Daten sicher steuern

Spickzettel: list · dict · tuple · JSON



DIE ENTSCHEIDENDE REGEL

Eckige Klammern [ ] sind beim Zugriff nicht automatisch eine Liste!

Bei list steht in [ ] eine Position (0, 1, 2 ...); bei dict ein Schlüssel (z. B. "heli").

01 / [ ] LIST

Liste: Positionen

Mehrere Einträge in Reihenfolge.
Der erste Eintrag hat Index 0.

BEISPIEL: INVENTAR

```
inventar = []
inventar.append("Energiezelle")
print(inventar[0])  # Energiezelle
```

02 / { } DICT

Dictionary: Namen

Ein Schlüssel gehört zu einem Wert.
Zugriff über den Schlüsselnamen.

BEISPIEL: ENERGIEWERT

```
inventar = {"Energiezelle": 80}
print(inventar["Energiezelle"])
inventar["Energiezelle"] = 100
```

03 / ( ) TUPLE

Tuple: feste Werte

Wie eine Liste, aber nicht änderbar.
Praktisch z. B. für Koordinaten.

BEISPIEL: POSITION

```
position = (10, 20, 30)
print(position[0])  # 10
# position[0] = 50  -> Fehler
```

04 / JSON-DATEI LESEN

Helikopter konfigurieren

JSON speichert strukturierte Daten in einer Datei. json.load() liest sie in Python ein.

DATEI heli_config.json

```
{
  "heli": {"zugang_offen": false},
  "cockpit": {
    "navigation_online": false,
    "rotor_online": false
  }
}
```

PYTHON verschachteltes dict

```
import json
with open("heli_config.json") as datei:
    config = json.load(datei)
config["heli"]["zugang_offen"] = True
config["cockpit"]["navigation_online"] = True
config["cockpit"]["rotor_online"] = True
```

`config["cockpit"]` → wählt die Gruppe `["rotor_online"]` → das Feld

Wichtig: json.load() liest nur. Änderungen speichern geht mit json.dump().

MERKHILFE **[ ] Liste · { } Dictionary · ( ) Tuple · JSON Datei-Format**

Bei Python heißt der Wahrheitswert True / False; in einer JSON-Datei true / false.