**Object Oriented Programming in Python**

* **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:**

**#statments**

## Objects

**The object is an entity that has a state and behavior associated with it. It may be any real-world object like a mouse, keyboard, chair, table, pen, etc. Integers, strings, floating-point numbers, even arrays, and dictionaries, are all objects.**

**Syntax**

**Object\_name=Classname()**

**An object consists of :**

* **State: It is represented by the attributes of an object. It also reflects the properties of an object.**
* **Behavior: It is represented by the methods of an object. It also reflects the response of an object to other objects.**
* **Identity: It gives a unique name to an object and enables one object to interact with other objects.**

**\_\_init\_\_() - it is a constructor which is automatically called when you create object.**

**Used for assigning attribute to object.**

**Self- every method in class has default argument which is self. It is used for each new object reference**

**If we have a method that takes no arguments, then we still have to have one argument.**

**Method- Function inside class is called as method**

**How to call method?**

**Obj\_name.method\_name()**

**instance variables- these attributes defined in constructor. They are not common to all object. For different object different attribute values we have to provide.**

**Class Attribute- These attributes defined outside constructor and inside class.**

**Class attributes does not require self.**

**Class attributes can be accessed by both class name and object name.**

**Types of Methods**

**Class method – used for dealing with class attribute**

**Class method called using both class name and object**

**Static method- It is just like a normal function.**

**Does not deal with class attribute or instance attribute.**

**Instance Methods- used for dealing with instance attribute**

**Setters, getters**

**Encapsulation:**

**Encapsulation puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data**

**Access scpecifier public private protected**

**Polymorphism: Many forms of one entity**

**Following are the types of Polymorphism**

**1 Method Overriding (run time)**

**2 Method overloading (compile time)**

**3 operator overloading (run time)**

**1 Method Overriding**

**An ability of class to change behavior of base class method in derived class**

**Have to define method with same name in derived class**

**2 Method overloading**

**Means same function name and different parameter**

**Python does not support this feature but we can demonstrate using dispatch decorator**

**3 operator overloading:**

**It is an ability to change existing logic of magic methods**

**In python for every operator magic methods are defined**

**Print(1+2) plus operator will call magic method \_\_add\_\_ in background**

**e.g. int.\_\_add\_\_(1,2)**

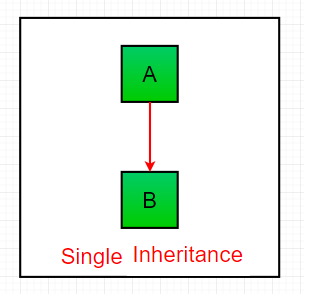
**Inheritance- used for accessing properties of base class into derived class**

**Base class/parent class/super class-from where properties (attribuites+methods) are derived**

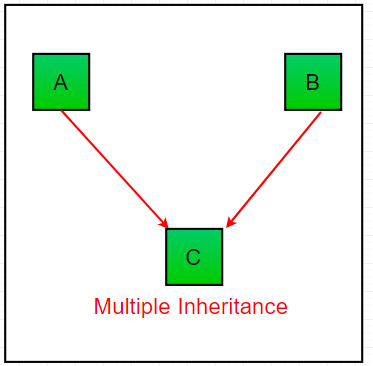
**Derived class/child class/sub class- where we can use properties of base class**

**Types of Inheritance:**

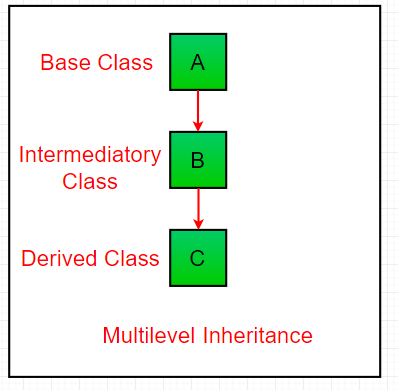
**Single Inheritance- Single-level inheritance enables a derived class to inherit characteristics from a single-parent class.**

****

**Multiple Inheritance- Multiple inheritance enables one derived class to inherit properties from more than one base class.**

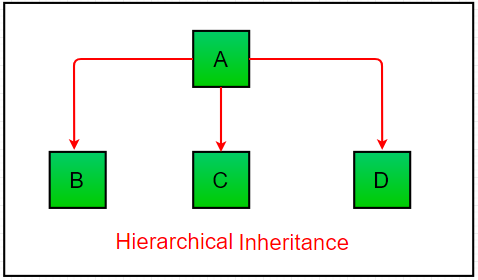
****

**Multilevel Inheritance- Multi-level inheritance enables a derived class to inherit properties from an immediate parent class which in turn inherits properties from his parent class.**

****

**Hierarchical Inheritance-**

**Hierarchical level inheritance enables more than one derived class to inherit properties from a parent class.**

****

**Hybrid Inheritance- Combination of two or more inheritance.**

## ****Data Abstraction****

**It hides the unnecessary code details from the user. Also,  when we do not want to give out sensitive parts of our code implementation and this is where data abstraction came.**

**e.g. hiding implementation of ATM machine.**