Python Method Overriding in Hindi

Python - Python Method Overriding

Method Overriding ये OOP का एक हिस्सा है | Method Overriding का इस्तेमाल inheritance में किया जाता है | इसमे एक ही नाम के function को अलग-अलग class में अलग-अलग definition दी जाती है | Method Overriding में function को override किया जाता है |

Syntax for Method Overriding

```
class Base:
 def func(self):
 func_body
class Derived(Base):
 def func(self):
 func_body
```

Example for Method Overriding

Example पर func() इस function को override किया गया है |

Source Code :

```
class MyClass1:
    def func(self):
        a = 10
        print("Value of a :",a)

class MyClass2(MyClass1):
    def func(self):
        b = 5
        print("Value of b :",b)

obj = MyClass2()
obj.func()
```

Output :

```
Value of b : 5
```

Try to Different Type of Method Overriding in Python

अगर आप C++ या Java का Method Overriding पढ़े होंगे तो निचे दिया हुआ Method Overriding का example बिलकुल support करता है |
लेकिन इस प्रकार का example; python में support नहीं करता है |

C++ या Java में अगर overriding में एक ही function का type अलग-अलग हो तो उस type का function; class पर ढूंढ के execute करता है |
लेकिन Python में एक ही function का अलग-अलग type हो तो, जो आखिरी derived class पर function का type होता है वही type execute किया जाता है |

Source Code :

```
class MyClass1:
    def func(self):
        a = 10
        print("Value of a :",a)
class MyClass2(MyClass1):
    def func(self, b):
        self.b = b
        print("Value of a :",self.b)

obj = MyClass2()
obj.func(20)
obj.func()
```

Output :

```
Value of a : 20
    obj.func()
TypeError: func() missing 1 required positional argument: 'b'
```

Try to Different Type of Method Overriding in Python

ऊपर दिए हुए example में और यह example में फर्क यह है कि,
सिर्फ MyClass1 और MyClass2 की positions को change किया गया है | MyClass1 ये derived class है और MyClass2 ये base class है |

Source Code :

```
class MyClass2:
    def func(self, b):
        self.b = b
        print("Value of a :",self.b)
class MyClass1(MyClass2):
    def func(self):
        a = 10
        print("Value of a :",a)

obj = MyClass1()
obj.func()
obj.func(20)
```

Output :

```
Value of a : 10
    obj.func(20)
TypeError: func() takes 1 positional argument but 2 were given
```