

Leitfaden: Terminal-Projekte mit Python

Einkaufsliste, Notenrechner, Temperatur-Tracker — Code, Terminal-Befehle, Methoden

Nalan Kara

2. September 2026

1. Terminal-Befehle für jedes Projekt

Diese Befehle brauchst du bei **jedem** der folgenden Projekte gleichermaßen.

Befehl	Bedeutung
<code>mkdir ordnername</code>	legt einen neuen Ordner an
<code>cd ordnername</code>	wechselt in den Ordner
<code>touch datei.py</code>	erstellt eine leere Python-Datei
<code>ls</code>	zeigt, was im aktuellen Ordner liegt
<code>pwd</code>	zeigt den aktuellen Pfad
<code>python dateiname.py</code>	führt die Datei aus (Name muss exakt stimmen!)

Ablauf, den du für jedes Projekt wiederholst:

```
mkdir projekt-name
cd projekt-name
touch programm.py
python programm.py
```

Hinweis: “CLI” ist kein Befehl, sondern nur die Bezeichnung für das Terminal selbst (*Command Line Interface*). Dateinamen wie `cli.py` sind frei wählbar — keine Pflicht.

2. Übersicht: Verwendete Methoden & Funktionen

Diese Tabelle fasst alles zusammen, was in den drei Projekten vorkommt.

Methode/Funktion	Gehört zu	Wofür
<code>.append(x)</code>	Liste	fügt x ans Ende der Liste hinzu
<code>.remove(x)</code>	Liste	entfernt das erste Element, das exakt x entspricht
<code>len(liste)</code>	eingebaut	gibt die Anzahl der Elemente zurück
<code>sum(liste)</code>	eingebaut	addiert alle Zahlen der Liste
<code>min(liste)</code>	eingebaut	gibt den kleinsten Wert zurück
<code>max(liste)</code>	eingebaut	gibt den größten Wert zurück
<code>float(x)</code>	eingebaut	wandelt x (z. B. Text aus <code>input()</code>) in eine Kommazahl um
<code>input("Text")</code>	eingebaut	fragt eine Eingabe ab, gibt immer einen String zurück
<code>print(x)</code>	eingebaut	gibt x in der Konsole aus
<code>for x in liste:</code>	Kontrollstruktur	durchläuft jedes Element der Liste einzeln
<code>while True:</code>	Kontrollstruktur	wiederholt einen Block, bis break kommt
<code>if / elif</code>	Kontrollstruktur	führt nur EINEN passenden Zweig pro Durchlauf aus
<code>break</code>	Kontrollstruktur	beendet die umschließende Schleife sofort

Wichtigste Merksätze aus den Übungen: - Durchschnitt = `sum(liste) / len(liste)` (Summe geteilt durch Anzahl) - Eine Variable, die eine Liste enthält, und eine Variable für einen einzelnen neuen/zu löschenden Wert **müssen unterschiedliche Namen haben** — sonst überschreibt eine die andere. - Werte aus `input()` sind immer Text — für Zahlen-Vergleiche/-Rechnungen mit `float()` umwandeln, **bevor** sie in die Liste kommen. - `elif` reagiert nur auf einen exakten Fall, `else` fängt **alles andere** ab — bei einem festen Menü (z. B. "Beenden") ist `elif auswahl == "5"` meist richtiger als `else`.

3. Projekt 1: Einkaufsliste

Terminal-Setup

```
mkdir einkaufsliste-cli
cd einkaufsliste-cli
```

```
touch einkaufsliste.py
python einkaufsliste.py
```

Vollständiger Code

```
einkaufsliste = []

while True:
    print("=== Einkaufsliste ===")
    print("1. Artikel hinzufügen")
    print("2. Liste anzeigen")
    print("3. Artikel entfernen")
    print("4. Anzahl der Artikel anzeigen")
    print("5. Beenden")

    auswahl = input("Wähle eine Option: ")

    if auswahl == "1":
        auswahl_artikel = input("Welches Artikel? ")
        einkaufsliste.append(auswahl_artikel)
        print("-", auswahl_artikel)

    elif auswahl == "2":
        for artikel in einkaufsliste:
            print(artikel)

    elif auswahl == "3":
        artikel_löschen = input("Welcher Artikel? ")
        if artikel_löschen in einkaufsliste:
            einkaufsliste.remove(artikel_löschen)
            print("Entfernt:", artikel_löschen)
        else:
            print("Artikel nicht gefunden.")

    elif auswahl == "4":
        print(len(einkaufsliste))

    elif auswahl == "5":
        print("Bis bald!")
        break
```

Schritt-Hilfen

1. **Grundgerüst:** `einkaufsliste = []` **vor** der Schleife, sonst wird sie bei jedem Durchlauf zurückgesetzt.
2. **Hinzufügen:** `.append()` fügt an — braucht eine eigene Eingabe-Variable

(auswahl_artikel), nicht auswahl (die steuert das Menü!).

3. **Anzeigen:** `for artikel in einkaufsliste:` iteriert Element für Element.
 4. **Entfernen:** eigene Variable + optional vorher prüfen mit `if artikel_löschen in einkaufsliste:`, um den Fehler `x not in list` zu vermeiden.
 5. **Anzahl:** `len(einkaufsliste)` — direkt in `print()` einsetzen, sonst wird der Wert nirgends angezeigt.
 6. **Beenden:** `elif` (nicht `else`) + `break`, sonst reagiert "Beenden" auf jede beliebige falsche Eingabe.
-

4. Projekt 2: Notenrechner

Terminal-Setup

```
mkdir noten-cli
cd noten-cli
touch noten.py
python noten.py
```

Vollständiger Code

```
note = []

while True:
    print("== Notenrechner ==")
    print("1. Note hinzufügen")
    print("2. Alle Noten anzeigen")
    print("3. Note entfernen")
    print("4. Anzahl der Noten anzeigen")
    print("5. Durchschnitt berechnen")
    print("6. Beenden")

    auswahl = input("Wähle eine Option: ")

    if auswahl == "1":
        auswahl_note = float(input("Welche Note? "))
        note.append(auswahl_note)
        print("-", auswahl_note)

    elif auswahl == "2":
        for noten in note:
            print(noten)

    elif auswahl == "3":
        note_löschen = float(input("Welche Note entfernen? "))
```

```

    if note_löschen in note:
        note.remove(note_löschen)
        print(note_löschen)
    else:
        print("Note nicht gefunden.")

elif auswahl == "4":
    print(len(note))

elif auswahl == "5":
    durchschnitt = sum(note) / len(note)
    print(durchschnitt)

elif auswahl == "6":
    print("Beenden!")
    break

```

Schritt-Hilfen

1. **Neu gegenüber der Einkaufsliste:** Noten sind Zahlen, keine Texte — deshalb `float(input(...))` **direkt** bei der Eingabe, nicht erst später.
2. **Löschen:** Auch hier `float()` verwenden, sonst passt der eingegebene Text ("2.0") nicht zur gespeicherten Zahl (2.0) — führt sonst zum `remove: x not in list`-Fehler.
3. **Durchschnitt (der neue Kernpunkt):**
 - Summe: `sum(note)`
 - Anzahl: `len(note)`
 - Formel: `sum(note) / len(note)` — **nicht** durch eine feste Zahl oder durch die Liste selbst teilen.
4. **Falle, auf die man achten sollte:** Ist die Liste leer, wirft `sum(note) / len(note)` einen `ZeroDivisionError` — das ließe sich mit einer `if`-Prüfung vorher abfangen (optional, als Erweiterung).

5. Projekt 3: Temperatur-Tracker

Terminal-Setup

```

mkdir temperatur-cli
cd temperatur-cli
touch temperaturen.py
python temperaturen.py

```

Vollständiger Code

```
temperaturen = []

while True:
    print("== Temperatur Tracker ==")
    print("1. Temperatur eintragen")
    print("2. Alle Temperaturen anzeigen")
    print("3. Temperatur entfernen")
    print("4. Anzahl der Einträge anzeigen")
    print("5. Durchschnitt berechnen")
    print("6. Höchste Temperatur anzeigen")
    print("7. Niedrigste Temperatur anzeigen")
    print("8. Beenden")

    auswahl = input("Wähle eine Option: ")

    if auswahl == "1":
        temperatur = float(input("Nenne eine Temperatur "))
        temperaturen.append(temperatur)
        print("-", temperatur)

    elif auswahl == "2":
        for temp in temperaturen:
            print(temp)

    elif auswahl == "3":
        temperatur_löschen = float(input("Welche Temperatur soll entfernt werden? "))
        if temperatur_löschen in temperaturen:
            temperaturen.remove(temperatur_löschen)
            print(temperatur_löschen)
        else:
            print("Nicht gefunden.")

    elif auswahl == "4":
        print(len(temperaturen))

    elif auswahl == "5":
        print(sum(temperaturen) / len(temperaturen))

    elif auswahl == "6":
        print(max(temperaturen))

    elif auswahl == "7":
        print(min(temperaturen))

    elif auswahl == "8":
```

```
print("Beenden")
break
```

Schritt-Hilfen

1. **Namenskonflikt vermeiden:** Die Liste heißt temperaturen (Mehrzahl), ein einzelner neuer Wert heißt temperatur (Einzahl) — **niemals denselben Namen** für Liste und Einzelwert verwenden, sonst überschreibt der Einzelwert die ganze Liste.
2. **Neu:** max(temperaturen) und min(temperaturen) — funktionieren nach demselben Prinzip wie sum(), geben aber direkt den größten/kleinsten Wert zurück, ohne dass du selbst vergleichen musst.
3. **Gleiche Regeln wie beim Notenrechner:** float() bei jeder Eingabe, Durchschnittsformel sum()/len(), elif+break beim Beenden.

6. Häufige Fehler aus allen drei Projekten (zum Nachschlagen)

Fehler / Symptom	Ursache	Lösung
Liste ist nach jedem Menü-Durchlauf leer	liste = [] steht innerhalb von while True:	Nach ganz oben verschieben, vor die Schleife
AttributeError: 'float' object has no attribute 'append' list.remove(x): x not in list	Liste und Einzelwert haben denselben Variablennamen	Unterschiedliche Namen verwenden (z. B. Liste vs. neue_note)
TypeError bei sum()	Eingabetyp passt nicht (Text vs. Zahl) oder Tippfehler Liste enthält Text statt Zahlen	float() bei der Eingabe nutzen; vorher mit if x in liste: prüfen float(input(...)) schon bei der Eingabe verwenden
Ergebnis von len()/sum() erscheint nirgends	Berechnung in eigener Zeile, aber nicht ausgegeben	Direkt in print(...) einsetzen
“Beenden” reagiert auf jede falsche Eingabe command not found beim Ausführen	else: statt elif auswahl == "X": verwendet python vor dem Dateinamen vergessen, oder Tippfehler im Namen	elif mit dem exakten Wert nutzen Immer python dateiname.py, Namen exakt wie in ls angezeigt

7. Genereller Aufbau für neue, ähnliche Übungen

Dieses Muster kannst du auf beliebige neue Themen übertragen (z. B. Kontaktliste, Ausgabenrechner):

```
liste = []                                # 1. Speicher anlegen (vor der Schleife!)

while True:                              # 2. Endlosschleife mit Ausstieg
    print("Menü ...")                    # 3. Optionen anzeigen
    auswahl = input("...")              # 4. Nutzerwahl abfragen

    if auswahl == "1":                  # 5. Hinzufügen
        wert = input("...")             # ggf. float()/int() nicht vergessen
        liste.append(wert)

    elif auswahl == "2":                # 6. Anzeigen
        for element in liste:
            print(element)

    elif auswahl == "3":                # 7. Entfernen (eigene Variable!)
        zu_entfernen = input("...")
        if zu_entfernen in liste:
            liste.remove(zu_entfernen)

    elif auswahl == "X":                # 8. Beenden
        break
```