HANSER

Erratum zu Abschnitt 29.4.2

Bernd Klein

Einführung in Python 3

Für Ein- und Umsteiger

Print-ISBN: 978-3-446-44133-0
E-Book-ISBN: 978-3-446-44151-4

Anbei finden Sie die korrigierte Version von Abschnitt 29.4.2 (S. 366 und 367).
Die geänderten Passagen sind gelb markiert.

```
red
rde
erd
edr
dre
der
permutations of the letters of the string "bin"
bin
bni
ibn
inb
nbi
nib
```

29.4.2 Variationen und Kombinationen

Wählt man aus einer n-elementigen Menge von Objekten k Objekte unter Berücksichtigung der Reihenfolge ohne Zurücklegen aus, so bezeichnet man dies als eine Variation ohne Wiederholung. Die Permutationen ergeben sich als ein Sonderfall für n = k.

Die Anzahl der verschiedenen k-Variationen von n Elementen lässt sich wie folgt berechnen:

$$\frac{n!}{(n-k)!} = n \cdot (n-1) \cdot (n-2) ... (n-k+1)$$

Der folgende Generator „variations" liefert alle k-Variationen ohne Wiederholungen:

```
def variations(items, k):
    if k==0:
        yield []
    else:
        for item in items:
            for v in variations(items, k-1):
                if item not in v:
                    yield [item] + v

for variation in variations(["a", "b", "c"], 2):
    print(variation)
```

Das Ergebnis lautet:

```
['a', 'b']
['a', 'c']
['b', 'a']
['b', 'c']
['c', 'a']
['c', 'b']
```

Spielt die Reihenfolge bei der Auswahl ohne Zurücklegen keine Rolle, so bezeichnet man dies als eine Kombination ohne Wiederholung. Bei der Lotterie 6 aus 49 handelt es sich beispielsweise um eine solche Kombination.

Die Anzahl der verschiedenen k-Kombinationen von n Elementen berechnet sich wie folgt:

$$\binom{n}{k} = \binom{n}{n-k} = \frac{n!}{k! \cdot (n-k)!} = \frac{n \cdot (n-1) \cdot (n-2) \ldots (n-k+1)}{k!}$$

Der folgende Generator „combinations" liefert alle k Kombinationen aus „objects" ohne Zurücklegen:

```
def combinations(objects, k):
    object = list(objects)
    if objects == [] or len(objects) < k or k == 0:
        yield []
    elif len(objects) == k:
        yield objects
    else:
        for combination in combinations(objects[1:], k-1):
            yield [objects[0]] + combination
        for combination in combinations(objects[1:], k):
            yield combination

combs = combinations(["a", "b", "c"], 2)
for c in combs:
    print(c)
```

Das Ergebnis lautet:

```
['a', 'b']
['a', 'c']
['b', 'c']
```

Falls Sie allerdings nur an einem Lottogewinn interessiert sind, können Sie auch die Funktion sample aus dem Modul random benützen.

```
random.sample(population, k)
```

population ist eine Sequenz (z.B. Liste), aus der k Elemente ausgewählt und in einer neuen Liste zurückgegeben werden. Dabei wird „population" aber nicht verändert.

```
>>> import random
>>> print(random.sample(range(1,50),6))
[19, 10, 30, 2, 14, 41]
```

Obige Zahlen können Sie gerne als Tipp für Ihren Lottoschein verwenden. Vergessen Sie bitte nicht den Autor dieses Buches, sollten Sie einen Sechser haben ☺.

29.5 Generatoren zähmen

Eine deutlich elegantere Methode besteht jedoch darin, einen Generator zu verwenden, der die ersten n Elemente eines anderen Generators ausgibt. Wir schreiben einen solchen Generator, den wir firstn nennen. Als Parameter erhält er einen Generator g und eine Anzahl n: