# 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.

ATUL KUMAR (LINKEDIN)
NOTES GALLERY (TELEGRAM)

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 :-

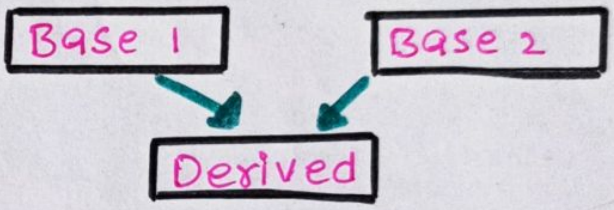
```
Class Vehicle:
def__init__(Self, brand, model, type):
Self. brand = brand
Self. model = model
Self. type = type
Self. gas_tank_size = 14
Vehicle_object = Vehicle('Honda', 'truck')
```

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)

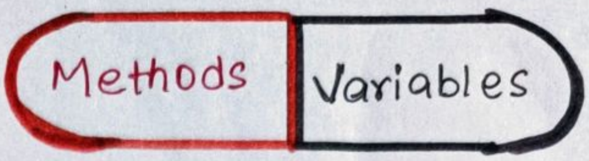
4]. **ENCAPSULATION** :- Using OOP in python, we can restrict access to methods and variable. This prevent data from direct modification which is called as Encapsulation.

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( )
```

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.

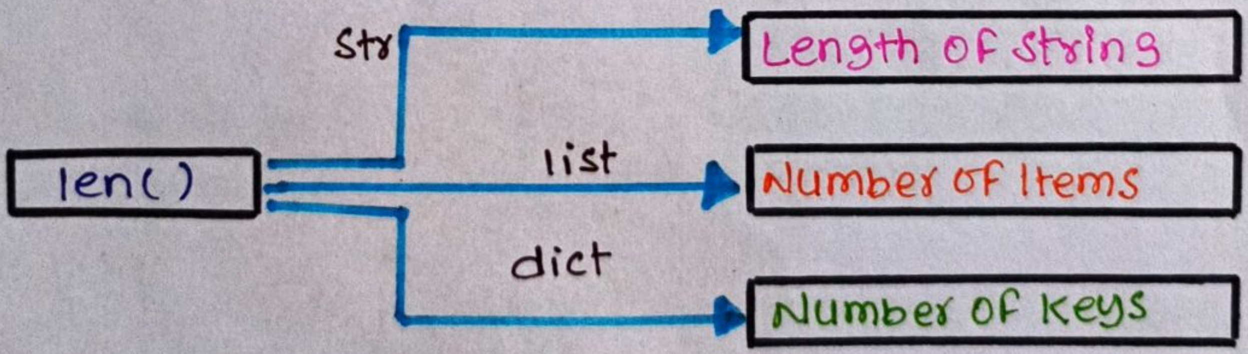
Example:-

```
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.
```



len( )
Str
Length of String
list
Number of Items
dict
Number of Keys