

# Python Projekte Tips

## Python-Projekt in venv Umgebung ausführen

### Projektordner erstellen

Wähle einen Ordner für dein Projekt, z. B.:

```
mkdir telegram-demo  
cd telegram-demo
```

### Virtuelle Umgebung erstellen

Damit wird ein Ordner `.venv/` angelegt, der die isolierte Python-Umgebung enthält.

```
python -m venv .venv
```

### Virtuelle Umgebung aktivieren

```
.\.venv\Scripts\Activate.ps1
```

Falls Scripts nicht ausgeführt werden dürfen → Rechte aktivieren:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

dann erneut die virtuelle Umgebung aktivieren

```
.\.venv\Scripts\Activate.ps1
```

Wenn alles geklappt hat wird in der Shell folgendes angezeigt:

```
(.venv) C:\Pfad\telegram-demo>
```

### benötigte Bibliotheken installieren

```
pip install python-telegram-bot
```

Anzeigen der installierten Module / Bibliotheken mit

```
pip list
```

## (Optional) Anforderungen speichern

Für die automatische Installation der benötigten Module kann eine requirements-Datei erstellt werden

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

So kann dann später die Projekt-Umgebung einfach wieder hergestellt / installiert werden. Z.B. nach dem Clonen eines GitHub-Repos

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

## venv deaktivieren (optional)

Nach der Installation und dem auflisten der installierten Module / Bibliotheken kann die venv-Umgebung wieder deaktiviert werden.

```
deactivate
```

## virtuelle Umgebung (venv) entfernen

Falls die venv-Umgebung noch aktiv sein sollte, diese zuerst deaktivieren:

```
deactivate
```

dann einfach im Explorer das .venv - Verzeichnis löschen oder per PowerShell

```
Remove-Item -Recurse -Force .\venv
```

oder

```
rmdir /s /q .venv
```

## Hinweise

- Wenn du mehrere venvs verwendet hast, einfach jeweils die Ordner löschen.
- Wenn du Probleme mit dem Aktivieren hattest (ExecutionPolicy unter Windows), gilt weiterhin:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```