**Abstract Class**

1.Create an abstract class with abstract and non-abstract methods.

from abc import ABC, abstractmethod

#Abstract Class

class Bank(ABC):

def bank\_info(self):

print("Welcome to bank")

@abstractmethod

def interest(self):

"Abstarct Method"

pass

#Sub class/ child class of abstract class

class SBI(Bank):

def balance(self):

print("Balance is 100")

s= SBI()

s.bank\_info ()

s.balance()

2. Create a sub class for an abstract class. Create an object in the child class for the abstract class and access the non-abstract methods

from abc import ABC, abstractmethod

class Polygon(ABC):

@abstractmethod

def noofsides(self):

pass

class Triangle(Polygon):

# overriding abstract method

def noofsides(self):

print("I have 3 sides")

class Pentagon(Polygon):

# overriding abstract method

def noofsides(self):

print("I have 5 sides")

class Hexagon(Polygon):

# overriding abstract method

def noofsides(self):

print("I have 6 sides")

class Quadrilateral(Polygon):

# overriding abstract method

def noofsides(self):

print("I have 4 sides")

# Driver code

R = Triangle()

R.noofsides()

K = Quadrilateral()

K.noofsides()

R = Pentagon()

R.noofsides()

K = Hexagon()

K.noofsides()

3. Create an instance for the child class in child class and call abstract methods

from abc import ABC, abstractmethod

class Animal(ABC):

def move(self):

pass

class Human(Animal):

def move(self):

print("I can walk and run")

class Snake(Animal):

def move(self):

print("I can crawl")

class Dog(Animal):

def move(self):

print("I can bark")

class Lion(Animal):

def move(self):

print("I can roar")

# Driver code

R = Human()

R.move()

K = Snake()

K.move()

R = Dog()

R.move()

K = Lion()

K.move()

4. Create an instance for the child class in child class and call non-abstract methods

import abc

class parent:

def geeks(self):

pass

class child(parent):

def geeks(self):

print("child class")

# Driver code

print( issubclass(child, parent))

print( isinstance(child(), parent))