Python-Projekt: Backup-Skript mit Fehlerprotokollierung

Lernziele

- Verzeichnisse und Dateien in Python verwalten
- Dateien und Ordner mit Python kopieren
- Fehler behandeln und abfangen
- Logdateien erstellen und auswerten
- Ein echtes Mini-Tool Schritt für Schritt entwickeln

Schritt 1 – „Hallo Backup“

Theorie:
Bevor wir mit Dateiarbeit anfangen, lassen wir unser Programm einfach starten.

Beispielcode:

```
print("Backup-Programm gestartet!")
```

Übung:
1. Schreibe ein Programm, das beim Start „Backup gestartet“ und am Ende „Backup beendet“ ausgibt.
2. Ergänze dein Programm so, dass dein Name und das aktuelle Datum angezeigt werden. (Tipp: import datetime)

Schritt 2 – Eine Datei kopieren

Theorie:
Mit shutil.copy2 können wir Dateien inklusive Metadaten kopieren.

Beispielcode:

```
import shutil

source_file = r"C:\Users\Public\Documents\TestQuelle\test.txt"
dest_file = r"D:\Backup\TestBackup\test.txt"

shutil.copy2(source_file, dest_file)
print("Datei kopiert!")
```

Übung:
1. Passe den Code an und kopiere eine eigene Datei.
2. Erstelle eine zweite Kopie mit einem neuen Namen, z. B. test_kopie.txt.
3. Was passiert, wenn die Datei nicht existiert? Teste es!

Schritt 3 – Einen ganzen Ordner kopieren

Theorie:
Mit os.walk durchlaufen wir alle Unterordner und Dateien.

Beispielcode:

```
import os
import shutil

SOURCE_DIR = r"C:\Users\Public\Documents\TestQuelle"
DEST_DIR = r"D:\Backup\TestBackup"

for root, dirs, files in os.walk(SOURCE_DIR):
    rel_path = os.path.relpath(root, SOURCE_DIR)
    target_dir = os.path.join(DEST_DIR, rel_path)
    os.makedirs(target_dir, exist_ok=True)

    for file in files:
        source_file = os.path.join(root, file)
        dest_file = os.path.join(target_dir, file)
        shutil.copy2(source_file, dest_file)
        print(f"Kopiert: {source_file} -> {dest_file}")
```

Übung:
1. Erstelle in deinem Quellverzeichnis mehrere Unterordner mit Dateien.
2. Lasse dein Programm den gesamten Ordner kopieren.
3. Ergänze den Code so, dass nach jeder kopierten Datei eine Meldung „Fertig“ ausgegeben wird.

Schritt 4 – Fehler behandeln

Theorie:
Programme sollen nicht abstürzen, wenn ein Fehler auftritt. Mit try/except fangen wir Fehler ab.

Beispielcode:

```
try:
    shutil.copy2(source_file, dest_file)
    print(f"[OK] {source_file}")
except Exception as e:
    print(f"[ERROR] {source_file} konnte nicht kopiert werden: {e}")
```

Übung:
1. Provoziere einen Fehler (z. B. sperre eine Datei oder verwende eine falsche Quelle).
2. Ergänze dein Programm so, dass es nicht abbricht, sondern weitermacht.
3. Schreibe zusätzlich eine Meldung „Backup läuft weiter…“ nach jedem Fehler.

Schritt 5 – Logdatei erstellen

Theorie:
Ein Logfile dokumentiert den gesamten Ablauf. Jeder Durchlauf bekommt eine eigene Datei.

Beispielcode:

```
import datetime

log_dir = r"C:\BackupLogs"
os.makedirs(log_dir, exist_ok=True)
log_file = os.path.join(log_dir,
f"backup_{datetime.datetime.now():%Y-%m-%d_%H-%M-%S}.log")

def log(message):
  with open(log_file, "a", encoding="utf-8") as f:
    f.write(message + "\n")
  print(message)

log("Backup gestartet")
# hier Backup-Code einfügen
log("Backup beendet")
```

Übung:
1. Erstelle ein Logfile, das „Backup gestartet“ und „Backup beendet“ enthält.
2. Ergänze dein Programm so, dass jede erfolgreich kopierte Datei ins Log geschrieben wird.
3. Füge auch Fehler mit [ERROR] ins Log ein.
4. Teste dein Programm zweimal hintereinander – es sollten zwei verschiedene Logdateien entstehen.

Erweiterungsideen

- Erstelle eine Funktion backup_directory(src, dest), die das gesamte Backup übernimmt.
- Ergänze eine Eingabeaufforderung (input()), sodass der Benutzer Quell- und Zielordner selbst eingeben kann.
- Baue ein Menü mit zwei Optionen:
  1. „Backup starten“
  2. „Log anzeigen“