Listen in Python

October 15, 2017

1 Listen

Bislang haben wir in einer Variable jeweils nur ein Element (Zahl oder String) gespeichert.

```
In [1]: student1 = "Max"
        student2 = "Monika"
        student3 = "Erik"
        student4 = "Franziska"
```

1.0.1 Eine Liste erstellen

Listen hingegen können mehrere Elemente enthalten. Man erzeugt eine Liste und füllt sie mit Elementen wie folgt:

Listenname = [Element1, Element2, Element3]

Natürlich kann eine Liste nicht nur drei sondern beliebig viele Elemente enthalten.

Konkret kann das dann so aussehen:

```
In [2]: students = ["Max", "Monika", "Erik", "Franziska"]
```

```
In [3]: marks = [4, 3, 2, 1]
```

Listen kannst du wie Zahlen und Strings per print()-Befehl ausgeben.

```
In [4]: print(students)
```

```
['Max', 'Monika', 'Erik', 'Franziska']
```

In einer Liste dürfen auch Strings und Zahlen nebeneinander vorkommen, doch davon ist abzuraten! Wir werden dafür später geeignetere Strukturen kennenlernen. :-)

1.0.2 Ein Element aus einer Liste auswählen

Du kannst auch auf die Elemente aus einer Liste einzeln zugreifen. Stell dir vor, dass alle Elemente in einer Liste durchnummeriert sind, aufsteigend von 0 an. Per **Index** kannst du ein Element über seine Position in der Liste anwählen:

Listenname[Position]

Konkret sieht das so aus:

```
In [5]: print(students[0])
```

```
Max
```

```
In [6]: print(students[1])
```

```
Monika
```

```
In [7]: print(students[2])
```

```
Erik
```

```
In [8]: print(students[3])
```

```
Franziska
```

Die ausgewählten Elemente aus einer Liste kannst du dann weiterverarbeiten wie du das von Variablen schon kennst. Dabei solltest du beachten, ob es sich bei den Elementen um Zahlen oder Strings handelt.

```
In [9]: print(students[0] + " & " + students[3])
```

```
Max & Franziska
```

```
In [10]: # den Notendurchschnitt ausrechnen
         print((marks[0] + marks[1] + marks[2] + marks[3]) / 4)
```

```
2.5
```

1.0.3 Ein weiteres Element an eine Liste anhängen

Möchtest du ein weiteres Element an deine Liste anhängen, verwendest du den `append()`-Befehl. Anders als die Befehle, die du schon kennst wie den print()-Befehl steht append() nicht für sich, sondern mit einem Punkt *hinter* dem Objekt, auf den append() angewendet wird:

Listenname.append(Element)

Du wirst im Zuge dieses Kurses ganz automatisch lernen, welche Befehle für sich stehen und welche angehängt werden und was genau das jeweils bedeutet :-)

```
In [11]: students.append("Moritz")
```

Jetzt schauen wir uns an, ob der Befehl auch funktioniert hat ;-)

```
In [12]: print(students)
```

```
['Max', 'Monika', 'Erik', 'Franziska', 'Moritz']
```

1.0.4 Die Länge einer Liste abfragen

Mit dem len-Befehl kannst du herausfinden, wie viele Elemente eine Liste enthält: **len(Listenname)**

```
In [13]: print(len(students))
```

```
5
```

1.0.5 Ein Element aus einer Liste entfernen

Um ein Element aus einer Liste zu entfernen, kannst du unter anderem die `pop()`-Funktion verwenden. Dann wird das letzte Element aus der Liste gelöscht. Die `pop()`-Funktion schreibst du wie `append()` mit einem Punkt getrennt hinter die Liste, aus der du das letzte Element entfernen möchtest.

```
In [14]: planets = ["Erde", "Mars", "Jupiter", "Saturn", "Pluto"]
```

```
In [15]: planets.pop()
         print(planets)
```

```
['Erde', 'Mars', 'Jupiter', 'Saturn']
```

Die Besonderheit bei `pop()` besteht darin, dass zusätzlich das gelöschte Element als Ergebnis zurückgeliefert wird.

```
In [16]: planets = ["Erde", "Mars", "Jupiter", "Saturn", "Pluto"]

         p = planets.pop()
         print(p)
```

```
Pluto
```

```
In [17]: print(planets)
```

```
['Erde', 'Mars', 'Jupiter', 'Saturn']
```