**Lab 2**

1. **To find the sum of square root of any three numbers.**

import math

def sum\_of\_square\_roots(num1, num2, num3):

return math.sqrt(num1) + math.sqrt(num2) + math.sqrt(num3)

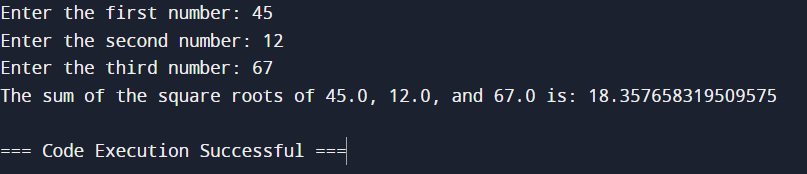
num1 = float(input("Enter the first number: "))

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

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

result = sum\_of\_square\_roots(num1, num2, num3)

print(f"The sum of the square roots of {num1}, {num2}, and {num3} is: {result}")



1. **To solve the quadratic equation.**

import math

def solve\_quadratic(a, b, c):

discriminant = b\*\*2 - 4\*a\*c

if discriminant > 0:

root1 = (-b + math.sqrt(discriminant)) / (2 \* a)

root2 = (-b - math.sqrt(discriminant)) / (2 \* a)

return root1, root2

elif discriminant == 0:

root = -b / (2 \* a)

return root,

else:

real\_part = -b / (2 \* a)

imaginary\_part = math.sqrt(-discriminant) / (2 \* a)

return (real\_part + imaginary\_part \* 1j, real\_part - imaginary\_part \* 1j)

a = float(input("Enter the coefficient a: "))

b = float(input("Enter the coefficient b: "))

c = float(input("Enter the coefficient c: "))

roots = solve\_quadratic(a, b, c)

if len(roots) == 2:

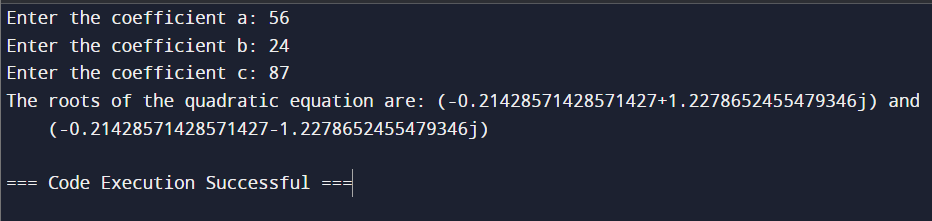
print(f"The roots of the quadratic equation are: {roots[0]} and {roots[1]}")

elif len(roots) == 1:

print(f"The root of the quadratic equation is: {roots[0]}")

else:

print(f"The roots of the quadratic equation are: {roots[0]} and {roots[1]}")



1. **Find GCD of two numbers**

def gcd(a, b):

while b:

a, b = b, a % b

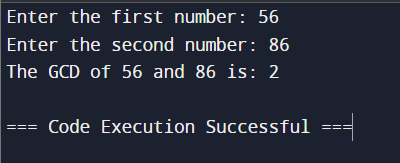
return a

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

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

result = gcd(num1, num2)

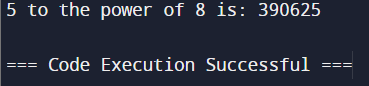
print(f"The GCD of {num1} and {num2} is: {result}")



1. **Compute a)5 to the power of 8**

result = 5 \*\* 8

print(f"5 to the power of 8 is: {result}")

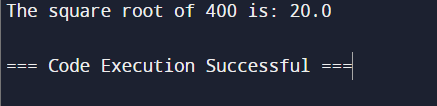


**b)square root of 400**

import math

result = math.sqrt(400)

print(f"The square root of 400 is: {result}")

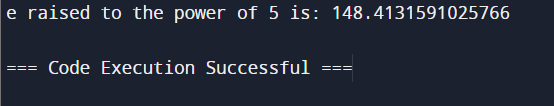


**c)exponent of 5**

import math

result = math.exp(5)

print(f"e raised to the power of 5 is: {result}")

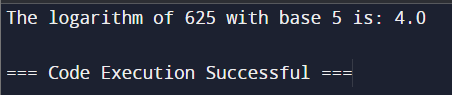


**d)Logarithm of 625 base 5**

import math

result = math.log(625, 5)

print(f"The logarithm of 625 with base 5 is: {result}")



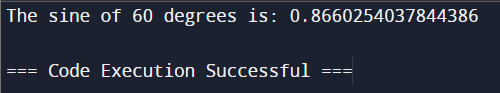
1. **Compute a)sin of 60 degree**

import math

radians = math.radians(60)

result = math.sin(radians)

print(f"The sine of 60 degrees is: {result}")

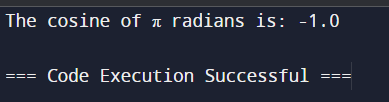


**b)cos of pi**

import math

result = math.cos(math.pi)

print(f"The cosine of π radians is: {result}")



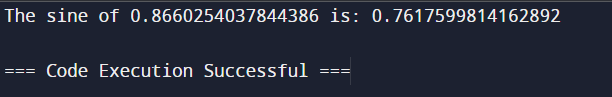
**c)sin(0.8660254037844386)**

import math

value = 0.8660254037844386

result = math.sin(value)

print(f"The sine of {value} is: {result}")



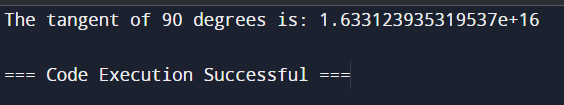
**d)tan of 90 degree**

import math

radians = math.radians(90)

result = math.tan(radians)

print(f"The tangent of 90 degrees is: {result}")



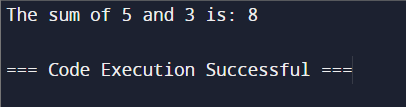
1. **Define a sum function with two parameters and call the function**

def sum(a, b):

return a + b

result = sum(5, 3)

print(f"The sum of 5 and 3 is: {result}")



1. **WAP to reverse a given string.**

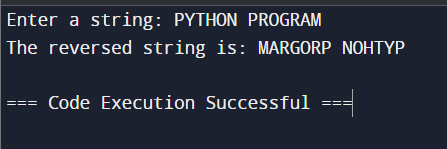
def reverse\_string(s):

return s[::-1]

string = input("Enter a string: ")

reversed\_string = reverse\_string(string)

print(f"The reversed string is: {reversed\_string}")



1. **Write a function to calculate the power of a number using recursion**

def power(base, exponent):

if exponent == 0:

return 1

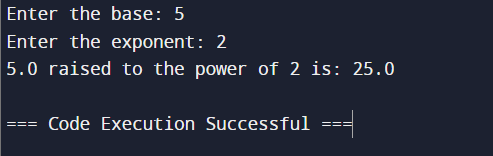
return base \* power(base, exponent - 1)

base = float(input("Enter the base: "))

exponent = int(input("Enter the exponent: "))

result = power(base, exponent)

print(f"{base} raised to the power of {exponent} is: {result}")



1. **Convert Decimal number to Binary**

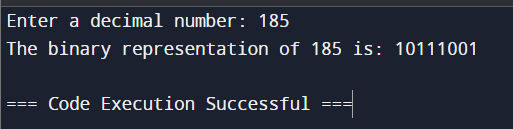
def decimal\_to\_binary(decimal):

return bin(decimal).replace("0b", "")

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

binary = decimal\_to\_binary(decimal)

print(f"The binary representation of {decimal} is: {binary}")



1. **Write a program in Python to check if a number is Krishnamurthy number.**

import math

def is\_krishnamurthy\_number(number):

digits = str(number)

sum\_of\_factorials = sum(math.factorial(int(digit)) for digit in digits)

return sum\_of\_factorials == number

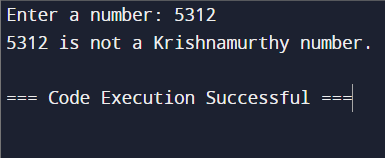
number = int(input("Enter a number: "))

if is\_krishnamurthy\_number(number):

print(f"{number} is a Krishnamurthy number.")

else:

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



1. **Write a program in Python to find the sum of digits of a number.**

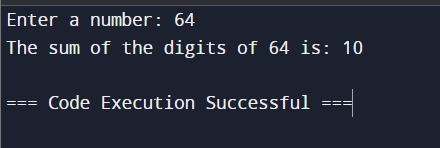
def sum\_of\_digits(number):

return sum(int(digit) for digit in str(number))

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

result = sum\_of\_digits(number)

print(f"The sum of the digits of {number} is: {result}")

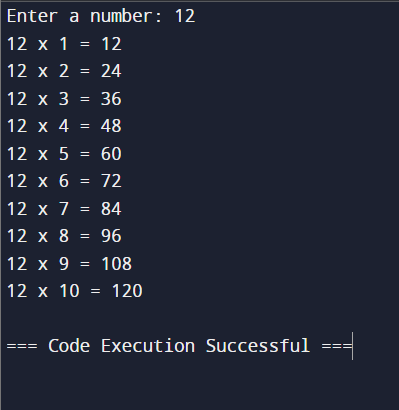


1. **Write a program in Python that prompts the user to input a number and prints its multiplication table.**

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

for i in range(1, 11):

print(f"{number} x {i} = {number \* i}")



1. **Write a Python program to print the first 6 terms of a geometric sequence starting with 2 and having a common ratio of 3.**

first\_term = 2

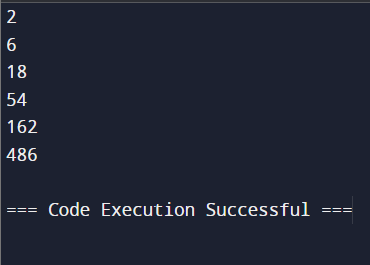
common\_ratio = 3

num\_terms = 6

for i in range(num\_terms):

term = first\_term \* (common\_ratio \*\* i)

print(term)



1. **Print the series upto N terms: 1,2,6,24,120,720 …**

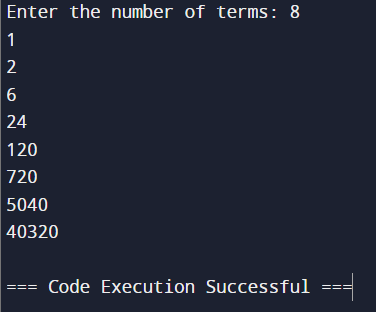
import math

N = int(input("Enter the number of terms: "))

for i in range(1, N + 1):

term = math.factorial(i)

print(term)



1. **Write a Python program that prompts the user to enter a base number and an exponent, and then calculates the power of the base to the exponent. The program should not use the exponentiation operator (\*\*) or the math.pow() function.**

def power(base, exponent):

result = 1

for \_ in range(exponent):

result \*= base

return result

base = float(input("Enter the base number: "))

exponent = int(input("Enter the exponent: "))

result = power(base, exponent)

print(f"{base} raised to the power of {exponent} is: {result}")

