Example:
```
class Human():
    life="Air & Water"
    #Instance Attributes
    def __init__(self,name,height,weight):
        self.name=name
        self.height=height
        self.weight=weight
    #Instance Method
    def Eating(self,food):
        return "{} is eating {}".format(self.name,food)
#Creating object of human class
Ram=Human('Ram',6,70)
Scott=Human('Scott',6,76)
print(Human.life)#Accessing class level variable

print("Height of {} is {}".format(Ram.name,Ram.height))
print("Height of {} is {}".format(Ram.name,Ram.weight))
print(Ram.Eating("Pizza"))

print("Height of {} is {}".format(Scott.name,Scott.height))
print("Height of {} is {}".format(Scott.name,Scott.weight))
print(Ram.Eating("Big Burger"))
```

Example:
```
class Basket():
    def __init__(self):
        self.basket=[]

    #Creating Instance Methods
    def Fill(self,fruit):
        self.basket.append(fruit)

    def Delete(self,fruit):
        if fruit in self.basket:
            self.basket.remove(fruit)

    def Show(self):
        for fruit  in self.basket:
            print(fruit)
Obj=Basket()
Obj.Fill("Apple")
Obj.Fill("Grapes")
Obj.Fill("Papaya")
Obj.Show()
print()
Obj.Delete("Apple")
Obj.Show()
```

Example:
```
class MyBanking():
    def __init__(self):
        print("Hello I am Always First Get Executed")
    def Display(self):
        print("Welcome to Display Method")
        print("Good Bye")
```

```
#Creating Object or Instance
MM=MyBanking()

Example:
class MyBanking():
    def __init__(self):
        print("Hello I am Always First Get Executed")
    def Display(self):
        print("Welcome to Display Method")
        print("Good Bye")

MM=MyBanking()
MM.Display()

Example:
class person:
    def __init__(self,name):
        self.name=name
    def display(self):
        print("Hello",self.name)
        return
MyObj=person('Raju')
MyObj.display()

Example:
class person:
    def __init__(self,name):
        self.name=name
    def display(self):
        print("Hello",self.name)
        return
person('Raju').display()

Example:
class Employee():
    def __init__(self,name,eid,loc):
        self.name=name
        self.eid=eid
        self.loc=loc
    def eDetails(self):
        print("Hello Name is:",self.name)
        print("Employee ID is:",self.eid)
        print("Employee Location is:",self.loc)
Emp=Employee("KSRaju",1139,"HYD")
Emp.eDetails()

Example:
class Employee():
   def eDetails(self):
        print("Hello Name is:",self.name)
        print("Employee ID is:",self.eid)
        print("Employee Location is:",self.loc)
Emp=Employee()
Emp.name='Raju'
Emp.eid=101
Emp.loc="HYD"
Emp.eDetails()
```

Example:

```
class Employee():
    def Emp(self,Name,Id,Loc):
        self.Name=Name
        self.Id=Id
        self.Loc=Loc

    def eDetails(self):
        print("Hello Name is:",self.Name)
        print("Employee ID is:",self.Id)
        print("Employee Location is:",self.Loc)

EE=Employee()
EE.Emp("Raju",1001,'Hyd')
EE.eDetails()
```

Example:

```
class Test():
    def __init__(self):
        print("Hei I am a Constructor.!")
    def Method(self):
        print("Hello I am a Method.!")
Tst_One=Test()
Tst_Two=Test()
Tst_Three=Test()
Tst_One.Method()
```

Example:

```
class Me():
    def __init__(self):
        print("Hei Constructor")
        print(id(self))
MM=Me()
MM.__init__()
MM.__init__()
```

Find the number of references of an object:

```
sys.getrefcount(objectreference)
```

EXAMPLE:

```
import sys
class Bank():
    pass
BB=Bank()
BB1=BB
BB2=BB1
print(sys.getrefcount(BB))
```

Note:
For Every object, PYTHON internally maintains one default reference variable 'self'

Python Default Constructor
When we do not include the constructor in the class or forget to declare it, then that becomes the default constructor. It does not perform any task but initializes the objects.

Example:

```
class Student:
    roll_num = 101
    name = "Joseph"
    def display(self):
        print(self.roll_num,self.name)

st = Student()
st.display()
```

Differences between Methods and Constructors:

Method

1. Method can be any name
2. It will be executed if we call that method
3. For an object, method can be called any number of times.
4. Method contains business logic

Constructor

1. Name should be always __init__
2. It will be executed automatically at the time of object creation.
3. For object, Constructor will be executed one-time
4. Constructor Inside we have to declare, initialize instance variables

In Python class we can represent data by using the following variables.

1.Instance Variables (Object Level Variables)
2.Static Variables (Class Level Variables)
3.Local variables (Method Level Variables)

Instance Variables:
If the value of a variable is varied from object to object, for every object a separate copy of instance variables will be created.

Where we can declare Instance Variables:

1 Inside Constructor by using self variable
2 Inside Instance Method by using self variable
3 Outside of the class by using object reference variable

Example:

```
class Emp():
    def __init__(self):
        self.eid=1
        self.ename='Rama'
        self.esal="$1000"

eDetails=Emp()
print(eDetails.__dict__)
```

Example:

```
class MyClass():
    def __init__(self):
        self.x=100

    def MyMethod(self):
        self.y=200
```

```
MM=MyClass()
MM.MyMethod()
print(MM.__dict__)

Example:
class MyClass():
#This is constructor
    def __init__(self):
        #Instance Variable with self
        self.x=100

#Instance Method
    def MyMethod(self):
        #Instance Variable with self
        self.y=200

MM=MyClass()
MM.MyMethod()
#Instance Variable with Object Reference
MM.z=300
print(MM.__dict__)

Example:
class MyClass():
    def __init__(self):
        self.x=100

    def MyMethod(self):
        self.x=200
        self.y=300

MM=MyClass()
MM.MyMethod()
MM.y=400
MM.z=500
print(MM.__dict__)

Example: Deleting Instance Variabes
class MyClass():
    def __init__(self):
        self.x=100

    def MyMethod(self):
        self.x=200
        self.y=300
        #Deleting Instance variables
        del self.x

MM=MyClass()
MM.MyMethod()
MM.y=400
#Deleting Instance variables
del MM.y
MM.z=500
print(MM.__dict__)
```

Example:

```
class MyClass():
    def __init__(self):
        self.x=100
        self.y=200

MM1=MyClass()
MM1.x=300
MM2=MyClass()
MM2.y=400
print(MM1.x,MM1.y)
print(MM2.x,MM2.y)
```

Static Variables:
The value of a variable is constant for every object
Class having static variable, shared by all objects of that class.
Access static variables by class name or object reference, recommended to use class name.

Instance Variable vs Static Variable:
In the case of instance variables for every object a seperate copy created
in the case of static variables for total class only one copy created

Example:

```
class MyClass():
#Static Variable
    x=1
    def __init__(self):
        self.y=2

MM1=MyClass()
MM1.y=333
MyClass.x=666
MM2=MyClass()
print(MM1.x,MM1.y)
print(MM2.x,MM2.y)
```

Example:

```
class MyClass():
    x=1
    def __init__(self):
        self.y=2

MM1=MyClass()
MM1.y=333
MM1.x=666
MM2=MyClass()
print(MM1.x,MM1.y)
print(MM2.x,MM2.y)
```

NOTE:
By object reference we can access instance variables
By class name we can access static variables