**Experiment 11**

**Aim:** Write python programs to implement Abstract methods and interfaces in python.

**Program:** Abstract method without working using "@abstractmethod".

>>> from abc import ABC

>>> class Vehicle(ABC):

... def Wheels(self):

... pass

...

>>>

>>> class Car(Vehicle):

... def Wheels(self):

... print("A car has 4 wheels")

...

>>>

>>> class Motorcycle(Vehicle):

... def Wheels(self):

... print("A motorcycle has 2 wheels")

...

>>>

>>> class Autorickshaw(Vehicle):

... def Wheels(self):

... print("An autorickshaw has 3 wheels")

...

>>>

>>> obj1=Car()

>>> obj2=Motorcycle()

>>> obj3=Autorickshaw()

>>> obj1.Wheels()

A car has 4 wheels

>>> obj2.Wheels()

A motorcycle has 2 wheels

>>> obj3.Wheels()

An autorickshaw has 3 wheels

**Program:** Abstract method using "@abstractmethod".

>>> from abc import ABC,abstractmethod

>>> class Vehicle(ABC):

... @abstractmethod

... def Wheels(self):

... pass

...

>>> class Car(Vehicle):

... def Wheels(self):

... print("A car has 4 wheels")

...

>>> class Tuktuk(Vehicle):

... def Wheels(Self):

... print("A tuktuk has 3 wheels")

...

>>> obj1=Car()

>>> obj2=Tuktuk()

>>> obj1.Wheels()

A car has 4 wheels

>>> obj2.Wheels()

A tuktuk has 3 wheels

**Program:** Abstract method without Zope interface Module.

>>> from abc import ABC

>>> class Vehicle(ABC):

... def seats(self):

... pass

... def wheels(self):

... pass

...

>>> class Car(Vehicle):

... def seats(self):

... print("Car has 6 seats")

... def wheels(self):

... print("Car has 4 wheels")

...

>>> class TukTuk(Vehicle):

... def seats(self):

... print("Bus has 64 seats")

... def wheels(self):

... print("Bus has 6 wheels")

...

>>> obj1=Car()

>>> obj2=TukTuk()

>>> obj1.seats()

Car has 6 seats

>>> obj1.wheels()

Car has 4 wheels

>>> obj2.seats()

Bus has 64 seats

>>> obj2.wheels()

Bus has 6 wheels

**Program:** Abstract method with Zope interface Module.

>>> from zope.interface import Interface, implementer

>>> class Vehicle(Interface):

... def seats():

... pass

...

>>> @implementer(Vehicle)

... class Car:

... def seats(self):

... print("Car: 6 seats")

...

>>> @implementer(Vehicle)

... class Bus:

... def seats(self):

... print("Bus: 64 seats")

...

>>> A1=Car()

>>> A2=Bus()

>>> A1.seats()

Car: 6 seats

>>> A2.seats()

Bus: 64 seats

**Conclusion:** Hence we have implemented Abstract methods and interfaces in python.