LU08c - Python in der Linux-Shell

Pfade in WSL

Linux und Windows unterscheiden sich erheblich in der Art und Weise, wie sie mit Ordnerpfaden umgehen. Hier sind die wesentlichen Unterschiede:

Pfad-Trennzeichen

- **Linux:** Verwendet den Schrägstrich `/` als Trennzeichen zwischen Ordnern.
    1. Beispiel: `/home/user/documents`
- **Windows:** Verwendet den Backslash `\` als Trennzeichen.
    1. Beispiel: `C:\Users\User\Documents`

Gross- und Kleinschreibung

- **Linux:** Unterscheidet zwischen Gros- und Kleinschreibung in Dateinamen und Ordnern. Das bedeutet, dass `File.txt` und `file.txt` zwei unterschiedliche Dateien sein können.
- **Windows:** Ignoriert standardmässig die Groß- und Kleinschreibung in Dateinamen. `File.txt` und `file.txt` werden als dieselbe Datei betrachtet.

Pfadstruktur

- **Linux:**
    - Es gibt ein einheitliches Wurzelverzeichnis `/`.
    - Alle Laufwerke und Geräte werden unterhalb von `/` eingebunden (z. B. `/mnt/drive1`).
    - Das Dateisystem folgt einer hierarchischen Struktur mit standardisierten Verzeichnissen wie `/etc`, `/bin`, `/usr`, und `/home`.
- **Windows:**
    - Jeder Laufwerksbuchstabe (z. B. `C:`, `D:`) hat sein eigenes Wurzelverzeichnis.
    - Die Struktur ist nicht einheitlich und hängt von der Konfiguration und Installation ab.
    - Benutzerverzeichnisse befinden sich oft unterhalb von `C:\Users\<Benutzername>`.

Absoluter vs. relativer Pfad

- **Linux:** Ein absoluter Pfad beginnt immer mit `/`, z. B. `/home/user/file`.
- **Windows:** Ein absoluter Pfad beginnt mit dem Laufwerksbuchstaben, z. B. `C:\Users\User\File`.

In WSL sind die Laufwerke C:, D:, ... unter `/mnt/` eingebunden. Um auf den Windowsordner `C:\BZZ\Python` zuzugreifen, verwenden Sie in WSL `/mnt/c/BZZ/Python`.

Virtual Environment

In den meisten Fällen überlassen wir es unserer IDE, das Virtual Environment zu erstellen und zu aktivieren. Falls wir unser Programm jedoch auf einem Server ausführen wollen, haben wir nur die Shell zur Verfügung. In diesem Fall müssen wir uns selber um das `venv` kümmern.

Voraussetzung

Für das erste Projekt in WSL muss zunächst `pip` installiert werden: `sudo apt install python3-pip`.

Projekt erstellen

Ich gehe davon aus, dass ich einen Ordner mit einem Python-Projekt erstellt habe. Dieser kann manuell mit `mkdir coolproject` erstellt werden. Möchte ich ein Projekt von GitHub verwenden, kann ich den Ordner mit `git clone` erstellen.

```
git clone https://github.com/BZZ-Commons/python-template coolproject
```

Dieser Befehl klont das Projekt von GitHub in den Ordner `coolproject`.

Virtual Environment erstellen

Wir verwenden das Modul `venv` um unser Virtual Environment zu erstellen.

`cd coolproject`	Wechsle in das Verzeichnis mit deinem Pythonprojekt
`python3 -m venv ./venv`	Führe das Modul `venv` aus und lege fest, wo der Ordner für das Virtual Environment angelegt wird.

```
Mit ''ls -l'' kannst du prüfen, ob der Ordner ''venv'' angelegt wurde.

<WRAP center round tip 60%>
Je nach Quelle wird der Zielordner ''venv'' oder ''.venv'' genannt.
Für die Funktionalität macht dies keinen Unterschied.
In meinen Beispielen verwende ich den Ordnernamen ''venv'' (ohne Punkt am Anfang).
</WRAP>

==== Virtual Environment aktivieren ====
Bevor wir das Pythonprojekt ausführen, sollten wir das Virtual Environment aktiveren.
Dadurch stellen wir sicher, dass alle Änderungen an der Umgebung nur dieses Projekt betreffen und nicht systemweite Änderungen gemacht werden.
```

```
<code bash>
source venv/bin/activate
```

Anhand des Eingabeprompts erkennen wir, dass das Virtual Environment aktiviert ist:

```
(venv) user@system:
```

Um das Virtual Environment zu deaktivieren, gib den Befehl `deactivate` ein.

Abhängigkeiten installieren

In der Regel hat jedes Pythonprojekt eine Datei `requirements.txt` mit einer Liste der benötigten Pakete. Dies erleichtert die Installation der Pakete, wenn wir das Projekt von GitHub klonen.

Installiere die Pakete immer in ein aktiviertes Virtual Environment. Andernfalls würden die Pakete systemweit installiert, was zu Konflikten zwischen verschiedenen Projekten führen kann.

```
pip3 install -r requirements.txt
```

Skript ausführen

Nun können wir unsere Pythonskripte ausführen:

```
python3 main.py
```

M122-LU089