**Object Oriented Programming System(OOP’s)**

For introducing real word entities, in our programming world we need object oriented concept. In object oriented concept, we are having so many important terminologies like,

1. class
2. object

**class :** class may be define as a blueprint of an object, in which we are defining object properties and action/behaviors. Hear, properties can be represented by variables and action or behavior can be represented by methods.(Class may be define as a blue-print that contains attributes like variables and methods).

**Syntax** for defining any class:

**Class class\_name**:

“doc string”

**Contractors-**

**Variables-**

Instance variable

Static variable

Local variable

**Methods-**

Instance method

Static method

Class method

**Object :** instance of a class is known as object.

**Properties of oops concept:**

1. abstraction (Data-security)
2. encapsulations (Data-security)
3. inheritance (Code-Reusability)
4. polymorphism (Code-Reusability)

**What are Constructors** :-- In any programming language, a constructor is a special method that is automatically invoked whenever an instance (object) of a class is created. There is no need to explicitly call it. Typically, the constructor is used to perform any necessary initializations when the object is being created. In Python, the constructor is a method named \_\_init\_\_. The first parameter of this method should be self, which refers to the instance or object of the current class.

**Syntax:**

**def \_\_init\_\_(self):**

**body of the constructor**

**In Python, Constructor is mandatory or not :---** No, it is not mandatory for a class to have a constructor. Whether a class includes a constructor depends entirely on the requirements. If any initialization is needed during object creation, then a constructor should be used. Otherwise, it is not necessary. A Python program remains valid even without a constructor.

class Test:

    def \_\_init\_\_(self):

        print("Constructor executed....!!!!!!!")

t = Test()

O/P:--

Constructor executed....!!!!!!!

##### **Can constructor called explicitly? :---**

##### Yes, we can call constructor explicitly with object name. But since the constructor gets executed automatically at the time of object creation, it is not recommended to call it explicitly.

class Student:

    def \_\_init\_\_(self):

        print("Constructor called............")

obj = Student() # Constructor called implecitilly or automatically when we are creating object...

obj.\_\_init\_\_() # we are calling explicitally constructor method

O/P:--

Constructor called............

Constructor called............

**Note:** Including a constructor is not mandatory. If we do not include a constructor, Python will internally provide an empty constructor. This can be verified using the dir(class\_name) built-in method.

# Constructor is not mandatory for any class, it is optional on the bases of our requirement.

Class Test:

    def m1(self):

        print(“Instence method executed….!!!!!!”)

t = Test()

t.m1()

print(dir(Test))

O/P:--

Instence method executed….!!!!!!

[‘\_\_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\_\_’,

‘m1’]

##### **How many parameters we passed in constructor:---**

##### Constructor can accept n number of parameters. It totally depends on our requirements. All values that need to be initialized during object creation should be passed to the constructor. The first parameter of the constructor should always refer to the current instance, which is typically denoted as self.

##### Without parameter (except self):

class Student:

    def \_\_init\_\_(self):

        print("Constructor called............")

        print(self) #

stu = Student()

O/P:--

Constructor called............

<\_\_main\_\_.Student object at 0x00000245668B3400>

**Hear,** self contains the current object address.

**With parameters:**

class Student:

    ''' This class is develop by Neeraj for demo'''

    def \_\_init\_\_(self,name,roll,marks):

        self.name=name

        self.roll=roll

        self.marks = marks

    def display(self):

        print("my name is", self.name)

        print("my roll no is", self.roll)

        print("my marks is", self.marks)

# help(Student)

obj1= Student("Neeraj",101,84)

print(obj1.name)

print(obj1.roll)

print(obj1.marks)

print(Student.\_\_doc\_\_)

obj1.display()

O/P:--

Neeraj

101

84

This class is develop by Neeraj for demo

my name is Neeraj

my roll no is 101

my marks is 84

**Multiple constructors in class:**

We can define multiple constructors (\_\_init\_\_()) methods in a class but always last one is executed.

class Student:

    ''' This class is develop by Neeraj for demo'''

    def \_\_init\_\_(self,name,roll,marks):

        self.name=name

        self.roll=roll

        self.marks = marks

    def \_\_init\_\_(self,name,roll,marks,city):

        self.name=name

        self.roll=roll

        self.marks = marks

        self.city = city

def display(self):

        print("my name is", self.name)

        print("my roll no is", self.roll)

        print("my marks is", self.marks)

        print("my city is", self.city)

# help(Student)

obj1= Student("Neeraj",101,84)

print(obj1.name)

print(obj1.roll)

print(obj1.marks)

print(Student.\_\_doc\_\_)

obj1.display()

O/P:---

obj1= Student("Neeraj",101,84)

TypeError: Student.\_\_init\_\_() missing 1 required positional argument: 'city'

Note :- This issue occurs because hear we define multiple constructor but out of all only last constructor will be called, and it required 4 arguments.

class Student:

    ''' This class is develop by Neeraj for demo'''

    def \_\_init\_\_(self,name,roll,marks):

        self.name=name

        self.roll=roll

        self.marks = marks

    def \_\_init\_\_(self,name,roll,marks,city):

        self.name=name

        self.roll=roll

        self.marks = marks

        self.city = city

    def display(self):

        print("my name is", self.name)

        print("my roll no is", self.roll)

        print("my marks is", self.marks)

        print("my city is", self.city)

obj1= Student("Neeraj",101,84,"Bhopal")

print(obj1.name)

print(obj1.roll)

print(obj1.marks)

print(obj1.city)

obj1.display()

O/P:--

Neeraj

101

84

Bhopal

my name is Neeraj

my roll no is 101

my marks is 84

my city is Bhopal

# Types of Variables in a Class in Python :---

**Inside a class, we can have three types of variables. They are:**

1. **Instance variables (object level variables)**
2. **Static variables (class level variables)**
3. **Local variables**

**1. Instance Variables in Python:**

If the value of a variable is changing from object to object then such variables are called as instance variables.

We declare instance variable:

1. **Through constructor**
2. **Through instance method**
3. **Through object**

# 1. Instance variable declare through constructor…………………

class Student:

    ''' This class is develop by Neeraj for demo'''

    def \_\_init\_\_(self,name,roll,marks,city):

        self.name=name

        self.roll=roll

        self.marks = marks

        self.city = city

    def display(self):

        print("my name is", self.name)

        print("my roll no is", self.roll)

        print("my marks is", self.marks)

        print("my city is", self.city)

stu1 = Student("Neeraj",101,"90","Bhopal")

stu2 = Student("Rahul",102,"92","Indore")

print(stu1.name)

print(stu2.name)

stu1.display()

stu2.display()

print(stu1.\_\_dict\_\_)

print(stu2.\_\_dict\_\_)

O/P:-----------------

Neeraj

Rahul

my name is Neeraj

my roll no is 101

my marks is 90

my city is Bhopal

my name is Rahul

my roll no is 102

my marks is 92

my city is Indore

{'name': 'Neeraj', 'roll': 101, 'marks': '90', 'city': 'Bhopal'}

{'name': 'Rahul', 'roll': 102, 'marks': '92', 'city': 'Indore'}

# 2. Instance variable declare through instance method…………………

class Student:

    ''' This class is develop by Neeraj for demo'''

    def display(self,name,roll,marks,city):

        self.name=name

        self.roll=roll

        self.marks = marks

        self.city = city

        print("my name is", self.name)

        print("my roll no is", self.roll)

        print("my marks is", self.marks)

        print("my city is", self.city)

stu = Student()

stu.display("Neeraj",101,"90","Bhopal")

print(stu.name)

stu.display("Rahul",102,"92","Indore")

print(stu.name)

print(stu.\_\_dict\_\_)

O/P:--

my name is Neeraj

my roll no is 101

my marks is 90

my city is Bhopal

Neeraj

my name is Rahul

my roll no is 102

my marks is 92

my city is Indore

Rahul

{'name': 'Rahul', 'roll': 102, 'marks': '92', 'city': 'Indore'}

# 3. Instance variable declare through object…………………

class Student:

    def \_\_init\_\_(self):

        print("This is constructor")

    def m1(self):

        print("This is instance method")

t=Student()

t.m1()

t.a=10

t.b=20

t.c=55

print(t.a)

print(t.b)

print(t.c)

print(t.\_\_dict\_\_)

O/P:--

This is constructor

This is instance method

10

20

55

{'a': 10, 'b': 20, 'c': 55}

**Accessing instance variables**

The instance variable can be accessed in two ways:

1. By using self variable
2. By using object name

**By using self variable:---** We can access instance variables within the class by using self variable.

# Access instence variable...........(by using self reference variable)

class Student:

   def \_\_init\_\_(self):

       self.a=10

       self.b=20

   def display(self):

       print(self.a)

       print(self.b)

s= Student()

s.display()

O/P:---

10

20

**By using object name :---** We can access instance variables outside of the class by using object name.

# Access instence variable...........(by using object name)

class Student:

    def \_\_init\_\_(self):

        self.a=10

        self.b=20

s= Student()

print(s.a)

print(s.b)

O/p:--

10

20

##### **2. Static Variables in Python**

If a variable's value remains consistent across all instances of a class, it is referred to as a static variable or class-level variable. These variables can be accessed using either the class name or an object name. However, it is strongly recommended to access static variables through the class name to enhance code clarity and avoid potential confusion.

**Declare static variable:**

1. Inside the class and outside of the methods
2. Inside Constructor
3. Inside instance method
4. Outside of the class

class Student:

    quali = "M.Tech"   # static variable declare inside the class

    def \_\_init\_\_(self,name,age):

        self.name = name

        self.age = age

        Student.school = "SHSS"  # static variable declare inside the constructor

    def display(self):

        Student.gread = "P.hd"   # static variable declare inside the instence variable

        print("Name = ",self.name)

        print("Age = ",self.age)

        print("Quali =",Student.quali)  # static variable access inside the class through

class name

        print("School = ",Student.school) # static variable access inside the class

through class name

        print("Gread = ",Student.gread) # static variable access inside the class

through class name

        print("Achivment = ",Student.achivment)   # static variable access inside the

class through class name

obj = Student("Neeraj",37)

Student.achivment="Gate-qualified"   # static variable declare outside of the class

print("Access through class\_Name = ",Student.quali) # static variable access outside the class through class name

print("Access through object = ",obj.quali) # static variable access outside the class through object

obj.display()

print("Access through class\_Name = ",Student.gread) # static variable access outside the class through class name

print("Access through class\_Name = ",Student.school)# static variable access outside the class through class name

print("Access through class\_Name = ",Student.achivment)# static variable access outside the class through class name

# obj.display()

print(dir(Student))

O/P:----

Access through class\_Name =  M.Tech

Access through object =  M.Tech

Name =  Neeraj

Age =  37

Quali = M.Tech

School =  SHSS

Gread =  P.hd

Achivment =  Gate-qualified

Access through class\_Name =  P.hd

Access through class\_Name =  SHSS

Access through class\_Name =  Gate-qualified

**Accessing static variables:-**

The **static** variable can be accessed in two ways:

1. By using class name (highly recommended).
2. By using object name.

**Accessing static variables either outside or inside of the class by using either class name or object name.**

class Student:

    ''' This class is develop by Neeraj for demo'''

    School\_name="SHSC"

    def \_\_init\_\_(self,name,roll,marks,city):

        self.name=name

        self.roll=roll

        self.marks = marks

        self.city = city

    def display(self):

        print("my name is", self.name)

        print("my roll no is", self.roll)

        print("my marks is", self.marks)

        print("my city is", self.city)

        print(Student.School\_name) # Access static variable by using class name

        print(stu.School\_name) # Access static variable by using object name

stu = Student("Neeraj",101,"90","Bhopal")

stu.display()

print(stu.School\_name)  # Access static variable by using object name

print(Student.School\_name) # Access static variable by using class name

O/P:--

my name is Neeraj

my roll no is 101

my marks is 90

my city is Bhopal

SHSC

SHSC

SHSC

SHSC

**3. Local Variables in Python:**

A variable declared inside a method is known as a local variable. Local variables are typically created for temporary use within methods. Their scope is restricted to the method in which they are declared, meaning they are not accessible outside of that method.

# Local variable...........

class Demo:

   def m(self):

       a=10 #Local Variable

       print(a)

   def n(self):

       print(a) #'a' is local variable of m() so it raise error

d=Demo()

d.m()

d.n()

O/P:--

print(a) #'a' is local variable of m()

NameError: name 'a' is not defined

If you want to access local variable outside of the block then use global keyword.

class Demo:

   def m(self):

       global a

       a=10 #Local Variable

       print("instence methos m()")

       print(a)

   def n(self):

       print("instence methos n()")

       print(a) #'a' is local variable of m()

d=Demo()

d.m()

d.n()

print("access local variable outside of the class")

print(a)

O/P:--

instence methos m()

10

instence methos n()

10

access local variable outside of the class

10

**Types of Methods in a Class**

In python, we can classify the methods into three types from the perspective of object oriented programming.

1. Instance Methods
2. Class Methods
3. Static Methods

**Instance Methods in Python:**

Instance methods are methods that operate on the instance variables of a class. They are associated with instances or objects, hence the name "instance methods." The first parameter of an instance method should be the self variable, which refers to the instance. In addition to self, instance methods can include other parameters as well.

# instence method

class Demo:

   def \_\_init\_\_(self, a):

       self.a=a

   def m(self):

       print(self.a)

d=Demo(10)

d.m()

O/P:-

10

**Class Methods in Python:**

Class methods are methods that operate on the class variables, also known as static variables, of the class. They are used when the method primarily interacts with class variables rather than instance variables.

1. Class methods should be declared with @classmethod.
2. Instead of self , hear we use cls
3. We can access class methods by using class name or object reference.

# class method

class Book:

    price = 1000 # static variable or class variable

    @classmethod   # with the help of class method we can modified static

variable or class variable

    def update\_price(cls):

        cls.price = 1500

    def display(self):

        print("Price of book =",Book.price)

obj = Book()

obj.update\_price()

obj.display()

O/P:---

Price of book = 1500

Example: 2

class Book:

    price = 1000 # static variable or class variable

    def \_\_init\_\_(self,name,author):

        self.name=name

        self.author = author

    @classmethod  # with the help of class method we can modified the value of

static variable or class variable

    def update\_price(cls):

        cls.price = 1500

    def display(self):

        print("Name of book = ",self.name)

        print("Name of author = ",self.author)

        print("Price of book =",Book.price)

obj = Book("Python","Guido van Rossum")

obj.update\_price()

obj.display()

O/P:-------

Name of book =  Python

Name of author =  Guido van Rossum

Price of book = 1500

**Static Methods in Python:** The static methods, in general, utility methods. Inside these methods we won’t use any instance or class variables. No arguments like cls or self are required at the time of declaration.

1. We can declare static method explicitly by using @staticmethod decorator.
2. We can access static methods by using class name or object reference.

# static method

class Student:

    @staticmethod

    def great():

        return "Thank for visiting our site"

    @staticmethod

    def great1():

        return "Please visit again"

    def normal():

        return "---------!!!!!----------"

obj = Student

# access static method

print("Access through class =",Student.great())

print("Access through object =",obj.great1())

# access normal method

print("Access through class =",Student.normal())

print("Access through object =",obj.normal())

O/P:--

Access through class = Thank for visiting our site

Access through object = Please visit again

Access through class = ---------!!!!!----------

Access through object = ---------!!!!!----------

**Mixed\_Examples for practice----------------------------------------**

# ----------------- class without constructor------------

class Student:

    def display():

        print("Without constructor")

obj = Student

obj.display()

# --------------------------------------------------

class Student:

    def display(self,name,age):

        self.name=name

        self.age = age

obj = Student()

obj.display("Neeraj",35)

# --------------------------------------------------

class Student:

    def \_\_init\_\_(self,quali):

        self.quali = quali

    def display(self,name,age):

        self.name=name

        self.age = age

        print("Name=",self.name)

        print("Age=",self.age)

        print("Quali=",self.quali)

        print("NAme=",name)

        print("Age=",age)

    def display1(self):

        print("Name=",self.name)

        print("Age=",self.age)

        print("Quali=",self.quali)

        print("We can not access name hear without self ")

        print("We can not access age hear without self ")

obj = Student("M.Tech")

obj.display("Neeraj",35)

obj.display1()

# --------------------------------------------------

class Student:

    def display(self,name,age):

        self.name=name

        self.age = age

        print("Name=",self.name)

        print("Age=",self.age)

        print("NAme=",name)

        print("Age=",age)

    def display1(self):

        print("Name=",self.name)

        print("Age=",self.age)

        print("We can not access name hear without self ")

        print("We can not access age hear without self ")

obj = Student()

obj.display("Neeraj",35)

obj.display1()

# --------------------------------------------------

class Student:

    def display(self,name,age):

        self.name=name

        self.age = age

        print("Name=",self.name)

        print("Age=",self.age)

        print("Name=",name)

        print("Age=",age)

        print("Quali=",self.quali)

    def display1(self):

        print("Name=",self.name)

        print("Age=",self.age)

        print("We can not access name hear without self ")

        print("We can not access age hear without self ")

obj = Student()

obj.quali = "M.tech"   # define instence variable through object.

obj.display("Neeraj",35)

obj.display1()

print(obj.name)

print(obj.age)

print(obj.quali)

# ----------------------------------------------------

class New\_class:

    Qualification = "M.Tech"

    def stu\_detail(name,age):

        print("Stu\_name",name)

        print("Stu\_age",age)

        print("Stu\_Qualification",New\_class.Qualification)

obj = New\_class

obj.stu\_detail("Neeraj Kumar",37)

# class with constructor ----------------------------------

class New\_class:

    "This is for testing perpose only"

    Qualification = "M.Tech"

    def \_\_init\_\_(self,name,age):

        self.name = name

        self.age = age

    def stu\_detail(self):

        print("Stu\_name",self.name)

        print("Stu\_age",self.age)

        print("Stu\_Qualification",New\_class.Qualification)

obj = New\_class("Neeraj Kumar",37)

obj.\_\_init\_\_("Rahul",35)

obj.stu\_detail()

# The dir() function returns all variables/properties and methods of the specified object, without the values. This function will return all the  built-in properties which are default for all object.

print(dir(New\_class))

print(New\_class.\_\_doc\_\_)

print(New\_class.\_\_module\_\_)

# constructor without parameters..........

class New\_class:

    "This is for testing perpose only"

    Qualification = "M.Tech"

    def \_\_init\_\_(self):

        print(".......Constructor without arguments....")

    def stu\_detail(self,name,age):

        print("Stu\_name",name)

        print("Stu\_age",age)

        print("Stu\_Qualification",New\_class.Qualification)

obj = New\_class()

obj.stu\_detail("Neeraj Kumar",37)

# constructor with parameters..........

class New\_class:

    "This is for testing perpose only"

    Qualification = "M.Tech"

    def \_\_init\_\_(self,name,age):

        self.name = name

        self.age = age

    def stu\_detail(self):

        print("Stu\_name",self.name)

        print("Stu\_age",self.age)

        print("Stu\_Qualification",New\_class.Qualification)

obj = New\_class("Neeraj Kumar",37)

obj.stu\_detail()

# constructor called explicitally----------------

class Student:

    ''' This class is develop by Neeraj for demo'''

    def \_\_init\_\_(self,name,roll,marks,city):

        self.name=name

        self.roll=roll

        self.marks = marks

        self.city = city

    def display(self):

        print("my name is", self.name)

        print("my roll no is", self.roll)

        print("my marks is", self.marks)

        print("my city is", self.city)

obj1= Student("Neeraj",101,84,"Bhopal")

obj1.\_\_init\_\_("Rahul",101,84,"Bhopal")

print(obj1.name)

print(obj1.roll)

print(obj1.marks)

obj1.display()

# ----------------------- multiple constructor at a time--------------------------

class Student:

    ''' This class is develop by Neeraj for demo'''

    def \_\_init\_\_(self,name,roll,marks):

        self.name=name

        self.roll=roll

        self.marks = marks

    def \_\_init\_\_(self,name,roll,marks,city):

        self.name=name

        self.roll=roll

        self.marks = marks

        self.city = city

    def display(self):

        print("my name is", self.name)

        print("my roll no is", self.roll)

        print("my marks is", self.marks)

        print("my city is", self.city)

# help(Student)

# obj1= Student("Neeraj",101,84)

obj1= Student("Neeraj",101,84,"Bhopal")

print(obj1.name)

print(obj1.roll)

print(obj1.marks)

obj1.display()

# ---------------------------- instance variable--------------------------------

#---------------------------- Through Constructor ------------------------------

class Student:

    "...........Through Contructor.........."

    def \_\_init\_\_(self,name,age):

        self.name = name

        self.age = age

    def display(self):

        print("Name=",self.name)

        print("Age=",self.age)

obj = Student("Neeraj",35)

obj.display()

class Student:

    "...........Through Contructor.........."

    def \_\_init\_\_(self,name,age):

        self.name = name

        self.age = age

obj = Student("Neeraj",37)

print(obj.\_\_dict\_\_)

#-------------------------- Through Instence method ------------------------------

class Student:

    "...........Through Instence Method.........."

    def display(self,name,age):

        self.name = name

        self.age = age

    def display1(self):

        print("Name=",self.name)

        print("Age=",self.age)

obj = Student()

obj.display("Neeraj",37)

obj.display1()

print(obj.\_\_dict\_\_)

class Student:

    "...........Through Instence Method.........."

    def display(self,name,age):

        self.name = name

        self.age = age

    def display1(self):

        print("Name=",self.name)

        print("Age=",self.age)

obj = Student()

obj.display("Neeraj",35)

obj.display1()

print(obj.\_\_dict\_\_)

#------------------------------- Through Object -------------------------------

class Student:

    "...........Through Object.........."

    def new(self):

        print("Name = ", self.name)

        print("Age = ",self.age)

obj = Student()

obj.name = "Neeraj"

obj.age = 37

print(obj.\_\_dict\_\_)

obj.new()

# ---------------------------- class variable--------------------------------

class Student:

    quali = "M.Tech"   # static variable declare inside the class

    def \_\_init\_\_(self,name,age):

        self.name = name

        self.age = age

        Student.school = "SHSS"  # static variable declare inside the constructor

    def display(self):

        Student.gread = "P.hd"   # static variable declare inside the instence variable

        print("Name = ",self.name)

        print("Age = ",self.age)

        print("Quali =",Student.quali)  # static variable access inside the class through class name

        print("School = ",Student.school) # static variable access inside the class through class name

        print("Gread = ",Student.gread) # static variable access inside the class through class name

        print("Achivment = ",Student.achivment)   # static variable access inside the class through class name

obj = Student("Neeraj",37)

Student.achivment="Gate-qualified"   # static variable declare outside of the class

print("Access through class\_Name = ",Student.quali) # static variable access outside the class through class name

print("Access through object = ",obj.quali) # static variable access outside the class through object

obj.display()

print("Access through class\_Name = ",Student.gread) # static variable access outside the class through class name

print("Access through class\_Name = ",Student.school)# static variable access outside the class through class name

print("Access through class\_Name = ",Student.achivment)# static variable access outside the class through class name

# ---------------------------- local variable--------------------------------

class Student:

    def local(self):

        global a

        a=10            # Local variable declare in instance method

        print("Value of a = ",a)

    @staticmethod

    def local2():

        global b

        b=20            # Local variable declare in method

    def local1(self):

        print("Value of a = ",a)

        print("Value of b = ",b)

obj = Student()

obj.local()

Student.local2()

obj.local1()

# ---------instence method------------------

class Student:

    def first(self):

        print("This is first instance method....")

    def second(self):

        self.first()

        print("This is second instance method....")

    def third(self):

        self.second()

        print("This is third instance method....")

obj = Student()

obj.third()

# --------------------class\_method------------------

class Book:

    price = 1000 # static variable or class variable

    def \_\_init\_\_(self,name,author):

        self.name=name

        self.author = author

    @classmethod  # with the help of class method we can modified static variable or class variable

    def update\_price(cls):

        cls.price = 1500

    def display(self):

        print("Name of book = ",self.name)

        print("Name of author = ",self.author)

        print("Price of book =",Book.price)

obj = Book("Python","Guido van Rossum")

obj.update\_price()

obj.display()

# ------------------------------------------------------------

class Book:

    price = 1000 # static variable or class variable

    @classmethod  # with the help of class method we can modified static variable or class variable

    def update\_price(cls):

        cls.price = 1500

    def display(self):

        print("Price of book =",Book.price)

obj = Book()

obj.update\_price()

obj.display()

# -------------------- static\_method------------------

class Student:

    @staticmethod

    def great():

        return "Thank for visiting our site"

    @staticmethod

    def great1():

        return "Please visit again"

    def normal():

        return "---------!!!!!----------"

obj = Student

# access static method

print("Access through class =",Student.great())

print("Access through object =",obj.great1())

# access normal method

print("Access through class =",Student.normal())

print("Access through object =",obj.normal())

# --------------------class & static\_method------------------

class Student:

    quali = "M.Tech"

    def \_\_init\_\_(self,name,age):

        self.name = name

        self.age = age

    @classmethod

    def update(cls,quali):

        cls.quali = quali

    def display(self):

        print("Name = ",self.name)

        print("Age = ",self.age)

        print("Quali = ",Student.quali)

    @staticmethod

    def great():

        print("Thanks to visit")

        print("visit again")

obj = Student("Neeraj",37)

Student.update("B.Tech")

obj.display()

Student.great()