Note

In [ ]:
```
In object oriented Programming :

    Properties --> variables/Attributes
    Behaviour  --> Functions/Methods
```

Types of Attributes/variables

In [ ]:
```
Types of Attributes/variables:
    1.Instance Varibles --> Object Level Variables
    2.Static Variables --> Class Level Variable
    3.Local Variables
```

Instance Variable

In [ ]:
```
Instance Variable --> if the value of the variable is varied from object to objectsuch type of variables are known as
                      instance variable .
                  --> for every object a seperate copy of variable is created
                  --> For declaring the instance variable you need to use  self
                  --> If you want to access any instance variable inside the class then you need to use self variable
                  --> If you want to access any instance variable outside the class then you need to use object reference
```

Example of Instance Variable

In [2]:
```
class Student:
    def __init__(self,name,classes,rollno):
        self.name=name   #"Pratyush"      #Pavan
        self.classes=classes  #12             #99
        self.rollno=rollno   #99          #100
        print("Student name is :",self.name)
        print("Student rollno is :",self.rollno)
        print("Student marks is :",self.classes)

    def talk(self):
        print("Student name is :",self.name)
        print("Student rollno is :",self.rollno)
        print("Student marks is :",self.classes)
x=Student("Pratyush",12,99)
print(x.name)
print("-------------")
y=Student("Pavan",99,100)
print(y.name)
```

```
Student name is : Pratyush
Student rollno is : 99
Student marks is : 12
Pratyush
-------------
Student name is : Pavan
Student rollno is : 100
Student marks is : 99
Pavan
```

In [3]:
```
class Mobile:
    def __init__(self,Brand,Processor,Color):
        self.x=Brand
        self.y=Processor
        self.c=Color

    def view_mobile(self):
        print("Mobile Brand is,",self.x)
        print("Mobile Processor is,",self.y)
        print("Mobile Color is,",self.c)
x = Mobile("Vivo","860 Snapdragon","White")

y = Mobile("Samsung","870 Snapdragon","Black")
y.view_mobile()
```

```
Mobile Brand is, Samsung
Mobile Processor is, 870 Snapdragon
Mobile Color is, Black
```

Static Variable

In [ ]:
```
static variable --> if the value of the variable is not varying with object to object then such type of variables are known
                    as static variable
                --> a single copy is created for whole class
                --> Static variable are declared inside the class outside the constructor
                --> If you want to access static variable inside the class then you can use either self or classname
                --> if you want to access static variable outside the class then you can use eithter object reference
                    or classname. but it is recoommnded to use class name
                --> Static variable are the only variable that we can access without creating the object
```

Example of Static Variable

In [4]:
```
class Student:
    college_name="Edyoda Digital University"
    def __init__(self,name,classes,rollno):
        self.name=name
        self.classes=classes
        self.rollno=rollno
        print("Student name is :",self.name)
        print("Student rollno is :",self.rollno)
        print("Student marks is :",self.classes)

    def talk(self):
        print("Student name is :",self.name)
        print("Student rollno is :",self.rollno)
        print("Student marks is :",self.classes)
        print("Student College Name is ,",self.college_name)
        print("Student College Name is ",Student.college_name)  #recommended
x=Student("Pratyush",12,99)
Student.college_name
print(x.college_name)
```

```
Student name is : Pratyush
Student rollno is : 99
Student marks is : 12
Edyoda Digital University
```

In [8]:
```
class Student:
    college_name="Edyoda Digital University"
    def __init__(self,name,classes,rollno):
        self.name=name
        self.classes=classes
        self.rollno=rollno
        print("Student name is :",self.name)
        print("Student rollno is :",self.rollno)
        print("Student marks is :",self.classes)

    def talk(self):
        print("Student name is :",self.name)
        print("Student rollno is :",self.rollno)
        print("Student marks is :",self.classes)
        print("Student College Name is ,",self.college_name)
        print("Student College Name is ",Student.college_name)  #recommended
s=Student("Arun","99","101")
print(s.college_name)
print(Student.college_name)
```

```
Student name is : Arun
Student rollno is : 101
Student marks is : 99
Edyoda Digital University
Edyoda Digital University
```

Local Variables

In [ ]:
```
local variables --> these variable are used to declare inside the method for the temperory requirement.
                --> Local variable are created at the time of function exectuion and once
                    function execution is done it is destroyed
                --> Local variable cannot be accessed outside the function
```

Example of Local Variables:

In [10]:
```
class test:
    def m1(self):
        a=100
        print(a)
    def m2(self):
        b=200
        print(b)
t=test()
t.m1()
t.m2()
```

```
100
200
```

Types of Methods

In [ ]:
```
Types of Methods
1.Instance method
2.Static method
3.Class method
```

Instance Methods

In [ ]:
```
Instance Methods(Object Level Method) --> if you are using instance variable inside any method then
                                          overall method will be an instance method.
                                      --> The first argument of the instance method will be always be self.
                                      --> Instance method are generally used to access instance variable.
                                      --> You can call instance method with the help of object reference outside the class
                                      --> if You want to call instance method within the class then you need to use self
                                          variable
```

Example of Instance Method

In [11]:
```
class Rectangle:
    def __init__(self,length,breadth):
        self.length=length
        self.breadth=breadth

    def Perimeter(self):
        return 2*(self.length+self.breadth)


    def Area(self):
        return self.length*self.breadth


x = Rectangle(100,200)
x.Perimeter()
x.Area()
```

Out[11]:
```
20000
```

In [12]:
```
class Circle:
    pi=3.14
    def __init__(self,radius):
        self.radius=radius

    def Perimeter(self):
        return 2*self.pi*self.radius

    def Area(self):
        return 2*self.pi*self.radius**2

c=Circle(10)
print(c.Perimeter())
print(c.Area())
c1=Circle(20)
print(c1.Area())
print(c1.Perimeter())
```

```
62.800000000000004
628.0
2512.0
125.60000000000001
```

In [13]:
```
class Circle:
    pi=3.14
    def __init__(self,radius):
        self.radius=radius

    def Perimeter(self):
        return 2*self.pi*self.radius

    def Area(self):
        return self.Perimeter() +200

c=Circle(10)
print(c.Perimeter())
print(c.Area())
```

```
62.800000000000004
262.8
```

Class Method

In [ ]:
```
Class Method --> if you are using static variable(class level) inside any method then overall
                 method will be a class method
             --> If you want a define a class method then you need to use @classmethod decorator
             --> if you are defining class method then you need to pass atleast one argument(cls) that is mandatory
             --> you can directly access class method without creating an object
             --> If you want to access the static variable inside the class method then you need to use cls variable
             --> If you want to access the static variable inside the instance method then you need to use class name
             --> You can call Class Method without Creating an Object.
```

Example of Class Method

In [14]:
```
class Animal:
    legs=4
    @classmethod
    def walk(cls,name):
        print(str(name)+" having "+str(Animal.legs)+" legs")

Animal.walk("lion")
dir(Animal)
```

```
lion having 4 legs
```

Out[14]:
```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 'legs',
 'walk']
```

Static Method

In [ ]:
```
Static method --> General Utility methods
              --> Inside these method we cannot use any instance or class variable
              --> We can declare static method by using @staticmethod decorator
              --> You can access static method with the help of class name or object reference
```

Example of Static Method

In [48]:
```
class Math:
    @staticmethod
    def add(x,y):
        return x+y
    @staticmethod
    def sub(x,y):
        return x-y
x=Math()
print(x.add(10,20))
print(Math.add(20,30))
```

```
30
50
```

Note

In [ ]:
```
--> In general programming we are only using instance and class methods
--> Static method are not generally used

     Instance method  --> 80%  --> object level methods
     Class Method ---> 15% --> class level methods
     Static method  --> 5%---> general utility method
```

Access Modifiers

In [ ]:
```
Access modifiers --> are keywords in object-oriented languages that set the accessibility of
                     classes, methods, and other members.
                 --> Access modifiers are a specific part of programming language syntax used to facilitate the encapsulation
                     of components.
```

Types of Access Modifiers

In [ ]:
```
Access Modifiers: are used to provide privacy to our data:
        1.Public
        2.Private
        3.Protected

Example:
    Youtube Public Videos
    LInkedin Profiles Without Privacy
    Road

Private:
    Facebook --> Profile Lock
    Instagram --> Private  lock


Protected means --> You will access and  your child classs
```

Public Access Modifiers

In [ ]:
```
Public Access Modifiers --> by default each attributes/variable is public in python
                        --> We can access public access modifier variables/attributes either within the class or outside the
                            class
Syntax for Creating public Variables:
    name="Anil"
```

Private Access Modifiers

In [ ]:
```
Private Access Modifiers --> can be accessed within the class.(from outside the class you cannot access)

Syntax for Creating private Variables:
    __name="Sujita"
```

Protected Access Modifiers

In [ ]:
```
Protected Access Modifiers --> can be accessed within the class and outisde the
                               class only in child class. We can specify protected access modifier with the help of single
                               underscore.

Syntax for Creating protected Variables:
      _name="Sagar"
```

Example of Each Type of Access Modifiers

In [15]:
```
class Test:
    x=10     #public
    __y=20    #private
    _z=300    #protected

    def  m1(self):
        print(Test.x)
        print(Test.__y)
        print(Test._z)

a=Test()
#a.m1()
a.x
#a.__y
a._z
```

Out[15]:
```
300
```