Assignment-3

Que:1. What are classes and object in python?

Class:

⇒ A class is a user-defined blueprint or prototype from which objects are created.

- classes provide a means of bundling data and functionality together.
- creating a new class creates a new type of object, allowing new instance of that type to be made.
- class instances can also have methods for modifying their state.
-

Syntax:

```
class className:
    # statement
    obj = className()
    print(obj.attr)
```

* Defining a class:

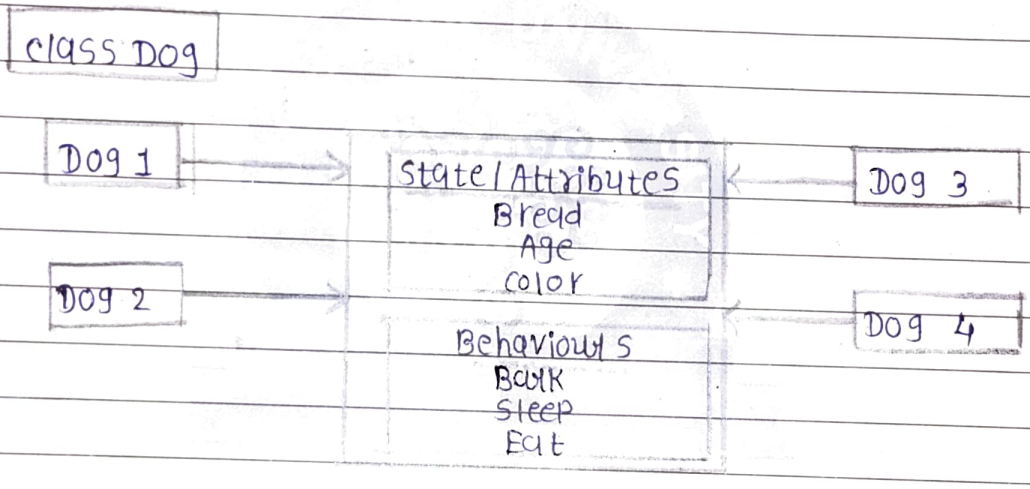
```
# python 3 Program to
# demonstrate defining
# a class

class Dog:
    pass
```

object :

An object is an instance of a class.
- A class is like a blueprint while an instance is a copy of the class with actual values.
- object consist : State
  Behaviour
  Identify

Example:



Que:2. Explain — init — () Function.

⇒ The Default — init — constructor in c++ and java. constructors are used to initializing the object's state.
- The task of constructors is to initialize to the data members of the class when an object of the class is created.
- like methods, a constructor also contain a collection of statements that are executed at the time of

executed at the time of object creation.
- The method is useful to do any initialization you want to do with your object.

Example :

```
class Person:
    # init method or Constructor
    def __init__(self, name):
        self.name = name
    # Sample Method
    def say_hi(self):
        print('Hello, my name is', self.name)
p = Person('Snehal')
p.say_hi()
```

Output :

Hello, my name is Snehal.

Que:3. What are classes and object in python?

⇒ This Answer and 1st Question answer both are same.
So, see and write to answer 1.

Que:4. What is Constructor ? Explain types of constructor with example.

⇒ Constructor are generally used for instantiating an object.

- The task of constructors is to initialize to the data members of the class when an object of the class is created.
- In Python the --init--() method is called the constructor and is always called when an object is created.

★ Syntax :

```
def __init__(self):
```

★ Types of Constructor :

• default Constructor :-
The default constructor is a simple constructor which doesn't accept any arguments.

Ex:

```
class GeekforGeeks:
    def __init__(self):
        self.geek = "GeekforGeeks"
    def print_Geek(self):
        Print(self.geek)
obj = GeekforGeeks()
obj.print_Geek()
```

Output: GeekforGeeks

Parameterized Constructor :-

Constructor with parameters is known as Parameterized constructor.

Ex:

```
Class Addition :
    first = 0
    second = 0
    answer = 0

    def __init__(self, f, s):
        self.first = f
        self.second = s
    def display(self):
        Print("first number = " + str(self.first))
        Print("second number = " + str(self.second))
        Print("Addition of two numbers = " + str(self.answer))
    def calculate(self):
        self.answer = self.first + self.second

obj 1 = Addition(1000, 2000)
obj 2 = Addition(10, 20)
obj 1.calculate()
obj 2.calculate()
bj 1.display()
bj 2.display()
```

Output :

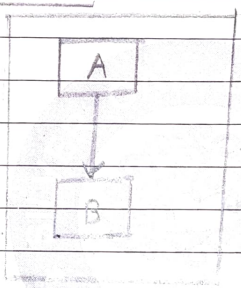
```
first number = 1000
second number = 2000
Addition of two numbers = 3000
first number = 10
second number = 20
Addition of two numbers = 30
```

Que:5. What is inheritance ? Explain types of inheritance.

⇒ Inheritance is defined as the mechanism of inheriting the properties of the base class to the child class.

* Types of Inheritance :

(1) Single inheritance :-

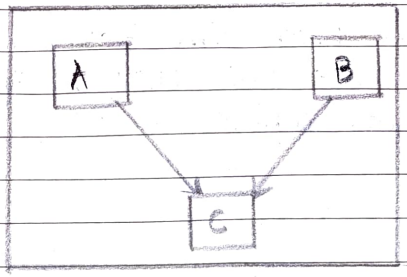


Ex:
```
class parent:
    def Func1 (self):
        print("This function is in Parent class:")
class child (Parent):
    def Func2 (self):
        Print("This function is in child class:)
object = child()
object. Func1()
object. Func2()
```

Output :

This function is in Parent class.
This function is in child class.

(2) Multiple inheritance :

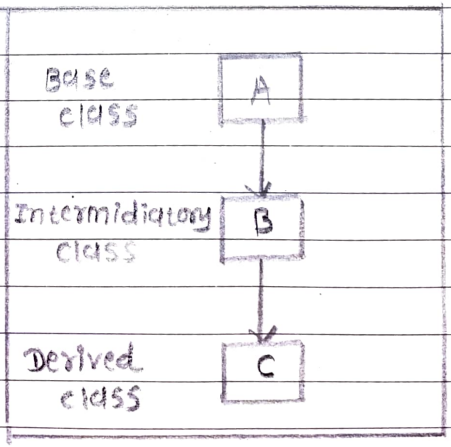


When a class can be derived from more than one base class this type of inheritance is called multiple inheritance.

(3) Multilevel inheritance :

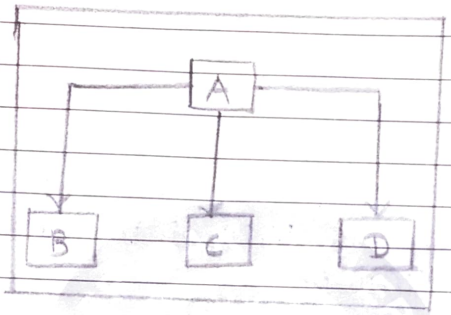
In multilevel inheritance, features of the base class and the derived class are further inherited into the new derived class.

- This is similar to a relationship representing a child and a grandfather.



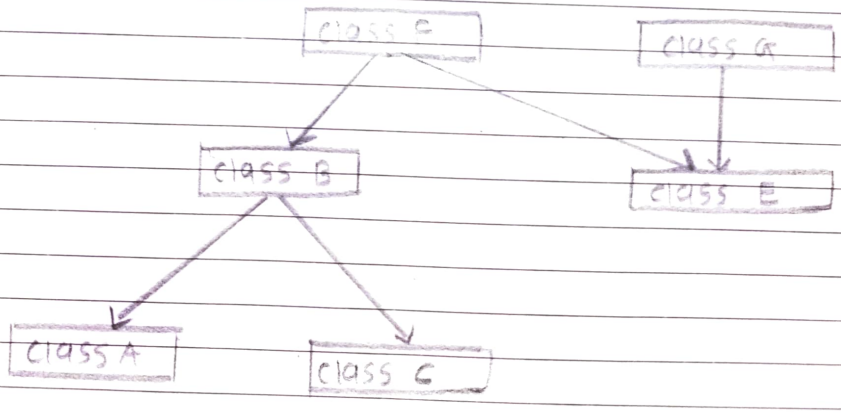
(4) Hierarchical Inheritance :

When more than one derived class are created from a single base this type of inheritance is called hierarchical inheritance.



(5) Hybrid Inheritance :

Inheritance consisting of multiple types of inheritance is called hybrid inheritance.



Que:6. What is the use of super keyword?

⇒ The super() function is used to give access to methods and properties of a parent or sibling class.

- The super() function returns an object that represents the parent class.

* Syntax:

```
super()
```

* Example:

```
class parent:
  def __init__(self, txt):
    self.message = txt

  def printmessage(self):
    print(self.message)
class child(parent):
  def __init__(self, txt):
    super().__init__(txt)

x = child("Hello, and welcome!")
x.printmessage()
```

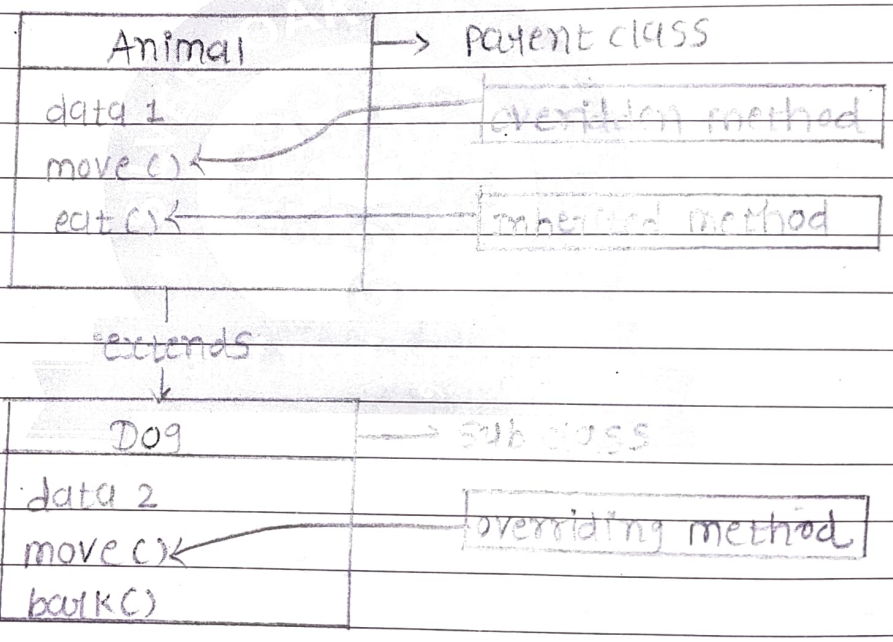
Output:

```
Hello, and welcome!
```

Que:1. What is method overriding ? Explain with example.

⇒ Method overriding is an ability of any OO Programming language that allows a subclass or child class to provide a specific or child class to provide a specific implementation of a method.

- That is already provided by one of its super-classes or parent classes.
- The method in the subclass is said to override the method in the super-class.



Example :

```
class parent():
    def __init__(self):
        self.value = "Inside Parent"
    def show(self):
        Print(self.value)
# Defining child class
class child(parent):
```

```
def --init-- (self):
    self.value = "Inside child"
def show (self):
    print (self.value)
obj 1 = parent()
obj 2 = child()

obj1.show()
obj2.show()
```

Output: Inside parent
Inside child

Que:8. What is self-variable constructor?

⇒ The self is used to represent the instance of the class. With this keyword, you can access the attributes and methods of the class in python.
- It binds the attributes with the given arguments.

★ Python class self constructor:

self is also used to refer to a variable field within the class.
- Here, if we have a variable within a method, self will not work.

Example:

```
class person:

def--init-- (self, John):
  self.name = John
def get-person-name (self):
  return self.name
```

Output: John