Filehandeling und Listcomprehension

Richard Müller, Tom Felber

13. Januar 2022

Python-Kurs

Gliederung

1. Wiederholung

2. Filehandeling

3. Listcomprehension

Gesamtübersicht

Gesamtübersicht

Themen der nächsten Stunden

- Referenzen Erklärung
- Klassen
- Imports
- Nützliche Funktionen zur Iteration
- Lambda
- Unpacking
- File handeling
- Listcomprehension
- Dekoratoren

Wiederholung

Wiederholung

Beim letzten Mal:

- Unpacking

```
*a, = 1, 2, 3
# a = [1, 2, 3]
```

```
def func(*args):
    print(args)

func(42, 1337, True, None, "Hello")
# gibt (42, 1337, True, None, "Hello") aus
```

- Lambda

```
def funktion(a1, a2):
    return a1 + a2

lambda a1, a2: a1 + a2
```

Filehandeling

open Befehl

`open()` ist eine builtin Funktion zum öffnen von Dateien

```
datei_objekt = open(datei_pfad, bearbeitungs_modus)
```

`open` hat verschiedene Bearbeitungsmodi

```
# reading
f = open("filename.txt", "r")
```

```
# writing
f = open("filename.txt", "w")
```

```
# append
f = open("filename.txt", "a")
```

File Objekte

Die `open` Funktion gibt ein `File` Objekt zurück. Was kann man damit machen ?

- Die Datei schließen mit `f.close()` Dies sollte immer passieren, wenn die Bearbeitung abgeschlossen ist.
- Den kompletten Inhalt der Datei lesen mit `f.read()`
- Zeile für Zeile den Inhalt lesen mit `f.readline()`
- In die Datei schreiben mit `f.write("content")`
- Die Stelle verändern an der gelesen / geschrieben wird mit `f.seek(position)`

Bearbeitungsmodi

Zusätzlich zu den normalen Modi gibt es den jeweiligen Modus mit einem + . Diese Modi haben mehr Rechte als der Originale.

Rechte / Modus	r	w	a	r+	w+	a+
Read	✓			✓	✓	✓
Write		✓	✓	✓	✓	✓
Create		✓	✓		✓	✓
Am Anfang starten	✓	✓		✓	✓	
Am Ende starten			✓			✓

Tipp: with open()

Mit `with open() as name:` wird die Datei automatisch wieder geschlossen, wenn das aufgemachte scope endet. `f.close()` kann nicht vergessen werden.

```
with open("examplefile.txt", "w+") as f:
    # Datei ist hier offen
    f.readline()
    f.write("test")
# Datei ist hier nicht länger geöffnet
```

Listcomprehension

Basics

list comprehensions sind eine Möglichkeit, Listen und andere `iterables` schneller und einfacher zu erstellen/manipulieren, als über herkömmliche Wege.

Die folgenden beiden Codeabschnitte führen zum genau gleichen Ergebnis:

```
# Standardweg
liste = []
for i in range(5):
    liste.append(i)
```

```
# list comprehension
liste = [i for i in range(5)]
```

Verwendung mit `if`

Man kann list comprehensions auch in Verbindung mit der `if` -Abfrage benutzen:

```
1 fruits = ["apple", "avocado", "banana", "strawberry", "
      raspberry"]
2 filt = [fruit for fruit in fruits if fruit.startswith("a")]
3 # ['apple', 'avocado']
```

oder zusätzlich mit else:

```
1 filt = [fruit if fruit.startswith("a") else "IIIHHH, eine
      Frucht ohne A!" for fruit in fruits]
2 # ['apple', 'avocado', 'IIIHHH, eine Frucht ohne A!', '
      IIIHHH, eine Frucht ohne A!', 'IIIHHH, eine Frucht ohne
      A!']
```

Verwendung mit mehreren Schleifen

Mehrere Schleifen zu verketten ist auch möglich:

```
1 nested = [["a", "b"], [1, 2, 3]]
2 multiple = [i for sub in nested for i in sub]
3 # ['a', 'b', 1, 2, 3]
```