1. Write a Python Program to Find LCM?

Ans1.

import logging as lg

lg.basicConfig(level = lg.INFO)

def LCM():

try:

lg.info('LCM of 2 numbers')

a=int(input('Enter the first number : '))

b=int(input('Enter the second number : '))

if a>b:

max1=a

else:

max1=b

while(True):

if max1%a==0 and max1%b==0:

lcm = max1

break

max1=max1+1

return lcm

except Exception as e:

return 'Error has occured'+str(e)

LCM()

1. Write a Python Program to Find HCF?

Ans2.

import logging as lg

lg.basicConfig(level = lg.INFO)

def HCF():

try:

lg.info('HCF of 2 numbers')

a=int(input('Enter the first number : '))

b=int(input('Enter the second number : '))

if a<b:

min1=a

else:

min1=b

for i in range(1,min1+1):

if a%i==0 and b%i==0:

hcf = i

return hcf

except Exception as e:

return 'Error has occured'+str(e)

HCF()

1. Write a Python Program to Convert Decimal to Binary, Octal and Hexadecimal?

Ans3.

class conv:

def \_\_init\_\_(self,n):

self.n=n

def D\_TO\_B(self):

l=[]

for i in range(self.n):

l.append(self.n%2)

self.n=self.n//2

if self.n==0:

break

return int(''.join(map(str, l[::-1])))

def D\_TO\_O(self):

l=[]

for i in range(self.n):

l.append(self.n%8)

self.n=self.n//8

if self.n==0:

break

return int(''.join(map(str, l[::-1])))

def D\_TO\_H(self):

l=[]

for i in range(self.n):

l.append(self.n%16)

self.n=self.n//16

if self.n==0:

break

return int(''.join(map(str, l[::-1])))

1. Write a Python Program To Find ASCII value of a character?

Ans4.

import logging as lg

lg.basicConfig(level = lg.INFO)

def ascii():

try:

lg.info('To convert character to ascii')

a=input('Enter the character:')

return ord(a)

except Exception as e:

return 'Error has occured: '+str(e)

ascii()

1. Write a Python Program to Make a Simple Calculator with 4 basic mathematical operations?

Ans5.

import logging as lg

lg.basicConfig(level = lg.INFO)

class calc:

def \_\_init\_\_(self,a,b):

self.a=a

self.b=b

def add(self):

try:

return self.a+self.b

except Exception as e:

return 'Error has occured: '+str(e)

def sub(self):

try:

return (abs(self.a-self.b))

except Exception as e:

return 'Error has occured: '+str(e)

def mul(self):

try:

return self.a\*self.b

except Exception as e:

return 'Error has occured: '+str(e)

def div(self):

try:

return (self.a)/(self.b)

except Exception as e:

return 'Error has occured: '+str(e)

c=calc(1,2)

#c.add or c.sub or c.mul or c.div