Practical No.7

A ) Design a class that store the information of student and display the same .

CODE:

class Student:

def \_\_init\_\_(self,name,age,mob\_no,percentage):

self.name = name

self.age = age

self.mob\_no = mob\_no

self.percentage = percentage

def display(self):

print(f" student name : {self.name}")

print(f" student age : {self.age}")

print(f" student mob\_no : {self.mob\_no}")

print(f" student percentage is : {self.percentage}")

s1= Student('Shruti',20,1234567890,'85%')

s2= Student('Gayatri',20,9876543210,'80%')

s3= Student('Ved',19,465873945,'75%')

print('\n--------student information-----------\n')

s1.display()

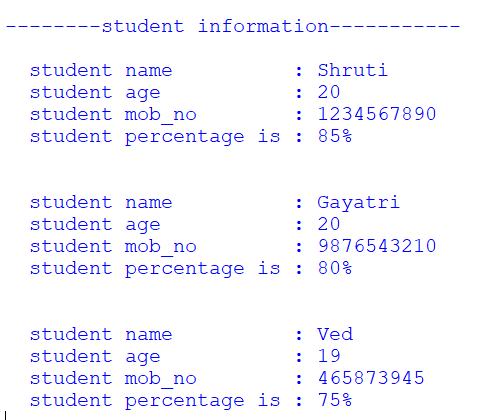
print("\n")

s2.display()

print("\n")

s3.display()

OUTPUT:



1. implement the concept of inheritance using python.
2. Single level inheritance :

CODE:

import math

class Area:

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

self.radius = radius

def find\_area(self):

ac = math.pi \*( math.sqrt(self.radius))

print(f" area of circle is : {ac}")

class Perimeter(Area):

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

Area.\_\_init\_\_(self,radius)

def find\_perimeter(self):

pc = (2 \* math.pi )\* (math.sqrt(self.radius))

print(f" perimeter of circle is : {pc}")

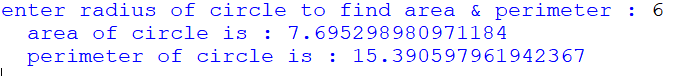
r = float(input('enter radius of circle to find area & perimeter : '))

p = Perimeter(r)

p.find\_area()

p.find\_perimeter()

OUTPUT:



1. Multiple inheritance :

CODE:

class Car :

def \_\_init\_\_(self,c\_cpy\_name,c\_model\_name,c\_fuel\_type):

self.c\_cpy\_name = c\_cpy\_name

self.c\_model\_name = c\_model\_name

self.c\_fuel\_type = c\_fuel\_type

def disp(self):

print('CAR :')

print(f" Comany Name : {self.c\_cpy\_name}")

print(f" Model Name : {self.c\_model\_name}")

print(f" fuel Type : {self.c\_fuel\_type}")

class Bike :

def \_\_init\_\_(self,b\_cpy\_name,b\_model\_name,b\_fuel\_type):

self.b\_cpy\_name = b\_cpy\_name

self.b\_model\_name = b\_model\_name

self.b\_fuel\_type = b\_fuel\_type

def disp(self):

print('BIKE :')

print(f" Comany Name : {self.b\_cpy\_name}")

print(f" Model Name : {self.b\_model\_name}")

print(f" fuel Type : {self.b\_fuel\_type}")

class Person (Bike,Car) :

def \_\_init\_\_(self,c\_cpy\_name,c\_model\_name, c\_fuel\_type,b\_cpy\_name,b\_model\_name,b\_fuel\_type,p\_name,p\_mob):

Car.\_\_init\_\_(self,c\_cpy\_name,c\_model\_name, c\_fuel\_type)

Bike.\_\_init\_\_(self,b\_cpy\_name,b\_model\_name,b\_fuel\_type)

self.p\_name = p\_name

self.p\_mob = p\_mob

def disp(self):

super().disp()

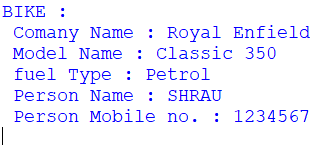
print(f" Person Name : {self.p\_name}")

print(f" Person Mobile no. : {self.p\_mob}")

p = Person( 'Toyota' , 'Corolla' , 'Petrol/disel/hybrid','Royal Enfield','Classic 350','Petrol','SHRAU' , '1234567')

p.disp()

OUTPUT:



1. Multi level inheritance :

CODE:

class Person:

def \_\_init\_\_(self,name,age):

self.name = name

self.age = age

def disp(self):

print(f" Name : {self.name}")

print(f" Age : {self.age}")

class Emp(Person):

def \_\_init\_\_(self,name,age,e\_id,position):

super().\_\_init\_\_(name,age)

self.e\_id = e\_id

self.position = position

def disp(self):

super().disp()

print(f" Employe Id : {self.e\_id}")

print(f" Employe position: {self.position}")

class Manage(Emp):

def \_\_init\_\_(self,name,age,e\_id,position,dept,teamsize):

super().\_\_init\_\_(name,age,e\_id,position)

self.dept = dept

self.teamsize = teamsize

def disp(self):

print('Information :')

super().disp()

print(f" Department : {self.dept}")

print(f" Teamsize : {self.teamsize}")

m1 = Manage('Reva',30,1102,'employee','Human resource',20)

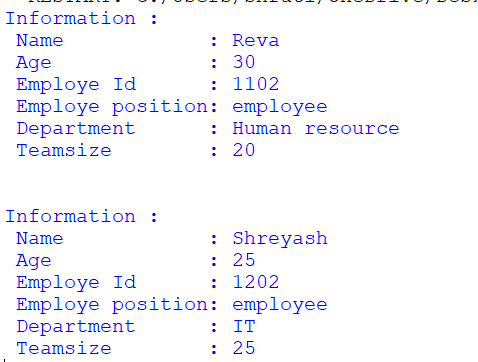
m2 = Manage('Shreyash',25,1202,'employee','IT',25)

m1.disp()

print("\n")

m2.disp()

OUTPUT:



1. Hybrid inheritance :

CODE:

class Student:

def \_\_init\_\_(self,name,age):

self.name = name

self.age = age

def disp(self):

print(f" Name : {self.name}")

print(f" Age : {self.age}")

class Subject(Student):

def \_\_init\_\_(self,name,age,s1):

Student.\_\_init\_\_(self,name,age)

self.s1 = s1

def sub(self):

Student.disp(self)

print(f" subject : {self.s1}")

class Internal(Subject):

def \_\_init\_\_(self,name,age,s1,i\_marks):

Subject.\_\_init\_\_(self,name,age,s1)

self.i\_marks = i\_marks

def i\_disp(self):

Subject.sub(self)

print(f" Internal Marks of 1st subject : {self.i\_marks}")

class External(Subject):

def \_\_init\_\_(self,name,age,s1,e\_marks):

Subject.\_\_init\_\_(self,name,age,s1)

self.e\_marks = e\_marks

def e\_disp(self):

#Subject.sub(self)

print(f" External Marks of 1st subject : {self.e\_marks}")

class Total(Internal,External):

def \_\_init\_\_(self,name,age,s1,i\_marks,e\_marks):

Internal.\_\_init\_\_(self,name,age,s1,i\_marks)

External.\_\_init\_\_(self,name,age,s1,e\_marks)

def total\_marks(self):

Internal.i\_disp(self)

External.e\_disp(self)

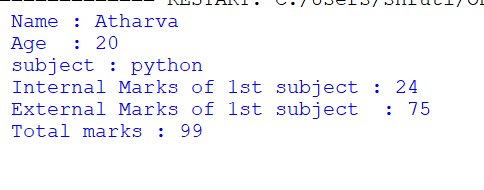
t = self.i\_marks + self.e\_marks

print(f" Total marks : {t}")

t = Total('Atharva',20,'python', 24,75)

t.total\_marks()

OUTPUT:



1. Hierachical inheritance :

CODE:

class Father:

def \_\_init\_\_(self,f\_amount):

self.f\_amount = f\_amount

def father\_cont(self):

print(f" father contribution :{self.f\_amount} ")

class Son1(Father):

def \_\_init\_\_(self,f\_amount,s1\_amount):

super().\_\_init\_\_(f\_amount)

self.s1\_amount = s1\_amount

def son1\_cont(self):

super().father\_cont()

total = self.f\_amount + self.s1\_amount

print(f" 1st son investment :{self.s1\_amount}")

print(f" total contribution of 1st son :{total} ")

class Son2(Father):

def \_\_init\_\_(self,f\_amount,s2\_amount):

super().\_\_init\_\_(f\_amount)

self.s2\_amount = s2\_amount

def son1\_cont(self):

super().father\_cont()

total = self.f\_amount + self.s2\_amount

print(f" 2nd son investment :{self.s2\_amount}")

print(f" total contribution of 2nd son :{total} ")

s1 = Son1( 500000,40000)

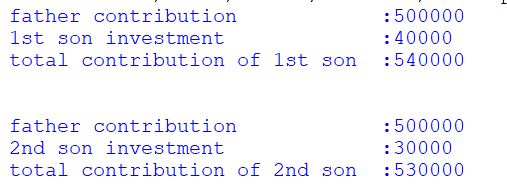
s2 = Son2( 500000,30000)

s1.son1\_cont()

print("\n")

s2.son1\_cont()

OUTPUT:



C ) Create a class called Numbers, which has a single class attribute called MULTIPLIER, and a constructor which takes the parameters x and y (these should all be numbers).

i. Write a method called add which returns the sum of the attributes x and y.

ii. Write a class method called multiply, which takes a single number

parameter a and returns the product of a and MULTIPLIER.

iii. Write a static method called subtract, which takes two number parameters, b and c, and returns b - c.

iv. Write a method called value which returns a tuple containing the values of x and y.

Make this method into a property, and write a setter and a deleter for manipulating the values of x and y.

CODE :

class Number :

multiplier = 8

def \_\_init\_\_(self,x,y):

self.\_x = x

self.\_y = y

def add(self):

return self.\_x + self.\_y

def multiply(self,a):

return a \* self.multiplier

@staticmethod

def subtract(b,c):

return b - c

@property

def value(self):

return (self.\_x,self.\_y)

@value.setter

def value(self,new\_val):

self.\_x,self.\_y = new\_val

@value.deleter

def value(self):

del self.\_x

del self.\_y

n = Number(12,8)

print(f" Addition : {n.add()}")

print(f" Multiplication : {n.multiply(12)}")

print(f" Subtraction : {Number.subtract(99,75)}")

print(f" Current Values : {n.value}")

n.value = (100,400)

print(f" Updated Values : {n.value}")

del n.value

OUTPUT:

