**DATA SCEINCE & MACHINE LEARNING:**

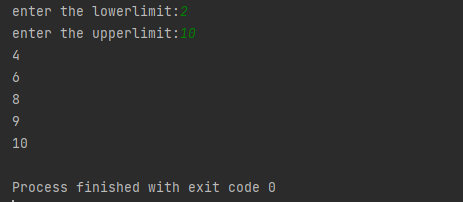
**LAB CYCLE 1**

1. Program to Print all non-Prime Numbers in an Interval

**Program**

first=int(input("enter the lowerlimit:"))  
last=int(input("enter the upperlimit:"))  
for num in range(first,last + 1):  
 if num > 1:  
 for i in range(2,num):  
 if(num % i == 0):  
 print(num)  
 break

**Output**

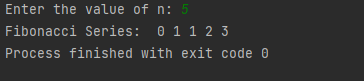


1. Program to print the first N Fibonacci numbers.

**Program**

n = int(input("Enter the value of n: "))  
a = 0  
b = 1  
sum = 0  
count = 1  
print("Fibonacci Series: ", end = " ")  
while(count <= n):  
 print(sum, end = " ")  
 count += 1  
 a = b  
 b = sum  
 sum = a + b

**Output**



1. Given sides of a triangle, write a program to check whether given triangle is an isosceles, equilateral or scalene.

**Program**

x=int(input("enter the side 1"))

y=int(input("enter the side 2"))

z=int(input("enter the side 3"))

if x == y == z:

print("Equilateral Triangle")

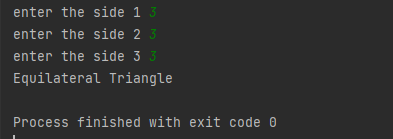
elif x == y or y == z or z == x:

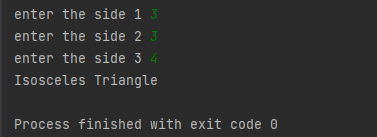
print("Isosceles Triangle")

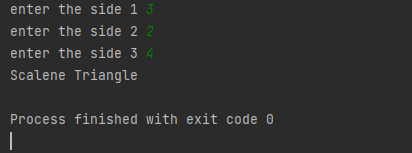
else:

print("Scalene Triangle")

**Output**







1. Program to check whether given pair of number is coprime

**Program**

def \_\_gcd(a, b):

if (a == 0 or b == 0): return 0

if (a == b): return a

if (a > b):

return \_\_gcd(a - b, b)

return \_\_gcd(a, b - a)

def coprime(a, b):

if (\_\_gcd(a, b) == 1):

print("Co-Prime")

else:

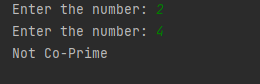
print("Not Co-Prime")

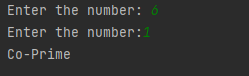
a = int(input("Enter the number:"))

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

coprime(a, b)

**Output**





1. Program to find the roots of a quadratic equation(rounded to 2 decimal places)

**Program**

from math import sqrt

print("Quadratic function : (a \* x^2) + b\*x + c")

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

b = float(input("b: "))

c = float(input("c: "))

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

if r > 0:

num\_roots = 2

x1 = (((-b) + sqrt(r))/(2\*a))

x2 = (((-b) - sqrt(r))/(2\*a))

print("There are 2 roots: %f and %f" % (x1, x2))

elif r == 0:

num\_roots = 1

x = (-b) / 2\*a

print("There is one root: ", x)

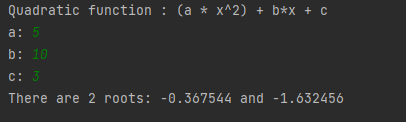
else:

num\_roots = 0

print("No roots, discriminant < 0.")

exit()

**Output**



1. Program to check whether a given number is perfect number or not(sum of factors =number)

**Program**

1. Program to display amstrong numbers upto 1000

**Program**

for num in range(1,1000):

temp=num

sum=0

while temp>0:

digit=temp%10

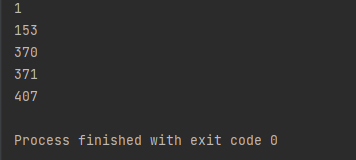
sum=sum+digit\*\*3

temp=temp//10

if sum==num:

print (num)

**Output**



1. Store and display the days of a week as a **List, Tuple, Dictionary, Set.** Also demonstrate different ways to store values in each of them. Display its type also.

Program

list = ["Sun","Mon","Tue","Wed","Thu","Fri","Sat"]

print(type(list))

print(list)

tuple = ("Sun","Mon","Tue","Wed","Thu","Fri","Sat")

print(type(tuple))

print(tuple)

set = {"Sun","Mon","Tue","Wed","Thu","Fri","Sat"}

print(type(set))

print(set)

dict = {

    "d1" : "Sun",

    "d2" : "Mon",

    "d3" : "Tue",

    "d4" : "Wed",

    "d5" : "Thu",

    "d6" : "Fri",

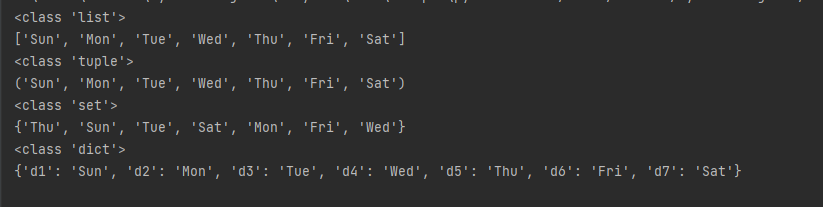
    "d7" : "Sat"

}

print(type(dict))

print(dict)

Output



1. Write a program to add elements of given 2 lists

**Program**

n = int(input("Enter the size of the list1 "))

lt1 = list(int(num) for num in input("Enter the list items separated by space ").strip().split())[:n]

n = int(input("Enter the size of the list2 "))

lt2 = list(int(num) for num in input("Enter the list items separated by space ").strip().split())[:n]

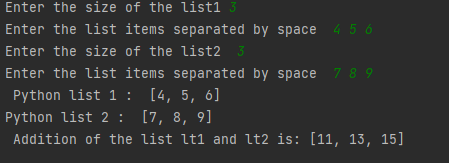
print(" Python list 1 : " , lt1)

print("Python list 2 : " ,lt2)

res\_lt = [lt1[x] + lt2[x] for x in range(len(lt1))]

print(" Addition of the list lt1 and lt2 is: " + str(res\_lt))

**Output**



1. Write a program to find the sum of 2 matrices using nested List.

**Program**

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

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

print("Enter the elements of First Matrix:")

m1= [[int(input()) for i in range(c)] for i in range(r)]

print("Enter the elements of Second Matrix:")

m2=[[int(input()) for i in range(c)] for i in range(r)]

print("First Matrix is: ")

for n in m1:

print(n)

print("Second Matrix is:")

for n in m2:

print(n)

r=[[0 for i in range(c)] for i in range(r)]

print("Matrix After Addition:")

for i in range(len(m1)):

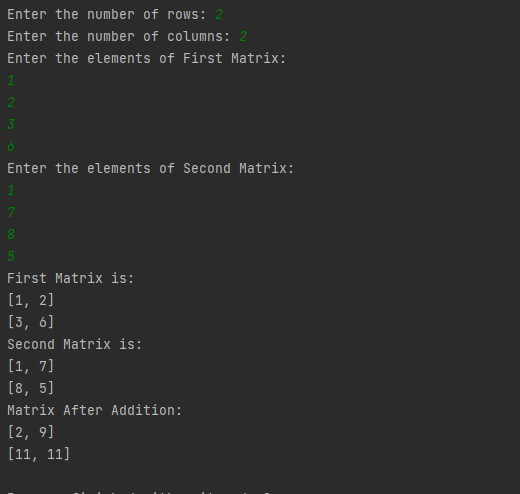
for j in range(len(m2)):

r[i][j] = m1[i][j] + m2[i][j]

for n in r:

print(n)

**Output**



1. Write a program to perform bubble sort on a given set of elements.

**Program**

def bubble\_sort(list1):

for i in range(0, len(list1) - 1):

for j in range(len(list1) - 1):

if (list1[j] > list1[j + 1]):

temp = list1[j]

list1[j] = list1[j + 1]

list1[j + 1] = temp

return list1

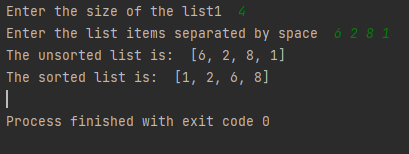
n = int(input("Enter the size of the list1 "))

list1 = list(int(num) for num in input("Enter the list items separated by space ").strip().split())[:n]

print("The unsorted list is: ", list1)

print("The sorted list is: ", bubble\_sort(list1))

**Output**



1. Program to find the count of each vowel in a string(use dictionary)

**Program**

vowels = 'aeiou'

user\_input = input("Enter a string: ")

string = user\_input.casefold()

count\_vowels = {}.fromkeys(vowels, 0)

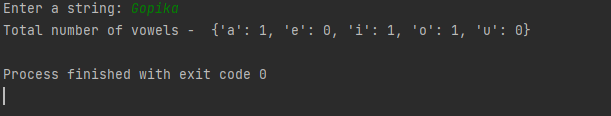
for x in string:

if x in count\_vowels:

count\_vowels[x] += 1

print("Total number of vowels - ", count\_vowels)

**Output**



1. Write a Python program that accept a positive number and subtract from this number the sum of its digits and so on. Continues this operation until the number is positive

Program

def repeat\_times(n):

s = 0

n\_str = str(n)

while (n > 0):

n -= sum([int(i) for i in list(n\_str)])

n\_str = list(str(n))

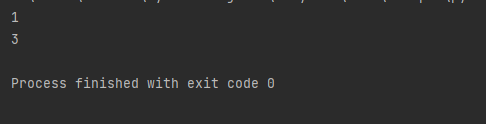
s += 1

return s

print(repeat\_times(9))

print(repeat\_times(21))

Output



1. Write a Python program that accepts a 10 digit mobile number, and find the digits which are absent in a given mobile number

**Program**

mobile = input('Please enter a mobile number: ' )

all = '0123456789'

print('missing digits are ', set(all) - set(mobile))

**Output**

