class Student:

def \_init\_(self,marks): #default constructor

self.marks=marks

self.\_\_marks=marks #private

def getter(self):

return self.\_\_marks

def setter(self,marks): o/p:79

self.\_\_marks=marks

s1=Student(79)

#set the data

s1.setter(79)

#get the data

ans=s1.getter()

print(ans)

abstraction

->hiding the necessary information

->providing useful onfo only

->abc module,abstract method

->@abstaract method

1.what is meant by decarator? what is the use of it?

2.whta is meant by ABC module? #in this abc is module and ABC is class

PROGRAM:

from abc import ABC,abstractmethod o/p:car is moving

class four\_wheeler(ABC):

@abstractmethod

def engine():

return "hello this is process of making car"

class swift(four\_wheeler):

def car\_start():

return"car is moving"

car\_1=swift

ans=car\_1.car\_start()

print(ans)

#INHERITANCE:

1.SINGLE

2.MULTILEVEL

2.MULTIPLE

4.HIRARCHIAL

5.HYBRID

#SINGLE INHERITANCE

class father: #parent class

def father\_method():

return "This is father method"

#inheriting father class

class child(father);

def child\_method():

return "This is child method"

parent\_object=father

child\_object=child

print(parent\_object.father\_method())

print(child\_object.child\_method())

print(child\_object.father\_method())

#MULTIPLE INHERITANCE

class father:

def father\_method():

return "This is father method"

class mother:

def mother\_method():

return "This is mother method"

class child(father,mother):

def child\_method():

return "I have properties of mother and father" o/p:This is father method

This is mother method

I have properties of mother and father

This is father method

This is mother method

father\_obj=father

mother\_obj=mother

child\_obj=child

print(father\_obj.father\_method())

print(mother\_obj.mother\_method())

print(child\_obj.child\_method())

print(child\_obj.father\_method())

print(child\_obj.mother\_method())

#MULTilevel INHERITANCE

class grandfather:

def grandfather\_method():

return "This is grandfather method"

class father:

def father\_method():

return "This is father method"

class child(grandfather,father):

def child\_method():

return "I have properties of grand father and father"

grandfather\_obj=grandfather

father\_obj=father

child\_obj=child

print(grandfather\_obj.grandfather\_method())

print(father\_obj.father\_method())

print(child\_obj.child\_method())

print(child\_obj.grandfather\_method())

print(child\_obj.father\_method())

o/p:

This is grandfather method

This is father method

I have properties of grand father and father

This is grandfather method

This is father method

#HIRARCHIAL INHERITANCE

class grandfather:

def grandfather\_method():

return "This is grandfather method"

class grandmother:

def grandmother\_method():

return "This is grandmother method"

class father:

def father\_method():

return "This is father method"

class child(grandfather,grandmother,father):

def child\_method():

return "I have properties of grand father,grandmother and father"

grandfather\_obj=grandfather

grandmother\_obj=grandmother

father\_obj=father

child\_obj=child

print(grandfather\_obj.grandfather\_method())

print(grandmother\_obj.grandmother\_method())

print(father\_obj.father\_method())

print(child\_obj.child\_method())

print(child\_obj.grandfather\_method())

print(child\_obj.grandmother\_method())

print(child\_obj.father\_method())

o/p:

This is grandfather method

This is grandmother method

This is father method

I have properties of grand father,grandmother and father

This is grandfather method

This is grandmother method

This is father method

#HYBRID

class grandfather:

def grandfather\_method():

return "This is grandfather method"

class father:

def father\_method():

return "This is father method"

class mother:

def mother\_method():

return "This is mother method"

class child(grandfather, father,mother):

def child\_method():

return "I have properties of grand father,mother and father"

grandfather\_obj=grandfather

father\_obj=father

mother\_obj=mother

child\_obj=child

print(grandfather\_obj.grandfather\_method())

print(father\_obj.father\_method())

print(mother\_obj.mother\_method())

print(child\_obj.child\_method())

print(child\_obj.grandfather\_method())

print(child\_obj.mother\_method())

print(child\_obj.father\_method())

o/p:

This is grandfather method

This is father method

This is mother method

I have properties of grand father,mother and father

This is grandfather method

This is mother method

This is father method

#POLYMORPHISAM

class Animal:

def speak():

return "animal is speaking"

class Bird(Animal):

def speak():

return "bird is speaking"

animal\_object=Animal

bird\_object=Bird

print(animal\_object.speak())

print(bird\_object.speak())

o/p:

animal is speaking

bird is speaking

SUM OF SQUAE OF ANUMBERS

n=3645

sum=0

while(n>0):

digit=n%10

sum=(digit\*digit)+sum

n=n//10

print(sum)

O/P:86