# OOPS

In Python, object-oriented Programming (OOPs) uses objects and classes in programming.

It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.

The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.

***Main concepts in OOPS***

* Class
* Objects
* Polymorphism
* Encapsulation
* Inheritance
* Data abstraction

## Class

A class is a collection of objects. A class contains the blueprints or the prototype from which the objects are being created. It is a logical entity that contains some attributes and methods.

Syntax:

class Classname:

statement 1

:

:

Statement n

### ****Self:****

1. Class methods must have an extra first parameter in the method definition. We do not give a value for this parameter when we call the method, Python provides it
2. If we have a method that takes no arguments, then we still have to have one argument.

### ****The \_\_init\_\_ constructor/method****

It is run as soon as an object of a class is instantiated. The method is useful to do any initialization you want to do with your object.

**What is an Object ?**

Everything is in Python treated as an object, including variable, function, list, tuple, dictionary, set, etc.

*Every object belongs to its class.*

For example - An integer variable belongs to integer class. An object is a real-life entity.

# Inheritance

To acquire the properties from one class to another we use Inheritance

It provides code reusability.

Inheritance allows **us to define a class that inherits all the methods and properties from another class**.

# *Types Of Inheritance:*

* Single Level Inheritance
* Multi level Inheritance
* Multiple Inheritance
* Hierarchical Inheritance
* Hybrid inheritance

# *Single level Inheritance:*

We have 2 classes in this: Base class, Derived class

When Derived class is inherited from Base class, It can use all the properties and methods from Base class

# *Multi level Inheritance:*

We have more than 2 classes here

It’s like Grand parent, parent, child relationship. Middle class is inherited from the base class and if the child class inherits the middle class, it gets all the properties & methods from 2 classes

# *Multi level Inheritance:*

Where one class inherits properties from more than 1 class at a time.

Class A(B,C)

Where B & C are 2 different classes and have no relation to each other

# Super keyword:

After creating an object for child class, using that we can access parent class methods

If a child class doesn’t have “init ” constructor that objects directly insanitize the parent class “init” constructor

But if a child class has an “init “ constructor we have to give super().\_\_init\_\_() in child “init” constructor

Then when we create an object for child class, it automatically insanitize the child class & parent class init constructor

# MRO Principle: (Method Resolution Order)

MRO is a concept used in inheritance. It is the order in which a method is searched for in a classes hierarchy and is especially useful in Python because Python supports multiple inheritance. In this case, the MRO would be **C -> B -> A**.

Since we have multiple Inheritance MRO is introduced