Python Operator Overloading in Hindi

hindilearn.in/tut/python/python-operator-overloading-in-hindi

Python - Python Operator Overloading

Python में किसी विशिष्ट Operator का मतलब या उसके बर्ताव को बदलने के लिए Operator Overloading किया जाता है |

जैसे कि, अभीतक देखा है कि '+' Operator से दो Numbers का addition, दो string का concatenate और दो sequences(list,tuple) का addition/merge देखा है , लेकिन
क्या दो class objects को '+' operator से add किया जा सकता है ? Example देखिये |

Trying to add Two Objects Without Overloading

Example पर देखे तो Operator Overloading के सिवाय दो class objects को add करने का प्रयास किया गया है | लेकिन A class के objects unsupported operands type होने के कारण 'TypeError' का exception raise होता है |

Output :

```
5
2
    print(obj1 + obj2)
TypeError: unsupported operand type(s) for +: 'A' and 'A'
```

अगर दो objects का addition करना हो तो Operator Overloading का इस्तेमाल करना पड़ता है |

Example for '+' Operator Overloading in Python

Operator Overloading में हर Operator के लिए अलग-अलग function का इस्तेमाल किया जाता है | जैसे कि,
निचे दिए गए example में '+' operator से दो objects को add करने के लिए '__add__()' function का इस्तेमाल किया गया है |

Source Code :

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

    def disp(self):
        return self.a

    def __add__(self, param):
        return A(self.a + param.a)

obj1 = A(5)
obj2 = A(2)

c = obj1 + obj2
print(c.disp())
```

Output :

```
7
```

Functions for Operator Overloading

Operator	Function	Meaning
+	__add__()	Addition
-	__sub__()	Subtraction
*	__mul__()	Multiplication
/	__truediv__()	Division
%	__mod__()	Modulus
//	__floordiv__()	Floor Division
**	__pow__()	Exponent
<	__lt__()	Less than
<=	__le__()	Less than or Equal to
>	__gt__()	Greater than
>=	__ge__()	Greater than or Equal to
==	__eq__()	Equal to
!=	__ne__()	Not Equal to
<<	__lshift__()	Left Shift

>>	__rshift__()	Right Shift
&	__and__()	Bitwise AND
|	__or__()	Bitwise OR
^	__xor__()	Bitwise XOR
~	__invert__()	Bitwise NOT
index	__getitem__(self, index)	Index
str	__str__()	String
len	__len__()	Length

Other Example for Operator Overloading

Source Code :

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

    def disp(self):
        return self.a

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

obj1 = A(5)
obj2 = A(2)

c = obj1 + obj2                #or obj1.__add__(obj2)
print("+ Operator = ",c.disp())
c = obj1 - obj2                #or obj1.__sub__(obj2)
print("- Operator = ",c.disp())
c = obj1 * obj2                #or obj1.__mul__(obj2)
print("* Operator = ",c.disp())
c = obj1 / obj2                #or obj1.__truediv__(obj2)
print("/ Operator = ",c.disp())
c = obj1 % obj2                #or obj1.__mod__(obj2)
print("% Operator = ",c.disp())
c = obj1 // obj2               #or obj1.__floordiv__(obj2)
print("// Operator = ",c.disp())
c = obj1 ** obj2               #or obj1.__pow__(obj2)
print("** Operator = ",c.disp())
c = obj1 < obj2                #or obj1.__lt__(obj2)
print("< Operator = ",c.disp())
c = obj1 <= obj2               #or obj1.__le__(obj2)
print("<= Operator = ",c.disp())
c = obj1 > obj2                #or obj1.__gt__(obj2)
print("> Operator = ",c.disp())
c = obj1 >= obj2               #or obj1.__ge__(obj2)
print(">= Operator = ",c.disp())
c = obj1 == obj2               #or obj1.__eq__(obj2)
print("== Operator = ",c.disp())
c = obj1 != obj2               #or obj1.__ne__(obj2)
print("!= Operator = ",c.disp())
```

Output :

```
+ Operator =  7
- Operator =  3
* Operator =  10
/ Operator =  2.5
% Operator =  1
// Operator =  2
** Operator =  25
< Operator =  False
<= Operator =  False
> Operator =  True
>= Operator =  True
== Operator =  False
!= Operator =  True
```