Python Object Oriented Programming (OOP) -

Python is a multiprogramming language as it supports different programing approach; in there most popular approach is to solve a programing problem by creating an object, this is known as object oriented programing lang.

There are 2 important terms of OOPs concept are - i) Class
ii) Object

i) Class - In python everything is an object. To create obj.s we require some model or plan or blueprint which is nothing but a class.

- We can write a class to represent some properties (attributes) & actions (behaviour) of an obj.

- Properties can be represented by variables. Actions can be represented by methods (func"s).

- Hence class can contain both variables & methods.

→
```
class mobile :          # plan
    def config(self):   # action
        print('RAM', 'ROM', 'MCP')

mob1 = mobile()         # constructor
mob2 = mobile()

mobile.config(mob1)     # class call
mobile.config(mob2)

mob1.config()           # obj call
mob2.config()
```

o/p →
```
RAM ROM MCP
RAM ROM MCP
RAM ROM MCP
RAM ROM MCP
```

- Within the python class we can represent data by using variables. There are 3 types of variables -
  i) Instance (obj. level)
  ii) Static (class level)
  iii) Local (method level)

Within the python class, we can represent operations by using methods i.e -

i) Instance method
ii) Class method
iii) Static method.

The bellow eg. demonstrates how to access the class & how to define methods & how to call a method by using class.

→
```
class student:
    def __init__(self):          # defining instance method
        self.name = 'Ram'
        self.roll_no = 201
        self.mark = 95

    def data(self):              # other method
        print("students name is :", self.name)
        print("The roll no is :", self.roll_no)
        print("The mark is :", self.mark)

s = student()                    # defining reference var.

print(s.name)                    # reference var.
print(s.roll_no)
print(s.mark)
s.data()
```

o/p → Ram
201
95

o/p → Student's name is : Ram
The roll no. is : 201
The mark is : 95

→ class student :
def __init__ (self, name, roll_no, mark) :
self.name = name
self.roll_no = roll_no
self.mark = mark
def data (self) :
print ("Student's name :", self.name)
print ("The roll no :", self.roll_no)
print ("The mark :", self.mark)

s1 = student ('Ram', 201, 85)
s2 = student ('Seeta', 301, 95)
s3 = student ('Lakkshman', 401, 75)
s4 = student ('Ravan', 501, 35)

s1.data()
s2.data()
s3.data()
s4.data()

o/p → Student's name : Ram
The roll no : 201
The mark : 85
Student's name : Seeta
The roll no : 301
The mark : 95

Students name : Lakshman
The roll no : 401
The mark : 75

Students name : Ravan
The roll no : 501
The mark : 35

self – A self is a parameter i.e reference to the current instance of the class. It is used to access variables that belongs to class.

Self is a default var. which is always pointing to the current obj. By using self we can access instance variables & instance methods of the obj. We should use the self within python class to refer the current obj.

* It does not have to be named as self, we can call it whatever we like, but it has to be 1st parameter of any func" in the class.

__init__ func" –

All the classes have a func" called __init__ i.e always executed when class is being initiated. The __init__ func" is called automatically everytime the class is being used to create a new obj. To use the __init__ func" to assign values to obj properties or other opera"s that are necessary to do when the obj. is being created.

Inheritance –

The method of inheriting the properties from parent class or super class into a child class or sub class is known as inheritance.

Advantages of inheritance –

- Code reusability
- It improves code reusability. We don't need to write same code over again.
- Using inheritance, we can inherite features of other classes & also add

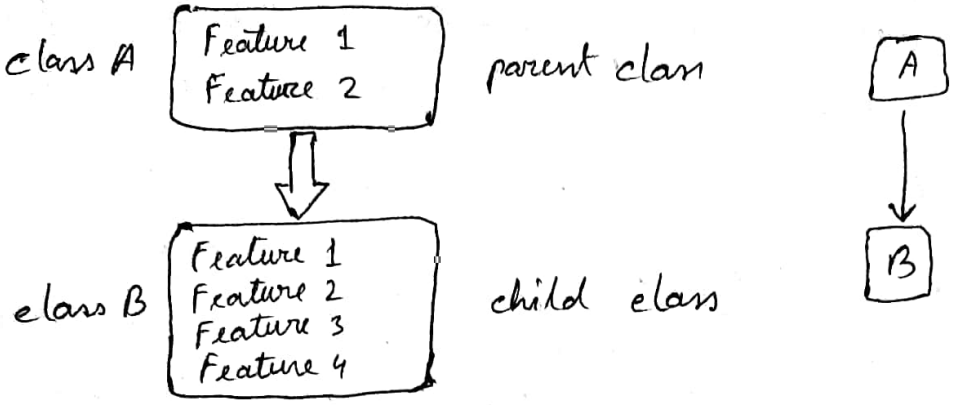
more features to derived class (reuse).
- Reduces the programmer's efforts. Programmer donot need to write same code & logic so it reduces efforts of coders.
- Readability - By implimenting concepts of inheritance, program looks more concyse & structured. That makes it ~~so~~ easy to read & understand. This way inheritance also improves readability of code.

Types of inheritance in python -

There are 5 types of inheritance in python. They are -
i) Single inheritance
ii) Multiple inheritance
iii) Multilevel inheritance
iv) Hierarchical inheritance
v) Hybrid inheritance

i) Single inheritance -

When 1 child class is derived from 1 parent class is called single inheritance.



→
```
class A :
    def feature1 (self) :
        print ('Feature 1')
        print ('Feature 2')
    def feature2 (self) :
        print ('Feature 2')
class B (A) :
    def feature3 (self) :
        print ('Feature 3')
    def feature4 (self) :
        print ('Feature 4')

a1 = A()
a1. feature1()
a1. feature2()

b1 = B()
b1. feature1()
b1. feature2()
b1. feature3()
b1. feature4()
```

o/p → Feature 1
Feature 2
Feature 1
Feature 2
Feature 3
Feature 4

→ class Brands :
    brand_name1 = 'Myntra'
    brand_name2 = 'Flipkart'
    brand_name3 = 'OLX'

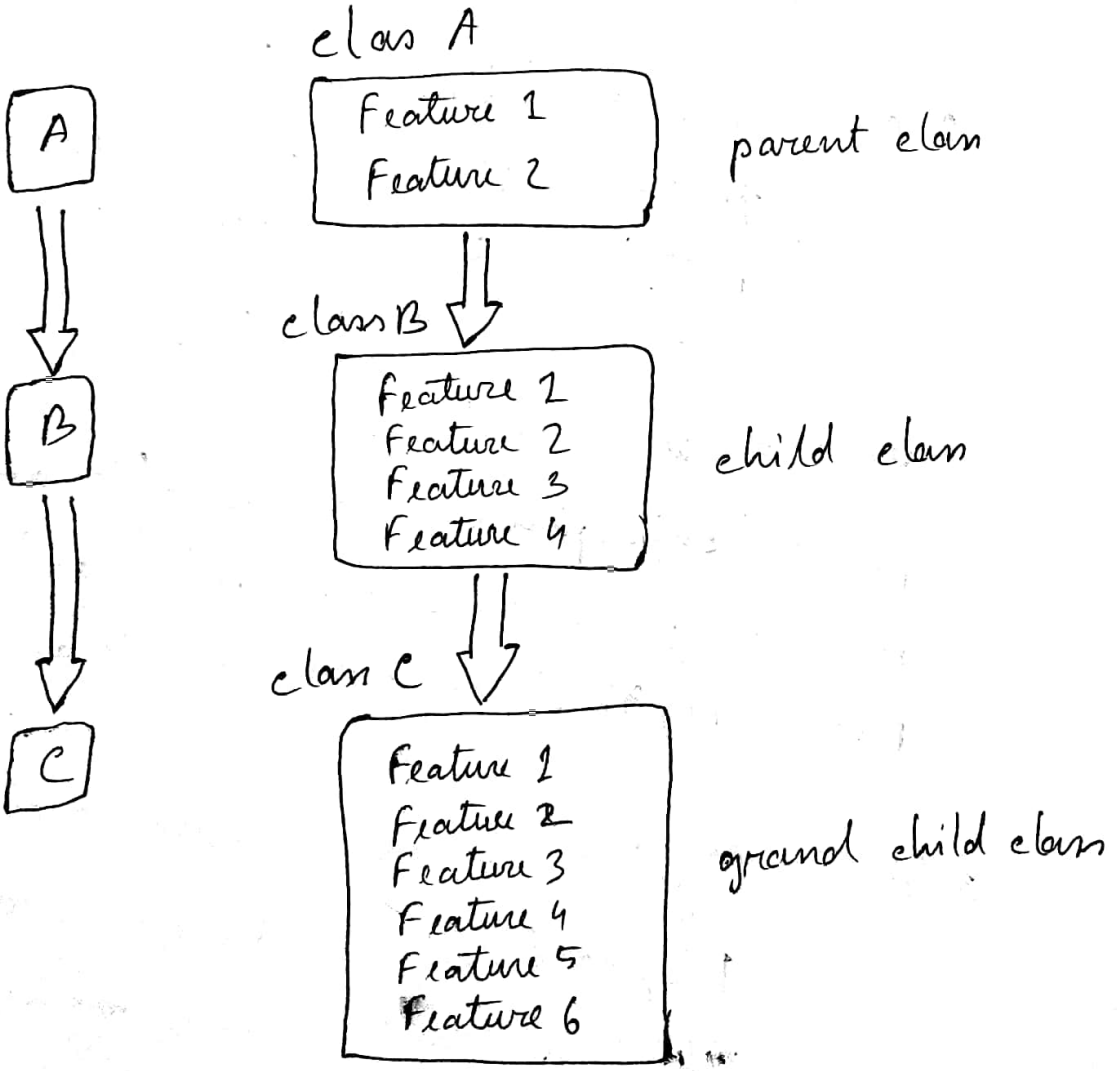
class Products (Brands):
    prod1 = 'Online cloth store'
    prod2 = 'Online ecomerce site'
    prod3 = 'Online buy sell store'

obj1 = Pproducts ()
print (obj1.brand_name1 + 'is an' + obj1.prod1)
print (obj1.brand_name2 + 'is an' + obj1.prod2)
print (obj1.brand_name3 + 'is an' + obj1.prod3)

o/p → Myntra is an Online cloth store
Flipkart is an Online ecomerce site
OLX is an online buy sell store.

iii) Multilevel Inheritance –

In this inheritance, we have one parent class & one child class that derived / inherited from parent class. We have a grand child class that is derived from the child class. This is known as multilevel inheritance.



→

```
class A:
    def feature1 (self):
        print ('Feature 1')
    def feature 2 (self):
        print ('Feature 2')

class B (A):
    def feature 3(self):
        print ('Feature 3')
    def feature 4 (self):
        print ('Feature 4')

class C (B):
    def feature 5(self):
        print ('Feature 5')
    def feature 6 (self):
        print ('Feature 6')

a1 = A()
b1 = B()
c1 = C()

c1. feature 1()
c2. feature 2()
c3. feature 3()
c4. feature 4()
c5. feature 5()
c6. feature 6()
```

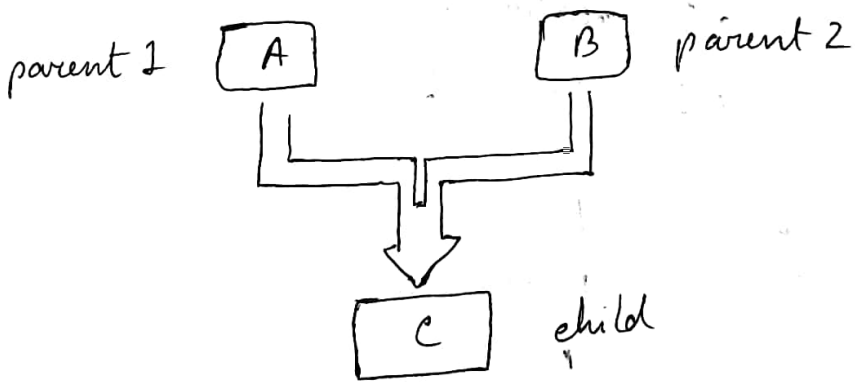
O/P →
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6

→
```
class Brands:
    brand_name1 = 'Myntra'
    brand_name2 = 'Flipkart'
    brand_name3 = 'OLX'

class Products (Brands):
    prod1 = 'Online cloth store'
    prod2 = 'Online ecomerce site'
    prod3 = 'Online buy sell store'

class Popularity (Products):
    prod1_pop = 95
    prod2_pop = 85
    prod3_pop = 55

obj1 = Popularity()
print (obj1.brand_name1 + 'is an' + obj1.prod1 +
    '& popularity of' + str (obj1.prod1_pop))
print(obj1.brand_name2 + 'is an' + obj1.prod2 +
    'popularity of' + str(obj1.prod2_pop))
print(obj1.brand_name3 + 'is an' + obj1.prod3 +
    'popularity of' + str (obj.prod3_pop))
```

o/p → Myntra is an Online cloth store popularity of 95.
Flipkart is an Online ecomerce site popularity of 85
OLX is an Online buy sell store popularity of 55

ii) Multiple Inheritance –

When child class is derived from more than 1 parent class is called multiple inheritance. In multiple inheritance, we have 2 parent classes (or more), one child class that inherits both parent class properties.



→
```
class A :
    def feature 1 (self):
        print ('Feature 1')
    def feature 2 (self):
        print ('Feature 2')

class B :
    def feature 3 (self):
        print ('Feature 3)'
    def feature 4 (self):
        print ('Feature 4)
```

```
class C(A, B):
    def feature5(self):
        print('Feature 5')
    def feature6(self):
        print('Feature 6')

c1 = C()
c1.feature1()
c1.feature2()
c1.feature3()
c1.feature4()
c1.feature5()
c1.feature6()
```

o/p →
```
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
```

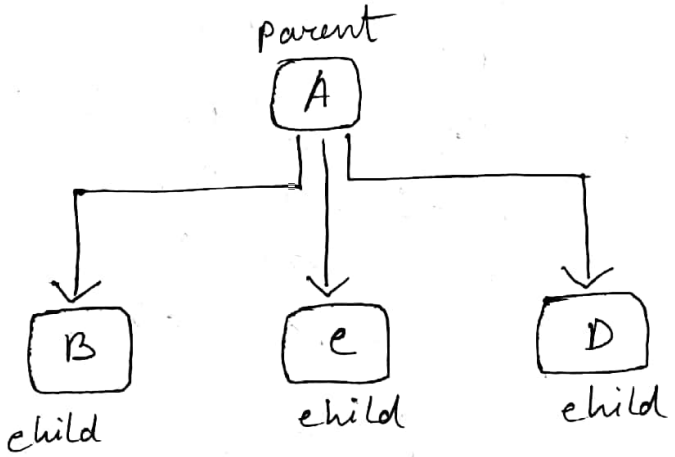
→ class Brands :

```
    brand1 = 'Myntra'
    brand2 = 'Flipkart'
    brand3 = 'OLX'
class Products :
    prod1 = 'Online cloths store'
    prod2 = 'Online ecomerce site'
    prod3 = 'Online buy sell store'
class Popularity (Brands, Products):
    prod1_pop = 95
    prod2_pop = 85
    prod3_pop = 65
obj1 = Popularity()
print(obj1.brand1 + " is an " + obj1.prod1 + " popularity of"
      + str(obj1.prod1_pop)) -
print(obj1.brand2 + " is an " + obj1.prod2 + " popularity of"
      + str(obj1.prod2_pop))
print(obj1.brand2 + "is an" + obj.prod3 + "popularity
      of" + str(obj.prod3_pop))
```

o/p → Myntra is an Online cloths store.
popularity of 95.
Flipkart is an Online ecomerce site
popularity of 85
OLX is an Online buy sell ~~site~~ store
popularity of 65

iv) Hierachical Inheritance –

When we derive or inherite more than one child class from a single parent class, it is called hierachical inheritance.



→
```
class A :
    def feature1(self):
        print ('Feature 1')
    def feature2 (self):
        print ('Feature 2')
class B (A):
    def feature 3 (self):
        print ('Feature 3')
    def feature 4 (self):
        print ('Feature 4')

class C (A):
    def feature 5 (self):
        print ('Feature 5')
    def feature 6 (self):
        print ('Feature 6')
```

```
class D(A):
    def feature 7(self):
        print('Feature 7')
    def feature 8(self):
        print('Feature 8').

a1 = feature 1 A()
b1 = B()
c1 = c()
d1 = D()

a1. feature 1
a1. feature 2
print('******')
b1. feature 1
b1. feature 2
b1. feature 3
b1. feature 4
print('*****')
c1. feature 1
c1. feature 2
c1. feature 35
c1. feature 6
print('*****')
d1. feature 1
d1. feature 2
d1. feature 7
d1. feature 8
```

O/P →

Feature 1
Feature 2
* * * * *
Feature 1
Feature 2
Feature 3
Feature 4
* * * * *
Feature 1
Feature 2
Feature 5
Feature 6
* * * * *
Feature 1
Feature 2
Feature 7
Feature 8

→

```
class Brands:
    brand1 = 'Myntra'
    brand2 = 'Flipkart'
    brand3 = 'OLX'.
class Products (Brands):
    prod1 = 'Online clothes store'
    prod2 = 'Online ~~economic~~ ecomerse site'
    prod3 = 'Online buy and sell store'
class Popularity (Brands):
    prod1_pop = 95
    prod2_pop = 85
    prod3_pop = 65
```

```
class Value(Brands):
    prod1_val = 'Excelent value'
    prod2_val = 'Better value'
    prod3_val = 'Good value'

obj1 = Products()
obj2 = Popularity()
obj3 = Value()

print(obj1.brand1 + 'is an' + obj1.prod1 + 'popularity of' +
      str(obj2.prod1_pop) + 'with' + obj3.prod1_val)
print(obj1.brand2 + 'is an' + obj1.prod2 + 'popularity of'
      + str(obj2.prod2_pop) + 'with' + obj3.prod2_val)
print(obj1.brand3 + 'is an' + obj1.prod3 + 'popularity of'
      + str(obj2.prod2_pop) + 'with' + obj.prod3_val)
```

o/p ⇒ Myntra is an Online cloths stor popularity of 95 with excellent value.
Flipkart is an Online ecomerce site popularity of 85 with better value
OLX is an Online buy sell store popularity of 65 with good value

→ class Employee :

company_name = "Google"

class Test_Engineer (Employee):

salary = ~~75,000~~ 75000

class Trainer (Employee):

salary = 100000

Ram = Test_Engineer

Seeta = Trainer

print ("Ram" + Ram.company_name + " salary is " + str (Ram.Salary))

print ("Seeta" + Seeta.company_name + " salary is " + str (Seeta.Salary))

o/p → Ram . Google salary is 75000

Seeta Google salary is 100000

→ class Car :

def __init__ (self, comp_name, Model, YOM):

self.comp_name = comp_name

self.Model = Model

self.YOM = YOM

~~def~~.

```
class BMW(Car):
    def display(self):
        print("Company name:", self.comp_name)
        print("Model name:", self.Model)
        print("Year of menufecture:" self.YOM)

class Audi(Car):
    def display(self):
        print("Company name:", self.comp_name)
        print("Model name:", self.Model)
        print("Year of menufecture:", self.YOM)

class TATA(Car):
    def display(self):
        print("Company name:", self.comp_name)
        print("Model name:", self.Model)
        print("Year of menufecture:", self.YOM)

e1 = BMW('BMW', 'X1', 2021)
e1.display()
print('*****')
e2 = Audi('Audi', 'Q7', 2020)
e2.display()
print('*****')
e3 = TATA('TATA', 'Nano', 2018)
e3.display()
```

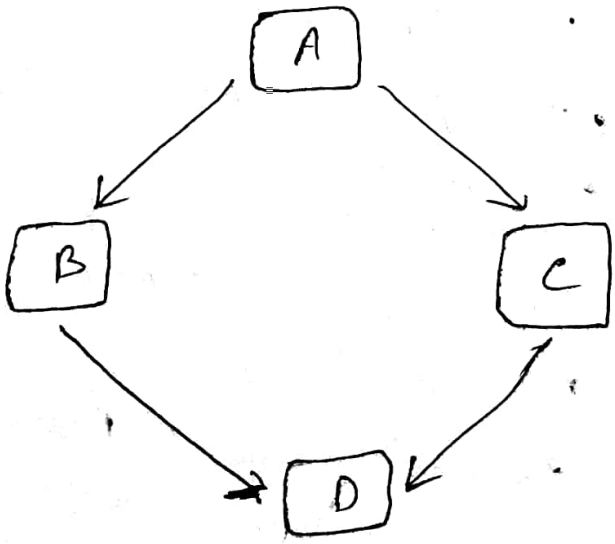
O/P →

```
Company name: BMW
Model name: X1
Year of menufacture: 2021
*****
Company name: Audi
Model name: Q7
Year of menufacture: 2020
```

Company name : TATA
Model name : Nano
Year of manufacture : 2018

Hybrid inheritance

This inheritance satisfies more than one form of inheritance. It may be consisting of all types of inheritance. Hybrid inheritance is the combination of single level, multilevel, multiple and hierachical inheritance. This type of inheritance is very helpful if we want to use concepts of inheritance without any limitations according to our requirement.



→
```
class PC:
    def fun1(self):
        print("This in PC class")

class Laptop(PC):
    def fun2(self):
        print("This is Laptop class inherited from
              PC class")

class Mouse(Laptop):
    def fun3(self):
        print("This is Mouse class inherited
              from Laptop class")

class Student(Mouse, Laptop):
    def fun4(self):
        print("This is student class
              inherited from Mouse & Laptop")

obj = Student()
obj.fun1()
obj.fun2()
obj.fun3()
obj.fun4()
```

O/P →
```
This in PC class
This in Laptop class inherited from PC class
This in Mouse class inherited from Laptop
  class
This is student class inherited from
 Mouse & Laptop
```

→ class Employee:

comp_name = "Test Yantra"

class Test_Engineer (Employee):

salary = 85000

class Trainer (Employee):

salary = 100000

class Python_Trainer (Trainer):

salary = 150000

class HR (Python_Trainer):

salary = 50000

class manager (HR, Trainer):

salary = 100000

```
Ram = Test_Engineer()
Seeta = Trainer()
Hanuman = Python_Trainer()
Ravan = HR()
Vishnu = manager():
print("Ram" + Ram.comp_name + "salary is " + str(Ram.salary))
print("Seeta" + Seeta.comp_name + "salary is " +
      str(Seeta.salary))
print("Hanuman" + Hanuman.comp_name + "salary is "
      + str(Hanuman.salary))
print("Ravan" + Ravan.comp_name + "salary is "
      + str(Ravan.salary))
print("Vishnu" + Vishnu.comp_name +
      "salary is " + str(Vishnu.salary))
```

o/p → Ram Test Yantra salary is 85000
Seeta Test Yantra salary is 100000
Hanuman Test Yantra salary is 150000
Ravan Test Yantra salary is 50000
Vishnu Test Yantra salary is 100000

→
```
class Car:
    def __init__(self, company, model, colour, YOM):
        self.company = company
        self.model = model
        self.colour = colour
        self.YOM = YOM

class Skoda(Car):
    def display(self):
        print("Company name:", self.company)
        print("Model:", self.model)
        print("Colour:", self.colour)
        print("Year of menufacture:", self.YOM)

class BMW(Car):
    def display(self):
        print("Company name:", self.company)
        print("Model:", self.model)
        print("Colour:", self.colour)
        print("Year of menufacture:", self.YOM)
```

```
class Hyundai(Skoda, BMW):
    def display(self):
        print("Company name:", self.company)
        print("Model:", self.model)
        print("Colour:", self.colour)
        print("Year of manufacture:", self.YOM)

class Maruti(Hyundai):
    def display(self):
        print("Company name:", self.company)
        print("Model:", self.model)
        print("Colour:", self.colour)
        print("Year of manufacture:", self.YOM)

c1 = Skoda('Skoda', 'Kushak', 'White', 2022)
c1.display()
print("*****")
c2 = BMW('BMW', 'X1', 'Red', 2020)
c2.display()
print("*****")
c3 = Hyundai('Hyundai', 'Creta', 'Black', 2021)
c3.display()
print("*****")
c4 = Maruti('Maruti', '800', 'Blue', 2003)
c4.display()
```

o/p → Company name: Skoda
Model: Kushak
Colour: White
Year of menufacture: 2022

* * * * *

Company name: BMW
Model: X1
Colour: Red
Year of menufacture: 2020

* * * * *

Company name: Hyundai
Model: Creta
Colour: Black
Year of menufacture: 2021

* * * * *

Company name: Maruti
Model: 800
Colour: Blue
Year of menufacture: 2003