EASYLEARNING.GURU

Operator Overloading

Operator Overloading

Operator Overloading allows same operator to have different meaning according to the context .

Example of overloading '+' operator.

```
>>>1+2                #sum
3
>>>'sun' +'set'       #concatenation
'sunset'
```

We can also overload the built in operators to make them work as we want. Python provide special methods for that. Example: addition, slicing, printing , and so on.

Main ideas of Operator Overloading

- Classes may override most built in type operations.
- Operators allow classes to integrate with python's object model.
- Methods named with double underscores (__X__) are special hooks.
- There are no defaults for operator overloading methods and none are required.
- Such methods are called automatically when instances appear in built-in operations.

Operator Overloading methods

EASYLEARNING.GURU

Method	Implements	Called for
__init__	Constructor	Object creation: X = class_name(args)
__add__	Operator +	X + Y , X += Y
__or__	Operator | (bitwise or)	X | Y , X | = Y
__repr__ , __str__	Printing	Print(X) , repr(X) , str(x)
__getattr__	Attribute fetch	X.Any
__setattr__	Attribute assignment	X.any = value
__delattr__	Attribute deletion	del X.any
__len__	Length	len(X)
__lt__ , __gt__	Comparisons	X < Y , X > Y
__bool__	Boolean tests	bool(X)
__contains__	Membership test	Item in X(any iterable)

Hands-On

Scope in Python

EASYLEARNING.GURU

```
# file scope.py
X = 11                  #Global name
def f():
    print X             #Access local
def g():
    X=22                #local variable
    print X
class C :
    X = 33              #class attribute
    def m(self):
        X = 44          #local variable in method
        self.X = 55     #instance attribute
```

```
print X             #11:module
f()                 #11:global
g()                 #22:local
print X             #11:module
obj = C()           #make instance
print obj.X         #33:class name inherited by
                        instance
obj.m()             #Attach attribute name X to
                        instance
print obj.x         #55 : instance
print C.X           #33:class
#print(C.m.X)       #FAILS: only visible in
                        method
#print(g.X)         #FAILS: only visible in
                        function
```

Hands-On