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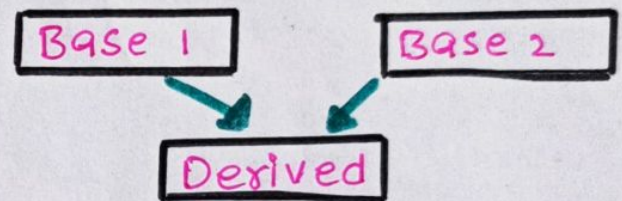
3]. INHERITANCE :-
Inheritance is a way of creating a new class for using details of an existing class without modifying it.

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

ATUL KUMAR (LINKEDIN)
NOTES GALLERY (TELEGRAM)

Base 1   Base 2

Derived

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)
def work (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 ( )
```

2

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

Methods | Variables

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 testclass 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 & accurace in different forms. Polymorphism means a use of single type entity (method, Operator, or object) to represent different types in different scenarios.

Example:-

```python
class Rabbit():
    def age(self):
        Print("determines age of rabbit")
    def colour(self):
        Print("determines colour of rabbit")
class Horse():
    def age(self):
        Print("determines age of horse")
    def colour(self):
        Print("determines colour of horse")
obj1 = Rabbit()
obj2 = Horse()
For type in (obj1, obj2):
    type.age()
    type.colour()
```

Output:-

determines age of rabbit.
determines colour of rabbit.

determines age of horse.
determines colour of horse.

str → Length of string

len() → list → Number of Items

dict → Number of keys

4.