#if else statements Program:

age=int(input("Enter the age"))

if age>=18:

    print("Eligible to vote")

else:

    print("Not eligible to vote")

#String character Program:

str1=("good morning students")

print(str1[::1])

print(str1[::-1])

str2=("How are you")

print(str2[::2])

print(str2[::-2])

#Print digits up to 20 Program:

for i in range(1,21,2):

    print(i)

for i in range(1,21):

    print(i)

#Except[7,9,12,16] display other nums up to 20 Program:

for i in range(1,21):

    if i not in [7,9,12,16]:

        print(i,end=' ')

#Finding number of Digits:

num=768124

count=0

temp=num

while temp>0:

    count+=1

    temp=temp//10

print("Number of digits:",count)

#Sum of digits and Product of digits and reverse it Program:

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

count=0

while num !=0:

    ld=num%10

    count+=1

    num=num//10

    print("Number of digts:",count)

    count=0

    sum=0

    prod=1

    while num !=0:

        ld=num%10

        sum+=ld

        prod\*=ld

        num=num//10

        print("sum of digits:", sum)

        print("product of digits:", prod)

        print("Reverse of the number:", str(num)[::-1])

#Num of digits is armstrong or not Program:

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

orig\_num = num

count = 0

temp = num

while temp != 0:

    count += 1

    temp //= 10

print("Number of digits:", count)

temp = num

armstrong\_sum = 0

while temp != 0:

    ld = temp % 10

    armstrong\_sum += ld \*\* count

    temp //= 10

if armstrong\_sum == orig\_num:

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

else:

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

#Prime no up to 30 Program:

print("Prime numbers up to 30:")

for num in range(2, 31):

    is\_prime = True

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

        if num % i == 0:

            is\_prime = False

            break

    if is\_prime:

        print(num)

#Prime no or not Prime Program:

for num in range(1, 31):

    count=0

    for i in range(1,num):

        if num % i ==0:

            count +=1

            if count == 1:

                print(num)

            elif count == 2:

                print(num)

            else:

                break

            if count == 2:

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

            else:

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

    else:

        print(num)

# Fizz Buzz Program:

for num in range(1,46):

    if num % 3 == 0 and num % 5 == 0:

        print("Fizz Buzz")

    elif num % 3 == 0:

        print("Fizz")

    elif num % 5 == 0:

        print("Buzz")

    else:

        print(num)

#Prime number up to 30 with serial numbers code:

print("Serial No.  Prime Number")

serial = 1

for num in range(2, 31):

    for i in range(2, num):

        if num % i == 0:

            break

    else:

        print(f"{serial}.         {num}")

        serial += 1

#Program to find the value 135:

num = 135

temp = num

count = 0

while num!= 0:

    ld = num%10

    count += 1

    num //= 10

sum = 0

num = temp

while num!= 0:

    ld = num % 10

    sum += ld \*\* count

    count -= 1

    num= num // 10

if temp == sum:

    print("neon number")

else:

    print("not a neon number")

#Program to find neon number upto 2000 with serial numbers:

print("Serial No.  Neon Number")

serial = 1

for num in range(1, 2001):

    sq = num \* num

    sum\_digits = 0

    while sq != 0:

        sum\_digits += sq % 10

        sq //= 10

    if sum\_digits == num:

        print(f"{serial}.         {num}")

        serial += 1

#Program to reverse the given number and check Palindrome or not:

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

orig\_num = num

rev = 0

while num != 0:

    rev = rev \* 10 + num % 10

    num //= 10

print(f"Reverse of the number: {rev}")

if orig\_num == rev:

    print(f"{orig\_num} is a palindrome.")

else:

    print(f"{orig\_num} is not a palindrome.")

#Program to find Palindrome nums from 1000 to 2000 with serial numbers:

print("Serial No.  Palindrome Number")

serial = 1

for num in range(1000, 1201):

    orig\_num = num

    rev = 0

    temp = num

    while temp != 0:

        rev = rev \* 10 + temp % 10

        temp //= 10

    if orig\_num == rev:

        print(f"{serial}.         {orig\_num}")

        serial += 1

n = int(input("Enter the number :"))

ser = 1

for num in range(1,1000):

    temp = num

    rev = 0

    while num != 0:

        ld = num % 10

        rev = rev \* 10 + ld

        num = num // 10

    if temp == rev:

        if n == temp:

            print(ser, ")", rev)

        ser += 1

# Program to check if a given number is a Harshad number:

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

temp = num

sum\_digits = 0

while temp != 0:

    sum\_digits += temp % 10

    temp //= 10

if num % sum\_digits == 0:

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

else:

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

#Program to find Harshad Number or Not:

num = int(input("enter the number : "))

temp = num

sum = 0

while num != 0:

    ld = num % 10

    sum = sum + ld

    num = num // 10

if temp % sum == 0:

    print("It is a Harshad Number")

else:

    print("It is not a Harshad Number")

#Program to find Factorial of Number:

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

fact = 1

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

    fact \*= i

print(f"Factorial of {num} is {fact}")

# Program to add two numbers:

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

b = int(input("Enter second number: "))

sum = a + b

print(f"Sum of {a} and {b} is {sum}")

#Program to find even or odd using if else:

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

if num%2==0:

    print(f"{num} is even")

else:

    print(f"{num} is odd")

#Program of List:

list\_d=eval(input("enter the list:"))

print(list\_d)

# Program to take 2 inputs and check if the given inputs are integers:

a = input("Enter first value: ")

b = input("Enter second value: ")

if a.isdigit() and b.isdigit():

    print("Both inputs are integers.")

else:

    print("One or both inputs are not integers.")

# Program to find minimum and maximum of two numbers:

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

b = int(input("Enter second number: "))

minimum = min(a, b)

maximum = max(a, b)

print(f"Minimum: {minimum}")

print(f"Maximum: {maximum}")

# Program to find maximum of three numbers a, b, c:

a = int(input("Enter value for a: "))

b = int(input("Enter value for b: "))

c = int(input("Enter value for c: "))

maximum = max(a, b, c)

print(f"Maximum of {a}, {b}, and {c} is {maximum}")

#Program to find Greatest of three numbers:

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

b=int(input("Enter second number :"))

c=int(input("Enter third number :"))

if a>b and a>c:

    print("a is grater than b and c ")

elif b>a and b>c:

    print("b is grater than a and c")

elif c>a and c>b:

    print("c is greater than a and b")

else:

    print("ALL ARE EQUAL")

# Program to check whether the data present in a list are integers or not:

data = [10, 'abc', 25, 3.5, 42, '5']

for item in data:

    if isinstance(item, int):

        print(f"{item} is an integer.")

    else:

        print(f"{item} is not an integer. ")

# Program to find how many elements are present inside the list:

data = [10, 20, 30, 40, 50]

count = len(data)

print(f"Number of elements in the list: {count}")

# Program to input a number and output the product of its digits:

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

prod = 1

temp = num

while temp != 0:

    prod \*= temp % 10

    temp //= 10

print(f"Product of digits of {num} is {prod}")

# Program to reverse the given number using floor division and modulus:

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

rev = 0

while num != 0:

    rev = rev \* 10 + num % 10

    num //= 10

print(f"Reversed number is: {rev}")

# Program to check if the given number is single, double, triple, or more digits:

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

abs\_num = abs(num)

if abs\_num < 10:

    print("Single digit number")

elif abs\_num < 100:

    print("Double digit number")

elif abs\_num < 1000:

    print("Triple digit number")

else:

    print("More than three digits")

# Program to find the ASCII values of a given string:

s = input("Enter a string: ")

for char in s:

    print(f"ASCII value of '{char}' is {ord(char)}")

#Program to reverse a string:

a=input("enter a string")

for i in a:

    print(ord(i),end='')

rev=''

str\_i=input("enter the string:")

for i in str\_i:

    rev=i+rev

print(rev)

# Program to find if the given character is uppercase, lowercase, or a numeric character:

char = input("Enter a character: ")

if len(char) != 1:

    print("Please enter a single character.")

elif char.isupper():

    print(f"{char} is an uppercase letter.")

elif char.islower():

    print(f"{char} is a lowercase letter.")

elif char.isdigit():

    print(f"{char} is a numeric character.")

else:

    print(f"{char} is a special character.")

# Program to process three lists as per the given conditions:

list1 = [2, 4, 6, 8, 10]

list2 = ["abc", "def", "ghi", "jkl", "xyz"]

list3 = [3, 5, 7, 9, 11]

if len(list1) == 5 and len(list2) == 5 and len(list3) == 5:

    part1 = list2[0] \* list1[0]

    part2 = list2[-1] \* list3[-1]

    result = part1 + part2

    print(result)

else:

    print("All lists must have a length of 5. ")

# Program to reverse the given string without slicing:

s = input("Enter a string: ")

rev = ""

for char in s:

    rev = char + rev

print(f"Reversed string: {rev}")

# Program to reverse the given number:

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

rev = 0

while num != 0:

    rev = rev \* 10 + num % 10

    num //= 10

print(f"Reversed number is: {rev}")

# Program to count the total number of unmatched parentheses in the input:

s = input("Enter a string of parentheses: ")

stack = []

unmatched\_close = 0

for char in s:

    if char == '(':

        stack.append(char)

    elif char == ')':

        if stack:

            stack.pop()

        else:

            unmatched\_close += 1

# Total unmatched = unmatched '(' + unmatched ')'

print(len(stack) + unmatched\_close)

# Program to count the number of positions where the bits differ in two binary strings:

a = input("Enter first binary string: ")

b = input("Enter second binary string: ")

if len(a) != len(b):

    print("Both binary strings must be of the same length.")

else:

    diff\_count = 0

    for i in range(len(a)):

        if a[i] != b[i]:

            diff\_count += 1

    print(diff\_count)

# Program to swap two numbers using a third variable:

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

b = int(input("Enter second number (b): "))

temp = a

a = b

b = temp

print(f"After swapping: a = {a}, b = {b}")

# Program to swap two numbers without using a temporary variable:

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

b = int(input("Enter second number (b): "))

a = a + b

b = a - b

a = a - b

print(f"After swapping: a = {a}, b = {b}")

#simple Progeam to swap two numbers:

a=20

b=50

a,b = b,a

print(a, b)

# Program to check whether two strings are palindrome or not using slicing:

str1 = input("Enter first string: ")

str2 = input("Enter second string: ")

if str1 == str1[::-1]:

    print(f'"{str1}" is a palindrome.')

else:

    print(f'"{str1}" is not a palindrome.')

if str2 == str2[::-1]:

    print(f'"{str2}" is a palindrome.')

else:

    print(f'"{str2}" is not a palindrome. ')

# Program to check whether two strings are palindrome or not without using slicing:

str1 = input("Enter first string: ")

str2 = input("Enter second string: ")

def is\_palindrome(s):

    rev = ""

    for char in s:

        rev = char + rev

    return s == rev

if is\_palindrome(str1):

    print(f'"{str1}" is a palindrome.')

else:

    print(f'"{str1}" is not a palindrome.')

if is\_palindrome(str2):

    print(f'"{str2}" is a palindrome.')

else:

    print(f'"{str2}" is not a palindrome. ')

# Program to check whether two numbers are palindrome or not:

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

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

def is\_palindrome(n):

    return str(n) == str(n)[::-1]

print(f"{num1} is a palindrome." if is\_palindrome(num1) else f"{num1} is not a palindrome.")

print(f"{num2} is a palindrome." if is\_palindrome(num2) else f"{num2} is not a palindrome.")

# Program to print the divisors of the number:

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

print(f"Divisors of {num} are:")

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

    if num % i == 0:

        print(i)

#write a program to add the divisors of the number:

def sum\_of\_divisors(n):

    total = 0

    for i in range(1, n):

        if n % i == 0:

            total += i

    return total

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

result = sum\_of\_divisors(number)

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

#Program to check given number is Perfect or deficient or abundent number:

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

s = sum(i for i in range(1, n//2 + 1) if n % i == 0)

print("Perfect" if s == n else "Deficient" if s < n else "Abundant")

#Program to print the numeric characters from the string:

s = input("Enter a string: ")

for c in s:

    if c.isdigit():

        print(c, end='')

#Program  to print all float items from given heterogeneous list which are less than 100:

items = [23, "hello", 99.9, 150.5, 42.0, True, None, 75.25, "100", 100.0, 3.14]

for item in items:

    if isinstance(item, float) and item < 100:

        print(item)

#Program to print all the string items from given homogeneous tuple if string lower case vowel character:

t=("apple", "dog", "banana", "kite", "idea", "echo")

for s in t:

    if s[-1] in "aeiou":

        print(s)

#Program to program to extract all the lowercase alphabets from given string collection:

def extract\_lowercase(s):

    return ''.join(c for c in s if c.islower())

input\_str = "HeLlo WorLd I am prajwal 2925"

result = extract\_lowercase(input\_str)

print(result)

#Program to extract all the complex items from given heterogeneous list collection:

def extract\_complex(lst):

    return [item for item in lst if isinstance(item, complex)]

data = [1, 2.5, "hello", 3 + 4j, [1, 2], 5j, 10, (2+0j)]

result = extract\_complex(data)

print(result)

# Program to find the area of the square:

side = float(input("Enter the length of the side of the square: "))

area = side \* side

print(f"The area of the square is {area}")

#Program to check Harshad Number or not:

num = int(input("enter the number : "))

temp = num

sum = 0

while num != 0:

    ld = num % 10

    sum = sum + ld

    num = num // 10

if temp % sum == 0:

    print("It is a Harshad Number")

else:

    print("It is not a Harshad Number")

#Program to display 0 1 1 2 3 if input is 5:

num = int(input("enter the number : "))

a, b = 0, 1

print(a, b, end=' ')

for \_ in range(num - 2):

    c = a + b

    print(c, end=' ')

    a,b = b, c

#Program to display output if input is ['mango' , 'apple' , 'orange' , 'kiwi'] and output is {'mango':'ao' ,} to all givrn input:

fruits = ['mango', 'apple', 'orange', 'kiwi']

output = {}

for fruit in fruits:

    vowels = ''.join([char for char in fruit if char in 'aeiou '])

    output[fruit] = vowels

    print(f"{fruit}: {vowels}")

#write a Program to check the given number is prime number or not:

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

if num > 1 and all(num % i != 0 for i in range(2, int(num\*\*0.5) + 1)):

    print("Prime")

else:

    print("Not Prime")

#Program to check weight of bags at airport:

weight = float(input("Enter the weight of the bag in kg: "))

base\_fee = 500

if weight <= 10:

    total\_cost = base\_fee

else:

    extra\_weight = weight - 10

    extra\_fee = int(extra\_weight) \* 200

    total\_cost = base\_fee + extra\_fee

print(f"Total amount to be paid: ₹{total\_cost}")

# Another simple Program to check weight of bags at airport:

d={}

seti={10,12,30,21,14}

for i in seti:

    if i <= 10:

        pay = 500

    else:

        pay = 500 + (i - 10) \* 200

    d[i] = pay

print(d)

#write a Program for ADD,MUl,Sub,Div with two inputs n1 and n2 by using functions:

def add(n1, n2):

    return n1 + n2

def sub(n1, n2):

    return n1 - n2

def mul(n1, n2):

    return n1 \* n2

def div(n1, n2):

    if n2 != 0:

        return n1 / n2

    else:

        return "Division by zero error"

n1 = float(input("Enter first number: "))

n2 = float(input("Enter second number: "))

print("Addition:", add(n1, n2))

print("Subtraction:", sub(n1, n2))

print("Multiplication:", mul(n1, n2))

print("Division:", div(n1, n2))

#Program to find Prime number or not by using functions:

def is\_prime(num):

    if num > 1 and all(num % i != 0 for i in range(2, int(num\*\*0.5) + 1)):

        return True

    else:

        return False

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

if is\_prime(num):

    print("Prime")

else:

    print("Not Prime")

#Program for reversing a number using functions:

def reverse\_number(n):

    return int(str(n)[::-1])

num = 12345

reversed\_num = reverse\_number(num)

print("Reversed number:", reversed\_num)

#Program for reversing a string using functions:

def reverse\_string(s):

    return s[::-1]

s = "Hello, World!"

reversed\_string = reverse\_string(s)

print("Reversed string:", reversed\_string)

#Program to find string is palindrome or not using functions:

def is\_palindrome(s):

    return s == s[::-1]

s = input("Enter a string: ")

if is\_palindrome(s):

    print("Palindrome")

else:

    print("Not Palindrome")

#Program to display Lucky winner if ticket last digit is 3 or 8 else display Not a Lucky winner:

def is\_lucky\_winner(ticket\_number):

    last\_digit = ticket\_number % 10

    return last\_digit in [3, 8]

ticket\_number = int(input("Enter the ticket number: "))

if is\_lucky\_winner(ticket\_number):

    print("Lucky winner")

else:

    print("Not a Lucky winner")

#Program to create a dictionary where a key is no from 1-10 and value of the square of the key:

squares = {x: x\*\*2 for x in range(1, 11)}

print(squares)

#Program to check the result of Kids in a Triangle game:

a = int(input("Enter angle 1: "))

b = int(input("Enter angle 2: "))

c = int(input("Enter angle 3: "))

# Check if the angles form a triangle

if a + b + c == 180 and a > 0 and b > 0 and c > 0:

    if a == b == c:

        print("Prize 3 (Equilateral Triangle)")

    elif a == 90 or b == 90 or c == 90:

        print("Prize 2 (Right Triangle)")

    else:

        print("Prize 1 (Valid Triangle)")

else:

    print("No Prize (Not a Triangle)")

**#Program to check the Parking fee of a car,bike,bus the customer enter vehicle type the system diplay fee:**

**fees = {**

**"car": 20,**

**"bike": 10,**

**"bus": 30,**

**}**

**vehicle = input("Enter vehicle type (car/bike/bus): ").lower()**

**if vehicle in fees:**

**print(f"Parking fee for a {vehicle}: ₹{fees[vehicle]}")**

**else:**

**print("Invalid vehicle type entered.")**

**#Program to find vowels using split functions:**

**d={}**

**a = ['apple', 'mango', 'cherry', 'kiwi']**

**for i in a:**

**v=''**

**for j in i:**

**if j in 'aeiou':**

**v+=j**

**d[i]=v**

**print(d)**

**#Program to check Double Bonaza,Bonaza,or does not get Bonaza in a card game:**

**def check\_bonanza(cards):**

**types = [card[0].upper() for card in cards]**

**numbers = [card[1:] for card in cards]**

**if types.count(types[0]) == 3 and numbers.count(numbers[0]) == 3:**

**print("Double Bonanza")**

**elif types.count(types[0]) == 3 or numbers.count(numbers[0]) == 3:**

**print("Bonanza")**

**else:**

**print("They do not get a bonanza")**

**cards = []**

**for i in range(3):**

**card = input(f"Enter card {i+1} (e.g., H5, D10, S2): ").strip()**

**cards.append(card)**

**check\_bonanza(cards)**

**#Program to find the only line of input contains a single integer from the candy group:**

**def candy\_group(n):**

**if n % 2 == 0:**

**return "Even"**

**else:**

**return "odd"**

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

**result = candy\_group(n)**

**print("The number is:", result)**

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

**result = candy\_group(n)**

**print("The number is:", result)**

**#Program to print 1 to 5 numbers by using recursion:**

**def num(n):**

**if n <= 5:**

**print(n)**

**n = n + 1**

**num(n)**

**num(1)**

**#Program to find factorial of a number using recursion:**

**def fact(n):**

**if n == 1 or n == 0:**

**return 1**

**else:**

**return n \* fact(n - 1)**

**factorial = fact(5)**

**print(f"factorial is: {factorial}")**

**#Program to find multiplication of a given number using recursion:**

**def multiply(n, m):**

**if m == 0:**

**return 0**

**elif m == 1:**

**return n**

**else:**

**return n + multiply(n, m - 1)**

**n = int(input("Enter the number to multiply: "))**

**m = int(input("Enter the multiplier: "))**

**result = multiply(n, m)**

**print(f"The result of {n} multiplied by {m} is: {result}")**

**#Another Program to find Multiplications of 5:**

**def mul\_1(n,i):**

**if i <= 10:**

**pro = n \* i**

**print(f"{n} \* {i} = {pro}")**

**mul\_1(n, i + 1)**

**mul\_1(5, 1)**

**#Program to find fibonacci series using recursion:**

**def fibonacci(n):**

**if n <= 0:**

**return []**

**elif n == 1:**

**return [0]**

**elif n == 2:**

**return [0, 1]**

**else:**

**fib\_series = fibonacci(n - 1)**

**fib\_series.append(fib\_series[-1] + fib\_series[-2])**

**return fib\_series**

**n = int(input("Enter the number of terms in the Fibonacci series: "))**

**fib\_series = fibonacci(n)**

**print("Fibonacci series:", fib\_series)**

**#Program to print star Pattern:**

**def star\_pattern(n):**

**if n > 0:**

**print('\*' \* n)**

**star\_pattern(n - 1)**

**n = int(input("Enter the number of rows for the star pattern: "))**

**star\_pattern(n)**

**#Program to Print Star Pattern in middle :**

**n = 5**

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

**print(' ' \* (n - i)  ,end='')**

**print('\* ' \* i)**

**#Another Program to Print Star Pattern in middle :**

**rows = 5**

**for i in range(rows, 0, -1):**

**spaces = ' ' \* (rows - i)**

**stars = '\* ' \* i**

**print(spaces + stars.strip())**

**#Another Program to Print Star Pattern in middle :**

**n = 5**

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

**print(' ' \* (n - i)  ,end='')**

**print('\* ' \* i)**

**for i in range(n , 0, -1):**

**print(' ' \* (n - i) ,end='')**

**print('\* ' \* i)**

**#Program to display Floyds Triangle Pattern:**

**n = 5**

**num = 1**

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

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

**print(num, end=' ')**

**num += 1**

**print()**

**#Program to Print Hollow Square Pattern imp for interview:**

**def hollow\_square(n):**

**for i in range(n):**

**for j in range(n):**

**if i == 0 or i == n - 1 or j == 0 or j == n - 1:**

**print("\*", end=" ")**

**else:**

**print(" ", end=" ")**

**print()**

**size = int(input("Enter the size of the square: "))**

**hollow\_square(size)**

**#Program to Print Number Pyramid Pattern:**

**n = int(input("Enter number of lines: "))**

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

**print(' ' \* (n - i), end='')**

**print((str(i) + ' ') \* i)**

**#Program to Print Permitation of a number:**

**def permutation(n , r):**

**if r > n:**

**return 0**

**fact\_n = 1**

**fact\_n\_r = 1**

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

**fact\_n \*= i**

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

**fact\_n\_r \*= i**

**res = fact\_n // fact\_n\_r**

**return res**

**print (permutation(5, 3))**

**#write a Program to find Combination of a number:**

**def combination(n, r):**

**if r > n:**

**return 0**

**fact\_n = 1**

**fact\_r = 1**

**fact\_n\_r = 1**

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

**fact\_n \*= i**

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

**fact\_r \*= i**

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

**fact\_n\_r \*= i**

**res = fact\_n // (fact\_r \* fact\_n\_r)**

**return res**

**print(combination(5, 3))**

**#Write a recursive function to calculate the sum of the first n natural numbers:**

**def sum\_natural\_numbers(n):**

**if n <= 0:**

**return 0**

**else:**

**return n + sum\_natural\_numbers(n - 1)**

**n = 5**

**result = sum\_natural\_numbers(n)**

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

**#Write a recursive function to calculate the factorial of n:**

**def factorial(n):**

**if n == 0 or n == 1:**

**return 1**

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

**number = 5**

**result = factorial(number)**

**print(f"Factorial of {number} is: {result}")**

**#Write a recursive function to print all elements in a list:**

**def print\_list\_elements(lst, index=0):**

**if index == len(lst):**

**return**

**print(lst[index])**

**print\_list\_elements(lst, index + 1)**

**my\_list = [10, 20, 30, 40, 50]**

**print("List elements:")**

**print\_list\_elements(my\_list)**

**#Program to Math course Problem:**

**def count\_digits\_excluding(number, exclude\_digit):**

**number\_str = str(number)**

**exclude\_digit\_str = str(exclude\_digit)**

**count = 0**

**for digit in number\_str:**

**if digit != exclude\_digit\_str:**

**count += 1**

**return count**

**number = int(input("Enter the numbers: "))**

**exclude\_digit = 4**

**result = count\_digits\_excluding(number, exclude\_digit)**

**print("Output:", result)**

**#Files in Python:**

**file1 = open("sample.txt",'r+')**

**file1.write("This is a sample text.")**

**file1.seek(0)**

**data = file1.read()**

**print(data)**

**file1.close**

**#Create a new file to store details Program:**

**with open("practice.txt", "w") as file:**

**file.write("Hi everyone , we are  learning File Handling I/O,using python, i like programming in python")**

**with open("practice.txt", "r") as file:**

**content = file.read()**

**print("Content of 'practice.txt':")**

**print(content)**

**#Program to create student class:**

**class Student:**

**def \_\_init\_\_(self, name, mark1, mark2, mark3):**

**self.name = name**

**self.mark1 = mark1**

**self.mark2 = mark2**

**self.mark3 = mark3**

**def print\_average(self):**

**average = (self.mark1 + self.mark2 + self.mark3) / 3**

**print(f"Student: {self.name}")**

**print(f"Average Marks: {average:.2f}")**

**student1 = Student("Alice", 85, 90, 78)**

**student1.print\_average()**

**#Program of a Circle using Constructor:**

**import math**

**class Circle:**

**def \_\_init\_\_(self, r):**

**self.radius = r**

**def area(self):**

**return math.pi \* self.radius \*\* 2**

**def circumference(self):**

**return 2 \* math.pi \* self.radius**

**circle1 = Circle(5)**

**print(f"Radius: {circle1.radius}")**

**print(f"Area: {circle1.area():.2f}")**

**print(f"Circumference: {circle1.circumference():.2f}")**

**#Inheritance Program:**

**class A:**

**def feature1(self):**

**print("feature1")**

**def feature2(self):**

**print("feature2")**

**class B:**

**def feature3(self):**

**print("feature3")**

**def feature4(self):**

**print("feature4")**

**a1=A()**

**a1.feature1()**

**a1.feature2()**

**b1=B()**

**b1.feature3()**

**b1.feature4()**

**#Multiple inheritence Program:**

**class Animal:**

**def speak(self):**

**print("Animal speaks")**

**class Dog(Animal):**

**def bark(self):**

**print("Dog barks")**

**class Puppy(Dog):**

**def weep(self):**

**print("Puppy weeps")**

**p = Puppy()**

**p.speak()**

**p.bark()**

**p.weep()**

**#Program with input as:'Prajwal##Ab##':**

**s = input("Enter the string: ")**

**hashes = s.count('#')**

**result = '#' \* hashes**

**result += s.replace('#', '')**

**print(result)**

**#Program to find strong number or not:**

**import math**

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

**sum\_fact = sum(math.factorial(int(d)) for d in str(num))**

**if sum\_fact == num:**

**print(f"{num} is a Strong number")**

**else:**

**print(f"{num} is not a Strong number")**

**#Program to find the number of characters in given string:**

**text = input("Enter a string: ")**

**result = ""**

**i = 0**

**while i < len(text):**

**count = 1**

**while i + 1 < len(text) and text[i] == text[i + 1]:**

**count += 1**

**i += 1**

**result += text[i] + str(count)**

**i += 1**

**print("Compressed output:", result)**

**#Program to find Password valid or not Program:**

**import string**

**def is\_valid\_password(password):**

**if not (8 <= len(password) <= 15):**

**return False**

**lower = sum(1 for c in password if c.islower())**

**upper = sum(1 for c in password if c.isupper())**

**digits = sum(1 for c in password if c.isdigit())**

**special = sum(1 for c in password if c in string.punctuation)**

**if lower < 2:**

**return False**

**if upper < 2:**

**return False**

**if not (1 <= digits <= 3):**

**return False**

**if special != 1:**

**return False**

**return True**

**password = input("Enter password: ")**

**if is\_valid\_password(password):**

**print("Valid Password ")**

**else:**

**print("Invalid Password")**

**#Program to check how many times integer is occured:**

**arr = list(map(int, input("Enter the integers separated by space: ").split()))**

**count = {}**

**for num in arr:**

**if num in count:**

**count[num] += 1**

**else:**

**count[num] = 1**

**for num, freq in count.items():**

**print(f"{num} occurs {freq} times")**

**#Program to check Armstrong number or not a Armstrong number:**

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

**n = len(str(num))**

**if num == sum(int(digit) \*\* n for digit in str(num)):**

**print("Armstrong number")**

**else:**

**print("Not an Armstrong number")**

**#Program to Print small letters into Capital letters:**

**name = input("Enter your name: ").title()**

**print(name)**

**#Program to check string with keys:**

**key = int(input())**

**a = '1024925'**

**res = []**

**for i in range(0,len(a)):**

**b = a[i:key+i]**

**if len(b) == key:**

**res.append(b)**

**print(res)**

**#Program to swap case of each character in string:**

**user\_input = input("Enter the string: ")**

**formatted\_output = ''.join(char.lower() if char.isupper() else char.upper() for char in user\_input)**

**print(formatted\_output)**

**#Program to check how many people does not get House:**

**heights = [10,8,20,3,2]**

**capacity = [12,2,16,4,10]**

**allotted = [False] \* len(capacity)**

**not\_allotted\_count = 0**

**for person\_height in heights:**

**allotted\_flag = False**

**for i in range(len(capacity)):**

**if not allotted [i] and capacity [i] >= person\_height:**

**allotted [i] = True**

**allotted\_flag = True**

**break**

**if not allotted\_flag:**

**not\_allotted\_count += 1**

**print("Number of persons without a house:",not\_allotted\_count)**

**#Program for Class Student using inheritence:**

**class Student:**

**def \_\_init\_\_(self, name, roll):**

**self.name, self.roll = name, roll**

**def show(self):**

**print(f"Name: {self.name}, Roll: {self.roll}")**

**s1 = Student("Prajwal", 104)**

**s2 = Student("Preetam", 107)**

**s1.show()**

**s2.show()**

**#Program of Class Computer using inheritence:**

**class Computer:**

**def \_\_init\_\_(self, brand, battery):**

**self.b, self.p, self.on = brand, battery, False**

**def status(self):**

**print(f"{self.b}: {'ON' if self.on else 'OFF'}, Battery: {self.p}%")**

**def on\_off(self, power):**

**self.on = power if self.p > 0 else False**

**c = Computer("DEll", 75)**

**c.status()**

**c.on\_off(True)**

**c.status()**

**#Program of Class student of average marks using inheritence:**

**class Student:**

**def \_\_init\_\_(self, m1, m2, m3):**

**self.marks = [m1, m2, m3]**

**def average(self):**

**return sum(self.marks) / 3**

**s = Student(80, 75, 90)**

**print("Average marks:", s.average())**

**#Program to check radius of area and circumference of circle:**

**class Circle:**

**def \_\_init\_\_(self, r):**

**self.r = r**

**def area(self):**

**return 3.14 \* self.r \* self.r**

**def circumference(self):**

**return 2 \* 3.14 \* self.r**

**r = float(input("Enter radius: "))**

**c = Circle(r)**

**print("Area:", c.area())**

**print("Circumference:", c.circumference())**

**#Progarm to check area of square and Perimeterof square:**

**class Square:**

**def \_\_init\_\_(self, side):**

**self.side = side**

**def area(self):**

**return self.side \* self.side**

**def perimeter(self):**

**return 4 \* self.side**

**side\_length = float(input("Enter the side of the square: "))**

**sq = Square(side\_length)**

**print("Area of square:", sq.area())**

**print("Perimeter of square:", sq.perimeter())**

**#Program to display amount in bank using abstract method:**

**from abc import ABC, abstractmethod**

**class Payment(ABC):**

**@abstractmethod**

**def pay(self, amount):**

**pass**

**class CreditCardPayment(Payment):**

**def pay(self, amount):**

**print("Connecting to bank server....")**

**print("Authenticating card details....")**

**print(f"Paid ₹{amount} using Credit Card.")**

**p = CreditCardPayment()**

**p.pay(500)**

class MathOperations:

    @staticmethod

    def add(a , b):

        return a + b

    @staticmethod

    def multiply(a , b):

        return a \* b

print(MathOperations.add(10 ,5))

print(MathOperations.multiply(4 , 3))

obj = MathOperations()

print(obj.add(7 , 8))

#Tkinter Program to add numbers:

import tkinter as tk

def add\_numbers():

    try:

        num1 = float(entery1.get())

        num2 = float(entery2.get())

        result.set(f"Result: {num1 + num2}")

    except ValueError:

        result.set("please enter valid numbers")

root = tk.Tk()

root.title("Addition App")

tk.Label(root, text="Enter first number:").grid(row=0, column=0, padx=10, pady=5)

entery1 = tk.Entry(root)

entery1.grid(row=0, column=1, padx=10, pady=5)

tk.Label(root, text="Enter second number:").grid(row=1, column=0, padx=10, pady=5)

entery2 = tk.Entry(root)

entery2.grid(row=1, column=1, padx=10, pady=5)

result = tk.StringVar()

tk.Label(root, textvariable=result).grid(row=2, columnspan=2, padx=10, pady=5)

add\_button = tk.Button(root, text="Add", command=add\_numbers)

add\_button.grid(row=3, columnspan=2, pady=10)

root.mainloop()

#Tkinter Program to substract numbers:

import tkinter as tk

def subtract\_numbers():

    try:

        num1 = float(entry1.get())

        num2 = float(entry2.get())

        result.set(f"Result: {num1 - num2}")

    except ValueError:

        result.set("Please enter valid numbers")

root = tk.Tk()

root.title("Subtraction App")

tk.Label(root, text="Enter first number:").grid(row=0, column=0, padx=10, pady=5)

entry1 = tk.Entry(root)

entry1.grid(row=0, column=1, padx=10, pady=5)

tk.Label(root, text="Enter second number:").grid(row=1, column=0, padx=10, pady=5)

entry2 = tk.Entry(root)

entry2.grid(row=1, column=1, padx=10, pady=5)

result = tk.StringVar()

tk.Label(root, textvariable=result).grid(row=2, columnspan=2, padx=10, pady=5)

subtract\_button = tk.Button(root, text="Subtract", command=subtract\_numbers)

subtract\_button.grid(row=3, columnspan=2, pady=10)

root.mainloop()

#Tkinter Program to multiply numbers:

import tkinter as tk

def mul\_numbers():

    try:

        num1 = float(entry1.get())

        num2 = float(entry2.get())

        result.set(f"Result: {num1 \* num2}")

    except ValueError:

        result.set("Please enter valid numbers")

root = tk.Tk()

root.title("mul App")

tk.Label(root, text="Enter first number:").grid(row=0, column=0, padx=10, pady=5)

entry1 = tk.Entry(root)

entry1.grid(row=0, column=1, padx=10, pady=5)

tk.Label(root, text="Enter second number:").grid(row=1, column=0, padx=10, pady=5)

entry2 = tk.Entry(root)

entry2.grid(row=1, column=1, padx=10, pady=5)

result = tk.StringVar()

tk.Label(root, textvariable=result).grid(row=2, columnspan=2, padx=10, pady=5)

mul\_button = tk.Button(root, text="mul", command=mul\_numbers)

mul\_button.grid(row=3, columnspan=2, pady=10)

root.mainloop()

#Tkinter Program to check Prime number:

import tkinter as tk

def is\_prime(n):

    if n < 2:

        return False

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

        if n % i == 0:

            return False

    return True

def check\_prime():

    try:

        num = int(entry.get())

        if is\_prime(num):

            result.set(f"{num} is a Prime number ")

        else:

            result.set(f"{num} is NOT a Prime number ")

    except ValueError:

        result.set("Please enter a valid integer")

root = tk.Tk()

root.title("Prime Checker App")

tk.Label(root, text="Enter a number:").grid(row=0, column=0, padx=10, pady=5)

entry = tk.Entry(root)

entry.grid(row=0, column=1, padx=10, pady=5)

result = tk.StringVar()

tk.Label(root, textvariable=result).grid(row=1, columnspan=2, padx=10, pady=5)

check\_button = tk.Button(root, text="Check Prime", command=check\_prime)

check\_button.grid(row=2, columnspan=2, pady=10)

root.mainloop()

#Tkinter Program to reverse a string:

import tkinter as tk

def reverse\_string():

    text = entry.get()

    if not text:

        result.set("Please enter a string")

    else:

        reversed\_text = text[::-1]

        result.set(f"Reversed: {reversed\_text}")

root = tk.Tk()

root.title("String Reverser App")

tk.Label(root, text="Enter a string:").grid(row=0, column=0, padx=10, pady=5)

entry = tk.Entry(root)

entry.grid(row=0, column=1, padx=10, pady=5)

result = tk.StringVar()

tk.Label(root, textvariable=result).grid(row=1, columnspan=2, padx=10, pady=5)

check\_button = tk.Button(root, text="Reverse String", command=reverse\_string)

check\_button.grid(row=2, columnspan=2, pady=10)

root.mainloop()

#Tkinter Program to reverse a integer:

import tkinter as tk

def reverse\_integer():

    try:

        num = int(entry.get())

        reversed\_num = int(str(num)[::-1])

        result.set(f"Reversed: {reversed\_num}")

    except ValueError:

        result.set("Please enter a valid integer")

root = tk.Tk()

root.title("Integer Reverser App")

tk.Label(root, text="Enter an integer:").grid(row=0, column=0, padx=10, pady=5)

entry = tk.Entry(root)

entry.grid(row=0, column=1, padx=10, pady=5)

result = tk.StringVar()

tk.Label(root, textvariable=result).grid(row=1, columnspan=2, padx=10, pady=5)

check\_button = tk.Button(root, text="Reverse Integer", command=reverse\_integer)

check\_button.grid(row=2, columnspan=2, pady=10)

root.mainloop()

#Tkinter Program to increment or decrement of numbers:

import tkinter as tk

value = 0

def increment():

    global value

    value += 1

    entry.delete(0, tk.END)

    entry.insert(0, str(value))

    result.set(f"Value: {value}")

def decrement():

    global value

    value -= 1

    entry.delete(0, tk.END)

    entry.insert(0, str(value))

    result.set(f"Value: {value}")

root = tk.Tk()

root.title("Simple Increment/Decrement")

entry = tk.Entry(root)

entry.grid(row=0, column=1, padx=10, pady=5)

entry.insert(0, str(value))

result = tk.StringVar()

result.set(f"Value: {value}")

tk.Button(root, text="Increment", command=increment).grid(row=1, column=0, padx=10, pady=5)

tk.Button(root, text="Decrement", command=decrement).grid(row=2, column=0, padx=10, pady=5)

tk.Label(root, textvariable=result).grid(row=1, column=1, rowspan=2, padx=10, pady=5)

root.mainloop()

Hi everyone , we are learning File Handling I/O,using python, i like programming in python