FH | JOANNEUM
University of Applied Sciences

Von Tabellen zur Automatisierung

Einstieg in Python für Excel-User

Ergänzende Folien zum lokalen Python Setup

Raphaele Raab

FH JOANNEUM
University of Applied Sciences

DIGITAL INNOVATION HUB SÜD

Digital Innovation Hub Süd

- Der *Digital Innovation Hub Süd (DIH SÜD)* ist ein **Kompetenznetzwerk**, das **Klein- und Mittelbetriebe (KMU)** bei der **digitalen Transformation** mit Expertise, Vernetzung und Infrastruktur unterstützt.
- Services:
  - Durchführung von Informationsveranstaltungen
  - Aktivitäten der Innovations- und Technologieberatung
  - **Durchführung von Qualifizierungsmaßnahmen (Schulungen)**
  - Begleitung bei der Entwicklung von Innovationen

Regionen und Netzwerk

- Den KMUs steht dabei das regionale Forschungs- und Innovationssystem folgender Bundesländer zur Verfügung: Steiermark, Kärnten, Burgenland, Osttirol
- Das Netzwerk des DIH Süd umfasst **Hochschulen**, **Forschungszentren** und **Inkubatoren** aus den genannten Regionen.

Finanzierung

Themenbereiche von DIH Süd

- Produktions- und Fertigungstechnologien
- Digitale Sicherheit
- **Daten & Künstliche Intelligenz**
- Digitale Geschäftsmodelle & -prozesse
- Nachhaltigkeit & Kreislaufwirtschaft
- Arbeit der Zukunft & Humanressourcen

VORSTELLUNG

FH JOANNEUM
University of Applied Sciences

Mag. Raphaele Raab MSc

Ausbildung

- Lehramtstudium Mathematik
  (KFU Graz)
- Masterstudium
  "Data and Information Science"
  (FH JOANNEUM Graz)

Berufliche Aktivität

- Hochschullektor an der FH JOANNEUM

Fokus in Forschung und Lehre

Scripting, Machine Learning, Data Preprocessing, Mathematik

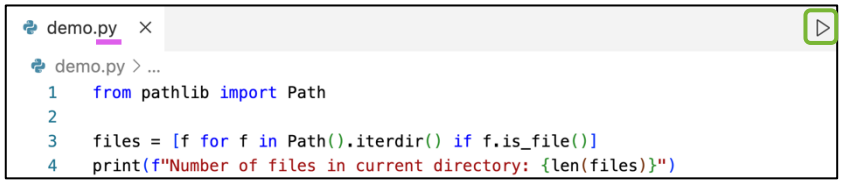
WICHTIGE BEGRIFFE

Programme, Skripte und Notebooks

- Ein **Programm** ist eine **Sammlung von Anweisungen** in einer bestimmten Programmiersprache, die der Computer ausführt, um eine bestimmte Aufgabe zu erledigen (z. B. Datenanalyse, Automatisierung).
- Ein Programm **besteht aus einem oder mehreren Skripten** (*.py*-Dateien) und weiteren Dateien (z.B. Konfigurationsdateien).
- Ein Skript ist ein einfaches Programm in einer einzelnen Datei – wird vereinfacht gesagt von oben nach unten ausgeführt.
- Ein **Jupyter Notebook** (*.ipynb*-Dateien) ist ein **interaktives Skript**, das Code, Ergebnisse und erklärenden Text kombiniert.

Skript

- Ein Skript-Datei in Python hat die Endung ***.py***.



- Um den darin enthaltenen Code auszuführen, klicken wir den **Run-Button** oder führen den Code mit direkt im Terminal mit folgendem Befehl aus:

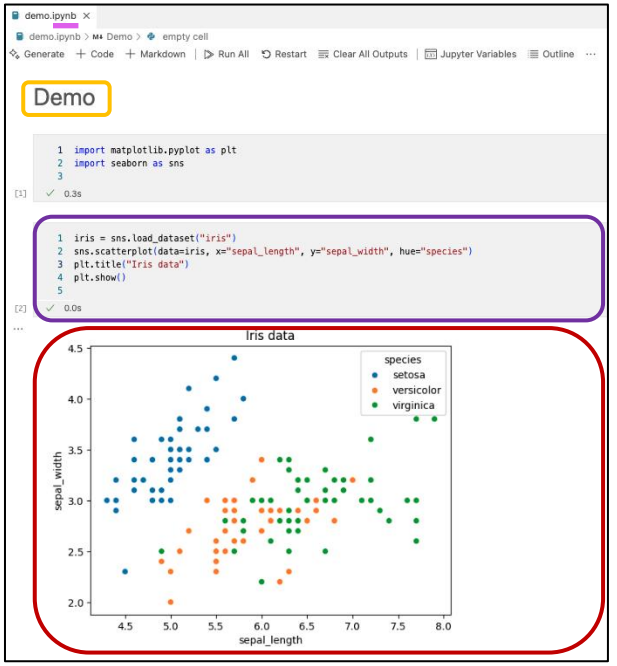
```
uv run <filename>
```

```
❯ uv run demo.py
```
← Terminal Input

```
Number of files in current directory: 4
```
← Terminal Output

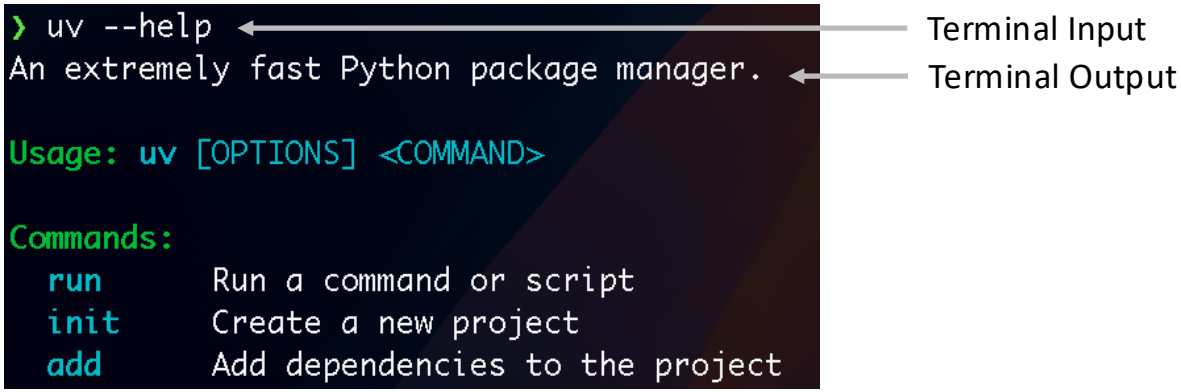
Juypter Notebook

- Ein Notebook hat die Endung ***.ipynb***.
- Es besteht aus mehreren Code-Zellen (**Input**).
- Der **Output** jeder Zelle wird direkt im Notebook angezeigt.
- **Text** wird in der Markup-Sprache *Markdown* geschrieben.



Terminal

- Terminal = Programm, das eine **textbasierte Schnittstelle** bereitstellt, über **die** man **Befehle an** den **Computer sendet**
- Shell = Software, die diese Befehle interpretiert und ausführt (z.B. *PowerShell* (*Windows*), *Bash*, …)



Wozu benötigt man ein Terminal?

- **Installation** von *Python* und *Python*-Paketen
- **Aktivierung von** sogenannten Virtual **Environments**
- **Ausführen von** *Python*-**Programmen**

Integrated Development Environment (IDE)

- Eine IDE ist ein Programm, das das Schreiben, Testen und Verwalten von Code erleichtert.
- Sie **kombiniert Funktionen** wie Terminalzugriff, Fehlermeldungen, Autovervollständigung und Debugger in einer Oberfläche.
- Wir können dafür **Visual Studio Code** (VS Code) als IDE **mit zusätzlichen Erweiterungen** (Extensions), insbesondere *Python* und *Jupyter*, verwenden.

Programmiersprache und Interpreter

- Jede Programmiersprache (z.B. Python) besitzt ein eigenes **Vokabular** (z.B. Schlüsselwörter wie `if`, `for`, `def`) und eine eigene **Grammatik** (Syntax).
- Damit der **Computer Code** in einer solchen Sprache **verstehen** und ausführen kann, braucht es einen **Interpreter** – ein Programm, das den Code Zeile für Zeile übersetzt und ausführt.
- Bei Python übernimmt das z.B. die Anwendung *python.exe* (unter *Windows*).

PYTHON INSTALLATION UND VIRTUAL ENVIRONMENTS

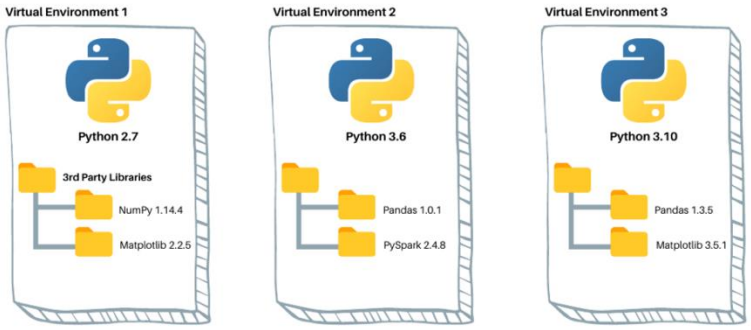
Herausforderungen bei der Installation

- Es existieren **verschiedene Python Versionen.**
- Es ist sinnvoll, sogenannte **Virtual Environments** zu **erstellen**.
- **Externe Python-Pakete** haben ebenfalls unterschiedliche Versionen und diese weisen **Abhängigkeiten** untereinander auf.

- Mit der Software **uv** lassen sich diese Herausforderungen effizient meistern.

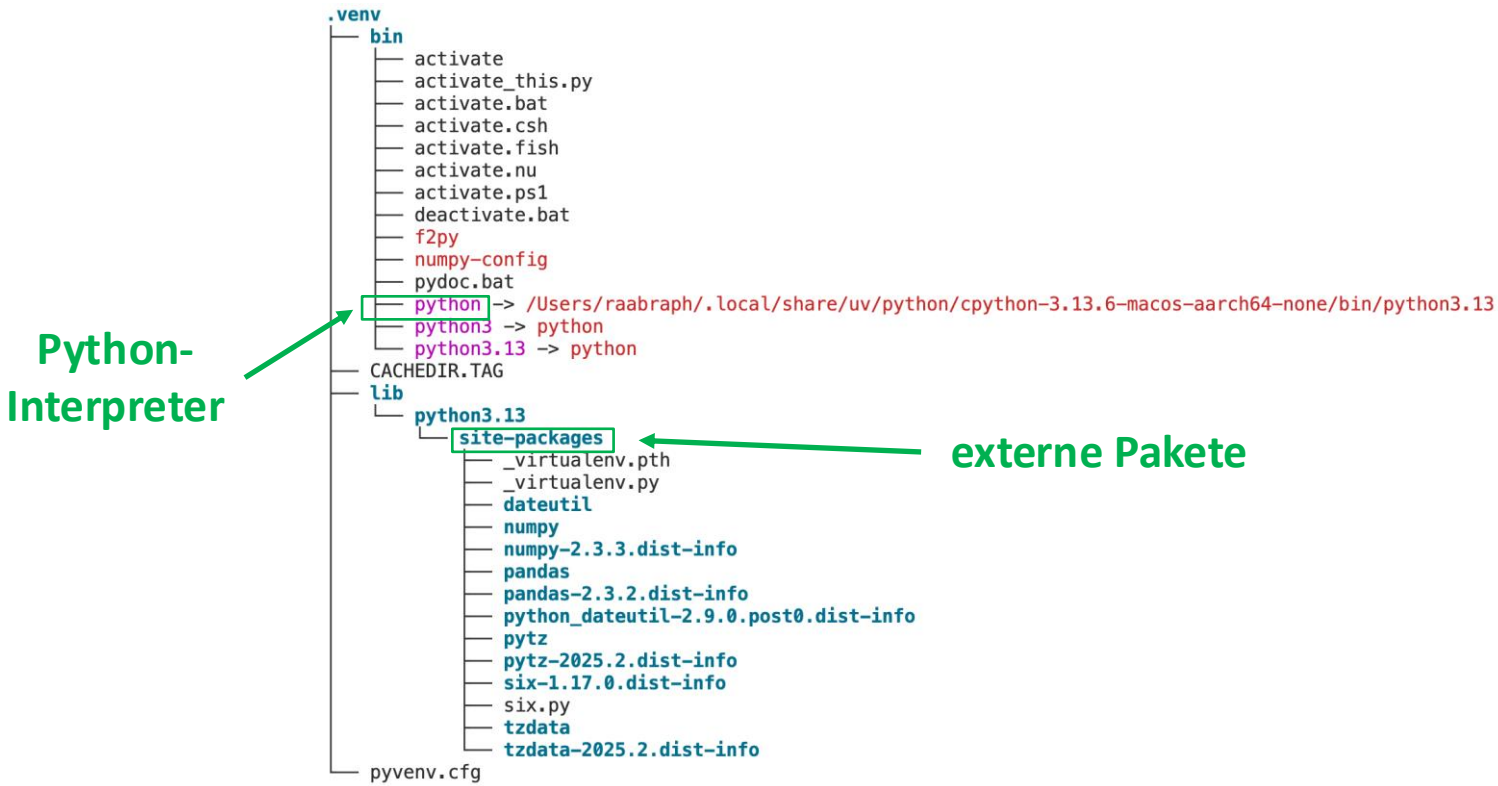
Virtual Environment

- = Ordnerstruktur (*.venv* Ordner), die alles Relevante beinhält um Python-Code ausführen zu können:
  Python-Interpreter (Windows: *python.exe*) und installierte **externe Python-Pakete** (z.B. *pandas*)



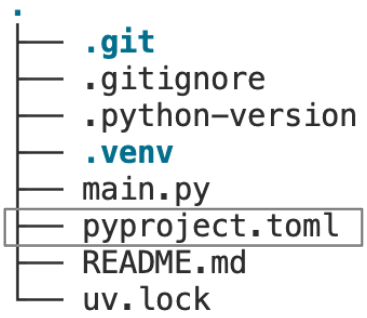
verschiedene Virtual Environments

Beispiel (MacOS)



Konfigurationsdateien

- Konfigurationsdateien (*pyproject.toml*, *uv.lock*) halten den Zustand eines solchen *Virtual Environments* fest.




```
[project]
name = "demo-uv"
version = "0.1.0"
description = "Add your description here"
readme = "README.md"
requires-python = ">=3.13"
dependencies = [
    "pandas>=2.3.2",
]
```

Python-Version

externe Python-Pakete

beispielhafte *pyproject.toml* Datei

Weshalb benötige ich ein *Virtual Environment*?

- unterschiedliche *Python*-Projekte erfordern unterschiedliche externe *Python*-Pakete
- komplexe Abhängigkeiten zwischen unterschiedlichen externen *Python*-Pakete
- *Linux* and *macOS* verfügen über eine vorinstallierte *Python*-Version, die bereits für interne Prozesse verwendet wird

→ Empfehlung:
Für jedes Projekt sollte ein eigenes Virtual Environment erstellt werden!

Installation uv

- Die Befehle zur Installation von **uv** finden unter https://docs.astral.sh/uv/.
- Kopiere den entsprechenden Befehl in die *PowerShell* (*Windows*) oder das *Terminal* (*macOS*).
- Um zu überprüfen, ob die Installation erfolgreich war, starte die *PowerShell* bzw. das *Terminal* neu und teste mit folgendem Befehl:

```
uv --help
```

Installation von Python mittels uv

- Mit folgendem Befehl (in *PowerShell* bzw. im *Terminal*) können wir eine bestimmte Python-Version installieren:

```
uv python install 3.13
```

Installation des Virtual Environments mittels uv

- Zur Installation eines Virtual Environments führen wir die folgenden beiden Befehle aus:

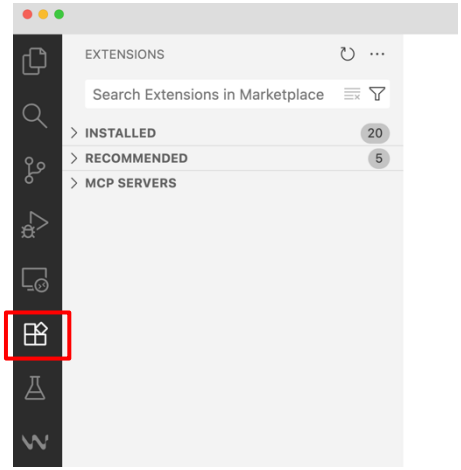
```
uv init --bare
```

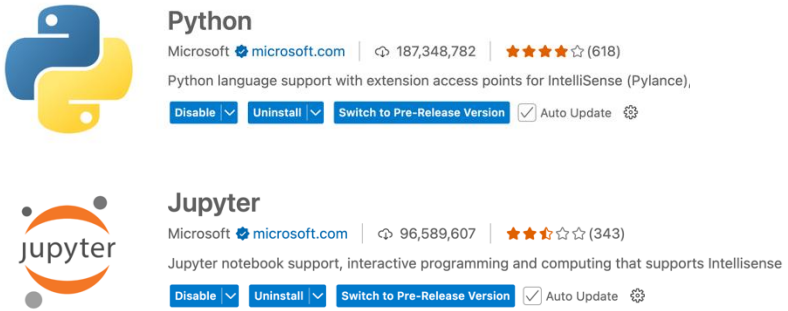
```
uv add <package-name>
```

- Damit erstellen wir die zuvor erwähnten Konfigurationsdateien, unser Virtual Environment (*.venv* Ordner) und fügen externe *Python*-Pakete hinzu. `<package-name>` muss durch die gewünschten Pakete (z.B. `ipykernel`, `pandas`) ersetzt werden.

VS Code Extensions

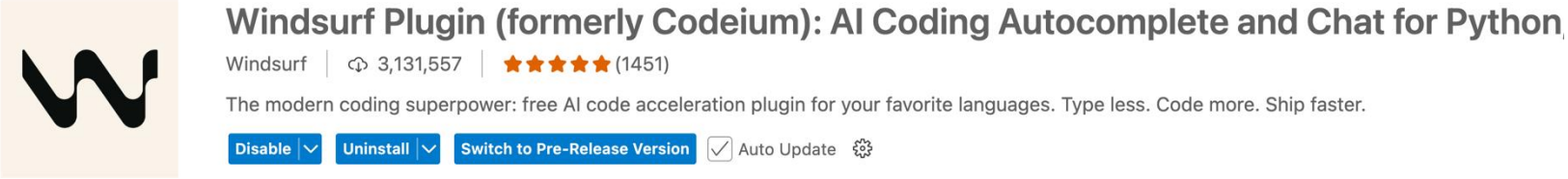
- In VS Code sollten auf jeden Fall die folgenden zwei **Extensions** installiert werden:





AI Coding Assistants

- Für die Verwendung von *GitHub Copilot* kann man folgender Anleitung folgen: https://code.visualstudio.com/docs/copilot/setup
- Sollte *Copilot* nicht funktionieren, kann man auch die Extension *Windsurf* verwenden.



WEITERE THEMEN

Code Formatter und Linter

- Ein **Formatter formatiert Code**, also z.B. passt Einrückungen, Abstände und Zeilenumbrüche an, damit der **Code einheitlich** und **gut lesbar** ist.
- Ein **Linter überprüft Code** auf Fehler oder Stilprobleme.
- Wir werden dafür *Ruff* verwenden.

Fehlerarten im Code

- **Syntax Error**
  Der Code verstößt gegen die Grammatik der Sprache – *Python* kann ihn nicht ausführen.

  `status = okay` *(fehlende Anführungszeichen)*

- **Runtime Error**
  Der Code ist grammatikalisch korrekt, bricht aber während der Ausführung ab.

  `1 / 0` *(Division durch 0)*

- **Logic Error**
  Der Code läuft ohne Fehlermeldung, liefert aber falsche Ergebnisse.

  z.B. Verwendung einer falschen Formel

Module und Pakete

- Als **Modul** bezeichnet man im Kontext von Python ein Python-Skript, das **von anderen Python-Skripten importiert** werden kann, um dessen Code mitzuverwenden.
- Ein **Paket** ist eine **Sammlung von Modulen**.
- Häufig werden nicht Module oder Pakete selbst geschrieben, sondern lediglich existierende verwendet (z.B. `pandas`, `matplotlib`, `streamlit`, `scikit-learn`).

Pfade und Working Directory

- Ein Pfad beschreibt den Ort einer Datei oder eines Ordners im Dateisystem.

  /Users/anna/project/data.csv (MacOS/Linux)

  C:\Users\Anna\project\data.csv (Windows)

- Das **Working Directory** (Arbeitsverzeichnis) ist der Ordner, **in dem ein Programm aktuell arbeitet** – also wo *Python* nach Dateien sucht oder neue erstellt.