CSV

CSV Daten verarbeiten in Python

Was sind CSV-Dateien?

CSV steht für "Comma-Separated Values" und bezeichnet ein einfaches Dateiformat, das zum Speichern tabellarischer Daten verwendet wird.

Eine typische CSV-Datei ist textbasiert und besteht aus Zeilen, wobei jede Zeile einen Datensatz repräsentiert.

Obwohl der Name "Comma-Separated Values" ein Komma als Standardtrennzeichen impliziert, können auch andere Zeichen wie Semikolon, Tab oder Leerzeichen verwendet werden.

Eine CSV-Datei kann eine Kopfzeile enthalten, die die Spaltennamen definiert. Dies erleichtert das Verständnis und die Verarbeitung der Daten.

Standardisierung

Standardisierung: Es gibt keinen strikten Standard für CSV-Dateien, was zu Inkonsistenzen bei der Handhabung verschiedener CSV-Formate führen kann.

Komplexere Datenstrukturen: Für verschachtelte oder hierarchische Datenstrukturen ist CSV nicht gut geeignet.

Beispiel

Name,Email,Telefon

Max Mustermann,max@example.com,0123-4567890

Erika Musterfrau,erika@example.com,0987-6543210

Sandra Beispiel,sandra@example.com,1122-334455

Die erste Zeile ist die Kopfzeile. Sie definiert die Namen der Spalten: "Name", "Email" und "Telefon".

Jede folgende Zeile entspricht einem Datensatz. In diesem Fall repräsentiert jede Zeile eine Person und ihre Kontaktdaten.

der CSV-Reader

Python bietet mit dem Modul csv eine komfortable Möglichkeit, CSV-Dateien zu parsen.

```
import csv

with open("test.csv") as file:
    # Erstellen eines CSV-Reader-Objekts
    csv_reader = csv.reader(file)

    # Durchlaufen jeder Zeile im CSV-Reader
    for zeile in csv_reader:
        # Drucken jeder Zeile als Liste
        print(zeile)
```

der Dict-Reader

Zum CSV-reader gibt es noch eine Alternative: den Dict-Reader. Er liest ebenfalls die Zeilen ein, allerdings als dict. Dazu muss ein Header gegeben sein. Falls die Datei keinen Header hat, kann er mit fieldnames übergeben werden.

```
import csv

with open(csv_datei_pfad) as file:
    # Erstellen eines CSV DictReader-Objekts
    csv_dict_reader = csv.DictReader(file)

    # Durchlaufen jeder Zeile im CSV DictReader
    for zeile in csv_dict_reader:
        # Drucken der Zeile als Dictionary
        print(zeile)
```

Fieldnames

Der Dict-Reader benötigt zum Lesen einen Header. Mit fieldnames kann man einen Header optional anbieten, wenn keiner in der CSV-Datei vorhanden ist.

```
feldnamen = ['Name', 'Email', 'Telefon']


with open(csv_datei_pfad) as datei:
    # Erstellen eines CSV DictReader-Objekts mit Feldnamen
    csv_dict_reader = csv.DictReader(datei, fieldnames=feldnamen)
```

CSV Writer

Um CSV-Dateien zu schreiben, gibt es den CSV-Writer.

```
import csv

daten = [
    {'Name': 'Max Mustermann', 'Email': 'max@example.com', 'Telefon': '0123-4567890'},
    {'Name': 'Erika Musterfrau', 'Email': 'erika@example.com', 'Telefon': '0987-6543210'},
]

# Öffnen der Datei zum Schreiben
with open('beispiel.csv', mode='w') as file:
    # Erstellen eines CSV Writer-Objekts
    writer = csv.writer(file)

    # Schreiben der Kopfzeile
    writer.writerow(['Name', 'Email', 'Telefon'])

for eintrag in daten:
        writer.writerow([eintrag['Name'], eintrag['Email'], eintrag['Telefon']])
```

CSV Dict Writer

```
persons = [
    {"id": 1, "name": "Klaus"},
    {"id": 2, "name": "Mohamad"},
]

with open("persons.csv", "w", newline='') as f:
    fieldnames = ["id", "name"]
    writer = csv.DictWriter(f, fieldnames=fieldnames, delimiter=",")
    writer.writeheader()
    writer.writerows(persons)
```