Chap.- 9

Polymorphism

- **using same to perform different methods**

In [14]:

```
class A:
    def display(self, x):
        print("Display without parameter")

    def display(self, x, y):
        print("Display x :",x,"y :",y)

ob = A()
# ob.display()
#
ob.display(1,2) # Display x : 1 y : 2
ob.display(5, 1) # Display x : 5 y : 1
```

```
Display x : 1 y : 2
Display x : 5 y : 1
```

In [11]:

```
class A:
    def display(self, x):
        print("Display without parameter")

    def display(self, x, y):
        print("Display x :",x,"y :",y)

ob = A()
# ob.display()
#
ob.display(1,2) # Display x : 1 y : 2
ob.display(5) # TypeError: display() missing 1 required positional argument: 'y
```

```
Display x : 1 y : 2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-11-8bbd0f12908e> in <module>
     10 #
     11 ob.display(1,2) # Display x : 1 y : 2
---> 12 ob.display(5) # Display x : 5 y : None

TypeError: display() missing 1 required positional argument: 'y'
```

In [15]:
```
class A:
    def display(self):
        print("Display without parameter")

    def display(self, x, y=None):
        print("Display x :",x,"y :",y)

ob = A()
ob.display(1,2) # Display x : 1 y : 2
ob.display(5) # Display x : 5 y : None
```

```
Display x : 1 y : 2
Display x : 5 y : None
```

In [24]:
```
class A:
    def display(self, x, y=None):
        if(y == None):
            print(x)
        else:
            print(x, y)
ob = A()
# ob.display()
#
ob.display(1,2) # 1 2
ob.display(5) # 5
```

```
1 2
5
```

- **Can't support overloading...**
- **In python we create illusion of using overloading.**

In [19]:
```
class A:
    def __init__(self, x, y):
        self.x = x
        self.y = y

p1 = A(2,3)
p2 = A(10,20)
print(p1 + p2)
# TypeError: unsupported operand type(s) for +: 'A' and 'A'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-e58a3b6646e9> in <module>
      6 p1 = A(2,3)
      7 p2 = A(10,20)
----> 8 print(p1 + p2)
      9 # TypeError: unsupported operand type(s) for +: 'A' and 'A'

TypeError: unsupported operand type(s) for +: 'A' and 'A'
```

- **Operator Overloading**

In [22]:
```python
class A:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        x = self.x + other.x
        y = self.y + other.y
        return(x,y)

p1 = A(2,3)
p2 = A(10,20)
print(p1 + p2) # (12, 23)
```

```
(12, 23)
```

In [26]:
```python
class A:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        x = self.x + other.x
        y = self.y + other.y
        return(x,y)

p1 = A(2,3)
p2 = A(10,20)
print(p1 * p2) # TypeError: unsupported operand type(s) for *: 'A' and 'A'

# fun name je hoy te sign use karvi
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-6c8a3007c8d0> in <module>
     11 p1 = A(2,3)
     12 p2 = A(10,20)
---> 13 print(p1 * p2) # TypeError: unsupported operand type(s) for *: 'A' and 'A'

TypeError: unsupported operand type(s) for *: 'A' and 'A'
```

In [28]:
```python
class A:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        x = self.x * other.x
        y = self.y * other.y
        return(x,y)

p1 = A(2,3)
p2 = A(10,20)
print(p1 + p2) # (20, 60)
```

```
(20, 60)
```

- **add**
- **sub**
- **mul**
- **truediv (/)**
- **floordiv (//)**
- **mod (%)**
- **pow (**)**
- **lt (<)**
- **gt (>)**
- **le (<=)**
- **ge (>=)**
- **ne (!=)**
- **eq (==)**

In [35]:

```
class A:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        x = self.x + other.x
        y = self.y + other.y
        return(x,y)
    def __sub__(self, other):
        x = self.x - other.x
        y = self.y - other.y
        return(x,y)
    def __mul__(self, other):
        x = self.x * other.x
        y = self.y * other.y
        return(x,y)
    def __truediv__(self, other):
        x = self.x / other.x
        y = self.y / other.y
        return(x,y)
    def __floordiv__(self, other):
        x = self.x // other.x
        y = self.y // other.y
        return(x,y)
    def __mod__(self, other):
        x = self.x % other.x
        y = self.y % other.y
        return(x,y)
    def __pow__(self, other):
        x = self.x ** other.x
        y = self.y ** other.y
        return(x,y)
    def __lt__(self, other):
        x = self.x < other.x
        y = self.y < other.y
        return(x,y)
    def __gt__(self, other):
        x = self.x > other.x
        y = self.y > other.y
        return(x,y)
    def __le__(self, other):
        x = self.x <= other.x
        y = self.y <= other.y
        return(x,y)
    def __ge__(self, other):
        x = self.x >= other.x
        y = self.y >= other.y
        return(x,y)
    def __ne__(self, other):
        x = self.x != other.x
        y = self.y != other.y
        return(x,y)
    def __eq__(self, other):
        x = self.x == other.x
        y = self.y == other.y
        return(x,y)


p1 = A(2,3)
p2 = A(10,20)
print(p1 + p2) # (12, 23)
print(p1 - p2) # (-8, -17)
print(p1 * p2) # (20, 60)
print(p1 / p2) # (0.2, 0.15)
```

```
print(p1 // p2) # (0, 0)
print(p1 % p2) # (2, 3)
print(p1 ** p2) # (1024, 3486784401)
print(p1 < p2) # (True, True)
print(p1 > p2) # (False, False)
print(p1 <= p2) # (True, True)
print(p1 >= p2) # (False, False)
print(p1 != p2) # (True, True)
print(p1 == p2) # (False, False)
```

```
(12, 23)
(-8, -17)
(20, 60)
(0.2, 0.15)
(0, 0)
(2, 3)
(1024, 3486784401)
(True, True)
(False, False)
(True, True)
(False, False)
(True, True)
(False, False)
```

P.b. = 680

In [45]:

```
class St:
    def __init__(self, name, rn, age, marks):
        self.n = name
        self.r = rn
        self.a = age
        self.m = marks
    def display(self):
        print("Name :",self.n)
        print("Roll No. :",self.r)
        print("Age :",self.a)
        print("Marks :",self.m)

    def __eq__(self, other):
        if(self.m == other.m):
            print("Both marks are Same :)")
            return True
        else:
            print("Both marks are not Same")
            return False

s1 = St('Romil', 84, 18, 24)
s2 = St('Yash' , 94, 18, 25)
s3 = St('Rudra' , 90, 18, 24)

print(s1 == s2) # Both marks are not Same
print(s1 == s3) # Both marks are Same :)
```

```
Both marks are not Same
False
Both marks are Same :)
True
```

. Inheritance :

In [51]:
```
class A:
    def demo(self):
        print("Class A")
class B:
    def dis(self):
        print("Class B")

ob = B()
ob.demo()
ob.dis() # AttributeError: 'B' object has no attribute 'demo'
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-51-2092b482f89d> in <module>
      7
      8 ob = B()
----> 9 ob.demo()
     10 ob.dis() # AttributeError: 'B' object has no attribute 'demo'

AttributeError: 'B' object has no attribute 'demo'
```

• Single (Simple) Inheritance

In [54]:
```
class A:
    def demo(self):
        print("Class A")
class B(A):
    def dis(self):
        print("Class B")

ob = B()
ob.demo() # Class A
ob.dis() # Class B
```

```
Class A
Class B
```

In [56]:
```
class A:
    def demo(self):
        print("Class A")
class B(A):
    def dis(self):
        print("Class B")

ob = A()
ob.demo() # Class A
ob.dis() # Class B
```

```
Class A

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-56-0d7de25aaab0> in <module>
      8 ob = A()
      9 ob.demo() # Class A
---> 10 ob.dis() # Class B

AttributeError: 'A' object has no attribute 'dis'
```

P.b. 671

In [63]:
```
class Book():
    def __init__(self):
        self.name = input("Enter Name of Book : ")
        self.no = input("Enter No. of Book : ")
        self.a = input("Enter Name of Author : ")
        self.pub = input("Enter Name of Publiser : ")
        self.isbn = input("Enter ISBN : ")
        self.y = input("Enter Year : ")

    def display(self):
        print("Name :", self.name)
        print("No. :", self.no)
        print("Name of Author :", self.a)
        print("Publiser :", self.pub)
        print("ISBN :", self.isbn)
        print("Year :", self.y)

class TextBook(Book):
    def __init__(self):
        super().__init__()
        self.co = input("Enter Course : ")

    def display(self):
        super().display()
        print("Course : ", self.co)

# -----------------------------------------------------------
B1 = TextBook()
B1.display()
```

```
Enter Name of Book : sdsd
Enter No. of Book : 234
Enter Name of Author : dfh
Enter Name of Publiser : jhg
Enter ISBN : 5214
Enter Year : 2045
Enter Course : adfgad
Name : sdsd
No. : 234
Name of Author : dfh
Publiser : jhg
ISBN : 5214
Year : 2045
Course :  adfgad
```

- ## Types of Inheritance

  ### 1.) Single P-C

  ### 2.) Multiple 2P-C

  ### 3.) Multilevel

  ### 4.) Heirachical

  ### 5.) Hybrid

- # Multiple Inheritance

In [69]:
```python
class Person():
    def __init__(self):
        self.name = input("Enter Name : ")
        self.age = input("Enter Age : ")

class Car():
    def __init__(self):
        self.model = input("Enter Model : ")
        self.Color = input("Enter Color : ")

class Parking(Person, Car):
    def __init__(self):
        Person.__init__(self)
        Car.__init__(self)
        self.pn = input("Enter Parking No. : ")
    def display(self):
        print("------------Person Details----------------")
        print("Person Name :", self.name)
        print("Person Age :", self.age)
        print("------------Car Details----------------")
        print("Car Model :", self.model)
        print("Car Color :", self.Color)
        print("------------Parking Details----------------")
        print("Parking No. :", self.pn)

ob = Parking()
ob.display()

# Enter Name : Romil
# Enter Age : 18
# Enter Model : Mustang 1969
# Enter Color : Black
# Enter Parking No. : 8
# ------------Person Details----------------
# Person Name : Romil
# Person Age : 18
# ------------Car Details----------------
# Car Model : Mustang 1969
# Car Color : Black
# ------------Parking Details----------------
# Parking No. : 8
```

```
Enter Name : Romil
Enter Age : 18
Enter Model : Mustang 1969
Enter Color : Black
Enter Parking No. : 8
------------Person Details----------------
Person Name : Romil
Person Age : 18
------------Car Details----------------
Car Model : Mustang 1969
Car Color : Black
------------Parking Details----------------
Parking No. : 8
```

- **Method Resolution Order (Left to Right)**

In [ ]:
```python

```