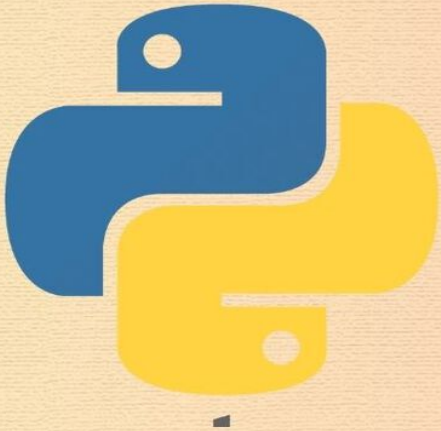
2023 NEW EDITION

FOR BEGINNERS



PYTHON

100 PROGRAMS

BEST FOR COMPUTER SCIENCE STUDENTS

DEVBRAT RUDRA

PYTHON

100 Program

By- Devbrat Rudra

www.rudyhelp.com

SET-1

1. Hello World Program:

```
print("Hello, World!")
```

2. Simple Calculator:

```
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
print("Sum:", a + b)
print("Difference:", a - b)
print("Product:", a * b)
print("Quotient:", a / b)
```

3. Factorial of a Number:

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

```
        return n * factorial(n - 1)

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

4. Fibonacci Sequence:

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

terms = int(input("Enter the number of terms: "))
print("Fibonacci sequence:")
for i in range(terms):
    print(fibonacci(i))
```

5. Check for Prime Number:

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True

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

```
if is_prime(num):
    print("Prime")
else:
    print("Not prime")
```

6. Simple Interest Calculator:

```
p = float(input("Enter the principal amount: "))
r = float(input("Enter the rate of interest: "))
t = float(input("Enter the time period: "))
interest = (p * r * t) / 100
print("Simple Interest:", interest)
```

7. Check for Even or Odd:

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

8. Area of a Circle:

```
import math

radius = float(input("Enter the radius of the circle: "))
area = math.pi * radius * radius
print("Area:", area)
```

9. List Comprehension:

```
squares = [i ** 2 for i in range(10)]
print("Squares:", squares)
```

10. Simple File Handling:

```
# Writing to a file
with open("output.txt", "w") as file:
    file.write("Hello, this is a sample text.")

# Reading from a file
with open("output.txt", "r") as file:
    data = file.read()
    print("Data from file:", data)
```

SET-2

1. Check for Palindrome:

```
def is_palindrome(s):
    return s == s[::-1]

string = input("Enter a string: ")
if is_palindrome(string):
    print("Palindrome")
else:
    print("Not a palindrome")
```

2. Find the Largest Among Three Numbers:

```
a = float(input("Enter the first number: "))
b = float(input("Enter the second number: "))
c = float(input("Enter the third number: "))
max_num = max(a, b, c)
print("Largest number:", max_num)
```

3. Print Multiplication Table:

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

4. Convert Celsius to Fahrenheit:

```
celsius = float(input("Enter temperature in Celsius: "))
fahrenheit = (celsius * 9/5) + 32
print("Temperature in Fahrenheit:", fahrenheit)
```

5. Simple String Operations:

```
string = "Hello, World!"
print("Length of the string:", len(string))
print("Uppercase:", string.upper())
print("Lowercase:", string.lower())
print("Reversed string:", string[::-1])
```

6. Bubble Sort Algorithm:

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

arr = [64, 34, 25, 12, 22, 11, 90]
bubble_sort(arr)
print("Sorted array:", arr)
```

7. Check Leap Year:

```
def is_leap_year(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    return False

year = int(input("Enter a year: "))
if is_leap_year(year):
    print("Leap year")
else:
    print("Not a leap year")
```

8. Count Vowels in a String:

```
def count_vowels(s):
    vowels = 'aeiouAEIOU'
    count = 0
```

```
    for char in s:
        if char in vowels:
            count += 1
    return count

string = input("Enter a string: ")
print("Number of vowels:", count_vowels(string))
```

9. Find the LCM of Two Numbers:

```
def compute_lcm(x, y):
    if x > y:
        greater = x
    else:
        greater = y

    while True:
        if greater % x == 0 and greater % y == 0:
            lcm = greater
            break
        greater += 1

    return lcm

num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
print("LCM:", compute_lcm(num1, num2))
```

10. Basic Class and Object:

```
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

length = float(input("Enter length of the rectangle: "))
width = float(input("Enter width of the rectangle: "))
rect = Rectangle(length, width)
print("Area of the rectangle:", rect.area())
```

SET-3

1. Check Anagram:

```
def is_anagram(s1, s2):
    return sorted(s1) == sorted(s2)

string1 = input("Enter the first string: ")
string2 = input("Enter the second string: ")
if is_anagram(string1, string2):
    print("Anagrams")
else:
    print("Not anagrams")
```

2. Generate a Random Number:

```
import random
```

```
print("Random number:", random.randint(1, 100))
```

3. Binary Search Algorithm:

```
def binary_search(arr, x):
    low = 0
    high = len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] < x:
            low = mid + 1
        elif arr[mid] > x:
            high = mid - 1
        else:
            return mid
    return -1

arr = [2, 3, 4, 10, 40]
x = 10
result = binary_search(arr, x)
if result != -1:
    print(f"Element found at index {result}")
else:
    print("Element not found")
```

4. Check Armstrong Number:

```
def is_armstrong(n):
```

```
    order = len(str(n))
    temp = n
    sum = 0
    while temp > 0:
        digit = temp % 10
        sum += digit ** order
        temp //= 10
    return n == sum

number = int(input("Enter a number: "))
if is_armstrong(number):
    print("Armstrong number")
else:
    print("Not an Armstrong number")
```

5. Generate a Simple Pattern:

```
n = 5
for i in range(n):
    print('* ' * (i + 1))
```

6. Linear Search Algorithm:

```
def linear_search(arr, x):
    for i in range(len(arr)):
        if arr[i] == x:
            return i
    return -1
```

```
arr = [4, 2, 1, 7, 5]
x = 7
result = linear_search(arr, x)
if result != -1:
    print(f"Element found at index {result}")
else:
    print("Element not found")
```

7. Calculate the Power of a Number:

```
base = int(input("Enter the base: "))
exponent = int(input("Enter the exponent: "))
result = base ** exponent
print("Result:", result)
```

8. Print the Fibonacci Series:

```
def fibonacci_series(n):
    a, b = 0, 1
    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b

terms = int(input("Enter the number of terms: "))
print("Fibonacci series:")
fibonacci_series(terms)
```

9. Merge Two Sorted Lists:

```
list1 = [1, 3, 5, 7]
list2 = [2, 4, 6, 8]
merged_list = sorted(list1 + list2)
print("Merged and sorted list:", merged_list)
```

10. Generate a Simple Pyramid Pattern:

```
n = 5
for i in range(n):
    print(" " * (n - i - 1) + "*" * (2 * i + 1))
```

SET-4

1. Check if a Number is Positive, Negative, or Zero:

```
num = float(input("Enter a number: "))
if num > 0:
    print("Positive number")
elif num < 0:
    print("Negative number")
else:
    print("Zero")
```

2. Generate a List of Prime Numbers within a Range:

```
def generate_primes(start, end):
    primes = []
    for num in range(start, end + 1):
        if num > 1:
```

```
            for i in range(2, num):
                if num % i == 0:
                    break
            else:
                primes.append(num)
    return primes

start_range = int(input("Enter the starting range: "))
end_range = int(input("Enter the ending range: "))
print("Prime numbers:", generate_primes(start_range, end_range))
```

3. Calculate the Area and Perimeter of a Rectangle:

```
length = float(input("Enter the length of the rectangle: "))
width = float(input("Enter the width of the rectangle: "))
area = length * width
perimeter = 2 * (length + width)
print(f"Area: {area}, Perimeter: {perimeter}")
```

4. Find the GCD of Two Numbers:

```
def compute_gcd(x, y):
    while y:
        x, y = y, x % y
    return x

num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
print("GCD:", compute_gcd(num1, num2))
```

5. Check if a Year is a Leap Year or Not Using Functions:

```
def is_leap_year(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True
            else:
                return False
        else:
            return True
    else:
        return False

year = int(input("Enter a year: "))
if is_leap_year(year):
    print("Leap year")
else:
    print("Not a leap year")
```

6. Print the Sum of Natural Numbers up to a Given Number:

```
n = int(input("Enter a number: "))
sum = 0
for i in range(1, n + 1):
    sum += i
print("Sum of natural numbers:", sum)
```

7. Reverse a String:

```
string = input("Enter a string: ")
reversed_string = string[::-1]
print("Reversed string:", reversed_string)
```

8. Check if a Number is a Perfect Number:

```
def is_perfect_number(n):
    sum = 0
    for i in range(1, n):
        if n % i == 0:
            sum += i
    return sum == n

number = int(input("Enter a number: "))
if is_perfect_number(number):
    print("Perfect number")
else:
    print("Not a perfect number")
```

9. Count the Number of Words in a String:

```
string = input("Enter a string: ")
word_count = len(string.split())
print("Number of words:", word_count)
```

10. Concatenate Two Strings:

```
string1 = input("Enter the first string: ")
string2 = input("Enter the second string: ")
concatenated_string = string1 + string2
print("Concatenated string:", concatenated_string)
```

SET-5

1. Check if a Number is a Perfect Square:

```
import math

def is_perfect_square(n):
    root = math.isqrt(n)
    return root * root == n

number = int(input("Enter a number: "))
if is_perfect_square(number):
    print("Perfect square")
else:
    print("Not a perfect square")
```

2. Implement a Stack Data Structure:

```
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)
```

```
    def pop(self):
        return self.items.pop()

    def is_empty(self):
        return self.items == []

stack = Stack()
stack.push(1)
stack.push(2)
stack.push(3)
print("Popped item:", stack.pop())
print("Stack is empty:", stack.is_empty())
```

3. Calculate the Area of a Triangle:

```
base = float(input("Enter the base of the triangle: "))
height = float(input("Enter the height of the triangle: "))
area = 0.5 * base * height
print("Area of the triangle:", area)
```

4. Find the ASCII Value of a Character:

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

5. Generate a Simple Diamond Pattern:

```
n = 5
```

```
for i in range(n):
    print(" " * (n - i - 1) + "* " * (i + 1))
for i in range(n - 1, 0, -1):
    print(" " * (n - i) + "* " * i)
```

6. Check if a Number is a Perfect Cube:

```
def is_perfect_cube(n):
    root = round(n ** (1/3))
    return root ** 3 == n

number = int(input("Enter a number: "))
if is_perfect_cube(number):
    print("Perfect cube")
else:
    print("Not a perfect cube")
```

7. Implement a Queue Data Structure:

```
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        self.items.insert(0, item)

    def dequeue(self):
        return self.items.pop()

    def is_empty(self):
```

```
        return self.items == []

queue = Queue()
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
print("Dequeued item:", queue.dequeue())
print("Queue is empty:", queue.is_empty())
```

8. Calculate the Power Set of a Set:

```
from itertools import chain, combinations

def power_set(s):
    return list(chain.from_iterable(combinations(s, r) for r in range(len(s) + 1)))

input_set = [1, 2, 3]
print("Power set:", power_set(input_set))
```

9. Swap Two Variables:

```
a = input("Enter the value of a: ")
b = input("Enter the value of b: ")
a, b = b, a
print("Value of a after swapping:", a)
print("Value of b after swapping:", b)
```

10. Print the Factors of a Number:

```
def print_factors(n):
    factors = []
    for i in range(1, n + 1):
        if n % i == 0:
            factors.append(i)
    return factors

number = int(input("Enter a number: "))
print("Factors:", print_factors(number))
```

SET-6

1. Check if a String is a Pangram:

```
import string

def is_pangram(s):
    alphabet = set(string.ascii_lowercase)
    return set(s.lower()) >= alphabet

input_string = input("Enter a string: ")
if is_pangram(input_string):
    print("Pangram")
else:
    print("Not a pangram")
```

2. Calculate the Volume of a Cylinder:

```
import math
```

```
radius = float(input("Enter the radius of the cylinder: "))
height = float(input("Enter the height of the cylinder: "))
volume = math.pi * radius * radius * height
print("Volume of the cylinder:", volume)
```

3. Check if a String is a Palindrome:

```
def is_palindrome(s):
    return s == s[::-1]

input_string = input("Enter a string: ")
if is_palindrome(input_string):
    print("Palindrome")
else:
    print("Not a palindrome")
```

4. Sort a List of Strings:

```
strings = ['apple', 'banana', 'cherry', 'date', 'elderberry']
sorted_strings = sorted(strings)
print("Sorted strings:", sorted_strings)
```

5. Generate a Simple Pascal's Triangle:

```
def generate_pascals_triangle(n):
    triangle = [[1]]
    for i in range(1, n):
        prev_row = triangle[-1]
```

```
        curr_row = [1] + [prev_row[j] + prev_row[j + 1] for j in range(i - 1)]
+ [1]
        triangle.append(curr_row)
    return triangle

rows = 5
print("Pascal's Triangle:")
for row in generate_pascals_triangle(rows):
    print(row)
```

6. Implement a Binary Search Tree:

```
class Node:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

class BinaryTree:
    def __init__(self):
        self.root = None

    def insert(self, value):
        if self.root is None:
            self.root = Node(value)
        else:
            self._insert_recursive(self.root, value)

    def _insert_recursive(self, node, value):
        if value < node.value:
```

```
            if node.left is None:
                node.left = Node(value)
            else:
                self._insert_recursive(node.left, value)
        elif value > node.value:
            if node.right is None:
                node.right = Node(value)
            else:
                self._insert_recursive(node.right, value)

# Example usage:
tree = BinaryTree()
tree.insert(5)
tree.insert(3)
tree.insert(7)
```

7. Implement a Linear Regression Model:

```
from sklearn.linear_model import LinearRegression
import numpy as np

X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
y = np.dot(X, np.array([1, 2])) + 3
reg = LinearRegression().fit(X, y)
print("Coef:", reg.coef_)
print("Intercept:", reg.intercept_)
```

8. Count the Number of Digits in an Integer:

```
number = int(input("Enter an integer: "))
num_digits = len(str(abs(number)))
print("Number of digits:", num_digits)
```

9. Generate a Random Password:

```
import random
import string

def generate_password(length):
    characters = string.ascii_letters + string.digits + string.punctuation
    password = ''.join(random.choice(characters) for _ in range(length))
    return password

password_length = 12
print("Generated password:", generate_password(password_length))
```

10. Calculate the Exponential Value:

```
base = float(input("Enter the base: "))
exponent = float(input("Enter the exponent: "))
result = base ** exponent
print("Result:", result)
```

SET-7

1. Find the Sum of Natural Numbers Using Recursion:

```
def sum_of_natural_numbers(n):
    if n <= 1:
```

```
        return n
    else:
        return n + sum_of_natural_numbers(n - 1)

num = int(input("Enter a number: "))
print("Sum of natural numbers:", sum_of_natural_numbers(num))
```

2. Validate an IP Address:

```
import socket

def is_valid_ip(ip):
    try:
        socket.inet_aton(ip)
        return True
    except socket.error:
        return False

ip_address = input("Enter an IP address: ")
if is_valid_ip(ip_address):
    print("Valid IP address")
else:
    print("Invalid IP address")
```

3. Calculate the Greatest Common Divisor (GCD) Using Recursion:

```
def gcd(x, y):
    if y == 0:
        return x
```

```
    else:
        return gcd(y, x % y)

num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
print("GCD:", gcd(num1, num2))
```

4. Implement a Queue using a List:

```
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        self.items.insert(0, item)

    def dequeue(self):
        if self.items:
            return self.items.pop()
        return None

    def is_empty(self):
        return self.items == []

queue = Queue()
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
print("Dequeued item:", queue.dequeue())
print("Queue is empty:", queue.is_empty())
```

5. Calculate the Power Set of a Set using Iterative Approach:

```
def power_set_iterative(s):
    power_set = [[]]
    for elem in s:
        for sub_set in power_set[:]:
            power_set.append(sub_set + [elem])
    return power_set

input_set = [1, 2, 3]
print("Power set (iterative):", power_set_iterative(input_set))
```

6. Print the Calendar of a Given Month and Year:

```
import calendar

year = int(input("Enter the year: "))
month = int(input("Enter the month: "))
print(calendar.month(year, month))
```

7. Find the Median of Three Values:

```
def find_median(a, b, c):
    return sorted([a, b, c])[1]

num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))
print("Median:", find_median(num1, num2, num3))
```

8. Implement a Binary Search Algorithm Using Recursion:

```
def binary_search_recursive(arr, low, high, x):
    if high >= low:
        mid = (high + low) // 2
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            return binary_search_recursive(arr, low, mid - 1, x)
        else:
            return binary_search_recursive(arr, mid + 1, high, x)
    else:
        return -1

arr = [2, 3, 4, 10, 40]
x = 10
result = binary_search_recursive(arr, 0, len(arr) - 1, x)
if result != -1:
    print(f"Element found at index {result}")
else:
    print("Element not found")
```

9. Find the Sum of Digits in a Number:

```
def sum_of_digits(n):
    return sum(int(digit) for digit in str(n))

number = int(input("Enter a number: "))
print("Sum of digits:", sum_of_digits(number))
```

10. Convert Decimal to Binary, Octal, and Hexadecimal:

```
dec = int(input("Enter a decimal number: "))
print("Binary:", bin(dec))
print("Octal:", oct(dec))
print("Hexadecimal:", hex(dec))
```

SET-8

1. Implement Selection Sort:

```
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        min_idx = i
        for j in range(i + 1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]

arr = [64, 25, 12, 22, 11]
selection_sort(arr)
print("Sorted array:", arr)
```

2. Find the Greatest Among Three Numbers:

```
a = float(input("Enter the first number: "))
b = float(input("Enter the second number: "))
c = float(input("Enter the third number: "))
```

```
max_num = max(a, b, c)
print("Greatest number:", max_num)
```

3. Implement Insertion Sort:

```
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key

arr = [12, 11, 13, 5, 6]
insertion_sort(arr)
print("Sorted array:", arr)
```

4. Convert Decimal to Binary:

```
dec = int(input("Enter a decimal number: "))
binary = bin(dec)
print("Binary:", binary[2:])
```

5. Convert Decimal to Octal:

```
dec = int(input("Enter a decimal number: "))
octal = oct(dec)
print("Octal:", octal[2:])
```

6. Convert Decimal to Hexadecimal:

```
dec = int(input("Enter a decimal number: "))
hexadecimal = hex(dec)
print("Hexadecimal:", hexadecimal[2:])
```

7. Implement a Bubble Sort Algorithm:

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

arr = [64, 34, 25, 12, 22, 11, 90]
bubble_sort(arr)
print("Sorted array:", arr)
```

8. Find the LCM and GCD of Two Numbers:

```
def compute_lcm(x, y):
    lcm = (x * y) // compute_gcd(x, y)
    return lcm

def compute_gcd(x, y):
    while y:
        x, y = y, x % y
    return x
```

```
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
print("LCM:", compute_lcm(num1, num2))
print("GCD:", compute_gcd(num1, num2))
```

9. Find the Factorial of a Number:

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

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

10. Implement Quick Sort:

```
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)

arr = [12, 11, 13, 5, 6, 7]
sorted_arr = quick_sort(arr)
```

```
print("Sorted array:", sorted_arr)
```

SET-9

1. Find the Sum of Elements in a List:

```
my_list = [1, 2, 3, 4, 5]
sum_of_elements = sum(my_list)
print("Sum of elements:", sum_of_elements)
```

2. Generate a Fibonacci Sequence Using a Loop:

```
def generate_fibonacci(n):
    fibonacci_sequence = [0, 1]
    for i in range(2, n):
        next_num = fibonacci_sequence[-1] + fibonacci_sequence[-2]
        fibonacci_sequence.append(next_num)
    return fibonacci_sequence

terms = 10
print("Fibonacci sequence:", generate_fibonacci(terms))
```

3. Calculate the Exponential Value Using a Loop:

```
base = int(input("Enter the base: "))
exponent = int(input("Enter the exponent: "))
result = 1
for _ in range(exponent):
    result *= base
```

```
print("Result:", result)
```

4. Implement Linear Search Algorithm Using a Loop:

```
def linear_search(arr, x):
    for i in range(len(arr)):
        if arr[i] == x:
            return i
    return -1

arr = [4, 2, 1, 7, 5]
x = 7
result = linear_search(arr, x)
if result != -1:
    print(f"Element found at index {result}")
else:
    print("Element not found")
```

5. Calculate the Area of a Triangle Using Heron's Formula:

```
import math

a = float(input("Enter the length of side a: "))
b = float(input("Enter the length of side b: "))
c = float(input("Enter the length of side c: "))
s = (a + b + c) / 2
area = math.sqrt(s * (s - a) * (s - b) * (s - c))
print("Area of the triangle:", area)
```

6. Implement a Merge Sort Algorithm:

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]

        merge_sort(left_half)
        merge_sort(right_half)

        i = j = k = 0

        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1

        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
```

```
            k += 1

arr = [12, 11, 13, 5, 6, 7]
merge_sort(arr)
print("Sorted array:", arr)
```

7. Find the Area of a Circle:

```
import math

radius = float(input("Enter the radius of the circle: "))
area = math.pi * radius * radius
print("Area of the circle:", area)
```

8. Implement a Binary Search Algorithm Using a Loop:

```
def binary_search(arr, x):
    low = 0
    high = len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] < x:
            low = mid + 1
        elif arr[mid] > x:
            high = mid - 1
        else:
            return mid
    return -1
```

```
arr = [2, 3, 4, 10, 40]
x = 10
result = binary_search(arr, x)
if result != -1:
    print(f"Element found at index {result}")
else:
    print("Element not found")
```

9. Check if a String is a Valid Email Address:

```
import re

def is_valid_email(email):
    return bool(re.match(r"[^@]+@[^@]+\.[^@]+", email))

input_email = input("Enter an email address: ")
if is_valid_email(input_email):
    print("Valid email address")
else:
    print("Invalid email address")
```

10. Generate a Random List of Numbers:

```
import random

random_list = random.sample(range(1, 100), 5)
print("Random list:", random_list)
```

SET-10

1. Find the Greatest Common Divisor (GCD) of Multiple Numbers:

```
import math

numbers = [24, 36, 48, 60, 72]
gcd = math.gcd(*numbers)
print("GCD of the numbers:", gcd)
```

2. Calculate the Standard Deviation of a List of Numbers:

```
import statistics

data = [1, 2, 3, 4, 5]
std_dev = statistics.stdev(data)
print("Standard deviation:", std_dev)
```

3. Generate a Random Password with Specific Requirements:

```
import random
import string

def generate_password(length, include_digits=True,
include_special_chars=True):
    characters = string.ascii_letters
    if include_digits:
        characters += string.digits
    if include_special_chars:
        characters += string.punctuation
    password = ''.join(random.choice(characters) for _ in range(length))
    return password
```

```
password_length = 12
print("Generated password:", generate_password(password_length))
```

4. Implement a Simple Calculator:

```
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):
    if y == 0:
        return "Cannot divide by zero"
    return x / y

num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
print("Sum:", add(num1, num2))
print("Difference:", subtract(num1, num2))
print("Product:", multiply(num1, num2))
print("Quotient:", divide(num1, num2))
```

5. Check if a Number is a Prime Number:

```
def is_prime(n):
    if n <= 1:
```

```
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

number = int(input("Enter a number: "))
if is_prime(number):
    print("Prime number")
else:
    print("Not a prime number")
```

6. Sort a List of Dictionaries by a Specific Key:

```
list_of_dicts = [{'name': 'John', 'age': 30}, {'name': 'Jane', 'age': 25}, {'name': 'Bob', 'age': 35}]
sorted_list = sorted(list_of_dicts, key=lambda x: x['age'])
print("Sorted list of dictionaries:", sorted_list)
```

7. Generate a Random Matrix:

```
import numpy as np

rows = 3
cols = 3
random_matrix = np.random.rand(rows, cols)
print("Random matrix:")
print(random_matrix)
```

8. Implement a Counter Class:

```
class Counter:
    def __init__(self):
        self.count = 0

    def increment(self):
        self.count += 1

    def decrement(self):
        self.count -= 1

    def reset(self):
        self.count = 0

counter = Counter()
counter.increment()
counter.increment()
print("Count:", counter.count)
counter.decrement()
print("Count:", counter.count)
counter.reset()
print("Count:", counter.count)
```

9. Find the Area of a Rectangle:

```
length = float(input("Enter the length of the rectangle: "))
width = float(input("Enter the width of the rectangle: "))
area = length * width
print("Area of the rectangle:", area)
```

10. Check if a String is a Valid URL:

```
import re

def is_valid_url(url):
    regex = re.compile(
        r'^(?:http|ftp)s?://'
        r'(?:(?:[A-Z0-9](?:[A-Z0-9-]{0,61}[A-Z0-9])?\.)+(?:[A-Z]{2,6}\.?|
[A-Z0-9-]{2,}\.?)|'
        r'localhost|'
        r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}|'
        r'\[?[A-F0-9]*:[A-F0-9:]+\]?)'
        r'(?::\d+)?'
        r'(?:/?|[/?]\S+)$', re.IGNORECASE)
    return re.match(regex, url) is not None

input_url = input("Enter a URL: ")
if is_valid_url(input_url):
    print("Valid URL")
else:
    print("Invalid URL")
```

Thank You