OOPS in PYTHON

1]. **CLASS :-** A class is a blueprint for the object. We can think of class as a sketch of a parrot with labels. It contains all the details about the name, colours, size etc.

Example :-

```
Class parrot:
pass
```

Class keyword to define an empty class parrot.

2]. **OBJECT :-** An object (instance) is an instantiation of a class. When class is defined, only description for object is defined, no memory or storage is allocated.

Example :-

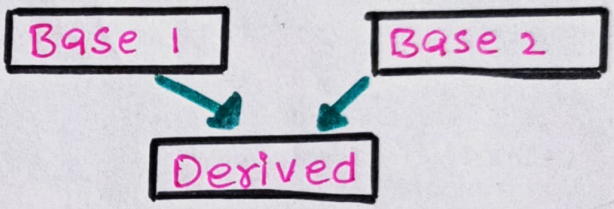
```
Class Vehicle:
def__init__(self, brand, model, type):
Self.brand = brand
Self.model = model
Self.type = type
Self.gas_tank_size = 14
Vehicle_object = Vehicle('Honda', 'truck')
```

3]. **INHERITANCE :-** Inheritance is a way of creating a new class for using details of an existing class without modifying it.

Example:-

```
Class Parent ( ):
def first( Self  ):
Print('first function ')
Class child(Parent  ):
def Second(self ):
Print(' Second function')
Ob = Child ( )
Ob. First (  )
Ob. Second ( )
```

Output:- First function
Second function



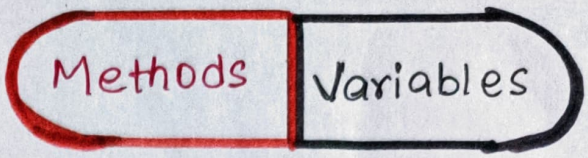
4]. **ENCAPSULATION** :- Using OOP in python, we can restrict access to methods and variable. This prevent data from direct modification which is called as Encapsulation.

Example:-

```
Class Employee:
def__init__(Self, name ,Salary , Project ):
Self.name = name
self.salary = Salary
self. Project = Project
def show ( Self  ):
Print("Name:" , Self.name, 'Salary':, Self Salary)
defwork (self ):
Print(Self.name, ' is working on' , self. Project)
# Creating object of a class.
emp = Employee('Ram ', 10,000, ' Python ')
# Calling Public method.
emp. show ( )
emp. work( )
```

Output:-

Name : Ram Salary : 10,000
Ram is working on python.



5]. **ABSTRACTION :-** Abstraction is used to hide the internal functionality of the function from the users. Abstraction can be achieved by using abstract classes and interfaces.

Example:-

```
From abc import ABC , abstractmethod
class Absclass (ABC):
def Print(self,x):
Print("Passed value:", x)
def task(self):
Print("We are inside Absclass task")
class test_class(Absclass ):
def task(self ):
Print("We are inside test_class task")
# Object of test_class created.
test_obj = test_class()
test_obj.task()
test.obj.Print(100)
```

Output:- We are inside test_class task
Passed value : 100

6]. **POLYMORPHISM :-** The literal meaning of polymorphism is condition & accurance in different forms. Polymorphism means a use of single type entity (method, Operator, or object) to represent different types in different scenarios.

Example:-

```
class Rabbit( ):
def age(self ):
Print("determines age of rabbit ")
def colour(self ):
Print ("determines colour of rabbit")
class Horse ( ):
defage(self):
Print("determines age of horse")
def colour(self ):
Print("determines colour of horse")
obj 1 = Rabbit( )
obj2 = Horse( )
For type in (obj 1, obj 2 ):
type.age( )
type.colour( )
```

Output:-

determines age of rabbit.
determines colour of rabbit.

determines age of horse.
determines colour of horse.