

Leitfaden: Mein erstes Python-CLI-Programm

Aufgaben-Manager im Terminal — Schritt für Schritt

Nalan Kara

30. August 2026

Ziel des Projekts

Ein kleines Terminal-Programm bauen, das ein Menü anzeigt und auf Nutzereingaben reagiert:

```
=== Aufgaben-Manager ===
```

1. Aufgabe hinzufügen
2. Aufgabe anzeigen
3. Beenden
4. Aufgaben löschen

Wähle eine Option:

Verwendete Grundlagen: Variablen, Listen, while-Schleife, if/elif, for-Schleife, input(), print(), Listen-Methoden (.append(), .remove()).

1. Projekt-Setup im Terminal

```
mkdir aufgaben-cli  
cd aufgaben-cli  
python3 -m venv venv  
source venv/bin/activate  
touch cli.py
```

Was diese Befehle tun:

Befehl	Bedeutung
mkdir	legt einen neuen Ordner an
cd	wechselt in den Ordner
python3 -m venv venv	erstellt eine isolierte Python-Umgebung nur für dieses Projekt
source venv/bin/activate	aktiviert diese Umgebung (erkennbar an (venv) vor der Zeile)
touch cli.py	erstellt eine leere Datei

Befehl	Bedeutung
--------	-----------

2. Setup in VS Code

1. **Ordner öffnen:** Datei → Ordner öffnen → aufgaben-cli auswählen
2. **Terminal öffnen:** Menü → Terminal → New Terminal
3. **Interpreter wählen:** Cmd/Strg + Shift + P → "Python: Select Interpreter" → den Eintrag mit ./venv/bin/python wählen
4. **Ausführen:** im Terminal python cli.py

Wichtig: Der Befehl **python** muss immer vor dem Dateinamen stehen (python cli.py), sonst meldet das Terminal "command not found" — die Datei ist ja kein eigenständiges Programm, sondern Code, den Python interpretieren muss.

3. Der fertige Code (Zeile für Zeile erklärt)

```
aufgaben = []
```

Eine leere Liste wird angelegt — das ist der "Speicher" für alle Aufgaben. Muss **vor** der Schleife stehen, sonst wäre sie bei jedem Durchlauf wieder leer.

```
while True:
```

Startet eine Schleife, deren Bedingung immer wahr ist — sie läuft unendlich, bis irgendwo break aufgerufen wird. Alles darunter eingerückte gehört zu dieser Schleife.

```
    print("=== Aufgaben-Manager ===")
    print("1. Aufgabe hinzufügen")
    print("2. Aufgabe anzeigen")
    print("3. Beenden")
    print("4. Aufgaben löschen")
```

Das Menü wird bei **jedem** Schleifendurchlauf neu angezeigt.

```
    auswahl = input("Wähle eine Option: ")
```

Das Programm hält an und wartet auf eine Tastatureingabe. Der eingegebene Wert wird **immer als Text (String)** gespeichert — auch Zahlen wie "1", nicht 1.

```
    if auswahl == "1":
        neue_aufgabe = input("Welche Aufgabe? ")
        aufgaben.append(neue_aufgabe)
        print("Hinzugefügt!")
```

Nur wenn `auswahl` genau "1" ist: neue Aufgabe abfragen, mit `.append()` ans **Ende** der Liste anhängen.

```
elif auswahl == "2":
    for aufgabe in aufgaben:
        print("-", aufgabe)
```

`elif` wird nur geprüft, wenn der `if` davor nicht zutraf. Die `for`-Schleife geht jedes Element der Liste einzeln durch und gibt es aus.

```
elif auswahl == "4":
    aufgabe_löschen = input("Lösche die Aufgabe: ")
    aufgaben.remove(aufgabe_löschen)
    print("Gelöscht!")
```

Fragt den **exakten** Text der zu löschenden Aufgabe ab. `.remove(x)` entfernt das erste Element, das genau `x` entspricht — bei Abweichung (Tippfehler, Groß-/Kleinschreibung) gibt es einen Fehler.

```
elif auswahl == "3":
    print("Bis bald!")
    break
```

Abschiedsnachricht, dann `break` — beendet die `while`-Schleife sofort. Das Programm läuft danach zu Ende.

4. Der komplette Code am Stück

```
aufgaben = []

while True:
    print("=== Aufgaben-Manager ===")
    print("1. Aufgabe hinzufügen")
    print("2. Aufgabe anzeigen")
    print("3. Beenden")
    print("4. Aufgaben löschen")

    auswahl = input("Wähle eine Option: ")

    if auswahl == "1":
        neue_aufgabe = input("Welche Aufgabe? ")
        aufgaben.append(neue_aufgabe)
        print("Hinzugefügt!")

    elif auswahl == "2":
        for aufgabe in aufgaben:
            print("-", aufgabe)
```

```

elif auswahl == "4":
    aufgabe_löschen = input("Lösche die Aufgabe: ")
    aufgaben.remove(aufgabe_löschen)
    print("Gelöscht!")

elif auswahl == "3":
    print("Bis bald!")
    break

```

5. Typische Stolperfallen (und wie du sie erkennst)

Problem	Ursache	Lösung
NameError: name 'aufgaben' is not defined	Variable wird benutzt, bevor sie definiert wurde	aufgaben = [] muss vor der ersten Nutzung stehen
IndentationError	Code, der zur Schleife gehören soll, ist nicht eingerückt	Betroffene Zeilen markieren (über die Zeilennummern klicken) und Tab drücken
Markierter Text verschwindet beim Einrücken	Mit Cmd+A alles markiert, statt ganze Zeilen	Über die Zeilennummern klicken/ziehen, nicht Cmd+A benutzen
elif wird als Fehler markiert	Zwischen if und elif steht anderer Code (z.B. eine fremde Schleife)	if/elif/elif müssen direkt aufeinanderfolgen
command not found: cli.py	python vor dem Dateinamen vergessen	Immer python cli.py schreiben, nicht nur cli.py
ValueError: list.remove(x): x not in list	Eingegebener Text stimmt nicht exakt mit dem Listen-Eintrag überein	Mit print(aufgaben) prüfen, was wirklich gespeichert ist

6. Merksätze zum Festhalten

- **pwd → ls → cd**: So orientierst du dich im Terminal.
- **while True + break**: Das Grundmuster für "läuft, bis der Nutzer aufhören will".
- **if / elif / elif**: Nur EIN Zweig läuft pro Durchlauf — sobald einer zutrifft, werden die anderen übersprungen.
- **.append()** fügt hinzu, **.remove()** entfernt ein bestimmtes Element — Gegenstücke zueinander.

- **Einrückung ist in Python keine Kosmetik**, sondern legt fest, was zu welchem Block (Schleife, if) gehört.
 - Bei jedem Fehler zuerst: **die letzte Zeile der Fehlermeldung lesen** — dort steht meist genau, was falsch ist.
-

7. Mögliche nächste Schritte

1. **Erweiterung mit bereits Gelerntem:** z.B. eine Zählung, wie viele Aufgaben insgesamt erledigt/offen sind.
2. **Daten dauerhaft speichern:** Aufgaben in eine Datei (aufgaben.json) schreiben, damit sie nach dem Beenden des Programms nicht verloren gehen (nächster Lernschritt: `with open(...), json.load()/json.dump()`).
3. **Argumente statt Menü:** Das Programm über `python cli.py add "Titel"` direkt steuern (nächster Lernschritt: Modul `argparse`).