# Scripting 4

TSBE Frühlingssemester 2016

http://smlz.github.io/tsbe-2016fs/scri/

Marco Schmalz

marco.schmalz@gibb.ch



# Kursübersicht

1. Bash und UNIX-Tools
2. Einführung in Python
3. Dicts und Funktionen
4. **I/O und externe Kommandos**
5. Repetition

# Heute

1. Prüfungsbesprechung
2. Repetition
3. `open()`
4. Pfad-Operationen
5. Externe Kommandos ausführen



# Funktionen

Definition und Anwendung:

```
def dividieren(dividend, divisor):
    return dividend / divisor

dividieren(6, 2)                    # -> 3.0

# Argumente können vertauscht werden
dividieren(divisor=3, dividend=15)  # -> 5.0
```

Default-Argumente:

```
def inkrement(zahl, schritt=1):  # Setze schritt standardmässig auf 1
    return zahl + schritt

inkrement(1)              # -> 2
inkrement(3, 2)           # -> 5
inkrement(40, schritt=2)  # -> 42
inkrement(zahl=41)        # -> 42
```

# Dokumentation von Funktionen

```
def add_two(number):
    '''Adds two to a given number, and returns the result.'''
    return number + 2

help(add_two)

# Output:
# Help on function add_two in module __main__:
#
# add_two(number)
#    Adds two to a given number, and returns the result.
```

Auch `docstrings` genannt.



# Rekursion

Eine Funktion kann sich selber aufrufen.

Beispiel: Palindrom überprüfen:

```
def isPalindrome(word):
    if len(word) <= 1:    # Trivialer Fall
        return True

    return word[0] == word[-1] and isPalindrome(word[1:-1])

word = 'Was it a cat I saw'.lower().replace(' ', '')

print(isPalindrome(word))   # -> True
```

# Rekursion: Beispiel Quicksort

```
def partition(lst, reference):
    '''Liste anhand eines Referenz-Elements aufsplitten'''
    smaller, equal, larger = [], [], []
    for element in lst:
        if element < reference:
            smaller.append(element)
        elif element == reference:
            equal.append(element)
        else:
            larger.append(element)
    return smaller, equal, larger

def quicksort(lst):
    '''Liste sortiern'''
    if len(lst) <= 1:  # Trivialer Fall
        return lst
    smaller, equal, larger = partition(lst, lst[0])
    return quicksort(smaller) + equal + quicksort(larger)

zahlen = [12, 34, 12, 56, 34, 9, 33, 18, 3, 7]
quicksort(zahlen)
# -> [3, 7, 9, 12, 12, 18, 33, 34, 34, 56]
```



# Dictionaries

Definition:

```
d = {}  # Leeres dict
jack = {
    'vorname': 'Jack',
    'nachname': 'Stoiker',
    'instrumente': ['Gitarre', 'Gesang'],
    'anzahl_alben': 1,
}
```

Zugriff und Update von Elementen:

```
print(jack['nachname'])   #  -> Stoiker

jack['anzahl_alben'] += 1
```

# Dictionaries (cont.)

Über alle Elemente iterieren:

```
for key, value in jack.items():
    print(key, '->', value)

# Output: (unsortiert)
# nachname -> Stoiker
# vorname -> Jack
# instrumente -> ['Gitarre', 'Gesang']
# anzahl_alben -> 2
```

Überprüfen ob unter einem Key etwas in einem `dict` steht:

```
if 'nachname' in jack:
    print(jack['nachname'])         # -> Stoiker

print('künstlername' in jack)       # -> False

print(jack.get('künstlername', '<Kein Künstlername>'))
                                    # -> <Kein Künstlername>
```



# File lesen

```
file = open('dateiname.ext', 'r')   # 'r' ist optional (default-Wert)

content = file.read()               # lies den gesamten Inhalt der Datei

file.close()                        # Datei schliessen
```

Datei automatisch schliessen:

```
with open('dateiname.ext') as file:
    content = file.read()

# Hier ist die Datei wieder geschlossen
```

# Datei schreiben

```
file = open('dateiname.ext', 'w')   # Löschte den Inhalt der Datei
file.write('Hallo Welt\n')          # Newlines müssen mitgegeben werden
file.close()                        # Jetzt ist der Inhalt geschrieben
file = open('dateiname.ext')        # Öffnen zum Lesen
content = file.read()               # content == 'Hallo Welt\n'
file.close()                        # Datei schliessen
```

Datei automatisch schliessen:

```
with open('dateiname.ext', 'w') as file:
    file.write('Hello World!\n')

with open('dateiname.ext') as file:
    content = file.read()
```