CSV (1)

- **CSV** (Comma Separated Values) sind uns schon einmal in einer Aufgabe zur Notenberechnung begegnet. Dieses Format eignet sich für Datentabellen, so wie sie beispielsweise in Excel verwendet werden. Als Trennzeichen kann man nicht nur Kommas sondern beliebige Zeichen verwenden. Das Trennzeichen sollte aber möglichst nicht in den Werten enthalten sein.
- Im Prinzip kann man csv-Dateien mit dem Methoden für Textdateien verarbeiten. Aber Python wäre nicht Python, wenn es kein spezielles Modul dafür gäbe. Beispiel:

```
import csv

with open("punkte.csv", "r", newline="") as fh:
    reader = csv.reader(fh, delimiter=";")
    for row in reader:
        print(row)
```

Die csv-Datei muss mit der Option newline="" geöffnet werden, sonst werden Zeilenumbrüche falsch interpretiert.

Die Funktion **reader** liefert ein Objekt, mit dem man anschließend durch die Datei laufen kann. Das Trennzeichen ist der Strichpunkt.

Beim Durchlaufen der Datei mit dem reader wird jede Zeile als Liste von Strings ausgegeben.

Beispiel Noten (1)

- Natürlich können die Zeilen nicht nur mit print ausgegeben, sondern auch verarbeitet werden. Beispiel:

```
with open("punkte.csv", "r", newline="") as fhi:
    reader = csv.reader(fhi, delimiter=";")
    with open("noten.txt", "w") as fho:
        print("Matrikel   A1   A2   A3   A4   A5   A6   A7   A8 Punkte Note", file=fho)

        for punkte in reader:
            summe = 0
            print(punkte[0].rjust(8, " "), end=" ", file=fho)

            for i in range(1,9):
                summe = summe + float(punkte[i])
                print(punkte[i].rjust(4, " "), end=" ", file=fho)

            print(str(summe).rjust(6, " "), end=" ", file=fho)
            print("1.0".rjust(4, " "), file=fho)   # Jeder kriegt eine 1.0 :-)
```

Beispiel Noten (2)

- Standardmäßig werden die Elemente als Strings eingelesen. Durch die Option quoting=csv. QUOTE_NONNUMERIC werden numerische Daten bei Einlesen in Floats umgewandelt. Beispiel:

```
with open("punkte.csv", "r", newline="") as fhi:
    reader = csv.reader(fhi, delimiter=";", quoting=csv.QUOTE_NONNUMERIC)
    with open("noten.txt", "w") as fho:
        print("Matrikel   A1   A2   A3   A4   A5   A6   A7   A8 Punkte Note", file=fho)

        for punkte in reader:
            print("{0:8.0f}".format(punkte[0]), end=" ", file=fho)

            for i in range(1,9):
                print("{0:4.1f}".format(punkte[i]), end=" ", file=fho)

            print("{0:6.1f}".format(sum(punkte[1:9])), end=" ", file=fho)
            print("{0:4.1f}".format(1.0), file=fho)   # Jeder kriegt eine 1.0 :-)
```

punkte enthält jetzt Floats, die anders formatiert werden müssen.

Da wir Floats haben, können wir zur Summenberechnung die Funktion sum verwenden.

Beispiel Noten (3)

- Im Beispiel könnte man die Ergebnisse auch wieder in eine csv-Datei ausgeben:

```
with open("punkte.csv", "r", newline="") as fhi:
    reader = csv.reader(fhi, delimiter=";", quoting=csv.QUOTE_NONNUMERIC)
    with open("noten.csv", "w", newline="") as fho:
        writer = csv.writer(fho, delimiter=";")
        writer.writerow(["Matrikel", "A1", "A2", "A3", "A4", "A5", "A6", "A7", "A8", "Punkte", "Note"])

        for punkte in reader:
            punkte = punkte + [sum(punkte[1:9]), 1.0]
            punkte[0] = "{0:.0f}".format(punkte[0])
            punkte[1:11] = map(lambda z:str(z).replace(".", ","),
                               punkte[1:11])
            writer.writerow(punkte)
```

Die Funktion **writer** liefert ein Objekt, mit dem man anschließend die Zeilen der csv-Datei scheiben kann. Das Trennzeichen ist der Strichpunkt.

Schreiben einer Zeile = Liste von Strings mit der Funktion **writerow**

Anhängen von Punktesumme und Note

Matrikelnummer formatieren

Ersetzen der Punkte in den Zahlen durch Kommas

Schreiben einer Zeile in die Ausgabedatei

Beispiel Noten (4)

Aufgaben

1. Gegeben sei eine csv-Datei mit einem quadratischen n×n-Zahlenschema. Beispiel:

   ```
   1,2,3
   4,5,6
   7,8,9
   ```

   Erstellen Sie ein Skript, das die Zahlen in eine numpy-Matrix einliest. Die Matrix soll danach erweitert werden durch die Zeilensummen, Spaltensummen, Spur und auf die Konsole ausgegeben werden. Das Skript soll ohne Änderung für beliebige n funktionieren, d.h. n muss aus der csv-Datei ermittelt werden.

   Konsolausgabe für das Beispiel von oben:

   ```
   [[ 1.  2.  3.  6.]
    [ 4.  5.  6. 15.]
    [ 7.  8.  9. 24.]
    [12. 15. 18. 15.]]
   ```