

Chap.- 9

Polymorphism

- using same to perform different methods

```
In [14]: 1 class A:
2         def display(self, x):
3             print("Display without parameter")
4
5         def display(self, x, y):
6             print("Display x :",x,"y :",y)
7
8 ob = A()
9 # ob.display()
10 #
11 ob.display(1,2) # Display x : 1 y : 2
12 ob.display(5, 1) # Display x : 5 y : 1
```

Display x : 1 y : 2

Display x : 5 y : 1

```
In [11]: 1 class A:
2         def display(self, x):
3             print("Display without parameter")
4
5         def display(self, x, y):
6             print("Display x :",x,"y :",y)
7
8 ob = A()
9 # ob.display()
10 #
11 ob.display(1,2) # Display x : 1 y : 2
12 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]: 1 class A:
2         def display(self):
3             print("Display without parameter")
4
5         def display(self, x, y=None):
6             print("Display x :",x,"y :",y)
7
8 ob = A()
9 ob.display(1,2) # Display x : 1 y : 2
10 ob.display(5) # Display x : 5 y : None
```

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

```
In [24]: 1 class A:
2         def display(self, x, y=None):
3             if(y == None):
4                 print(x)
5             else:
6                 print(x, y)
7
8 ob = A()
9 # ob.display()
10 #
11 ob.display(1,2) # 1 2
12 ob.display(5) # 5
```

1 2
5

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

```
In [19]: 1 class A:
2         def __init__(self, x, y):
3             self.x = x
4             self.y = y
5
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                                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]:

```

1 class A:
2     def __init__(self, x, y):
3         self.x = x
4         self.y = y
5
6     def __add__(self, other):
7         x = self.x + other.x
8         y = self.y + other.y
9         return(x,y)
10
11 p1 = A(2,3)
12 p2 = A(10,20)
13 print(p1 + p2) # (12, 23)

```

(12, 23)

In [26]:

```

1 class A:
2     def __init__(self, x, y):
3         self.x = x
4         self.y = y
5
6     def __add__(self, other):
7         x = self.x + other.x
8         y = self.y + other.y
9         return(x,y)
10
11 p1 = A(2,3)
12 p2 = A(10,20)
13 print(p1 * p2) # TypeError: unsupported operand type(s) for *: 'A' and 'A'
14
15 # 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]:

```

1 class A:
2     def __init__(self, x, y):
3         self.x = x
4         self.y = y
5
6     def __add__(self, other):
7         x = self.x * other.x
8         y = self.y * other.y
9         return(x,y)
10
11 p1 = A(2,3)
12 p2 = A(10,20)
13 print(p1 + p2) # (20, 60)

```

(20, 60)

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

In [35]:

```
1 class A:
2     def __init__(self, x, y):
3         self.x = x
4         self.y = y
5
6     def __add__(self, other):
7         x = self.x + other.x
8         y = self.y + other.y
9         return(x,y)
10    def __sub__(self, other):
11        x = self.x - other.x
12        y = self.y - other.y
13        return(x,y)
14    def __mul__(self, other):
15        x = self.x * other.x
16        y = self.y * other.y
17        return(x,y)
18    def __truediv__(self, other):
19        x = self.x / other.x
20        y = self.y / other.y
21        return(x,y)
22    def __floordiv__(self, other):
23        x = self.x // other.x
24        y = self.y // other.y
25        return(x,y)
26    def __mod__(self, other):
27        x = self.x % other.x
28        y = self.y % other.y
29        return(x,y)
30    def __pow__(self, other):
31        x = self.x ** other.x
32        y = self.y ** other.y
33        return(x,y)
34    def __lt__(self, other):
35        x = self.x < other.x
36        y = self.y < other.y
37        return(x,y)
38    def __gt__(self, other):
39        x = self.x > other.x
40        y = self.y > other.y
41        return(x,y)
42    def __le__(self, other):
43        x = self.x <= other.x
44        y = self.y <= other.y
45        return(x,y)
46    def __ge__(self, other):
47        x = self.x >= other.x
48        y = self.y >= other.y
49        return(x,y)
50    def __ne__(self, other):
51        x = self.x != other.x
52        y = self.y != other.y
53        return(x,y)
54    def __eq__(self, other):
55        x = self.x == other.x
56        y = self.y == other.y
57        return(x,y)
58
59
60 p1 = A(2,3)
61 p2 = A(10,20)
62 print(p1 + p2) # (12, 23)
63 print(p1 - p2) # (-8, -17)
64 print(p1 * p2) # (20, 60)
65 print(p1 / p2) # (0.2, 0.15)
```

```

66 print(p1 // p2) # (0, 0)
67 print(p1 % p2) # (2, 3)
68 print(p1 ** p2) # (1024, 3486784401)
69 print(p1 < p2) # (True, True)
70 print(p1 > p2) # (False, False)
71 print(p1 <= p2) # (True, True)
72 print(p1 >= p2) # (False, False)
73 print(p1 != p2) # (True, True)
74 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]:

```

1 class St:
2     def __init__(self, name, rn, age, marks):
3         self.n = name
4         self.r = rn
5         self.a = age
6         self.m = marks
7     def display(self):
8         print("Name :",self.n)
9         print("Roll No. :",self.r)
10        print("Age :",self.a)
11        print("Marks :",self.m)
12
13    def __eq__(self, other):
14        if(self.m == other.m):
15            print("Both marks are Same :)")
16            return True
17        else:
18            print("Both marks are not Same")
19            return False
20
21 s1 = St('Romil', 84, 18, 24)
22 s2 = St('Yash' , 94, 18, 25)
23 s3 = St('Rudra' , 90, 18, 24)
24
25 print(s1 == s2) # Both marks are not Same
26 print(s1 == s3) # Both marks are Same :)

```

```

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

```

. Inheritance :

```
In [51]: 1 class A:
2         def demo(self):
3             print("Class A")
4 class B:
5         def dis(self):
6             print("Class B")
7
8 ob = B()
9 ob.demo()
10 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]: 1 class A:
2         def demo(self):
3             print("Class A")
4 class B(A):
5         def dis(self):
6             print("Class B")
7
8 ob = B()
9 ob.demo() # Class A
10 ob.dis() # Class B
```

Class A
Class B

```
In [56]: 1 class A:
2         def demo(self):
3             print("Class A")
4 class B(A):
5         def dis(self):
6             print("Class B")
7
8 ob = A()
9 ob.demo() # Class A
10 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]: 1 class Book():
2         def __init__(self):
3             self.name = input("Enter Name of Book : ")
4             self.no = input("Enter No. of Book : ")
5             self.a = input("Enter Name of Author : ")
6             self.pub = input("Enter Name of Publisier : ")
7             self.isbn = input("Enter ISBN : ")
8             self.y = input("Enter Year : ")
9
10        def display(self):
11            print("Name :", self.name)
12            print("No. :", self.no)
13            print("Name of Author :", self.a)
14            print("Publisier :", self.pub)
15            print("ISBN :", self.isbn)
16            print("Year :", self.y)
17
18        class TextBook(Book):
19            def __init__(self):
20                super().__init__()
21                self.co = input("Enter Course : ")
22
23            def display(self):
24                super().display()
25                print("Course :", self.co)
26
27        # -----
28        B1 = TextBook()
29        B1.display()
```

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

```

1 class Person():
2     def __init__(self):
3         self.name = input("Enter Name : ")
4         self.age = input("Enter Age : ")
5
6 class Car():
7     def __init__(self):
8         self.model = input("Enter Model : ")
9         self.Color = input("Enter Color : ")
10
11 class Parking(Person, Car):
12     def __init__(self):
13         Person.__init__(self)
14         Car.__init__(self)
15         self.pn = input("Enter Parking No. : ")
16     def display(self):
17         print("-----Person Details-----")
18         print("Person Name :", self.name)
19         print("Person Age :", self.age)
20         print("-----Car Details-----")
21         print("Car Model :", self.model)
22         print("Car Color :", self.Color)
23         print("-----Parking Details-----")
24         print("Parking No. :", self.pn)
25
26 ob = Parking()
27 ob.display()
28
29 # Enter Name : Romil
30 # Enter Age : 18
31 # Enter Model : Mustang 1969
32 # Enter Color : Black
33 # Enter Parking No. : 8
34 # -----Person Details-----
35 # Person Name : Romil
36 # Person Age : 18
37 # -----Car Details-----
38 # Car Model : Mustang 1969
39 # Car Color : Black
40 # -----Parking Details-----
41 # 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 []:

1

