合肥大學
HEFEI UNIVERSITY

Datenbanken

15. psycopg Installieren

Thomas Weise (汤卫思)
tweise@hfuu.edu.cn

Institute of Applied Optimization (IAO)
School of Artificial Intelligence and Big Data
Hefei University
Hefei, Anhui, China

应用优化研究所
人工智能与大数据学院
合肥大学
中国安徽省合肥市

Version: 2025-08-29

Databases

Dies ist ein Kurs über Datenbanken an der Universität Hefei (合肥大学).

Die Webseite mit dem Lehrmaterial dieses Kurses ist https://thomasweise.github.io/databases (siehe auch den QR-Kode unten rechts). Dort können Sie das Kursbuch (in Englisch) und diese Slides finden. Das Repository mit den Beispielen finden Sie unter https://github.com/thomasWeise/databasesCode.

Outline

Einleitung

Einleitung

python™

- Wir wollen nun von Python aus auf unsere Datenbank zugreifen.

Einleitung

python™

The Python programming language logo is under the copyright of its owners.

Copyright © Gabriella Albano and the Psycopg team

- Wir wollen nun von Python aus auf unsere Datenbank zugreifen.
- Dafür verwenden wir das Paket psycopg[45].

Einleitung

python™

- Wir wollen nun von Python aus auf unsere Datenbank zugreifen.
- Dafür verwenden wir das Paket psycopg[45].
- Das müssen wir zuerst installieren.

Einleitung

python™

- Wir wollen nun von Python aus auf unsere Datenbank zugreifen.
- Dafür verwenden wir das Paket psycopg[45].
- Das müssen wir zuerst installieren. Das geht auf zwei Arten.

Einleitung

python™

- Wir wollen nun von Python aus auf unsere Datenbank zugreifen.
- Dafür verwenden wir das Paket psycopg[45].
- Das müssen wir zuerst installieren. Das geht auf zwei Arten:
  1. Für den Produktivbetrieb von Programmen würden wir Pakete immer über das Terminal in einem virtuellen Environment installieren.

Einleitung

- Wir wollen nun von Python aus auf unsere Datenbank zugreifen.
- Dafür verwenden wir das Paket psycopg[45].
- Das müssen wir zuerst installieren. Das geht auf zwei Arten:
  1. Für den Produktivbetrieb von Programmen würden wir Pakete immer über das Terminal in einem virtuellen Environment installieren.
  2. Arbeiten wir mit PyCharm, dann könne wir das virtuelle Environment genauso gut in PyCharm direkt verwenden, wobei dieser Ansatz nur für Softwareentwicklung und nicht für den Produktivbetrieb geeignet ist.

Virtuelle Environments (1)

- Aber was sind virtuelle Environments (virtual environments) eigentlich?

Virtuelle Environments (1)

- Aber was sind virtuelle Environments (virtual environments) eigentlich?
- Für Python gibt es sehr sehr viele verschiedene Softwarepakete, die zusätzliche Funktionalität anbieten.

Virtuelle Environments (1)

- Aber was sind virtuelle Environments (virtual environments) eigentlich?
- Für Python gibt es sehr sehr viele verschiedene Softwarepakete, die zusätzliche Funktionalität anbieten.
- Fast alle gibt es in verschiendenen Versionen, weil sich Software ja weiterentwickelt.

Virtuelle Environments (1)

- Aber was sind virtuelle Environments (virtual environments) eigentlich?
- Für Python gibt es sehr sehr viele verschiedene Softwarepakete, die zusätzliche Funktionalität anbieten.
- Fast alle gibt es in verschiendenen Versionen, weil sich Software ja weiterentwickelt.
- Manchmal bietet eine neue Version neue Funktionalität an, manchmal wird auch eine alte Funktionalität entfernt, machmal ändern sich Application Programming Interfaces (APIs).

Virtuelle Environments (1)

- Aber was sind virtuelle Environments (virtual environments) eigentlich?
- Für Python gibt es sehr sehr viele verschiedene Softwarepakete, die zusätzliche Funktionalität anbieten.
- Fast alle gibt es in verschiendenen Versionen, weil sich Software ja weiterentwickelt.
- Manchmal bietet eine neue Version neue Funktionalität an, manchmal wird auch eine alte Funktionalität entfernt, machmal ändern sich Application Programming Interfaces (APIs).
- Nicht alle Versionen eines Paketes sind also kompatibel.

Virtuelle Environments (1)

- Aber was sind virtuelle Environments (virtual environments) eigentlich?
- Für Python gibt es sehr sehr viele verschiedene Softwarepakete, die zusätzliche Funktionalität anbieten.
- Fast alle gibt es in verschiendenen Versionen, weil sich Software ja weiterentwickelt.
- Manchmal bietet eine neue Version neue Funktionalität an, manchmal wird auch eine alte Funktionalität entfernt, machmal ändern sich Application Programming Interfaces (APIs).
- Nicht alle Versionen eines Paketes sind also kompatibel.
- Jedes Paket kann selber wieder andere Pakete brauchen.

Virtuelle Environments (1)

- Aber was sind virtuelle Environments (virtual environments) eigentlich?
- Für Python gibt es sehr sehr viele verschiedene Softwarepakete, die zusätzliche Funktionalität anbieten.
- Fast alle gibt es in verschiendenen Versionen, weil sich Software ja weiterentwickelt.
- Manchmal bietet eine neue Version neue Funktionalität an, manchmal wird auch eine alte Funktionalität entfernt, machmal ändern sich Application Programming Interfaces (APIs).
- Nicht alle Versionen eines Paketes sind also kompatibel.
- Jedes Paket kann selber wieder bestimmte Versionen andere Pakete brauchen.
- Hat man verschiedene Python Programme, so kann es sein, dass ein Paket NumPy mindest in Version 2.3.0 braucht und ein anderes NumPy höchstens in Version 1.26.4.

Virtuelle Environments (1)

- Aber was sind virtuelle Environments (virtual environments) eigentlich?
- Für Python gibt es sehr sehr viele verschiedene Softwarepakete, die zusätzliche Funktionalität anbieten.
- Fast alle gibt es in verschiendenen Versionen, weil sich Software ja weiterentwickelt.
- Manchmal bietet eine neue Version neue Funktionalität an, manchmal wird auch eine alte Funktionalität entfernt, machmal ändern sich Application Programming Interfaces (APIs).
- Nicht alle Versionen eines Paketes sind also kompatibel.
- Jedes Paket kann selber wieder bestimmte Versionen andere Pakete brauchen.
- Hat man verschiedene Python Programme, so kann es sein, dass ein Paket NumPy mindest in Version 2.3.0 braucht und ein anderes NumPy höchstens in Version 1.26.4.
- Dann kann man beide Programme nicht "zusammen" installieren.

Virtuelle Environments (2)

- Die Lösung für dieses Problem sind virtuelle Environments.

Virtuelle Environments (2)

- Die Lösung für dieses Problem sind virtuelle Environments.
- Ein virtuelles Environment ist im Grunde ein Ordner im System, in dem sich eine eigenständige Python-Installation befindet.

Virtuelle Environments (2)

- Die Lösung für dieses Problem sind virtuelle Environments.
- Ein virtuelles Environment ist im Grunde ein Ordner im System, in dem sich eine eigenständige Python-Installation befindet.
- In diesem Ordner sind dann auch alle installierten Pakete.

Virtuelle Environments (2)

- Die Lösung für dieses Problem sind virtuelle Environments.
- Ein virtuelles Environment ist im Grunde ein Ordner im System, in dem sich eine eigenständige Python-Installation befindet.
- In diesem Ordner sind dann auch alle installierten Pakete.
- Nun kann man jedes Python-Programm, dass man braucht, in ein separates virtuelles Environment installieren.

Virtuelle Environments (2)

- Die Lösung für dieses Problem sind virtuelle Environments.
- Ein virtuelles Environment ist im Grunde ein Ordner im System, in dem sich eine eigenständige Python-Installation befindet.
- In diesem Ordner sind dann auch alle installierten Pakete.
- Nun kann man jedes Python-Programm, dass man braucht, in ein separates virtuelles Environment installieren.
- Somit können keine (vermeidbaren) Versionskonflikte mehr auftreten.

Virtuelle Environments (2)

- Die Lösung für dieses Problem sind virtuelle Environments.
- Ein virtuelles Environment ist im Grunde ein Ordner im System, in dem sich eine eigenständige Python-Installation befindet.
- In diesem Ordner sind dann auch alle installierten Pakete.
- Nun kann man jedes Python-Programm, dass man braucht, in ein separates virtuelles Environment installieren.
- Somit können keine (vermeidbaren) Versionskonflikte mehr auftreten.
- Gleichgültig ob wir psycopg für den Produktiveinsatz brauchen oder ob wir nur bequem in PyCharm Software entwickeln wollen, wir werden also auf jeden Fall ein virtuelles Environment verwenden.

Installation im Terminal

Programme im Terminal

- Datenbanken sind Speicher für große Mengen strukturierter Daten.

Programme im Terminal

- Datenbanken sind Speicher für große Mengen strukturierter Daten.
- Sie bilden die Backends von modernen Applikationen.

Programme im Terminal

- Datenbanken sind Speicher für große Mengen strukturierter Daten.
- Sie bilden die Backends von modernen Applikationen.
- Sie beinhalten die Daten, die Geschäfte antreiben sowie den Inhalt riesiger Webseiten.

Programme im Terminal

- Datenbanken sind Speicher für große Mengen strukturierter Daten.
- Sie bilden die Backends von modernen Applikationen.
- Sie beinhalten die Daten, die Geschäfte antreiben sowie den Inhalt riesiger Webseiten.
- Wir können uns auf Datenbanken über Applikationen und Programme verbinden, um mit den Daten zu arbeiten.

Programme im Terminal

- Datenbanken sind Speicher für große Mengen strukturierter Daten.
- Sie bilden die Backends von modernen Applikationen.
- Sie beinhalten die Daten, die Geschäfte antreiben sowie den Inhalt riesiger Webseiten.
- Wir können uns auf Datenbanken über Applikationen und Programme verbinden, um mit den Daten zu arbeiten.
- Diese sind oftmals Middlewares, wie Web Services oder Application Servers, die Geschäftslogik implementieren und dafür sorgen, dass wir mit den Daten korrekt arbeiten.

Programme im Terminal

- Datenbanken sind Speicher für große Mengen strukturierter Daten.
- Sie bilden die Backends von modernen Applikationen.
- Sie beinhalten die Daten, die Geschäfte antreiben sowie den Inhalt riesiger Webseiten.
- Wir können uns auf Datenbanken über Applikationen und Programme verbinden, um mit den Daten zu arbeiten.
- Diese sind oftmals Middlewares, wie Web Services oder Application Servers, die Geschäftslogik implementieren und dafür sorgen, dass wir mit den Daten korrekt arbeiten.
- Solche Applikationen sind Programme, die, wenn wir sie mit Python entwickelt haben, oftmals im Terminal laufen.

Programme im Terminal

- Datenbanken sind Speicher für große Mengen strukturierter Daten.
- Sie bilden die Backends von modernen Applikationen.
- Sie beinhalten die Daten, die Geschäfte antreiben sowie den Inhalt riesiger Webseiten.
- Wir können uns auf Datenbanken über Applikationen und Programme verbinden, um mit den Daten zu arbeiten.
- Diese sind oftmals Middlewares, wie Web Services oder Application Servers, die Geschäftslogik implementieren und dafür sorgen, dass wir mit den Daten korrekt arbeiten.
- Solche Applikationen sind Programme, die, wenn wir sie mit Python entwickelt haben, oftmals im Terminal laufen.
- Niemals würde man sie in einer IDE wie PyCharm starten.

Programme im Terminal

- Datenbanken sind Speicher für große Mengen strukturierter Daten.
- Sie bilden die Backends von modernen Applikationen.
- Sie beinhalten die Daten, die Geschäfte antreiben sowie den Inhalt riesiger Webseiten.
- Wir können uns auf Datenbanken über Applikationen und Programme verbinden, um mit den Daten zu arbeiten.
- Diese sind oftmals Middlewares, wie Web Services oder Application Servers, die Geschäftslogik implementieren und dafür sorgen, dass wir mit den Daten korrekt arbeiten.
- Solche Applikationen sind Programme, die, wenn wir sie mit Python entwickelt haben, oftmals im Terminal laufen.
- Niemals würde man sie in einer IDE wie PyCharm starten.

> *The only proper way to run a Python application in a productive scenario is in the terminal.*
>
> — *Thomas Weise (汤卫思) [48], 2024*

Psycopg im Terminal und Virtuellen Environment

- Beginnen wir also mit einer Installation im Terminal.

Psycopg im Terminal und Virtuellen Environment

- Beginnen wir also mit einer Installation im Terminal.
- Wir machen die Installation hier nur unter Ubuntu Linux.

Psycopg im Terminal und Virtuellen Environment

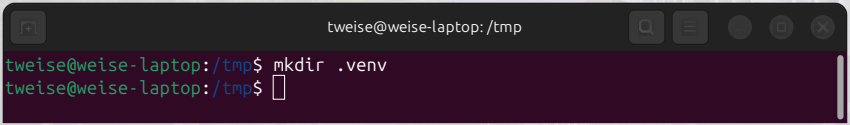
- Beginnen wir also mit einer Installation im Terminal.
- Wir machen die Installation hier nur unter Ubuntu Linux.
- Unter Microsoft Windows funktioniert es fast genauso.

Psycopg im Terminal und Virtuellen Environment

- Beginnen wir also mit einer Installation im Terminal.
- Wir machen die Installation hier nur unter Ubuntu Linux.
- Unter Microsoft Windows funktioniert es fast genauso Sie können Beispiele dafür im Kurs *Programming with Python* [48] finden.

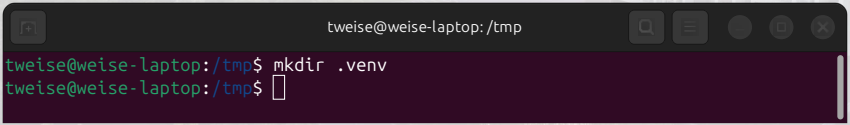
Psycopg im Terminal und Virtuellen Environment

- Wir öffnen also ein Terminal via Ctrl + Alt + T.



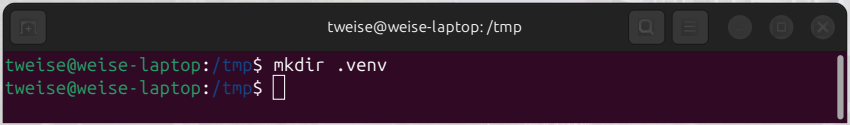
Psycopg im Terminal und Virtuellen Environment

- Wir öffnen also ein Terminal via Ctrl + Alt + T.
- Wir gehen in das Verzeichnis, in dem wir später unser Program ausführen möchten.



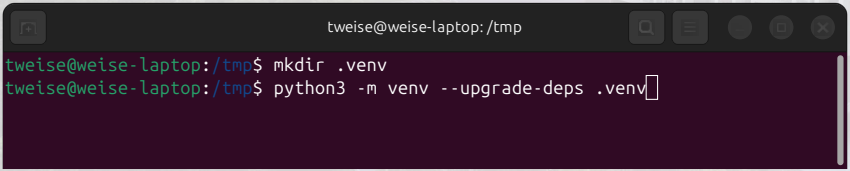
Psycopg im Terminal und Virtuellen Environment

- Wir öffnen also ein Terminal via Ctrl + Alt + T.
- Wir gehen in das Verzeichnis, in dem wir später unser Program ausführen möchten.
- Wir erstellen einen neuen Ornder namens `.venv` um dort das neue virtuelle Environment rein zu installieren.



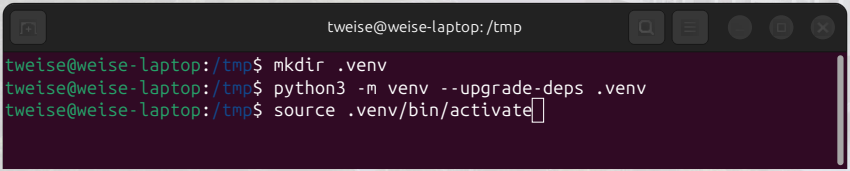
Psycopg im Terminal und Virtuellen Environment

- Wir initialisieren ein leeres virtuelles Environment in diesem Ordnung mit dem Kommando `python3 -m venv --upgrade-deps .venv` und Enter.



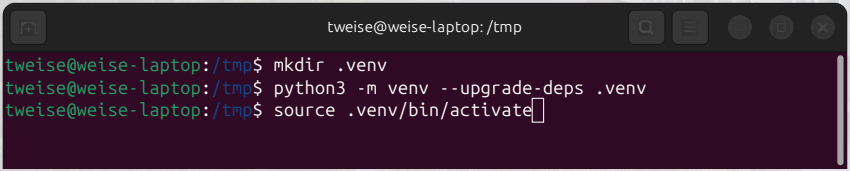
Psycopg im Terminal und Virtuellen Environment

- Das Virtuelle Environment wurde erstellt.



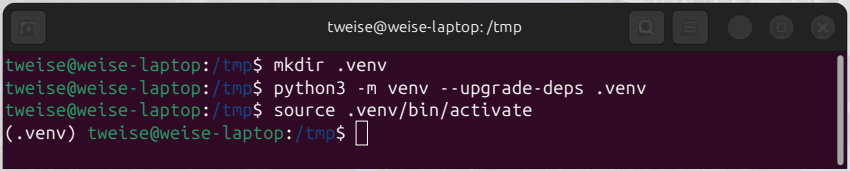
Psycopg im Terminal und Virtuellen Environment

- Das Virtuelle Environment wurde erstellt.
- Wir aktivieren das Environment mit dem Kommando `source .venv/bin/activate` und Enter.



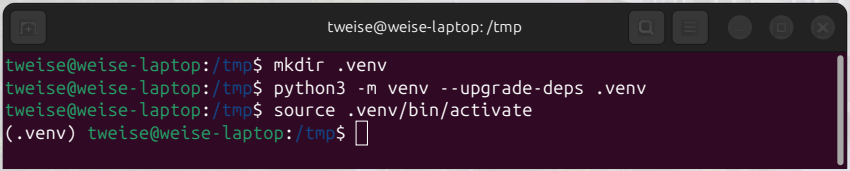
Psycopg im Terminal und Virtuellen Environment

- Solange ein virtuelles Environment aktiv ist, werden alle neue Pakete in dieses Environment installiert und Python-Programme laden ihre benötigten Pakete aus diesem Environment.



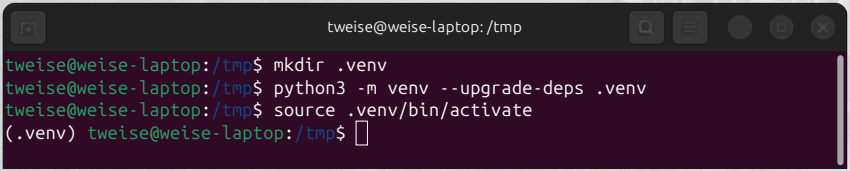
Psycopg im Terminal und Virtuellen Environment

- Solange ein virtuelles Environment aktiv ist, werden alle neue Pakete in dieses Environment installiert und Python-Programme laden ihre benötigten Pakete aus diesem Environment.
- Sie sehen auch, dass sich der Promt des Terminals geändert hat, was as aktive Environment anzeigt.



Psycopg im Terminal und Virtuellen Environment

- Solange ein virtuelles Environment aktiv ist, werden alle neue Pakete in dieses Environment installiert und Python-Programme laden ihre benötigten Pakete aus diesem Environment.
- Sie sehen auch, dass sich der Promt des Terminals geändert hat, was as aktive Environment anzeigt.
- Beachten Sie: Das Environment ist nur im aktuellen Terminal aktiv.



Psycopg im Terminal und Virtuellen Environment

- Unter Python installieren wir Pakete normalerwise mit Hilfe des Paketmanagers pip.

```
tweise@weise-laptop: /tmp
tweise@weise-laptop:/tmp$ source .venv/bin/activate
(.venv) tweise@weise-laptop:/tmp$ pip install psycopg
Collecting psycopg
  Using cached psycopg-3.2.4-py3-none-any.whl.metadata (4.3 kB)
Collecting typing-extensions>=4.6 (from psycopg)
  Using cached typing_extensions-4.12.2-py3-none-any.whl.metadata (3.0 kB)
Using cached psycopg-3.2.4-py3-none-any.whl (198 kB)
Using cached typing_extensions-4.12.2-py3-none-any.whl (37 kB)
Installing collected packages: typing-extensions, psycopg
Successfully installed psycopg-3.2.4 typing-extensions-4.12.2
(.venv) tweise@weise-laptop:/tmp$
```

Psycopg im Terminal und Virtuellen Environment

- Unter Python installieren wir Pakete normalerwise mit Hilfe des Paketmanagers pip.
- pip nutzt automatisch das aktuelle virtuelle Environment, wenn eines aktiv ist.

```
tweise@weise-laptop: /tmp
tweise@weise-laptop:/tmp$ source .venv/bin/activate
(.venv) tweise@weise-laptop:/tmp$ pip install psycopg
Collecting psycopg
  Using cached psycopg-3.2.4-py3-none-any.whl.metadata (4.3 kB)
Collecting typing-extensions>=4.6 (from psycopg)
  Using cached typing_extensions-4.12.2-py3-none-any.whl.metadata (3.0 kB)
Using cached psycopg-3.2.4-py3-none-any.whl (198 kB)
Using cached typing_extensions-4.12.2-py3-none-any.whl (37 kB)
Installing collected packages: typing-extensions, psycopg
Successfully installed psycopg-3.2.4 typing-extensions-4.12.2
(.venv) tweise@weise-laptop:/tmp$
```

Psycopg im Terminal und Virtuellen Environment

- Unter Python installieren wir Pakete normalerwise mit Hilfe des Paketmanagers pip.
- pip nutzt automatisch das aktuelle virtuelle Environment, wenn eines aktiv ist.
- Wir installieren psycopg in unser neues Environment in dem wir `pip install psycopg` schreiben und Enter drücken.

```
tweise@weise-laptop: /tmp
tweise@weise-laptop:/tmp$ source .venv/bin/activate
(.venv) tweise@weise-laptop:/tmp$ pip install psycopg
Collecting psycopg
  Using cached psycopg-3.2.4-py3-none-any.whl.metadata (4.3 kB)
Collecting typing-extensions>=4.6 (from psycopg)
  Using cached typing_extensions-4.12.2-py3-none-any.whl.metadata (3.0 kB)
Using cached psycopg-3.2.4-py3-none-any.whl (198 kB)
Using cached typing_extensions-4.12.2-py3-none-any.whl (37 kB)
Installing collected packages: typing-extensions, psycopg
Successfully installed psycopg-3.2.4 typing-extensions-4.12.2
(.venv) tweise@weise-laptop:/tmp$
```

Psycopg im Terminal und Virtuellen Environment

- Unter Python installieren wir Pakete normalerwise mit Hilfe des Paketmanagers pip.
- pip nutzt automatisch das aktuelle virtuelle Environment, wenn eines aktiv ist.
- Wir installieren psycopg in unser neues Environment in dem wir `pip install psycopg` schreiben und Enter drücken.
- Der pip-Installer schlägt nun in PyPI nach und lädt das Paket herunter.

```
tweise@weise-laptop: /tmp
tweise@weise-laptop:/tmp$ source .venv/bin/activate
(.venv) tweise@weise-laptop:/tmp$ pip install psycopg
Collecting psycopg
  Using cached psycopg-3.2.4-py3-none-any.whl.metadata (4.3 kB)
Collecting typing-extensions>=4.6 (from psycopg)
  Using cached typing_extensions-4.12.2-py3-none-any.whl.metadata (3.0 kB)
Using cached psycopg-3.2.4-py3-none-any.whl (198 kB)
Using cached typing_extensions-4.12.2-py3-none-any.whl (37 kB)
Installing collected packages: typing-extensions, psycopg
Successfully installed psycopg-3.2.4 typing-extensions-4.12.2
(.venv) tweise@weise-laptop:/tmp$
```

Psycopg im Terminal und Virtuellen Environment

- pip nutzt automatisch das aktuelle virtuelle Environment, wenn eines aktiv ist.
- Wir installieren psycopg in unser neues Environment in dem wir `pip install psycopg` schreiben und Enter drücken.
- Der pip-Installer schlägt nun in PyPI nach und lädt das Paket herunter.
- Hier wurde psycopg in Version `3.2.4` heruntergeladen und installiert.

```
tweise@weise-laptop: /tmp
tweise@weise-laptop:/tmp$ source .venv/bin/activate
(.venv) tweise@weise-laptop:/tmp$ pip install psycopg
Collecting psycopg
  Using cached psycopg-3.2.4-py3-none-any.whl.metadata (4.3 kB)
Collecting typing-extensions>=4.6 (from psycopg)
  Using cached typing_extensions-4.12.2-py3-none-any.whl.metadata (3.0 kB)
Using cached psycopg-3.2.4-py3-none-any.whl (198 kB)
Using cached typing_extensions-4.12.2-py3-none-any.whl (37 kB)
Installing collected packages: typing-extensions, psycopg
Successfully installed psycopg-3.2.4 typing-extensions-4.12.2
(.venv) tweise@weise-laptop:/tmp$
```

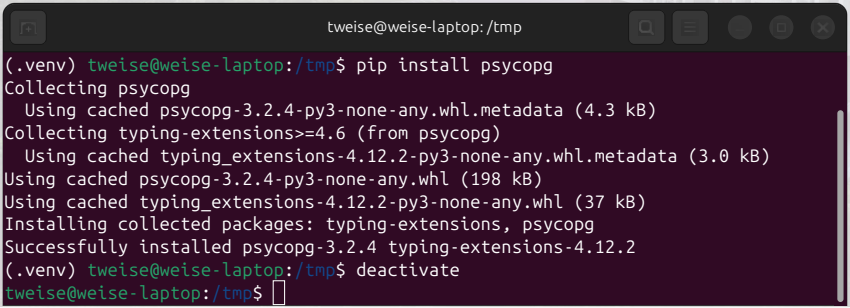
Psycopg im Terminal und Virtuellen Environment

- Wir installieren psycopg in unser neues Environment in dem wir `pip install psycopg` schreiben und Enter drücken.
- Der pip-Installer schlägt nun in PyPI nach und lädt das Paket herunter.
- Hier wurde psycopg in Version `3.2.4` heruntergeladen und installiert.
- Bei Ihnen wird bestimmt eine neue Version installiert.

```
tweise@weise-laptop: /tmp
tweise@weise-laptop:/tmp$ source .venv/bin/activate
(.venv) tweise@weise-laptop:/tmp$ pip install psycopg
Collecting psycopg
  Using cached psycopg-3.2.4-py3-none-any.whl.metadata (4.3 kB)
Collecting typing-extensions>=4.6 (from psycopg)
  Using cached typing_extensions-4.12.2-py3-none-any.whl.metadata (3.0 kB)
Using cached psycopg-3.2.4-py3-none-any.whl (198 kB)
Using cached typing_extensions-4.12.2-py3-none-any.whl (37 kB)
Installing collected packages: typing-extensions, psycopg
Successfully installed psycopg-3.2.4 typing-extensions-4.12.2
(.venv) tweise@weise-laptop:/tmp$
```

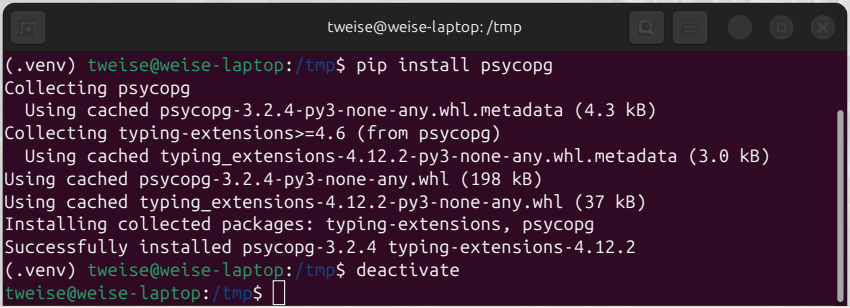
Psycopg im Terminal und Virtuellen Environment

- Wir sind fertig und können das virtuelle Environment deaktivieren.



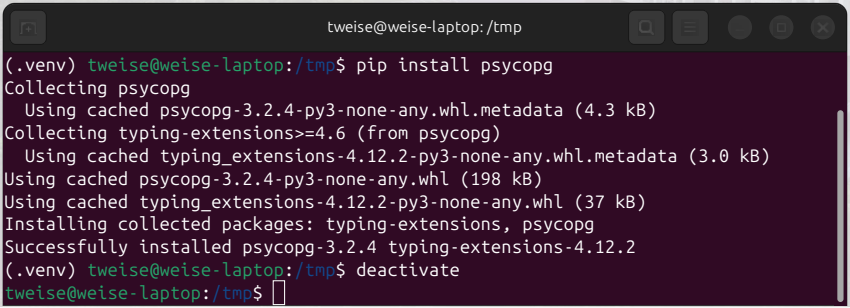
Psycopg im Terminal und Virtuellen Environment

- Wir sind fertig und können das virtuelle Environment deaktivieren.
- Der Prompt ändert sich wieder zurück.



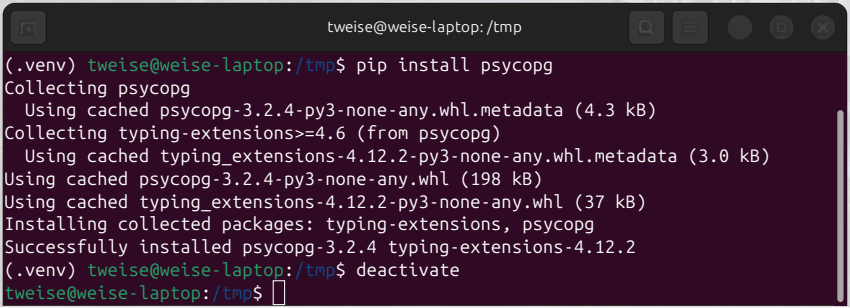
Psycopg im Terminal und Virtuellen Environment

- Wir sind fertig und können das virtuelle Environment deaktivieren.
- Der Prompt ändert sich wieder zurück.
- Von jetzt an wird wieder die Systeminstallation von Python verwendet.



Psycopg im Terminal und Virtuellen Environment

- Wir sind fertig und können das virtuelle Environment deaktivieren.
- Der Prompt ändert sich wieder zurück.
- Von jetzt an wird wieder die Systeminstallation von Python verwendet.
- Wenn wir das Paket brauchen, dann müssen wir das Environment wieder aktivieren.



PyCharm: Beispiele Herunterladen und Pyscopg Installieren

Git und GitHub

- Die PyCharm IDE erlaubt uns, im Grunde in einem Schritt die Beispiel-Kodes für diesen Kurs von GitHub herunterzuladen und die notwendigen Python-Pakete installieren.

Git und GitHub

- Die PyCharm IDE erlaubt uns, im Grunde in einem Schritt die Beispiel-Kodes für diesen Kurs von GitHub herunterzuladen und die notwendigen Python-Pakete installieren.
- Dabei treffen wir auf zwei Konzepte: Git und die bereits diskutierten virtuellen Environments.

Git und GitHub

- Die PyCharm IDE erlaubt uns, im Grunde in einem Schritt die Beispiel-Kodes für diesen Kurs von GitHub herunterzuladen und die notwendigen Python-Pakete installieren.
- Dabei treffen wir auf zwei Konzepte: Git und die bereits diskutierten virtuellen Environments.
- Git ist ein verteiltes Versionsmanagementsystem – es erlaubt uns, sogenannte Repositorie zu erstellen, die im Grunde Verzeichnisse sind deren Änderungs- und Versionsgeschichte mit gespeichert wird und über die man kollaborativ arbeiten kann.

Git und GitHub

- Die PyCharm IDE erlaubt uns, im Grunde in einem Schritt die Beispiel-Kodes für diesen Kurs von GitHub herunterzuladen und die notwendigen Python-Pakete installieren.
- Dabei treffen wir auf zwei Konzepte: Git und die bereits diskutierten virtuellen Environments.
- Git ist ein verteiltes Versionsmanagementsystem – es erlaubt uns, sogenannte Repositorie zu erstellen, die im Grunde Verzeichnisse sind deren Änderungs- und Versionsgeschichte mit gespeichert wird und über die man kollaborativ arbeiten kann.
- Jedes Teammitglied arbeitet dazu auf einer lokalen Kopie des Kodes und kann ihre lokalen Änderungen in das Repository "committen" sowie die Änderungen anderer Teammitglieder herunterladen.

Git und GitHub

- Die PyCharm IDE erlaubt uns, im Grunde in einem Schritt die Beispiel-Kodes für diesen Kurs von GitHub herunterzuladen und die notwendigen Python-Pakete installieren.
- Dabei treffen wir auf zwei Konzepte: Git und die bereits diskutierten virtuellen Environments.
- Git ist ein verteiltes Versionsmanagementsystem – es erlaubt uns, sogenannte Repositorie zu erstellen, die im Grunde Verzeichnisse sind deren Änderungs- und Versionsgeschichte mit gespeichert wird und über die man kollaborativ arbeiten kann.
- Jedes Teammitglied arbeitet dazu auf einer lokalen Kopie des Kodes und kann ihre lokalen Änderungen in das Repository "committen" sowie die Änderungen anderer Teammitglieder herunterladen.
- GitHub ist eine Webseite, auf der wir Git-Repositories online anlegen können.

Git und GitHub

- Die PyCharm IDE erlaubt uns, im Grunde in einem Schritt die Beispiel-Kodes für diesen Kurs von GitHub herunterzuladen und die notwendigen Python-Pakete installieren.
- Dabei treffen wir auf zwei Konzepte: Git und die bereits diskutierten virtuellen Environments.
- Git ist ein verteiltes Versionsmanagementsystem – es erlaubt uns, sogenannte Repositorie zu erstellen, die im Grunde Verzeichnisse sind deren Änderungs- und Versionsgeschichte mit gespeichert wird und über die man kollaborativ arbeiten kann.
- Jedes Teammitglied arbeitet dazu auf einer lokalen Kopie des Kodes und kann ihre lokalen Änderungen in das Repository "committen" sowie die Änderungen anderer Teammitglieder herunterladen.
- GitHub ist eine Webseite, auf der wir Git-Repositories online anlegen können.
- Alle Beispiele dieses Kurses sind im Git Repository `https://github.com/thomasWeise/databasesCode` auf GitHub gespeichert.

Git und Klonen

- Der Prozess des Herunterladens eines Git-Repositories (inklusive seiner Versiongeschichte) wird *klonen* genannt.

Git und Klonen

- Der Prozess des Herunterladens eines Git-Repositories (inklusive seiner Versiongeschichte) wird *klonen* genannt.
- PyCharm ist eine IDE für die Softwareentwicklung in Python die Git unterstützt.

Git und Klonen

- Der Prozess des Herunterladens eines Git-Repositories (inklusive seiner Versiongeschichte) wird *klonen* genannt.
- PyCharm ist eine IDE für die Softwareentwicklung in Python die Git unterstützt.
- Sie können PyCharm verwenden, um Git-Repositories zu klonen und ein lokales virtuelles Environment zu erstellen, in dem die benötigten Pakete der Repositories installiert werden.

Git und Klonen

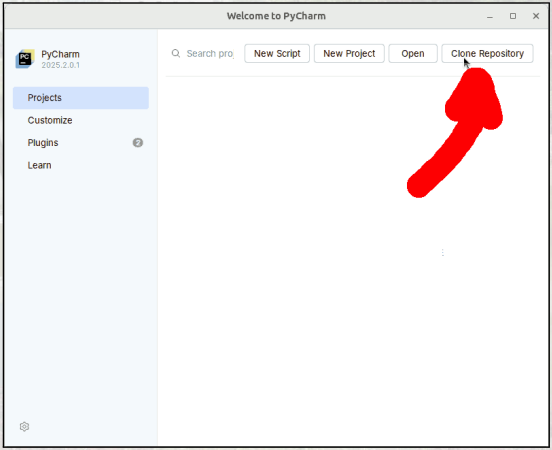
- Der Prozess des Herunterladens eines Git-Repositories (inklusive seiner Versiongeschichte) wird *klonen* genannt.
- PyCharm ist eine IDE für die Softwareentwicklung in Python die Git unterstützt.
- Sie können PyCharm verwenden, um Git-Repositories zu klonen und ein lokales virtuelles Environment zu erstellen, in dem die benötigten Pakete der Repositories installiert werden.
- Da wir auf unsere PostgreSQL Datenbank über das Python-Paket psycopg zugreifen, definiert unser Repository dieses Paket als benötigte Abhängigkeit.

Git und Klonen

- Der Prozess des Herunterladens eines Git-Repositories (inklusive seiner Versiongeschichte) wird *klonen* genannt.
- PyCharm ist eine IDE für die Softwareentwicklung in Python die Git unterstützt.
- Sie können PyCharm verwenden, um Git-Repositories zu klonen und ein lokales virtuelles Environment zu erstellen, in dem die benötigten Pakete der Repositories installiert werden.
- Da wir auf unsere PostgreSQL Datenbank über das Python-Paket psycopg zugreifen, definiert unser Repository dieses Paket als benötigte Abhängigkeit.
- Sie können also in einem Schritt das Repository mit den Beispielen klonen und die benötigten Pakete installieren.

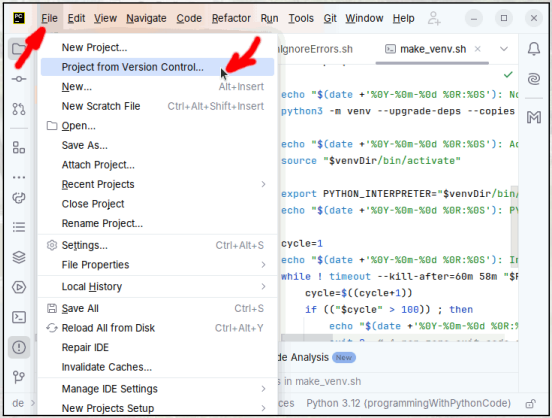
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Um ein Repository zu klonen, klicken Sie im Willkommensbildschirm von PyCharm auf Clone Repository.



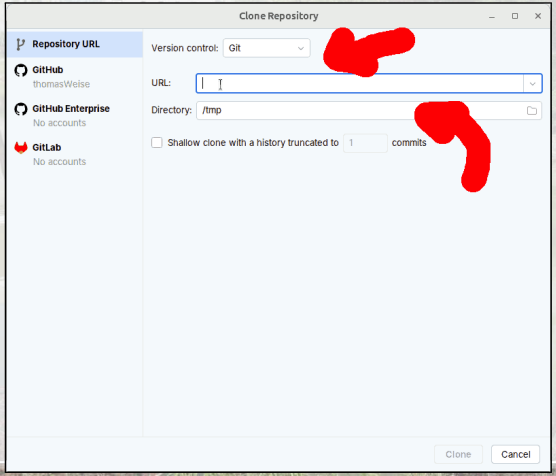
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Alternativ können Sie auch im File-Menü auf Project from Version Control... klicken.



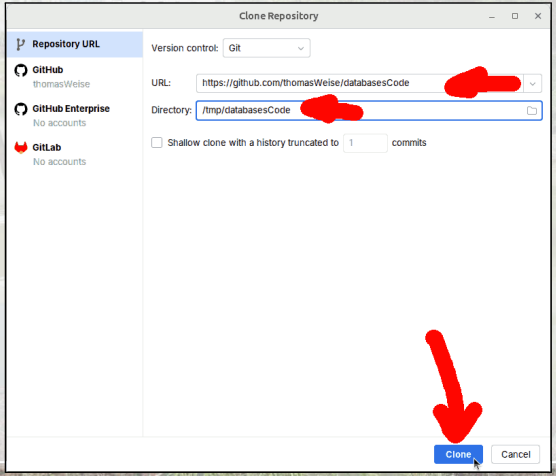
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Ein neues Formal öffnet sich. Wir müssen hier die URL des Git Repositories eingeben sowie den lokalen Ort wo die Dateien gespeichert werden sollen.



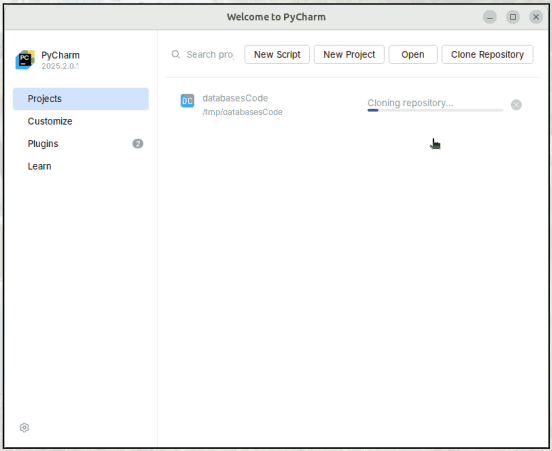
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Die URL unseres Repositories ist `https://github.com/thomasWeise/databasesCode`. Also lokalen Ort wähle ich den temporären Ordner meines Systems aus, weil ich die Daten gleich wieder löschen will. Sie werden einen anderen Ort auswählen. Wir klicken auf Clone.



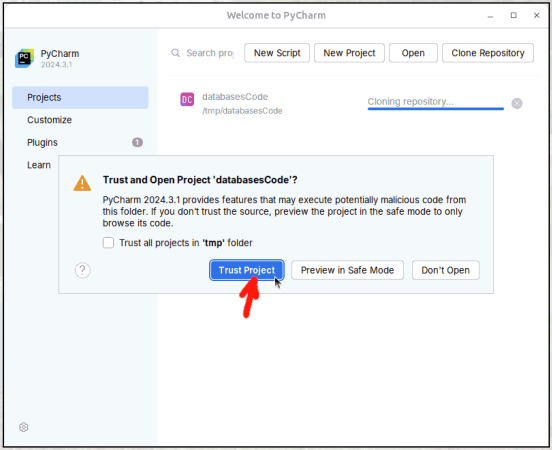
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Das Klonen beginnt und das Repository wird heruntergeladen.



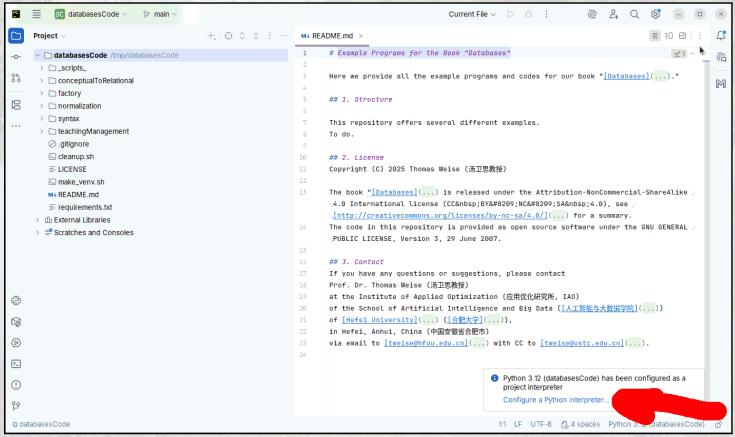
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Eventuell werden wir gefragt, ob wir diesem Projekt vertrauen wollen. Wenn Sie geprüft haben, dass alles OK ist, können Sie dem zustimmen und auf Trust Project klicken.



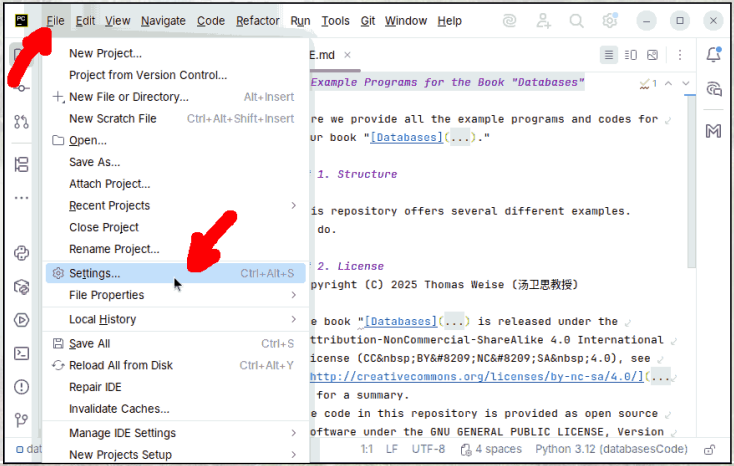
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Das Klonen ist fertig. Das Repository wurde heruntergeladen. Eventuell bemerkt PyCharm bereits, dass dieses Projekt Python-Pakete braucht und bietet uns an, einen Python-Interpreter zu konfigurieren.



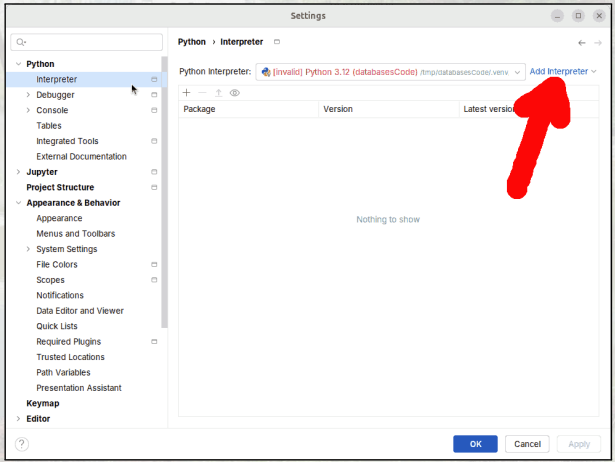
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Wir können den Python-Interpreter auch konfigurieren, in dem wir auf das Menü File klicken und dann Settings... auswählen, oder in dem wir direkt Ctrl + Alt + S drücken.



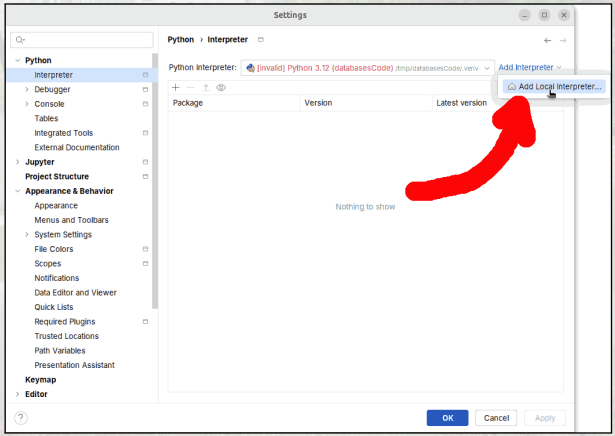
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Wir gehen zu den Python-Einstellungen und wählen Interpreter aus. Dann klicken wir Add Interpreter.



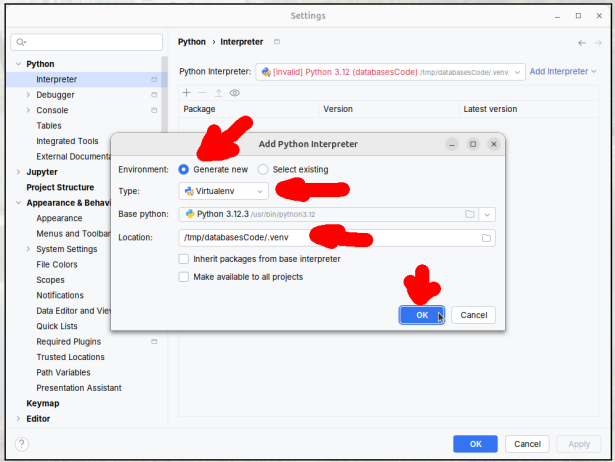
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- ... und dann klicken wir auf Add Local Interpreter... .



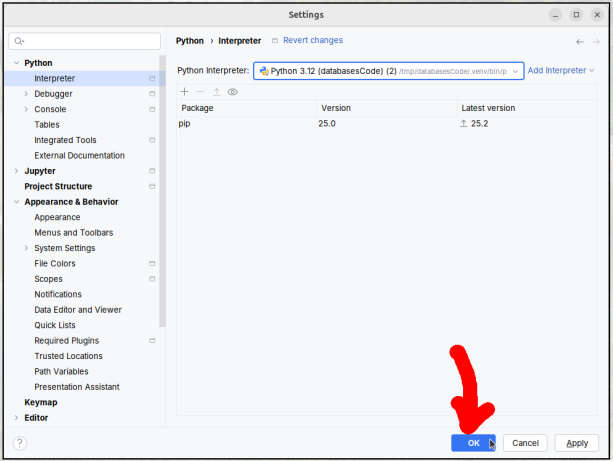
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- In dem sich öffnenden Dialog wählen wir Generate new aus. Als *Type* wählen wir Virtualenv. Als Ordner fügen wir einfach `.venv` an den Projektordner an. Dann klicken wir OK.



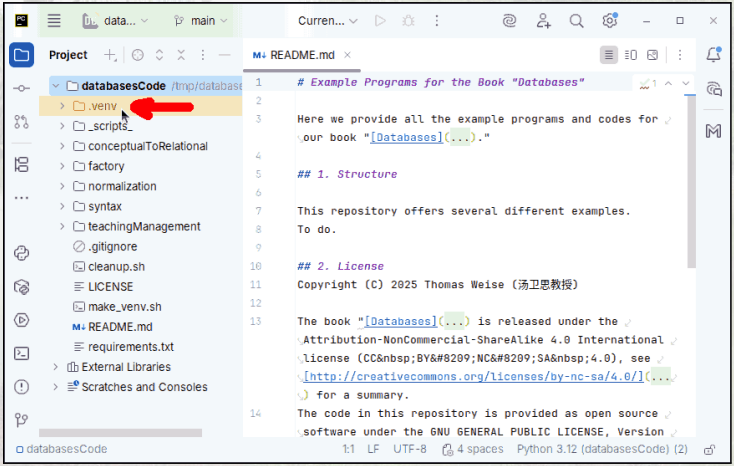
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Und dann klicken wir nochmal OK.



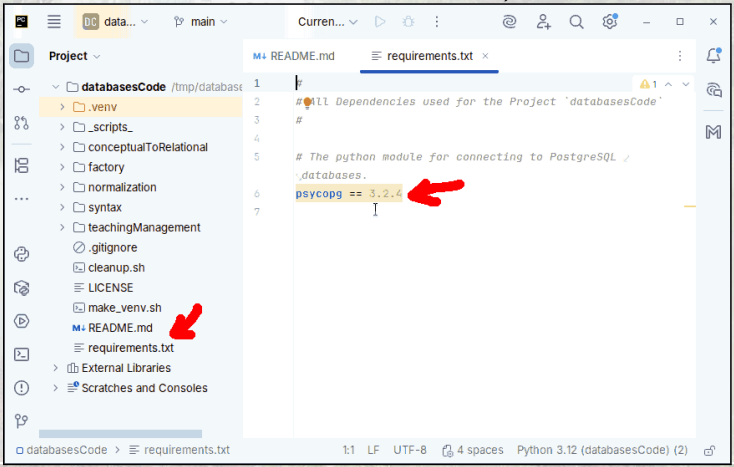
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Wie Sie sehen, wurde der Ordner `.venv` erstellt. Hier befindet sich das virtuelle Environment und hier werden die benötigten Python-Pakete lokal installiert.



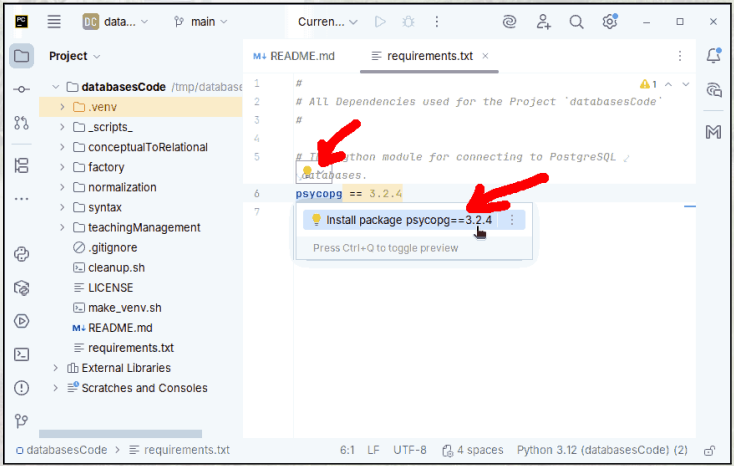
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Wir öffnen die Datei `requirements.txt` im Verzeichnis des Projekts. Diese Datei listet alle Python-Pakete, die unser Projekt braucht. Dazu gehört `psycopg` in Version `3.2.4`. (Natürlich werden wir die Version in Zukunft updaten.)



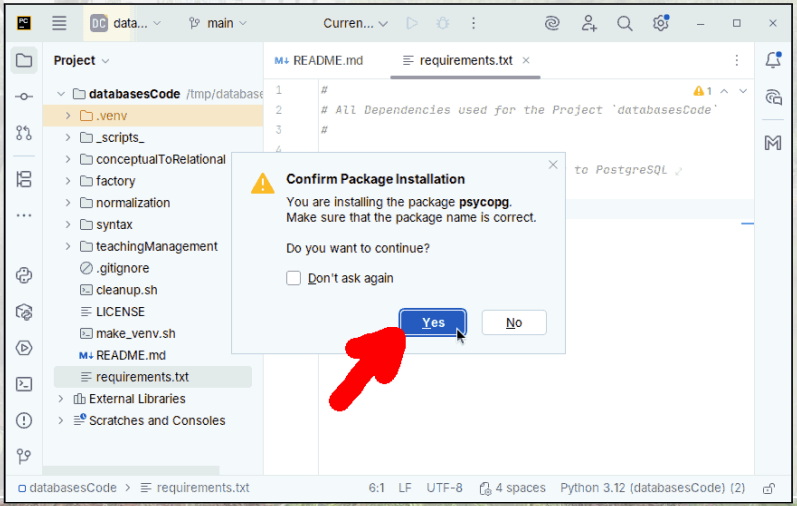
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Wir klicken auf die gelben Glühbirne bzw. die gelben Notifikationen. woraufhin sich ein Dialog öffnen, der uns fragt, ob wir nicht psycopg installieren möchten. Wir klicken da drauf.



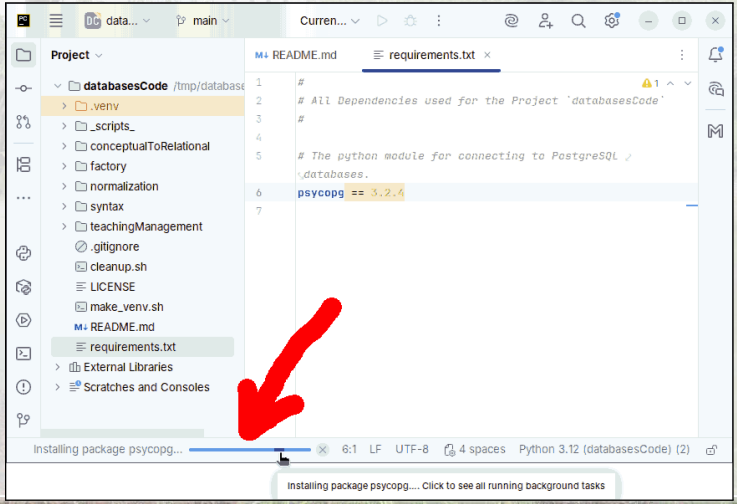
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Wir werden gebeten, die Installation zu bestätigen. Wir klicken auf Yes.



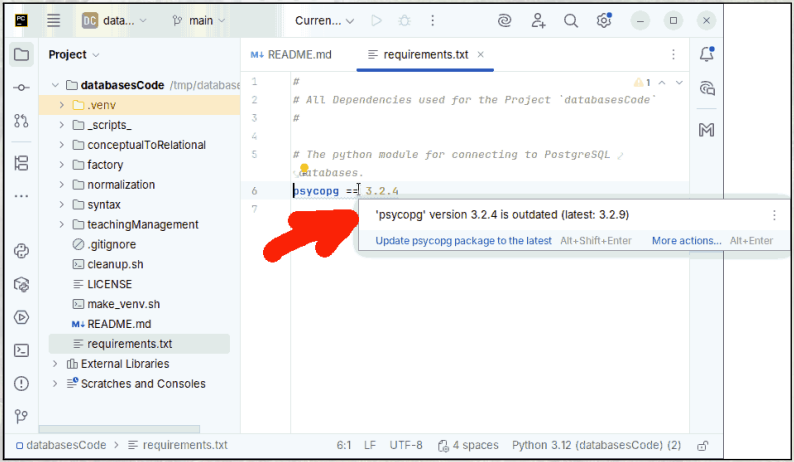
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Das Paket wird heruntergeladen.



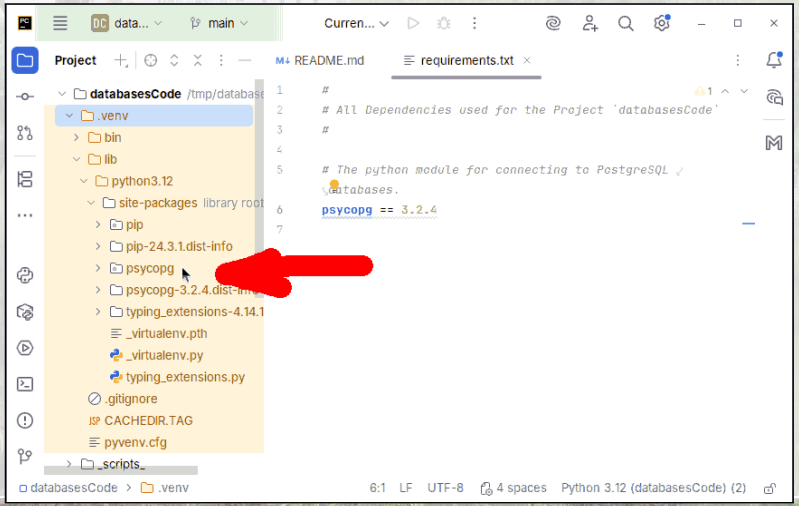
Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Das Paket wurde installiert. Gegebenenfalls informiert uns PyCharm, dass eine neue Version des Pakets zur Verfügung steht.



Beispiel-Repository Klonen und Psycopg Installieren mit PyCharm

- Tatsächlich können wir das Paket im Ordner `.venv` sehen.



Zusammenfassung

Zusammenfassung

- Es gibt zwei grundlegende Arten Python-Pakete zu installieren.

Zusammenfassung

- Es gibt zwei grundlegende Arten Python-Pakete zu installieren.
- Wollen wir eine Applikation im Produktiveinsatz benutzen, dann installieren wir die Pakete im Terminal.

Zusammenfassung

- Es gibt zwei grundlegende Arten Python-Pakete zu installieren.
- Wollen wir eine Applikation im Produktiveinsatz benutzen, dann installieren wir die Pakete im Terminal.
- Arbeitem wir mit PyCharm in der Software-Entwicklung, dann wollen können wir Pakete auch direkt in der IDE installieren.

Zusammenfassung

- Es gibt zwei grundlegende Arten Python-Pakete zu installieren.
- Wollen wir eine Applikation im Produktiveinsatz benutzen, dann installieren wir die Pakete im Terminal.
- Arbeitem wir mit PyCharm in der Software-Entwicklung, dann wollen können wir Pakete auch direkt in der IDE installieren.
- Beide Arten verwenden virtuelle Environments.

Zusammenfassung

- Es gibt zwei grundlegende Arten Python-Pakete zu installieren.
- Wollen wir eine Applikation im Produktiveinsatz benutzen, dann installieren wir die Pakete im Terminal.
- Arbeitem wir mit PyCharm in der Software-Entwicklung, dann wollen können wir Pakete auch direkt in der IDE installieren.
- Beide Arten verwenden virtuelle Environments.
- Sie können mehr über diese Verfahren in unserem Schwesterkurs *Programming with Python* [48] lernern.

Zusammenfassung

- Es gibt zwei grundlegende Arten Python-Pakete zu installieren.
- Wollen wir eine Applikation im Produktiveinsatz benutzen, dann installieren wir die Pakete im Terminal.
- Arbeitem wir mit PyCharm in der Software-Entwicklung, dann wollen können wir Pakete auch direkt in der IDE installieren.
- Beide Arten verwenden virtuelle Environments.
- Sie können mehr über diese Verfahren in unserem Schwesterkurs *Programming with Python* [48] lernern.
- So oder so, wir können nun psycopg verwenden, um von Python aus auf PostgreSQL-Datenbanken zuzugreifen.

谢谢您们！

Thank you!

Vielen Dank!

References I

[1] Daniel J. Barrett. *Efficient Linux at the Command Line*. Sebastopol, CA, USA: O'Reilly Media, Inc., Feb. 2022. ISBN: 978-1-0981-1340-7 (siehe S. 103, 104).

[2] Tim Berners-Lee. *Re: Qualifiers on Hypertext links...* Geneva, Switzerland: World Wide Web project, European Organization for Nuclear Research (CERN) und Newsgroups: alt.hypertext, 6. Aug. 1991. URL: https://www.w3.org/People/Berners-Lee/1991/08/art-6484.txt (besucht am 2025-02-05) (siehe S. 105).

[3] Tim Berners-Lee, Larry Masinter und Mark P. McCahill. *Uniform Resource Locators (URL)*. Request for Comments (RFC) 1738. Wilmington, DE, USA: Internet Engineering Task Force (IETF), Dez. 1994. URL: https://www.ietf.org/rfc/rfc1738.txt (besucht am 2025-02-05) (siehe S. 104).

[4] Alex Berson. *Client/Server Architecture*. 2. Aufl. Computer Communications Series. New York, NY, USA: McGraw-Hill, 29. März 1996. ISBN: 978-0-07-005664-0 (siehe S. 102).

[5] Ethan Bommarito und Michael Bommarito. *An Empirical Analysis of the Python Package Index (PyPI)*. arXiv.org: Computing Research Repository (CoRR) abs/1907.11073. Ithaca, NY, USA: Cornell Universiy Library, 26. Juli 2019. doi:10.48550/arXiv.1907.11073. URL: https://arxiv.org/abs/1907.11073 (besucht am 2024-08-17). arXiv:1907.11073v2 [cs.SE] 26 Jul 2019 (siehe S. 103).

[6] Ed Bott. *Windows 11 Inside Out*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Feb. 2023. ISBN: 978-0-13-769132-6 (siehe S. 103).

[7] Ron Brash und Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2018. ISBN: 978-1-78862-936-2 (siehe S. 102).

[8] Jason Cannon. *High Availability for the LAMP Stack*. Shelter Island, NY, USA: Manning Publications, Juni 2022 (siehe S. 104).

[9] David Clinton und Christopher Negus. *Ubuntu Linux Bible*. 10. Aufl. Bible Series. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., 10. Nov. 2020. ISBN: 978-1-119-72233-5 (siehe S. 104).

[10] Paul Deitel, Harvey Deitel und Abbey Deitel. *Internet & World Wide Web: How to Program*. 5. Aufl. Hoboken, NJ, USA: Pearson Education, Inc., Nov. 2011. ISBN: 978-0-13-299045-5 (siehe S. 105).

References II

[11] Justin Dennison, Cherokee Boose und Peter van Rysdam. *Intro to NumPy*. Centennial, CO, USA: ACI Learning. Birmingham, England, UK: Packt Publishing Ltd, Juni 2024. ISBN: 978-1-83620-863-1 (siehe S. 103).

[12] Luca Ferrari und Enrico Pirozzi. *Learn PostgreSQL*. 2. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Okt. 2023. ISBN: 978-1-83763-564-1 (siehe S. 103).

[13] Michael Goodwin. *What is an API?* Armonk, NY, USA: International Business Machines Corporation (IBM), 9. Apr. 2024. URL: https://www.ibm.com/topics/api (besucht am 2024-12-12) (siehe S. 102).

[14] Charles R. Harris, K. Jarrod Millman, Stéfan van der Walt, Ralf Gommers, Pauli "pv" Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke und Travis E. Oliphant. "Array programming with NumPy". *Nature* 585:357–362, 2020. London, England, UK: Springer Nature Limited. ISSN: 0028-0836. doi:10.1038/S41586-020-2649-2 (siehe S. 103).

[15] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (siehe S. 103).

[16] Matthew Helmke. *Ubuntu Linux Unleashed 2021 Edition*. 14. Aufl. Reading, MA, USA: Addison-Wesley Professional, Aug. 2020. ISBN: 978-0-13-668539-5 (siehe S. 104).

[17] John Hunt. *A Beginners Guide to Python 3 Programming*. 2. Aufl. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2023. ISBN: 978-3-031-35121-1. doi:10.1007/978-3-031-35122-8 (siehe S. 104).

[18] *Information Technology – Universal Coded Character Set (UCS)*. International Standard ISO/IEC 10646:2020. Geneva, Switzerland: International Organization for Standardization (ISO) und International Electrotechnical Commission (IEC), Dez. 2020 (siehe S. 101).

[19] *Python 3 Documentation. Installing Python Modules*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. URL: https://docs.python.org/3/installing (besucht am 2024-08-17) (siehe S. 103).

[20] Robert Johansson. *Numerical Python: Scientific Computing and Data Science Applications with NumPy, SciPy and Matplotlib*. New York, NY, USA: Apress Media, LLC, Dez. 2018. ISBN: 978-1-4842-4246-9 (siehe S. 103).

References III

[21] Jay LaCroix. *Mastering Ubuntu Server*. 4. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Sep. 2022. ISBN: 978-1-80323-424-3 (siehe S. 104).

[22] Kent D. Lee und Steve Hubbard. *Data Structures and Algorithms with Python*. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2015. ISBN: 978-3-319-13071-2. doi:10.1007/978-3-319-13072-9 (siehe S. 104).

[23] Marc-André Lemburg. *Python Database API Specification v2.0*. Python Enhancement Proposal (PEP) 249. Beaverton, OR, USA: Python Software Foundation (PSF), 12. Apr. 1999. URL: https://peps.python.org/pep-0249 (besucht am 2025-02-02) (siehe S. 103).

[24] Gloria Lotha, Aakanksha Gaur, Erik Gregersen, Swati Chopra und William L. Hosch. "Client-Server Architecture". In: *Encyclopaedia Britannica*. Hrsg. von The Editors of Encyclopaedia Britannica. Chicago, IL, USA: Encyclopædia Britannica, Inc., 3. Jan. 2025. URL: https://www.britannica.com/technology/client-server-architecture (besucht am 2025-01-20) (siehe S. 102).

[25] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 104).

[26] Carl Meyer. *Python Virtual Environments*. Python Enhancement Proposal (PEP) 405. Beaverton, OR, USA: Python Software Foundation (PSF), 13. Juni 2011–24. Mai 2012. URL: https://peps.python.org/pep-0405 (besucht am 2024-12-25) (siehe S. 105).

[27] Cameron Newham und Bill Rosenblatt. *Learning the Bash Shell – Unix Shell Programming: Covers Bash 3.0*. 3. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., 2005. ISBN: 978-0-596-00965-6 (siehe S. 102).

[28] NumPy Team. *NumPy*. San Francisco, CA, USA: GitHub Inc und Austin, TX, USA: NumFOCUS, Inc. URL: https://numpy.org (besucht am 2025-02-02) (siehe S. 103).

[29] Regina O. Obe und Leo S. Hsu. *PostgreSQL: Up and Running*. 3. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Okt. 2017. ISBN: 978-1-4919-6336-4 (siehe S. 103).

[30] Robert Orfali, Dan Harkey und Jeri Edwards. *Client/Server Survival Guide*. 3. Aufl. Chichester, West Sussex, England, UK: John Wiley and Sons Ltd., 25. Jan. 1999. ISBN: 978-0-471-31615-2 (siehe S. 102).

References IV

[31] Yasset Pérez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, Tobias Ternent, Stephen J. Eglen, Daniel S. Katz, Tom J. Pollard, Alexander Konovalov, Robert M. Flight, Kai Blin und Juan Antonio Vizcaíno. "Ten Simple Rules for Taking Advantage of Git and GitHub". *PLOS Computational Biology* 12(7), 14. Juli 2016. San Francisco, CA, USA: Public Library of Science (PLOS). ISSN: 1553-7358. doi:10.1371/JOURNAL.PCBI.1004947 (siehe S. 102).

[32] pip Developers. *pip Documentation v24.3.1*. Beaverton, OR, USA: Python Software Foundation (PSF), 27. Okt. 2024. URL: https://pip.pypa.io (besucht am 2024-12-25) (siehe S. 103).

[33] *PostgreSQL Essentials: Leveling Up Your Data Work*. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2024 (siehe S. 103).

[34] Abhishek Ratan, Eric Chou, Pradeeban Kathiravelu und Dr. M.O. Faruque Sarker. *Python Network Programming*. Birmingham, England, UK: Packt Publishing Ltd, Jan. 2019. ISBN: 978-1-78883-546-6 (siehe S. 102).

[35] Mark Richards und Neal Ford. *Fundamentals of Software Architecture: An Engineering Approach*. Sebastopol, CA, USA: O'Reilly Media, Inc., Jan. 2020. ISBN: 978-1-4920-4345-4 (siehe S. 102).

[36] Ellen Siever, Stephen Figgins, Robert Love und Arnold Robbins. *Linux in a Nutshell*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Sep. 2009. ISBN: 978-0-596-15448-6 (siehe S. 103).

[37] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mai 2023. ISBN: 978-1-0981-3391-7 (siehe S. 102).

[38] Alkin Tezuysal und Ibrar Ahmed. *Database Design and Modeling with PostgreSQL and MySQL*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2024. ISBN: 978-1-80323-347-5 (siehe S. 103).

[39] *The Python Package Index (PyPI)*. Beaverton, OR, USA: Python Software Foundation (PSF), 2024. URL: https://pypi.org (besucht am 2024-08-17) (siehe S. 103).

[40] Linus Torvalds. "The Linux Edge". *Communications of the ACM (CACM)* 42(4):38–39, Apr. 1999. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0001-0782. doi:10.1145/299157.299165 (siehe S. 103).

[41] Mariot Tsitoara. *Beginning Git and GitHub: Version Control, Project Management and Teamwork for the New Developer*. New York, NY, USA: Apress Media, LLC, März 2024. ISBN: 979-8-8688-0215-7 (siehe S. 102, 104).

References V

[42] Marat Valiev, Bogdan Vasilescu und James D. Herbsleb. "Ecosystem-Level Determinants of Sustained Activity in Open-Source Projects: A Case Study of the PyPI Ecosystem". In: *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/SIGSOFT FSE'2018)*. 4.–9. Nov. 2018, Lake Buena Vista, FL, USA. Hrsg. von Gary T. Leavens, Alessandro F. Garcia und Corina S. Păsăreanu. New York, NY, USA: Association for Computing Machinery (ACM), 2018, S. 644–655. ISBN: 978-1-4503-5573-5. doi:10.1145/3236024.3236062 (siehe S. 103).

[43] Bruce M. Van Horn II und Quan Nguyen. *Hands-On Application Development with PyCharm*. 2. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Okt. 2023. ISBN: 978-1-83763-235-0 (siehe S. 103).

[44] Sander van Vugt. *Linux Fundamentals*. 2. Aufl. Hoboken, NJ, USA: Pearson IT Certification, Juni 2022. ISBN: 978-0-13-792931-3 (siehe S. 103).

[45] Daniele "dvarrazzo" Varrazzo, Federico "fogzot" Di Gregorio und Jason "jerickso" Erickson. *Psycopg*. London, England, UK: The Psycopg Team, 2010–2023. URL: https://www.psycopg.org (besucht am 2025-02-02) (siehe S. 5–10, 103).

[46] "Virtual Environments and Packages". In: *Python 3 Documentation*. *The Python Tutorial*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. Kap. 12. URL: https://docs.python.org/3/tutorial/venv.html (besucht am 2024-12-24) (siehe S. 105).

[47] Thomas Weise (汤卫思). *Databases*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), Institute of Applied Optimization (应用优化研究所, IAO), 2025. URL: https://thomasweise.github.io/databases (besucht am 2025-01-05) (siehe S. 102).

[48] Thomas Weise (汤卫思). *Programming with Python*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), Institute of Applied Optimization (应用优化研究所, IAO), 2024–2025. URL: https://thomasweise.github.io/programmingWithPython (besucht am 2025-01-05) (siehe S. 26–37, 89–94, 104).

[49] Kevin Wilson. *Python Made Easy*. Birmingham, England, UK: Packt Publishing Ltd, Aug. 2024. ISBN: 978-1-83664-615-0 (siehe S. 103).

[50] Martin Yanev. *PyCharm Productivity and Debugging Techniques*. Birmingham, England, UK: Packt Publishing Ltd, Okt. 2022. ISBN: 978-1-83763-244-2 (siehe S. 103).

References VI

[51] Kinza Yasar und Craig S. Mullins. *Definition: Database Management System (DBMS)*. Newton, MA, USA: TechTarget, Inc., Juni 2024. URL: https://www.techtarget.com/searchdatamanagement/definition/database-management-system (besucht am 2025-01-11) (siehe S. 102).

[52] François Yergeau. *UTF-8, A Transformation Format of ISO 10646*. Request for Comments (RFC) 3629. Wilmington, DE, USA: Internet Engineering Task Force (IETF), Nov. 2003. URL: https://www.ietf.org/rfc/rfc3629.txt (besucht am 2025-02-05). See Unicode and[18] (siehe S. 104).

[53] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, Juni 2017. ISBN: 978-1-78439-687-9 (siehe S. 102).

Glossary (in English) I

API An *Application Programming Interface* is a set of rules or protocols that enables one software application or component to use or communicate with another[13].

Bash is a the shell used under Ubuntu Linux, i.e., the program that "runs" in the terminal and interprets your commands, allowing you to start and interact with other programs[7,27,53]. Learn more at https://www.gnu.org/software/bash.

client In a client-server architecture, the client is a device or process that requests a service from the server. It initiates the communication with the server, sends a request, and receives the response with the result of the request. Typical examples for clients are web browsers in the internet as well as clients for database management systems (DBMSes), such as psql.

client-server architecture is a system design where a central server receives requests from one or multiple clients[4,24,30,34,35]. These requests and responses are usually sent over network connections. A typical example for such a system is the World Wide Web (WWW), where web servers host websites and make them available to web browsers, the clients. Another typical example is the structure of database (DB) software, where a central server, the DBMS, offers access to the DB to the different clients. Here, the client can be some terminal software shipping with the DBMS, such as psql, or the different applications that access the DBs.

DB A *database* is an organized collection of structured information or data, typically stored electronically in a computer system. Databases are discussed in our book *Databases*[47].

DBMS A *database management system* is the software layer located between the user or application and the DB. The DBMS allows the user/application to create, read, write, update, delete, and otherwise manipulate the data in the DB[51].

Git is a distributed Version Control Systems (VCS) which allows multiple users to work on the same code while preserving the history of the code changes[37,41]. Learn more at https://git-scm.com.

GitHub is a website where software projects can be hosted and managed via the Git VCS[31,41]. Learn more at https://github.com.

Glossary (in English) II

IDE An *Integrated Developer Environment* is a program that allows the user do multiple different activities required for software development in one single system. It often offers functionality such as editing source code, debugging, testing, or interaction with a distributed version control system. For Python, we recommend using PyCharm.

Linux is the leading open source operating system, i.e., a free alternative for Microsoft Windows[1,15,36,40,44]. We recommend using it for this course, for software development, and for research. Learn more at https://www.linux.org. Its variant Ubuntu is particularly easy to use and install.

Microsoft Windows is a commercial proprietary operating system[6]. It is widely spread, but we recommend using a Linux variant such as Ubuntu for software development and for our course. Learn more at https://www.microsoft.com/windows.

NumPy is a fundamental package for scientific computing with Python, which offers efficient array datastructures[11,14,20]. Learn more at https://numpy.org[28].

pip is the standard tool to install Python software packages from the PyPI repository[19,32]. To install a package `thepackage` hosted on PyPI, type `pip install thepackage` into the terminal. Learn more at https://packaging.python.org/installing.

PostgreSQL An open source object-relational DBMS[12,29,33,38]. See https://postgresql.org for more information.

psql is the client program used to access the PostgreSQL DBMS server.

psycopg or, more exactly, `psycopg 3`, is the most popular PostgreSQL adapter for Python, implementing the Python DB API 2.0 specification[23]. Learn more at https://www.psycopg.org[45].

PyCharm is the convenient Python Integrated Development Environment (IDE) that we recommend for this course[43,49,50]. It comes in a free community edition, so it can be downloaded and used at no cost. Learn more at https://www.jetbrains.com/pycharm.

PyPI The Python Package Index (PyPI) is an online repository that provides the software packages that you can install with pip[5,39,42]. Learn more at https://pypi.org.

Glossary (in English) III

Python The Python programming language[17,22,25,48], i.e., what you will learn about in our book[48]. Learn more at https://python.org.

server In a client-server architecture, the server is a process that fulfills the requests of the clients. It usually waits for incoming communication carring the requests from the clients. For each request, it takes the necessary actions, performs the required computations, and then sends a response with the result of the request. Typical examples for servers are web servers[8] in the internet as well as DBMSes. It is also common to refer to the computer running the server processes as server as well, i.e., to call it the "server computer"[21].

terminal A terminal is a text-based window where you can enter commands and execute them[1,9]. Knowing what a terminal is and how to use it is very essential in any programming- or system administration-related task. If you want to open a terminal under Microsoft Windows, you can Druck auf ⊞+R, dann Schreiben von `cmd`, dann Druck auf ↵. Under Ubuntu Linux, Ctrl+Alt+T opens a terminal, which then runs a Bash shell inside.

Ubuntu is a variant of the open source operating system Linux[9,16]. We recommend that you use this operating system to follow this class, for software development, and for research. Learn more at https://ubuntu.com. If you are in China, you can download it from https://mirrors.ustc.edu.cn/ubuntu-releases.

URI A *Uniform Resource Identifier* is an identifier for an abstract or physical resource in the internet[52]. It can be a URL, a name, or both. URIs are supersets of URLs. The connection strings of the PostgreSQL DBMS are examples for URIs.

URL A *Uniform Resource Locator* identifies a resource in the WWW and a way to obtain it by describing a network access mechanism. The most notable example of URLs is the text you write into web browsers to visit websites[3]. URLs are subsets of Uniform Resource Identifiers (URIs).

VCS A *Version Control System* is a software which allows you to manage and preserve the historical development of your program code[41]. A distributed VCS allows multiple users to work on the same code and upload their changes to the server, which then preserves the change history. The most popular distributed VCS is Git.

Glossary (in English) IV

virtual environment A virtual environment is a directory that contains a local Python installation[26,46]. It comes with its own package installation directory. Multiple different virtual environments can be installed on a system. This allows different applications to use different versions of the same packages without conflict, because we can simply install these applications into different virtual environments.

WWW World Wide Web[2,10]