1. maximum of three number.

# Get input from the user for three numbers

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

num3 = float(input("Enter the third number:"))

# Find the maximum of the three numbers without using max function

max\_number = num1

if num2 > max\_number:

max\_number = num2

if num3 > max\_number:

max\_number = num3

# Print the result

print(f"The maximum of {num1}, {num2}, and {num3} is: {max\_number}")

output. Enter the first number: 1

Enter the second number: 2

Enter the third number:3

The maximum of 1.0, 2.0, and 3.0 is: 3.0

2. python program for factorial of a number

def factorial(n):

if n == 0 or n == 1:

return 1

else:

return n \* factorial(n-1)

# Get input from the user

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

# Check if the number is negative

if num < 0:

print("Factorial is not defined for negative numbers.")

else:

result = factorial(num)

print(f"The factorial of {num} is: {result}")

output. Enter a number: 10

The factorial of 10 is: 3628800

This program defines a function `factorial` that calculates the factorial of a number using recursion. The user is prompted to enter a number, and the program then checks if the number is negative (since factorial is not defined for negative numbers). If the number is non-negative, the program calculates and prints the factorial.

3. python program for simple and compound interest

def calculate\_simple\_interest(principal, rate, time):

simple\_interest = (principal \* rate \* time) / 100

return simple\_interest

def calculate\_compound\_interest(principal, rate, time):

compound\_interest = principal \* (1 + rate / 100) \*\* time - principal

return compound\_interest

# Get input from the user

principal\_amount = float(input("Enter the principal amount: "))

interest\_rate = float(input("Enter the annual interest rate: "))

time\_period = float(input("Enter the time period (in years): "))

# Calculate simple interest

simple\_interest\_result = calculate\_simple\_interest(principal\_amount, interest\_rate, time\_period)

# Calculate compound interest

compound\_interest\_result = calculate\_compound\_interest(principal\_amount, interest\_rate, time\_period)

# Display the results

print(f"\nSimple Interest: {simple\_interest\_result:.2f}")

print(f"Compound Interest: {compound\_interest\_result:.2f}")

output.

Enter the principal amount: 200

Enter the annual interest rate: 1000

Enter the time period (in years): 3

Simple Interest: 6000.00

Compound Interest: 266000.00

4. check armstong number python program

def is\_armstrong\_number(number):

# Convert the number to a string to find the number of digits

num\_str = str(number)

num\_digits = len(num\_str)

# Calculate the sum of each digit raised to the power of the number of digits

sum\_of\_digits = sum(int(digit) \*\* num\_digits for digit in num\_str)

# Check if the sum is equal to the original number

return sum\_of\_digits == number

# Get input from the user

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

# Check if the number is an Armstrong number

if is\_armstrong\_number(num):

print(f"{num} is an Armstrong number.")

else:

print(f"{num} is not an Armstrong number.")

output:

Enter a number: 324

324 is not an Armstrong number.

5. print all prime numbers in an interval

def is\_prime(num):

if num < 2:

return False

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

if num % i == 0:

return False

return True

def print\_primes\_in\_interval(start, end):

print(f"Prime numbers in the interval [{start}, {end}]:")

for num in range(start, end + 1):

if is\_prime(num):

print(num, end=" ")

# Get input from the user for the interval

start\_num = int(input("Enter the start of the interval: "))

end\_num = int(input("Enter the end of the interval: "))

# Print prime numbers in the specified interval

print\_primes\_in\_interval(start\_num, end\_num)

output.

Enter the start of the interval: 23

Enter the end of the interval: 54

Prime numbers in the interval [23, 54]:

23 29 31 37 41 43 47 53

6. check whether a number is prime or not

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

# Get input from the user

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

# Check if the number is prime

if is\_prime(num):

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

else:

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

output:

Enter a number: 3

3 is a prime number.

7. for nth fibonacci number

def fibonacci(n):

if n <= 0:

return "Invalid input. Please enter a positive integer for n."

elif n == 1:

return 0

elif n == 2:

return 1

else:

a, b = 0, 1

for \_ in range(n - 2):

a, b = b, a + b

return b

# Get input from the user

n = int(input("Enter the value of n for the nth Fibonacci number: "))

# Find and print the nth Fibonacci number

result = fibonacci(n)

print(f"The {n}th Fibonacci number is: {result}")

output: Enter the value of n for the nth Fibonacci number: 8

The 8th Fibonacci number is: 13

8. sum of squares of first n natural number

Method1

square of first n

# natural numbers

**def** squaresum(n):

    # Iterate i from 1

    # and n finding

    # square of i and

    # add to sum.

    sm **=** 0

**for** i **in** range(1, n**+**1):

        sm **=** sm **+** (i **\*** i)

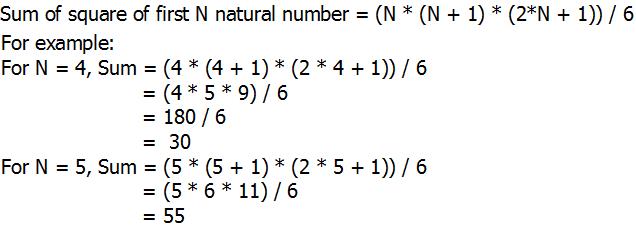
**return** sm

# Driven Program

n **=** 4

print(squaresum(n))

output :30



Method 2,.

def sum\_of\_squares(n):

if n <= 0:

return "Invalid input. Please enter a positive integer for n."

else:

return sum(i\*\*2 for i in range(1, n + 1))

# Get input from the user

n = int(input("Enter a positive integer (n) to find the sum of squares: "))

# Calculate and print the sum of squares

result = sum\_of\_squares(n)

print(f"The sum of the squares of the first {n} natural numbers is: {result}")

output:

Enter a positive integer (n) to find the sum of squares: 5

The sum of the squares of the first 5 natural numbers is: 55

9. for cube sum of first n natural number

def sum\_of\_cubes(n):

if n <= 0:

return "Invalid input. Please enter a positive integer for n."

else:

return sum(i\*\*3 for i in range(1, n + 1))

# Get input from the user

n = int(input("Enter a positive integer (n) to find the sum of cubes: "))

# Calculate and print the sum of cubes

result = sum\_of\_cubes(n)

print(f"The sum of the cubes of the first {n} natural numbers is: {result}")

Enter a positive integer (n) to find the sum of cubes: 9

The sum of the cubes of the first 9 natural numbers is: 2025

9. find sum of array

# Get input from the user for the array of numbers

numbers = input("Enter the array of numbers separated by spaces: ").split()

# Convert the input strings to integers

numbers = [int(num) for num in numbers]

# Calculate and print the sum of the array

array\_sum = sum(numbers)

print(f"The sum of the array is: {array\_sum}")

output:

Enter the array of numbers separated by spaces: 7 9 7 6

The sum of the array is: 29

10find largest element in an array

# Get input from the user for the array of numbers

numbers = input("Enter the array of numbers separated by spaces: ").split()

# Convert the input strings to integers

numbers = [int(num) for num in numbers]

# Find the largest element in the array

largest\_element = max(numbers)

# Print the result

print(f"The largest element in the array is: {largest\_element}")

output:

Enter the array of numbers separated by spaces: 4 5 6 5

The largest element in the array is: 6