OOP:

Classes and Objects in python

class:

In [1]:
```
class democlass:        # called class
    a=10
```

object:

In [3]:
```
demo_Object=democlass()      # called object and call class.
print(demo_Object.a)
```

```
10
```

self:

```
we use any variable in place of self.
self work as an object and always use as a parameter when we make function in class.
```

In [8]:
```
class democlass2:
    a=10
    def sum(self):        # always pass one argument when we make function here we use variable self
          print (20+30)    # how we make function in class
demo_Object2=democlass2() # object declare here
print (demo_Object2.a)
demo_Object2.sum()        # call function
```

```
10
50
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [10]:
```
class demo:
    a=10
    def sum1(self):
        print(self.a)  # always we follow this syntax when we make function in class and then call.
object=demo()
object.sum1()
```

```
10
```

In [11]:
```
class demo:
    a=10
    def sum1(self):
        self.c=self.a * self.a           # 10 * 10
        print (self.c)
object=demo()
object.sum1()
```

```
100
```

we use more arguments with self

```
methods:
```

In [16]:
```
class student:
    a=10
    def sum2(self,a,b):
          print (a+b)
object=student()
object.sum2(10,20)    # by passing parameters
```

```
30
```

constructor:

```
we call automatically .
define constructure with __init__ keyword and always we use self variable
object bnaty hi call ho jata ha
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
In [20]: class student:
             def __init__ (self):
                 print ("here we call constructor automatically when we make object:")
         object=student()
```

```
here we call constructor automatically when we make object:
```

```
In [23]: class student:
             def __init__ (self, name, address):
                 self.name=name
                 self.address=address
         object=student("Arooj", "Fsd")
         print(object)
```

```
<__main__.student object at 0x0000029F284A5000>
```

```
In [25]: print(object.name)
```

```
Arooj
```

```
In [26]: print(object.address)
```

```
Fsd
```

```
In [33]: class Person:
           def __init__(self, name, age):
             self.name = name
             self.age = age

           def myfunc(self):
             print("Hello my name is " + self.name)

         p1 = Person("John", 36)
         p1.myfunc()
```

```
Hello my name is John
```

2: *str*

we use second method of constructor is _ *str* _ :

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [32]:
```python
class student:
    def __init__ (self, name, address):
        self.name=name
        self.address=address
    def __str__ (self):
        s="name="+self.name +"\n "+ "address="+self.address
        return s
object=student("Arooj", "Fsd")
print(object)
```

```
name=Arooj
 address=Fsd
```

In [37]:
```python
class apple:
    def __init__(self, color, flavour):
        self.color = color
        self.flavour= flavour
    def __str__(self):
        return "this apple is {} and its flavour is {}" .format(self.color, self.flavour)
jonagold =apple("red", "sweet")
print(jonagold.color)
print(jonagold)
```

```
red
this apple is red and its flavour is sweet
```

In [40]:
```python
class apple:
    "HI , my name is arooj"               # show this string in output by using help
    def __init__(self, color, flavour):
        self.color = color
        self.flavour= flavour
    def __str__(self):
        return "this apple is{} and its flavour is {}" .format(self.color, self.flavour)
jonagold =apple("red", "sweet")
```

In [41]:
```python
help(apple)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
Help on class apple in module __main__:

class apple(builtins.object)
 |  apple(color, flavour)
 |  
 |  HI , my name is arooj
 |  
 |  Methods defined here:
 |  
 |  __init__(self, color, flavour)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |  
 |  __str__(self)
 |      Return str(self).
 |  
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |  
 |  __dict__
 |      dictionary for instance variables (if defined)
 |  
 |  __weakref__
 |      list of weak references to the object (if defined)
```

Inheritance :

aik object ko hm multiple classes mn call kr skty hain

single inheritance:

In [43]:
```python
class A:
    def displayA(self):
        print("welcome to class A")
class B(A):                     #chng here class B joined with class A
    def displayB(self):
        print("welcome to class B")
obj=B()
obj.displayA()
obj.displayB()
```

```
welcome to class A
welcome to class B
```

Multilevel inheritance:

In [45]:
```python
class A:
    def displayA(self):
        print("welcome to class A")
class B(A):                     #chng here class B joined with class A
    def displayB(self):
        print("welcome to class B")
class C(B):                     #chng here class C joined with class B
    def displayC(self):
        print("welcome to class C")
obj=C()
obj.displayA()
obj.displayB()
obj.displayC()
```

```
welcome to class A
welcome to class B
welcome to class C
```

Multiple inheritance:

```
only support python language not others like java php etc
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
In [47]: class A:
             def displayA(self):
                 print("welcome to class A")
         class B():
             def displayB(self):
                 print("welcome to class B")
         class C(A,B):                  #chng here class C joined with class A,B directly.
             def displayC(self):
                 print("welcome to class C")
         obj=C()
         obj.displayA()
         obj.displayB()
         obj.displayC()
```

```
welcome to class A
welcome to class B
welcome to class C
```

```
In [48]: class Animal:
          sound=""
          def __init__(self,name):
                 self.name=name
          def speak(self):
             print("{sound} I'm {name}! {sound}".format(name=self.name, sound=self.sound))
         class Cat(Animal):
          sound="Meow!"
         myLuna=Cat("Luna")
         myLuna.speak()
```

```
Meow! I'm Luna! Meow!
```

In [49]:
```python
# in this case we create two objects:
class Animal:
 sound=""
 def __init__(self,name):
        self.name=name
 def speak(self):
    print("{sound} I'm {name}! {sound}".format(name=self.name, sound=self.sound))
class Cat(Animal):
 sound="Meow!"
myLuna=Cat("Luna")
myLuna.speak()

class Cow(Animal):
 sound="Meooo!"
mycow=Cow("mily")
mycow.speak()
```

```
Meow! I'm Luna! Meow!
Meooo! I'm mily! Meooo!
```

In [52]:
```python
class clothing:
    material =""
    def __init__(self, name):
        self.name = name
    def checkmaterial(self):
        print("this {} is made of {}".format(self.name, self.material))
class shirt (clothing):
    material="cotton"
polo = shirt ("polo")
polo.checkmaterial()
```

```
this polo is made of cotton
```

object composition:

```
dictionary >>.key .value .name these are three methods
without inheritance we call function  in composition
another class method we call in our class
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Inheritance will extend the functionality with extra features allows overriding of methods

```
Composition, we can only use that class we can not modify or extend the functionality of it.
It will not provide extra features.
```

In [53]:
```
class Component:
    def __init__(self):
        print('Component class object created')

    def m1(self):
        print('Component class m1() method executed')

 # make second class here.
class Composite:
    def __init__(self):
        # creating object of component class

        self.obj1 = Component()
        print('Composite class object also created...')

     # composite class instance method
    def m2(self):

        print('Composite class m2() method executed...')

        # calling m1() method of component class
        self.obj1.m1()


# creating object of composite class
obj2 = Composite()

# calling m2() method of composite class
obj2.m2()
```

```
Component class object created
Composite class object also created...
Composite class m2() method executed...
Component class m1() method executed
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js