

# Einführung in Python — Übung 2

Rebecca Breu, Bastian Tweddell

Juni 2009

## Datentypen II

**Aufgabe 1** Die häufigsten Buchstaben im deutschen Alphabet sind A, D, E, H, I, N, R, S, T, U. Schreiben Sie eine Funktion, welche testet, ob ein String ausschließlich aus solchen Buchstaben besteht.

**1.1** Wie kann man die Funktion unabhängig von Groß-/Kleinschreibung machen?

**1.2** Wie bekommt man die Buchstaben des übergebenen Strings heraus, die nicht in den häufigsten Buchstaben enthalten sind?

**Aufgabe 2** (Morsecode) Schreiben Sie eine Funktion, welche Strings in Morsecode übersetzt. Die Übersetzung soll unabhängig sein von Groß-/Kleinschreibung.

A	· -	J	· - - -	S	· · ·
B	- · · ·	K	- · -	T	-
C	- · - ·	L	· - - ·	U	· · -
D	- · ·	M	- -	V	· · · -
E	·	N	- ·	W	· - -
F	· · - ·	O	- - -	X	- · · -
G	- - ·	P	· - - ·	Y	- - - -
H	· · · ·	Q	- - · -	Z	- - · ·
I	· ·	R	· - ·		

## Objektorientierte Programmierung

**Aufgabe 3** (Punkt-Klasse)

**3.1** Programmieren Sie die Punkt-Klasse aus der Vorlesung nach. Sorgen Sie für eine leserliche Ausgabe unter Verwendung mit `print` und stellen Sie sicher, dass zwei Punkte als gleich gelten, wenn ihre  $x$ - und  $y$ -Werte übereinstimmen.

**3.2** Implementieren Sie den  $+$ - und den  $*$ -Operator für zwei Punkte:

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 + x_2 \\ y_1 + y_2 \end{pmatrix}, \quad \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} * \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = (x_1 * x_2) + (y_1 * y_2)$$

Folgendes soll möglich sein:

```
>>> Punkt(1, 2) + Punkt(4, 2)
(5, 4)
>>> Punkt(1, 2) * Punkt(4, 2)
8
```

**3.3** Implementieren Sie die Norm eines Punktes mithilfe des eben implementierten Skalarprodukts:

$$\|p\| = \sqrt{p * p}$$

#### Aufgabe 4 (Konto)

**4.1** Implementieren Sie eine Klasse `Konto` mit Attributen für die Kontonummer, den Kontostand und den Namen des Kontoinhabers. Das Konto soll nicht überzogen werden können, sorgen Sie dafür, dass ein Zugriff auf den Kontostand keinen negativen Kontostand zur Folge hat.

**4.2** Zusätzlich zu den Kontodaten soll der Zinssatz festgehalten werden, der für alle Konten gleich ist. Implementieren Sie eine Methode, die den Kunden über den aktuellen Zinssatz informiert und eine weitere Methode, welche die Zinsen zum aktuellen Kontostand ausrechnet.

## Pythons Standardbibliothek

**Aufgabe 5** (Passwort-Generator) Schreiben Sie ein Programm, welches ein acht Zeichen langes Passwort generiert. *Hinweis:* Schauen Sie sich `string.letters` und `string.digits` an!

**Aufgabe 6** Schreiben Sie ein Programm, welches alle Dateien eines Verzeichnisses mit Endung `htm` umbenennt, sodass die neue Endung `html` lautet.

**Aufgabe 7** (CSV) Gegeben sei folgende Daten-Datei:

```
"Mr. Spock", "Vulcan"
"Freddie ""Nightmare"" Krueger, Jr.", "Elm Street 42,
Springfield"
```

(Der Zeilenumbruch im zweiten Datensatz ist gewollt!). Die Datensätze bestehen aus zwei Feldern, Name und Adresse. Lesen sie alle Datensätze ein und geben Sie sie auf dem Bildschirm aus.

*Hinweis:* Diesen Datensatz möchte man nicht selbst parsen. (Warum?)

**Aufgabe 8** (Spaß mit regulären Ausdrücken) Schreiben Sie ein Programm, welches alle `html`-Tags aus einer `html`-Datei entfernt. (`html`-Tags sind in spitzen Klammern eingeschlossen, also z.B. `<body>`, `</body>`.)

**Aufgabe 9** (Einweg-Chat) Schreiben Sie einen einfachen „Chat-Server“ und „Chat-Client“. Der Server horcht auf eingehende Socket-Verbindungen. Sobald eine Verbindung hergestellt ist, liest der Server die Daten vom Client und gibt sie auf dem Bildschirm aus. Der Client liest Eingaben von der Tastatur und sendet diese an den Server. „Chatten“ Sie mit Ihrem Nachbarn!

## Zusätzliche Aufgaben

**Aufgabe 10** (MauMau) Implementieren Sie ein einfaches MauMau-Spiel ohne Sonderregeln für bestimmte Karten. Gehen Sie dabei wie folgt vor:

**10.1** (Grundlagen) Vorschlag für eine Strukturierung in Klassen:

- Klasse `Karte` mit Attributen `farbe` und `wert` und einer Methode `kompatibel(self, other)`, die überprüft, ob zwei Karten aufeinander gelegt werden können.
- Klasse für ein gemischtes Kartendeck, welche von `list` erbt:

```
class Deck32(list):
    WERTE = ["7", "8", "9", "10", "Bube", "Dame", "Koenig", "As"]
    FARBEN = ["Karo", "Herz", "Pik", "Kreuz"]

    def __init__(self):
        karten = []

        for farbe in self.FARBEN:
            for wert in self.WERTE:
                karten.append(Karte(farbe, wert))

        random.shuffle(karten)

        list.__init__(self, karten)
```

- Klasse `Spieler`. Der `init`-Methode wird ein Kartendeck-Objekt übergeben, davon werden sechs Karten entfernt und als Handkarten gespeichert, weitere Parameter (Spielername) sind denkbar. Zusätzlich benötigt der Spieler Methoden zum Durchführen eines Spielzuges. Möchte man verschiedene Typen von Spielern einsetzen (Computerspieler, menschlicher Spieler), unterscheiden sich diese Methoden. Man könnte dann die Klassen für die verschiedenen Spieler von einer allgemeinen `Spieler`-Klasse erben lassen.

Lassen Sie das Programm so lange laufen, bis ein Spieler gewonnen hat oder keine Karten mehr vom Talon gezogen werden können. Dafür könnten eigene Ausnahmen nützlich sein, welche von der allgemeinen `Exception` abgeleitet werden:

```
class WinException(Exception):
    pass

class TalonEmptyException(Exception):
    pass
```

**10.2** (Spezialkarten) Noch Lust? Implementieren Sie die Sonderregeln für die Acht (Aussetzen), die Sieben (zwei Karten ziehen) und den Buben (Farbwunsch).

**10.3** (Für Profis) Die sortierte Kartenliste im Kartendeck kann einfacher erzeugt werden. Man kann entweder eine List Comprehension verwenden oder einen eigenen Generator `combine` schreiben, welchen man wie folgt verwendet:

```
karten = list(combine(...))
```

**Aufgabe 11** (Für Profis) Wie man einfache Klassen als beliebige Strukturen verwenden kann, wurde in der Vorlesung gezeigt. Verbessern Sie das Prinzip und schreiben Sie eine Klasse `Bunch`, deren `init`-Methode beliebige Keyword-Argumente nimmt und diese in entsprechenden Attributen speichert:

```
punkt = Bunch(x=2, y=3)
print punkt.x, punkt.y

person = Bunch(vorname="Homer", nachname="Simpson", telefon=123456)
print person.vorname, person.nachname, person.telefon
```

War das nicht genug? Weitere Aufgaben:

<http://www.pythonchallenge.com>