



Programming with Python

5. Programme Erstellen und Ausführen

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

Programming with Python



Dies ist ein Kurs über das Programmieren mit der Programmiersprache Python an der Universität Hefei (合肥大学).

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



Outline

1. Einleitung
2. Das Erste PyCharm Projekt und Programm
3. Programm im Terminal ausführen
4. Programm in Python Interpreter in PyCharm Eingeben
5. Programm in Python Interpreter in Terminal Eingeben
6. Zusammenfassung





Einleitung



Einleitung

- Jetzt haben wir PyCharm und Python installiert.



Einleitung



- Jetzt haben wir PyCharm und Python installiert.
- Nun wollen wir unser erstes Python-Programm schreiben und ausführen.

Einleitung



- Jetzt haben wir PyCharm und Python installiert.
- Nun wollen wir unser erstes Python-Programm schreiben und ausführen.
- Das Programm soll einfach „Hello World!“ zum standard output stream (stdout) schreiben.



Einleitung

- Jetzt haben wir PyCharm und Python installiert.
- Nun wollen wir unser erstes Python-Programm schreiben und ausführen.
- Das Programm soll einfach „Hello World!“ zum standard output stream (stdout) schreiben.
- Es besteht daher nur aus dem Statement `print("Hello World!")`.



Einleitung

- Jetzt haben wir PyCharm und Python installiert.
- Nun wollen wir unser erstes Python-Programm schreiben und ausführen.
- Das Programm soll einfach „Hello World!“ zum standard output stream (stdout) schreiben.
- Es besteht daher nur aus dem Statement `print("Hello World!")`.

```
1 print("Hello World!")
```

↓ python3 very_first_program.py ↓

```
1 Hello World!
```

Programme Ausführen



- Es gibt 4 grundsätzliche Methoden, Python-Programme auszuführen.

Programme Ausführen



- Es gibt 4 grundsätzliche Methoden, Python-Programme auszuführen
 1. Wir können das Programm in der PyCharm Integrated Development Environment (IDE) in eine Python-Datei schreiben und in PyCharm ausführen.

Programme Ausführen



- Es gibt 4 grundsätzliche Methoden, Python-Programme auszuführen
 1. Wir können das Programm in der PyCharm Integrated Development Environment (IDE) in eine Python-Datei schreiben und in PyCharm ausführen.
 2. Wir können das Programm auch in einem normalen Text-Editor schreiben.

Programme Ausführen



- Es gibt 4 grundsätzliche Methoden, Python-Programme auszuführen
 1. Wir können das Programm in der PyCharm Integrated Development Environment (IDE) in eine Python-Datei schreiben und in PyCharm ausführen.
 2. Wir können das Programm auch in einem normalen Text-Editor schreiben.
Python-Programme sind ja im Grunde normale Text-Dateien.

Programme Ausführen



- Es gibt 4 grundsätzliche Methoden, Python-Programme auszuführen
 1. Wir können das Programm in der PyCharm Integrated Development Environment (IDE) in eine Python-Datei schreiben und in PyCharm ausführen.
 2. Wir können das Programm auch in einem normalen Text-Editor schreiben. Python-Programme sind ja im Grunde normale Text-Dateien. Dann können wir das Programm mit dem Befehl `python3 programName` im Terminal ausführen.

Programme Ausführen



- Es gibt 4 grundsätzliche Methoden, Python-Programme auszuführen
 1. Wir können das Programm in der PyCharm Integrated Development Environment (IDE) in eine Python-Datei schreiben und in PyCharm ausführen.
 2. Wir können das Programm auch in einem normalen Text-Editor schreiben.
Python-Programme sind ja im Grunde normale Text-Dateien. Dann können wir das Programm mit dem Befehl `python3 programName` im Terminal ausführen.
 3. Wir können auch die Python-Interpreter Konsole in PyCharm öffnen und das Programm Zeile-für-Zeile eintippen und ausführen.

Programme Ausführen



- Es gibt 4 grundsätzliche Methoden, Python-Programme auszuführen
 1. Wir können das Programm in der PyCharm Integrated Development Environment (IDE) in eine Python-Datei schreiben und in PyCharm ausführen.
 2. Wir können das Programm auch in einem normalen Text-Editor schreiben.
Python-Programme sind ja im Grunde normale Text-Dateien. Dann können wir das Programm mit dem Befehl `python3 programName` im Terminal ausführen.
 3. Wir können auch die Python-Interpreter Konsole in PyCharm öffnen und das Programm Zeile-für-Zeile eintippen und ausführen.
 4. Natürlich können wir genausogut den Python-Interpreter im normalen Terminal öffnen und die Befehle Zeile-für-Zeile dort eintippen.

Programme Ausführen



- Es gibt 4 grundsätzliche Methoden, Python-Programme auszuführen
 1. Wir können das Programm in der PyCharm Integrated Development Environment (IDE) in eine Python-Datei schreiben und in PyCharm ausführen.
 2. Wir können das Programm auch in einem normalen Text-Editor schreiben.
Python-Programme sind ja im Grunde normale Text-Dateien. Dann können wir das Programm mit dem Befehl `python3 programName` im Terminal ausführen.
 3. Wir können auch die Python-Interpreter Konsole in PyCharm öffnen und das Programm Zeile-für-Zeile eintippen und ausführen.
 4. Natürlich können wir genausogut den Python-Interpreter im normalen Terminal öffnen und die Befehle Zeile-für-Zeile dort eintippen.
- Schauen wir uns diese Möglichkeiten einmal an.



Das Erste PyCharm Projekt und Programm



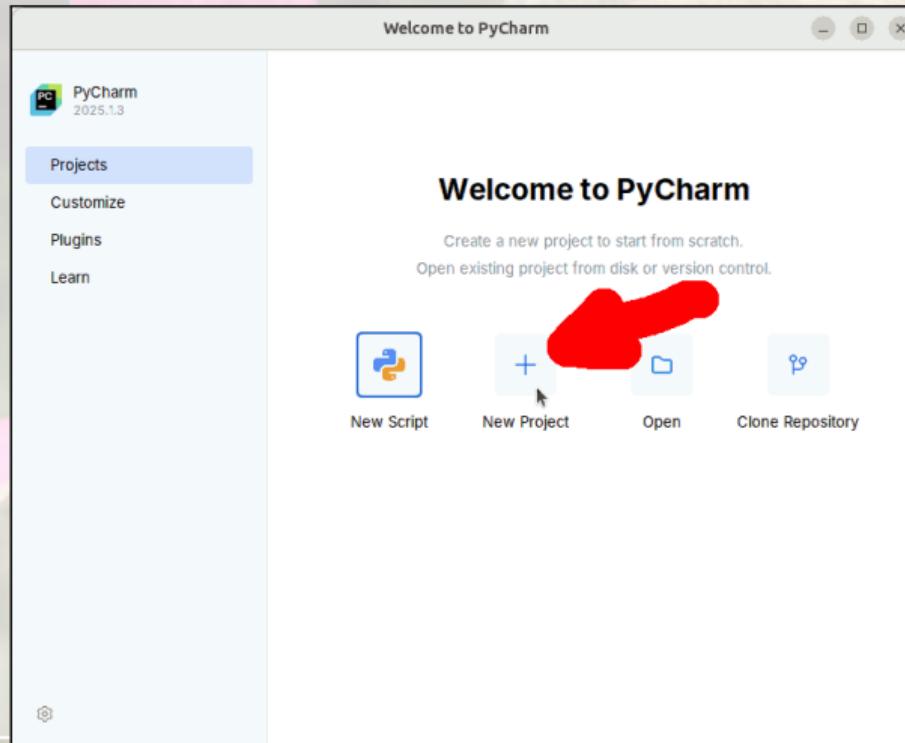
Das Erste PyCharm Projekt und Programm

- OK, los geht's.



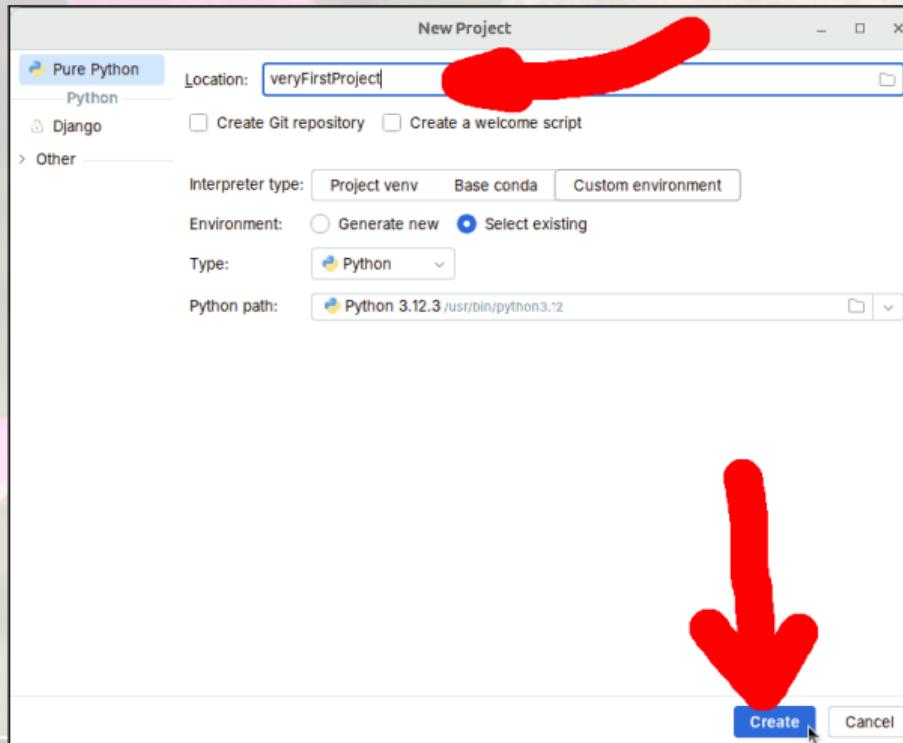
Das Erste PyCharm Projekt und Programm

- Um ein neues Projekt in PyCharm zu erstellen, klicken wir auf **New Project** im Willkommensbildschirm.



Das Erste PyCharm Projekt und Programm

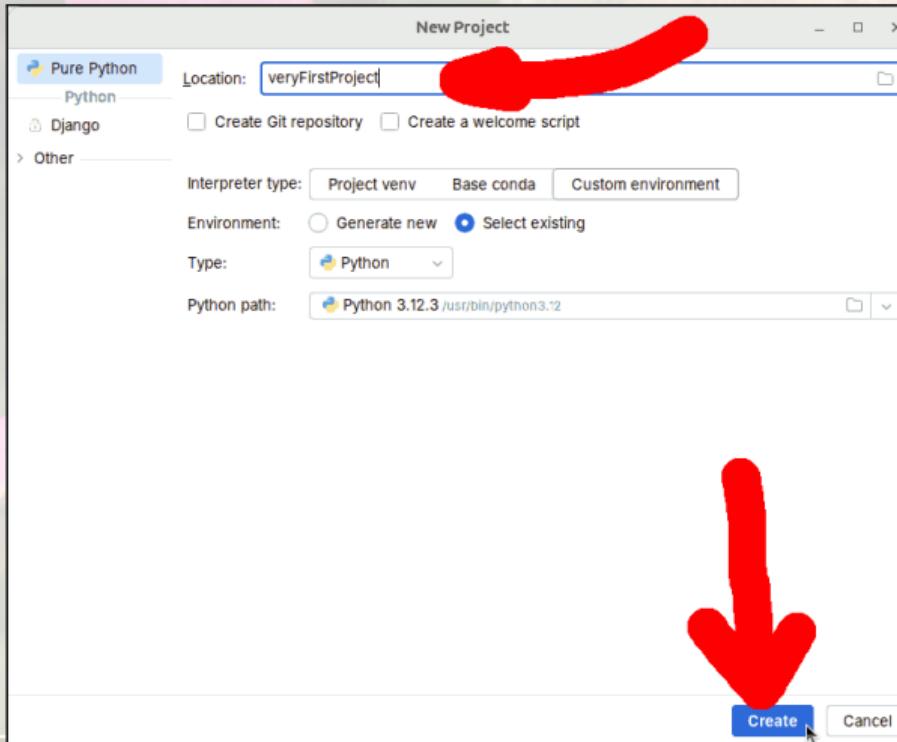
- Wir wählen links **Pure Python** aus und dann einen Namen für das Projekt.



Das Erste PyCharm Projekt und Programm

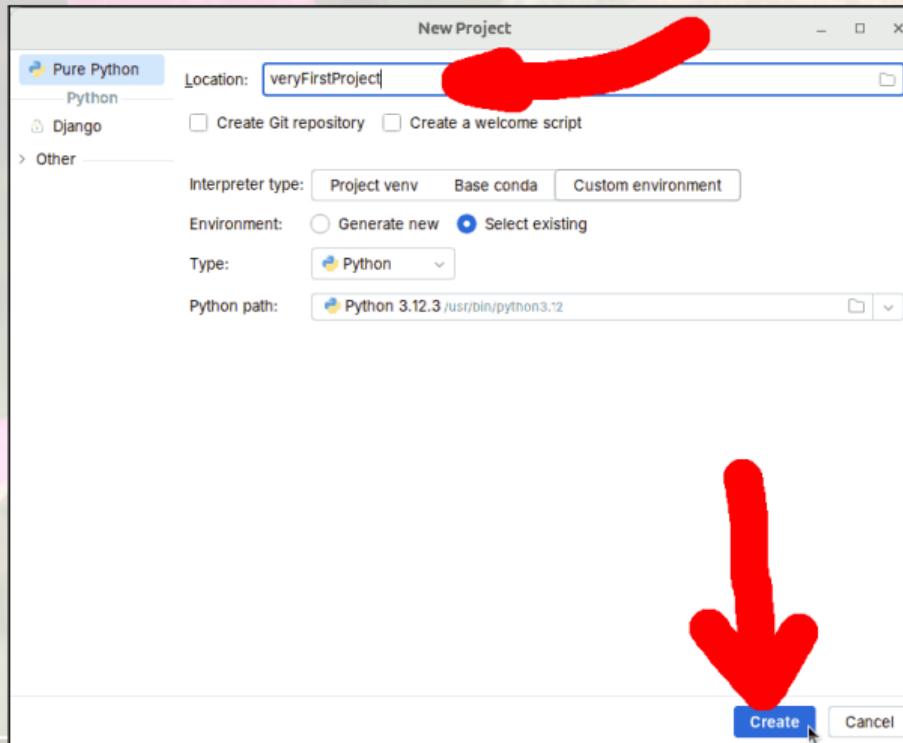


- Wir wählen links **Pure Python** aus und dann einen Namen für das Projekt.
- Wir wählen **veryFirstProject** als Name.



Das Erste PyCharm Projekt und Programm

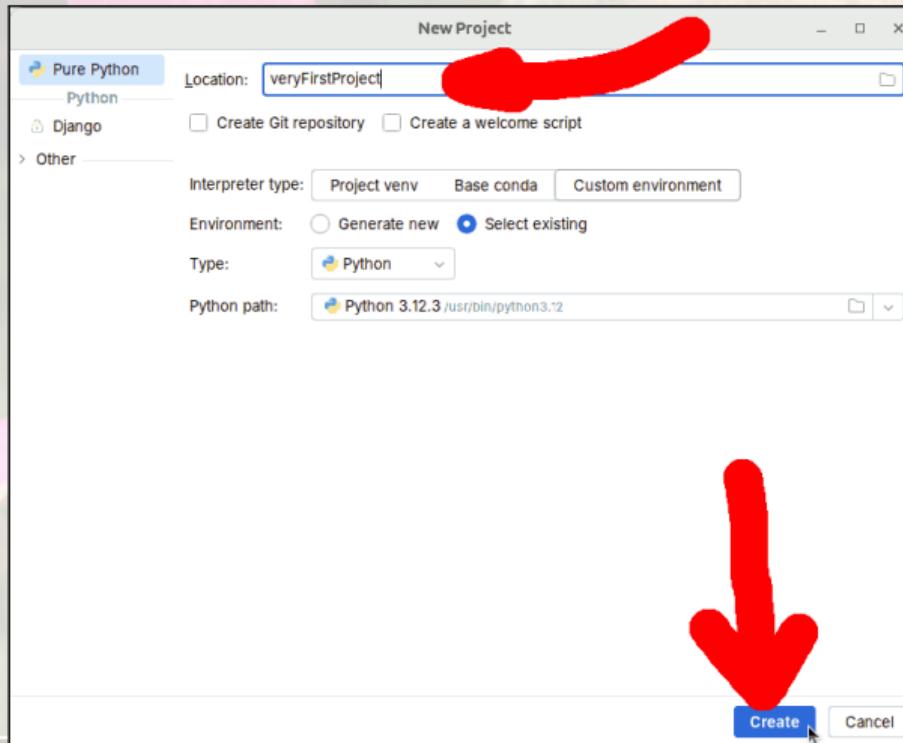
- Wir wählen veryFirstProject als Name.
- Wir wählen auch das Verzeichnis aus, in dem das Projekt gespeichert werden soll.



Das Erste PyCharm Projekt und Programm

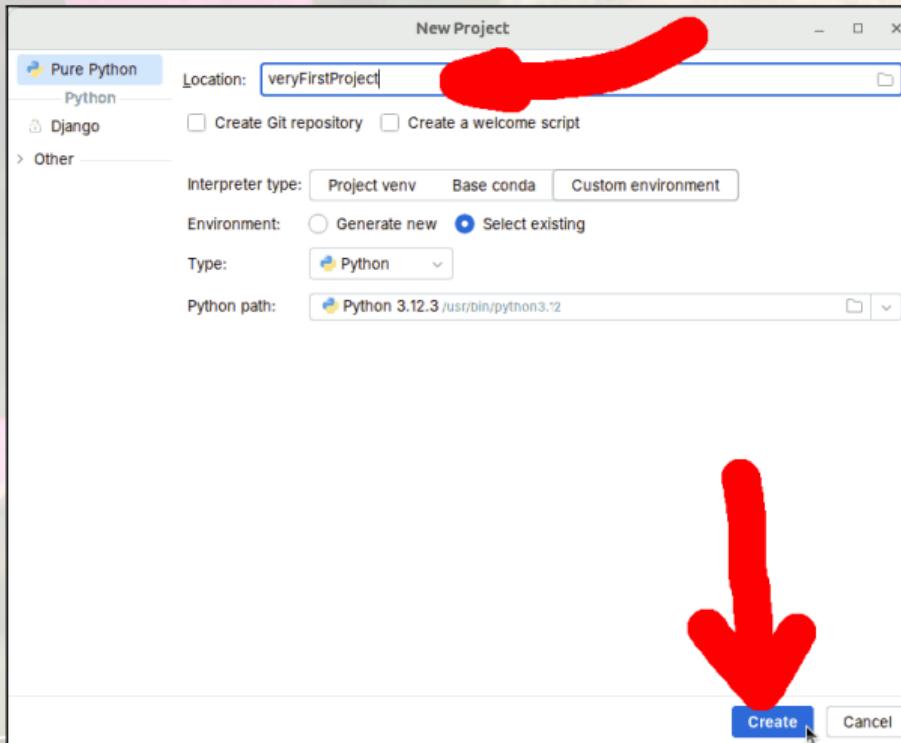


- Wir lassen die anderen Einstellungen auf den Standardwerten und/oder wählen unsere Python-Installation als **Custom Environment** aus.



Das Erste PyCharm Projekt und Programm

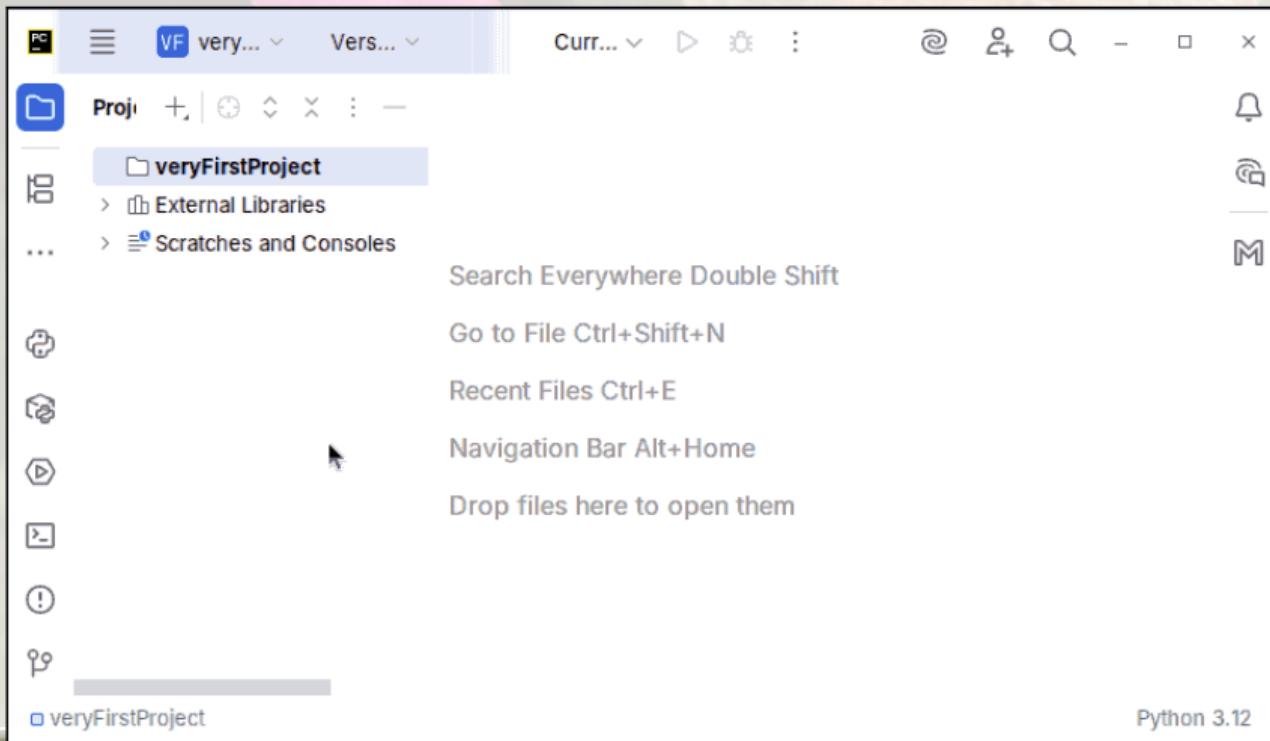
- Dann klicken wir auf **Create**.



Das Erste PyCharm Projekt und Programm



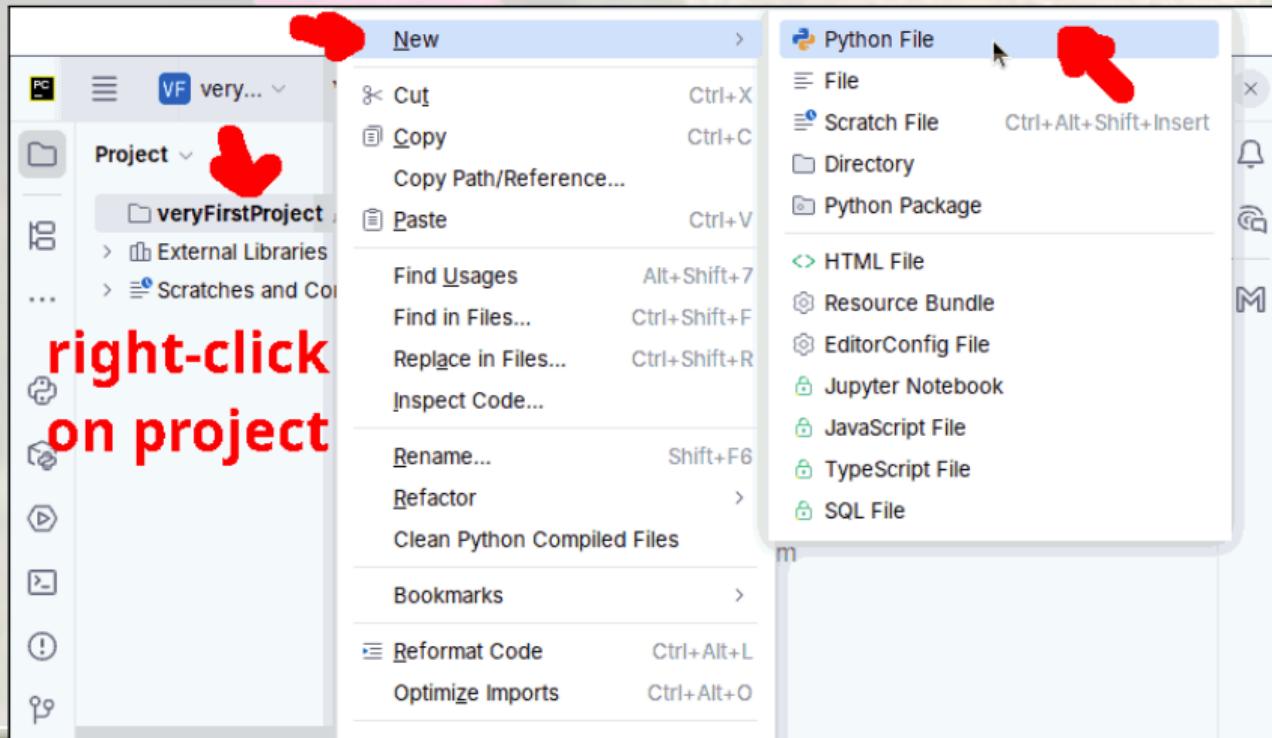
- Ein neues, leeres Projekt wurde erstellt.



Das Erste PyCharm Projekt und Programm



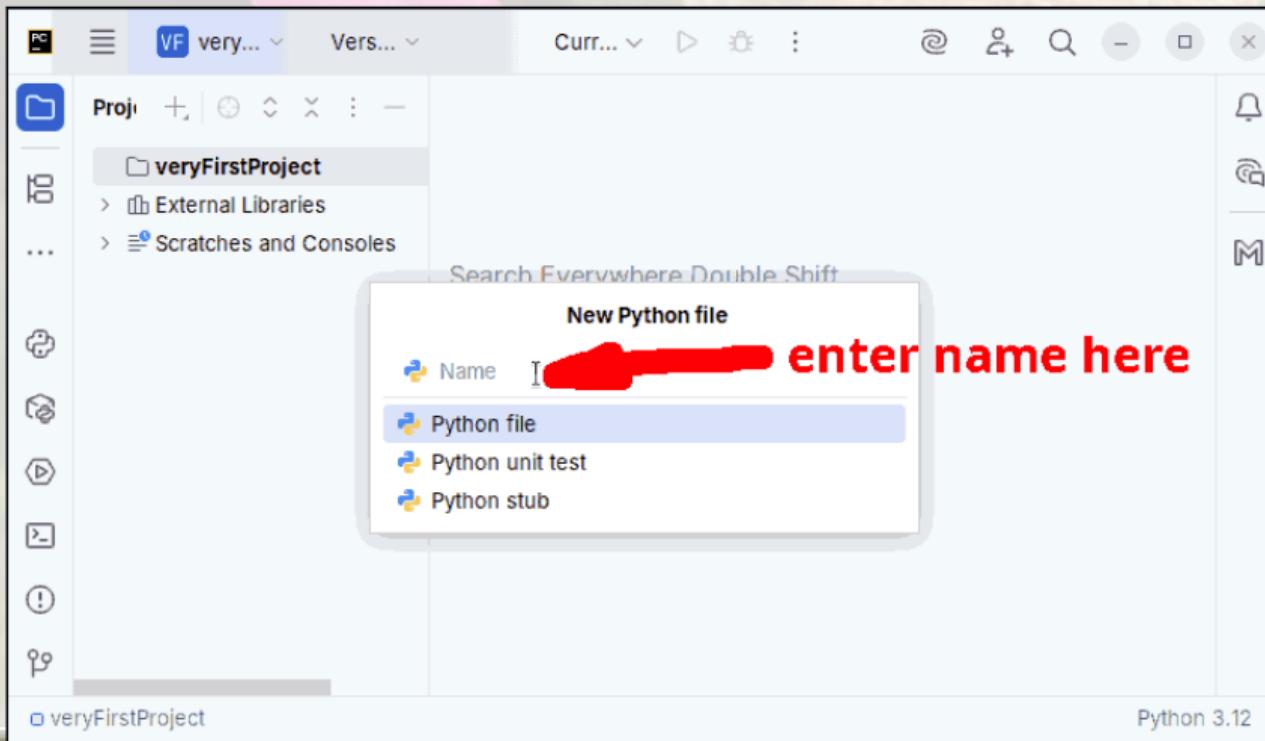
- Wir erstellen eine Python-Datei in diesem Projekt durch Rechtsklick auf den Projektordner `veryFirstProject` und dann durch auswählen von `New > Python File`.



Das Erste PyCharm Projekt und Programm



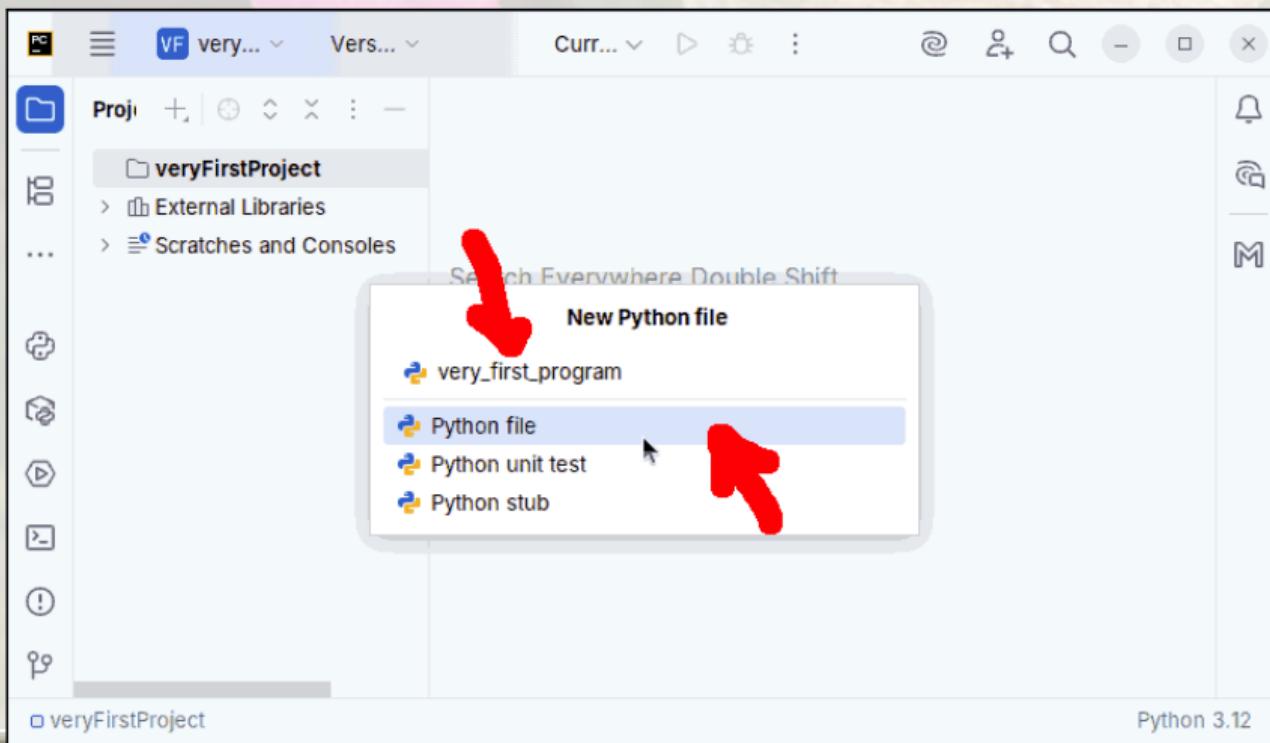
- In dem sich öffnenden Dialog können wir den Dateiname eingeben.



Das Erste PyCharm Projekt und Programm



- Wir nennen unsere Datei `very_first_program` und drücken ↴.



Das Erste PyCharm Projekt und Programm



- Die neue, leere Datei `very_first_program.py` wurde im Projektordner `veryFirstProject` erstellt.

The screenshot shows the PyCharm IDE interface. On the left is the Project tool window, which displays the structure of the 'veryFirstProject' directory. Inside the directory, there is a single file named 'very_first_program.py'. The main editor window on the right shows the content of this file, which is currently empty, indicated by a single character '1' at the top left of the code area. The status bar at the bottom provides information about the current file ('veryFirstProject > very_first_program.py'), the code editor settings ('1:1 LF UTF-8 4 spaces'), and the Python interpreter version ('Python 3.12').

Das Erste PyCharm Projekt und Programm



- Nun tippen wir das Programm `print("Hello World!")` ab.

The screenshot shows the PyCharm IDE interface. On the left is the Project tool window, which displays a project named 'veryFirstProject' containing a single file 'very_first_program.py'. The code editor on the right shows the following content:

```
1 print("Hello World!")
```

The status bar at the bottom indicates the file is 'veryFirstProject > very_first_program.py', the encoding is 'UTF-8', and the Python version is 'Python 3.12'.

Das Erste PyCharm Projekt und Programm



- Nun tippen wir das Programm `print("Hello World!")` ab.
- PyCharm speichert die Datei automatisch für uns.

A screenshot of the PyCharm IDE interface. The left sidebar shows a project named "veryFirstProject" containing a file "very_first_program.py". The code editor window displays the following Python code:

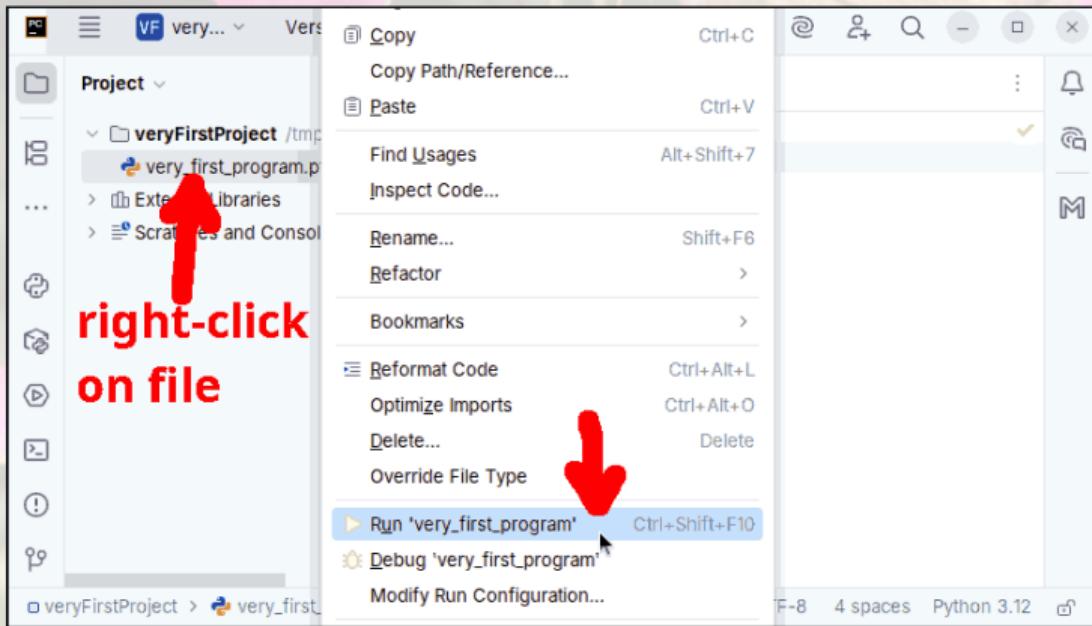
```
1 print("Hello World!")
```

The status bar at the bottom indicates the file is "veryFirstProject > very_first_program.py", and the settings are "2:1 LF UTF-8 4 spaces Python 3.12".

Das Erste PyCharm Projekt und Programm



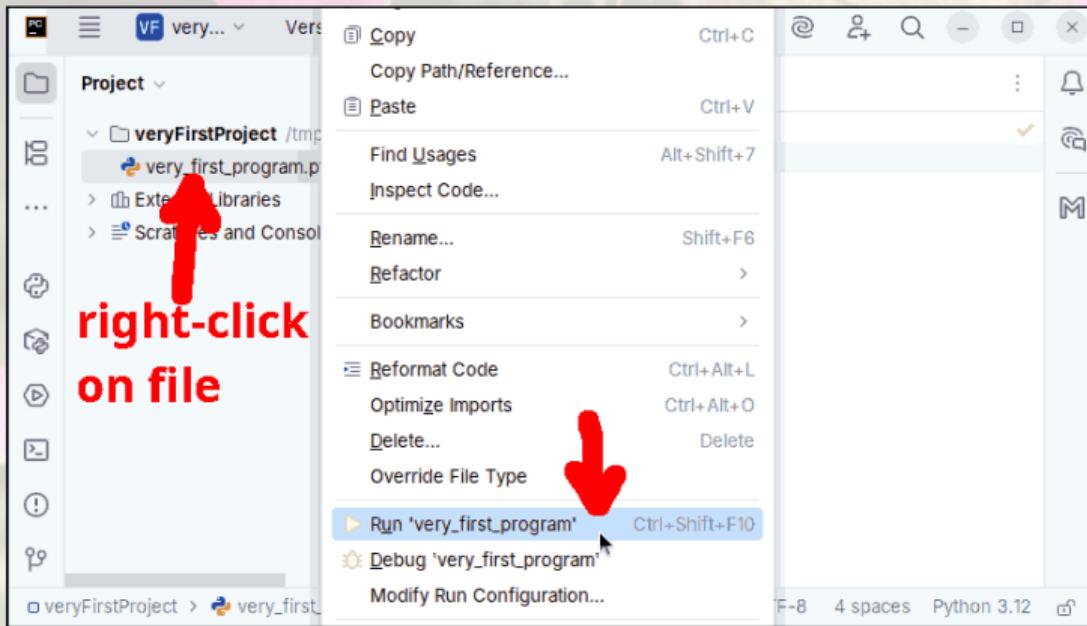
- Um das Programm auszuführen, rechtsklicken wir auf die Programmdatei und wählen Run 'very_first_program' aus.



Das Erste PyCharm Projekt und Programm



- Um das Programm auszuführen, rechtsklicken wir auf die Programmdatei und wählen Run 'very_first_program' aus.
- Alternativ könnten wir auch **Ctrl** + **↑** + **F10** drücken.



Das Erste PyCharm Projekt und Programm



- Tatsächlich: In der Konsolenfläche im PyCharm-Fenster erscheint der Text „Hello World!“.

```
Project veryFirstProject / very_first_program.py
Run very_first_program
print("Hello World!")

Process finished with exit code 0
```

The screenshot shows the PyCharm interface. On the left is the Project tool window with a folder icon, showing a project named 'veryFirstProject' containing a file 'very_first_program.py'. The main editor window displays the Python code: 'print("Hello World!")'. Below the editor is the Run tool window, which shows the command '/usr/bin/python3.12 /tmp/veryFirstProject/very_first_program.py' and the output 'Hello World!'. At the bottom, the status bar indicates the file path 'veryFirstProject > very_first_program.py', encoding 'UTF-8', and the Python version 'Python 3.12'.

Das Erste PyCharm Projekt und Programm



- Tatsächlich: In der Konsolenfläche im PyCharm-Fenster erscheint der Text „Hello World!“.
- Zusätzlich sehen wir auch, wie das Programm ausgeführt wurde, nämlich den Python-Interpreter mit dem Pfad zu unserer Datei als Parameter.

```
print("Hello World!")
```

Run

```
/usr/bin/python3.12 /tmp/veryFirstProject/very_first_program.py
Hello World!
```

Process finished with exit code 0

veryFirstProject > very_first_program.py

2:1 LF UTF-8 4 spaces Python 3.12

Das Erste PyCharm Projekt und Programm



- Wir bekommen auch „Process finished with exit code 0“ angezeigt: Unser Programm ist erfolgreich und ohne Fehler abgelaufen.

The screenshot shows the PyCharm IDE interface. On the left is the Project tool window with a 'veryFirstProject' folder containing 'very_first_program.py'. The code editor shows the following script:

```
print("Hello World!")
```

In the Run tool window below, a terminal session is displayed:

```
/usr/bin/python3.12 /tmp/veryFirstProject/very_first_program.py
Hello World!
Process finished with exit code 0
```

The status bar at the bottom indicates the file is 'veryFirstProject > very_first_program.py', and the settings are '2:1 LF' with 'UTF-8' encoding, '4 spaces' indentation, and 'Python 3.12' selected.



Program im Terminal ausführen



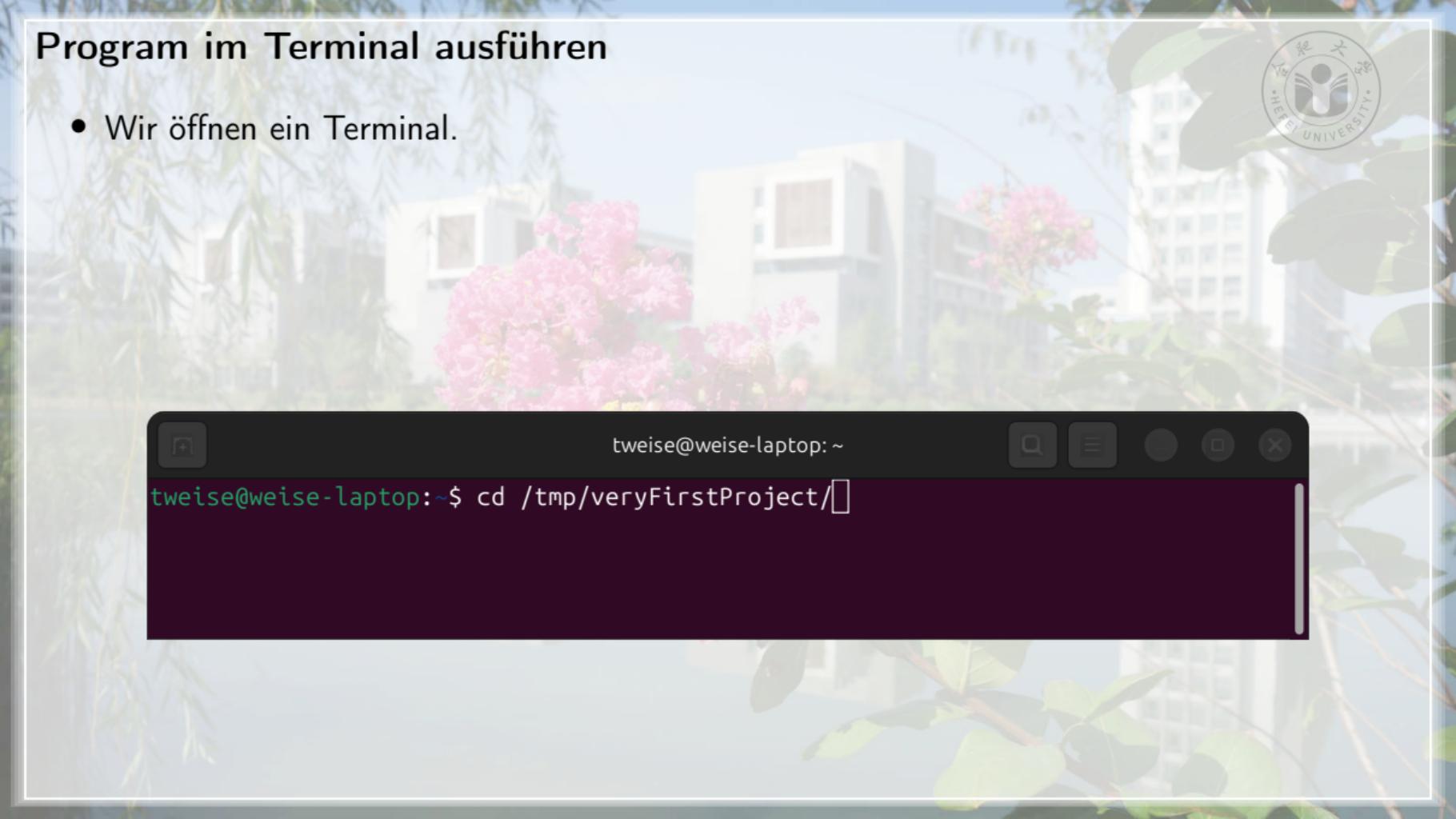
Program im Terminal ausführen

- Führen wir nun das selbe Program im normalen Terminal aus.



Program im Terminal ausführen

- Wir öffnen ein Terminal.

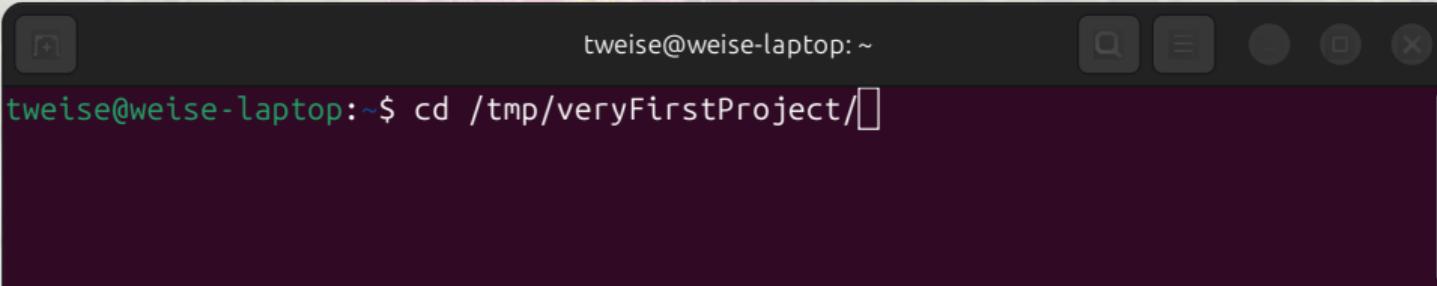
A semi-transparent background image of a modern building with large windows and greenery, including a bush with pink flowers, is visible behind the terminal window.

```
tweise@weise-laptop:~$ cd /tmp/veryFirstProject/[]
```



Program im Terminal ausführen

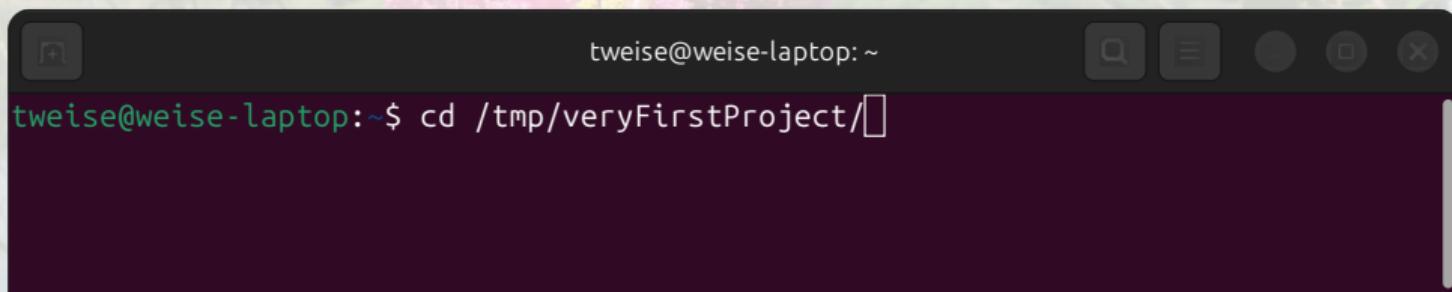
- Wir öffnen ein Terminal. (Unter Ubuntu Linux durch Drücken von **Ctrl**+**Alt**+**T**, unter Microsoft Windows durch Druck auf **Windows**+**R**, dann Schreiben von **cmd**, dann Druck auf **↵**.)



```
tweise@weise-laptop:~$ cd /tmp/veryFirstProject/
```

Program im Terminal ausführen

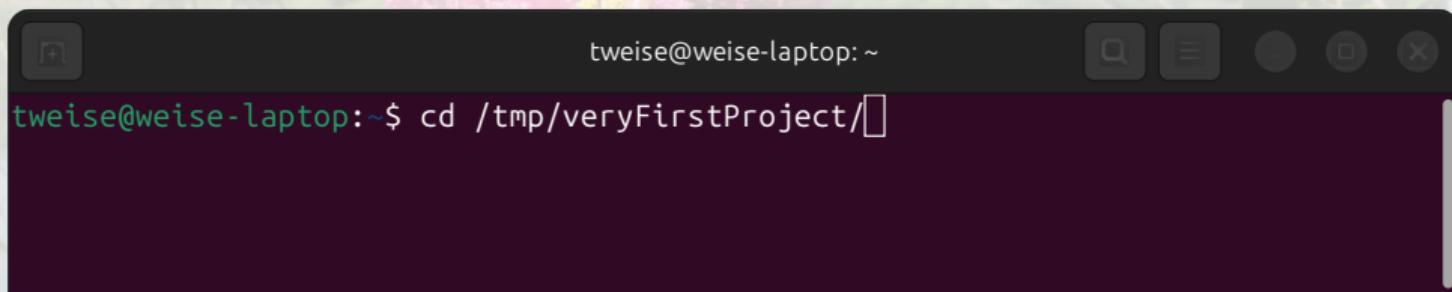
- Wir öffnen ein Terminal. (Unter Ubuntu Linux durch Drücken von **Ctrl**+**Alt**+**T**, unter Microsoft Windows durch Druck auf **Windows**+**R**, dann Schreiben von **cmd**, dann Druck auf **Enter**.)
- Wir wechseln in das Projektverzeichnis, wo sich die auszuführende Python-Datei befindet.



```
tweise@weise-laptop:~$ cd /tmp/veryFirstProject/
```

Program im Terminal ausführen

- Wir öffnen ein Terminal. (Unter Ubuntu Linux durch Drücken von **Ctrl**+**Alt**+**T**, unter Microsoft Windows durch Druck auf **Windows**+**R**, dann Schreiben von **cmd**, dann Druck auf **Enter**.)
- Wir wechseln in das Projektverzeichnis, wo sich die auszuführende Python-Datei befindet.
- Das Kommando dafür ist **cd directory** gefolgt von **Enter**.

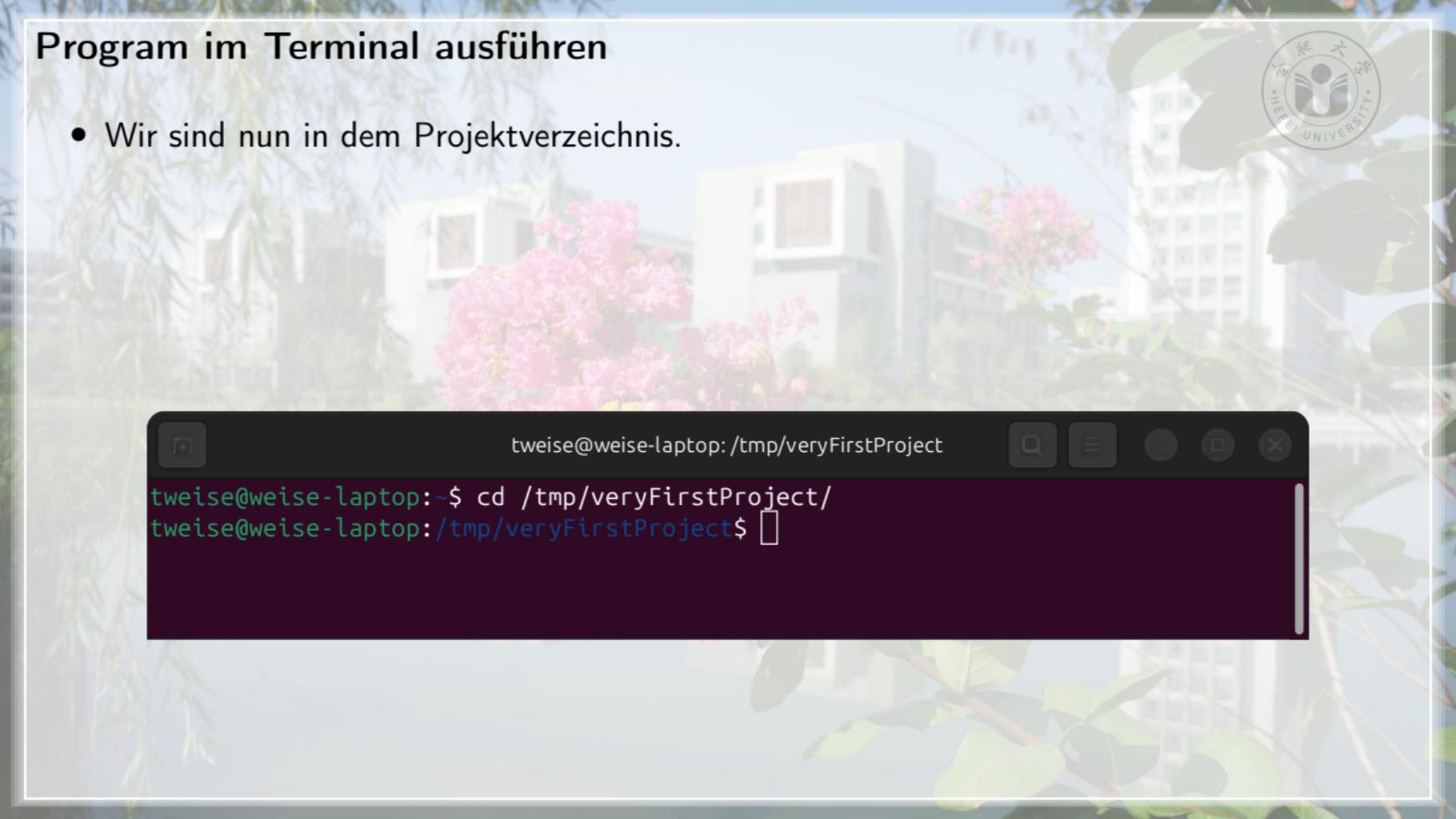


A screenshot of a terminal window titled "tweise@weise-laptop:~". The window contains the command "tweise@weise-laptop:~\$ cd /tmp/veryFirstProject/". The terminal has a dark background and light-colored text. The window title bar includes icons for search, minimize, maximize, and close.

```
tweise@weise-laptop:~$ cd /tmp/veryFirstProject/
```

Program im Terminal ausführen

- Wir sind nun in dem Projektverzeichnis.

A blurred background image of a modern building with large windows and pink flowers in the foreground.

```
tweise@weise-laptop: /tmp/veryFirstProject
tweise@weise-laptop:~$ cd /tmp/veryFirstProject/
tweise@weise-laptop:/tmp/veryFirstProject$ 
```

A terminal window showing a command-line interface. The prompt is "tweise@weise-laptop: /tmp/veryFirstProject". The user types "cd /tmp/veryFirstProject/" followed by an enter key. The prompt then changes to "tweise@weise-laptop:/tmp/veryFirstProject\$". The terminal has a dark theme with light-colored text and icons.

Program im Terminal ausführen

- Wir führen ein Programm „program.py“ mit dem Befehl `python3 program.py` (gefolgt von ↵) aus.

A semi-transparent background image of a modern building with large windows and greenery in front, featuring pink flowers.

```
tweise@weise-laptop: /tmp/veryFirstProject
tweise@weise-laptop:~$ cd /tmp/veryFirstProject/
tweise@weise-laptop:/tmp/veryFirstProject$ python3 very_first_program.py
```

A terminal window showing the command line interface. The user is in their home directory (~). They change to the directory /tmp/veryFirstProject, then run the command python3 very_first_program.py. The terminal has a dark theme with light-colored text and icons.

Program im Terminal ausführen

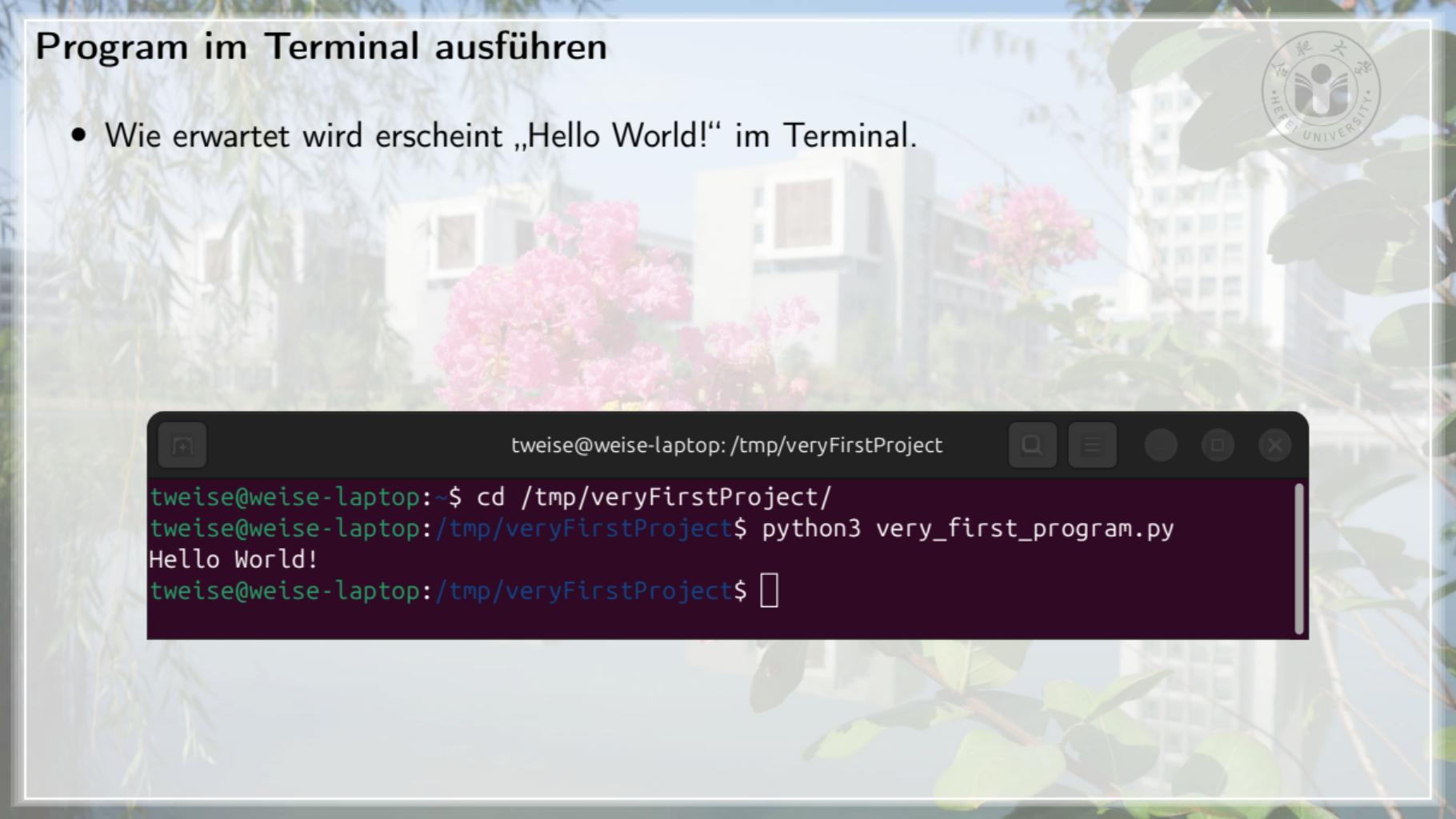
- Wir führen ein Programm „program.py“ mit dem Befehl `python3 program.py` (gefolgt von ↵) aus. In unserem Fall ist der Dateiname „very_first_program.py“.



```
tweise@weise-laptop: /tmp/veryFirstProject
tweise@weise-laptop:~$ cd /tmp/veryFirstProject/
tweise@weise-laptop:/tmp/veryFirstProject$ python3 very_first_program.py
```

Program im Terminal ausführen

- Wie erwartet wird erscheint „Hello World!“ im Terminal.

A faint background image of a modern building with large windows and pink flowers in the foreground.

```
tweise@weise-laptop:~/tmp/veryFirstProject
tweise@weise-laptop:~/tmp/veryFirstProject$ cd /tmp/veryFirstProject/
tweise@weise-laptop:/tmp/veryFirstProject$ python3 very_first_program.py
Hello World!
tweise@weise-laptop:/tmp/veryFirstProject$ 
```

Program im Terminal ausführen

- Führen wir nun das selbe Program im normalen Terminal aus.

Gute Praxis

Die einzig **richtige Art**, Python Programme im Produktiveinsatz auszuführen, ist sie im Terminal mit dem Python Interpreter als Programdatei zu starten.

Program im Terminal ausführen

- Führen wir nun das selbe Program im normalen Terminal aus.

Gute Praxis

Die einzig **richtige Art**, Python Programme im Produktiveinsatz auszuführen, ist sie im Terminal mit dem Python Interpreter als Programdatei zu starten.

- Alle anderen Arten sind vielleicht während der Entwicklung nützlich, haben aber nichts im Produktiveinsatz verloren.

Program im Terminal ausführen

- Führen wir nun das selbe Program im normalen Terminal aus.

Gute Praxis

Die einzig **richtige Art**, Python Programme im Produktiveinsatz auszuführen, ist sie im Terminal mit dem Python Interpreter als Programdatei zu starten.

- Alle anderen Arten sind vielleicht während der Entwicklung nützlich, haben aber nichts im Produktiveinsatz verloren.
- Das gilt ganz besonders für das Ausführen mit Hilfe von PyCharm.

Program im Terminal ausführen

- Führen wir nun das selbe Program im normalen Terminal aus.

Gute Praxis

Die einzig **richtige Art**, Python Programme im Produktiveinsatz auszuführen, ist sie im Terminal mit dem Python Interpreter als Programdatei zu starten.

- Alle anderen Arten sind vielleicht während der Entwicklung nützlich, haben aber nichts im Produktiveinsatz verloren.
- Das gilt ganz besonders für das Ausführen mit Hilfe von PyCharm. Machen Sie das niemals im Produktiveinsatz.



Programm in Python Interpreter in PyCharm Eingeben



Programm in Python Interpreter in PyCharm Eingeben



- Nun wollen wir ein Programm Schritt-für-Schritt in den Python-Interpreter in PyCharm eingeben und ausführen.

Programm in Python Interpreter in PyCharm Eingeben



- Wir drücken den -Button auf der vertikalen Knopfliste auf der linken Seite des PyCharm-Fensters.

```
print("Hello World!")
```

The screenshot shows the PyCharm IDE interface. On the left, there's a vertical toolbar with several icons. A red arrow points to the first icon from the top, which is the 'Run' button (represented by a play symbol). To its right, another icon is partially visible, representing the 'Python Console'. The main workspace shows a project named 'veryFirstProject' with a file 'very_first_program.py' containing the code 'print("Hello World!")'. Below the code editor is a terminal window showing the output of the run command: '/usr/bin/python3.10 /home/tweise/local/programming/python/programmingWithPythonCode/00_v... Hello World!'. At the bottom, status information includes 'Process finished with exit code 0', the file path '00_veryFirstProject > very_first_program.py', and the Python version 'Python 3.10'.

Programm in Python Interpreter in PyCharm Eingeben



- Die PyCharm Python-Interpreter-Konsole öffnet sich.

The screenshot shows the PyCharm IDE interface. On the left is the Project tool window, which displays a folder named 'veryFirstProject' containing a file named 'very_first_program.py'. The code in this file is:

```
print("Hello World!")
```

Below the Project tool window is the Python Console tool window. It shows the command line environment with the Python interpreter path and some initial imports:/usr/bin/python3.10 /snap/pycharm-community/397/plugins/pyt> Special Variables

```
import sys; print('Python %s on %s' % (sys.version, sys pla
```

```
sys.path.append(['/home/tweise/local/programming/python/pro
```

```
>>> Python Console
```

```
>>> |
```

The status bar at the bottom indicates the file is 'Python Console.py', the encoding is 'UTF-8', and the Python version is 'Python 3.10'.

Programm in Python Interpreter in PyCharm Eingeben



- Wir tippen das „Hello World!“-Programm ein, i.e., `print("Hello World!")`, und drücken ↴.

```
very_first_program.py
1 print("Hello World!")
2

Python Console
>>> /usr/bin/python3.10 /snap/pycharm-community/397/plugins/pyt
>>> import sys; print('Python %s on %s' % (sys.version, sys pla
>>> sys.path.extend(['/home/tweise/local/programming/python/pro
>>> Python Console
>>> print("Hello World!")
```

Programm in Python Interpreter in PyCharm Eingeben



- Die Ausgabe „Hello World!“ erscheint.

The screenshot shows the PyCharm IDE interface. On the left, the Project tool window displays a project named 'veryFirstProject' containing a file 'very_first_program.py'. The code in this file is:

```
print("Hello World!")
```

In the center, the Editor tool window shows the same code. Below it, the Python Console tab is active, displaying the output of running the program:

```
/usr/bin/python3.10 /snap/pycharm-community/397/plugins/pyt
> Special Variables
import sys; print('Python %s on %s' % (sys.version, sys pla
sys.path.extend(['/home/tweise/local/programming/python/pro
Python Console>>> print("Hello World!")
Hello World!
>>>
```

The status bar at the bottom indicates the file is 'Python Console.py', the encoding is 'UTF-8', and the Python version is 'Python 3.10'.



Programm in Python Interpreter in Terminal Eingeben



Programm in Python Interpreter in Terminal Eingeben



- Jetzt werden wir das Programm in den Python-Interpreter im Terminal eingeben.

Programm in Python Interpreter in Terminal Eingeben



- Wir öffnen ein Terminal.

A screenshot of a dark-themed terminal window. The title bar shows the user 'tweise@weise-laptop: ~'. The command line displays the text 'tweise@weise-laptop:~\$ python3' followed by a cursor. The window has standard OS X-style controls at the top right.

Programm in Python Interpreter in Terminal Eingeben



- Wir öffnen ein Terminal. (Unter Ubuntu Linux durch Drücken von **Ctrl** + **Alt** + **T**, unter Microsoft Windows durch Druck auf **Windows** + **R**, dann Schreiben von `cmd`, dann Druck auf **Enter**.)

A screenshot of a terminal window titled "tweise@weise-laptop: ~". The window has a dark theme with light-colored icons. In the terminal, the command `tweise@weise-laptop:~$ python3` is typed and highlighted in green. The background of the slide shows a blurred image of a traditional Chinese building with a tiled roof and trees.

```
tweise@weise-laptop:~$ python3
```

Programm in Python Interpreter in Terminal Eingeben



- Wir öffnen ein Terminal. (Unter Ubuntu Linux durch Drücken von **Ctrl** + **Alt** + **T**, unter Microsoft Windows durch Druck auf **Windows** + **R**, dann Schreiben von `cmd`, dann Druck auf **Enter**.)
- Wir geben `python3` ein und drücken **Enter**.

A screenshot of a terminal window titled "tweise@weise-laptop: ~". The window contains the command "tweise@weise-laptop:~\$ python3" in white text on a dark background. The terminal has a standard Linux-style interface with icons at the top and a scroll bar on the right.

Programm in Python Interpreter in Terminal Eingeben



- Die Python-Interpreter-Konsole öffnet sich im Terminal.

A screenshot of a terminal window titled "tweise@weise-laptop: ~". The window contains the following text:

```
tweise@weise-laptop:~$ python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Programm in Python Interpreter in Terminal Eingeben



- Wir tippen das „Hello World!“-Programm ein, i.e., `print("Hello World!")`, und drücken ↵.

The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads "tweise@weise-laptop: ~". The window contains the following text:

```
tweise@weise-laptop:~$ python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
```

Programm in Python Interpreter in Terminal Eingeben



- Die Ausgabe „Hello World!“ erscheint.

A screenshot of a terminal window titled "tweise@weise-laptop: ~". The window contains the following text:

```
tweise@weise-laptop:~$ python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
Hello World!
>>> 
```

Programm in Python Interpreter in Terminal Eingeben



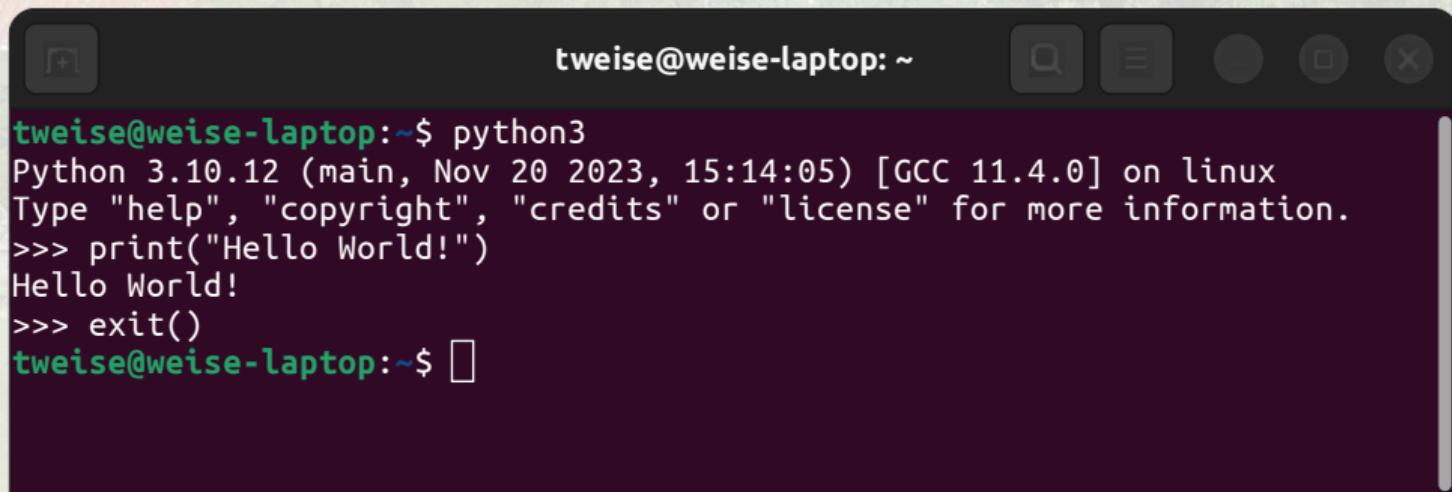
- Um den interaktiven Python-Interpreter wieder zu verlassen, tippen wir `exit()` ein und drücken .

```
tweise@weise-laptop:~$ python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
Hello World!
>>> exit()
```

Programm in Python Interpreter in Terminal Eingeben



- Wir sind zurück im normalen Terminal.



A screenshot of a terminal window titled "tweise@weise-laptop: ~". The window contains the following text:

```
tweise@weise-laptop:~$ python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
Hello World!
>>> exit()
tweise@weise-laptop:~$
```



Zusammenfassung



Zusammenfassung



- Wir haben vier Arten kennengelernt, wie wir Python-Programme ausführen können.



Zusammenfassung



- Wir haben vier Arten kennengelernt, wie wir Python-Programme ausführen können.
- Auf der einen Seite können wir Programme als Textdateien mit der Endung `.py` speichern.

Zusammenfassung



- Wir haben vier Arten kennengelernt, wie wir Python-Programme ausführen können.
- Auf der einen Seite können wir Programme als Textdateien mit der Endung `.py` speichern. Diese können wir dann entweder im Terminal oder in PyCharm ausführen.

Zusammenfassung



- Wir haben vier Arten kennengelernt, wie wir Python-Programme ausführen können.
- Auf der einen Seite können wir Programme als Textdateien mit der Endung `.py` speichern. Diese können wir dann entweder im Terminal oder in PyCharm ausführen.
- Auf der anderen Seite können wir Programme auch Zeile-für-Zeile in einer interaktiven Python-Interpreter-Session direkt in den Interpreter eintippen.

Zusammenfassung



- Wir haben vier Arten kennengelernt, wie wir Python-Programme ausführen können.
- Auf der einen Seite können wir Programme als Textdateien mit der Endung `.py` speichern. Diese können wir dann entweder im Terminal oder in PyCharm ausführen.
- Auf der anderen Seite können wir Programme auch Zeile-für-Zeile in einer interaktiven Python-Interpreter-Session direkt in den Interpreter eintippen. Auch das können wir entweder im Terminal oder in PyCharm machen.
- Natürlich werden wir unsere Programme in „richtigen“ Projekten immer in Dateien speichern.

Zusammenfassung



- Wir haben vier Arten kennengelernt, wie wir Python-Programme ausführen können.
- Auf der einen Seite können wir Programme als Textdateien mit der Endung `.py` speichern. Diese können wir dann entweder im Terminal oder in PyCharm ausführen.
- Auf der anderen Seite können wir Programme auch Zeile-für-Zeile in einer interaktiven Python-Interpreter-Session direkt in den Interpreter eintippen. Auch das können wir entweder im Terminal oder in PyCharm machen.
- Natürlich werden wir unsere Programme in „richtigen“ Projekten immer in Dateien speichern.
- Aber zum Kennenlernen von Python ist eine interaktive Nutzung des Interpreters sehr geeignet¹.



谢谢您们！
Thank you!
Vielen Dank!



References I



- [1] "An Informal Introduction to Python". In: *Python 3 Documentation. The Python Tutorial*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. URL: <https://docs.python.org/3/tutorial/introduction.html> (besucht am 2025-07-11) (siehe S. 69–74).
- [2] 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. 80, 81).
- [3] Kent L. Beck. *JUnit Pocket Guide*. Sebastopol, CA, USA: O'Reilly Media, Inc., Sep. 2004. ISBN: 978-0-596-00743-0 (siehe S. 82).
- [4] Ed Bott. *Windows 11 Inside Out*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Feb. 2023. ISBN: 978-0-13-769132-6 (siehe S. 80).
- [5] Ron Brash und Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2018. ISBN: 978-1-78862-936-2 (siehe S. 80).
- [6] Josh Centers. *Take Control of iOS 18 and iPadOS 18*. San Diego, CA, USA: Take Control Books, Dez. 2024. ISBN: 978-1-990783-55-5 (siehe S. 80).
- [7] 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. 81).
- [8] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (siehe S. 80).
- [9] 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. 81).
- [10] 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. 81).
- [11] ."exit – Terminate a Process". In: *POSIX.1-2024: The Open Group Base Specifications Issue 8, IEEE Std 1003.1™-2024 Edition*. Hrsg. von Andrew Josey. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) und San Francisco, CA, USA: The Open Group, 8. Aug. 2024. URL: <https://pubs.opengroup.org/onlinepubs/9799919799/functions/exit.html> (besucht am 2024-10-30) (siehe S. 80).

References II



- [12] Andrew Josey, Hrsg. *POSIX.1-2024: The Open Group Base Specifications Issue 8, IEEE Std 1003.1™-2024 Edition*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) und San Francisco, CA, USA: The Open Group, 8. Aug. 2024. URL: <https://pubs.opengroup.org/onlinepubs/9799919799> (besucht am 2024-10-30).
- [13] ."stderr, stdin, stdout – Standard I/O Streams". In: *POSIX.1-2024: The Open Group Base Specifications Issue 8, IEEE Std 1003.1™-2024 Edition*. Hrsg. von Andrew Josey. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE) und San Francisco, CA, USA: The Open Group, 8. Aug. 2024. URL: <https://pubs.opengroup.org/onlinepubs/9799919799/functions/stdin.html> (besucht am 2024-10-30) (siehe S. 81).
- [14] 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. 81).
- [15] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 81).
- [16] 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. 80).
- [17] A. Jefferson Offutt. "Unit Testing Versus Integration Testing". In: *Test: Faster, Better, Sooner – IEEE International Test Conference (ITC'1991)*. 26.–30. Okt. 1991, Nashville, TN, USA. Los Alamitos, CA, USA: IEEE Computer Society, 1991. Kap. Paper P2.3, S. 1108–1109. ISSN: 1089-3539. ISBN: 978-0-8186-9156-0. doi:10.1109/TEST.1991.519784 (siehe S. 82).
- [18] Michael Olan. "Unit Testing: Test Early, Test Often". *Journal of Computing Sciences in Colleges (JCSC)* 19(2):319–328, Dez. 2003. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 1937-4771. doi:10.5555/948785.948830. URL: <https://www.researchgate.net/publication/255673967> (besucht am 2025-09-05) (siehe S. 82).
- [19] Ashwin Pajankar. *Python Unit Test Automation: Automate, Organize, and Execute Unit Tests in Python*. New York, NY, USA: Apress Media, LLC, Dez. 2021. ISBN: 978-1-4842-7854-3 (siehe S. 82).
- [20] Ernest E. Rothman, Rich Rosen und Brian Jepson. *Mac OS X for Unix Geeks*. 4. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Sep. 2008. ISBN: 978-0-596-52062-5 (siehe S. 80).

References III



- [21] Per Runeson. "A Survey of Unit Testing Practices". *IEEE Software* 23(4):22–29, Juli–Aug. 2006. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 0740-7459. doi:10.1109/MS.2006.91 (siehe S. 82).
- [22] Ahmad Sahar. *iOS 26 Programming for Beginners*. 10. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Nov. 2025. ISBN: 978-1-80602-393-6 (siehe S. 82).
- [23] 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. 80).
- [24] Drew Smith. *Modern Apple Platform Administration – macOS and iOS Essentials* (2025). Birmingham, England, UK: Packt Publishing Ltd, Feb. 2025. ISBN: 978-1-80580-309-6 (siehe S. 80).
- [25] George K. Thiruvathukal, Konstantin Läufer und Benjamin Gonzalez. "Unit Testing Considered Useful". *Computing in Science & Engineering* 8(6):76–87, Nov.–Dez. 2006. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1521-9615. doi:10.1109/MCSE.2006.124. URL: <https://www.researchgate.net/publication/220094077> (besucht am 2024-10-01) (siehe S. 82).
- [26] 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. 80).
- [27] 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. 81).
- [28] Sander van Vugt. *Linux Fundamentals*. 2. Aufl. Hoboken, NJ, USA: Pearson IT Certification, Juni 2022. ISBN: 978-0-13-792931-3 (siehe S. 80).
- [29] 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. 81).
- [30] Kevin Wilson. *Python Made Easy*. Birmingham, England, UK: Packt Publishing Ltd, Aug. 2024. ISBN: 978-1-83664-615-0 (siehe S. 81).
- [31] Martin Yanev. *PyCharm Productivity and Debugging Techniques*. Birmingham, England, UK: Packt Publishing Ltd, Okt. 2022. ISBN: 978-1-83763-244-2 (siehe S. 81).

References IV



- [32] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, Juni 2017. ISBN: 978-1-78439-687-9 (siehe S. 80).



Glossary (in English) I

Bash is 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^{5,16,32}. Learn more at <https://www.gnu.org/software/bash>.

exit code When a process terminates, it can return a single integer value (the exit status code) to indicate success or failure¹¹. Per convention, an exit code of 0 means success. Any non-zero exit code indicates an error. Under Python, you can terminate the current process at any time by calling `exit` and optionally passing in the exit code that should be returned. If `exit` is not explicitly called, then the interpreter will return an exit code of 0 once the process normally terminates. If the process was terminated by an uncaught `Exception`, a non-zero exit code, usually 1, is returned.

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. On Apple systems, Xcode is often used.

iOS is the operating system that powers Apple iPhones^{6,24}. Learn more at <https://www.apple.com/ios>.

iPadOS is the operating system that powers Apple iPads⁶. Learn more at <https://www.apple.com/ipados>.

IT information technology

Linux is the leading open source operating system, i.e., a free alternative for Microsoft Windows^{2,8,23,26,28}. 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.

macOS or Mac OS is the operating system that powers Apple Mac(intosh) computers^{20,24}. Learn more at <https://www.apple.com/macos>.

Microsoft Windows is a commercial proprietary operating system⁴. 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>.



Glossary (in English) II

- PyCharm** is the convenient Python IDE that we recommend for this course^{27,30,31}. 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>.
- Python** The Python programming language^{10,14,15,29}, i.e., what you will learn about in our book²⁹. Learn more at <https://python.org>.
- stderr** The *standard error stream* is one of the three pre-defined streams of a console process (together with the standard input stream (`stdin`) and the `stdout`)¹³. It is the text stream to which the process writes information about errors and exceptions. If an uncaught `Exception` is raised in Python and the program terminates, then this information is written to standard error stream (`stderr`). If you run a program in a terminal, then the text that a process writes to its `stderr` appears in the console.
- stdin** The *standard input stream* is one of the three pre-defined streams of a console process (together with the `stdout` and the `stderr`)¹³. It is the text stream from which the process reads its input text, if any. The Python instruction `input` reads from this stream. If you run a program in a terminal, then the text that you type into the terminal while the process is running appears in this stream.
- stdout** The *standard output stream* is one of the three pre-defined streams of a console process (together with the `stdin` and the `stderr`)¹³. It is the text stream to which the process writes its normal output. The `print` instruction of Python writes text to this stream. If you run a program in a terminal, then the text that a process writes to its `stdout` appears in the console.
- terminal** A terminal is a text-based window where you can enter commands and execute them^{2,7}. 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 + , dann Schreiben von `cmd`, dann Druck auf . Under Ubuntu Linux, + + opens a terminal, which then runs a Bash shell inside.
- Ubuntu** is a variant of the open source operating system Linux^{7,9}. 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>.



Glossary (in English) III

unit test Software development is centered around creating the program code of an application, library, or otherwise useful system. A *unit test* is an *additional* code fragment that is not part of that productive code. It exists to execute (a part of) the productive code in a certain scenario (e.g., with specific parameters), to observe the behavior of that code, and to compare whether this behavior meets the specification^{3,17–19,21,25}. If not, the unit test fails. The use of unit tests is at least threefold: First, they help us to detect errors in the code. Second, program code is usually not developed only once and, from then on, used without change indefinitely. Instead, programs are often updated, improved, extended, and maintained over a long time. Unit tests can help us to detect whether such changes in the program code, maybe after years, violate the specification or, maybe, cause another, depending, module of the program to violate its specification. Third, they are part of the documentation or even specification of a program.

Xcode is offers the tools for developing, testing, and distributing applications as well as an IDE for Apple platforms such as macOS and iOS²².