# Python Cheat Sheet

## venv
(Virtuelle Umgebungen)

**Was ist das?**

Eine virtuelle Umgebung isoliert Python-Pakete für ein Projekt, damit Abhängigkeiten nicht mit anderen Projekten kollidieren.

**Aufsetzen:**

```
python -m venv .venv

# Aktivieren
# Linux/Mac:
source .venv/bin/activate
# Windows (PowerShell):
.venv\Scripts\Activate.ps1

# Windows(CMD)

.venv/Scripts/activate

# Deaktivieren
deactivate
```

## dotenv

**Was ist das?**

.env Dateien speichern geheime Daten (API-Keys, Passwörter) außerhalb des Codes.

**Installation:**

```
pip install python-dotenv
```

**.env Datei:**

```
SECRET_KEY=MYAPIKEY
DEBUG=True
```

**Python Code:**

```
from dotenv import load_dotenv
import os

load_dotenv()  # .env laden
print(os.getenv("SECRET_KEY"))
```

## pip

**Was ist das?**

pip ist der Python-Paketmanager.

**Beispiele:**

```
# Paket installieren
pip install python-dotenv

# Paket deinstallieren
pip uninstall python-dotenv
```

## requirements.txt

**Was ist das?**

Eine Datei, die alle Projekt-Abhängigkeiten auflistet.

**Erstellen:**

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

**Installieren:**

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

## Projektstruktur & Module

Beispielstruktur

```
my_project/
|── .venv/
|── my_module/
|   |── __init__.py   # macht es zu einem Paket
|   |── utils.py
|── main.py
|── requirements.txt
|── .gitignore
|── README.md
```

**__init__.py**

Kennzeichnet ein Verzeichnis als Python-Paket.
Kann leer sein oder Importe bündeln:

```
# my_module/__init__.py
from .utils import my_function
```

**Import im Code:**

```
from my_module import my_function
```