

Python cheat sheet 3

1 Numpy

Ein wichtiges **package** (Paket bzw Bibliothek) heißt **numpy**. Diese Bibliothek muss vor erster Benutzung heruntergeladen werden. Sie bietet die Möglichkeit, zufällige Entscheidungen zu treffen und schöne rechteckige Tabellen zu benutzen (und SEHR viel mehr). Um **numpy** zu benutzen, beginnt man das Programm normalerweise mit:

```
import numpy as np
```

Danach startet man ein Befehl, die aus der **numpy**-Bibliothek stammt, immer mit **np**. Ein paar Beispiele, die mit zufälligen Entscheidungen zu tun haben:

<code>np.random.random()</code>	<i>gibt zurück eine zufällige Zahl zwischen 0 und 1 (z.B 0.276543)</i>
<code>np.random.randint(5, 10)</code>	<i>gibt zurück eine zufällige <u>ganze</u> Zahl zwischen (in diesem Fall) 5 und 9</i>
<code>np.random.choice(['Katze', 'Hund'])</code>	<i>gibt zurück ein zufälliges <u>Element</u> dieser Liste</i>

Und ein paar Beispiele mit Tabellen:

<code>np.full((3, 3), 'X')</code>	<i>gibt zurück eine 3x3 Tabelle voll mit 'X'-Zeichen</i>
<code>board = np.zeros((8, 8))</code>	<i>gibt zurück eine 8x8 Tabelle voll mit 0-Zeichen</i>
<code>board[0][2] = 3</code>	<i>ändert das Element in erster Zeile und dritter Spalte zu 3</i>

2 If-Sätze

Was unter einem **if**-Satz geschrieben ist, passiert, falls die Bedingung erfüllt ist. Falls nicht, sind die Bedingungen der **elif**-Sätzen überprüft. Falls immer noch keine Bedingung erfüllt ist, passiert, was unter **else**-Satz steht. Man kann beliebig viele **elif**-Sätze benutzen - oder gar keinen. Ein **else**-Satz ist auch nicht notwendig. Ein Beispiel:

```
if regen_wahrsch < 0.1:
    print("Heute kein Regen!")
elif 0.1 <= regen_wahrsch < 0.4:
    print("Regen ist unwahrscheinlich.")
elif 0.4 <= regen_wahrsch < 0.8:
    print("Regen ist ziemlich wahrscheinlich.")
else:
    print("Nimm Regenschirm mit! Es wird definitiv regnen!")
```

Weitere Beispiele für mögliche Bedingungen:

<code>if my_number <= 3:</code>	<i>falls die Variable my_number kleiner oder gleich 3 ist</i>
<code>if 5 > 7:</code>	<i>falls nummer 5 größer als 7 ist (passiert natürlich nie)</i>
<code>if username != 'Hana':</code>	<i>falls die Variable username nicht gleich 'Hana' ist</i>
<code>if geld == 0:</code>	<i>falls die Variable geld genau gleich 0 ist</i>
<code>if stop_game:</code>	<i>falls die Variable stop_game gleich True ist</i>

3 Input und Drucken

Input ermöglicht den Benutzer, mit dem Programm zu interagieren. Manchmal ist es nützlich, die Angabe in eine Zahl zu verändern (z.B. um mit der Zahl später zu rechnen).

```
name = input("Was ist dein Name? ")    speichert die Eingabe als die Variable name
alter = int(input("Wie alt bist du? ")) speichert die Eingabe als eine ganze Zahl unter die Variable alter
f"Mein Name ist {name}."               ermöglicht die Werte der Variablen in einem Text zu benutzen
```

4 Schleifen

Oft benutzt man eine `for`-Schleife. Hier kann man drei Parameter auswählen: `range(start, stop, step)`, also Start, Ende und Schritt. Dabei wird die letzte ausgegebene Zahl strikt kleiner als die `stop`-Zahl. Um rückwärts zu gehen, muss man -1 als Schritt angeben.

```
for x in range(9):          -> 0, 1, 2, 3, 4, 5, 6, 7, 8
for reihe in range(-2, 8, 2): -> -2, 0, 2, 4, 6
for variable in range(100, 95, -1): -> 100, 99, 98, 97, 96
```

Wenn man nicht weiß, wie viel mal etwas zu wiederholen ist, kann es nützlich sein, stattdessen eine `while`-Schleife zu benutzen. Hier wird die Schleife beliebig lang wiederholt - so lang, bis der Benutzer ein Passwort mit mehr als 5 Zeichen schreibt und das Befehl `break` erreicht ist.

```
while True:
    password = input("Bitte schreiben Sie Ihr Passwort: ")
    if len(password) < 5:
        print("Das Passwort muss mindestens 5 Zeichen haben!")
    else:
        break
```

5 Funktionen

Eine Funktion (`function`) ist ein Stück Code, den man an beliebig vielen Stellen benutzen kann. Es kann eine oder mehrere Variablen ((`arguments`)) in Klammern haben und es kann mit dem Befehl `return` etwas zurückgeben. Muss aber nicht. Eine super einfache Funktion wäre z.B.:

```
def print_dreimal_hallo():
    print("Hallo!")
    print("Hallo zum zweiten mal!")
    print("Hallo zum dritten mal!")
```

Wenn man diese Funktion schließlich benutzen will, schreibt man an der Stelle:

```
print_dreimal_hallo()
```

Und eine etwas kompliziertere Funktion mit einer Variable und zwei möglichen Zurückgaben:

```
def ist_durch_2_teilbar(nummer):
    """
    Überprüft, ob die angegebene Nummer durch 2 teilbar ist.
    """
    if nummer%2 == 0:
        return True
    else:
        return False
```

Diese Funktion kann man wieder sehr einfach benutzen, z.B. für die Zahl 17:

```
ist_durch_2_teilbar(17)
```