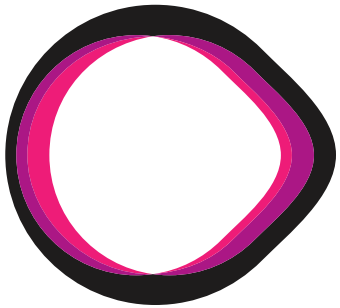
Python – Lektion 5
Dateien lesen und schreiben, Pathlib Modul, Listen und Dictionaries im Detail



OST
Ostschweizer
Fachhochschule

Rückblick

- Tupel Packing und Unpacking (mit Rest)

```
t = ('Mia', 'Müller', 'Mattstr. 10', 9500, 'Wil')
vorname, nachname, *adresse = t
```

- Funktionsparameter im Detail

Rückblick

- Tupel Packing und Unpacking (mit Rest)

```
t = ('Mia', 'Müller', 'Mattstr. 10', 9500, 'Wil')
vorname, nachname, *adresse = t
```

- Funktionsparameter im Detail

  `foo(1, 2, 3, d=4, e=5)` → `def foo(a, *args, **kwargs):`

  `*` packt überschüssige Positionsargumente in ein Tupel

  `**` packt überschüssige Schlüsselwortargumente in ein Dictionary

Rückblick

- Tupel Packing und Unpacking (mit Rest)

```
t = ('Mia', 'Müller', 'Mattstr. 10', 9500, 'Wil')
vorname, nachname, *adresse = t
```

- Funktionsparameter im Detail

  `foo(*noten)` → `foo(4.5, 6, 4)`

  Iterable mit `*` entpacken und als Positionsargumente übergeben

  `foo(**noten)` → `foo(ET=4.5, Py=6, Comm=4)`

  Iterable mit `**` entpacken und als Schlüsselwortargumente übergeben

Rückblick

- Tupel Packing und Unpacking
- Funktionsparameter im Detail

```
[*noten, 4, 5, 6] → [4.5, 6, 4, 4, 5, 6]
```

Iterable mit * in eine Liste entpacken

```
{**noten, **noten2} → {"ET": 4.5, "Py": 6, "Comm": 4, "Sig": 5}
```

Dictionaries mit ** in anderes Dictionary entpacken

Rückblick

- List Comprehensions

mit `if` am Ende für das Filtern von Elementen:

```
values = [expression1
          for item in iterable
          if condition]
```

mit inline `if` vor der for-Schleife für die Entscheidung zwischen zwei möglichen Ausdrücken:

```
values = [expression1
          if condition else expression2
          for item in iterable]
```

Rückblick

- Set Comprehensions

mit `if` am Ende für das Filtern von Elementen:

```
values = {expression1
          for item in iterable
          if condition}
```

mit inline `if` vor der for-Schleife für die Entscheidung zwischen zwei möglichen Ausdrücken:

```
values = {expression1
          if condition else expression2
          for item in iterable}
```

Rückblick

- Dict Comprehensions

mit `if` am Ende für das Filtern von Elementen:

```
values = {item:expression1
          for item in iterable
          if condition}
```

mit inline `if` vor der for-Schleife für die Entscheidung zwischen zwei möglichen Ausdrücken:

```
values = {item:expression1
          if condition else item:expression2
          for item in iterable}
```

Heutige Themen

- Dateien lesen und schreiben
- Pathlib Modul
- Listen im Detail
- Flaches und tiefes Kopieren
- Dictionaries im Detail

Dateien lesen und schreiben

- `open()`
- `with`
- `read()`
- `write()`

`http://localhost:8888/notebooks/dateien.ipynb`

Datei öffnen

- Datei mit der `open()`-Funktion öffnen:

```
# Lesen
with open("dokument.txt") as f:
# Lesen
with open("dokument.txt", "r") as f:
# Schreiben
with open("dokument.txt", "w") as f:
# Anhängen
with open("dokument.txt", "a") as f:
# Binär lesen
with open("dokument.txt", "rb") as f:
# Binär schreiben
with open("dokument.txt", "wb") as f:
```

- Weitere Parameter findet man in der Hilfe[1]

[1] `https://docs.python.org/3/library/functions.html#open`

Dateien lesen und schreiben

- Datei lesen:

```
inhalt = f.read()           # gesamte Datei lesen
inhalt = f.read(n)          # n Zeichen lesen
zeilen = f.readlines()      # Liste aller Zeilen
```

- Datei schreiben:

```
f.write("hello")            # String schreiben
f.writelines(["1\n", "2\n"]) # Liste von Strings
```

- Datei schliessen:

```
f.close()
```

Pathlib Modul

- Methoden für den Umgang mit Dateien, Verzeichnissen und allgemein Path Objekten:

`http://localhost:8888/notebooks/pathlib.ipynb`

Listen im Detail

- Listen
  - Elemente hinzufügen, ersetzen und entfernen
  - Sortieren
- Flaches und tiefes Kopieren

`http://localhost:8888/notebooks/listen.ipynb`

Listen im Detail

- Elemente hinzufügen:

```
liste.append("x")
liste.insert(2,"y")
liste += [3, 4]
liste.extend([5, 6])
```

- Elemente ersetzen:

```
liste[1] = "B"
liste[3:] = ["C", "D"]
```

- Elemente entfernen:

```
element = liste.pop()
element = liste.pop(0)
element = liste.remove("D")
```

- Elemente sortieren:

```
sortiert = sorted(liste)
```

Dictionaries im Detail

- Dictionaries
  - Elemente hinzufügen, ersetzen und entfernen

`http://localhost:8888/notebooks/dict.ipynb`

Dictionaries im Detail

- Auf Werte/Schlüssel zugreifen:

```
title = book["Title"]
title = book.get("Title")
author = book.get("Author", "not provided")
all_values = book.values()
all_keys = book.keys()
all_items = book.items()
```

- Elemente hinzufügen:

```
book["Title"] = "Numerisches Python"
book.setdefault("Erscheinungsjahr", 2017)
```

- Elemente entfernen:

```
book.pop("Author")
book.popitem()
book.clear()
```