

## **10 Praktische Offline-Programmier-Tipps**

### **1. Kommentiere wie fuer dein Zukunfts-Ich**

- Beschreibe, was der Code macht.
- Erkläre auch warum du es so machst.

### **2. Plane echte Datei-Szenarien**

- Sonderzeichen, Leerzeichen, Dateiendungen beachten.
- Verwende `os.path` und Fehlerprüfungen.

### **3. Logik fuer Wiederholung und Undo einbauen**

- Erstelle Backup-Kopien oder Log-Dateien.
- Ermöglicht späteres Rückgängigmachen.

### **4. Baue dir kleine Werkzeuge zuerst**

- Tools wie `get_all_files_by_ext` oder `convert_image` zuerst schreiben.
- Sparten dir später viel Zeit im GUI-Code.

### **5. Mache regelmässige Mini-Tests**

- Arbeite mit Checklisten.
- So findest du Fehler bevor sie kompliziert werden.

### **6. Halte Abhaengigkeiten minimal**

- Verwende Standardbibliotheken oder stabile Module (`tkinter`, `os`, `pydub`, `Pillow`).
- Vermeide schwer installierbare Pakete.

### **7. Teste mit Dummydaten**

- Erstelle Testdateien wie `test1.mp3`, `leer.mp3`, `fehler.wav`.
- Ideal zum schnellen Offline-Testen.

### **8. Denke in Modulen statt App**

- GUI, Konvertierung, Fortschrittsbalken einzeln behandeln.

## **10 Praktische Offline-Programmier-Tipps**

- Später verknuepfen.

### **9. Fuehre ein Programmier-Tagebuch**

- Notiere, was du gemacht, getestet oder beobachtet hast.
- Hilft nach Pausen sehr.

### **10. Erstelle dein eigenes Offline-Wiki**

- Halte Dokus, Snippets, Fehlermeldungen lokal in Markdown oder Textdateien.
- Spart dir jedes Mal Googeln.