Python is an ***object-oriented programming (OOP) language***.

***Everything*** is in Python ***treated as an object***.

Every object belongs to its class.

Including variable, function, list, tuple, dictionary, set, etc.

An object is a **real-life entity**.

An object is an **instance of a class**.

An object is the collection of data/ attribute and functions/ methods.

A class is a **collection of objects**.

A class is a **blueprint/ template** through which objects are created.

A class is like an **object constructor**.

\_\_init\_\_() Method

All classes have a function called \_\_init\_\_(), which is always executed when the class is being initiated.

The \_\_init\_\_() function is called automatically every time the class is being used to create a new object.

self parameter

The self-parameter is a **reference to the current instance of the class** and is used to access variables that belong to the class.

It does not have to be named self, you can call it whatever you like, but it must be the first parameter of any function in the class

* Object.\_\_dict\_\_ => to display all the attributes of an object. (Returns a dictionary of attribute)
* hasattr(object, attribute) => to check object has an attribute or not (return Boolean value)
* getattr(object, attribute) => return the value of the attribute of that object
* getattr(object, attribute, third argument) => the function return third argument as default value when the attribute is not present for that object
* delattr(object, attribute) => to delete an attribute of an object

class attribute => common attribute for all the objects such as organization name

instance/ object attribute => unique attribute of an object such as student name and roll number

if an object has instance attribute & class attribute as same so the attribute which going to print is instance attribute if there is no instance attribute then class attribute is going to print

Internally how methods are called in python

ClassName.Function(ObjectName) is same as ObjectName.Function()

self = Object

Object1.Function(object2) = Function(self, arg) = self => object1 & arg => object2

All class methods by default want to pass self as a parameter

Static methods

Static method is also called as class method because it not related to object.

Static method do not take self as a parameter.

Need to pass **@staticmethod** **decorator** to stop binding of the method to the object.

They are utility functions need to check something in the class.

Factory Method

Used to return object of a class.

To make Private variable & Protected variable

Use \_\_VariableName to make it private & Use \_VariableName to make it protected.

Public variables are available everywhere.

Private variables are not access outside the class.

Protected variables are available in class and inherited sub class.

In python public is same as protected that doesn’t mean we should use protected to modify the protected variable outside class.

If they are same, why different type because pythons think programmer are sensible. If you are modifying, it outside class you can but you shouldn’t do it.

Polymorphism

Ability to take multiple forms.

**Method Overriding**

In Method overriding parent and child class have same named functions within inheritance

It calls the function form child class, if the child class doesn’t have the function then it go for parent class.

Meta Class

A metaclass in Python is a class of a class that defines how a class behaves. A class is itself an instance of a metaclass.

Object Class

Every class inherits from object class.

Every object class have three methods.

1. \_\_new\_\_ : Used to create a new object. We do not override it.
2. \_\_init\_\_: Used to initialize an object attribute. Generally, we override the init method
3. \_\_str\_\_: by default, it provides the location of class where it store but we can override it and provide a meaningful description.

mro() = Method resolution order

Abstract Class

Abstract classes contain abstract methods these methods are declared but they have no implementation, and they are compulsory implemented by child class.

Use module abc and abstractmethod decorator

Object of abstract class cannot be created

Need to implement all the abstract methods in the child class