

Datenspeicherung und Datenaustausch über Textdateien

1. In eine Textdatei schreiben:

Mit `open()` wird ein Datei-Handler, hier das Objekt `datei` erstellt. Er ermöglicht den Dateizugriff. Es wird die Textdatei erstellt.

```
1 datei = open("Textdatei.txt", "a")
2 datei.write("Hallo, guten Tag")
3 datei.close()
```

einfacher mit with:

with arbeitet ähnlich wie eine Kontrollstruktur, das Objekt für den

```
1 with open('Textdatei.txt', 'w') as inhalt:
2     inhalt.write('Guten Tag')
```

Zugriff wird mit **as** definiert. Wenn die Einrückung beendet ist, wird alles automatisch geschlossen (praktisch, wenn mehrere Dateien geöffnet wurden; hinter `with` kann `open()`, durch Komma getrennt, mehrmals aufgerufen werden).

Parameter von `open()`:

Modus	Auswirkung auf Datei	Position i. d. Datei
r	öffnet die Datei ausschließlich zum Lesen.	Anfang
w	öffnen zum Schreiben dabei löschen aller Inhalte.	Anfang
a	öffnen, neue Inhalte werden an das Ende geschrieben.	Ende
r+	schreiben und lesen, aktualisiert ab der Position bei Aufruf!, der Rest bleibt erhalten	Anfang
w+	schreiben und lesen (Inhalte werden überschrieben)	Anfang

Anwendung

Das folgende Programm speichert Temperatureingaben mit Datum und Uhrzeit. Bei einer leeren Eingabe wird das Programm beendet.

temperaturwerte.py

```
1 from time import *
2 with open("Temperaturwerte.txt", "a") as datei:
3     while True:
4         text = input("Temperaturwert: ")
5         if text == "":
6             break
7         datei.write(asctime())
8         datei.write("    "+text +"\n")
```

Anweisungen
with - Struktur

Anweisungen
while - Schleife

Die `while` – Schleife wird solange durchlaufen, bis sie mit `break` verlassen wird. Die Methode `write` erzeugt keinen Zeilenumbruch (im Gegensatz zu `print`), er wird mit `"\n"` hinzugefügt.

2. Aus einer Textdatei lesen

Datei als Ganzes
auslesen:

```
1 with open('Textdatei.txt','r') as datei:  
2     print(datei.read())
```

Die Methode `read` liest die gesamte Datei aus und gibt sie zurück – hier wird alles mit `print` ausgegeben.

Datei zeilenweise auslesen:

```
1 with open('Textdatei.txt','r') as datei:  
2     for zeile in datei:  
3         print("Inhalt aus Datei:", zeile, end="!")
```

Bei jedem Schleifendurchlauf wird eine Zeile aus der Datei geholt und in die Variable `zeile` geschrieben. Bei der Ausgabe kann es sinnvoll sein, den Zeilenumbruch von `print` zu verhindern, da jede Zeile aus der Datei bereits einen eigenen Zeilenumbruch mitbringt.

Aufgabe 1:

Nachdem im Programm `temperaturwerte.py` alle Werte eingegeben wurden, soll zum Schluß der komplette Inhalt der Datei ausgegeben werden.

Hilfe: Für den Sprung an den Dateianfang dient die Funktion: `seek(0)` (`datei.seek(0)`).

Datenaustausch

Damit unterschiedliche Programme Daten austauschen und verarbeiten können, gibt es semi-strukturierte Datenformate. Hier wird die Information, um was es sich bei den Daten handelt mitgeliefert.

1. Dateiformat **CSV**

– eines der am häufigsten verwendeten Formate zum Darstellen, Importieren und Exportieren von Tabellendaten. CSV wird im Allgemeinen zwischen Programmen verwendet, die normalerweise keine Daten austauschen können. Merkmale von CSV:

- Daten werden schlank und leicht lesbar dargestellt
- jeder Datensatz hat exakt den gleichen Aufbau
- die erste Zeile enthält keine Werte, sondern die **Spaltenbezeichnungen**

```
1 "Anlage", "GwhNr", "Temp1", "Temp2", "Temp3", "luftf" Gewaechshaus.csv  
2 "MA", "2", "23.15", "27.99", "32.22", "52"  
3 "HD", "1", "25.15", "21.89", "23.72", "63"  
4 "LU", "2", "21.15", "24.17", "31.42", "52"  
5 "HD", "3", "25.17", "31.26", "26.19", "60"
```

```

1 import csv
2 with open("Gewaechshaus.csv") as datei:
3     csv = csv.DictReader(datei)
4     for zeile in csv:
5         print(zeile)
6         print(zeile['Anlage'])

```

csv_auslesen.py

- 3: Der CSV.DictReader ordnet jedem Wert den passenden Spaltennamen als Schlüssel zu.
- 4,5: Jeder Datensatz wird mit Schlüsseln und Werte in einem Dictionary (alphanumerischem Array) gespeichert.
- 6: Die Werte können in der [] Klammer angesprochen werden.

Ausgabe:

```

>>> %Run csv_auslesen.py
{'Anlage': 'MA', 'GwhNr': '2', 'Temp1': '23.15', 'Temp2': '27.99', 'Temp3': '32.22', 'luftf': '52'}
MA

```

- 8,9: Jede Zeile wird als Schlüssel / Wert- Kombination ausgelesen. items() gibt eine Liste der Schlüssel-Wert- Paare zurück.

csv_auslesen.py

```

1 import csv
2 with open("Gewaechshaus.csv") as datei:
3     csv = csv.DictReader(datei)
4     for zeile in csv:
5         #print(zeile)
6         #print(zeile['Anlage'])
7         print("Neue Zeile:")
8         for schlüssel, wert in zeile.items():
9             print(schlüssel, '->', wert)

```

Ausgabe:

Aufgabe 2:

Werten Sie die Datei *Gewaechshaus.csv* aus: es soll der Durchschnittswert der Luftfeuchte der Anlagen in Heidelberg (HD) ermittelt werden.

```

Neue Zeile:
Anlage -> HD
GwhNr -> 3
Temp1 -> 25.17
Temp2 -> 31.26
Temp3 -> 26.19
luftf -> 60

```

2. Datenformat JSON

JavaScript Object Notation. Ähnlich wie CSV ist es ein lesbares Format, mit dem Daten gespeichert werden können. Es lassen sich relativ komplexe Datenstrukturen abbilden, die dabei noch gut lesbar sind.

Alle Informationen liegen als Schlüssel-Wert Paare vor. Der Wert eines Schlüssel-Wert-Paares kann ein einzelner Wert oder wiederum eine Listenart sein.

Syntax:

Allgemein	Objekt	Objekt mit Array
<pre>{ KEY1: VALUE1, KEY2: VALUE2, KEY3: VALUE3 }</pre>	<pre>{ "name": "Franz", "alter": 27, "hobby": "Python" }</pre>	<pre>{ "name": "Karin", "alter": 26, "hobby": ["Python", "Jaca", "C++"] }</pre>

Beispiel 1: *tel.json*

```
1 {
2   "Tom": ["0172 567 343", "03202 67231"],
3   "Anna": "",
4   "Tina": "0201 89755"
5 }
```

```
1 import json
2 with open('tel.json','r') as datei:
3     telefon = json.load(datei)
4     for name in telefon.keys():
5         print(name, telefon[name])
6
7 print("\n", telefon)
```

- 3: load() lädt die Datei und konvertiert sie in ein Python-Format
- 4: Die Schlüssel werden in *name* gespeichert
- 5: Schlüssel und der passende Wert zum Schlüssel werden ausgegeben

Beispiel 2:

```
1 import json
2
3 element = {'Anlage': 'HD', 'GwhNr': '3',
4 'Temp1': '25.17', 'Temp2': '26.17', 'Temp3': '25.77',
5 'luftf': '60'}
6
7 print(json.dumps(element, indent=4))
8
9 with open('text_json.json','w') as datei:
10     datei.write(json.dumps(element, indent=4))
```

json_1.py

Die Methode *dumps()* gibt das erzeugte JSON-Element zurück. Als Parameter werden die Daten und die Tiefe der Einrückung übergeben.

```
>>> %Run json_1.py
{
    "Anlage": "HD",
    "GwhNr": "3",
    "Temp1": "25.17",
    "Temp2": "26.17",
    "Temp3": "25.77",
    "luftf": "60"
}
```