České vysoké učení technické v Praze FIT

Programování v Pythonu

Jiří Znamenáček

Příprava studijního programu Informatika je podporována projektem financovaným z Evropského sociálního fondu a rozpočtu hlavního města Prahy.

Praha & EU: Investujeme do vaší budoucnosti

Python – Třídění a porovnávání

Úvod

Již jsme se říkali, jak se v Python'u třídí vestavěné typy. Také jsme si řekli, že je-li dostupná „magická metoda“ `__lt__()`, je ke třídění pomocí funkce *sorted()* použita právě ona.

Jelikož nyní již umíme pracovat s objekty a víme něco o jejich „magických metodách“, podíváme se, jak našim vlastním objektům naimplementovat možnost jejich řazení podle zvolených kritérií.

→ Použité příklady upraveny podle dokumentace.

__lt__()

Za porovnávání instancí objektů pomocí funkce *sort()* je tedy zodpovědná „magická metoda“ `OBJEKT.__lt__(self, other)`. Parametr *self* odpovídá aktuální instanci objektu, druhý parametr *other* je pak vstupem pro druhou instanci, s níž dochází k porovnání.

→ V Python'u 2.x tuto pozici zastávala standardní porovnávací funkce *cmp(a, b)*, resp. odpovídající magická metoda *__cmp__()*.

Vlastní implementace musí vracet logickou hodnotu na základě porovnání nějakých odpovídajících si vlastností obou objektů, v nejjednodušším případě tedy např. dvou stejných atributů:

```
def __lt__(self, other):
    return self.ATRIBUT < other.ATRIBUT
```

→ Pro metodu *__lt__()*, tedy „menší než“ (*less than*), jde pochopitelně o odpovídající porovnání `<` .

Příklad I

Definujme si vlastní třídu pro ukládání informací o žácích:

```
class Student:

    def __init__(self, jméno, známka, věk):
        self.jméno = jméno
        self.známka = známka
        self.věk = věk

    def __repr__(self):
        return repr((self.jméno, self.známka, self.věk))
```

Pokud nyní zavedeme *seznam* několika žáků s příslušnými údaji..

```
studenti = [
    Student('John', 'A', 15),
    Student('Jane', 'B', 12),
    Student('Dave', 'B', 10),
]
```

..přímo setřídit je nemůžeme, protože daná třída pro to neposkytuje žádnou podporu a její instance se tudíž hlásí jako *netříditelné*:

```
>>> sorted(studenti)
Traceback (most recent call last):
  File "<pyshell#43>", line 1, in <module>
    sorted(studenti)
TypeError: unorderable types: Student() < Student()
```

Příklad II

Upravme nyní tuto třídu, tak aby třídění umožňovala:

```
class Student:

    def __init__(self, jméno, známka, věk):
        self.jméno = jméno
        self.známka = známka
        self.věk = věk

    def __repr__(self):
        return repr((self.jméno, self.známka, self.věk))

    def __lt__(self, jiný_student):
        return self.věk < jiný_student.věk
```

Nyní při stejném *seznamu* žáků s příslušnými údaji..

```
studenti = [
    Student('John', 'A', 15),
    Student('Jane', 'B', 12),
    Student('Dave', 'B', 10),
]
```

..už třídění provést můžeme – žáci se setřídí podle svého věku:

```
>>> studenti
[('John', 'A', 15), ('Jane', 'B', 12), ('Dave', 'B', 10)]

>>> sorted(studenti)
[('Dave', 'B', 10), ('Jane', 'B', 12), ('John', 'A', 15)]
```

Porovnávání

I. Zadefinování metody *__lt__()* umožňuje kromě třídění objektů pomocí vestavěné funkce `sorted()` též jejich porovnávání pomocí operátoru `<` a jemu odpovídající funkce *operator.lt()* modulu *operator*:

```
>>> Student('John', 'A', 15) < Student('Jane', 'B', 12)
False
>>> Student('Jane', 'B', 12) < Student('John', 'A', 15)
True

>>> import operator
>>> operator.lt( Student('John', 'A', 15), Student('Jane', 'B', 12
False
>>> operator.lt( Student('Jane', 'B', 12), Student('John', 'A', 15
True
```

II. Potřebujete-li své objekty nejen třídit, ale též uvedeným způsobem porovnávat, musíte dodefinovat i další porovnávací magické metody. Následující tabulka uvádí přehled jejich volání při příslušných operacích mezi objekty `x` a `y`:

operátor	metoda
`x < y`	`x.__lt__(y)`
`x <= y`	`x.__le__(y)`
`x == y`	`x.__eq__(y)`
`x != y`	`x.__ne__(y)`
`x > y`	`x.__gt__(y)`
`x >= y`	`x.__ge__(y)`

- Jména metod v modulu *operator* jsou nepřekvapivě obdobná.
- Konvenčně vracejí porovnávací metody přímo booleovské hodnoty `True` nebo `False`, ale místo nich mohou vracet cokoliv, co bude dále jako pravdivostní hodnota vyhodnoceno.