LAB CYCLE-1

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

**PROGRAM**

import math  
def is\_not\_prime(n):  
 ans = False  
 for i in range(2, int(math.sqrt(n)) + 1):  
 if n % i == 0:  
 ans = True  
 return ans  
print("Nonprime numbers between 1 to 10:")  
for x in filter(is\_not\_prime, range(1, 10)):  
 print(x)

**OUTPUT**

Nonprime numbers between 1 to 10:

4

6

8

9

**2. Program to print the first N Fibonacci numbers.**

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

**OUTPUT**

Enter the nth value: 6

Fibonacci Series : 0 1 1 2 3 5

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

**PROGRAM**

print("Enter lengths of the triangle")  
x = int(input("x: "))  
y = int(input("y: "))  
z = int(input("z: " ))  
if (x==y==z):  
 print("Equilateral Triangle")  
elif(x==y or y==z or z==x):  
 print("Isolates Triangle")  
else:  
 print("Scalene Triangle")

**OUTPUT**

Enter lengths of the triangle

x: 5

y: 8

z: 8

Isolates Triangle

**4. Program to check whether given pair of number is coprime.**

**PROGRAM**

def are\_coprime(a, b):  
 hcf = 1  
  
 for i in range(1, a + 1):  
 if a % i == 0 and b % i == 0:  
 hcf = i  
  
 return hcf == 1  
  
first = int(input('Enter first number: '))  
second = int(input('Enter second number: '))  
  
if are\_coprime(first, second):  
 print('%d and %d are CO-PRIME' % (first, second))  
else:  
 print('%d and %d are NOT CO-PRIME' % (first, second))

**OUTPUT**

Enter first number: 21

Enter second number: 6

21 and 6 are NOT CO-PRIME

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

**PROGRAM**

import cmath  
a=float(input("Enter the value of a: "))  
b=float(input("Enter the value of b:" ))  
c=float(input("Enter the value of c:"))  
d=(b\*b)-(4\*a\*c)  
if(d>0):  
 r1 = (-b + cmath.sqrt(d)) / (2 \* a)  
 r2 = (-b - cmath.sqrt(d)) / (2 \* a)  
 print("The roots are real and different",r1,r2)  
elif(d==0):  
 r1 = r2 = -b / 2 \* a  
 print("roots are real and equal", r1)  
else:  
 real = -b / (2 \* a)  
 img = cmath.sqrt(d) / (2 \* a)  
 print("complex roots", real, "+", img, "and", real, "-", img)

**OUTPUT**

Enter the value of a: 4

Enter the value of b:5

Enter the value of c:6

complex roots -0.625 + 1.0532687216470449j and -0.625 - 1.0532687216470449j

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

**PROGRAM**

n = int(input("Enter the Number:"))  
sum = 0  
for i in range(1, n):  
 if(n % i == 0):  
 sum = sum + i  
if(sum == n):  
 print("The Entered Number Is a Perfect Number")  
else:  
 print("The Entered Number Is Not a Perfect Number")

**OUTPUT**

Enter the Number:125

The Entered Number Is Not a Perfect Number

**7. Program to display amstrong numbers upto 1000.**

**PROGRAM**

l = 100  
u = 1000  
for num in range(l, u + 1):  
 order=len(str(num))  
 sum = 0  
 temp = num  
 while temp > 0:  
 digit = temp % 10  
 sum += digit \*\* order  
 temp //=10  
 if num == sum:  
 print(num)

**OUTPUT**

153

370

371

407

**8. 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))

**OUTPUT**

<class 'list'>

['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']

<class 'tuple'>

('Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat')

<class 'set'>

{'Thu', 'Mon', 'Sun', 'Sat', 'Fri', 'Tue', 'Wed'}

<class 'dict'>

{'d1': 'Sun', 'd2': 'Mon', 'd3': 'Tue', 'd4': 'Wed', 'd5': 'Thu', 'd6': 'Fri', 'd7': 'Sat'}

**9. Write a program to add elements of given 2 lists.**

**PROGRAM**

thislist=["Athira","Bhagya","Jeena"]  
thislist2=["Simmy","Meenu","Nidhin"]  
thislist.extend(thislist2)  
print(thislist

**OUTPUT**

['Athira', 'Bhagya', 'Jeena', 'Simmy', 'Meenu', 'Nidhin']

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

**PROGRAM**

x = [[1,12,2],  
 [4,5,12],  
 [8,10,15]]  
y = [[1,12,2],  
 [4,5,12],  
 [8,10,15]]  
res = [[0,0,0],  
 [0,0,0],  
 [0,0,0]]  
  
for i in range(len(x)):  
 for j in range(len(x[0])):  
 res[i][j] = x[i][j] + y[i][j]  
for r in res:  
 print(r)

**OUTPUT**

[2, 24, 4]

[8, 10, 24]

[16, 20, 30]

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

**PROGRAM**

a = []  
number = int(input("Please Enter the Total Number of Elements : "))  
for i in range(number):  
 value = int(input("Please enter the %d Element of List1 : " %i))  
 a.append(value)  
  
for i in range(number -1):  
 for j in range(number - i - 1):  
 if(a[j] > a[j + 1]):  
 temp = a[j]  
 a[j] = a[j + 1]  
 a[j + 1] = temp  
  
print("The Sorted List in Ascending Order : ", a)

**OUTPUT**

Please Enter the Total Number of Elements : 5

Please enter the 0 Element of List1 : 8

Please enter the 1 Element of List1 : 45

Please enter the 2 Element of List1 : 1

Please enter the 3 Element of List1 : 2

Please enter the 4 Element of List1 : 89

The Sorted List in Ascending Order : [1, 2, 8, 45, 89]

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

**PROGRAM**

string=input("Enter string:")  
vowels=0  
for i in string:  
 if i=='a' or i=='e' or i=='i' or i=='o' or i=='u' or i=='A' or i=='E' or i=='I' or i=='O' or i=='U':  
 vowels=vowels+1  
print("Number of vowels are:")  
print(vowels

**OUTPUT**

Enter string:ABSTRACT

Number of vowels are:

2

**13. 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

3

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

**PROGRAM**

def absent\_digits(n):  
 all\_nums = set([0,1,2,3,4,5,6,7,8,9])  
 n = set([int(i) for i in n])  
 n = n.symmetric\_difference(all\_nums)  
 n = sorted(n)  
 return n  
print(absent\_digits([9,5,2,6,0,1,4,6,8,4]))

**OUTPUT**

[3, 7]