JSON - Dateien lesen und schreiben

JavaScrip Object Notation ist ein Text-basiertes Datenformat. Es ist einfacher und kompakter als XML und wird beinahe überall unterstützt. Ein Python ist der resultierende Datentyp ein Dictionary, welches i.d.r. wieder unter-Dictionaries enthält.

Weil wir es können, schreiben wir ein JSON-File als normalen Text gerade frisch auf die Platte:

In [2]:
```
json_string = """{
  "Herausgeber": "Xema",
  "Nummer": "1234-5678-9012-3456",
  "Deckung": 2e+6,
  "Waehrung": "EURO",
  "Inhaber":
  {
    "Name": "Mustermann",
    "Vorname": "Max",
    "maennlich": true,
    "Hobbys": ["Reiten", "Golfen", "Lesen"],
    "Alter": 42,
    "Kinder": [],
    "Partner": null
  }
}"""

with open('Data/.2_test.json', 'w') as output_file:
    output_file.write(json_string)
```

Das Einlesen einer JSON-Datei erfolgt mit `json.load(file_object)`

In [3]:
```
import json

json_data = None
with open('Data/.2_test.json', 'r') as input_file:
    json_data = json.load(input_file)

print(json_data)
```

```
{'Herausgeber': 'Xema', 'Nummer': '1234-5678-9012-3456', 'Deckung': 2000000.
0, 'Waehrung': 'EURO', 'Inhaber': {'Name': 'Mustermann', 'Vorname': 'Max', 'm
aennlich': True, 'Hobbys': ['Reiten', 'Golfen', 'Lesen'], 'Alter': 42, 'Kinde
r': [], 'Partner': None}}
```

Da wir in diesem Beispiel die JSON-Daten bereits in einem String haben, ginge es natürlich auch direkt mit json.load**s**(string), ohne 'Umweg' über eine Datei.

In [4]:
```
json_data = json.loads(json_string)
print(json_data)
```

```
{'Herausgeber': 'Xema', 'Nummer': '1234-5678-9012-3456', 'Deckung': 2000000.
0, 'Waehrung': 'EURO', 'Inhaber': {'Name': 'Mustermann', 'Vorname': 'Max', 'm
aennlich': True, 'Hobbys': ['Reiten', 'Golfen', 'Lesen'], 'Alter': 42, 'Kinde
r': [], 'Partner': None}}
```

Die Ausgabe von Dictionaries in JSON-Dateien oder in strings erfolgt mit json.dump(dict, file) resp. json.dump**s**(dict)

In [5]:
```
with open('Data/.2_output.json', 'w') as output_file:
    json.dump(json_data, output_file, indent=4)

json_string_compact = json.dumps(json_data)
json_string_pretty = json.dumps(json_data, indent=4)
print(f"Compact: {json_string_compact}")
print(f"\nPretty: \n{json_string_pretty}")
```

```
Compact: {"Herausgeber": "Xema", "Nummer": "1234-5678-9012-3456", "Deckung":
2000000.0, "Waehrung": "EURO", "Inhaber": {"Name": "Mustermann", "Vorname": "
Max", "maennlich": true, "Hobbys": ["Reiten", "Golfen", "Lesen"], "Alter": 4
2, "Kinder": [], "Partner": null}}

Pretty:
{
    "Herausgeber": "Xema",
    "Nummer": "1234-5678-9012-3456",
    "Deckung": 2000000.0,
    "Waehrung": "EURO",
    "Inhaber": {
        "Name": "Mustermann",
        "Vorname": "Max",
        "maennlich": true,
        "Hobbys": [
            "Reiten",
            "Golfen",
            "Lesen"
        ],
        "Alter": 42,
        "Kinder": [],
        "Partner": null
    }
}
```

indent=4 sorgt dafür, dass die Ausgabe Zeilenumbrüche erhält und so besser lesbar für Menschen wird. Wenn JSON nur dazu verwendet wird, Daten zu persistieren oder zwischen Maschinen auszutauschen, kann darauf verzichtet werden.