CSV und JSON

Datenpersistenz - Warum?

Programme verlieren
ihre Daten beim Beenden

Beispiel: Eine To-do-Liste

```
todos = []
# ...
todos = ["Einkaufen", "Python lernen"]
# Programm wird beendet
# Beim nächsten Start:
# todos ist anfangs wieder leer!
```

CSV-Dateien - Einfach und Praktisch

CSV = Comma-Separated Values

Beispiel einer CSV-Datei students.csv:

```
name,alter,studienfach
Anna,22,Informatik
Ben,19,Mathematik
Clara,24,Informatik
```

CSV Lesen und Schreiben

Python's csv-Modul macht's einfach

Verschiedene Trennzeichen möglich (nicht nur Kommas)

Beispiel: Semikolon als Trennzeichen, Anführungszeichen für Texte

```python
import csv

# Lesen einer CSV-Datei
with open('stud.csv', 'r', encoding='utf-8') as file:
    reader = csv.reader(file, delimiter=';')
    for row in reader:
        print(row)  # row ist eine Liste

# Schreiben einer CSV-Datei
students = [
    ['Name', 'Alter', 'Lieblingsfach'],
    ['Anna', '22', 'Machine Learning'],
    ['Ben', '19', 'Software Engineering']
]

with open('neue_stud.csv', 'w', newline='',
          encoding='utf-8') as file:
    writer = csv.writer(file, delimiter=';',
                        quotechar='"',
                        quoting=csv.QUOTE_NONNUMERIC)
    writer.writerows(students)
```

CSV Lesen und Schreiben

```
"Name";"Alter";"Lieblingsfach"
"Anna";"22";"Machine Learning"
"Ben";"19";"Software Engineering"
```

JSON - Für komplexere Daten

JSON = JavaScript Object Notation

Unterstützt verschachtelte Strukturen

Beispiel einer JSON-Datei *kurse.json*:

```
{
    "kurse": [
        {
            "name": "Programmierung",
            "dozent": "Prof. Schmidt",
            "teilnehmer": [
                {"name": "Anna", "matrikelnr": 12345},
                {"name": "Ben", "matrikelnr": 12346}
            ],
            "aktiv": true,
            "credits": 6
        }
    ]
}
```

JSON in Python

Modul json für einfaches Lesen und Schreiben

Python-Datentypen werden automatisch konvertiert

```python
import json

# Python-Daten
kurs_daten = {
    "name": "Programmierung",
    "teilnehmer": ["Anna", "Ben"],
    "bestanden": True,
    "durchschnitt": 2.3
}

# Schreiben
with open('kurs.json', 'w', encoding='utf-8') as f:
    json.dump(kurs_daten, f, indent=4)

# Lesen
with open('kurs.json', 'r', encoding='utf-8') as f:
    geladene_daten = json.load(f)
```

Praxisbeispiel: Todo-Liste persistent machen

```
import json

class TodoList:
    def __init__(self):
        self.todos = self.load_todos()

    def add_todo(self, task):
        self.todos.append({"task": task, "done": False})
        self.save_todos()

    def mark_done(self, index):
        self.todos[index]["done"] = True
        self.save_todos()

    def load_todos(self):
        try:
            with open('todos.json', 'r') as f:
                return json.load(f)
        except FileNotFoundError:
            return []

    def save_todos(self):
        with open('todos.json', 'w') as f:
            json.dump(self.todos, f, indent=2)
```

JSON in Web-Anwendungen

Flask's `jsonify` für HTTP-Responses

Nützlich für: Web-APIs, Datenexport, AJAX-Anfragen

```python
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/api/todos')
def get_todos():
    todos = [
        {"task": "Flask lernen", "done": True},
        {"task": "JSON verstehen", "done": False}
    ]
    return jsonify(todos)

@app.route('/api/student/<int:matrikelnr>')
def get_student(matrikelnr):
    student = {
        "name": "Anna Schmidt",
        "matrikelnr": matrikelnr,
        "kurse": ["Programmierung", "Datenbanken"]
    }
    return jsonify(student)
```

APIs konsumieren

Mit Python (requests-Modul):

```python
import requests

# API aufrufen und JSON-Daten empfangen
response = requests.get('http://localhost:5000/api/todos')
todos = response.json()
print(todos)  # Liste von Dictionaries

# Einzelnen Studenten abfragen
response = requests.get('http://localhost:5000/api/student/12345')
student = response.json()
print(f"Kurse von {student['name']}: {student['kurse']}")
```

APIs konsumieren

```
// Grundlegende Verwendung von fetch (ohne Error Handling)
function getTodos() {
    fetch('http://localhost:5000/api/todos')
    .then(function(response) {
        return response.json();
    })
    .then(function(data) {
        console.log(data);
    });
}
```

EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

Die Drei Extra-Minuten

CSV-Dateien richtig handhaben:

```
import csv

# Robust CSV lesen
with open('daten.csv', 'r', encoding='utf-8') as f:
    reader = csv.DictReader(f, delimiter=';')
    for row in reader:
        # row ist jetzt ein Dictionary mit Spaltennamen als Keys
        print(f"{row['name']}: {row['alter']}")

# Sicher CSV schreiben
with open('export.csv', 'w', newline='', encoding='utf-8') as f:
    fieldnames = ['name', 'alter', 'kurs']
    writer = csv.DictWriter(f, fieldnames=fieldnames)
    writer.writeheader()  # Schreibt die Spaltenüberschriften
    writer.writerow({
        'name': 'Anna',
        'alter': '22',
        'kurs': 'Informatik'
    })
```

Die Drei Extra-Minuten

```
# Häufige Fehler vermeiden
try:
    with open('daten.json', 'r', encoding='utf-8') as f:
        data = json.load(f)
except FileNotFoundError:
    # Datei existiert nicht
    data = {"records": []}
except json.JSONDecodeError:
    # Ungültiges JSON
    print("Fehler beim Lesen der JSON-Datei!")
```

Die Drei Extra-Minuten

```
# Problematisch bei mehreren gleichzeitigen Zugriffen!
with open('shared_data.json', 'r+') as f:
    data = json.load(f)
    data['counter'] += 1  # Race Condition möglich!
    f.seek(0)             # zurück an den Dateianfang
    json.dump(data, f)
    f.truncate()          # alten Rest abschneiden
```

Programme verlieren Daten beim Beenden – Persistenz speichert sie dauerhaft

CSV = Comma-Separated Values: einfaches, tabellarisches Textformat

Pythons csv-Modul liest und schreibt CSV (auch mit anderen Trennzeichen)

JSON = JavaScript Object Notation: unterstützt verschachtelte Strukturen

Das json-Modul konvertiert Python-Datentypen automatisch

jsonify und requests ermöglichen Web-APIs und den Datenaustausch