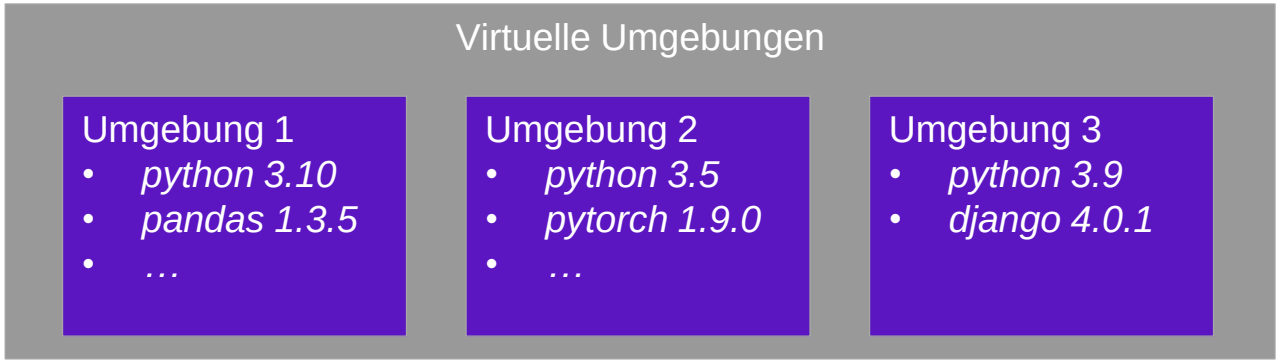
DataCraft

9
Virtuelle Umgebungen

Packages und virtuelle Umgebungen

DataCraft

- Es gibt tausende Erweiterungen für Python (Packages / Pakete oder Libraries / Bibliotheken genannt)
- Packages haben Voraussetzungen an die Python-Version und an die Version von anderen Packages, die sie nutzen
- Idee:



Pip vs. Python 1

DataCraft

- Pip Packages sind Python Bibliotheken / Module
- Conda Packages können zusätzlich binäre Module und ausführbare Dateien enthalten
- Mittleweile kann in Pip in sogenannten Wheels auch kompilierter Code enthalten sein.
- Conda ist gleichzeitig ein Umgebungsmanagement-Tool und kann Abhängigkeiten von nicht-Python-Code auflösen

Pip vs. Python 2

DataCraft

- In der Datenanalyse wird meistens conda verwendet, ansonsten meistens pip + ein Umgebungstool (z.B. venv, poetry, pipenv).
- Beide können parallel auf einem Rechner verwendet sein, nur ein Mix innerhalb einer Umgebungen funktioniert nicht immer
- conda bietet leider längst nicht alle Packages, die es für pip gibt
- Mamba ist eine schnellere Version von conda.

  `conda install mamba`

  oder https://github.com/conda-forge/miniforge

Conda Befehle

DataCraft

Conda Befehl	Beschreibung
conda create --name NAME	Eine Umgebung erzeugen
conda env remove --name NAME	Eine Umgebung löschen
conda env list	Alle Umgebungen auflisten
conda activate NAME	Eine Umgebung aktivieren
conda deactivate NAME	Eine Umgebung deaktivierung
conda install PACKAGENAME	Ein Package installieren (mit Möglichkeiten der Versionsangabe)
conda uninstall PACKAGENAME	Ein Package deinstallieren
conda list	Alle Packages auflisten
conda update --all	Alle Packages einer Umgebung aktualisieren
conda clean --all	Heruntergeladene Dateien löschen

- statt --option kann auch -o verwendet werden (z.B. --name entspricht –n)
- Mit -y wird die Sicherheits-Nachfrage übersprungen

Pip vs. Python

DataCraft

- Python-Projekte haben in der Regel eine **requirements.txt**. Das ist eine Datei, in der die benötigten Packages und Versionen definiert sind. Diese Datei kann direkt von pip verarbeitet werden

- Das conda-Pendant ist die Datei **environment.yml** (yaml = yet another markup language)

environment.yml

```
name: beispiel
channels:
  - default
  - conda-forge
dependencies:
  - python=3.9
  - pandas=1.4
```

requirements.txt

```
python == 3.9
pandas >= 1.4
seaborn ~= 0.11.*
```

environment.yml und requirements.txt

DataCraft

conda

Installation der benötigten Packages

```
conda env create –f environment.yml
```

Eine environment-Datei erzeugen

```
conda env export > environment.yml
```

Eine bestehende Umgebung aktualisieren

```
conda env update –f environment.yml
```

pip

Installation der benötigten Packages

```
pip install –r requirements.txt
```

Eine requirements-Datei erzeugen

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

Update der Packages und der requirements.txt am besten mit Package upgrade-requirements

- In der environment.yml kann auch angegeben werden, dass einige Package mit pip installiert werden sollen.

```
dependencies:
  - pip:
    - PACKAGENAME
```

- Für ein Projekt sollten diese Dateien nur die minimale Menge an Abhängigkeiten enthalten. Daher ist eine Nachbearbeitung per Texteditor notwendig.

- Erstellen einer requirements.txt aus einer conda-Umgebung

```
pip list --format=freeze > requirements.txt
```

- Windows besitzt seit Version 10 ein Linux-Umgebung (WSL)
- Server in einer Produktivumgebung laufen meistens auf Linux
- Mit WSL können Entwickler direkt in der späteren Umgebung entwickeln
- Installation in Powershell

```
wsl --install
```

DataCraft

- VS Code unterstützt Remote-Rechner und WSL mittels der Erweiterung [Remote Development Extension Pack]

- Installation von Python unter WSL

```
sudo apt update
sudo apt install python3 python3-pip
```

- Installation von miniconda unter WSL

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh
source ~/.bashrc
rm Miniconda3-latest-Linux-x86_64.sh

# optional: WSL soll nicht automatisch mit conda starten
conda config --set auto_activate_base false
```

Docker-Umgebungen

DataCraft

- Docker ist ein Tool, um die Bereitstellung von Applikationen plattformübergreifend zu automatisieren
- Dazu werden sogenannte Container erzeugt
- Docker-Container können überall ausgeführt werden, lokal, bei einem externen Dienstanbieter oder in der Cloud
- Im Prinzip wie virtuelle Maschinen (so wie WSL), aber effizienter, da kein eigenes Betriebssystem