**1. Python program to check whether the given integer is a prime number or not.**

def is\_prime(n):

    if n < 2:

        return False

    for i in range(2, n):

        if n % i == 0:

            return False

    return True

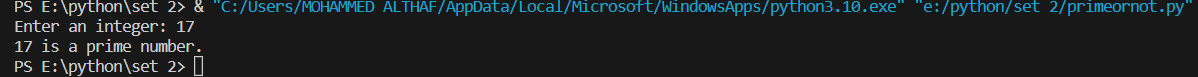
num = int(input("Enter an integer: "))

if is\_prime(num):

    print(f"{num} is a prime number.")

else:

    print(f"{num} is not a prime number.")



**2. Python program to generate the prime numbers from 1 to N.**

def generate\_primes(N):

    primes = []

    for num in range(2, N + 1):

        is\_prime = True

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

            if num % i == 0:

                is\_prime = False

                break

        if is\_prime:

            primes.append(num)

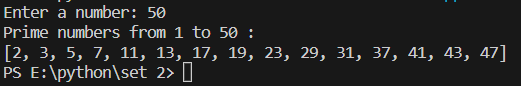
    return primes

N = int(input("Enter a number: "))

prime\_numbers = generate\_primes(N)

print("Prime numbers from 1 to", N, ":")

print(prime\_numbers)



**4. Python program to print the numbers from a given number n till 0 using recursion.**

def print\_numbers(n):

    if n < 0:

        return

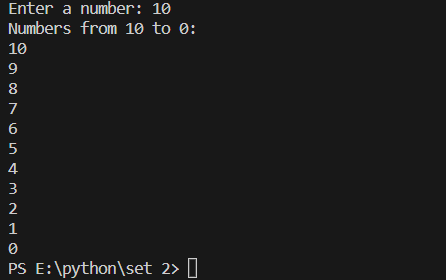
    print(n)

    print\_numbers(n-1)

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

print("Numbers from", num, "to 0:")

print\_numbers(num)



**5. Python program to display the sum of n numbers using a list.**

def sum\_of\_numbers(numbers):

    return sum(numbers)

n = int(input("Enter the count of numbers: "))

numbers = []

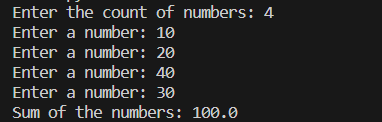
for i in range(n):

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

    numbers.append(number)

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

print("Sum of the numbers:", total\_sum)



**6. Python program to implement linear search**

def linear\_search(arr, target):

    for i in range(len(arr)):

        if arr[i] == target:

            return i

    return -1

numbers = [5, 2, 8, 10, 3, 7]

print(numbers)

target\_number = int(input("Enter the number to search: "))

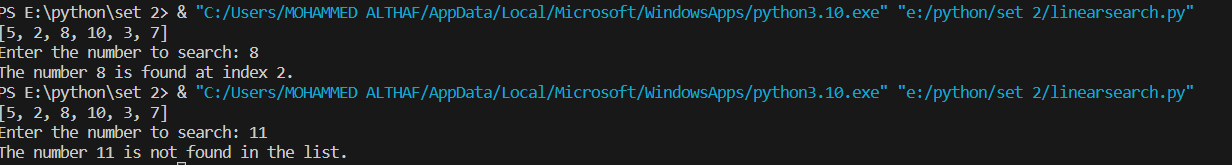
index = linear\_search(numbers, target\_number)

if index != -1:

    print(f"The number {target\_number} is found at index {index}.")

else:

    print(f"The number {target\_number} is not found in the list.")



**7. Python program to find the odd numbers in an array.**

def find\_odd\_numbers(arr):

    odd\_numbers = []

    for num in arr:

        if num % 2 != 0:

            odd\_numbers.append(num)

    return odd\_numbers

numbers = [2, 5, 8, 11, 14, 17, 20]

print("array:",numbers)

odd\_nums = find\_odd\_numbers(numbers)

print("Odd numbers in the array:")

print(odd\_nums)



**8. Python program to find the largest number in a list without using built-in functions.**

def find\_largest\_number(numbers):

    if len(numbers) == 0:

        return None

    largest = numbers[0]

    for num in numbers:

        if num > largest:

            largest = num

    return largest

numbers = [5, 2, 8, 10, 3, 7]

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

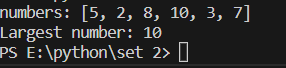
print("numbers:",numbers)

if largest\_num is not None:

    print("Largest number:", largest\_num)

else:

    print("The list is empty.")



**9. Python program to delete an element from a list by index**

def delete\_element\_by\_index(lst, index):

    if index < 0 or index >= len(lst):

        return False

    del lst[index]

    return True

numbers = [5, 2, 8, 10, 3, 7]

print("Original list:", numbers)

index = int(input("Enter the index of the element to delete: "))

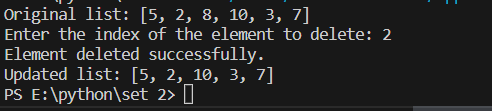
if delete\_element\_by\_index(numbers, index):

    print("Element deleted successfully.")

    print("Updated list:", numbers)

else:

    print("Invalid index. Element not deleted.")



**10. Python program to implement matrix addition.**

def matrix\_addition(matrix1, matrix2):

    rows = len(matrix1)

    cols = len(matrix1[0])

    result = [[0] \* cols for \_ in range(rows)]

    for i in range(rows):

        for j in range(cols):

            result[i][j] = matrix1[i][j] + matrix2[i][j]

    return result

print("Enter elements of Matrix 1:")

rows1 = int(input("Enter the number of rows: "))

cols1 = int(input("Enter the number of columns: "))

matrix1 = []

for i in range(rows1):

    row = []

    for j in range(cols1):

        element = int(input(f"Enter element at position ({i+1}, {j+1}): "))

        row.append(element)

    matrix1.append(row)

print("\nEnter elements of Matrix 2:")

rows2 = int(input("Enter the number of rows: "))

cols2 = int(input("Enter the number of columns: "))

matrix2 = []

for i in range(rows2):

    row = []

    for j in range(cols2):

        element = int(input(f"Enter element at position ({i+1}, {j+1}): "))

        row.append(element)

    matrix2.append(row)

if rows1 != rows2 or cols1 != cols2:

    print("Matrix addition is not possible.")

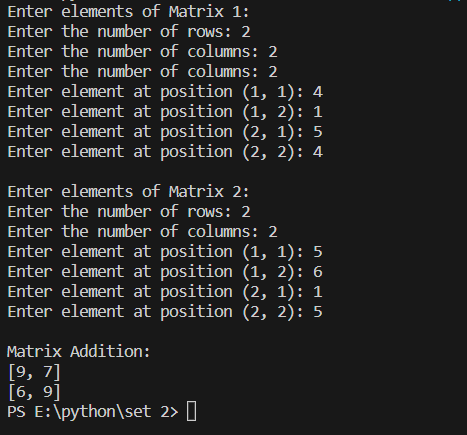
else:

    result\_matrix = matrix\_addition(matrix1, matrix2)

    print("\nMatrix Addition:")

    for row in result\_matrix:

        print(row)



**11. Python program to implement matrix multiplication**

def matrix\_multiplication(matrix1, matrix2):

    rows1 = len(matrix1)

    cols1 = len(matrix1[0])

    rows2 = len(matrix2)

    cols2 = len(matrix2[0])

    if cols1 != rows2:

        return None

    result = [[0] \* cols2 for \_ in range(rows1)]

    for i in range(rows1):

        for j in range(cols2):

            for k in range(cols1):

                result[i][j] += matrix1[i][k] \* matrix2[k][j]

    return result

print("Enter elements of Matrix 1:")

rows1 = int(input("Enter the number of rows: "))

cols1 = int(input("Enter the number of columns: "))

matrix1 = []

for i in range(rows1):

    row = []

    for j in range(cols1):

        element = int(input(f"Enter element at position ({i+1}, {j+1}): "))

        row.append(element)

    matrix1.append(row)

print("\nEnter elements of Matrix 2:")

rows2 = int(input("Enter the number of rows: "))

cols2 = int(input("Enter the number of columns: "))

matrix2 = []

for i in range(rows2):

    row = []

    for j in range(cols2):

        element = int(input(f"Enter element at position ({i+1}, {j+1}): "))

        row.append(element)

    matrix2.append(row)

if cols1 != rows2:

    print("Matrix multiplication is not possible.")

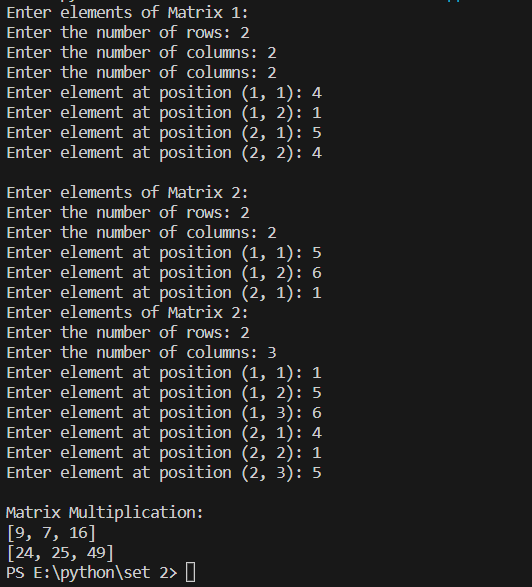
else:

    result\_matrix = matrix\_multiplication(matrix1, matrix2)

    print("\nMatrix Multiplication:")

    for row in result\_matrix:

        print(row)



**12. Python program to check leap year.**

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

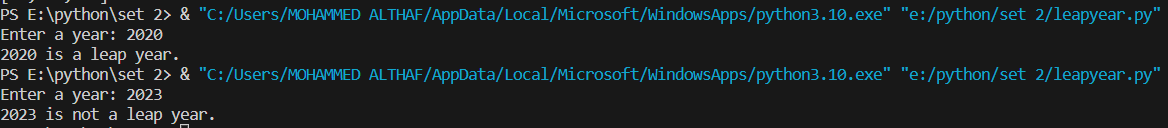
year = int(input("Enter a year: "))

if is\_leap\_year(year):

    print(f"{year} is a leap year.")

else:

    print(f"{year} is not a leap year.")



**13. Python program to find the Nth term in a Fibonacci series using recursion.**

def fibonacci(n):

    if n <= 0:

        return None

    elif n == 1:

        return 0

    elif n == 2:

        return 1

    else:

        return fibonacci(n - 1) + fibonacci(n - 2)

n = int(input("Enter the value of N: "))

nth\_term = fibonacci(n)

if nth\_term is not None:

    print(f"The {n}th term in the Fibonacci series is: {nth\_term}")

else:

    print("Invalid value of N.")



**14. Python program to print all the items in a dictionary.**

def print\_dictionary\_items(dictionary):

    for key, value in dictionary.items():

        print(key, ":", value)

student = {

    "name": "Althaf",

    "age": 23,

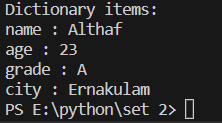
    "grade": "A",

    "city": "Ernakulam"

}

print("Dictionary items:")

print\_dictionary\_items(student)



**15. Python program to draw a circle of squares using Turtle.**

import turtle

def draw\_square(side\_length):

    for \_ in range(4):

        turtle.forward(side\_length)

        turtle.right(90)

def draw\_circle\_of\_squares(num\_squares, initial\_side\_length):

    angle = 360 / num\_squares

    for \_ in range(num\_squares):

        draw\_square(initial\_side\_length)

        turtle.right(angle)

# Initialize Turtle

turtle.speed(0)  # Set the drawing speed (0 is the fastest)

turtle.bgcolor("black")  # Set the background color

turtle.color("white")  # Set the pen color

# Input parameters

num\_squares = int(input("Enter the number of squares: "))

initial\_side\_length = int(input("Enter the side length of the squares: "))

# Draw the circle of squares

draw\_circle\_of\_squares(num\_squares, initial\_side\_length)

# Hide Turtle

turtle.hideturtle()

# Exit on click

turtle.exitonclick()



