★ Object Oriented Programming .

- In python everything is an object.
  (i.e) list, tuple, dict, ~~etc~~ function, etc.
- We can even create objects of our type using 'class' keyword.

1) Classes.

users defined objects are created using 'class' Keyword.

eg:- class Sample (object):
↓ keyword
↓ name of class u wanna create.

X = Sample ( ) → # instance of class Sample.
X
O/P = main_. Sample.

- We can defined class attributes and methods
- Attributes - it is a characteristic of an object.
- Method - it is an operation we can perform with the object.

2) Attribute :- self. attribute name = something

# syntax ↑

__init__( ) ⇐ # allows you to initialize the attribute.

eg:- (i)

```
class Dog (object):
        def __init__(self, breed):
            self.breed = breed.
```

← # methods they look like function but as they are within the class they are called methods

instance of Dog object type ←

```
sam = Dog (breed = 'Lab')
sam
o/p = <__main__.Dog
```

(ii)

```
sam = Dog()
sam
o/p = error.
```

required 2 arguments & you have passed only 1

here it shows 2 arguements :: self itself is an arguement as here we are dealing with attributes, Thus it is showing 1 arguement is passed & you need to pass another one.

3) **Methods** :- Methods are functions defined inside the body of a class. They are used to perform operations with the attributes of our objects.

- Methods are used in the encapsulation concept of the OOP paradigm.

Eg:- (i) class Circle (object):

```
    #class object attributes
    pi = 3.14            ← # instance of class Circle

    def __init__(self, radius = 1):
        self.radius = radius   → # attribute of class Circle

c = Circle(radius)
c.radius
O/P = 1
c.pi
O/P = 3.14.
```

(ii) class Circle (object):

```
    #class object attributes
    pi = 3.14

# 1st method of class Circle to get radius ← { def __init__(self, radius):
                                                   self.radius = radius

# 2nd method of class Circle to find area. ← { def area(self):
                                                   # radius**2 * pi
                                                   return (self.radius**2) * Circle.p

c = Circle(radius = 100)
c.area()
O/P = 31400.0.
```