Comprehensions

Dominik Pataky

20. November 2017

Python-Kurs 2017

Gliederung

1. Basics

2. List Comprehension

3. Dict Comprehension

4. Generators

Basics

Basics

Comprehensions sind eine bequeme Art und Weise, um Funktoren (Datenstrukturen, die andere Datenstrukturen beinhalten) mit kleinen Expressions zu erstellen und zu füllen und sind in allen modernen Sprachen vorhanden.

List Comprehension

List Comprehension

Grundlegender Syntax: `[ EXPRESSION for LAUFVARIABLE in ITERABLE (if FILTER) ]`

EXPRESSION Ist ein beliebiger Ausdruck (man stelle sich ein implizites `return` vor), etwa ein Wert, eine Variable, eine Gleichung, etc ...
`EXPRESSION` wird am Ende in der Liste abgelegt.

LAUFVARIABLE Eine beliebige Variable, die in *EXPRESSION* und *FILTER* zur Verfügung steht

ITERABLE Ist häufig etwas wie `range()` oder eine andere Liste.

FILTER Eine optionale boolean expression, womit Einträge gefiltert werden (falls `False`). Nützlich, wenn z.B. nur gerade Zahlen übernommen werden sollen, usw...

List Comprehension - Beispiel

```
1 varList = [var*8 for var in range(10)]
2 # => [0, 8, 16, 24, 32, 40, 48, 56, 64, 72]
3
4
5 # Mit Filter (hier: Fuer i ist gerade)
6 evenVarList = [var*8 for var in range(10) if var % 2 == 0]
7 # => [0, 16, 32, 48, 64]
```

Dict Comprehension

Dict Comprehension

Grundlegender Syntax: `{ KEY : VALUE for LAUFVARIABLE in ITERABLE (if FILTER) }`

Fast der gleiche Syntax, nur dieses Mal mit 2 Expressions: *KEY* und *VALUE*. Ansonsten gelten die gleichen Regeln.

Dict Comprehension - Beispiel

```
liste = ["Fritz", "Alex", "Nadine", "Peter", "Anna"]

names = {key: len(key) for key in liste}
# => {'Peter': 5, 'Fritz': 5, 'Alex': 4, 'Anna': 4, 'Nadine': 6}
```

Generators

Generators

Generator Ein Objekt, über das iteriert werden kann. Wenn ein Element daraus verwendet wurde, ist es nicht mehr in dem Generatorobjekt enthalten.

Die grundlegende Syntax ist gleich der einer *List Comprehension*. Da sich `list` und `dict` auch aus Iterables bauen lassen, gilt prinzipiell:

```
list(EXPRESSION for VARIABLE in ITERABLE) ==
[EXPRESSION for VARIABLE in ITERABLE]
```

und

```
dict((KEY, VALUE) for VARIABLE in ITERABLE) ==
{KEY:VALUE for VARIABLE in ITERABLE}
```

Aber: Generators verhalten sich anders als Lists oder Dicts!