Arbeiten mit virtuellen Python-Umgebungen unter Windows Handout

Einführung: Warum virtuelle Umgebungen wichtig sind

Wenn du mit Python arbeitest, brauchst du oft unterschiedliche Versionen von Bibliotheken für verschiedene Projekte. Ohne virtuelle Umgebungen landen alle Pakete global auf deinem System. Das kann zu Versionskonflikten führen, z. B. wenn ein Projekt `pandas 1.4` und ein anderes `pandas 2.0` benötigt. Außerdem spart eine virtuelle Umgebung Speicherplatz auf Laufwerk C, weil nur das installiert wird, was du wirklich brauchst – an einem einzigen, isolierten Ort.

Vorteile:

- Keine Konflikte zwischen Projekten
- Schnelles Updaten oder Wechseln von Versionen
- Klarer Speicherort für alle Projektpakete
- Einfaches Weitergeben per `requirements.txt`

Was ist `pip`?

`pip` ist der Paketmanager von Python. Mit `pip` kannst du zusätzliche Bibliotheken wie `pandas` installieren, updaten oder entfernen. Jede virtuelle Umgebung enthält ihre eigene Kopie von `pip`.

Vorinstallierte Pakete anzeigen

```
pip list
```

Warum gibt es `python.exe` und `pip.exe` in jeder virtuellen Umgebung?

Wenn du `venv` erstellst, wird eine Kopie von `python.exe` und `pip.exe` in den `Scripts`-Ordner gelegt. So hat jedes Projekt seinen eigenen Python-Interpreter und Paketmanager – unabhängig von anderen Projekten oder der globalen Installation.

Vorbereitung: Terminal öffnen

Öffne unter Windows die **Eingabeaufforderung (CMD)** oder **PowerShell**:

1. Drücke **Win + R**, tippe `cmd` oder `powershell` ein und bestätige.
2. Navigiere in dein Projektverzeichnis mit `cd`.

1. Virtuelle Umgebung erstellen

```
mkdir mein_projekt
cd mein_projekt
python -m venv venv
```

2. Virtuelle Umgebung aktivieren (PowerShell-Fehler vermeiden)

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
venv\Scripts\activate
```

3. Python-Programm starten

```
python app.py
```

4. Pakete installieren mit pip (Beispiel mit pandas)

```
pip install pandas
pip install numpy matplotlib
```

5. Abhängigkeiten speichern

```
pip freeze > requirements.txt
```

6. Abhängigkeiten installieren

```
pip install -r requirements.txt
```

7. Virtuelle Umgebung deaktivieren

```
deactivate
```

Beispiel: Eine neue venv für dasselbe Projekt erstellen

Manchmal willst du eine zweite Umgebung für dasselbe Projekt anlegen – z. B. zum Testen anderer Abhängigkeiten.

So geht's:

```
python -m venv venv_test
venv_test\Scripts\activate
```

```
pip install -r requirements.txt
pip list
```

Zusammenfassung: Wichtige Windows-Befehle

Befehl	Beschreibung
`mkdir mein_projekt`	Neuen Ordner erstellen
`cd mein_projekt`	In Ordner wechseln
`python -m venv venv`	Neue virtuelle Umgebung erstellen
`venv\Scripts\activate`	Virtuelle Umgebung aktivieren
`Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned`	PowerShell-Sicherheit anpassen
`pip install paketname`	Neues Paket installieren
`pip list`	Installierte Pakete anzeigen
`pip freeze > requirements.txt`	Abhängigkeiten speichern
`pip install -r requirements.txt`	Abhängigkeiten aus Datei installieren
`python app.py`	Python-Skript ausführen
`deactivate`	Virtuelle Umgebung deaktivieren