Python-Kurzreferenz

Variablen und Wertezuweisung (Assignment)

```
name = "Darwin"
```

Einfach Datentypen

str - Zeichenkette (String), `"Hallo Welt"`

int - Ganzzahl (Integer), `234`

float - Gleitkommazahl (Floating point number), `42.23`

bool - Boolesche Werte (Boolean)- `True`/`False`

Container-Datentypen

Listen

```
names = ["Noether", "Darwin", "Lovelace"]
```

Elemente werden durch den Index adressiert - z.B: `names[0]`

Dictionaris

- Schlüssel/Werte-Paar

```
person_and_birth_years = {"Noether": 1882, "Darwin": 1809, "Lovelace": 1815}
```

- Werte werden über die Schlüssel adressiert - z.B: `person_and_birth_years["Noether"]`

Operatoren

- `+`, `-`, `*`, `/`
- `==`, `!=`, `<`, `>`, `=<`, `=>`
- `not`, `and`, `or`

for-Schleifen (for-Loops)

```
for <Variable> in <Liste/Interable>:
    <Auszuführender Block>
```

Bedingte Anweisung (Conditionals)

Ausführung bei Bedingung

```
if <Bedingung>:
    <Auszuführender Block>
```

Ausführung bei Bedingung und Alternative, wenn Bedingung nicht erfüllt

```
if <Bedingung>:
    <Auszuführender Block>
else:
    <Auszuführender Block>
```

Verschiedene Bedingungen und Alternative, wenn Bedingung nicht erfüllt

```
if <Bedingung 1>:
    <Auszuführender Block>
elif <Bedingung 2>:
    <Auszuführender Block>
else:
    <Auszuführender Block>
```

Kommentare

- Alles was rechts von einem # steht, wird nicht interpretiert

Bibliotheken einbinden

Importieren eines Paketes

```
import csv
```

Importieren eines Modules einer Bibliothek

```
import urllib.request
[...]
urllib.request.urlopen
```

Import mit Modul mit einem anderen Namen

```
import pandas as pd
```

Funktionen und Methoden nutzen

Funktionen

- Funktionen eine Gruppierug von Anweisungen
- Funktionen können keine bis mehrer Parameter besitzen
- Funktionaufruf durch runde Klammern “()”
- Beispiel:
    - `print("Hello World!")`
    - `type(counter)`
    - len([5, 23, 52])

Methoden

- Methoden sind Funktionen, die an Objekte gebunden sind

Beispiele - `name.upper()` - `name.replace("und", "oder")`

Datein öffnen

Mit `open(<Dateinname>` wird ein sogenannter File-Handle erzeugt. Der ganze Inhalt kann dann mittel der Methode `read` der File-Handles eingelesen werden:

```
my_file_handle = open("Meine Daten.txt"):
file_content = my_file_handle.read()
```

Alternativ kann man auch Zeile für Zeile eine Datein einlesen:

```
for line in open("Meine_Daten.txt"):
    print(line)
```

Leseempfehlung

- “Automate the Boring Stuff with Python”, Al Sweigart, https://automatetheboringstuff.com/