**Lab 1**

1. **WAP to convert temperature from Celsius to Fahrenheit.**

def celsius\_to\_fahrenheit():

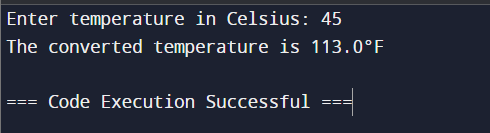
celsius = float(input("Enter temperature in Celsius: "))

fahrenheit = (celsius \* 9/5) + 32

return fahrenheit

fahrenheit = celsius\_to\_fahrenheit()

print(f"The converted temperature is {fahrenheit}°F")



1. **WAP to swap two numbers**

def swap\_numbers(a, b):

a, b = b, a

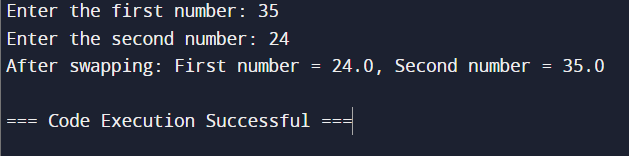
return a, b

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

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

num1, num2 = swap\_numbers(num1, num2)

print(f"After swapping: First number = {num1}, Second number = {num2}")



1. **WAP to check whether a given year is leap year or not**

def is\_leap\_year(year):

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):

return True

else:

return False

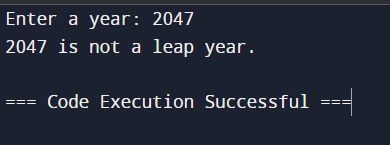
year = int(input("Enter a year: "))

if is\_leap\_year(year):

print(f"{year} is a leap year.")

else:

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



1. **WAP to display reverse of a number**

def reverse\_number(number):

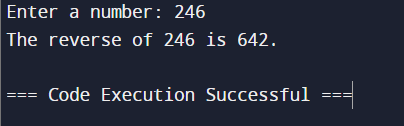
reversed\_number = int(str(number)[::-1])

return reversed\_number

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

reversed\_number = reverse\_number(number)

print(f"The reverse of {number} is {reversed\_number}.")



1. **WAP to find factors of a given number**

def find\_factors(number):

factors = []

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

if number % i == 0:

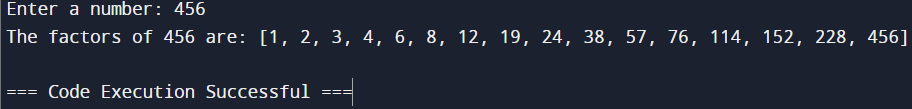
factors.append(i)

return factors

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

factors = find\_factors(number)

print(f"The factors of {number} are: {factors}")



1. **WAP to generate prime number series up to n**

def is\_prime(num):

if num <= 1:

return False

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

if num % i == 0:

return False

return True

def generate\_prime\_series(n):

prime\_series = []

for num in range(2, n + 1):

if is\_prime(num):

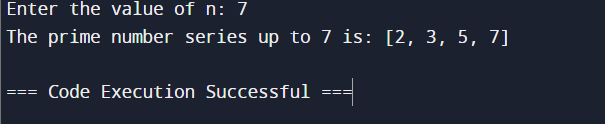
prime\_series.append(num)

return prime\_series

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

prime\_series = generate\_prime\_series(n)

print(f"The prime number series up to {n} is: {prime\_series}")



1. **Write a program which makes use of function to display all such numbers which are divisible by 7 but are not a multiple of 5, between 1000 and 2000.**

def find\_numbers():

result = []

for num in range(1000, 2001):

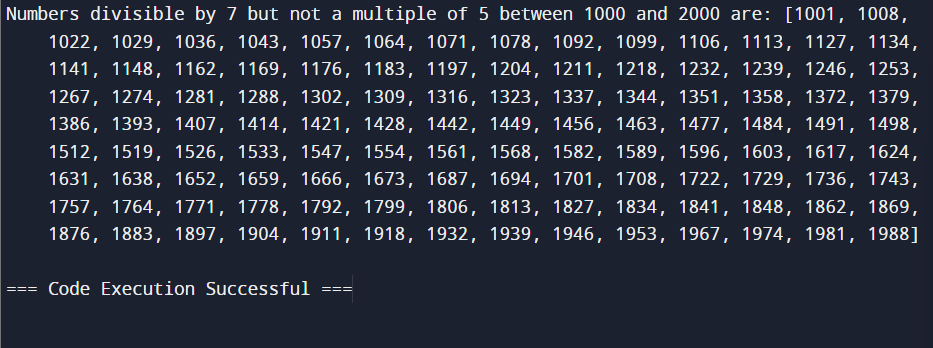
if num % 7 == 0 and num % 5 != 0:

result.append(num)

return result

numbers = find\_numbers()

print(f"Numbers divisible by 7 but not a multiple of 5 between 1000 and 2000 are: {numbers}")



1. **WAP to check whether a number is a palindrome or not**

def is\_palindrome(number):

original\_number = str(number)

reversed\_number = original\_number[::-1]

return original\_number == reversed\_number

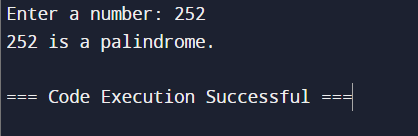
number = int(input("Enter a number: "))

if is\_palindrome(number):

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

else:

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



1. **WAP to check whether a)is a perfect number**

def is\_perfect\_number(number):

if number < 1:

return False

sum\_of\_divisors = sum(i for i in range(1, number) if number % i == 0)

return sum\_of\_divisors == number

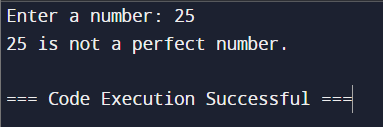
number = int(input("Enter a number: "))

if is\_perfect\_number(number):

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

else:

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



**b)is an Armstrong number**

def is\_armstrong\_number(number):

digits = str(number)

num\_digits = len(digits)

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

return sum\_of\_powers == number

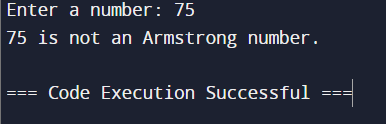
number = int(input("Enter a number: "))

if is\_armstrong\_number(number):

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

else:

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



1. **WAP to generate the Fibonacci series upto n.**

def fibonacci\_series(n):

series = []

a, b = 0, 1

while a <= n:

series.append(a)

a, b = b, a + b

return series

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

fib\_series = fibonacci\_series(n)

print(f"The Fibonacci series up to {n} is: {fib\_series}")

