Intro

- class --> Blueprint of an object
- Object ---> Real world entity/Instance of a class
- self :-
  - It represents the instance of a class
  - By using self keyword we can access variables and methods of class
  - Used to create an instance variable
  - self holds reference to the instance itself
  - helps to distinguish between local and instance variables
  - self is just a keyword, we can use any name
- init :-
  - Automatically get called/executed when we are creating a new object of a class

In [1]:

```python
class ClassName:                        # Class Name
    class_var = "Class Variable"        # Class Variable
    def __init__(self,a,b):
        print("Class __init__ Method")
        self.a = a
        self.b = b                      # Instance variable
        self.company_name = "TCS"
        area = "Pune"                   # Local

    def method1(self):                  # Class Method
        print("This is Method 1")

Object = ClassName(1,5) # Creating an object of a class
```

```
Class __init__ Method
```

In [2]:

```python
Object.class_var
```

```
'Class Variable'
```

In [3]:

```python
print(Object.a)
print(Object.b)
print(Object.company_name)
```

```
1
5
TCS
```

In [4]:

```
Object.method1()
```

```
This is Method 1
```

Inheritance

- It allowes a class to inherit the properties (variables and methods) of other class(parent/base class)
- Resuable and Readable
- Types of Inheritance :-
  - 1. Single Inh - (Parent > Child)
  - 2. Multiple Inh - (Father & Mother > Child)
  - 3. Multilevel - (GrandParent > Parent > Child)
  - 4. Hirarchical Inh - (Parent > Child1,Child2,..ChildN)
  - 5. Hybrid Inh -(More than one Inh/Combinational Inh)

Encapsulation :-

- Used to restrict the aceess of the variable and methods from outside the class
- Help us to prevent an accidental change of data from outside the class
- Public Variable and Public Method :-
  - Can be accessed / Modeified from outside the class
- Private Variable & Method :-
  - Can not be accessed / Modeified from outside the class
  - __VarName
  - __MethodName
- Name Mangling :-
  - Used to access / modify private variable and method from outside the class
  - ObjectName._ClassName__VarName/__MethodName

Polymorphism :-

- Having Many Forms
- Same method names with different functionalities in the different classes
- len(),min(),max(),sorted(),sum().......
- Overloading Method :-
    - Same Method in Parent and Child class
    - Child class Method Overrried Parent class Method

Abstraction

- Blueprint of your project/Other class
    - We can not creare an object of the abstract class
    - Abstract class is only used for declaration of the methods
    - In the abstract class we are not supposed to implement those methods
    - @abstractmethod decorator is used to define an abstract method
    - The abstract class is a child class of ABC(Import abc)
    - Help us to hide internal frunctionality of the function

```
In [ ]:
```