JSON und YAML Merkblatt

JSON

JSON ist die Abkürzung für **JavaScript Object Notation** und ein weit verbreitetes Datenformat, welches von allen populären Programmiersprachen unterstützt und eingesetzt wird.

JSON Datentypen

Alle in JavaScript verwendeten Datentypen können somit auch im JSON-Format genutzt und ineinander veschachtelt werden.

Nullwert

Wie in JavaScript ist der Bezeichner für ein nicht vorhandenes Objekt **null**.

Bool'sche Werte

Bool'sche Werte werden mit **true** und **false** repräsentiert.

Zahlen

Zahlen sind entweder per Deziamlpunkt getrennte Dezimalzahlen oder einfache Integerwerte. Dezimalzahlen können über ein negatives Vorzeichen verfügen. Exponenten werden mit **e** eingeleitet, gefolgt von dem Exponenten mit oder ohne Vorzeichen.

Strings

Strings werden in doppelte oder einfache Anführungszeichen eingeschlossen. Sämtliche Unicode-Zeichen sind zulässig. Escapesequenzen beginnen mit \.

Listen

Eine Liste in JSON ist eine Aneinanderreihung von durch Komma getrennten Elementen beliebigen Typs, die durch eckige Klammern eingeschlossen werden:

```
[null, 1, 'zwei', true]
```

Assoziative Listen

Assoziative Listen sind Listen von Schlüssel-Wert-Paaren, wobei der Schlüssel immer ein String sein muss. Assoziative Listen werden durch geschweifte Klammern eingeschlossen, Schlüssel und Wert durch **:** getrennt und Schlüssel-Wert-Paare mit Komma aneinandergereiht:

```
{'Name': 'Paul', 'Alter': 42}
```

JSON Beispiel

Diese Datenstruktur enthält alle in **JSON** gültigen Datentypen:

```
{
  "Name": "Kurt Kleber",
  "lebt": true,
  "Alter": 27,
  "Ponies": ['Kalle', 'Klaudia', 'Kim']
  "Adresse": {
    "Strasse": "Ponystrasse 1",
    "Stadt": "Ponyhausen",
  },
  "Ehefrau": null
}
```

JSON in Python

Ein Modul zum Umgang mit JSON-Daten ist Teil der Python Standardbibliothek und muss daher nicht extra installiert werden.

Import

```
import json
```

Python zu JSON

So wird eine Python-Datenstruktur zu einem JSON-String formatiert:

```
json_string = json.dumps(['beliebige',
    {'daten': ('struktur', None, 1.0, 2)}])
```

Achtung: Die Schlüssel der Python-Dictionaries dürfen nur vom Typ String sein!

JSON zu Python

Folgendermaßen wird ein JSON-String in eine Python-Datenstruktur umgewandelt:

```
python_struktur = json.loads(
    "a': [1, null, true, 'b']')
```

YAML

Insbesondere Konfigurationsdateien liegen häufig im **YAML**-Format vor, wobei **YAML** für **YAML Ain't Markup Language** steht. Ein Vorteil von **YAML** ist die für Menschen einfache Lesbarkeit der vorliegenden Daten.

YAML Grundlagen

Wie auch **JSON** unterstützt **YAML** Skalare (Einzelwerte), Listen und assoziative Listen, die beliebig ineinander geschachtelt werden können.

Skalare

YAML verfügt über die Datentypen bool'scher Wert, Zahlenwert und String, wobei Zahlenwerte die wissenschaftliche Exponentennotation mit **e** wie in **JSON** auch unterstützen. Bool'sche Werte sind **true** oder **false**. Strings werden in einfache oder doppelte Hochkommata gesetzt. Jedes Skalar in einer YAML-Datei hat einen Bezeichner gefolgt von einem Doppelpunkt und dem zugehörigen Wert.

```
hallo: "Welt"
wahr: true
zahl: 3.1e-3
```

Listen

Listen werden mit einem Bezeichner gefolgt von einem Doppelpunkt und zeilenweise eingerückten, mit führendem Minuszeichen versehenden Listenelementen dargestellt:

```
liste:
  - "eins"
  - 2
  - true
```

Ausserdem können Listen in einer Inline-Notation mit eckigen Klammern definiert werden:

```
liste: [eins, 2, true]
```

Dabei kann auf Anführungszeichen bei Strings verzichtet werden.

Assoziative Listen

Assoziative Listen sind Schlüssel-Wert-Paare. Auf den Namen der Liste folgen zeilenweise die eingerückten Schlüssel-Wert-Paare, mit einem Stringbezeichner ohne Anführungszeichen gefolgt von einem Doppelpunkt gefolgt von dem Wert:

```
assoziative_liste:
  name: 'john'
  alter: 42
```

In Inline-Notation mit geschweiften Klammern:

```
assoziative_liste: {name: john, alter: 42}
```

Weitere YAML Syntax

Kommentare

Kommentare werden in **YAML** mit einem vorangestellten Rautezeichen eingeleitet:

```
# Ein Kommentar
```

Multiple Dokumente

YAML bietet die Möglichkeit, mehrere Dokumente in einer einzigen Datei abzulegen. Diese werden mittels dreier Minuszeichen unterteilt:

```
name: john
---
name: terry
```

Dabei darf in der Zeile nach dem Trennzeichen kein Kommentar stehen!

Sonderzeichen

Zur Darstellung der Sonderzeichen **' [] {} : > | '** muss der String in Hochkommata stehen, beispielsweise

```
ein_string: "[so]"
```

YAML in Python

Installation

Die Installation der **YAML**-Bindings in Python erfolgt mittels **pip**:

```
pip3 install pyyaml
```

Import

```
import yaml
```

Python zu YAML

```
yaml_string = yaml.dump(
    {'liste': ['eins', 2]})
```

Achtung: Wie bei **JSON** müssen die Schlüssel der Python-Dictionaries vom Typ String sein!

YAML zu Python

```
python_struktur = yaml.load(
    '[eins, zwei]')
```

DATAMICS
machine intelligence consulting services