in array:", max_element)
```

35. Write a Python Program for Array Rotation.

Explanation:

This program rotates an array by a given number of positions.

Logic:

- Use slicing to rotate the array.

Program:

```
def rotate_array(arr, d):
    return arr[d:] + arr[:d]

arr = [1, 2, 3, 4, 5]
d = 2
rotated_arr = rotate_array(arr, d)
print("Rotated array:", rotated_arr)
```

36. Write a Python Program to Split the Array and Add the First Part to the End.

Explanation:

This program splits the array at a given position and adds the first part to the end.

Logic:

- Use slicing to split the array and concatenate the parts.

Program:

```
def split_and_add(arr, k):
    return arr[k:] + arr[:k]

arr = [1, 2, 3, 4, 5]
k = 2
result = split_and_add(arr, k)
print("Array after splitting and adding:", result)
```

37. Write a Python Program to Check if a Given Array is Monotonic.

Explanation:

This program checks if an array is monotonic (either entirely non-increasing or non-decreasing).

Logic:

- Compare adjacent elements to determine if the array is monotonic.

Program:

```
def is_monotonic(arr):
    increasing = decreasing = True
    for i in range(1, len(arr)):
        if arr[i] > arr[i - 1]:
            decreasing = False
        if arr[i] < arr[i - 1]:
            increasing = False
    return increasing or decreasing

arr = [1, 2, 2, 3]
print("Is array monotonic?", is_monotonic(arr))
```

38. Write a Python Program to Add Two Matrices.

Explanation:

This program adds two matrices of the same dimensions.

Logic:

- Use nested loops to iterate through the matrices and add corresponding elements.

Program:

```
def add_matrices(matrix1, matrix2):
    return [[matrix1[i][j] + matrix2[i][j] for j in range(len(matrix1[0]))] for i in range(len(m
atrix1))]

matrix1 = [[1, 2], [3, 4]]
matrix2 = [[5, 6], [7, 8]]
result = add_matrices(matrix1, matrix2)
print("Sum of matrices:", result)
```

39. Write a Python Program to Multiply Two Matrices.

Explanation:

This program multiplies two matrices.

Logic:

- Use nested loops to perform matrix multiplication.

Program:

```
def multiply_matrices(matrix1, matrix2):
    result = [[0 for _ in range(len(matrix2[0]))] for _ in range(len(matrix1))]
    for i in range(len(matrix1)):
        for j in range(len(matrix2[0])):
            for k in range(len(matrix2)):
                result[i][j] += matrix1[i][k] * matrix2[k][j]
    return result

matrix1 = [[1, 2], [3, 4]]
matrix2 = [[5, 6], [7, 8]]
result = multiply_matrices(matrix1, matrix2)
print("Product of matrices:", result)
```

40. Write a Python Program to Transpose a Matrix.

Explanation:

This program transposes a given matrix (rows become columns and vice versa).

Logic:

- Use list comprehension to swap rows and columns.

Program:

```
def transpose_matrix(matrix):
    return [[matrix[j][i] for j in range(len(matrix))] for i in range(len(matrix[0]))]

matrix = [[1, 2], [3, 4], [5, 6]]
transposed = transpose_matrix(matrix)
print("Transposed matrix:", transposed)
```

41. Write a Python Program to Sort Words in Alphabetic Order.

Explanation:

This program sorts a list of words in alphabetical order.

Logic:

- Use the sorted() function or the sort() method to sort the words.

Program:

```
words = ["banana", "apple", "cherry"]
sorted_words = sorted(words)
print("Sorted words:", sorted_words)
```

42. Write a Python Program to Remove Punctuation From a String.

Explanation:

This program removes all punctuation marks from a given string.

Logic:

- Use the string.punctuation constant to identify punctuation marks.
- Use a loop or list comprehension to filter out punctuation.

Program:

```
import string

def remove_punctuation(input_string):
    # Create a translation table to remove punctuation
    translator = str.maketrans('', '', string.punctuation)
    return input_string.translate(translator)

input_string = "Hello, World! This is a test."
result = remove_punctuation(input_string)
print("String without punctuation:", result)
```

43. Write a Python Program to Check if the Given Number is a Disarium Number.

Explanation:

A Disarium number is a number where the sum of its digits raised to the power of their respective positions equals the number itself.

Logic:

- Convert the number to a string to access individual digits and their positions.
- Calculate the sum of digits raised to the power of their positions.
- Compare the sum with the original number.

Program:

```python
def is_disarium(number):
    num_str = str(number)
    total = sum(int(digit) ** (index + 1) for index, digit in enumerate(num_str))
    return total == number

num = 175
if is_disarium(num):
    print(f"{num} is a Disarium number.")
else:
    print(f"{num} is not a Disarium number.")
```

44. Write a Python Program to Print All Disarium Numbers Between 1 to 100.

Explanation:

This program prints all Disarium numbers between 1 and 100.

Logic:

- Use the is_disarium() function from the previous program.
- Iterate through numbers from 1 to 100 and check if they are Disarium.

Program:

```python
def is_disarium(number):
    num_str = str(number)
    total = sum(int(digit) ** (index + 1) for index, digit in enumerate(num_str))
    return total == number

print("Disarium numbers between 1 and 100:")
for num in range(1, 101):
    if is_disarium(num):
        print(num, end=" ")
```

45. Write a Python Program to Check if the Given Number is a Happy Number.

Explanation:

A Happy number is a number that eventually reaches 1 when replaced by the sum of the squares of its digits. If it loops endlessly, it is not a Happy number.

Logic:

- Use a loop to repeatedly calculate the sum of the squares of the digits.
- Stop when the number becomes 1 (Happy) or enters a cycle (Not Happy).

Program:

```
def is_happy(number):
    seen = set()
    while number != 1 and number not in seen:
        seen.add(number)
        number = sum(int(digit) ** 2 for digit in str(number))
    return number == 1

num = 19
if is_happy(num):
    print(f"{num} is a Happy number.")
else:
    print(f"{num} is not a Happy number.")
```

46. Write a Python Program to Print All Happy Numbers Between 1 and 100.

Explanation:

This program prints all Happy numbers between 1 and 100.

Logic:

- Use the is_happy() function from the previous program.
- Iterate through numbers from 1 to 100 and check if they are Happy.

Program:

```
def is_happy(number):
    seen = set()
    while number != 1 and number not in seen:
        seen.add(number)
        number = sum(int(digit) ** 2 for digit in str(number))
    return number == 1

print("Happy numbers between 1 and 100:")
for num in range(1, 101):
    if is_happy(num):
        print(num, end=" ")
```

47. Write a Python Program to Determine Whether the Given Number is a Harshad Number.

Explanation:

A Harshad number is a number that is divisible by the sum of its digits.

Logic:

- Calculate the sum of the digits of the number.
- Check if the number is divisible by the sum.

Program:

```python
def is_harshad(number):
    if number == 0:
        return False
    digit_sum = sum(int(digit) for digit in str(number))
    return number % digit_sum == 0

num = 18
if is_harshad(num):
    print(f"{num} is a Harshad number.")
else:
    print(f"{num} is not a Harshad number.")
```

48. Write a Python Program to Print All Pronic Numbers Between 1 and 100.

Explanation:

A Pronic number is a number that is the product of two consecutive integers (e.g., 6=2×36=2×3).

Logic:

- Iterate through numbers from 1 to 100.
- Check if the number can be expressed as n×(n+1)$n\times(n+1)$.

Program:

```python
def is_pronic(number):
    for n in range(int(number**0.5) + 1):
        if n * (n + 1) == number:
            return True
    return False

print("Pronic numbers between 1 and 100:")
for num in range(1, 101):
    if is_pronic(num):
        print(num, end=" ")
```

49. Write a Python Program to Multiply All Numbers in the List.

Explanation:

This program multiplies all numbers in a given list.

Logic:

- Use a loop or the math.prod() function to calculate the product.

Program:

```
from math import prod

numbers = [1, 2, 3, 4, 5]
result = prod(numbers)
print("Product of numbers in the list:", result)
```

50. Write a Python Program to Find the Smallest Number in a List.

Explanation:

This program finds the smallest number in a given list.

Logic:

- Use the min() function or iterate through the list to find the smallest number.

Program:

```
numbers = [10, 20, 4, 45, 99]
smallest = min(numbers)
print("Smallest number in the list:", smallest)
```

51. Write a Python Program to Find the Largest Number in a List.

Explanation:

This program finds the largest number in a given list.

Logic:

- Use the max() function or iterate through the list to find the largest number.

Program:

```
numbers = [10, 20, 4, 45, 99]
largest = max(numbers)
print("Largest number in the list:", largest)
```

52. Write a Python Program to Find the Second Largest Number in a List.

Explanation:

This program finds the second largest number in a given list.

Logic:

- Sort the list in descending order and select the second element.

Program:

```
numbers = [10, 20, 4, 45, 99]
numbers_sorted = sorted(numbers, reverse=True)
second_largest = numbers_sorted[1]
print("Second largest number in the list:", second_largest)
```

53. Write a Python Program to Find N Largest Elements from a List.

Explanation:

This program finds the NN largest elements in a given list.

Logic:

- Sort the list in descending order and select the first NN elements.

Program:

```
numbers = [10, 20, 4, 45, 99]
N = 3
largest_elements = sorted(numbers, reverse=True)[:N]
print(f"{N} largest elements in the list:", largest_elements)
```

54. Write a Python Program to Print Even Numbers in a List.

Explanation:

This program prints all even numbers in a given list.

Logic:

- Use a loop or list comprehension to filter even numbers.

Program:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_numbers = [num for num in numbers if num % 2 == 0]
print("Even numbers in the list:", even_numbers)
```

55. Write a Python Program to Print Odd Numbers in a List.

Explanation:

This program prints all odd numbers in a given list.

Logic:

- Use a loop or list comprehension to filter odd numbers.

Program:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
odd_numbers = [num for num in numbers if num % 2 != 0]
print("Odd numbers in the list:", odd_numbers)
```

56. Write a Python Program to Remove Empty Lists from a List.

Explanation:

This program removes all empty lists from a given list.

Logic:

- Use a loop or list comprehension to filter out empty lists.

Program:

```python
lists = [[1, 2], [], [3, 4], [], [5]]
non_empty_lists = [lst for lst in lists if lst]
print("List after removing empty lists:", non_empty_lists)
```

57. Write a Python Program to Clone or Copy a List.

Explanation:

This program creates a copy of a given list.

Logic:

- Use the copy() method or slicing to create a copy.

Program:

```python
original_list = [1, 2, 3, 4, 5]
copied_list = original_list.copy()
```

58. Write a Python Program to Count Occurrences of an Element in a List.

Explanation:

This program counts how many times a given element appears in a list.

Logic:

- Use the count() method to count occurrences.

Program:

```python
numbers = [1, 2, 3, 4, 2, 2, 5]
element = 2
count = numbers.count(element)
print(f"Element {element} occurs {count} times in the list.")
```

59. Write a Python Program to Find Words Which Are Greater Than Given Length k*k*.

Explanation:

This program finds words in a list that are longer than a given length k*k*.

Logic:

- Use a loop or list comprehension to filter words based on their length.

Program:

```
words = ["apple", "banana", "cherry", "date", "elderberry"]
k = 5
long_words = [word for word in words if len(word) > k]
print(f"Words longer than {k} characters:", long_words)
```

60. Write a Python Program for Removing a Character from a String.

Explanation:

This program removes a specific character from a given string.

Logic:

- Use the replace() method or list comprehension to remove the character.

Program:

```
def remove_char(input_string, char):
    return input_string.replace(char, '')

input_string = "Hello, World!"
char_to_remove = "o"
result = remove_char(input_string, char_to_remove)
print("String after removing character:", result)
```

61. Write a Python Program to Split and Join a String.

Explanation:

This program splits a string into a list of words and then joins them back into a single string.

Logic:

- Use the split() method to split the string.
- Use the join() method to join the list into a string.

Program:

```
def split_and_join(input_string):
    words = input_string.split(" ")
    return "-".join(words)

input_string = "Hello World This is Python"
result = split_and_join(input_string)
print("String after splitting and joining:", result)
```

62. Write a Python Program to Check if a Given String is a Binary String.

Explanation:

This program checks if a string consists only of '0's and '1's.

Logic:

- Use the set() function to check if all characters are either '0' or '1'.

Program:

```python
def is_binary_string(input_string):
    return set(input_string).issubset({'0', '1'})

input_string = "1010101"
if is_binary_string(input_string):
    print("The string is a binary string.")
else:
    print("The string is not a binary string.")
```

63. Write a Python Program to Find Uncommon Words from Two Strings.

Explanation:

This program finds words that are not common between two strings.

Logic:

- Split both strings into words.
- Use set operations to find uncommon words.

Program:

```python
def uncommon_words(str1, str2):
    words1 = set(str1.split())
    words2 = set(str2.split())
    return words1.symmetric_difference(words2)

str1 = "Hello world"
str2 = "Hello Python"
result = uncommon_words(str1, str2)
print("Uncommon words:", result)
```

64. Write a Python Program to Find All Duplicate Characters in a String.

Explanation:

This program finds all characters that appear more than once in a string.

Logic:

- Use a dictionary to count character occurrences.
- Filter characters with counts greater than 1.

Program:

```
def find_duplicates(input_string):
    char_count = {}
    for char in input_string:
        char_count[char] = char_count.get(char, 0) + 1
    return [char for char, count in char_count.items() if count > 1]

input_string = "Hello World"
result = find_duplicates(input_string)
print("Duplicate characters:", result)
```

65. Write a Python Program to Check if a String Contains Any Special Character.

Explanation:

This program checks if a string contains any special characters.

Logic:

- Use the string.punctuation constant to identify special characters.
- Check if any character in the string is in the set of special characters.

Program:

```
import string

def contains_special_char(input_string):
    return any(char in string.punctuation for char in input_string)

input_string = "Hello@World!"
if contains_special_char(input_string):
    print("The string contains special characters.")
else:
    print("The string does not contain special characters.")
```

66. Write a Python Program to Extract Unique Dictionary Values.

Explanation:

This program extracts unique values from a dictionary.

Logic:

- Use the set() function to extract unique values.

Program:

```
my_dict = {'a': 1, 'b': 2, 'c': 1, 'd': 3}
unique_values = set(my_dict.values())
print("Unique values in the dictionary:", unique_values)
```

67. Write a Python Program to Find the Sum of All Items in a Dictionary.

Explanation:

This program calculates the sum of all values in a dictionary.

Logic:

- Use the sum() function to calculate the sum of dictionary values.

Program:

```
my_dict = {'a': 10, 'b': 20, 'c': 30}
total_sum = sum(my_dict.values())
print("Sum of all items in the dictionary:", total_sum)
```

68. Write a Python Program to Merge Two Dictionaries.

Explanation:

This program merges two dictionaries into one.

Logic:

- Use the update() method or dictionary unpacking (**) to merge dictionaries.

Program:

```
dict1 = {'a': 1, 'b': 2}
dict2 = {'c': 3, 'd': 4}

# Method 1: Using update()
merged_dict = dict1.copy()
merged_dict.update(dict2)

# Method 2: Using dictionary unpacking
merged_dict = {**dict1, **dict2}

print("Merged dictionary:", merged_dict)
```

69. Write a Python Program to Convert Key-Values List to a Flat Dictionary.

Explanation:

This program converts a list of key-value pairs into a flat dictionary.

Logic:

- Use dictionary comprehension or the dict() constructor.

Program:

```python
keys = ['a', 'b', 'c']
values = [1, 2, 3]

# Using zip() and dict()
flat_dict = dict(zip(keys, values))

print("Flat dictionary:", flat_dict)
```

70. Define a Class with a Generator to Iterate Numbers Divisible by 7 Between 0 and n*n*.

Explanation:

This program defines a class with a generator that yields numbers divisible by 7 within a given range.

Logic:

- Use a generator function (yield) to iterate through numbers and yield those divisible by 7.

Program:

```python
class DivisibleBySeven:
    def __init__(self, n):
        self.n = n

    def generate(self):
        for num in range(self.n + 1):
            if num % 7 == 0:
                yield num

n = 50
generator = DivisibleBySeven(n).generate()
print("Numbers divisible by 7 between 0 and", n, ":", list(generator))
```

71. Define a Class Person and Its Two Child Classes: Male and Female.

Explanation:

This program defines a base class Person and two child classes Male and Female, each with a method getGender.

Logic:

- Use inheritance to create child classes.
- Override the getGender method in each child class.

Program:

```
class Person:
    def getGender(self):
        return "Unknown"

class Male(Person):
    def getGender(self):
        return "Male"

class Female(Person):
    def getGender(self):
        return "Female"

male = Male()
female = Female()
print("Male:", male.getGender())
print("Female:", female.getGender())
```

72. Generate All Sentences with Given Subjects, Verbs, and Objects.

Explanation:

This program generates all possible sentences using given lists of subjects, verbs, and objects.

Logic:

- Use nested loops or list comprehensions to combine subjects, verbs, and objects.

Program:

```
subjects = ["I", "You"]
verbs = ["Play", "Love"]
objects = ["Hockey", "Football"]

sentences = [f"{s} {v} {o}." for s in subjects for v in verbs for o in objects]
for sentence in sentences:
    print(sentence)
```

73. Compress and Decompress a String.

Explanation:

This program compresses a string by removing duplicates and decompresses it back to the original form.

Logic:

- Use zlib for compression and decompression.

Program:

```python
import zlib

def compress_string(input_string):
    return zlib.compress(input_string.encode())

def decompress_string(compressed_string):
    return zlib.decompress(compressed_string).decode()

input_string = "hello world!hello world!hello world!hello world!"
compressed = compress_string(input_string)
decompressed = decompress_string(compressed)
print("Compressed:", compressed)
print("Decompressed:", decompressed)
```

74. Binary Search Function for a Sorted List.

Explanation:

This program implements a binary search algorithm to find an item in a sorted list.

Logic:

- Use a loop to repeatedly divide the list into halves and search for the item.

Program:

```python
def binary_search(sorted_list, item):
    low = 0
    high = len(sorted_list) - 1

    while low <= high:
        mid = (low + high) // 2
        guess = sorted_list[mid]
        if guess == item:
            return mid
        if guess > item:
            high = mid - 1
        else:
            low = mid + 1
    return None

sorted_list = [1, 3, 5, 7, 9]
item = 5
index = binary_search(sorted_list, item)
print("Index of", item, ":", index)
```

75. Generator to Print Numbers Divisible by 5 and 7 Between 0 and n*n*.

Explanation:

This program uses a generator to print numbers divisible by both 5 and 7 within a given range.

Logic:

- Use a generator function to yield numbers divisible by both 5 and 7.

Program:

```python
def divisible_by_5_and_7(n):
    for num in range(n + 1):
        if num % 5 == 0 and num % 7 == 0:
            yield num

n = 100
result = list(divisible_by_5_and_7(n))
print("Numbers divisible by 5 and 7 between 0 and", n, ":", ",".join(map(str, result)))
```

76. Generator to Print Even Numbers Between 0 and n*n*.

Explanation:

This program uses a generator to print even numbers within a given range.

Logic:

- Use a generator function to yield even numbers.

Program:

```python
def even_numbers(n):
    for num in range(n + 1):
        if num % 2 == 0:
            yield num

n = 10
result = list(even_numbers(n))
print("Even numbers between 0 and", n, ":", ",".join(map(str, result)))
```

77. Fibonacci Sequence Using List Comprehension.

Explanation:

This program generates the Fibonacci sequence using list comprehension.

Logic:

- Use list comprehension to generate Fibonacci numbers based on the formula:

f(n)=f(n−1)+f(n−2)$f(n)=f(n-1)+f(n-2)$

Program:

```
def fibonacci(n):
    return [fib(i) for i in range(n + 1)]

def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n - 1) + fib(n - 2)

n = 8
result = fibonacci(n)
print("Fibonacci sequence up to", n, ":", ",".join(map(str, result)))
```

78. Extract Username from an Email Address.

Explanation:

This program extracts the username from an email address.

Logic:

- Split the email address at the @ symbol and take the first part.

Program:

```
def extract_username(email):
    return email.split('@')[0]

email = "john@google.com"
username = extract_username(email)
print("Username:", username)
```

79. Using List Comprehensions, Create a Function That Finds All Even Numbers from 1 to the Given Number.

Explanation:

This program uses list comprehension to find all even numbers between 1 and a given number.

Logic:

- Use list comprehension to filter even numbers using the condition num % 2 == 0.

Program:

```python
def find_even_numbers(n):
    return [num for num in range(1, n + 1) if num % 2 == 0]

n = 10
print("Even numbers from 1 to", n, ":", find_even_numbers(n))
```

80. Create a Function That Takes a List of Strings and Integers, and Filters Out the List So That It Returns a List of Integers Only.

Explanation:

This program filters out integers from a list containing both strings and integers.

Logic:

- Use list comprehension with isinstance() to check if an element is an integer.

Program:

```python
def filter_integers(mixed_list):
    return [item for item in mixed_list if isinstance(item, int)]

mixed_list = [1, "apple", 2, "banana", 3, "cherry"]
print("Filtered integers:", filter_integers(mixed_list))
```

81. Given a List of Numbers, Create a Function Which Returns the List but with Each Element's Index Added to Itself.

Explanation:

This program adds the index of each element to the element itself.

Logic:

- Use list comprehension to add the index to each element.

Program:

```python
def add_index_to_elements(numbers):
    return [num + idx for idx, num in enumerate(numbers)]

numbers = [1, 2, 3, 4, 5]
print("List with index added:", add_index_to_elements(numbers))
```

82. Create a Function That Takes the Height and Radius of a Cone as Arguments and Returns the Volume of the Cone Rounded to the Nearest Hundredth.

Explanation:

This program calculates the volume of a cone using the formula:

$$\text{Volume} = \frac{1}{3}\pi r^2 h$$

Logic:

- Use the formula and round the result to the nearest hundredth.

Program:

```python
import math

def cone_volume(radius, height):
    volume = (1/3) * math.pi * (radius ** 2) * height
    return round(volume, 2)

radius = 3
height = 5
print("Volume of the cone:", cone_volume(radius, height))
```

83. Create a Function That Takes a List of Numbers Between 1 and 10 (Excluding One Number) and Returns the Missing Number.

Explanation:

This program finds the missing number in a list of numbers from 1 to 10.

Logic:

- Use set difference to find the missing number.

Program:

```python
def find_missing_number(numbers):
    full_set = set(range(1, 11))
    given_set = set(numbers)
    return (full_set - given_set).pop()

numbers = [1, 2, 3, 4, 6, 7, 8, 9, 10]
print("Missing number:", find_missing_number(numbers))
```

84. Write a Function That Takes a List and a Number as Arguments. Add the Number to the End of the List, Then Remove the First Element of the List. The Function Should Then Return the Updated List.

Explanation:

This program adds a number to the end of the list and removes the first element.

Logic:

- Use append() to add the number and pop(0) to remove the first element.

Program:

```python
def update_list(lst, num):
    lst.append(num)
    lst.pop(0)
    return lst

lst = [1, 2, 3, 4]
num = 5
print("Updated list:", update_list(lst, num))
```

85. Create a Function That Takes a String and Returns a String with Its Letters in Alphabetical Order.

Explanation:

This program sorts the letters of a string in alphabetical order.

Logic:

- Use the sorted() function to sort the characters and join them back into a string.

Program:

```python
def sort_string_alphabetically(input_string):
    return ''.join(sorted(input_string))

input_string = "python"
print("Sorted string:", sort_string_alphabetically(input_string))
```

86. Create a Function That Takes in Two Lists and Returns True if the Second List Follows the First List by One Element, and False Otherwise.

Explanation:

This program checks if the second list is the first list shifted to the right by one element.

Logic:

- Compare the second list with the first list shifted by one.

Program:

```
def is_shifted(list1, list2):
    return list2 == list1[1:] + [list1[0]]

list1 = [1, 2, 3, 4]
list2 = [2, 3, 4, 1]
print("Is list2 a shifted version of list1?", is_shifted(list1, list2))
```

87. A Group of Friends Have Decided to Start a Secret Society. The Name Will Be the First Letter of Each of Their Names, Sorted in Alphabetical Order. Create a Function That Takes in a List of Names and Returns the Name of the Secret Society.

Explanation:

This program generates the name of a secret society by taking the first letter of each name, sorting them alphabetically, and joining them.

Logic:

- Extract the first letter of each name, sort them, and join into a single string.

Program:

```
def secret_society_name(names):
    return ''.join(sorted([name[0] for name in names]))

names = ["Alice", "Bob", "Charlie", "David"]
print("Secret society name:", secret_society_name(names))
```