**AIM:-**

Write a program to visualize the usage of following :

i)Methods and Lambda methods

ii).Class and objects

iii).Inheritance (OOP)

iv).try…except (Exception Handling)

**THEORY:-**

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

Syntax – def method\_name(self):

Lambda method - A lambda method is a small anonymous method.

A lambda method can take any number of arguments, but can only have one expression.

Syntax : lambda arguments : expression.

**PROGRAM:-**

**A]**

class experiment:

def sum():

a=int(input("enter first number:-"))

b=int(input("enter second nmuber:-"))

print("sum of",a,"and",b,"is",a+b)

obj=experiment

obj.sum()

**B]**

a=int(input("enter first number:-"))

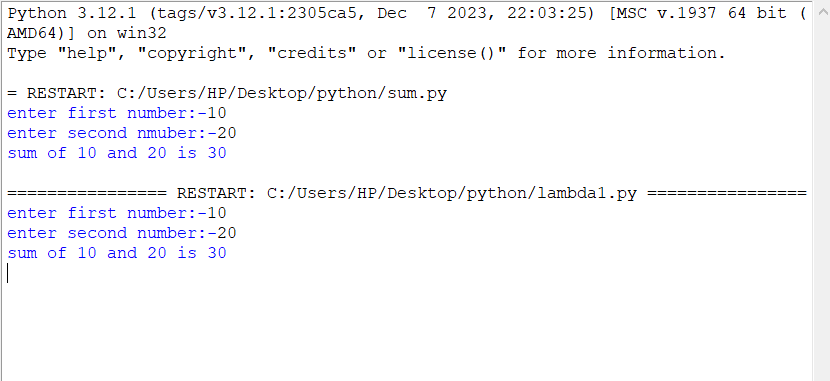
b=int(input("enter second number:-"))

c=lambda a,b:a+b

d=c(a,b)

print("sum of",a,"and",b,"is",d)

**OUTPUT:-**



**THEORY:-**

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

Syntax :

class class\_name:

obj1 = class\_name()

**PROGRAM:-**

class student:

def information():

name=input("enter the name of the student:-")

age=int(input("enter the age of the student:-"))

marks=float(input("enter total marks:-"))

print("NAME OF STUDENT:-",name)

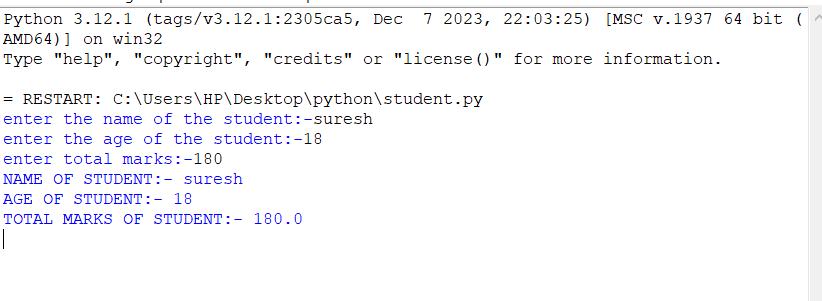
print("AGE OF STUDENT:-",age)

print("TOTAL MARKS OF STUDENT:-",marks)

obj=student

obj.information()

**OUTPUT:-**



**THEORY:-**

Inheritance allows us to define a class that inherits all the methods and properties from another class.

**Parent class** is the class being inherited from, also called base class.

**Child class** is the class that inherits from another class, also called derived class.

**PROGRAM:-**

**A]**

class Father():

def father\_info(self):

self.father\_name = str(input("Enter the name of the father: "))

self.father\_age = int(input("Enter the age of father: "))

class Mother():

def mother\_info(self):

self.mother\_name = str(input("Enter the mother's name: "))

self.mother\_age = int(input("Enter the age of the mother: "))

class Child(Father, Mother):

def son\_info(self):

son\_name = str(input("Enter the son's name: "))

son\_age = int(input("Enter the age of the son: "))

print("The father of", son\_name, "is", self.father\_name)

print("The mother of", son\_name, "is", self.mother\_name)

print("The age of father is", self.father\_age)

print("The age of mother is", self.mother\_age)

obj1 = Child()

obj1.father\_info()

obj1.mother\_info()

obj1.son\_info()

**B]**

class grandfather:

def ginfo(self):

self.gname=input("enter name of grandfather:-")

self.gage=int(input("enter age of grandfather:-"))

class father(grandfather):

def finfo(self):

self.fname=input("enter name of father:-")

self.fage=int(input("enter age of father:-"))

class son(father):

def sinfo(self):

self.sname=input("enter name of son:-")

self.sage=int(input("enter age of son:-"))

print("grandson of",self.gname,"is",self.sname)

print("son of ",self.fname,"is",self.sname)

print("age of grandfather is:-",self.gage)

print("age of father is:-",self.fage)

print("age of son is:-",self.sage)

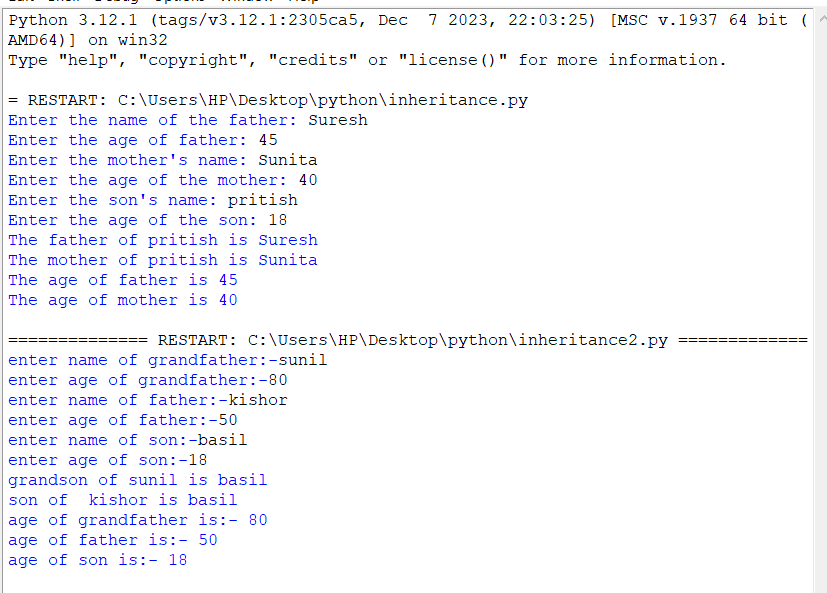
obj=son()

obj.ginfo()

obj.finfo()

obj.sinfo()

**OUTPUT:-**



**THEORY:-**

The try block lets you test a block of code for errors.

The except block lets you handle the error.

The else block lets you execute code when there is no error.

The finally block lets you execute code, regardless of the result of the try- and except blocks.

**PROGRAM:-**

try :

num1 = int(input("Enter the numerator :- "))

num2 = int(input("Enter the denominator :- "))

quo = num1/num2

print("The quotient after division is" , quo)

except ZeroDivisionError:

print("The denominator can never be zero ")

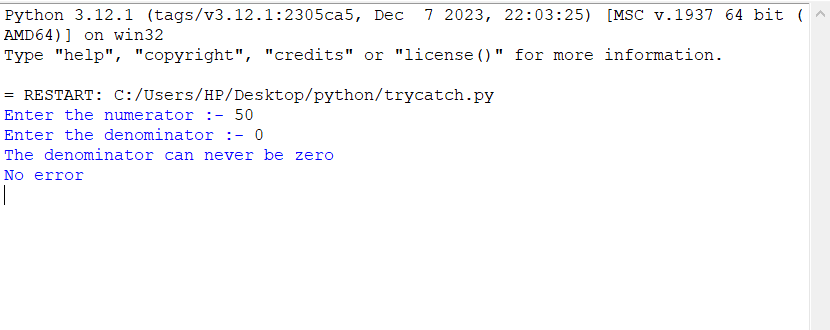
else :

print("There is no error ")

finally :

print("No error ")

**OUTPUT:-**



**CONCLUSION:-**

We executed methods , classes , objects , multiple inheritance , multi-level inheritance and try…except blocks.