Tag 1: Einrichten von Python-Umgebungen und Arbeiten mit Jupyter Notebooks

Ziel: Die Studierenden lernen, wie sie Python-Umgebungen erstellen, Abhängigkeiten verwalten und Jupyter Notebooks für interaktives Programmieren nutzen können.

Tagesablauf

Zeit	Thema	Beschreibung
9:00 - 9:30	**Python-Installation & Umgebungen**	Überprüfung der Python-Version, Installation und Umgebungen.
9:30 - 10:15	**Erstellen virtueller Umgebungen**	Hands-on: Erstellen und Aktivieren von `venv` und `conda`.
10:15 - 10:45	**Verwalten von Abhängigkeiten**	Installieren von Paketen mit `pip` und `conda`.
10:45 - 11:15	**Einführung in Jupyter Notebooks**	Nutzen von Jupyter für interaktives Programmieren.
11:15 - 12:00	**Arbeiten mit Jupyter-Zellen**	Code- und Markdown-Zellen in Notebooks verwenden.
12:00 - 12:30	**Grundlegende Python-Übungen**	Variablen, Schleifen und Funktionen in Jupyter.
12:30 - 13:00	**Q&A und Troubleshooting**	Fehlerbehebung und Klärung offener Fragen.

Detaillierter Plan

9:00 - 9:30: Python-Installation & Einführung in Umgebungen

Themen:

- Überprüfen, ob Python installiert ist:

  ```
  python --version
  ```

- Installation von Python (falls nötig) über python.org oder Anaconda.
- Einführung in **virtuelle Umgebungen**:
  - Warum sind sie wichtig?
  - Unterschied zwischen **System-Python** und **virtuellen Umgebungen**.
- Vergleich von `pip` vs. `conda`.

Fragen für die Gruppe:

- Warum sollten wir für verschiedene Projekte separate Umgebungen verwenden?
- Welche Probleme könnten auftreten, wenn man **keine** virtuelle Umgebung benutzt?

9:30 - 10:15: Erstellen virtueller Umgebungen

Themen:

- Erstellen einer virtuellen Umgebung mit `venv`:

```
python -m venv my_env
source my_env/bin/activate  # Mac/Linux
my_env\Scripts\activate  # Windows
```

- ODER mit `conda`:

```
conda create --name my_env python=3.10
conda activate my_env
```

- Überprüfung der aktiven Umgebung:

```
python --version
```

Übung:

- Studierende erstellen ihre eigene virtuelle Umgebung und testen, ob sie aktiviert wurde.

10:15 - 10:45: Verwalten von Abhängigkeiten

Themen:

- Installieren von Paketen mit `pip`:

```
pip install notebook
```

- Liste installierter Pakete anzeigen:

```
pip list
```

- `requirements.txt` zur Dokumentation der Pakete nutzen:

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

Übung:

- Studierende installieren `notebook`, `numpy`, `pandas`, `matplotlib` in ihrer Umgebung.

10:45 - 11:15: Einführung in Jupyter Notebooks

Themen:

- Was ist **Jupyter Notebook** und warum ist es nützlich?
- Starten von Jupyter:

```
jupyter notebook
```

- Erstellen und Umbenennen eines Notebooks.

Übung:

- Öffnen eines neuen Jupyter Notebooks und Testen einer ersten Code-Zelle:

```
print("Willkommen in Jupyter!")
```

11:15 - 12:00: Arbeiten mit Jupyter-Zellen

Themen:

- **Arten von Zellen**:
  - Code-Zellen für Python-Code.
  - Markdown-Zellen für Dokumentation.
- **Nützliche Jupyter-Shortcuts**:
  - `Shift + Enter`: Zelle ausführen.
  - `Esc + A` / `Esc + B`: Neue Zelle über / unter einer anderen einfügen.
  - `Esc + D, D`: Zelle löschen.

Übung:

- Erstellen einer Markdown-Zelle:

```
# Mein erstes Jupyter Notebook
Dies ist eine Markdown-Zelle für **formatierte Texte**.
```

- Hinzufügen einer Code-Zelle:

```
for i in range(5):
    print(f"Nummer: {i}")
```

- Einführung in **Jupyter Magics**:

```
%timeit sum(range(1000))
```

- Ausgabe aller verfügbaren Magics:

```
%lsmagic
```

12:00 - 12:30: Grundlegende Python-Übungen in Jupyter

Themen:

- Variablen und Datentypen testen:

```
message = "Hallo, Jupyter!"
print(message)
```

- **Übung:**
  - Studierende sollen eine Funktion schreiben, die die Quadratsumme einer Liste berechnet:

```
def summe_quadrate(zahlen):
    return sum([x**2 for x in zahlen])

print(summe_quadrate([1, 2, 3, 4]))
```

  - **Zusatzübung:**
    - Eine Markdown-Zelle erstellen, in der sie eine kurze Dokumentation zu ihrer Funktion schreiben.

12:30 - 13:00: Q&A und Troubleshooting

Themen:

- Überprüfung, ob alle Studierenden erfolgreich:
  - Eine virtuelle Umgebung erstellt haben.

    - `notebook` und andere Pakete installiert haben.
    - Ein Jupyter Notebook gestartet haben.
- **Typische Fehler beheben**:
    - "Python nicht gefunden" → Umgebung nicht aktiviert.
    - "Modul nicht gefunden" → Paket nicht in der aktiven Umgebung installiert.
- Zusammenfassung der **wichtigsten Konzepte**:
    - Wie erstelle ich eine virtuelle Umgebung?
    - Wie starte ich ein Jupyter Notebook?
    - Wie verwalte ich Abhängigkeiten?

Hausaufgabe für Tag 1

1. **Erstelle eine neue virtuelle Umgebung** und installiere `numpy`, `pandas`, `matplotlib`.
2. **Schreibe eine Funktion in Jupyter Notebook**, die eine Liste verarbeitet.
3. **Nutze Markdown**, um eine kurze Erklärung zu deinem Notebook hinzuzufügen.
4. **Teste, ob `%timeit` richtig funktioniert**, indem du eine Zeiterfassung für deine Funktion durchführst.