CSV Datei einlesen mit Python

Viele Daten liegen als CSV-Datei vor bzw. können sehr einfach in dem Format CSV gespeichert werden. Was ist eine CSV-Datei überhaupt? Schaut man sich die Bedeutung der einzelnen Buchstaben an, wird vieles klarer:

- C = Comma = Komma
- S = separated = getrennte
- V = values = Werte

Unsere Daten liegen also durch Kommas voneinander getrennt in einer Datei vor. Meistens sind in der ersten Linie die Namen der Datenspalten hinterlegt.

Schauen wir uns den Inhalt einer CSV-Datei an:

nachname,vorname,geburtstag
Müller,Mike,1980-03-05
Sommer,Elke,1987-05-02
Schuster,Johanna,1993-10-10

Allerdings muss man wissen, dass das Trennzeichen nicht zwingend ein Komma sein muss, aber das Dateiformat weiterhin CSV sich nennt.

Gerne werden auch folgende Trennzeichen (engl. delimiters) verwendet:

- Semikolon ;
- Doppelpunkt :
- TAB \t

Verfügt man über Daten, die in einer Exceltabelle vorliegen, kann man diese direkt aus Excel heraus im CSV-Format abspeichern. Excel erlaubt zusätzlich die Angabe, welche Trennzeichen verwendet werden sollen.

Und hier kommt der Vorteil von fertigen Bibliotheken. Einfach nutzen ohne große Probleme.

CSV-Datei mit der in Python eingebauten Bibliothek auslesen

Python verfügt bei der Standardinstallation bereits über eine CSV-Bibliothek. Diese können wir einfach über `import csv` in unser Python-Programm einbinden.

Zum Testen speichern wir unsere Adressdaten von oben in die Textdatei mit dem Namen „adressen.csv“.

```
import csv
```

Jetzt müssen wir unsere CSV-Datei zum Auslesen öffnen:

```
import csv
with open('adressen.csv') as csvdatei:
```

Wir können nun unser Programm ausführen, aber es passiert noch nichts. Wir benötigen unser `csv.reader`, der uns aus geöffneten CSV-Datei ein „csv.reader object“ macht:

```
import csv
with open('adressen.csv') as csvdatei:
    csv_reader_object = csv.reader(csvdatei, delimiter=',')
    print(csv_reader_object)
```

Unserem `csv.reader` übergeben wir unsere geöffnete CSV-Datei und können auch das Trennzeichen über die Anweisung `delimiter=','` mitgeben. Als Standard ist das Komma hinterlegt, daher müssten wir bei einem Komma als Trennzeichen nicht einmal den „delimiter“ festlegen.

```
import csv
with open('adressen.csv') as csvdatei:
    csv_reader_object = csv.reader(csvdatei)
    print(csv_reader_object)
```

Jetzt haben wir unser „csv.reader object“, dass wir über die `for … in` Konstruktion durchlaufen und unsere Daten ausgeben können.

```
import csv
with open("adressen.csv") as csvdatei:
    csv_reader_object = csv.reader(csvdatei)
    for row in csv_reader_object:
        print(row)
```

Unsere Daten liegen in Form einer Liste vor und können dementsprechend genutzt werden. Als Ausgabe erhalten wir:

['nachname', 'vorname', 'geburtstag']
['Müller', 'Mike', '1980-03-05']
['Sommer', 'Elke', '1987-05-02']
['Schuster', 'Johanna', '1993-10-10']

Jetzt können wir noch die erste Zeile abfangen und unsere Daten nutzen. Unser kompletter Programmcode:

```
import csv

with open("adressen.csv") as csvdatei:
    csv_reader_object = csv.reader(csvdatei)

    zeilennummer = 0
    for row in csv_reader_object:

        if zeilennummer == 0:
            print(f'Spaltennamen sind: {", ".join(row)}')
        else:
            print(f'- Nachname: {row[0]} \t| Vorname: {row[1]} \t| Geburtstag:
{row[2]}.')
        zeilennummer += 1

    print(f'Anzahl Datensätze: {zeilennummer-1}')
```

CSV-Datei einlesen als Datentyp Dictionary

Anstelle einer Liste können wir auch den Datentyp „Dictionary“ erhalten. Dazu gibt es den `csv.DictReader`. Unsere eingelesene Datei liegt danach als Wörterbuch („Dictionary“) vor.

```
import csv
with open("adressen.csv") as csvdatei:
    csv_reader_object = csv.DictReader(csvdatei)
    for row in csv_reader_object:
        print(row)
```

Die erste Zeile wird automatisch für Indizes verwendet:

OrderedDict([('nachname', 'Müller'), ('vorname', 'Mike'), ('geburtstag', '1980-03-05')])

OrderedDict([('nachname', 'Sommer'), ('vorname', 'Elke'), ('geburtstag', '1987-05-02')])

OrderedDict([('nachname', 'Schuster'), ('vorname', 'Johanna'), ('geburtstag', '1993-10-10')])

Funktionen und Klassen des Moduls „csv“

Welche Funktionen und Klassen das Modul „csv“ beinhaltet, kann man über `print(dir(csv))` ausgeben lassen.