



# Programming with Python

## 10. Der Datentyp bool

Thomas Weise (汤卫思)  
[tweise@hfu.edu.cn](mailto:tweise@hfu.edu.cn)

School of Artificial Intelligence and Big Data  
Hefei University  
Hefei, Anhui, China

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. Vergleiche
3. Boolesche/Logische Operatoren
4. Zusammenfassung





# Einleitung



# Einleitung



- Wir haben bereits Vergleiche von Zahlen, z. B.  $5 < 6$ , erwähnt.

# Einleitung



- Wir haben bereits Vergleiche von Zahlen, z. B. `5 < 6`, erwähnt.
- Diese können entweder `True` (Wahr) oder `False` (Falsch) als Ergebnis haben.

# Einleitung



- Wir haben bereits Vergleiche von Zahlen, z. B. `5 < 6`, erwähnt.
- Diese können entweder `True` (Wahr) oder `False` (Falsch) als Ergebnis haben.
- Diese beiden Werte formen einen weiteren grundlegenden Datentyp in Python: `bool`.

# Einleitung



- Wir haben bereits Vergleiche von Zahlen, z. B. `5 < 6`, erwähnt.
- Diese können entweder `True` (Wahr) oder `False` (Falsch) als Ergebnis haben.
- Diese beiden Werte formen einen weiteren grundlegenden Datentyp in Python: `bool`.
- Die beiden Werte dieses Datentyps sind von grundlegender Wichtigkeit wenn ein Programm Entscheidungen auf der Basis von Daten trifft.



# Vergleiche



# Vergleiche



- Als wir über die Datentypen `int` und `float` gesprochen haben, haben wir bereits Vergleiche verwendet.

# Vergleiche



- Als wir über die Datentypen `int` und `float` gesprochen haben, haben wir bereits Vergleiche verwendet.
- Python unterstützt 6 verschiedene Vergleiche.

# Vergleiche



- Als wir über die Datentypen `int` und `float` gesprochen haben, haben wir bereits Vergleiche verwendet.
- Python unterstützt 6 verschiedene Vergleiche
  1. gleich:  $a = b$  entspricht `a == b`,

# Vergleiche



- Als wir über die Datentypen `int` und `float` gesprochen haben, haben wir bereits Vergleiche verwendet.
- Python unterstützt 6 verschiedene Vergleiche
  1. gleich:  $a = b$  entspricht `a == b`,
  2. ungleich:  $a \neq b$  entspricht `a != b`,

# Vergleiche



- Als wir über die Datentypen `int` und `float` gesprochen haben, haben wir bereits Vergleiche verwendet.
- Python unterstützt 6 verschiedene Vergleiche
  1. gleich:  $a = b$  entspricht `a == b`,
  2. ungleich:  $a \neq b$  entspricht `a != b`,
  3. kleiner als:  $a < b$  entspricht `a < b`,

# Vergleiche



- Als wir über die Datentypen `int` und `float` gesprochen haben, haben wir bereits Vergleiche verwendet.
- Python unterstützt 6 verschiedene Vergleiche
  1. gleich:  $a = b$  entspricht `a == b`,
  2. ungleich:  $a \neq b$  entspricht `a != b`,
  3. kleiner als:  $a < b$  entspricht `a < b`,
  4. kleiner oder gleich:  $a \leq b$  entspricht `a <= b`,

# Vergleiche



- Als wir über die Datentypen `int` und `float` gesprochen haben, haben wir bereits Vergleiche verwendet.
- Python unterstützt 6 verschiedene Vergleiche
  1. gleich:  $a = b$  entspricht `a == b`,
  2. ungleich:  $a \neq b$  entspricht `a != b`,
  3. kleiner als:  $a < b$  entspricht `a < b`,
  4. kleiner oder gleich:  $a \leq b$  entspricht `a <= b`,
  5. größer als:  $a > b$  entspricht `a > b`, und

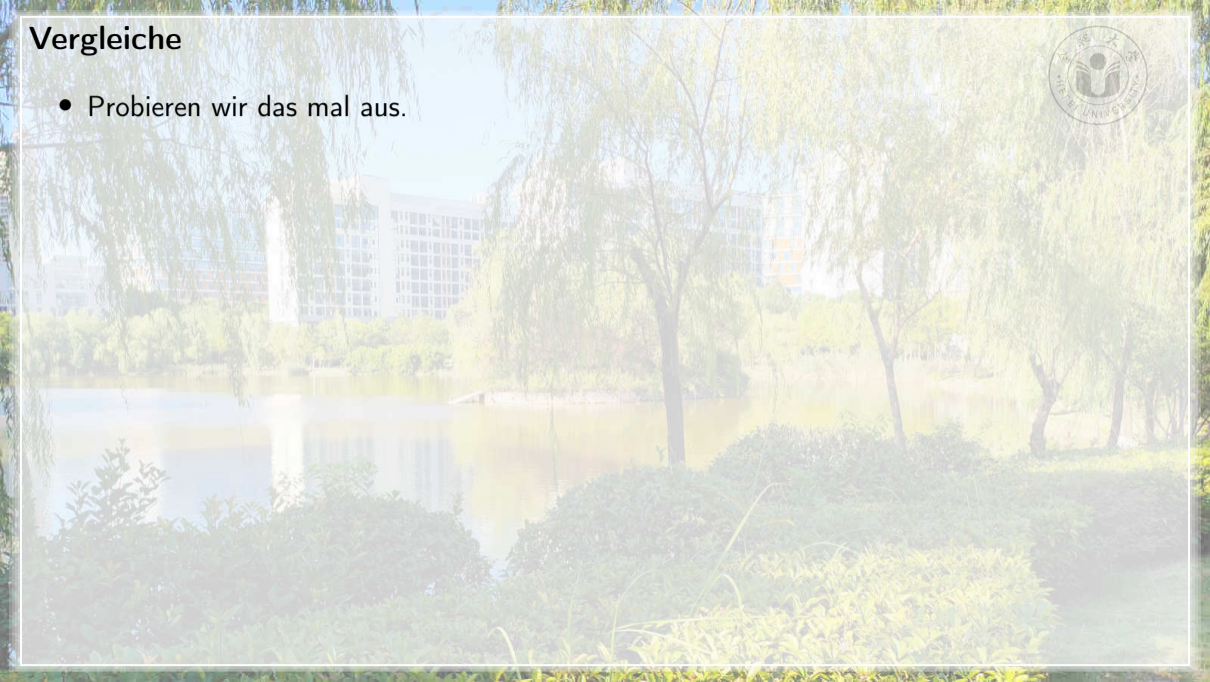
# Vergleiche



- Als wir über die Datentypen `int` und `float` gesprochen haben, haben wir bereits Vergleiche verwendet.
- Python unterstützt 6 verschiedene Vergleiche
  1. gleich:  $a = b$  entspricht `a == b`,
  2. ungleich:  $a \neq b$  entspricht `a != b`,
  3. kleiner als:  $a < b$  entspricht `a < b`,
  4. kleiner oder gleich:  $a \leq b$  entspricht `a <= b`,
  5. größer als:  $a > b$  entspricht `a > b`, und
  6. größer oder gleich:  $a \geq b$  entspricht `a >= b`.

# Vergleiche

- Probieren wir das mal aus.



# Vergleiche



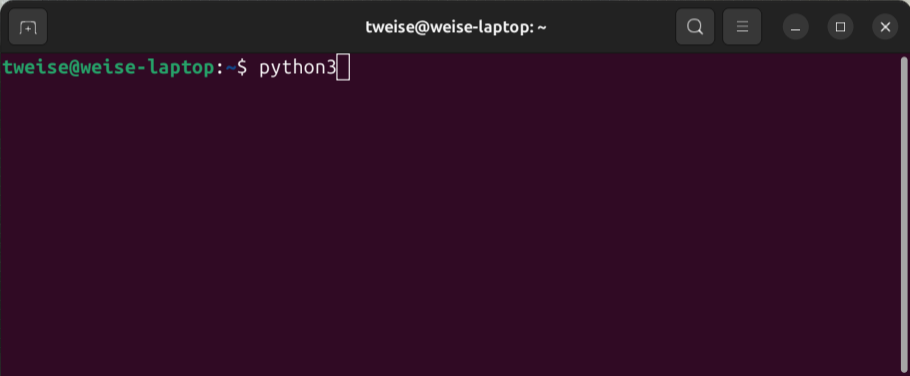
- Wir öffnen ein Terminal (Unter Ubuntu Linux durch Drücken von `Ctrl` + `Alt` + `T`, unter Microsoft Windows durch Druck auf `Windows` + `R`, dann Schreiben von `cmd`, dann Druck auf `↵`.)



# Vergleiche



- Wir schreiben `python3` und drücken .



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

A terminal window with a dark background. The title bar shows 'tweise@weise-laptop: ~'. The terminal content shows the prompt 'tweise@weise-laptop:~\$' followed by the command 'python3' and a cursor. The window has standard Linux window controls (minimize, maximize, close) and a search icon.

# Vergleiche

- Der Python-Interpreter startet.



```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> █
```

# Vergleiche

- Wir testen, ob  $6 > 6$  gilt.



```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 6 > 6
```

# Vergleiche

- Wir testen, ob  $6 > 6$  gilt. Nein, tut es nicht.



```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 6 > 6  
False  
>>> 
```

# Vergleiche

- Wir testen, ob  $6 \geq 6$  gilt.



```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 6 > 6  
False  
>>> 6 >= 6
```

# Vergleiche

- Wir testen, ob  $6 \geq 6$  gilt. Ja, tut es.



```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 6 > 6  
False  
>>> 6 >= 6  
True  
>>> 
```

# Vergleiche

- Wir testen, ob  $6 = 6$  gilt.



```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 6 > 6  
False  
>>> 6 >= 6  
True  
>>> 6 == 6
```

# Vergleiche

- Wir testen, ob  $6 = 6$  gilt. Ja, tut es.



```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 6 > 6  
False  
>>> 6 >= 6  
True  
>>> 6 == 6  
True  
>>> 
```

# Vergleiche

- Wir testen, ob  $6 \leq 6$  gilt.



```
tweise@weise-laptop: ~  
twaise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 6 > 6  
False  
>>> 6 >= 6  
True  
>>> 6 == 6  
True  
>>> 6 <= 6
```

# Vergleiche

- Wir testen, ob  $6 \leq 6$  gilt. Ja, tut es.



```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 6 > 6  
False  
>>> 6 >= 6  
True  
>>> 6 == 6  
True  
>>> 6 <= 6  
True  
>>> 
```

# Vergleiche

- Wir testen, ob  $6 < 6$  gilt.



```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 6 > 6  
False  
>>> 6 >= 6  
True  
>>> 6 == 6  
True  
>>> 6 <= 6  
True  
>>> 6 < 6
```

# Vergleiche

- Wir testen, ob  $6 < 6$  gilt. Nein, tut es nicht.



```
tweise@weise-laptop: ~  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 6 > 6  
False  
>>> 6 >= 6  
True  
>>> 6 == 6  
True  
>>> 6 <= 6  
True  
>>> 6 < 6  
False  
>>> 
```

# Vergleiche

- Wir testen, ob  $6 \neq 6$  gilt.



```
tweise@weise-laptop: ~  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 6 > 6  
False  
>>> 6 >= 6  
True  
>>> 6 == 6  
True  
>>> 6 <= 6  
True  
>>> 6 < 6  
False  
>>> 6 != 6
```

# Vergleiche

- Wir testen, ob  $6 \neq 6$  gilt. Nein, tut es nicht.



```
tweise@weise-laptop: ~  
>>> 6 > 6  
False  
>>> 6 >= 6  
True  
>>> 6 == 6  
True  
>>> 6 <= 6  
True  
>>> 6 < 6  
False  
>>> 6 != 6  
False  
>>> 
```

# Vergleiche

- Wir testen, ob  $5 > 6$  gilt.



```
tweise@weise-laptop: ~  
>>> 6 > 6  
False  
>>> 6 >= 6  
True  
>>> 6 == 6  
True  
>>> 6 <= 6  
True  
>>> 6 < 6  
False  
>>> 6 != 6  
False  
>>> 5 > 6
```

# Vergleiche

- Wir testen, ob  $5 > 6$  gilt. Nein, tut es nicht.



```
tweise@weise-laptop: ~  
>>> 6 >= 6  
True  
>>> 6 == 6  
True  
>>> 6 <= 6  
True  
>>> 6 < 6  
False  
>>> 6 != 6  
False  
>>> 5 > 6  
False  
>>> 
```

# Vergleiche

- Wir testen, ob  $5 \geq 6$  gilt.



```
tweise@weise-laptop: ~  
>>> 6 >= 6  
True  
>>> 6 == 6  
True  
>>> 6 <= 6  
True  
>>> 6 < 6  
False  
>>> 6 != 6  
False  
>>> 5 > 6  
False  
>>> 5 >= 6
```

# Vergleiche

- Wir testen, ob  $5 \geq 6$  gilt. Nein, tut es nicht.



```
tweise@weise-laptop: ~  
>>> 6 == 6  
True  
>>> 6 <= 6  
True  
>>> 6 < 6  
False  
>>> 6 != 6  
False  
>>> 5 > 6  
False  
>>> 5 >= 6  
False  
>>> 
```

# Vergleiche

- Wir testen, ob  $5 = 6$  gilt.



```
tweise@weise-laptop: ~  
>>> 6 == 6  
True  
>>> 6 <= 6  
True  
>>> 6 < 6  
False  
>>> 6 != 6  
False  
>>> 5 > 6  
False  
>>> 5 >= 6  
False  
>>> 5 == 6
```

# Vergleiche

- Wir testen, ob  $5 = 6$  gilt. Nein, tut es nicht.



```
tweise@weise-laptop: ~  
>>> 6 <= 6  
True  
>>> 6 < 6  
False  
>>> 6 != 6  
False  
>>> 5 > 6  
False  
>>> 5 >= 6  
False  
>>> 5 == 6  
False  
>>> 
```

# Vergleiche

- Wir testen, ob  $5 \leq 6$  gilt.



```
tweise@weise-laptop: ~  
>>> 6 <= 6  
True  
>>> 6 < 6  
False  
>>> 6 != 6  
False  
>>> 5 > 6  
False  
>>> 5 >= 6  
False  
>>> 5 == 6  
False  
>>> 5 <= 6
```

# Vergleiche

- Wir testen, ob  $5 \leq 6$  gilt. Ja, tut es.



```
tweise@weise-laptop: ~  
>>> 6 < 6  
False  
>>> 6 != 6  
False  
>>> 5 > 6  
False  
>>> 5 >= 6  
False  
>>> 5 == 6  
False  
>>> 5 <= 6  
True  
>>> 
```

# Vergleiche

- Wir testen, ob  $5 < 6$  gilt.



```
tweise@weise-laptop: ~  
>>> 6 < 6  
False  
>>> 6 != 6  
False  
>>> 5 > 6  
False  
>>> 5 >= 6  
False  
>>> 5 == 6  
False  
>>> 5 <= 6  
True  
>>> 5 < 6
```

# Vergleiche

- Wir testen, ob  $5 < 6$  gilt. Ja, tut es.



```
tweise@weise-laptop: ~  
>>> 6 != 6  
False  
>>> 5 > 6  
False  
>>> 5 >= 6  
False  
>>> 5 == 6  
False  
>>> 5 <= 6  
True  
>>> 5 < 6  
True  
>>> 
```

# Vergleiche

- Wir testen, ob  $5 \neq 6$  gilt.



```
tweise@weise-laptop: ~  
>>> 6 != 6  
False  
>>> 5 > 6  
False  
>>> 5 >= 6  
False  
>>> 5 == 6  
False  
>>> 5 <= 6  
True  
>>> 5 < 6  
True  
>>> 5 != 6
```

# Vergleiche

- Wir testen, ob  $5 \neq 6$  gilt. Ja, tut es.



```
tweise@weise-laptop: ~  
>>> 5 > 6  
False  
>>> 5 >= 6  
False  
>>> 5 == 6  
False  
>>> 5 <= 6  
True  
>>> 5 < 6  
True  
>>> 5 != 6  
True  
>>> 
```

# Vergleiche

- Wir testen, ob  $6 > 5$  gilt.



```
tweise@weise-laptop: ~  
>>> 5 > 6  
False  
>>> 5 >= 6  
False  
>>> 5 == 6  
False  
>>> 5 <= 6  
True  
>>> 5 < 6  
True  
>>> 5 != 6  
True  
>>> 6 > 5
```

# Vergleiche

- Wir testen, ob  $6 > 5$  gilt. Ja, tut es.



```
tweise@weise-laptop: ~  
>>> 5 >= 6  
False  
>>> 5 == 6  
False  
>>> 5 <= 6  
True  
>>> 5 < 6  
True  
>>> 5 != 6  
True  
>>> 6 > 5  
True  
>>> 
```

# Vergleiche



```
tweise@weise-laptop: ~  
>>> 5 >= 6  
False  
>>> 5 == 6  
False  
>>> 5 <= 6  
True  
>>> 5 < 6  
True  
>>> 5 != 6  
True  
>>> 6 > 5  
True  
>>> 6 >= 5
```

# Vergleiche



- Wir testen, ob  $6 \geq 5$  gilt. Ja, tut es.

```
tweise@weise-laptop: ~  
>>> 5 == 6  
False  
>>> 5 <= 6  
True  
>>> 5 < 6  
True  
>>> 5 != 6  
True  
>>> 6 > 5  
True  
>>> 6 >= 5  
True  
>>> 
```

# Vergleiche

- Wir testen, ob  $6 == 5$  gilt.



```
tweise@weise-laptop: ~  
>>> 5 == 6  
False  
>>> 5 <= 6  
True  
>>> 5 < 6  
True  
>>> 5 != 6  
True  
>>> 6 > 5  
True  
>>> 6 >= 5  
True  
>>> 6 == 5
```

# Vergleiche

- Wir testen, ob  $6 == 5$  gilt. Nein, tut es nicht.



```
tweise@weise-laptop: ~  
>>> 5 <= 6  
True  
>>> 5 < 6  
True  
>>> 5 != 6  
True  
>>> 6 > 5  
True  
>>> 6 >= 5  
True  
>>> 6 == 5  
False  
>>> 
```

# Vergleiche

- Wir testen, ob  $6 \leq 5$  gilt.



```
tweise@weise-laptop: ~  
>>> 5 <= 6  
True  
>>> 5 < 6  
True  
>>> 5 != 6  
True  
>>> 6 > 5  
True  
>>> 6 >= 5  
True  
>>> 6 == 5  
False  
>>> 6 <= 5
```

# Vergleiche

- Wir testen, ob  $6 \leq 5$  gilt. Nein, tut es nicht.



```
tweise@weise-laptop: ~  
>>> 5 < 6  
True  
>>> 5 != 6  
True  
>>> 6 > 5  
True  
>>> 6 >= 5  
True  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 
```

# Vergleiche

- Wir testen, ob  $6 < 5$  gilt.



```
tweise@weise-laptop: ~  
>>> 5 < 6  
True  
>>> 5 != 6  
True  
>>> 6 > 5  
True  
>>> 6 >= 5  
True  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5
```

# Vergleiche

- Wir testen, ob  $6 < 5$  gilt. Nein, tut es nicht.



```
tweise@weise-laptop: ~  
>>> 5 != 6  
True  
>>> 6 > 5  
True  
>>> 6 >= 5  
True  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 
```

# Vergleiche

- Wir testen, ob  $6 \neq 5$  gilt.



```
tweise@weise-laptop: ~  
>>> 5 != 6  
True  
>>> 6 > 5  
True  
>>> 6 >= 5  
True  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5
```

# Vergleiche



- Wir testen, ob  $6 \neq 5$  gilt. Ja, tut es.

```
tweise@weise-laptop: ~  
>>> 6 > 5  
True  
>>> 6 >= 5  
True  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 
```

# Vergleiche



- Wir können auch `floats` miteinander und mit `ints` vergleichen.

```
tweise@weise-laptop: ~  
>>> 6 > 5  
True  
>>> 6 >= 5  
True  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5
```

# Vergleiche



- Wir können auch `floats` miteinander und mit `ints` vergleichen. Dabei werden `floats` ohne Nachkommastelle mit `ints` betrachtet.

```
tweise@weise-laptop: ~  
>>> 6 > 5  
True  
>>> 6 >= 5  
True  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5
```

# Vergleiche

- Wir vergleichen 5.5 mit 5.



```
tweise@weise-laptop: ~  
>>> 6 > 5  
True  
>>> 6 >= 5  
True  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5
```

# Vergleiche



- Wir vergleichen 5.5 mit 5. Es ist natürlich nicht das selbe.

```
tweise@weise-laptop: ~  
>>> 6 >= 5  
True  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5  
False  
>>> 
```

# Vergleiche

- Wir testen, ob  $5.0 == 5$  gilt.



```
tweise@weise-laptop: ~  
>>> 6 >= 5  
True  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5  
False  
>>> 5.0 == 5
```

# Vergleiche

- Wir testen, ob  $5.0 == 5$  gilt. Ja, tut es.



```
tweise@weise-laptop: ~  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5  
False  
>>> 5.0 == 5  
True  
>>> 
```

# Vergleiche



- Wir können Vergleiche auch verketteten (chainen).

```
tweise@weise-laptop: ~  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5  
False  
>>> 5.0 == 5  
True  
>>> 3 < 4 < 5 < 6
```

# Vergleiche



- Wir können Vergleiche auch verketteten (chainen). Dabei ist das Ergebnis nur dann `True`, wenn alle Teilvergleiche auch `True` sind.

```
tweise@weise-laptop: ~  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5  
False  
>>> 5.0 == 5  
True  
>>> 3 < 4 < 5 < 6
```

# Vergleiche



- Wir testen, ob  $3 < 4 < 5 < 6$  stimmt.

```
tweise@weise-laptop: ~  
>>> 6 == 5  
False  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5  
False  
>>> 5.0 == 5  
True  
>>> 3 < 4 < 5 < 6
```

# Vergleiche



- Wir testen, ob  $3 < 4 < 5 < 6$  stimmt. Tut es:  $3 < 4$  stimmt,  $4 < 5$  stimmt, und  $5 < 6$  stimmt.

```
tweise@weise-laptop: ~  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5  
False  
>>> 5.0 == 5  
True  
>>> 3 < 4 < 5 < 6  
True  
>>> 
```

# Vergleiche



- Wir testen, ob  $5 \geq 4 > 4 \geq 3$  gilt.

```
tweise@weise-laptop: ~  
>>> 6 <= 5  
False  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5  
False  
>>> 5.0 == 5  
True  
>>> 3 < 4 < 5 < 6  
True  
>>> 5 >= 4 > 4 >= 3
```

# Vergleiche



- Wir testen, ob  $5 \geq 4 > 4 \geq 3$  gilt. Nein, tut es nicht. Es stimmt zwar, dass  $5 \geq 4$  und  $4 \geq 3$ , aber  $4 > 4$  stimmt nicht, weshalb der ganze Vergleich `False` ergibt.

```
tweise@weise-laptop: ~  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5  
False  
>>> 5.0 == 5  
True  
>>> 3 < 4 < 5 < 6  
True  
>>> 5 >= 4 > 4 >= 3  
False  
>>> 
```

# Vergleiche



- Die Funktion `type(x)` liefert uns den Datentyp von `x`.

```
tweise@weise-laptop: ~  
>>> 6 < 5  
False  
>>> 6 != 5  
True  
>>> 5.5 == 5  
False  
>>> 5.0 == 5  
True  
>>> 3 < 4 < 5 < 6  
True  
>>> 5 >= 4 > 4 >= 3  
False  
>>> type(True)
```

# Vergleiche



- Die Funktion `type(x)` liefert uns den Datentyp von `x`. `type(True)` ergibt daher `bool`, was als `<class 'bool'>` ausgegeben wird.

```
tweise@weise-laptop: ~  
>>> 6 != 5  
True  
>>> 5.5 == 5  
False  
>>> 5.0 == 5  
True  
>>> 3 < 4 < 5 < 6  
True  
>>> 5 >= 4 > 4 >= 3  
False  
>>> type(True)  
<class 'bool'>  
>>> 
```

# Vergleiche



- Das Ergebnis von `5 == 5` ist `True`.

```
tweise@weise-laptop: ~  
>>> 6 != 5  
True  
>>> 5.5 == 5  
False  
>>> 5.0 == 5  
True  
>>> 3 < 4 < 5 < 6  
True  
>>> 5 >= 4 > 4 >= 3  
False  
>>> type(True)  
<class 'bool'  
>>> type(5 == 5)
```

# Vergleiche

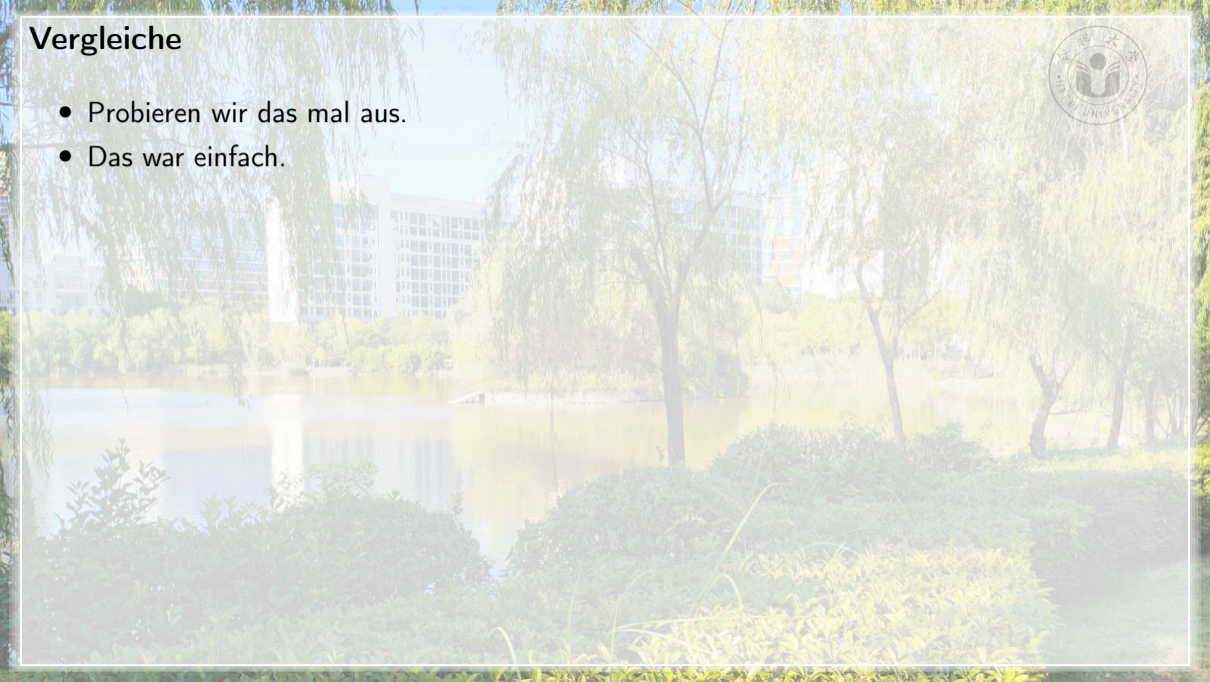


- Das Ergebnis von `5 == 5` ist `True`. `type(5 == 5)` ist daher `type(True)` und ergibt wieder `bool`.

```
tweise@weise-laptop: ~  
>>> 5.5 == 5  
False  
>>> 5.0 == 5  
True  
>>> 3 < 4 < 5 < 6  
True  
>>> 5 >= 4 > 4 >= 3  
False  
>>> type(True)  
<class 'bool'>  
>>> type(5 == 5)  
<class 'bool'>  
>>> 
```

# Vergleiche

- Probieren wir das mal aus.
- Das war einfach.





# Boolesche/Logische Operatoren



# Boolesche/Logische Operatoren



- Die wichtigsten Operationen die wir mit Booleschen Werten machen können sind die bekannten Booleschen Operatoren **and** (und), **or** (oder), und **not** (nicht).

# Boolesche/Logische Operatoren



- Die wichtigsten Operationen die wir mit Booleschen Werten machen können sind die bekannten Booleschen Operatoren `and` (und), `or` (oder), und `not` (nicht).
- Eine Konjunktion, also `and`, ist `True` dann und nur dann wenn beide Operanden auch `True` sind. Andernfalls ist das Ergebnis `False`.

a	b	a and b
False	False	False
False	True	False
True	False	False
True	True	True

# Boolesche/Logische Operatoren



- Die wichtigsten Operationen die wir mit Booleschen Werten machen können sind die bekannten Booleschen Operatoren `and` (und), `or` (oder), und `not` (nicht).
- Eine Konjunktion, also `and`, ist `True` dann und nur dann wenn beide Operanden auch `True` sind. Andernfalls ist das Ergebnis `False`.
- Eine Disjunktion, also `or`, ist `True` wenn wenigstens einer der beiden Operanden `True` ist. Andernfalls ist das Ergebnis `False`.

a	b	a or b
False	False	False
False	True	True
True	False	True
True	True	True

# Boolesche/Logische Operatoren



- Die wichtigsten Operationen die wir mit Booleschen Werten machen können sind die bekannten Booleschen Operatoren `and` (und), `or` (oder), und `not` (nicht).
- Eine Konjunktion, also `and`, ist `True` dann und nur dann wenn beide Operanden auch `True` sind. Andernfalls ist das Ergebnis `False`.
- Eine Disjunktion, also `or`, ist `True` wenn wenigstens einer der beiden Operanden `True` ist. Andernfalls ist das Ergebnis `False`.
- Eine Negation, also `not`, ist `True` wenn ihr Operant `False` ist. Andernfalls ist das Ergebnis `False`.

a	<code>not a</code>
<code>False</code>	<code>True</code>
<code>True</code>	<code>False</code>

# Boolesche/Logische Operatoren ausprobieren



- Probieren wir das mal aus.

```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> █
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `False and False`?

```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> False and False
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `False and False`? Das ergibt `False`.

```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> False and False  
False  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `False and True`?

```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> False and False  
False  
>>> False and True
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `False and True`? Das ergibt `False`.

```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> False and False  
False  
>>> False and True  
False  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `True and False`?

```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> False and False  
False  
>>> False and True  
False  
>>> True and False
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `True and False`? Das ergibt `False`.

```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> False and False  
False  
>>> False and True  
False  
>>> True and False  
False  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `True and True`? Das ergibt `True`.

```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> False and False  
False  
>>> False and True  
False  
>>> True and False  
False  
>>> True and True
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `True and True`? Das ergibt `True`.

```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> False and False  
False  
>>> False and True  
False  
>>> True and False  
False  
>>> True and True  
True  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `False or False`?

```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> False and False  
False  
>>> False and True  
False  
>>> True and False  
False  
>>> True and True  
True  
>>> False or False
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `False or False`? Das ergibt `False`.

```
tweise@weise-laptop: ~  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> False and False  
False  
>>> False and True  
False  
>>> True and False  
False  
>>> True and True  
True  
>>> False or False  
False  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `False or True`?

```
tweise@weise-laptop: ~  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> False and False  
False  
>>> False and True  
False  
>>> True and False  
False  
>>> True and True  
True  
>>> False or False  
False  
>>> False or True
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `False or True`? Das ergibt `True`.

```
tweise@weise-laptop: ~  
>>> False and False  
False  
>>> False and True  
False  
>>> True and False  
False  
>>> True and True  
True  
>>> False or False  
False  
>>> False or True  
True  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `True or False`?

```
tweise@weise-laptop: ~  
>>> False and False  
False  
>>> False and True  
False  
>>> True and False  
False  
>>> True and True  
True  
>>> False or False  
False  
>>> False or True  
True  
>>> True or False
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `True or False`? Das ergibt `True`.

```
tweise@weise-laptop: ~  
>>> False and True  
False  
>>> True and False  
False  
>>> True and True  
True  
>>> False or False  
False  
>>> False or True  
True  
>>> True or False  
True  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `True or True`?

```
tweise@weise-laptop: ~  
>>> False and True  
False  
>>> True and False  
False  
>>> True and True  
True  
>>> False or False  
False  
>>> False or True  
True  
>>> True or False  
True  
>>> True or True
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `True or True`? Das ergibt `True`.

```
tweise@weise-laptop: ~  
>>> True and False  
False  
>>> True and True  
True  
>>> False or False  
False  
>>> False or True  
True  
>>> True or False  
True  
>>> True or True  
True  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `not True`?

```
tweise@weise-laptop: ~  
>>> True and False  
False  
>>> True and True  
True  
>>> False or False  
False  
>>> False or True  
True  
>>> True or False  
True  
>>> True or True  
True  
>>> not True
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `not True`? Das ergibt `False`.

```
tweise@weise-laptop: ~  
>>> True and True  
True  
>>> False or False  
False  
>>> False or True  
True  
>>> True or False  
True  
>>> True or True  
True  
>>> not True  
False  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `not False`?

```
tweise@weise-laptop: ~  
>>> True and True  
True  
>>> False or False  
False  
>>> False or True  
True  
>>> True or False  
True  
>>> True or True  
True  
>>> not True  
False  
>>> not False
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `not False`? Das ergibt `True`.

```
tweise@weise-laptop: ~  
>>> False or False  
False  
>>> False or True  
True  
>>> True or False  
True  
>>> True or True  
True  
>>> not True  
False  
>>> not False  
True  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `(True or False) and ((False or True) or (False and False))`?

```
tweise@weise-laptop: ~  
>>> False or False  
False  
>>> False or True  
True  
>>> True or False  
True  
>>> True or True  
True  
>>> not True  
False  
>>> not False  
True  
>>> (True or False) and ((False or True) or (False and False))
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `(True or False) and ((False or True) or (False and False))`? Das ergibt `True`.

```
tweise@weise-laptop: ~  
>>> False or True  
True  
>>> True or False  
True  
>>> True or True  
True  
>>> not True  
False  
>>> not False  
True  
>>> (True or False) and ((False or True) or (False and False))  
True  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `(True or False) and ((False or True) or (False and False))`? Das ergibt `True`. `(True or False)` ist `True`, `(False or True)` auch, und `(False and False)` ist `False`.

```
tweise@weise-laptop: ~  
>>> False or True  
True  
>>> True or False  
True  
>>> True or True  
True  
>>> not True  
False  
>>> not False  
True  
>>> (True or False) and ((False or True) or (False and False))  
True  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `(True or False) and ((False or True) or (False and False))`? Das ergibt `True`. `(True or False)` ist `True`, `(False or True)` auch, und `(False and False)` ist `False`. Also haben wir `True and (True or False)`.

```
tweise@weise-laptop: ~  
>>> False or True  
True  
>>> True or False  
True  
>>> True or True  
True  
>>> not True  
False  
>>> not False  
True  
>>> (True or False) and ((False or True) or (False and False))  
True  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `(True or False) and ((False or True) or (False and False))`? Das ergibt `True`. `(True or False)` ist `True`, `(False or True)` auch, und `(False and False)` ist `False`. Also haben wir `True and (True or False)`. Also haben wir `True and True`.

```
tweise@weise-laptop: ~  
>>> False or True  
True  
>>> True or False  
True  
>>> True or True  
True  
>>> not True  
False  
>>> not False  
True  
>>> (True or False) and ((False or True) or (False and False))  
True  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `(5 < 4) or (6 < 9 < 8)`?

```
tweise@weise-laptop: ~  
>>> False or True  
True  
>>> True or False  
True  
>>> True or True  
True  
>>> not True  
False  
>>> not False  
True  
>>> (True or False) and ((False or True) or (False and False))  
True  
>>> (5 < 4) or (6 < 9 < 8)
```

# Boolesche/Logische Operatoren ausprobieren



- Was ergibt `(5 < 4) or (6 < 9 < 8)`? Das ergibt `False`.

```
tweise@weise-laptop: ~  
>>> True or False  
True  
>>> True or True  
True  
>>> not True  
False  
>>> not False  
True  
>>> (True or False) and ((False or True) or (False and False))  
True  
>>> (5 < 4) or (6 < 9 < 8)  
False  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren

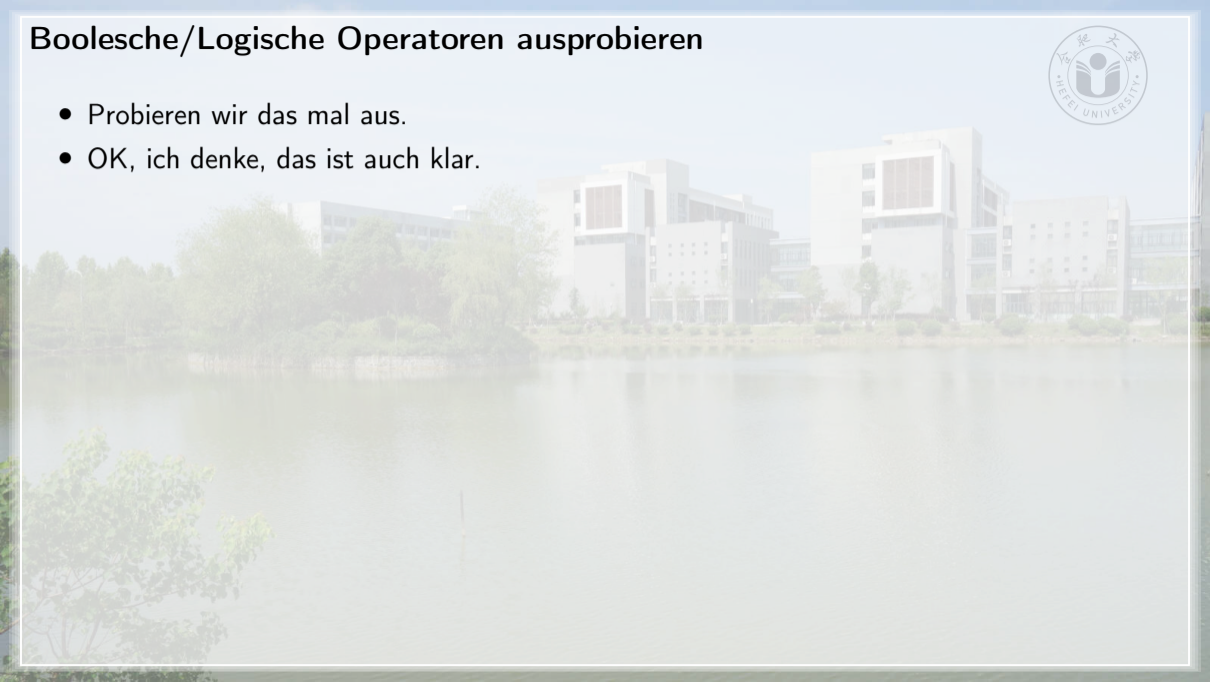


- Was ergibt `(5 < 4) or (6 < 9 < 8)`? Das ergibt `False`. `5 < 4` ist `False` und `6 < 9 < 8` ist auch `False`.

```
tweise@weise-laptop: ~  
>>> True or False  
True  
>>> True or True  
True  
>>> not True  
False  
>>> not False  
True  
>>> (True or False) and ((False or True) or (False and False))  
True  
>>> (5 < 4) or (6 < 9 < 8)  
False  
>>> 
```

# Boolesche/Logische Operatoren ausprobieren

- Probieren wir das mal aus.
- OK, ich denke, das ist auch klar.





# Zusammenfassung



# Zusammenfassung



- Die Booleschen Werte `True` und `False` werden von dem Datentyp `bool` bereitgestellt.

# Zusammenfassung



- Die Booleschen Werte `True` und `False` werden von dem Datentyp `bool` bereitgestellt.
- Sie sind oftmals das Ergebnis von Vergleichen.

# Zusammenfassung



- Die Booleschen Werte `True` und `False` werden von dem Datentyp `bool` bereitgestellt.
- Sie sind oftmals das Ergebnis von Vergleichen.
- Sie können mit den bekannten Operatoren `and`, `or`, und `not` verbunden werden.

# Zusammenfassung



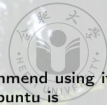
- Die Booleschen Werte `True` und `False` werden von dem Datentyp `bool` bereitgestellt.
- Sie sind oftmals das Ergebnis von Vergleichen.
- Sie können mit den bekannten Operatoren `and`, `or`, und `not` verbunden werden.
- Das ist relativ einfach zu verstehen.



谢谢你们！  
Thank you!  
Vielen Dank!



# Glossary (in English) I



**Linux** is the leading open source operating system, i.e., a free alternative to Microsoft Windows<sup>1,4,9-11</sup>. 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<sup>2</sup>. 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>.

**Python** The Python programming language<sup>6-8,12</sup>, i.e., what you will learn about in our book<sup>12</sup>. Learn more at <https://python.org>.

**Ubuntu** is a variant of the open source operating system Linux<sup>3,5</sup>. 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>.