Python Class

#Python Notes

OOP’s Concept

Class

- A class is a blueprint for the object.
- Classes are defined by the **"Class"** keyword.
- Class name in python is preceded with class keyword followed by colon (:).
- Classes commonly contains data field to store the data and methods for defining behaviours.
- **Syntax :-**

```
class MyClass:
```

Object

- Object is an instance of a class.

- **Syntax :-**

  myc= MyClass()

- Here, **myc is object of class MyClass**.

Example

File Edit View Navigate Code Help

scratch.py

```python
class MyClass:

    def func1(self):
        print('Hello')

    def func2(self,name):
        print('Name is : '+name)

# create a object of MyClass
myc = MyClass()

# Calling function
myc.func1()
myc.func2("Akash Technolabs")
```

MyClass › func2()

Run: scratch

```
Hello
Name is : Akash Technolabs

Process finished with exit code 0
```

Note

- Everything in a class is indented, just like the code in the function, loop, if statement, etc. Anything not indented is not in the class

File Edit View Navigate Code Help

scratch.py

```python
class MyClass:

    def func1(self):
        print('Hello')

    def func2(self,name):
        print('Name is : '+name)

# create a object of MyClass
myc = MyClass()

# Calling function
myc.func1()
myc.func2("Akash Technolabs")
```

Difference between Method and Function

Method	Functions
Method is called by its name, but it is associated to an object (dependent).	Function is block of code that is also called by its name. (independent)
It may or may not return any data.	It may or may not return any data.
A method can operate on the data (instance variables) that is contained by the corresponding class	Function does not deal with Class and its instance concept.

What is self?

- All methods in python including some special methods like initializer have first parameter self .
- This parameter refers to the object which invokes the method.
- When you create new object the self parameter in the __init__ method is automatically set to reference the object you have just created.

Sum Of 2 Numbers Using Class

scratch.py

```python
class MyClass:

    def func1(self,n1,n2):
        ans=n1+n2
        print('Ans is : ',ans)

# create a object of MyClass
myc = MyClass()

# Calling function
myc.func1(10,20)
```

Run: scratch

```
Ans is :  30

Process finished with exit code 0
```

Python Constructors

- A constructor is a class function that instantiates an object to predefined values.
- It begins with a double underscore (_). It __init__() method.
- There are two types of Constructor.
  - Default Constructor
  - Parameterized Constructor

Default Constructor Example

scratch_1.py

```python
class Myclass:

    #Default Constructor
    def __init__(self):
        print("Hello World")

m1 = Myclass()

```

Myclass › __init__()

Run: scratch_1

```
Hello World

Process finished with exit code 0
```

Parameterized Constructor Example

scratch_1.py

```python
class Myclass:

    #Parameterized Constructor
    def __init__(self,name):
        print("Value is ",name)

m1 = Myclass("Akash Technolabs")

```

Myclass › __init__()

Run: scratch_1

```
Value is  Akash Technolabs

Process finished with exit code 0
```

Assign String Value to Class Variable Using Method

scratch_1.py

```python
class Myclass:
    name=""

    def func1(self):
        print("Hello Function1")

    def func2(self,name):
       self.name=name

    def func3(self):
        print("Value is ", self.name)

m1 = Myclass()
m1.func1()
m1.func2("Akash Technolabs")
m1.func3()
```

Run: scratch_1

```
C:\Users\Devanshi\PycharmProjects\demo\venv\Scr
Hello Function1
Value is  Akash Technolabs

Process finished with exit code 0
```

Assign String Value to Class Variable Using Constructor

File Edit View Navigate Code Help

scratch.py

```python
class MyClass:
    name = ""

    def __init__(self, name):
        self.name = name

    def func1(self):
        print('Name is : ',self.name)

# create a object of MyClass
myc = MyClass("Akash Technolabs")
```

MyClass › __init__()

Run: scratch

```
Name is :  Akash Technolabs

Process finished with exit code 0
```

Example 2

File Edit View Navigate Code Help

scratch.py

```python
class MyClass:
    n1 = 0
    n2 = 0
    # Constructor
    def __init__(self,n1,n2):
        self.n1 = n1
        self.n2 = n2
    # Function
    def func1(self):
        ans=self.n1+self.n2
        print('Ans is : ',ans)

# create a object of MyClass
myc = MyClass(10,20)

# Calling function
myc.func1()
```

MyClass

Run: scratch

```
Ans is :  30

Process finished with exit code 0
```

Inheritance

- Inheritance allows programmer to create a general class first then later extend it to more specialized class.
- Using inheritance you can inherit all access data fields and methods, plus you can add your own methods and fields, thus inheritance provide a way to organize code, rather than rewriting it from scratch.

Syntax to create a subclass is:

- class SubClass(SuperClass):

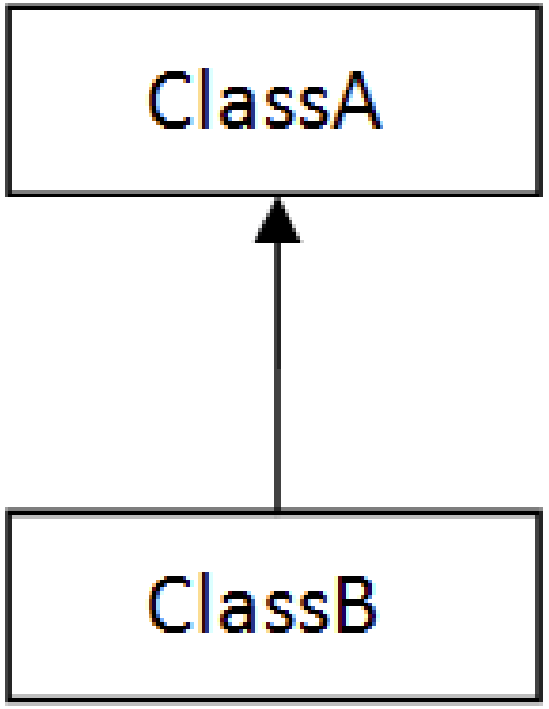
```
    # data fields
    # instance methods
```

Types of Inheritance

- Single-Level inheritance
- Multi-Level inheritance
- Multiple inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

Single level Inheritance

- When one class inherits another class, it is known as single level inheritance.


ClassA
ClassB

Example

File Edit View Navigate Code Help

scratch.py

```python
# define parent class
class Parent:

    def __init__(self):
        print("Calling parent constructor")

    def parentMethod(self):
        print("Calling parent method")

# define child class
class Child(Parent):
    def __init__(self):
        print ("Calling child constructor")

    def childMethod(self):
        print ("Calling child method")

c = Child()             # object of child
c.childMethod()         # child class method
c.parentMethod()        # parent class method
```

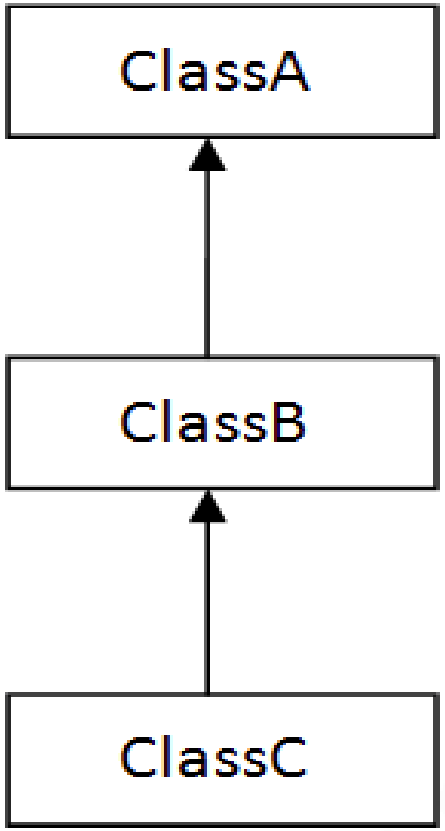
Output

```
Run: scratch
Calling child constructor
Calling child method
Calling parent method

Process finished with exit code 0
```

Multi level Inheritance

- When one class inherits another class which is further inherited by another class, it is known as multi level inheritance.


ClassA
ClassB
ClassC

Example

File Edit View Navigate Code Help

scratch.py

```python
# define parent class
class Parent:
    def __init__(self):
        print("Calling parent constructor")

    def parentMethod(self):
        print("Calling parent method")

# define child class
class Child(Parent):
    def __init__(self):
        print ("Calling child constructor")

    def childMethod(self):
        print ("Calling child method")
```

Cont…

File Edit View Navigate Code Help

scratch.py

```python
        print ("Calling child constructor")

    def childMethod(self):
        print ("Calling child method")

# define sub child class
class SubChild(Child):
    def __init__(self):
        print ("Calling sub child constructor")

    def subchildMethod(self):
        print ("Calling sub child method")


sc = SubChild()          # object of sub child
sc.subchildMethod()      # sub child class method
sc.childMethod()         # child class method
sc.parentMethod()        # parent class method
```

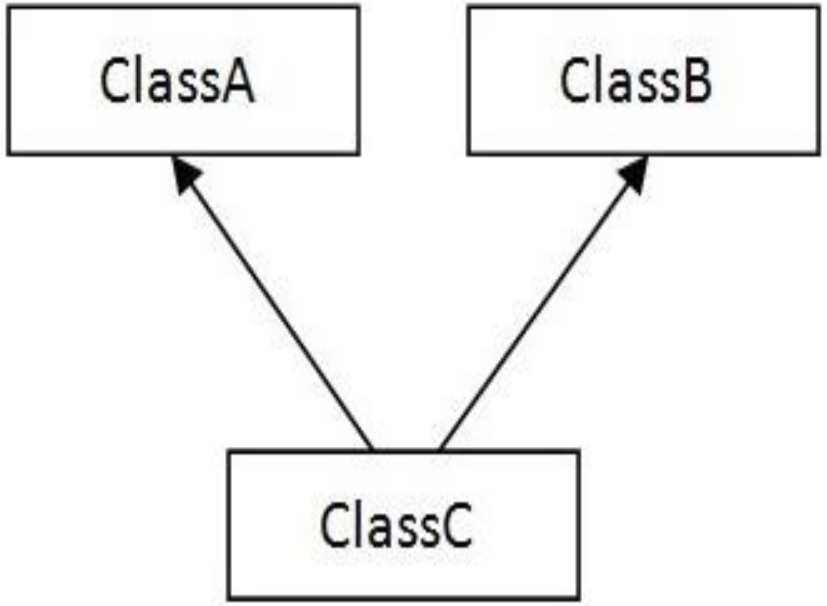
Output

```
Calling sub child constructor
Calling sub child method
Calling child method
Calling parent method

Process finished with exit code 0
```

Multiple Inheritance

- If a class can inherit members from more than one base class is known as Multiple Inheritance.


ClassA
ClassB
ClassC

Syntax

- Syntax :-

```
class A:        # define your class A
.....

class B:        # define your class B
.....

class C(A, B):   # subclass of A and B
.....
```

Example

File Edit View Navigate Code Help

scratch.py

```python
# define parent class1
class MyParentClass1():

    def method_Parent1(self):
        print("Parent1 method called")

# define parent class2
class MyParentClass2():

    def method_Parent2(self):
        print("Parent2 method called")

# define Child class
class ChildClass(MyParentClass1, MyParentClass2):

    def child_method(self):
        print("child method")

c = ChildClass()          # object of child
c.method_Parent1()        # parent class1 method
c.method_Parent2()        # parent class2 method
c.child_method()          # child class method
```

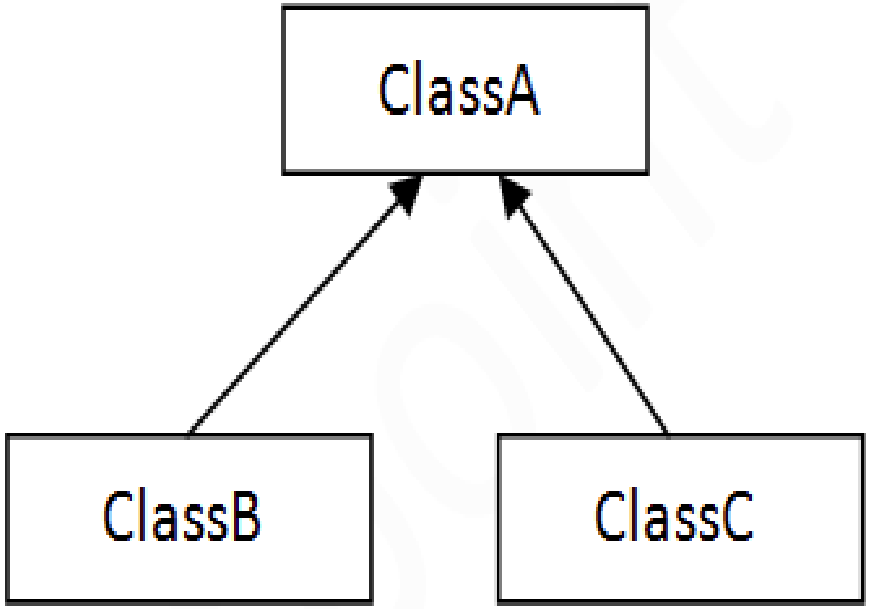
Output

```
Run: scratch
Parent1 method called
Parent2 method called
child method

Process finished with exit code 0
```

Hierarchical Inheritance

- If more than one class can inherit members from one class is known as Hierarchical Inheritance.


ClassA
ClassB
ClassC

Example

File Edit View Navigate Code Help

scratch.py

```python
# define parent class
class Parent:
    def __init__(self):
        print("Calling parent constructor")

    def parentMethod(self):
        print("Calling parent method")

# define child class
class Child(Parent):
    def __init__(self):
        print ("Calling child constructor")

    def childMethod(self):
        print ("Calling child method")
```

Cont…

File Edit View Navigate Code Help

scratch.py

```python
    def childMethod(self):
        print ("Calling child method")

# define sub child class
class Child2(Parent):
    def __init__(self):
        print ("Calling child2 constructor")

    def childMethod2(self):
        print ("Calling child2 method")


sc = Child2()           # object of child
sc.childMethod2()       # Child 2 class method
sc.parentMethod()       # parent class method
```

Output

```
Calling child2 constructor
Calling child2 method
Calling parent method

Process finished with exit code 0
```

Hybrid Inheritance

- Hybrid Inheritance is implemented by combining more than one type of inheritance.

Example

File Edit View Navigate Code Help

scratch.py

```python
# define parent class1
class MyParentClass1():

    def method_Parent1(self):
        print("Parent1 method called")

# define parent class2
class MyParentClass2():

    def method_Parent2(self):
        print("Parent2 method called")

# define Child class
class ChildClass(MyParentClass1, MyParentClass2):  #Multiple Inheritance

    def child_method(self):
        print("child method")
```

Cont…

File Edit View Navigate Code Help

scratch.py

```python

# define Child class
class ChildClass(MyParentClass1, MyParentClass2):  #Multiple Inheritance

    def child_method(self):
        print("child method")

# define Child class2
class ChildClass2(MyParentClass1):  #Hierarchical Inheritance

    def child_method2(self):
        print("child method2")

c = ChildClass()          # object of child
c.method_Parent1()        # parent class1 method
c.method_Parent2()        # parent class2 method
c.child_method()          # child class method

c2=ChildClass2()          # object of child class 2
c2.child_method2()        # child class2 method
c2.method_Parent1()       # parent class1 method

```

Output

```
Parent1 method called
Parent2 method called
child method
child method2
Parent1 method called

Process finished with exit code 0
```

Polymorphism

- Polymorphism is an ability (in OOP) to use common interface for multiple form (data types).

- Types of Polymorphism :-
  - Overloading Methods
  - Overriding Methods

Overriding Methods

- **Override** means having two methods with the same name but doing different tasks. It means that one of the methods overrides the other.
- If there is any method in the superclass and a method with the same name in a subclass, then by executing the method, the method of the corresponding class will be executed.

Example 1

File Edit View Navigate Code Help

scratch.py

```python
# define parent class
class ParentClass():

    def func1(self):
        print("Parent method called")

# define Child class
class ChildClass(ParentClass):

    def func1(self):
        print("Child method called")

# object of child
c = ChildClass()
c.func1()
```

Run: scratch

```
Child method called

Process finished with exit code 0
```

Example 2

File Edit View Navigate Code Help

scratch.py

```python
# define parent class
class ParentClass():

    def func1(self):
        print("Parent method called")

# define Child class
class ChildClass(ParentClass):

    def func1(self):
        print("Child method called")

# object of child
c = ChildClass()
c.func1()

# object of parent
p=ParentClass()
p.func1()
```

Output

```
Run: scratch
Child method called
Parent method called

Process finished with exit code 0
```

Overloading Methods

- Python does not supports method overloading.

- We may overload the methods but can only use the latest defined method.

Example

File Edit View Navigate Code Help

scratch.py

```python
# define class
class MyClass():

    def sum(self,n1,n2):
        ans=n1+n2
        print("ans is : ",ans)

    def sum(self,n1,n2,n3):
        ans=n1+n2+n3
        print("ans is : ",ans)

# object of class
p=MyClass()

# p.sum(10,20)

p.sum(10,20,30)
```

Run: scratch

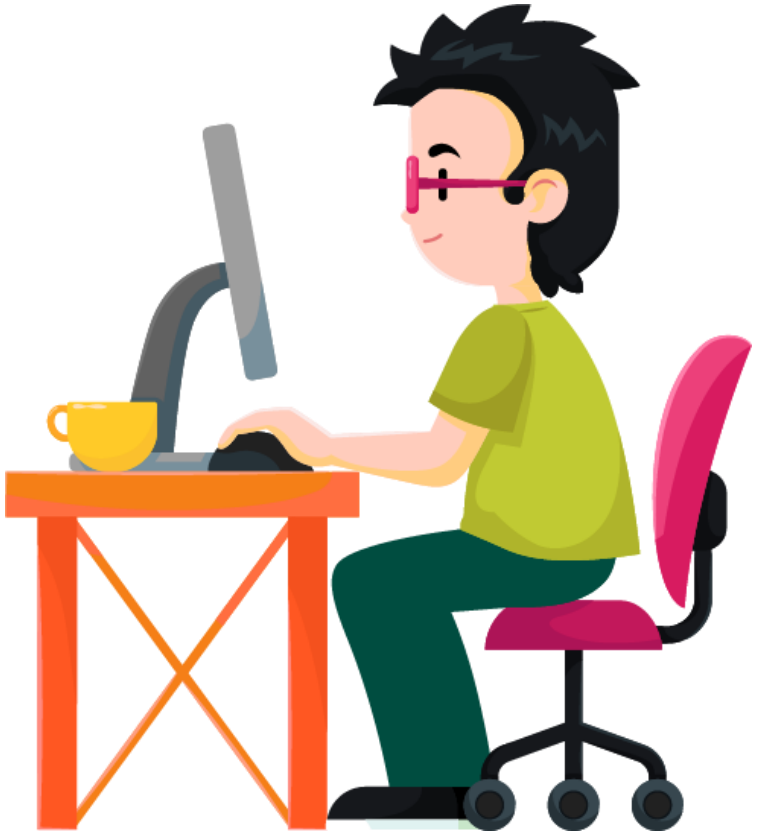
```
ans is :  60

Process finished with exit code 0
```

Explanation

- In the previous example, we have defined two sum method, but we can only use the second sum method, as python does not supports method overloading.
- We may define many method of same name and different argument but we can only use the latest defined method.
- Calling the other method will produce an error.

Get Exclusive Video Tutorials



www.aptutorials.com

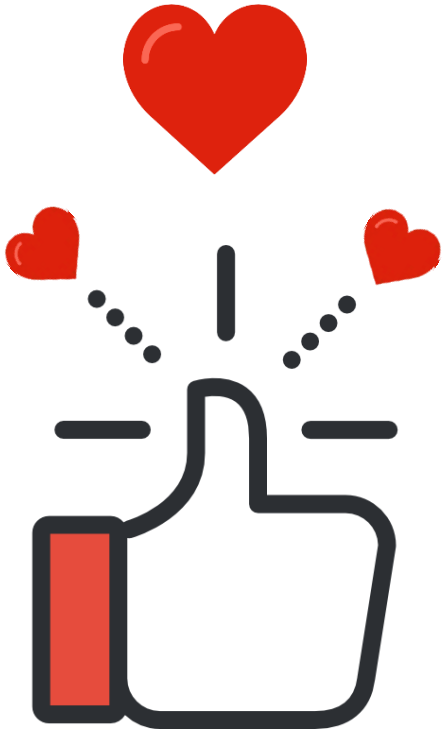
https://www.youtube.com/user/Akashtips

Get More Details

www.akashsir.com

If You Liked It !

Rating Us Now



Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET

Sulekha

https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad

Connect With Me

Akash Padhiyar
#AkashSir

www.akashsir.com
www.akashtechnolabs.com
www.akashpadhiyar.com
www.aptutorials.com

Social Info

Akash.padhiyar

Akashpadhiyar

Akash_padhiyar

+91 99786-21654

#Akashpadhiyar
#aptutorials