Operadores de comparação

Juliana Pirolla - Revisão

Podemos definir operadores de comparação nas classes usando:

Operação	Tradução
`a == b`	`a.__eq__(b)`
`a != b`	`a.__ne__(b)`
`a < b`	`a.__lt__(b)`
`a > b`	`a.__gt__(b)`
`a <= b`	`a.__le__(b)`
`a >= b`	`a.__ge__(b)`

Como exemplo prático, podemos definir uma classe qualquer e implementar tais operadores chamando os métodos mágicos de comparação descritos acima.

```
class Operadores:
    def __init__(self, x):
        self._val = x

    def __eq__(self, other):
        return self._val == other._val

    # def __ne__(self, other):
        # return self._val != other._val

    def __le__(self, other):
        return self._val <= other._val

    def __lt__(self, other):
        return self._val < other._val

    # def __ge__(self, other):
        # return self._val >= other._val
```

```
    # def __gt__(self, other):
        # return self._val > other._val
```

Note que definimos todos os possíveis só para treinar a estrutura, mas alguns métodos (os comentados) não precisam ser definidos posto que o Python compreende inversão de ordem, ou seja, uma vez definimos o operador $>$ e sabemos que `a > b`, automaticamnete sabemos também que `b < a`. O mesmo vale para o processo de negação.

Apesar desses pontos, **precisamos** definir os operadores que indicam complementariedade, ou seja, apenas de $<=$ ser equivalente à not $<$, precisamos definir ao menos um operador $<=$ ou $>=$ para realizar esse tipo de comparação.

```
c1 = Operadores(5)
c2 = Operadores(10)
```

```
print(c1 < c2)
print(c1 <= c2)
print(c1 != c2)
print(c1 == c2)
print(c2 < c1)
```

```
True
True
True
False
False
```

Assim, para executar as operações entre os objetos c1 e c2, foram chamados os métodos que definimos anteriormente na classe `Operadores`.

Em geral, precisamos apenas definir operadores de $==$, $<=$ e $<$.

Total order

Um conjunto possui ordem total se todos os seus elementos podem ser comparados de forma consistente, ou seja, se houver um operador de ordenação $\mathcal{S}$ se, para quaisquer $a, b, c \in \mathcal{S}$ temos:

- $a \preceq a$;
- Se $a \preceq b$ e $b \preceq c$, então $a \preceq c$;
- Se $a \preceq b$ e $b \preceq a$, então $a = b$;
- $a \preceq b$ ou $b \preceq a$.

Decorador @total_ordering

Se os objetos da classe formam um conjunto com ordem total, então podemos recorrer ao decorador @total_ordering do módulo functools e definir apenas doi operadores - O método `__eq__` - Mais apenas **um** dos seguinte métodos: `__lt__`, `__le__`, `__gt__`, or `__ge__`. Os outros métodos serão definidos apropriadamente de forma automática.

Note que ele criará apenas os métodos de comparação que **ainda não foram definidos** e pode não ser a mais eficiente.

Vamos refazer a classe definida anteriormente usando isso.

```
from functools import total_ordering

@total_ordering
class Operadores:
    def __init__(self, x):
        self._val = x

    def __eq__(self, other):
        return self._val == other._val

    def __le__(self, other):
        return self._val <= other._val
```

```
a1 = Operadores(5)
a2 = Operadores(10)
print(a1 < a2)
print(a1 >= a2)
print(a1 == a2)
```

```
True
False
False
```