

Natürlich. Hier ist dein **detaillierter Lernzettel zu pathlib**, perfekt für dich als Python-Einsteiger mit dem Ziel, **alle Pfade, Dateien und Ordner professionell, stabil und offline mit pathlib zu verwalten**.

Lernzettel: pathlib – Ordner und Dateien in Python einfach & sicher verwalten

Was ist pathlib?

pathlib ist ein Modul in Python, das dir hilft, mit **Pfaden, Ordnern und Dateien zu arbeiten** – aber nicht als einfache Zeichenketten (str), sondern als **intelligente Objekte**.

Beispiel aus dem Alltag:

Stell dir vor, du willst in deinem Haus (Projektordner) einen **Ordner für jedes Familienmitglied** (Account) anlegen, z. B.:

```
projekt/
├── accounts/
│   ├── mama/
│   ├── papa/
│   └── ich/
```

Mit pathlib kannst du solche Ordner *automatisch* erstellen und auch in ihnen arbeiten, **ohne dich um Windows/Linux-Unterschiede zu kümmern**.

Grundlagen – Path verwenden

1. Path importieren

```
from pathlib import Path # Importiert die Hauptklasse Path aus pathlib
```

2. Einen Pfad definieren (aber noch nichts erstellen!)

```
pfad = Path("accounts") / "ich"
# Das ist wie: "accounts/ich"
# Der "/" verknüpft die Ordner sauber miteinander (nicht als Text!)
```

3. Einen Ordner erstellen

```
pfad.mkdir(parents=True, exist_ok=True)
# Erstellt den Ordner "accounts/ich"
```

```
# parents=True: erstellt auch accounts, wenn es den noch nicht gibt
# exist_ok=True: kein Fehler, wenn Ordner schon existiert
```

Ordnerstruktur aufbauen (z. B. Accountsystem)

Beispiel: Account-Ordner mit Unterordnern

```
account = Path("accounts") / "benutzer123"

(account / "logs").mkdir(parents=True, exist_ok=True)
# Erstellt: accounts/benutzer123/logs

(account / "configs").mkdir(parents=True, exist_ok=True)
# Erstellt: accounts/benutzer123/configs

(account / "otp").mkdir(parents=True, exist_ok=True)
# Erstellt: accounts/benutzer123/otp
```

So kannst du **strukturierte Accountordner** bauen wie:

```
accounts/
├── benutzer123/
│   ├── logs/
│   ├── configs/
│   └── otp/
```

Dateien erstellen und beschreiben

Textdatei schreiben

```
datei = account / "configs" / "info.txt" # Dateipfad
datei.write_text("Dies ist ein Account.") # Inhalt in Datei schreiben
```

Textdatei lesen

```
text = datei.read_text() # Holt den Inhalt zurück als String
```

Prüfen, ob etwas existiert

```
if datei.exists():
    print("Datei existiert.")

if (account / "otp").exists():
    print("OTP-Ordner vorhanden.")
```

JSON speichern (z. B. Konfigurationen)

Speichern:

```
import json
config_pfad = account / "configs" / "config.json"

daten = {"email": "max@example.com", "smtp": "smtp.example.com"}

config_pfad.write_text(json.dumps(daten, indent=4))
# Wandelt das dict in einen lesbaren JSON-String um und schreibt ihn in die
Datei
```

Laden:

```
if config_pfad.exists():
    inhalt = json.loads(config_pfad.read_text())
    # Liest die Datei, wandelt sie von JSON-Text zurück in ein dict
```

Auflisten von Ordnerinhalten

```
alle_accounts = Path("accounts").iterdir()
# Gibt alle Unterelemente im Ordner "accounts" zurück (als Path-Objekte)

for ordner in alle_accounts:
    if ordner.is_dir():
        print("Account gefunden:", ordner.name)
```

Datei/Ordner löschen (optional)

Achtung: `pathlib` löscht **nur leere Ordner**

```
(account / "otp" / "geheim.key").unlink()
# Löscht eine Datei

(account / "otp").rmdir()
# Löscht den leeren Ordner "otp"
```

Tipps für deine Anwendung

Ziel	Lösung mit <code>pathlib</code>
Accounts in Ordnern speichern	<code>Path("accounts") / accountname</code>
Unterordner automatisch anlegen	<code>.mkdir(parents=True, exist_ok=True)</code>
JSON als Config-Dateien speichern	<code>json.dump()</code> mit <code>.write_text()</code>
Dateien prüfen/lesen/schreiben	<code>.exists()</code> , <code>.read_text()</code> , <code>.write_text()</code>
Mehrere Accounts verwalten	<code>.iterdir()</code> auf dem Hauptordner



Merke dir

- `Path("xyz") / "abc"` ist die beste Methode für **Pfadkombinationen**
- `makedirs()` erstellt Ordner → `parents=True` ist wichtig!
- Alles basiert auf **Objekten**, nicht auf Text
- Plattformübergreifend sicher (kein Problem mit \ oder /)
- Für dein Projekt perfekt geeignet: mehrere Accounts, OTP-Dateien, Logs usw.

Wenn du willst, kann ich dir noch einen **Offline-Testplan** schreiben, mit dem du Schritt für Schritt das Verhalten von `pathlib` üben kannst.

Sag einfach Bescheid.