Python Lösungen – Schritt für Schritt erklärt

Zu jeder Aufgabe: Vorgehen (wie du drangehst) + Lösung (Code).

Tipp: Lie

1. Kleine Python-Programme

1.1 Nutzer begrüßen

Vorgehe

```
name = input("Wie heißt du? ")
```

```
print(f
```

1.2 Taschenrechner

Vorgehe
Dann m

```
a = float(input("Erste Zahl: "))
```

```
b = flo
```

1.3 Primzahl prüfen

Vorgehe
zahl-1 (

```
def ist_primzahl(zahl):
```

```
    if
```

1.4 Zahlen erraten

Vorgehe
Hinweis

```
import random
```

2. APIs

2.1 Nutzer von JSONPlaceholder holen

Vorgehe
Dicts) u

```
import requests
```

2.2 Wetterdaten (Open-Meteo, kein API-Key nötig)

Vorgehe
der Antv

```
import requests
```

2.3 Zufallswitz-Funktion

Vorgehe
als eige

```
import requests
```

2.4 API-Daten zusätzlich lokal speichern

Vorgehe

```
import requests
```

```
import
```

3. Daten verarbeiten

3.1 Summe, Durchschnitt, Min, Max ohne eingebaute Funktionen

Vorgehe
intern ar

```
zahlen = [4, 8, 2, 9, 5]
```

3.2 Längstes Wort + Anfangsbuchstaben zählen

Vorgehe
.get(key

```
woerter = ["Apfel", "Birne", "Ananas", "Kiwi"]
```

3.3 Produkte filtern & sortieren

Vorgehe

```
produkte = [
    {"na
```

3.4 CSV einlesen & Umsatz berechnen

Vorgehe
Strings

```
import csv
```

4. Dateien lesen/schreiben

4.1 Text in Datei schreiben

Vorgehe
geschlo

```
def notiz_schreiben(text):
    with
```

4.2 Datei zeilenweise lesen

Vorgehe
String.

```
def notiz_lesen():
    with
```

4.3 Tagebuch mit Anhängen

Vorgehe

```
from datetime import datetime
```

4.4 JSON schreiben & lesen

Vorgehe
Python-

```
import json
```

5. Fehlerbehandlung

5.1 Erneut fragen bei ungültiger Zahl

Vorgehe
weiter.

```
while True:
    try
```

5.2 Sichere Division

Vorgehe

```
def sichere_division(a, b):
    try
```

5.3 Datei-Fehler abfangen

Vorgehe
generisc

```
try:
    wit
```

5.4 Eigene Exception-Klasse

Vorgehe
verletzt

```
class UngueltigesAlterError(Exception):
    pas
```

6. Funktionen und Klassen

6.1 Palindrom prüfen

Vorgehe

```
def ist_palindrom(text):
    tex
```

6.2 Klasse Auto mit Altersberechnung

Vorgehe

```
from datetime import datetime
```

6.3 Klasse Warenkorb

Vorgehe

berechn

```
class Warenkorb:
```

```
    def
```

6.4 Vererbung: Mitarbeiter → Manager

Vorgehe
Konstru

```
class Mitarbeiter:
```

```
    def
```

7. Listen/Dictionaries

7.1 Duplikate entfernen ohne `set()`

Vorgehe

```
def ohne_duplikate(liste):
```

```
    erg
```

7.2 Liste umdrehen ohne `.reverse()`

Vorgehe

```
def liste_umdrehen(liste):
    erg
```

7.3 Zwei Listen zu Dictionary

Vorgehe
um.

```
namen = ["Anna", "Ben", "Cem"]
alter =
```

7.4 Wörter nach Länge gruppieren

Vorgehe
vorhand

```
woerter = ["Hut", "Apfel", "Bus", "Sonne"]
```

8. Einfache Algorithmen

8.1 Bubble Sort

Vorgehe
sind. Na

```
def bubble_sort(liste):
    n =
```

8.2 Binäre Suche

Vorgehe
rechte H

```
def binaere_suche(liste, ziel):
    lin
```

8.3 Fibonacci-Folge

Vorgehe

```
def fibonacci(n):
    fol
```

8.4 Anagramm prüfen

Vorgehe

```
def sind_anagramme(wort1, wort2):
    ret
```

8.5 Fakultät (rekursiv)

Vorgehe
mit klein

```
def fakultaet(n):
    if
```

9. Datenbanken (SQLite)

9.1 Datenbank + Tabelle erstellen

Vorgehe
String ü

```
import sqlite3
```

9.2 Person hinzufügen

Vorgehe
immer c

```
def person_hinzufuegen(name, alter):
    cur
```

9.3 Alle Personen anzeigen

Vorgehe

```
def alle_personen_anzeigen():
    cur
```

9.4 Löschen & Aktualisieren

```
def person_loeschen(id):
    cur
```

9.5 Verknüpfte Tabelle mit JOIN

Vorgehe
Tabeller

```
cursor.execute("""
    CRE
```

10. Kleine Webanwendungen (FastAPI)

10.1 Einfacher Endpunkt

Vorgehe
wird. Rü

```
from fastapi import FastAPI
```

10.2 Todo-API (im Speicher)

Vorgehe
entgege

```
from fastapi import FastAPI
```

```
from py
```

10.3 Mit SQLite verbinden

Vorgehe
persone

```
import sqlite3
```

```
from fa
```

10.4 Als erledigt markieren

Vorgehe

```
@app.patch("/todos/{id}")
```

```
def tod
```

10.5 Einfaches HTML-Frontend

Vorgehe
.map()/S

```
<div id="liste"></div>
```

11. Debugging

11.1 Fehler bewusst einbauen und lesen

Vorgehe
was ger

```
def begruessung(nam):   # Tippfehler: sollte 'name' heißen
    ret
```

→ Die Fehlermeldung nennt exakt Zeile und Variable. Erste Anlaufstelle bei jedem Fehler.

11.2 Print-Debugging

Vorgehe
enthält,

```
def summe_liste(liste):
    sum
```

11.3 Python-Debugger (`breakpoint()`)

Vorgehe
Variable

```
def berechne(a, b):
    erg
```

Im Debugger-Prompt: p ergebnis (Wert anzeigen), n (nächste Zeile), c (weiterlaufen lassen).

11.4 Fehler nur durch Lesen finden

Vorgehe
jedem Z

```
def bubble_sort(liste):
    n =
```

→ Bei j = n - i - 1 (letzter Wert der Range) wird auf liste[j + 1] zugegriffen, was aber außerhalb der Liste liegt. Klassischer Off-by-one-Fehler.

Merksatz zum Vorgehen bei jeder neuen Aufgabe

1. Verst

Wenn du bei einer der Lösungen etwas nicht verstehst oder es lieber gemeinsam nochmal Zeile für Zeile durchgehen willst, sag einfach welche.