In [ ]:
```
#Agenda of the day:

         1. OOPS in Python.
         (Object Oriented Programming)

Python Supports both Procedure Oriented and Object Oriented Programming

- Procedure Oriented----> Every Operation Done By Creating Functions
- Object Oriented ---->   Every Operation Done By Creating Object &
                                                    Also Classes.
#Introduction:
 --> In 1960s OOps(Simula) was initiated by alan Kay.
---> with help of C++ which was available in software market.
---> Furtherly It was adopted by many programming Languages.
        Those are
               --- C++
               --- Java
               --- Java Script
               --- Python

#Where we can use oops mostly?
   1. Real-Time Systems
   2. Artificial Intelligence
   3. Expert Systems
   4. Cilent-Server Projects
   5. Object -Oriented Databases...etc
```

In [ ]:
```
#What is Object Oriented Program?
---Its a Different way of Structuring a software program by bundling the
  properties/attributes and behaviours/actions into individual objects.
      or
    Its deals with classes and Objects.
#Keypoint:
    It used to Structure a program into simple,resuable,pieces of code.
```

In [ ]:
```
#Contents of OOPs:
1. Class
2. Objects
3. Method and Construtor Method (class variables and Instance variables)
4. Inherirance
         - Single Level
         - Multi Level
         - Hierarchical
         - Multiple
5. Polymorphsim -(Method Overiding)
6. Data Abstraction
7. Data Encapusulation & Data Hiding.
```

In [ ]:
```
#Class: What is a class?

   - A class is a Collection of Objects.
      or
```

- A class is blueprint of the object and object must follows that blueprint logic or rules.
- A class is a Logical Entity...

In [ ]:
```
#How to create class?
#syntax:
class class_name:
    #class variables             -Data Members
    #class methods
    #construtor method
```

In [ ]:
```
#What is Object? (Its a Physical Entity)
An Object (Instance) of a class that follws the class logic.
```

In [ ]:
```
#How to Create Object?
class CAR:
    #Variables       ---> Class Variables
    #Methods         ----> Class Methods

objectname= classname()    ---> Object Creation for class
objectname.variablename    ---> Accessing class variable by object calling
objectname.methodname()     ---> Accessing class method by object  calling.
```

In [18]:
```
#Example: (class,Object,,construtor method, class methods,class variables
# instance variables,object creation,accessing the class datamembers)
class Myclass:
    sums=0                 #class variables
    def __init__(self,a,b,c):   #contructor method
        self.sums = a+b+c   #self keyword is used to access of members of class
        print("Construtor is invoked")    #a,b - instance variables
    def printSum(self,a,b,c):      #class method

        return self.sums
        #print("sum of a and b is :",self.sums)

a = int(input("enter a value"))
b = int(input("enter b value"))
c = int(input("enter c value"))
obj = Myclass(a,b,c)    #object creation--then constuctor method will invoke
                         #accessing values of class variables

print(obj.printSum(a,b,c))        #accessing the class method definition
```

```
enter a value100
enter b value500
enter c value400
Construtor is invoked
1000
```

In [ ]:
```
#Inheritance:
     what is the Inheritance?
        Parent class------------> Child class
            from        Properties  to
                      (accuring)
```

A class (Derived or child class) which inherits or accuring the properties of another class(base or parent class) is called Inheritance.

#Note : By Using Inheritance we make code resuablity.

In [38]:
```
#Single_Level Inheritance( 1 Base class,1 Child Class)
class arthematic:           #base class
    a = 100
    b =  300
    def add(self):
        total = self.a+self.b
        print("sum of the a nd b is:",total)
class subtraction(arthematic):   #inheritance applied       #child class
    c = 500
    d = 600
    def sub(self):
        subs = self.d-self.c
        print("substraction of c and d is:",subs)
pb = arthematic()
cb = subtraction()
print(cb.a)
print(cb.b)          #accessing base class variables and methods
cb.add()
print(cb.c)
print(cb.d)       #accessing self(child) class variables and methods
cb.sub()
```

```
100
300
sum of the a nd b is: 400
500
600
substraction of c and d is: 100
```

In [ ]:
```
#Multi-Level Inheritance:
Parent1class-------> Parent2/child1class---->Child2class
```

In [ ]:
```
#Example: one or more parents
class addition:
    c = 90
    d =50
    def __init__(self,a,b):
        self.a = a
        self.b = b
    def add(self,a,b):
        total = self.a+self.b
        return total
class substraction(addition):   #level-1 inheritance
    def sub(self,a,b):
        subs = self.b-self.a
        return subs

class multiplication(substraction):   #level-inheritance
    def mul(self,a,b):
        multi = self.a * self.b
        return multi
```

```
a = int(input("enter a value"))
b = int(input("enter b value"))
```