

Leitfaden: Dateien öffnen mit with, open, as, r und w

Lernübung – Python Grundlagen

Nalan Kara

30. August 2026

Überblick

```
with open("aufgaben.json", "r") as f:  
    inhalt = json.load(f)
```

Diese eine Zeile enthält vier Bausteine, die zusammengehören: `with`, `open(...)`, den Modus ("`r`" oder "`w`") und `as f`. In diesem Leitfaden wird jeder Baustein einzeln erklärt.

1. Was `open()` überhaupt macht

```
open("aufgaben.json", "r")
```

Ein normaler Funktionsaufruf mit **zwei Argumenten**:

1. der **Dateiname** ("`aufgaben.json`")
2. der **Modus** — wie du die Datei nutzen willst

`open()` verbindet dich mit der Datei und gibt dir ein sogenanntes **Datei-Objekt** zurück, mit dem du anschließend lesen oder schreiben kannst.

2. Was "`r`" und "`w`" bedeuten

Modus	Bedeutung	Was passiert
" <code>r</code> "	read (lesen)	Datei wird nur zum Lesen geöffnet. Schreiben ist nicht möglich.

Modus	Bedeutung	Was passiert
"w"	write (schreiben)	Datei wird zum Schreiben geöffnet. Der komplette bisherige Inhalt wird gelöscht und durch das Neue ersetzt.
"a"	append (anhängen)	Neuer Inhalt wird ans Ende angehängt, nichts wird gelöscht.

Beispiel aus dem eigenen Projekt:

```
def aufgaben_laden():
    with open("aufgaben.json", "r") as f:    # nur lesen
        return json.load(f)

def aufgaben_speichern(liste):
    with open("aufgaben.json", "w") as f:    # komplett neu schreiben
        json.dump(liste, f, indent=2)
```

`aufgaben_laden()` nutzt "r", weil hier nur der bestehende Inhalt abgerufen wird. `aufgaben_speichern()` nutzt "w", weil die **komplette aktuelle Liste** jedes Mal neu in die Datei geschrieben werden soll.

3. Warum man eine Datei nicht einfach offen lässt

Jede geöffnete Datei muss irgendwann wieder **geschlossen** werden — sonst drohen Datenverlust oder Zugriffsprobleme für andere Programme.

Die umständliche, manuelle Variante:

```
f = open("aufgaben.json", "r")
inhalt = json.load(f)
f.close()    # darf nicht vergessen werden!
```

Das Problem: Passiert zwischen `open()` und `close()` ein Fehler (z. B. eine Exception), wird `f.close()` **nie erreicht** — die Datei bleibt offen hängen.

4. Was with löst: automatisches Schließen

```
with open("aufgaben.json", "r") as f:
    inhalt = json.load(f)
# hier ist f schon automatisch geschlossen
```

`with` übernimmt das Schließen **automatisch** — garantiert, selbst wenn im eingerückten Block ein Fehler auftritt. Deshalb ist `with open(...) as f:` in echtem Python-Code der Standard-Weg, mit Dateien zu arbeiten. Die manuelle `open()/close()`-Variante sieht man kaum noch.

5. Warum `as f` nötig ist

`as f` bedeutet wörtlich: *“benenne das Ergebnis von `open(...)` mit dem Namen `f`”*.

`f` ist einfach eine Variable — der Name ist frei wählbar (häufig `f` für *file*, manchmal auch aussagekräftigere Namen). Alles, was innerhalb des Blocks mit der Datei gemacht werden soll (lesen, schreiben), läuft über diese Variable:

```
json.load(f)           # lesen
json.dump(liste, f)    # schreiben
```

Ohne `as f` gäbe es innerhalb des Blocks keine Möglichkeit, auf die geöffnete Datei zuzugreifen.

6. Die komplette Struktur zusammengefasst

```
with open(DATEINAME, MODUS) as VARIABLENNAME:
    # hier mit VARIABLENNAME arbeiten (lesen/schreiben)
# hier ist die Datei automatisch schon zu
```

Die **gesamte Zeile bis zum Doppelpunkt gehört zusammen** — genau wie bei `if ...:`, `for ...:` oder `while ...:`. Der eingerückte Block darunter ist der einzige Ort, an dem die Datei “offen” und nutzbar ist.

7. Häufiger Anfängerfehler

Falsch — `with` und `as` auf getrennte Zeilen aufgeteilt:

```
with open("aufgaben.json", "r"):
    as f:
    return json.load(f)
```

→ Führt zu `SyntaxError: expected ':'` bzw. `SyntaxError: invalid syntax`, weil Python `as f` als eigenständige, unvollständige Anweisung interpretiert.

Richtig — alles in einer zusammenhängenden Zeile:

```
with open("aufgaben.json", "r") as f:
    return json.load(f)
```

8. Merksatz

with open(Datei, Modus) as Variable: ist eine zusammenhängende Zeile. with kümmert sich automatisch ums Schließen. as Variable gibt dir den Zugriff auf die geöffnete Datei innerhalb des Blocks. "r" = nur lesen, "w" = überschreibend schreiben, "a" = anhängen.

9. Vollständiges Beispiel zum Nachvollziehen

```
import json

def aufgaben_laden():
    with open("aufgaben.json", "r") as f:
        return json.load(f)

def aufgaben_speichern(liste):
    with open("aufgaben.json", "w") as f:
        json.dump(liste, f, indent=2)

aufgaben = aufgaben_laden()

aufgaben.append("Neue Aufgabe")
aufgaben_speichern(aufgaben)

print(aufgaben_laden())
```

Ablauf Schritt für Schritt:

1. `aufgaben_laden()` öffnet die Datei im Lesemodus, liest den JSON-Inhalt, gibt ihn als Python-Liste zurück, schließt die Datei automatisch.
2. `.append(...)` verändert die Liste **nur im Arbeitsspeicher**.
3. `aufgaben_speichern(aufgaben)` öffnet dieselbe Datei im Schreibmodus, überschreibt sie komplett mit der aktuellen Liste, schließt sie automatisch.
4. Der letzte `print(aufgaben_laden())` beweist: Die Änderung ist jetzt **dauerhaft** in der Datei — nicht nur im Programm.