

# 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[Index]**

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']
```

# Merkblatt - Listen in Python

September 22, 2017

## 0.1 Listen

Listen ermöglichen es dir, direkt mehrere Einträge auf einmal zu speichern. Hier z.B. erfassen wir eine Liste von 4 Studierenden:

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

        last_student = students.pop()
        print(last_student)
        print(students)
```

Über den + - Operator kannst du 2 Listen miteinander verknüpfen!

```
In [1]: students = ["Max", "Monika", "Erik", "Franziska"] + ["ABCDEF"]
        print(students)

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

Der del - Befehl entfernt einen Eintrag aus einer Liste. Hierbei wird ein Element mit einem bestimmten Index (hier: 4. Element, weil das 4. Element wird über [3] angesprochen, weil ja bei 0 angefangen wird zu zählen)

```
In [3]: students = ["Max", "Monika", "Erik", "Franziska", "ABCDEF"]
        del students[3]
        print(students)

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

Der .remove() - Befehl entfernt einen Eintrag nach Wert. Sprich, hier wird dann der Eintrag "Monika" aus der Liste entfernt.

```
In [12]: students = ["Max", "Monika", "Erik", "Franziska", "ABCDEF"]
         students.remove("Monika")
         print(students)
```

## 0.2 List Comprehensions

Mit Hilfe von List Comprehensions kannst du recht einfach eine Liste in eine andere Liste umwandeln.

```
In [2]: xs = [1, 2, 3, 4, 5, 6, 7, 8]
```

```
ys = [x * x for x in xs]
# ys = []
# for x in xs:
#     ys.append(x * x)
```

```
print(xs)
print(ys)
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

```
[1, 4, 9, 16, 25, 36, 49, 64]
```

### 0.2.1 List Comprehensions können aber noch viel mehr!

```
In [4]: students = ["Max", "Monika", "Erik", "Franziska"]
```

```
lengths = [len(student) for student in students]
```

```
#lengths = []
#for student in students:
#    lengths.append(len(student))
```

```
print(lengths)
```

```
[3, 6, 4, 9]
```

### 0.2.2 Praktisch auch für's Zeichnen von Grafiken!

```
In [2]: %matplotlib inline
import matplotlib.pyplot as plt
```

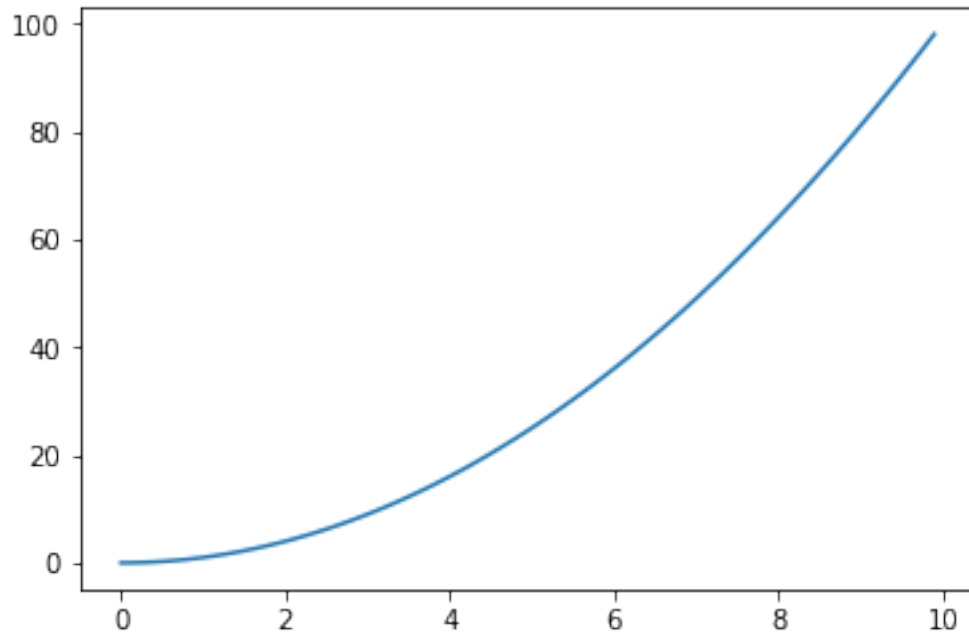
```
In [4]: xs = [x / 10 for x in range(0, 100)]
ys = [x * x for x in xs]
```

```
print(len(xs))
print(len(ys))
```

```
plt.plot(xs, ys)
plt.show()
```

```
100
```

```
100
```



### 0.3 Listen verschachteln

In Python ist es erlaubt, Listen ineinander zu verschachteln. Das erlaubt uns z.B. eine Matrix zu modellieren:

```
In [1]: liste = [  
        ["Berlin", "München", "Köln"],  
        ["Budapest", "Pécs", "Sopron"]  
    ]
```

```
In [5]: liste[0][0]
```

```
Out[5]: 'Berlin'
```

```
In [ ]:
```