合肥大學
HEFEI UNIVERSITY

Programming with Python

7. Der Datentyp int

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: 2026-01-29

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

Einleitung

Was wir schon wissen

- Wir können Python-Programme im PyCharm Integrated Development Environment (IDE) und auch im Terminal ausführen.

Was wir schon wissen

- Wir können Python-Programme im PyCharm Integrated Development Environment (IDE) und auch im Terminal ausführen.
- Wir kennen bereits zwei Python Kommandos

Was wir schon wissen

- Wir können Python-Programme im PyCharm Integrated Development Environment (IDE) und auch im Terminal ausführen.
- Wir kennen bereits zwei Python Kommandos:
  1. `print("Hello World!")` druckt den Text „Hello World!“ in das Terminal.

Was wir schon wissen

- Wir können Python-Programme im PyCharm Integrated Development Environment (IDE) und auch im Terminal ausführen.
- Wir kennen bereits zwei Python Kommandos:
  1. `print("Hello World!")` druckt den Text „Hello World!“ in das Terminal.
  2. `exit()` beendet den Python-Interpreter.

Was wir schon wissen

- Wir können Python-Programme im PyCharm Integrated Development Environment (IDE) und auch im Terminal ausführen.
- Wir kennen bereits zwei Python Kommandos:
  1. `print("Hello World!")` druckt den Text „Hello World!“ in das Terminal.
  2. `exit()` beendet den Python-Interpreter.
- Wäre es nicht komisch, wenn `print` *nur* „Hello World!“ ausgeben könnte?

Was wir schon wissen

- Wir können Python-Programme im PyCharm Integrated Development Environment (IDE) und auch im Terminal ausführen.
- Wir kennen bereits zwei Python Kommandos:
  1. `print("Hello World!")` druckt den Text „Hello World!“ in das Terminal.
  2. `exit()` beendet den Python-Interpreter.
- Wäre es nicht komisch, wenn `print` *nur* „Hello World!“ ausgeben könnte?
- Das würde keinen Sinn ergeben.

Was wir schon wissen

- Wir können Python-Programme im PyCharm Integrated Development Environment (IDE) und auch im Terminal ausführen.
- Wir kennen bereits zwei Python Kommandos:
  1. `print("Hello World!")` druckt den Text „Hello World!“ in das Terminal.
  2. `exit()` beendet den Python-Interpreter.
- Wäre es nicht komisch, wenn `print` *nur* „Hello World!“ ausgeben könnte?
- Das würde keinen Sinn ergeben.
- `print` ist eine Funktion, die einen Parameter hat.

Was wir schon wissen

- Wir können Python-Programme im PyCharm Integrated Development Environment (IDE) und auch im Terminal ausführen.
- Wir kennen bereits zwei Python Kommandos:
  1. `print("Hello World!")` druckt den Text „Hello World!“ in das Terminal.
  2. `exit()` beendet den Python-Interpreter.
- Wäre es nicht komisch, wenn `print` *nur* „Hello World!“ ausgeben könnte?
- Das würde keinen Sinn ergeben.
- `print` ist eine Funktion, die einen Parameter hat.
- Dieser Parameter sollte ein Text sein[1]

[1] (zumindest nehmen wir das vereinfachend an)

Was Sinn ergibt

- Auch die Funktion `exit` hat einen (optionalen) Parameter.

Was Sinn ergibt

- Auch die Funktion `exit` hat einen (optionalen) Parameter.
- Nämlich den Exit-Kode des Programmes[12], eine Ganzzahl, wobei 0 für „Erfolg“ steht.

Was Sinn ergibt

- Auch die Funktion `exit` hat einen (optionalen) Parameter.
- Nämlich den Exit-Kode des Programmes[12], eine Ganzzahl, wobei 0 für „Erfolg“ steht.
- Wir erkennen: Es ergibt Sinn, verschiedene Datentypen zu unterscheiden.

Was Sinn ergibt

- Auch die Funktion `exit` hat einen (optionalen) Parameter.
- Nämlich den Exit-Kode des Programmes[12], eine Ganzzahl, wobei 0 für „Erfolg“ steht.

- Wir erkennen: Es ergibt Sinn, verschiedene Datentypen zu unterscheiden.
- Manchmal wollen wir mit Text arbeiten.

Was Sinn ergibt

- Auch die Funktion `exit` hat einen (optionalen) Parameter.
- Nämlich den Exit-Kode des Programmes[12], eine Ganzzahl, wobei 0 für „Erfolg“ steht.

- Wir erkennen: Es ergibt Sinn, verschiedene Datentypen zu unterscheiden.
- Manchmal wollen wir mit Text arbeiten.
- Manchmal wollen wir mit Zahlen rechnen.

Was Sinn ergibt

- Auch die Funktion `exit` hat einen (optionalen) Parameter.
- Nämlich den Exit-Kode des Programmes[12], eine Ganzzahl, wobei 0 für „Erfolg“ steht.

- Wir erkennen: Es ergibt Sinn, verschiedene Datentypen zu unterscheiden.
- Manchmal wollen wir mit Text arbeiten.
- Manchmal wollen wir mit Zahlen rechnen.
- Manchmal brauchen wir nur Ja/Nein-Unterscheidungen.

Was Sinn ergibt

- Auch die Funktion `exit` hat einen (optionalen) Parameter.
- Nämlich den Exit-Kode des Programmes[12], eine Ganzzahl, wobei 0 für „Erfolg“ steht.

- Wir erkennen: Es ergibt Sinn, verschiedene Datentypen zu unterscheiden.
- Manchmal wollen wir mit Text arbeiten.
- Manchmal wollen wir mit Zahlen rechnen.
- Manchmal brauchen wir nur Ja/Nein-Unterscheidungen.
- Datentypen unterstützen verschiedene Operationen

Was Sinn ergibt

- Auch die Funktion `exit` hat einen (optionalen) Parameter.
- Nämlich den Exit-Kode des Programmes[12], eine Ganzzahl, wobei 0 für „Erfolg“ steht.

- Wir erkennen: Es ergibt Sinn, verschiedene Datentypen zu unterscheiden.
- Manchmal wollen wir mit Text arbeiten.
- Manchmal wollen wir mit Zahlen rechnen.
- Manchmal brauchen wir nur Ja/Nein-Unterscheidungen.
- Datentypen unterstützen verschiedene Operationen, Zahlen können z. B. addiert, subtrahiert, oder multipliziert werden

Was Sinn ergibt

- Auch die Funktion `exit` hat einen (optionalen) Parameter.
- Nämlich den Exit-Kode des Programmes[12], eine Ganzzahl, wobei 0 für „Erfolg“ steht.

- Wir erkennen: Es ergibt Sinn, verschiedene Datentypen zu unterscheiden.
- Manchmal wollen wir mit Text arbeiten.
- Manchmal wollen wir mit Zahlen rechnen.
- Manchmal brauchen wir nur Ja/Nein-Unterscheidungen.
- Datentypen unterstützen verschiedene Operationen, Zahlen können z. B. addiert, subtrahiert, oder multipliziert werden, Texte können miteinander verkettet oder z. B. in Groß- bzw. Kleinschreibung umgewandelt werden.

Was wir jetzt lernen

- Nun wollen wir die einfachen Datentypen von Python kennenlernen

Was wir jetzt lernen

- Nun wollen wir die einfachen Datentypen von Python kennenlernen:
  - `int`: der Datentyp für die ganzen Zahlen $\mathbb{Z}$

Was wir jetzt lernen

- Nun wollen wir die einfachen Datentypen von Python kennenlernen:
  - `int`: der Datentyp für die ganzen Zahlen $\mathbb{Z}$,
  - `float`: der Datentyp für eine Untermenge der reellen Zahlen $\mathbb{R}$

Was wir jetzt lernen

- Nun wollen wir die einfachen Datentypen von Python kennenlernen:
  - `int`: der Datentyp für die ganzen Zahlen $\mathbb{Z}$,
  - `float`: der Datentyp für eine Untermenge der reellen Zahlen $\mathbb{R}$,
  - `bool`: Boolesche Werte, die entweder `True` (Wahr) oder `False` (Falsch) seien können

Was wir jetzt lernen

- Nun wollen wir die einfachen Datentypen von Python kennenlernen:
  - `int`: der Datentyp für die ganzen Zahlen $\mathbb{Z}$,
  - `float`: der Datentyp für eine Untermenge der reellen Zahlen $\mathbb{R}$,
  - `bool`: Boolesche Werte, die entweder `True` (Wahr) oder `False` (Falsch) seien können,
  - `str`: Text-fragmente beliebiger Länge

Was wir jetzt lernen

- Nun wollen wir die einfachen Datentypen von Python kennenlernen:
  - `int`: der Datentyp für die ganzen Zahlen $\mathbb{Z}$,
  - `float`: der Datentyp für eine Untermenge der reellen Zahlen $\mathbb{R}$,
  - `bool`: Boolesche Werte, die entweder `True` (Wahr) oder `False` (Falsch) seien können,
  - `str`: Text-fragmente beliebiger Länge, und
  - `None`: nichts, das Ergebnis einer Operation die keinen Rückgabewert hat.

Was wir jetzt lernen

- Nun wollen wir die einfachen Datentypen von Python kennenlernen:
  - `int`: der Datentyp für die ganzen Zahlen $\mathbb{Z}$,
  - `float`: der Datentyp für eine Untermenge der reellen Zahlen $\mathbb{R}$,
  - `bool`: Boolesche Werte, die entweder `True` (Wahr) oder `False` (Falsch) seien können,
  - `str`: Text-fragmente beliebiger Länge, und
  - `None`: nichts, das Ergebnis einer Operation die keinen Rückgabewert hat.
- Wir fangen mit `int` an.

Rechnen mit Ganzen Zahlen

Der Datentyp int

- Das rechnen mit ganzen Zahlen ist das Erste, was man in der Grundschulmathematik lernen.

Der Datentyp int

- Das rechnen mit ganzen Zahlen ist das Erste, was man in der Grundschulmathematik lernen.
- Es ist auch das Erste, dass Sie hier lernen.

Der Datentyp int

- Das rechnen mit ganzen Zahlen ist das Erste, was man in der Grundschulmathematik lernen.
- Es ist auch das Erste, dass Sie hier lernen.
- *Integer* ist ein lateinisches Wort das „ganz“ oder „intakt“ bedeutet.

Der Datentyp int

- Das rechnen mit ganzen Zahlen ist das Erste, was man in der Grundschulmathematik lernen.
- Es ist auch das Erste, dass Sie hier lernen.
- *Integer* ist ein lateinisches Wort das „ganz“ oder „intakt“ bedeutet.
- Die ganzen Zahlen umfassen daher die negativen Ganzzahlen, 0, und die positiven Ganzzahlen – alle ohne Kommastellen.

Der Datentyp int

- Das rechnen mit ganzen Zahlen ist das Erste, was man in der Grundschulmathematik lernen.
- Es ist auch das Erste, dass Sie hier lernen.
- *Integer* ist ein lateinisches Wort das „ganz“ oder „intakt“ bedeutet.
- Die ganzen Zahlen umfassen daher die negativen Ganzzahlen, 0, und die positiven Ganzzahlen – alle ohne Kommastellen.
- Viele Programmiersprachen bieten verschiedene Datentypen mit verschiedenen Wertebereichen für Ganzzahlen.

Der Datentyp int

- Das rechnen mit ganzen Zahlen ist das Erste, was man in der Grundschulmathematik lernen.
- Es ist auch das Erste, dass Sie hier lernen.
- *Integer* ist ein lateinisches Wort das „ganz“ oder „intakt“ bedeutet.
- Die ganzen Zahlen umfassen daher die negativen Ganzzahlen, 0, und die positiven Ganzzahlen – alle ohne Kommastellen.
- Viele Programmiersprachen bieten verschiedene Datentypen mit verschiedenen Wertebereichen für Ganzzahlen. In Java ist `byte` z. B. ein Ganzzahltyp mit Wertebereich $-2^7..2^7-1$ wohingegen `long` den Wertebereich $-2^{63}..2^{63}-1$ abdeckt.

Der Datentyp int

- Das rechnen mit ganzen Zahlen ist das Erste, was man in der Grundschulmathematik lernen.
- Es ist auch das Erste, dass Sie hier lernen.
- *Integer* ist ein lateinisches Wort das „ganz“ oder „intakt“ bedeutet.
- Die ganzen Zahlen umfassen daher die negativen Ganzzahlen, 0, und die positiven Ganzzahlen – alle ohne Kommastellen.
- Viele Programmiersprachen bieten verschiedene Datentypen mit verschiedenen Wertebereichen für Ganzzahlen. In Java ist `byte` z. B. ein Ganzzahltyp mit Wertebereich $-2^7..2^7-1$ wohingegen `long` den Wertebereich $-2^{63}..2^{63}-1$ abdeckt. Der C17-Standard für C listed mindestens zehn Ganzzahltypen[20].

Der Datentyp int

- Das rechnen mit ganzen Zahlen ist das Erste, was man in der Grundschulmathematik lernen.
- Es ist auch das Erste, dass Sie hier lernen.
- *Integer* ist ein lateinisches Wort das „ganz“ oder „intakt“ bedeutet.
- Die ganzen Zahlen umfassen daher die negativen Ganzzahlen, 0, und die positiven Ganzzahlen – alle ohne Kommastellen.
- Viele Programmiersprachen bieten verschiedene Datentypen mit verschiedenen Wertebereichen für Ganzzahlen. In Java ist `byte` z. B. ein Ganzzahltyp mit Wertebereich $-2^7..2^7-1$ wohingegen `long` den Wertebereich $-2^{63}..2^{63}-1$ abdeckt. Der C17-Standard für C listed mindestens zehn Ganzzahltypen[20].
- Python 3 hat nur einen einzigen Datentyp für die Ganzzahlen: `int`.

Der Datentyp int

- Das rechnen mit ganzen Zahlen ist das Erste, was man in der Grundschulmathematik lernen.
- Es ist auch das Erste, dass Sie hier lernen.
- *Integer* ist ein lateinisches Wort das „ganz“ oder „intakt“ bedeutet.
- Die ganzen Zahlen umfassen daher die negativen Ganzzahlen, 0, und die positiven Ganzzahlen – alle ohne Kommastellen.
- Viele Programmiersprachen bieten verschiedene Datentypen mit verschiedenen Wertebereichen für Ganzzahlen. In Java ist `byte` z. B. ein Ganzzahltyp mit Wertebereich $-2^7..2^7-1$ wohingegen `long` den Wertebereich $-2^{63}..2^{63}-1$ abdeckt. Der C17-Standard für C listed mindestens zehn Ganzzahltypen[20].
- Python 3 hat nur einen einzigen Datentyp für die Ganzzahlen: `int`.
- Dieser Datentyp hat im Grunde einen unbegrenzten Wertebereich.

Der Datentyp int

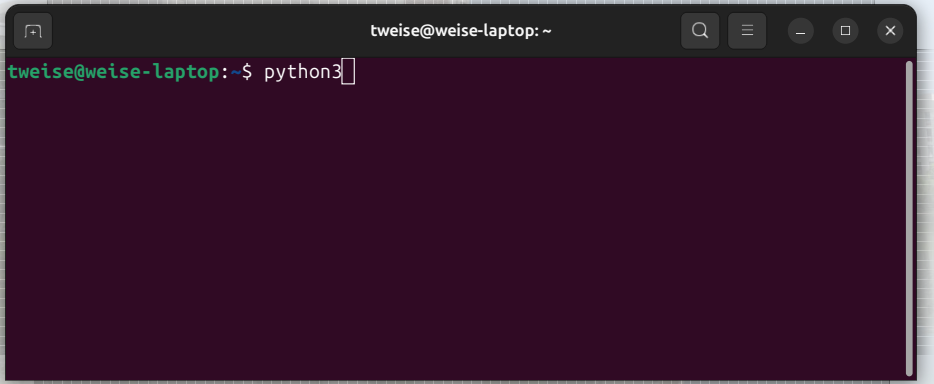
- Das rechnen mit ganzen Zahlen ist das Erste, was man in der Grundschulmathematik lernen.
- Es ist auch das Erste, dass Sie hier lernen.
- *Integer* ist ein lateinisches Wort das „ganz“ oder „intakt“ bedeutet.
- Die ganzen Zahlen umfassen daher die negativen Ganzzahlen, 0, und die positiven Ganzzahlen – alle ohne Kommastellen.
- Viele Programmiersprachen bieten verschiedene Datentypen mit verschiedenen Wertebereichen für Ganzzahlen. In Java ist `byte` z. B. ein Ganzzahltyp mit Wertebereich $-2^7..2^7-1$ wohingegen `long` den Wertebereich $-2^{63}..2^{63}-1$ abdeckt. Der C17-Standard für C listed mindestens zehn Ganzzahltypen[20].
- Python 3 hat nur einen einzigen Datentyp für die Ganzzahlen: `int`.
- Dieser Datentyp hat im Grunde einen unbegrenzten Wertebereich.
- Naja, begrenzt durch den Speicher Ihres Computers.

Grundrechenarten

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

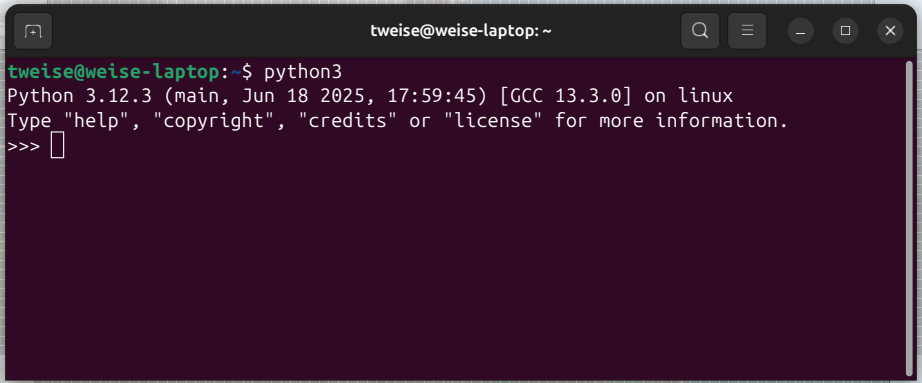
Grundrechenarten

- Wir schreiben `python3` und drücken ↵.



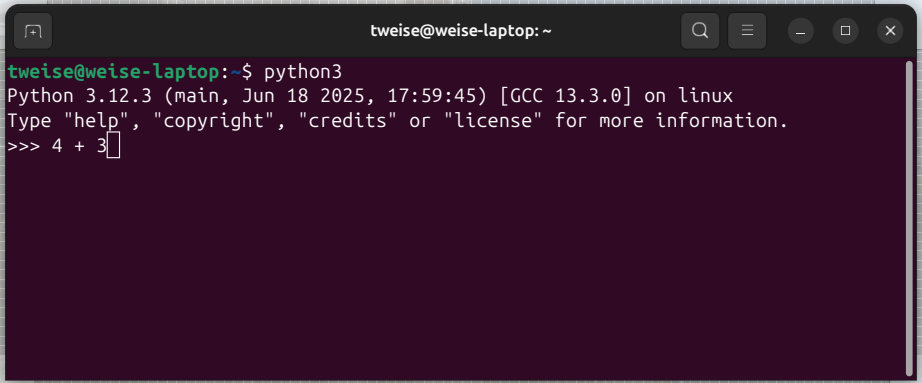
Grundrechenarten

- Der Python-Interpreter startet.



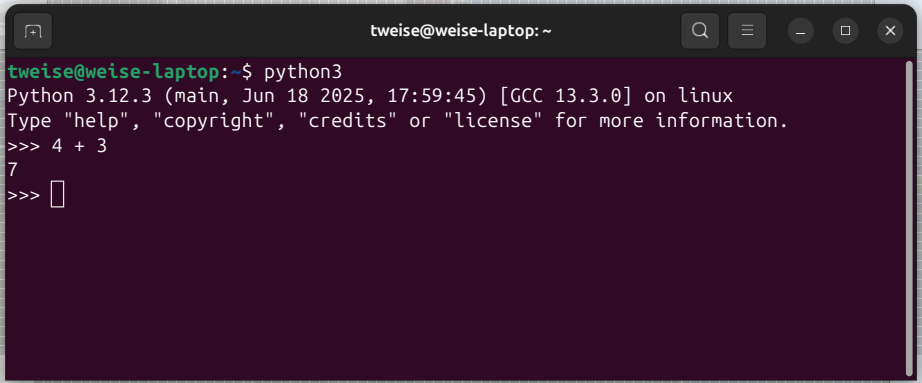
Grundrechenarten

- Wir schreiben `4 + 3` und drücken ↵.



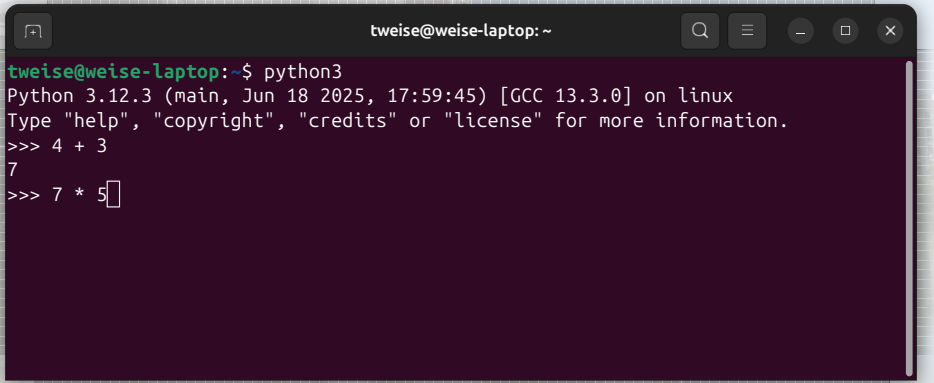
Grundrechenarten

- Das Ergebnis erscheint.



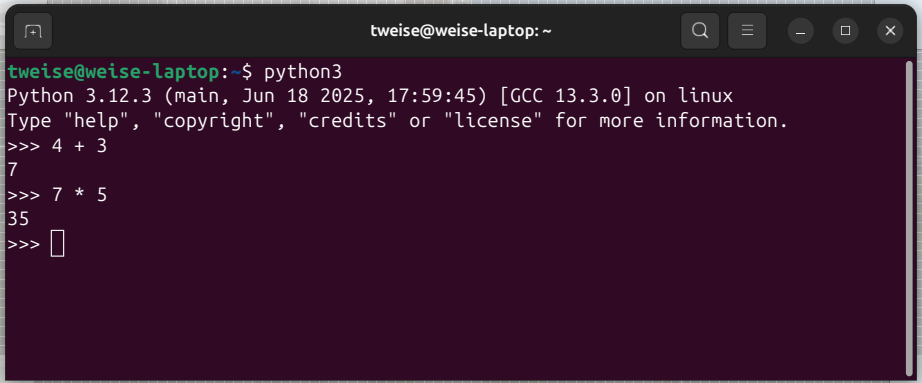
Grundrechenarten

- Wir schreiben `7 * 5` und drücken ↵.



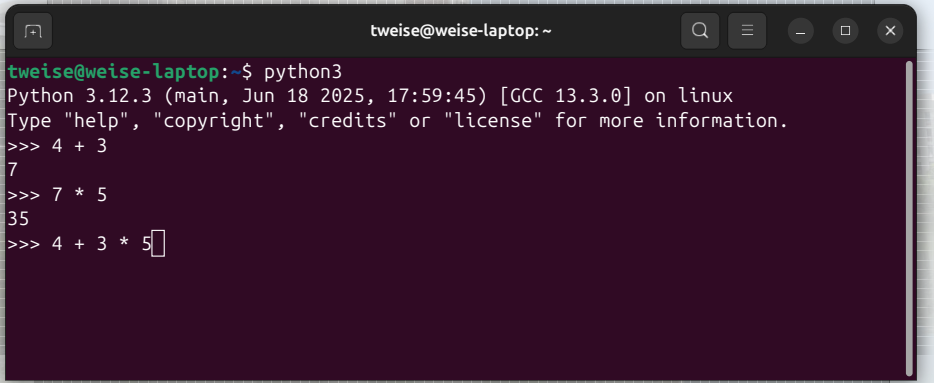
Grundrechenarten

- Das Ergebnis erscheint.



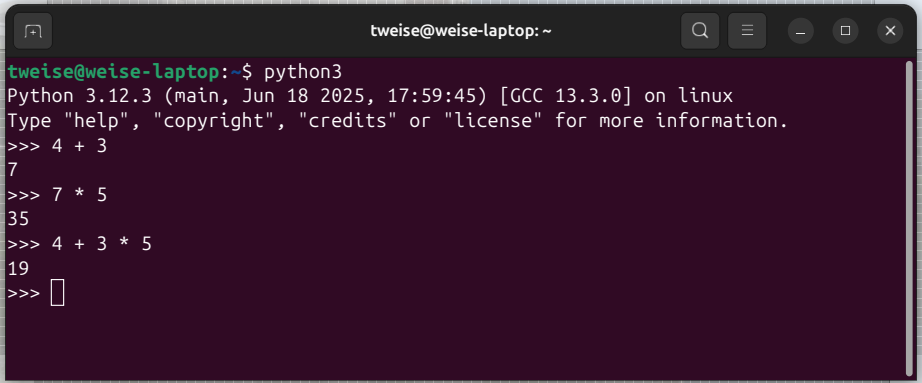
Grundrechenarten

- Wir schreiben `4 + 3 * 5` und drücken ↵.



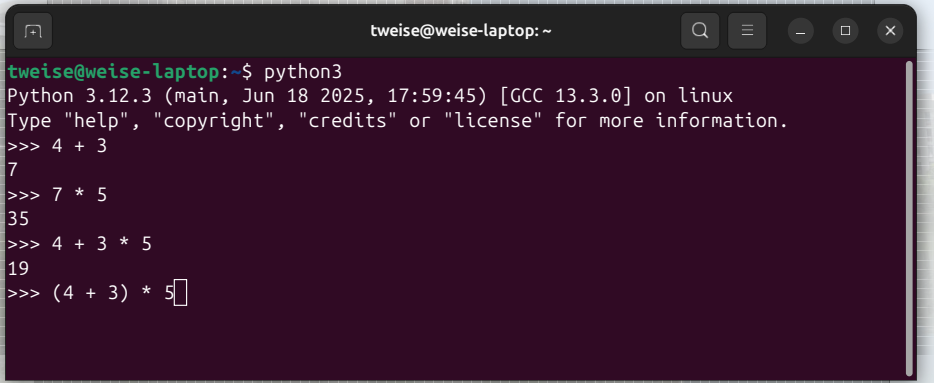
Grundrechenarten

- Das Ergebnis erscheint: Python beherrscht die Operatorreihenfolge!



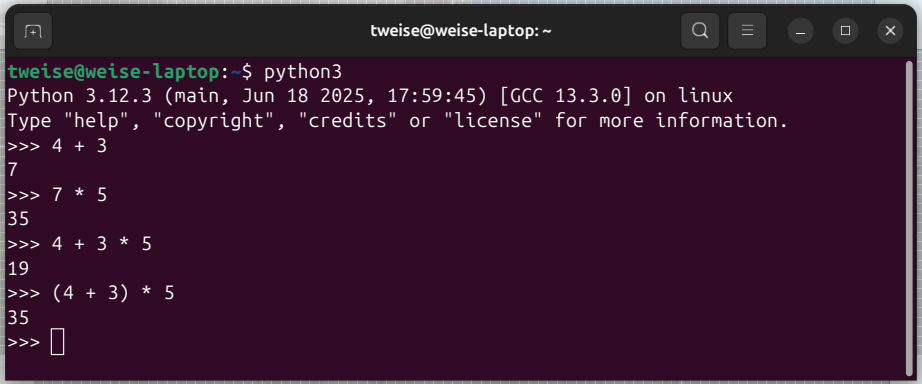
Grundrechenarten

- Wir schreiben `(4 + 3)* 5` und drücken ↵.



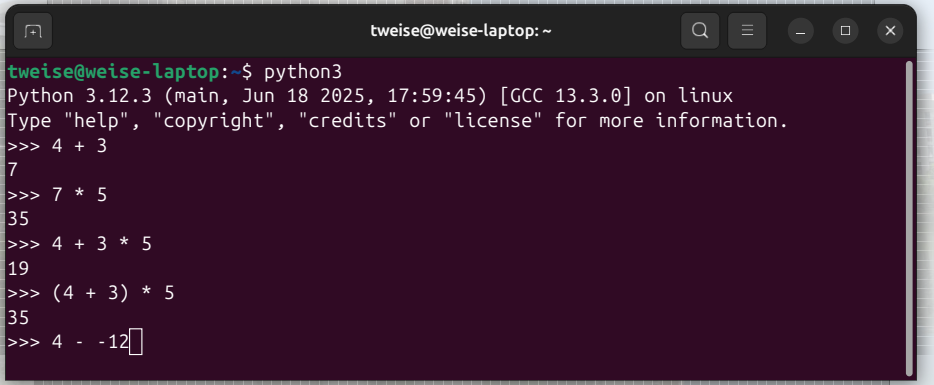
Grundrechenarten

- Das Ergebnis erscheint: Python beherrscht Klammerrechnung.



Grundrechenarten

- Wir schreiben `4 - -12` und drücken ↵.



Grundrechenarten

- Das Ergebnis erscheint: Python beherrscht negative Zahlen.

Grundrechenarten

- Wir schreiben `((4 + 3)* (4 - -12)- 5)* 3` und drücken ↵.

Grundrechenarten

- Das Ergebnis erscheint: `= (7 * 16 - 5)* 3 = 107 * 3 = 321`.

Grundrechenarten

- Wir schreiben die Ganzzahldivision `32 // 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `8` erscheint.

Grundrechenarten

- Wir schreiben die Ganzzahldivision `33 // 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `8` erscheint (der Rest wäre 1).

Grundrechenarten

- Wir schreiben die Ganzzahldivision `34 // 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `8` erscheint (der Rest wäre 2).

Grundrechenarten

- Wir schreiben die Ganzzahldivision `35 // 4` und drücken ↵.

Grundrechenarten

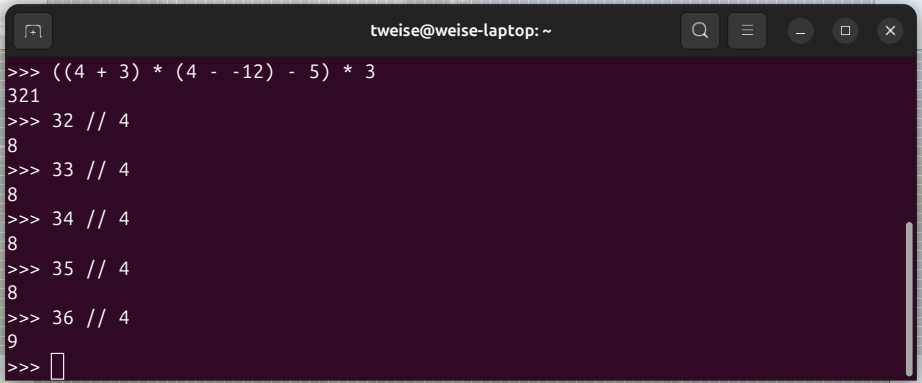
- Das Ergebnis `8` erscheint (der Rest wäre 3).

Grundrechenarten

- Wir schreiben die Ganzzahldivision `36 // 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `9` erscheint (der Rest wäre 0).



Grundrechenarten

- Wir schreiben die Fließkommadivision `32 / 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `8.0` erscheint (Fließkommazahlen lernen wir später).

Grundrechenarten

- Wir schreiben die Fließkommadivision `33 / 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `8.25` erscheint (Fließkommazahlen lernen wir später).

tweise@weise-laptop: ~

```
>>> 33 // 4
8
>>> 34 // 4
8
>>> 35 // 4
8
>>> 36 // 4
9
>>> 32 / 4
8.0
>>> 33 / 4
8.25
>>>
```

Grundrechenarten

- Wir schreiben die Fließkommadivision `34 / 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `8.5` erscheint (Fließkommazahlen lernen wir später).

Grundrechenarten

- Wir schreiben die Fließkommadivision `35 / 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `8.75` erscheint (Fließkommazahlen lernen wir später).

Grundrechenarten

- Wir schreiben die Fließkommadivision `36 / 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `9.0` erscheint (Fließkommazahlen lernen wir später).

Grundrechenarten

- Wir berechnen den Rest der Ganzzahldivision `33 % 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `1` erscheint.

Grundrechenarten

- Wir berechnen den Rest der Ganzzahldivision `34 % 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `2` erscheint.

Grundrechenarten

- Wir berechnen den Rest der Ganzzahldivision `35 % 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `3` erscheint.

Grundrechenarten

- Wir berechnen den Rest der Ganzzahldivision `36 % 4` und drücken ↵.

Grundrechenarten

- Das Ergebnis `0` erscheint.

Grundrechenarten

- Wir schreiben `exit()` um den Interpreter zu verlassen und drücken ↵.

Grundrechenarten

- Wir sind zurück im normalen Terminal.

Grundrechenarten

- In Python gibt es zwei Arten von Divisionen

Grundrechenarten

- In Python gibt es zwei Arten von Divisionen
  1. die Ganzzahldivision `//` liefert ganzzahlige Ergebnisse, wobei der Rest mit `%` berechnet werden kann

Grundrechenarten

- In Python gibt es zwei Arten von Divisionen
  1. die Ganzzahldivision `//` liefert ganzzahlige Ergebnisse, wobei der Rest mit `%` berechnet werden kann,
  2. die Fließkommadivison liefert Ergebnisse, die keine `int`-Werte mehr sind und Kommastellen haben[33].

Grundrechenarten

- In Python gibt es zwei Arten von Divisionen
  1. die Ganzzahldivision `//` liefert ganzzahlige Ergebnisse, wobei der Rest mit `%` berechnet werden kann,
  2. die Fließkommadivison liefert Ergebnisse, die keine `int`-Werte mehr sind und Kommastellen haben[33]. Wir lernen den Datentyp `float` später kennen.

Grundrechenarten

- In Python gibt es zwei Arten von Divisionen
  1. die Ganzzahldivision `//` liefert ganzzahlige Ergebnisse, wobei der Rest mit `%` berechnet werden kann,
  2. die Fließkommadivison liefert Ergebnisse, die keine `int`-Werte mehr sind und Kommastellen haben[33]. Wir lernen den Datentyp `float` später kennen.

Gute Praxis

Seien Sie immer vorsichtig und passen gut auf, welchen Divisionsoperator Sie mit dem Datentyp `int` verwenden.

Grundrechenarten

- In Python gibt es zwei Arten von Divisionen
  1. die Ganzzahldivision `//` liefert ganzzahlige Ergebnisse, wobei der Rest mit `%` berechnet werden kann,
  2. die Fließkommadivison liefert Ergebnisse, die keine `int`-Werte mehr sind und Kommastellen haben[33]. Wir lernen den Datentyp `float` später kennen.

Gute Praxis

Seien Sie immer vorsichtig und passen gut auf, welchen Divisionsoperator Sie mit dem Datentyp `int` verwenden. Wenn Sie ein ganzzahliges Ergebnis brauchen, nutzen Sie immer `//`.

Grundrechenarten

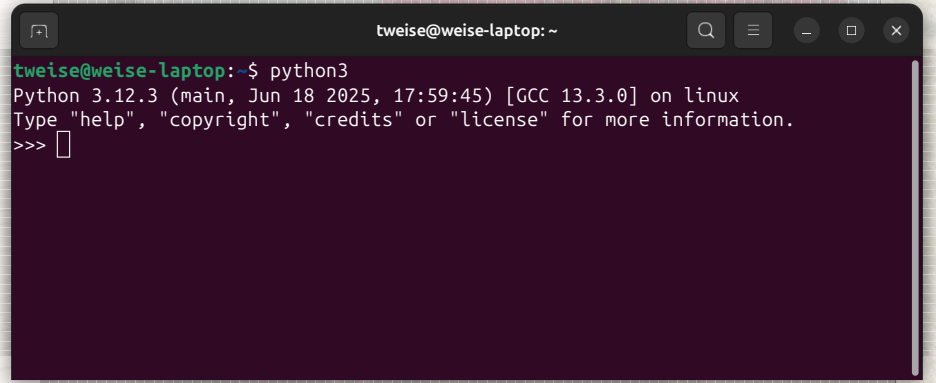
- In Python gibt es zwei Arten von Divisionen
  1. die Ganzzahldivision `//` liefert ganzzahlige Ergebnisse, wobei der Rest mit `%` berechnet werden kann,
  2. die Fließkommadivison liefert Ergebnisse, die keine `int`-Werte mehr sind und Kommastellen haben[33]. Wir lernen den Datentyp `float` später kennen.

Gute Praxis

Seien Sie immer vorsichtig und passen gut auf, welchen Divisionsoperator Sie mit dem Datentyp `int` verwenden. Wenn Sie ein ganzzahliges Ergebnis brauchen, nutzen Sie immer `//`. Merken Sie sich, dass `/` *immer* `float`-Werte zurückliefert, selbst wenn das Ergebnis eine Ganzzahl ist.

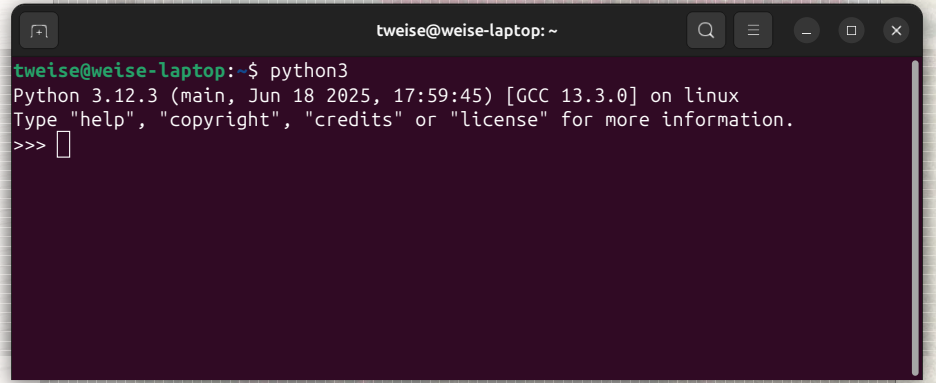
Potenzen von Ganzzahlen

- In Python stellt der `**` Operator das Potenzieren dar.



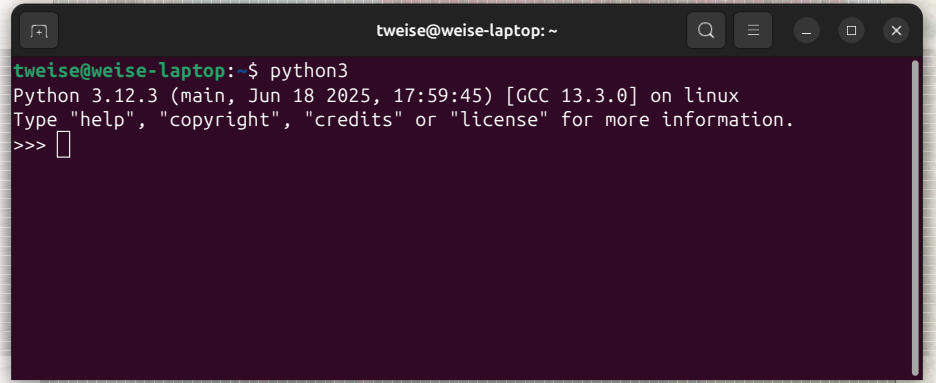
Potenzen von Ganzzahlen

- In Python stellt der `**` Operator das Potenzieren dar.
- `a ** b` ist equivalent zu a^b.



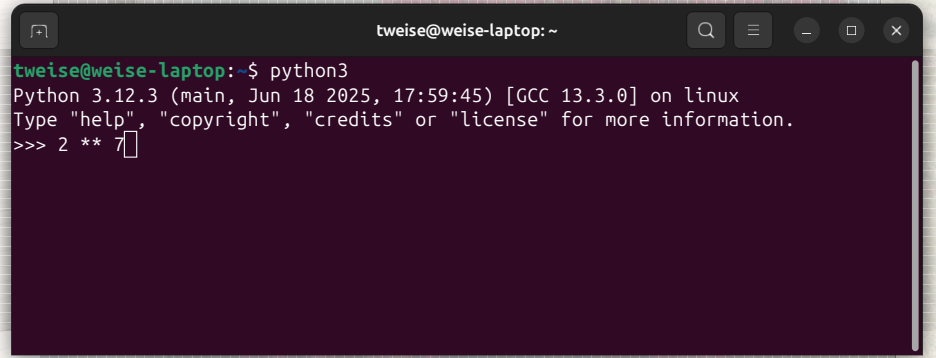
Potenzen von Ganzzahlen

- In Python stellt der `**` Operator das Potenzieren dar.
- `a ** b` ist equivalent zu a^b.
- Wir öffnen also wieder ein Terminal und starten eine interaktive Python-Session...



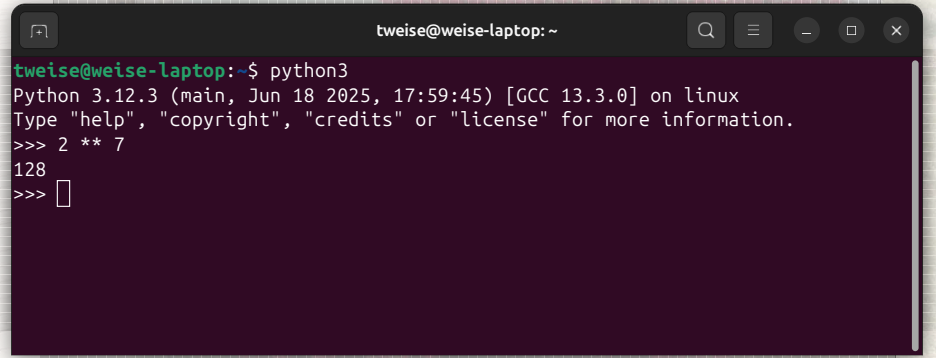
Potenzen von Ganzzahlen

- 2^7 kann als `2 ** 7` geschrieben werden.



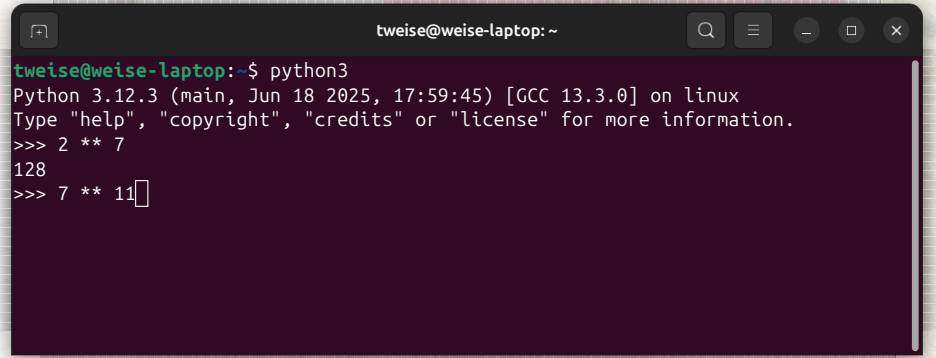
Potenzen von Ganzzahlen

- 2^7 kann als `2 ** 7` geschrieben werden und ergibt 128.



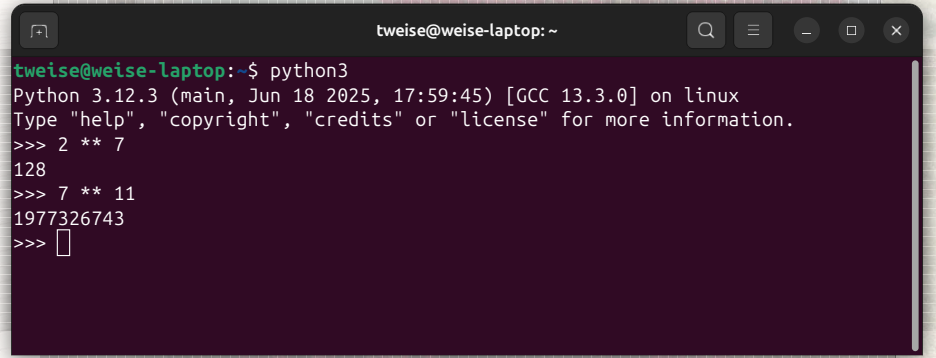
Potenzen von Ganzzahlen

- 7^{11} kann als `7 ** 11` geschrieben werden.



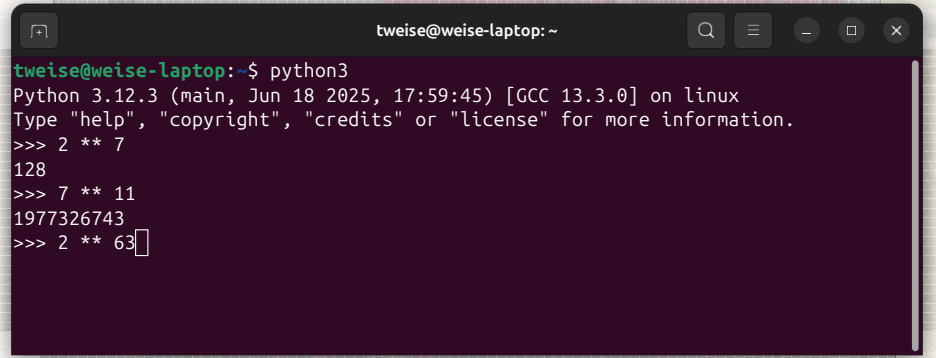
Potenzen von Ganzzahlen

- 7^{11} kann als `7 ** 11` geschrieben werden und ergibt 1 977 326 743.



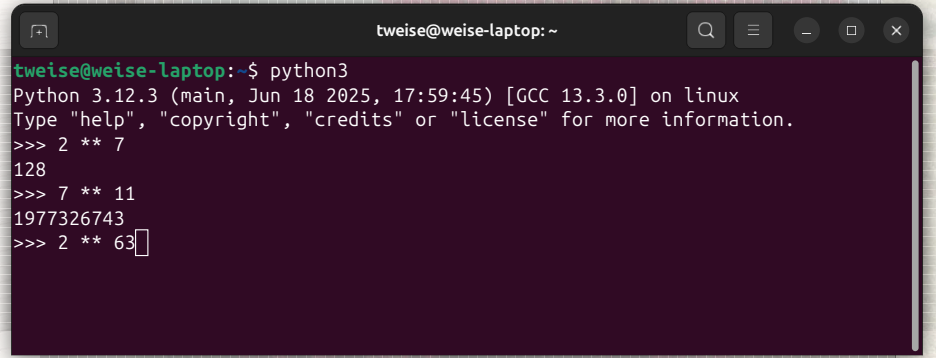
Potenzen von Ganzzahlen

- In vielen Programmiersprachen sind die größten ganzzahligen Datentypen 64 bit breit.



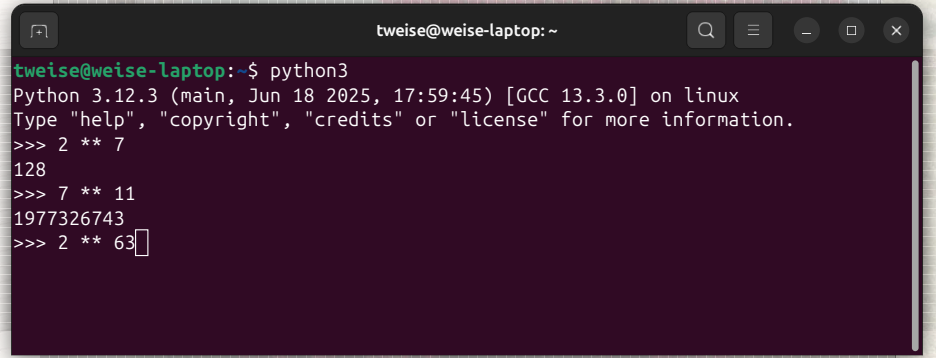
Potenzen von Ganzzahlen

- In vielen Programmiersprachen sind die größten ganzzahligen Datentypen 64 bit breit.
- Sind sie vorzeichenbehaftet, ergibt dass den Wertebereich $-2^{63}..2^{63}-1$.



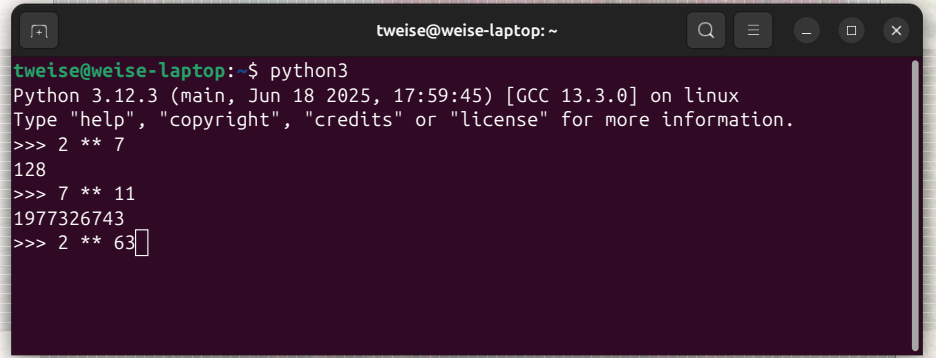
Potenzen von Ganzzahlen

- In vielen Programmiersprachen sind die größten ganzzahligen Datentypen 64 bit breit.
- Sind sie vorzeichenbehaftet, ergibt dass den Wertebereich $-2^{63}..2^{63}-1$.
- Ohne Vorzeichen (immer positiv) haben sie den Wertebereich $0..2^{64}-1$.



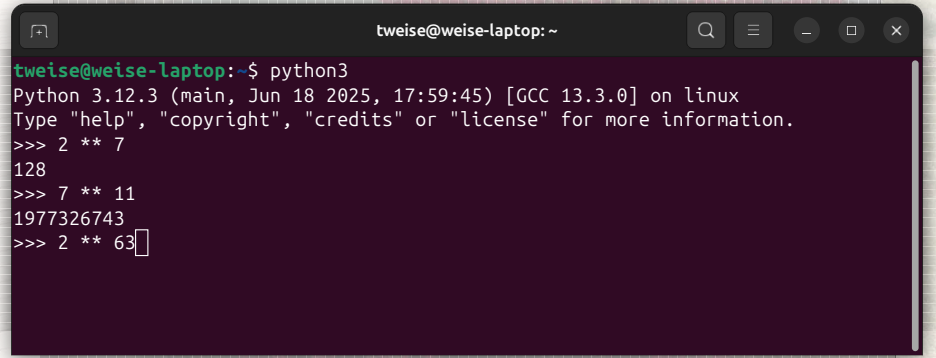
Potenzen von Ganzzahlen

- In vielen Programmiersprachen sind die größten ganzzahligen Datentypen 64 bit breit.
- Ohne Vorzeichen (immer positiv) haben sie den Wertebereich $0..2^{64}-1$.
- Python's `int` ist vorzeichenbehaftet hat aber eine (theoretisch) unbegrenzte Größe.



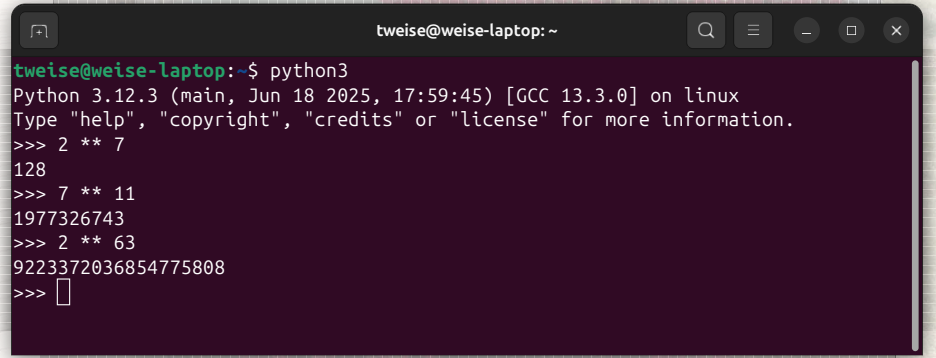
Potenzen von Ganzzahlen

- In vielen Programmiersprachen sind die größten ganzzahligen Datentypen 64 bit breit.
- Berechnen wir 2^{63} als `2 ** 63`.



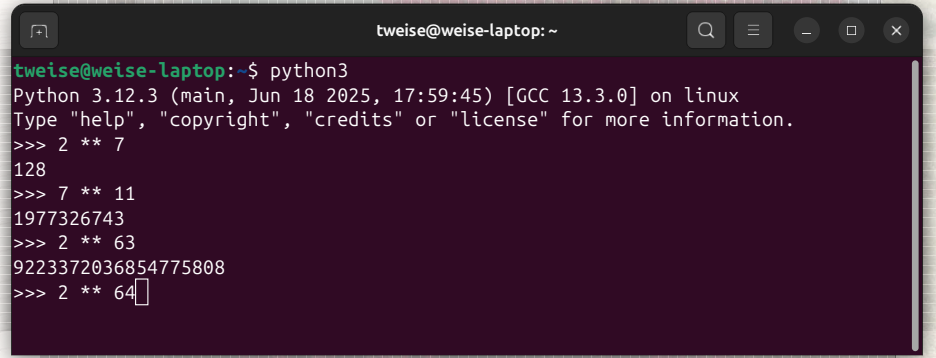
Potenzen von Ganzzahlen

- In vielen Programmiersprachen sind die größten ganzzahligen Datentypen 64 bit breit.
- Berechnen wir 2^{63} als `2 ** 63`, so bekommen wir 9 223 372 036 854 775 808.



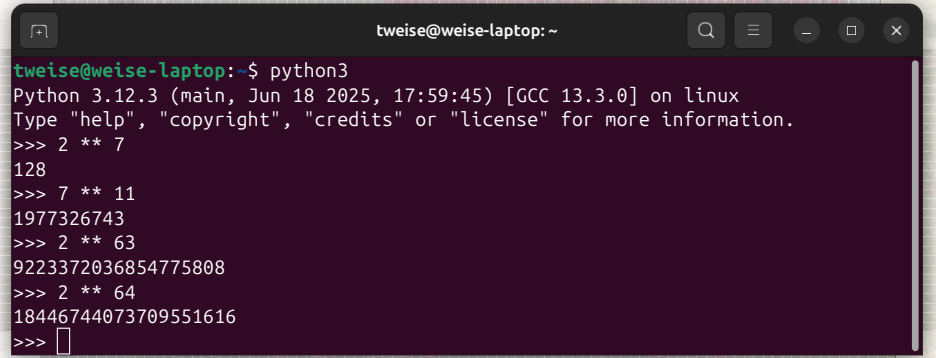
Potenzen von Ganzzahlen

- In vielen Programmiersprachen sind die größten ganzzahligen Datentypen 64 bit breit.
- Berechnen wir 2^{63} als `2 ** 63`, so bekommen wir 9 223 372 036 854 775 808.
- 2^{64} kann als `2 ** 64` geschrieben werden.



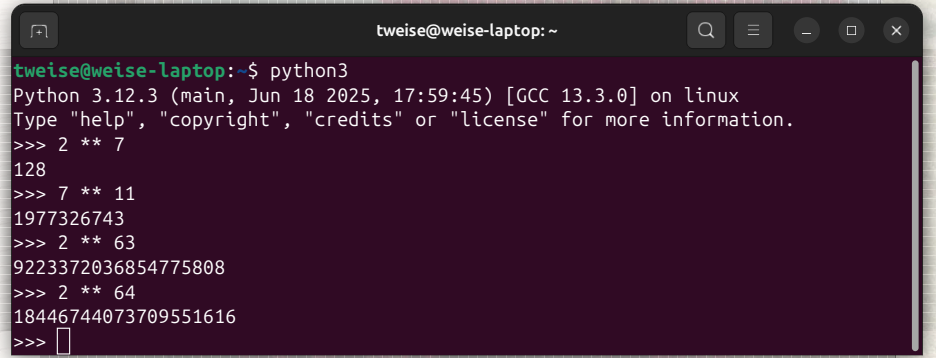
Potenzen von Ganzzahlen

- 2^{64} kann als `2 ** 64` geschrieben werden und ergibt 18 446 744 073 709 551 616.



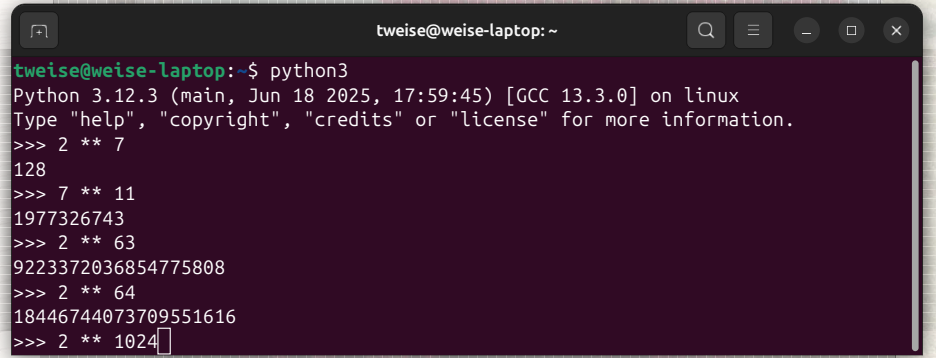
Potenzen von Ganzzahlen

- Probieren wir mal eine wirklich große Zahl.



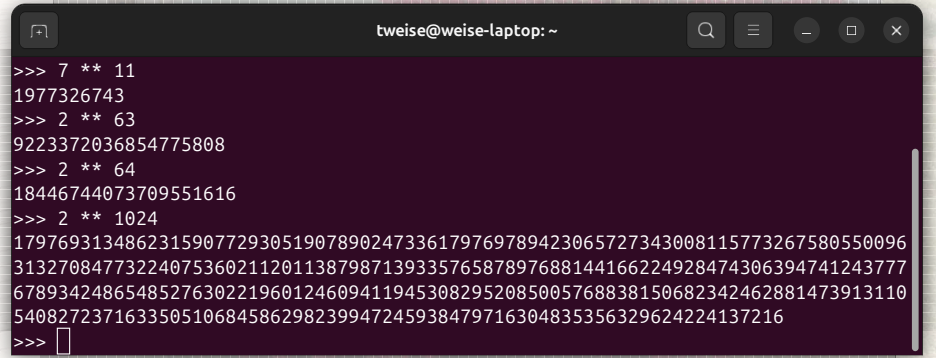
Potenzen von Ganzzahlen

- Probieren wir mal eine wirklich große Zahl.
- 2^{1024} kann als `2 ** 1024` geschrieben werden.



Potenzen von Ganzzahlen

- Probieren wir mal eine wirklich große Zahl.
- 2^{1024} kann als `2 ** 1024` geschrieben werden und ergibt ... sehr viel.



Binäres Zahlensystem und Bit-weise Operatoren

- Wie Sie ja wissen, werden alle Dinge im Computer elementar durch 0en und 1en dargestellt.

Binäres Zahlensystem und Bit-weise Operatoren

- Wie Sie ja wissen, werden alle Dinge im Computer elementar durch 0en und 1en dargestellt.
- Durch das sogenannte binäre Zahlensystem können beliebige dezimale Zahlen dargestellt werden.

Binäres Zahlensystem und Bit-weise Operatoren

- Wie Sie ja wissen, werden alle Dinge im Computer elementar durch 0en und 1en dargestellt.
- Durch das sogenannte binäre Zahlensystem können beliebige dezimale Zahlen dargestellt werden.
- binär-0 ist dezimal-0

Binäres Zahlensystem und Bit-weise Operatoren

- Wie Sie ja wissen, werden alle Dinge im Computer elementar durch 0en und 1en dargestellt.
- Durch das sogenannte binäre Zahlensystem können beliebige dezimale Zahlen dargestellt werden.
- binär-0 ist dezimal-0, binär-1 ist dezimal-1

Binäres Zahlensystem und Bit-weise Operatoren

- Wie Sie ja wissen, werden alle Dinge im Computer elementar durch 0en und 1en dargestellt.
- Durch das sogenannte binäre Zahlensystem können beliebige dezimale Zahlen dargestellt werden.
- binär-0 ist dezimal-0, binär-1 ist dezimal-1, binär-10 ist dezimal-2

Binäres Zahlensystem und Bit-weise Operatoren

- Wie Sie ja wissen, werden alle Dinge im Computer elementar durch `0`en und `1`en dargestellt.
- Durch das sogenannte binäre Zahlensystem können beliebige dezimale Zahlen dargestellt werden.
- binär-`0` ist dezimal-0, binär-`1` ist dezimal-1, binär-`10` ist dezimal-2, binär-`11` ist dezimal-3

Binäres Zahlensystem und Bit-weise Operatoren

- Wie Sie ja wissen, werden alle Dinge im Computer elementar durch 0en und 1en dargestellt.
- Durch das sogenannte binäre Zahlensystem können beliebige dezimale Zahlen dargestellt werden.
- binär-0 ist dezimal-0, binär-1 ist dezimal-1, binär-10 ist dezimal-2, binär-11 ist dezimal-3, binär-100 ist dezimal-4

Binäres Zahlensystem und Bit-weise Operatoren

- Wie Sie ja wissen, werden alle Dinge im Computer elementar durch 0en und 1en dargestellt.
- Durch das sogenannte binäre Zahlensystem können beliebige dezimale Zahlen dargestellt werden.
- binär-0 ist dezimal-0, binär-1 ist dezimal-1, binär-10 ist dezimal-2, binär-11 ist dezimal-3, binär-100 ist dezimal-4, binär-101 ist dezimal-5

Binäres Zahlensystem und Bit-weise Operatoren

- Wie Sie ja wissen, werden alle Dinge im Computer elementar durch 0en und 1en dargestellt.
- Durch das sogenannte binäre Zahlensystem können beliebige dezimale Zahlen dargestellt werden.
- binär-0 ist dezimal-0, binär-1 ist dezimal-1, binär-10 ist dezimal-2, binär-11 ist dezimal-3, binär-100 ist dezimal-4, binär-101 ist dezimal-5, binär-110 ist dezimal-6

Binäres Zahlensystem und Bit-weise Operatoren

- Wie Sie ja wissen, werden alle Dinge im Computer elementar durch 0en und 1en dargestellt.
- Durch das sogenannte binäre Zahlensystem können beliebige dezimale Zahlen dargestellt werden.
- binär-0 ist dezimal-0, binär-1 ist dezimal-1, binär-10 ist dezimal-2, binär-11 ist dezimal-3, binär-100 ist dezimal-4, binär-101 ist dezimal-5, binär-110 ist dezimal-6, und so weiter.

Binäres Zahlensystem und Bit-weise Operatoren

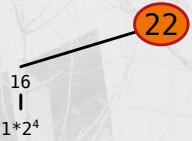
- Wie Sie ja wissen, werden alle Dinge im Computer elementar durch 0en und 1en dargestellt.
- Durch das sogenannte binäre Zahlensystem können beliebige dezimale Zahlen dargestellt werden.
- binär-0 ist dezimal-0, binär-1 ist dezimal-1, binär-10 ist dezimal-2, binär-11 ist dezimal-3, binär-100 ist dezimal-4, binär-101 ist dezimal-5, binär-110 ist dezimal-6, und so weiter.
- So weren auch die `int`-Werte in Python letztendlich als Binärzahlen dargestellt.

Binäres Zahlensystem und Bit-weise Operatoren

- Frischen wir noch einmal auf, wie das funktioniert.

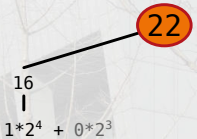
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



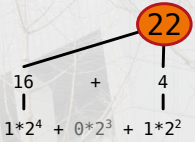
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



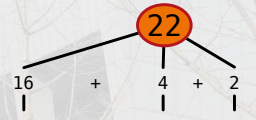
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



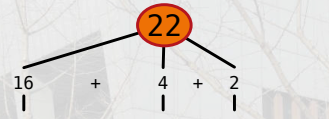
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



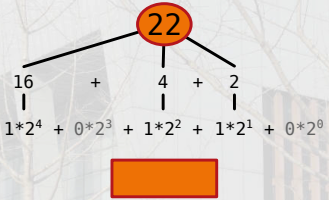
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



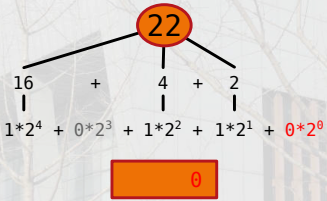
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



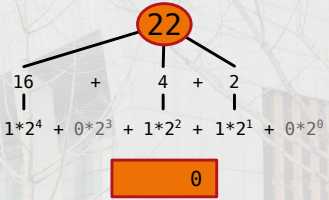
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



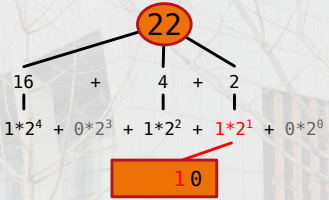
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



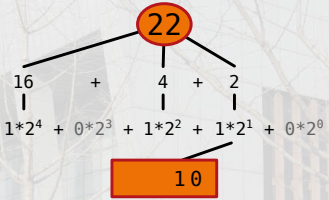
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



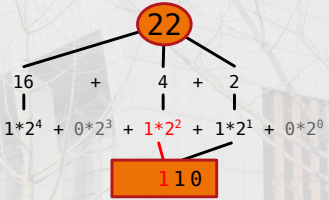
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



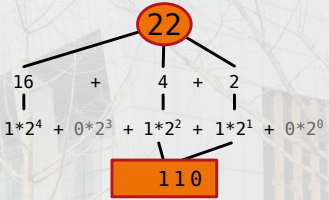
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



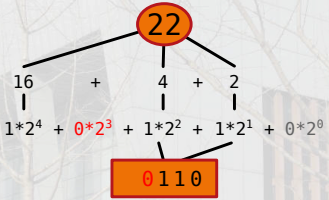
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



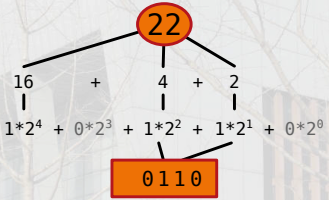
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



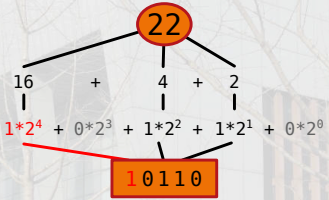
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



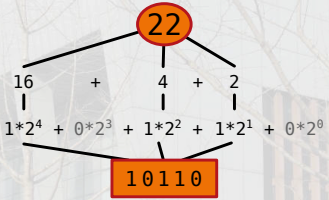
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$.



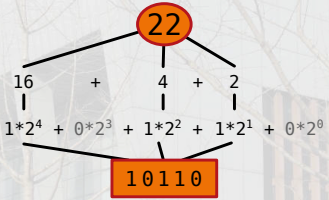
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 22 als Summe von Zweierpotenzen aus, so bekommt man $22 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$. Im Binärsystem ergibt das also 10110



Binäres Zahlensystem und Bit-weise Operatoren

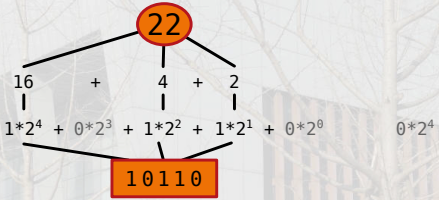
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.



15

Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

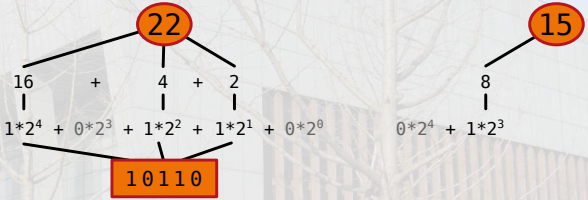


15

$0 * 2^4$

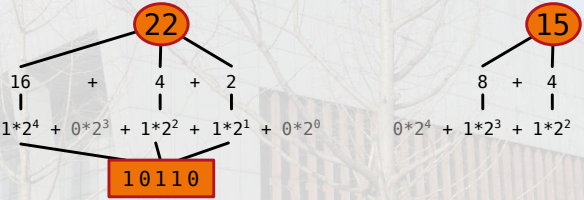
Binäres Zahlensystem und Bit-weise Operatoren

- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.



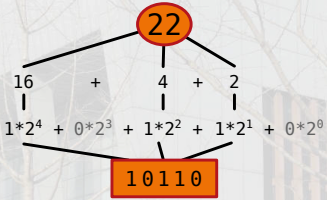
Binäres Zahlensystem und Bit-weise Operatoren

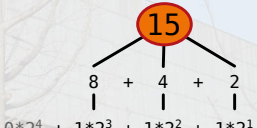
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.



Binäres Zahlensystem und Bit-weise Operatoren

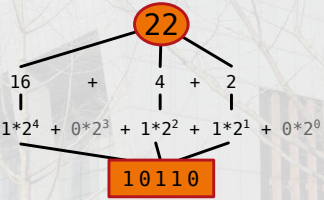
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

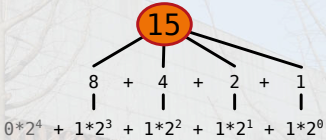




Binäres Zahlensystem und Bit-weise Operatoren

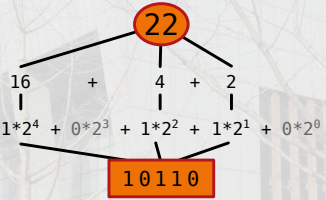
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

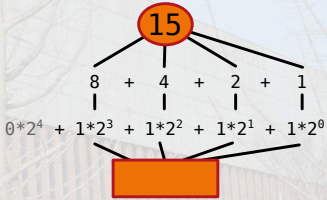




Binäres Zahlensystem und Bit-weise Operatoren

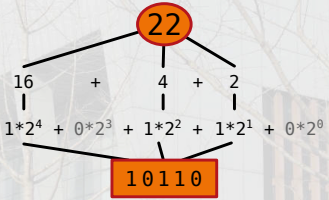
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

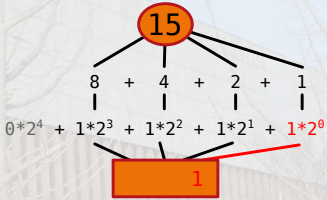




Binäres Zahlensystem und Bit-weise Operatoren

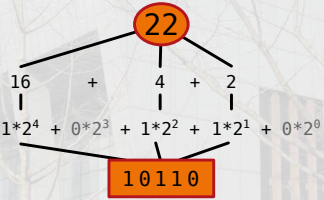
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

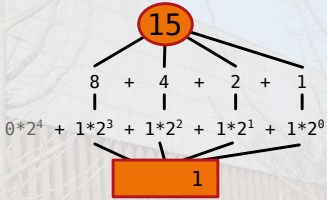




Binäres Zahlensystem und Bit-weise Operatoren

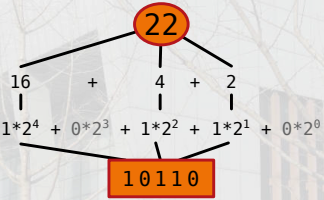
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

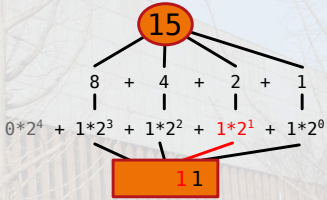




Binäres Zahlensystem und Bit-weise Operatoren

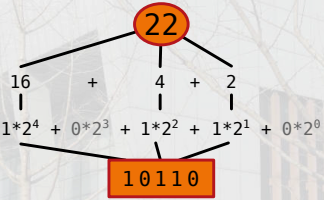
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

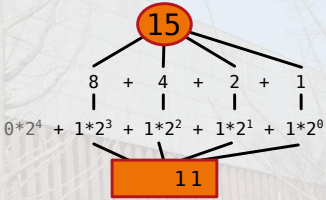




Binäres Zahlensystem und Bit-weise Operatoren

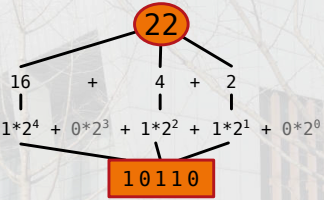
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

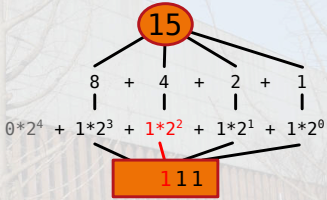




Binäres Zahlensystem und Bit-weise Operatoren

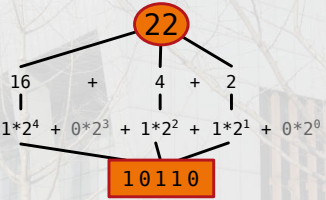
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

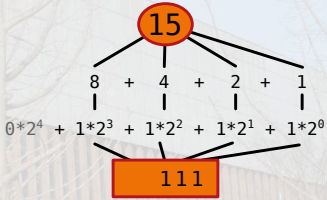




Binäres Zahlensystem und Bit-weise Operatoren

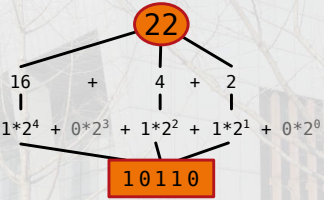
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

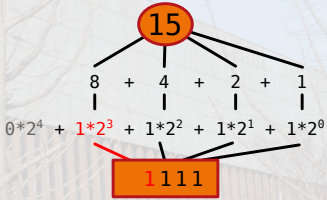




Binäres Zahlensystem und Bit-weise Operatoren

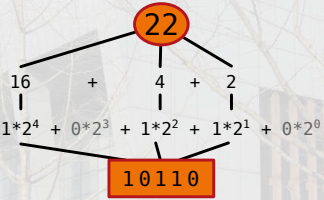
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

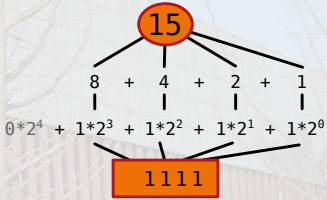




Binäres Zahlensystem und Bit-weise Operatoren

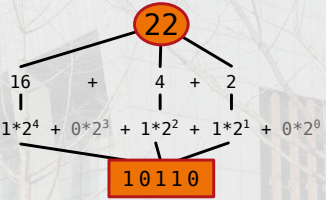
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

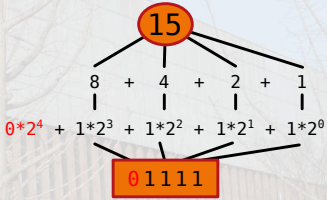




Binäres Zahlensystem und Bit-weise Operatoren

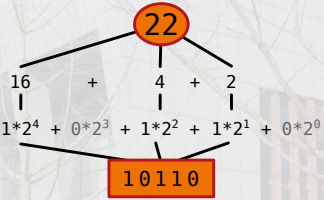
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$.

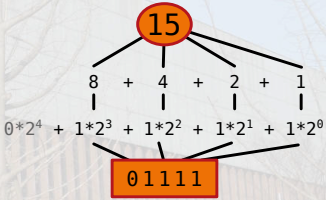




Binäres Zahlensystem und Bit-weise Operatoren

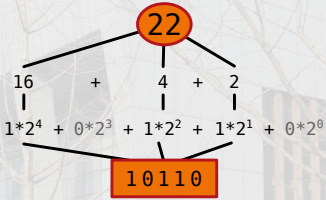
- Drückt man die Zahl 15 als Summe von Zweierpotenzen aus, so bekommt man $15 = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$. Im Binärsystem ergibt das also 01111

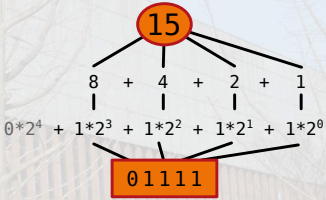




Binäres Zahlensystem und Bit-weise Operatoren

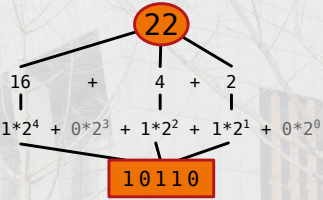
- Auf solche Bitketten können logische Operatoren Bit-weise angewandt werden.

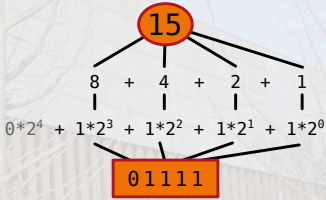




Binäres Zahlensystem und Bit-weise Operatoren

- Auf solche Bitketten können logische Operatoren Bit-weise angewandt werden.
- Werden zwei Bitketten mit bit-weisem „und“, bit-weisem „oder“, oder bit-weisem „exklusiven oder“ verbunden, dann wird der Operator jeweils auf die Bits am gleichen Index angewendet.



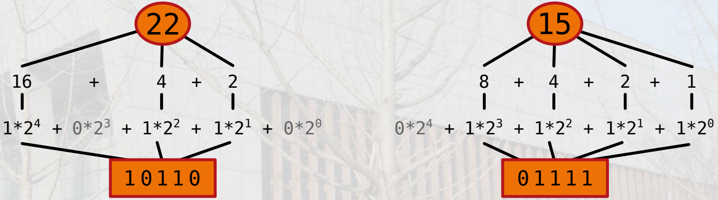


Binäres Zahlensystem und Bit-weise Operatoren

- Werden zwei Bitketten mit bit-weisem „und“, bit-weisem „oder“, oder bit-weisem „exklusiven oder“ verbunden, dann wird der Operator jeweils auf die Bits am gleichen Index angewendet.
- Schauen wir uns das mal an.

Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „oder“ in Python wird als `|` ausgedrückt.



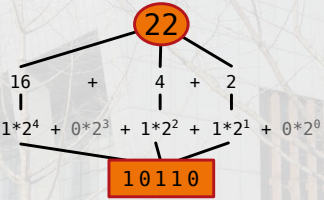
|

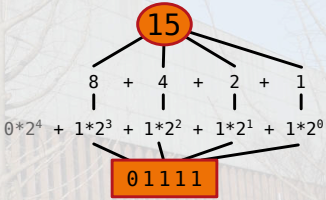
---

or

Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „oder“ in Python wird als `|` ausgedrückt. Es gilt `0|0 == 0`, `0|1 == 1`, `1|0 == 1`, und `1|1 == 1`.



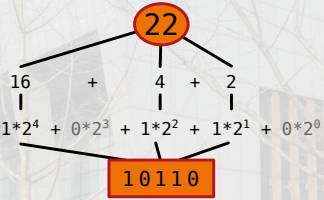


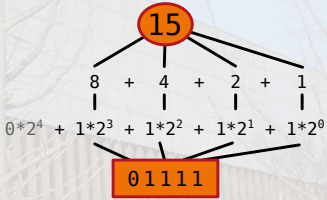
|

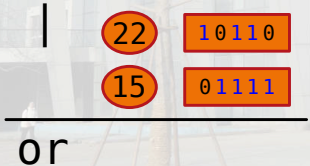
or

Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „oder“ in Python wird als `|` ausgedrückt. Es gilt `0|0 == 0`, `0|1 == 1`, `1|0 == 1`, und `1|1 == 1`.

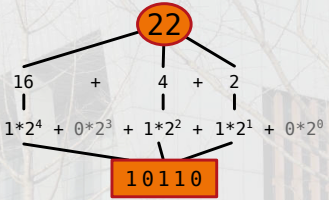


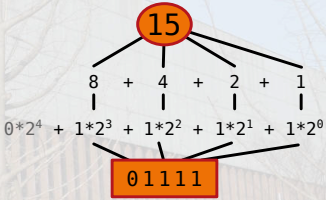


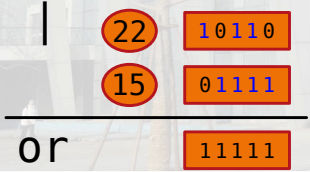


Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „oder“ in Python wird als `|` ausgedrückt. Es gilt `0|0 == 0`, `0|1 == 1`, `1|0 == 1`, und `1|1 == 1`.

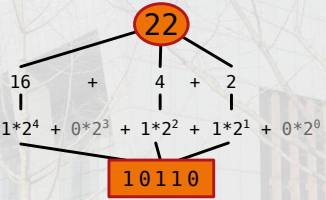


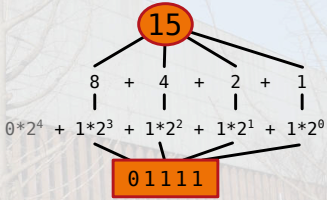


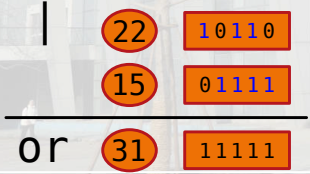


Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „oder“ in Python wird als `|` ausgedrückt. Es gilt `0|0 == 0`, `0|1 == 1`, `1|0 == 1`, und `1|1 == 1`.

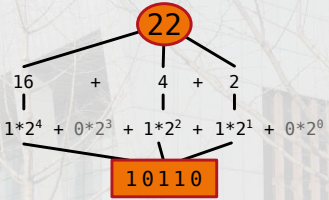


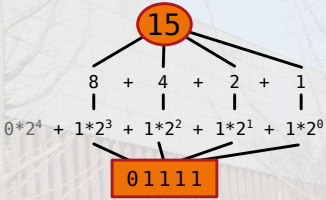


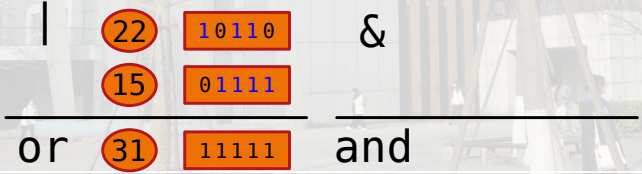


Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „und“ in Python wird als `&` ausgedrückt.

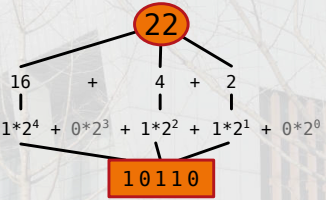


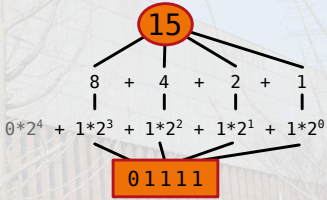


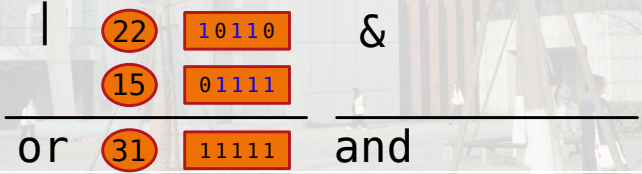


Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „und“ in Python wird als `&` ausgedrückt. Es gilt `0&0 == 0`, `0&1 == 0`, `1&0 == 0`, und `1&1 == 1`.

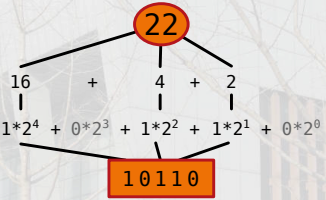


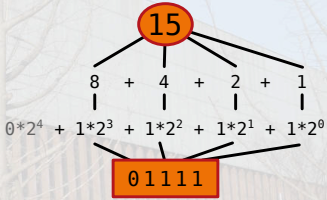


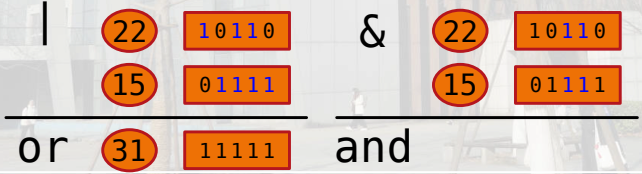


Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „und“ in Python wird als `&` ausgedrückt. Es gilt `0&0 == 0`, `0&1 == 0`, `1&0 == 0`, und `1&1 == 1`.

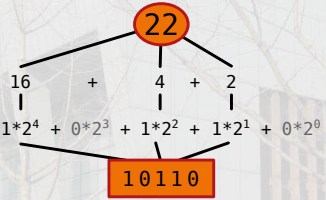


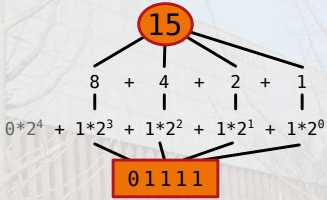


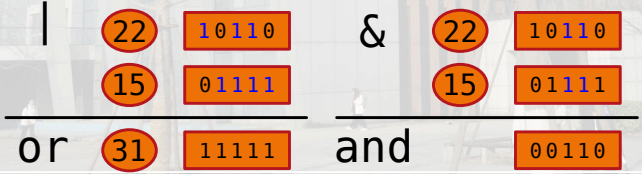


Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „und“ in Python wird als `&` ausgedrückt. Es gilt `0&0 == 0`, `0&1 == 0`, `1&0 == 0`, und `1&1 == 1`.

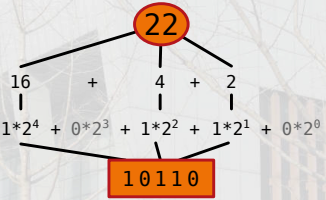


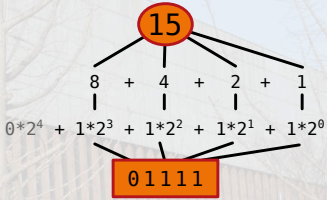




Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „und“ in Python wird als `&` ausgedrückt. Es gilt `0&0 == 0`, `0&1 == 0`, `1&0 == 0`, und `1&1 == 1`.

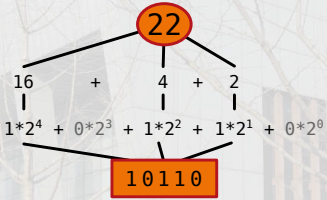


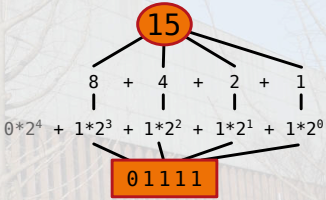


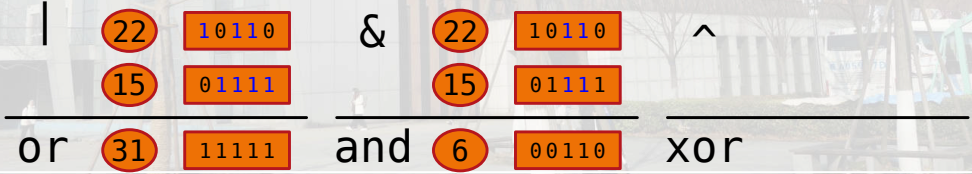
22	10110		&	22	10110	
15	01111			15	01111	
or	31	11111		and	6	00110

Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „exklusives oder“ in Python wird als ^ ausgedrückt.

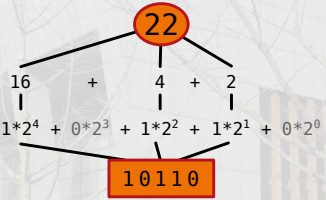


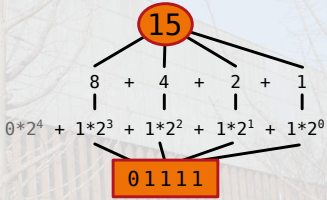


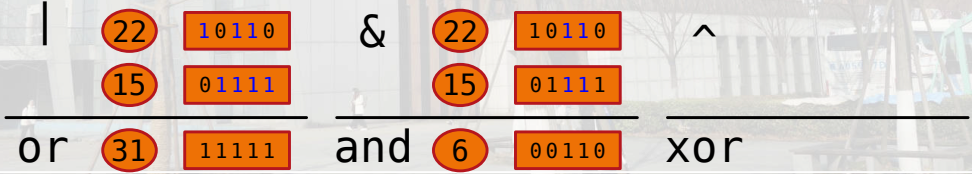


Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „exklusives oder“ in Python wird als `^` ausgedrückt. Es gilt `0^0 == 0`, `0^1 == 1`, `1^0 == 1`, und `1^1 == 0`.

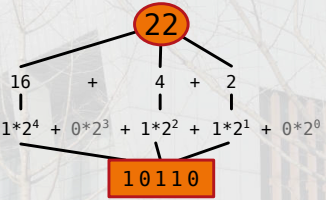


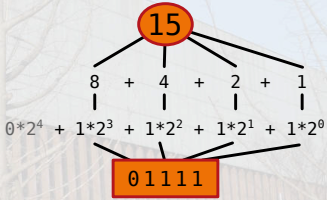




Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „exklusives oder“ in Python wird als `^` ausgedrückt. Es gilt `0^0 == 0`, `0^1 == 1`, `1^0 == 1`, und `1^1 == 0`.

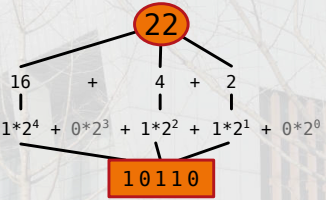


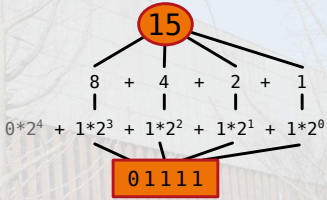


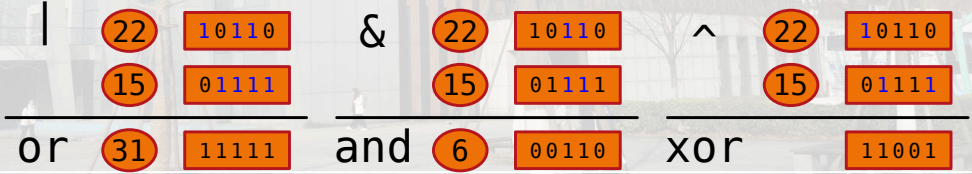
	22	10110		&	22	10110		^	22	10110
	15	01111			15	01111			15	01111
or	31	11111		and	6	00110		xor		

Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „exklusives oder“ in Python wird als `^` ausgedrückt. Es gilt `0^0 == 0`, `0^1 == 1`, `1^0 == 1`, und `1^1 == 0`.

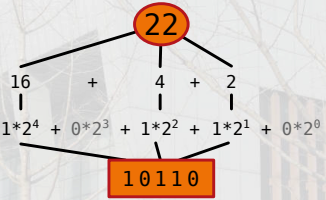


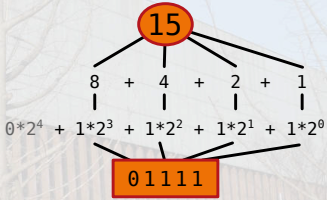


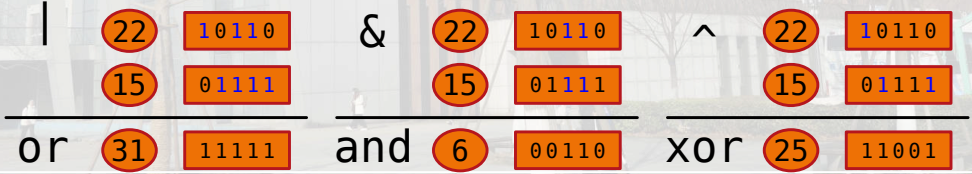


Binäres Zahlensystem und Bit-weise Operatoren

- Bit-weises „exklusives oder“ in Python wird als `^` ausgedrückt. Es gilt `0^0 == 0`, `0^1 == 1`, `1^0 == 1`, und `1^1 == 0`.







Kurzer Auffrischungskurs: Zahlensysteme

- In der Informatik sind das binäre Zahlensystem (Basis 2), das oktale Zahlensystem (Basis 8), und das Hexadezimale Zahlensystem (Basis 16) weit verbreitet und wichtig.

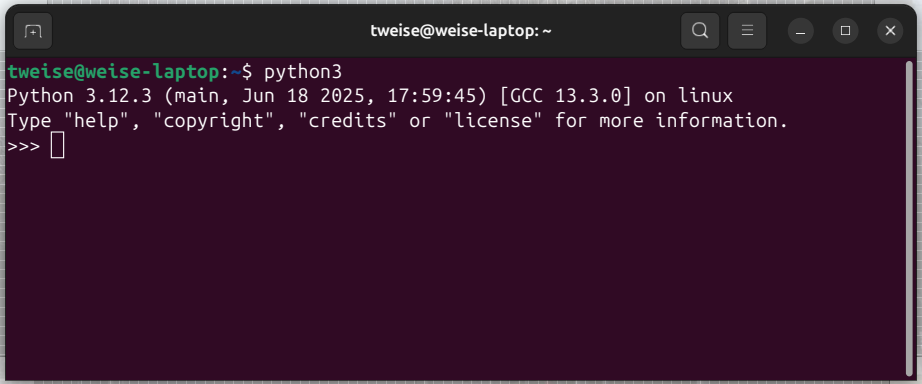
Kurzer Auffrischungskurs: Zahlensysteme

- Dezimal (dec), Binär (bin), Oktal (oct), und Hexadezimal (hex)

dec	bin								oct			hex		dec	bin								oct			hex		dec	bin								oct			hex	
10	1	64	32	16	8	4	2	1	64	8	1	16	1	10	1	64	32	16	8	4	2	1	64	8	1	16	1	10	1	64	32	16	8	4	2	1	64	8	1	16	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	1	0	2	0	0	0	0	0	1	0	0	0	2	0	2
0	3	0	0	0	0	0	1	1	0	0	3	0	3	0	4	0	0	0	0	1	0	0	0	0	4	0	4	0	5	0	0	0	0	1	0	1	0	0	5	0	5
0	6	0	0	0	0	1	1	0	0	0	6	0	6	0	7	0	0	0	0	1	1	1	0	0	7	0	7	0	8	0	0	0	1	0	0	0	0	1	0	0	8
0	9	0	0	0	1	0	0	1	0	1	1	0	9	1	0	0	0	0	1	0	1	0	0	1	2	0	a	1	1	0	0	0	1	0	1	1	0	1	3	0	b
1	2	0	0	0	1	1	0	0	0	1	4	0	c	1	3	0	0	0	1	1	0	1	0	1	5	0	d	1	4	0	0	0	1	1	1	0	0	1	6	0	e
1	5	0	0	0	1	1	1	1	0	1	7	0	f	1	6	0	0	1	0	0	0	0	0	2	0	1	0	1	7	0	0	1	0	0	0	1	0	2	1	1	1
1	8	0	0	1	0	0	1	0	0	2	2	1	2	1	9	0	0	1	0	0	1	1	0	2	3	1	3	2	0	0	0	1	0	1	0	0	0	2	4	1	4
2	1	0	0	1	0	1	0	1	0	2	5	1	5	2	2	0	0	1	0	1	1	0	0	2	6	1	6	2	3	0	0	1	0	1	1	1	0	2	7	1	7
2	4	0	0	1	1	0	0	0	0	3	0	1	8	2	5	0	0	1	1	0	0	1	0	3	1	1	9	2	6	0	0	1	1	0	1	0	0	3	2	1	a
2	7	0	0	1	1	0	1	1	0	3	3	1	b	2	8	0	0	1	1	1	0	0	0	3	4	1	c	2	9	0	0	1	1	1	0	1	0	3	5	1	d
3	0	0	0	1	1	1	1	0	0	3	6	1	e	3	1	0	0	1	1	1	1	1	0	3	7	1	f	3	2	0	1	0	0	0	0	0	0	4	0	2	0
3	3	0	1	0	0	0	0	1	0	4	1	2	1	3	4	0	1	0	0	0	1	0	0	4	2	2	2	3	5	0	1	0	0	0	1	1	0	4	3	2	3
3	6	0	1	0	0	1	0	0	0	4	4	2	4	3	7	0	1	0	0	1	0	1	0	4	5	2	5	3	8	0	1	0	0	1	1	0	0	4	6	2	6
3	9	0	1	0	0	1	1	1	0	4	7	2	7	4	0	0	1	0	1	0	0	0	0	5	0	2	8	4	1	0	1	0	1	0	0	1	0	5	1	2	9
4	2	0	1	0	1	0	1	0	0	5	2	2	a	4	3	0	1	0	1	0	1	1	0	5	3	2	b	4	4	0	1	0	1	1	0	0	0	5	4	2	c
4	5	0	1	0	1	1	0	1	0	5	5	2	d	4	6	0	1	0	1	1	1	0	0	5	6	2	e	4	7	0	1	0	1	1	1	1	0	5	7	2	f
4	8	0	1	1	0	0	0	0	0	6	0	3	0	4	9	0	1	1	0	0	0	1	0	6	1	3	1	5	0	0	1	1	0	0	1	0	0	6	2	3	2
5	1	0	1	1	0	0	1	1	0	6	3	3	3	5	2	0	1	1	0	1	0	0	0	6	4	3	4	5	3	0	1	1	0	1	0	1	0	6	5	3	5

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Schauen wir uns noch ein paar weitere Beispiele für bit-weise Operatoren und Zahlensysteme in Python an.

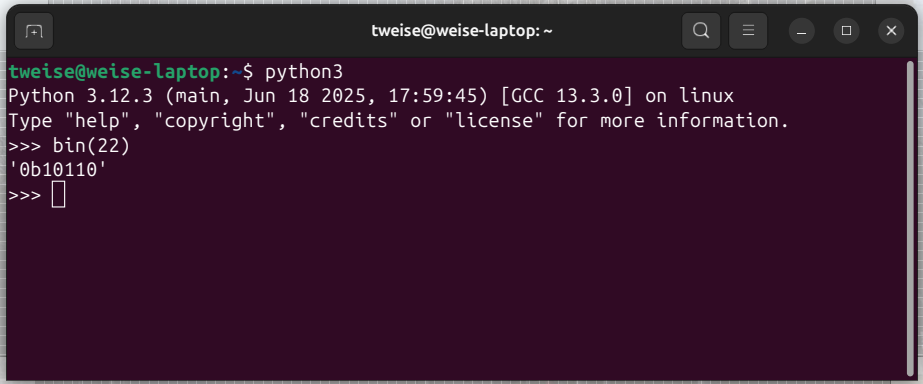


Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.

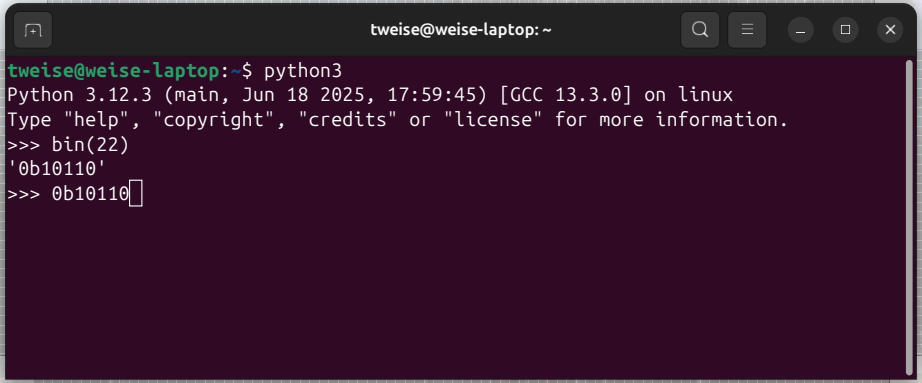
Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.



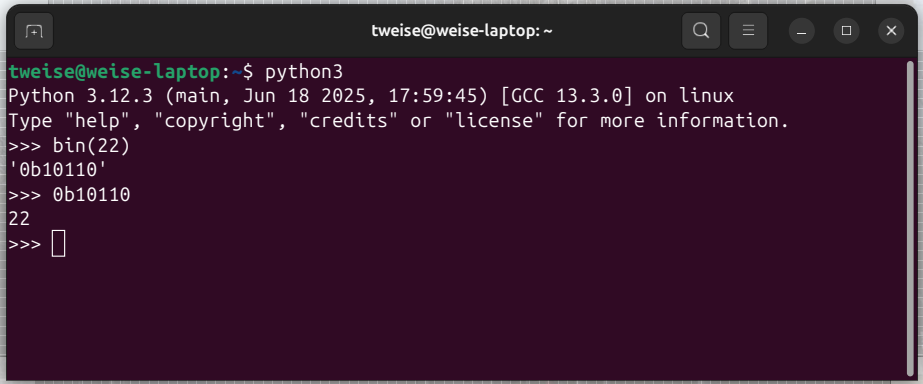
Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Eine solche Zahl – mit Präfix `0b` – kann man einfach so in Python schreiben und sie wird dann als Binärzahl intepretiert.



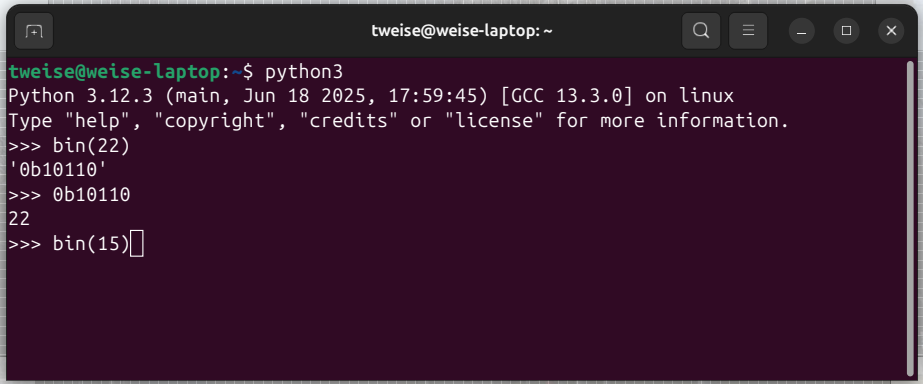
Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Eine solche Zahl – mit Präfix `0b` – kann man einfach so in Python schreiben und sie wird dann als Binärzahl intepretiert.



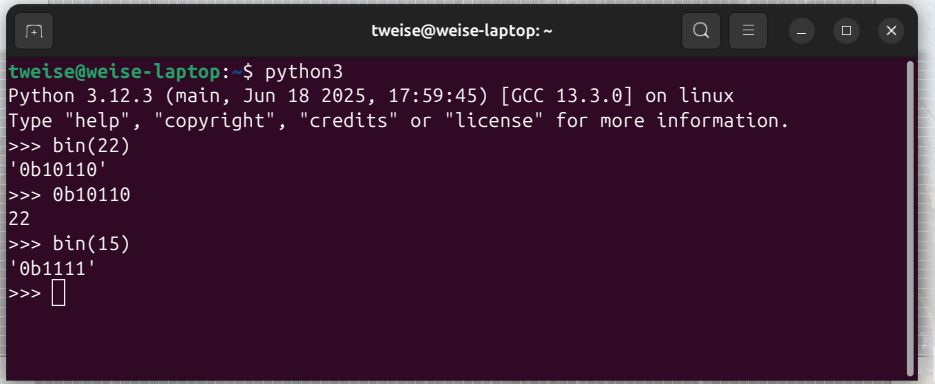
Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.



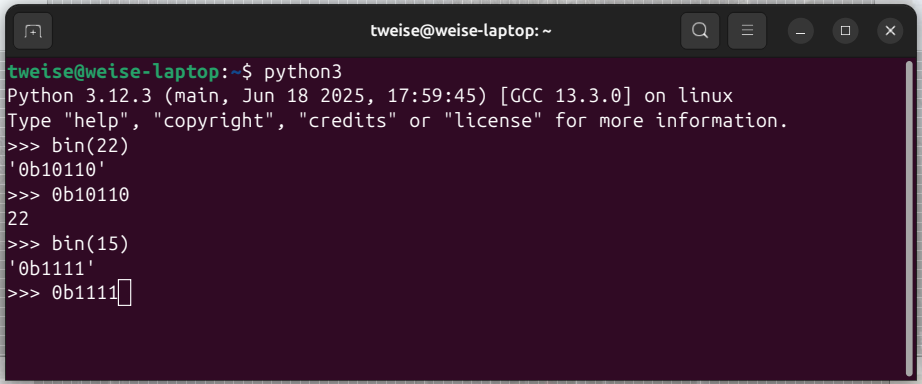
Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.



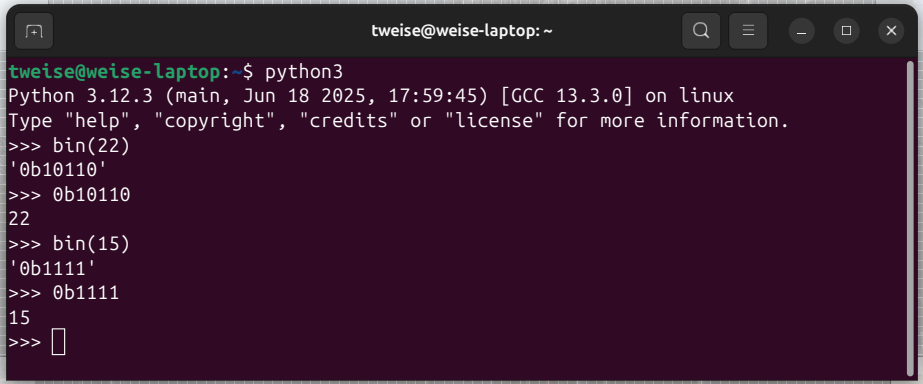
Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Eine solche Zahl – mit Präfix `0b` – kann man einfach so in Python schreiben und sie wird dann als Binärzahl intepretiert.



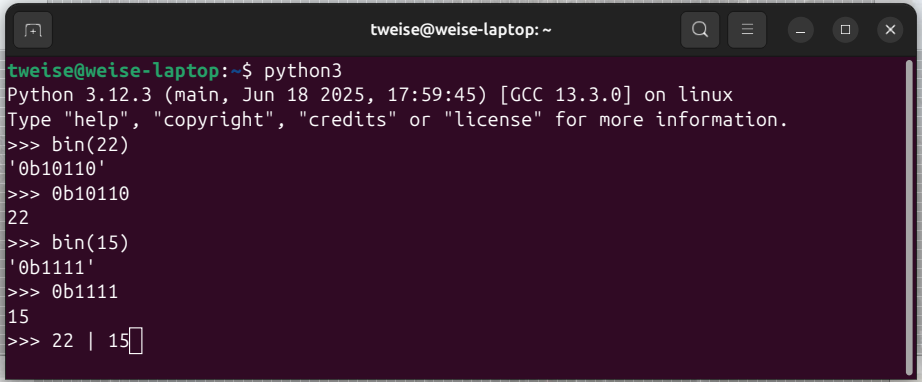
Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Eine solche Zahl – mit Präfix `0b` – kann man einfach so in Python schreiben und sie wird dann als Binärzahl intepretiert.



Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Führen wir die bit-weise „oder“-Operation `22|15 == 31` aus.



Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Führen wir die bit-weise „oder“-Operation `22|15 == 31` aus.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Führen wir die bit-weise „und“-Operation `22&15 == 6` aus.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Führen wir die bit-weise „und“-Operation `22&15 == 6` aus.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Führen wir die bit-weise „exklusive oder“-Operation `22^15 == 25` aus.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Führen wir die bit-weise „exklusive oder“-Operation `22^15 == 25` aus.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Den Bitstring `0b10110` einen Schritt nach links zu schieben, also `22<<1` zu berechnen, bedeutet ergibt `0b101100` – am rechten Ende wurde eine `0` angefügt. Das ist equivalent zu einer Multiplikation mit 2.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Den Bitstring `0b10110` einen Schritt nach links zu schieben, also `22<<1` zu berechnen, bedeutet ergibt `0b101100` – am rechten Ende wurde eine `0` angefügt. Das ist equivalent zu einer Multiplikation mit 2.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Den Bitstring `0b10110` zwei Schritte nach rechts zu schieben, also `22>>1` zu berechnen, bedeutet ergibt `0b101` – am die `10` am rechten Ende verschwinden. Das ist equivalent zu einer Ganzzahldivision durch 4.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Den Bitstring `0b10110` zwei Schritte nach rechts zu schieben, also `22>>1` zu berechnen, bedeutet ergibt `0b101` – am die `10` am rechten Ende verschwinden. Das ist equivalent zu einer Ganzzahldivision durch 4.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Die Funktion `bin(x)` wandelt die `int`-Zahl `x` in einen Text um, der den Wert als Binärzahl darstellt und das Präfix `0b` hat.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Das Hexadezimalsystem ist in der Informatik weit verbreitet. Es hat die Basis 16 und Ziffern `0123456789ABCDEF`, wodurch sich Zahlen kompakt darstellen lassen und jede Ziffer 4 Bit entspricht.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Das Hexadezimalsystem ist in der Informatik weit verbreitet. Es hat die Basis 16 und Ziffern `0123456789ABCDEF`, wodurch sich Zahlen kompakt darstellen lassen und jede Ziffer 4 Bit entspricht.
- Die Funktion `hex(x)` übersetzt die Zahl `x` zu einem Text mit den hexadezimalen Ziffern und Präfix `0`

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Wir können Zahlen auch direkt im Hexadezimalsystem angeben, wobei wieder das Präfix `0x` verwendet wird.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Wir können Zahlen auch direkt im Hexadezimalsystem angeben, wobei wieder das Präfix `0x` verwendet wird.
- Python interpretiert solche Zahlen dann als Hexadezimalzahlen und rechnet sie in entsprechende `int`-Werte um.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Das Oktalsystem ist in der Informatik weit verbreitet. Es hat die Basis 8 und Ziffern `01234567`, wodurch jede Ziffer 3 Bit entspricht.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Das Oktalsystem ist in der Informatik weit verbreitet. Es hat die Basis 8 und Ziffern `01234567`, wodurch jede Ziffer 3 Bit entspricht.
- Die Funktion `oct(x)` übersetzt die Zahl `x` zu einem Text mit den oktalen Ziffern und Präfix `0o`.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Wir können Zahlen auch direkt im Oktalsystem angeben, wobei wieder das Präfix 0o verwendet wird.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Wir können Zahlen auch direkt im Oktalsystem angeben, wobei wieder das Präfix `0o` verwendet wird.
- Python interpretiert solche Zahlen dann als Oktalzahlen und rechnet sie in entsprechende `int`-Werte um.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Sie werden sich nun vielleicht fragen: „Woher weiß ein `int`, dass es im hexadezimalen, dezimalen, binären, oder oktalen Format eingegeben wurde?“

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Sie werden sich nun vielleicht fragen: „Woher weiß ein `int`, dass es im hexadezimalen, dezimalen, binären, oder oktalen Format eingegeben wurde?“
- Das weiß es eben nicht!

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Sie werden sich nun vielleicht fragen: „Woher weiß ein `int`, dass es im hexadezimalen, dezimalen, binären, oder oktalen Format eingegeben wurde?“
- Das weiß es eben nicht!
- Diese Formate sind nur Textformate für die Ein- und Ausgabe von `int`-Werten.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Sie werden sich nun vielleicht fragen: „Woher weiß ein `int`, dass es im hexadezimalen, dezimalen, binären, oder oktalen Format eingegeben wurde?“
- Das weiß es eben nicht!
- Diese Formate sind nur Textformate für die Ein- und Ausgabe von `int`-Werten.
- Sie können diese in Ihrem Programmkode verwenden, im Python-Interpreter, or in der Eingabe Ihrer Programme.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Sie werden sich nun vielleicht fragen: „Woher weiß ein `int`, dass es im hexadezimalen, dezimalen, binären, oder oktalen Format eingegeben wurde?“
- Das weiß es eben nicht!
- Diese Formate sind nur Textformate für die Ein- und Ausgabe von `int`-Werten.
- Sie können diese in Ihrem Programmkode verwenden, im Python-Interpreter, or in der Eingabe Ihrer Programme.
- Sie werden alle in die selbe `int` Struktur umgerechnet.

Weitere Beispiele für Binärarithmetik und Zahlensysteme

- Sie werden sich nun vielleicht fragen: „Woher weiß ein `int`, dass es im hexadezimalen, dezimalen, binären, oder oktalen Format eingegeben wurde?“
- Das weiß es eben nicht!
- Diese Formate sind nur Textformate für die Ein- und Ausgabe von `int`-Werten.
- Sie können diese in Ihrem Programmkode verwenden, im Python-Interpreter, or in der Eingabe Ihrer Programme.
- Sie werden alle in die selbe `int` Struktur umgerechnet.
- Und danach spielt es gar keine Rolle mehr, in welchem Format die ursprünglich spezifiziert waren.

Zusammenfassung

Zusammenfassung

- Damit haben wir unseren ersten Datentyp in Python kennengelernt.

Zusammenfassung

- Damit haben wir unseren ersten Datentyp in Python kennengelernt.
- `int` repräsentiert Ganzzahlen, die positiv oder negativ seien können und 0 einschließen.

Zusammenfassung

- Damit haben wir unseren ersten Datentyp in Python kennengelernt.
- `int` repräsentiert Ganzzahlen, die positiv oder negativ seien können und 0 einschließen.
- Wir können mit diesen Zahlen normal rechnen, wobei Python die Operatorenrangfolge mathematisch korrekt beachtet.

Zusammenfassung

- Damit haben wir unseren ersten Datentyp in Python kennengelernt.
- `int` repräsentiert Ganzzahlen, die positiv oder negativ seien können und 0 einschließen.
- Wir können mit diesen Zahlen normal rechnen, wobei Python die Operatorenrangfolge mathematisch korrekt beachtet.
- Zu beachten sind besonders die Divisionsoperatoren `/` und `//`

Zusammenfassung

- Damit haben wir unseren ersten Datentyp in Python kennengelernt.
- `int` repräsentiert Ganzzahlen, die positiv oder negativ seien können und 0 einschließen.
- Wir können mit diesen Zahlen normal rechnen, wobei Python die Operatorenrangfolge mathematisch korrekt beachtet.
- Zu beachten sind besonders die Divisionsoperatoren `/` und `//`: `//` führt eine Ganzzahldivision durch, wobei Nachkommastellen wegfallen.

Zusammenfassung

- Damit haben wir unseren ersten Datentyp in Python kennengelernt.
- `int` repräsentiert Ganzzahlen, die positiv oder negativ seien können und 0 einschließen.
- Wir können mit diesen Zahlen normal rechnen, wobei Python die Operatorenrangfolge mathematisch korrekt beachtet.
- Zu beachten sind besonders die Divisionsoperatoren `/` und `//`: `//` führt eine Ganzzahldivision durch, wobei Nachkommastellen wegfallen. `/` hingegen liefert immer Fließkommazahlen zurück (lernen wir später) und niemals `int`s[33].

Zusammenfassung

- Damit haben wir unseren ersten Datentyp in Python kennengelernt.
- `int` repräsentiert Ganzzahlen, die positiv oder negativ seien können und 0 einschließen.
- Wir können mit diesen Zahlen normal rechnen, wobei Python die Operatorenrangfolge mathematisch korrekt beachtet.
- Zu beachten sind besonders die Divisionsoperatoren `/` und `//`: `//` führt eine Ganzzahldivision durch, wobei Nachkommastellen wegfallen. `/` hingegen liefert immer Fließkommazahlen zurück (lernen wir später) und niemals `int`s[33].
- Wir können auch mit der binären Repräsentation dieser Zahlen rechnen und sie in verschiedene Zahlensysteme, die Sie sicher aus Grundlagenvorlesungen kennen, umrechnen.

Zusammenfassung

- Damit haben wir unseren ersten Datentyp in Python kennengelernt.
- `int` repräsentiert Ganzzahlen, die positiv oder negativ seien können und 0 einschließen.
- Wir können mit diesen Zahlen normal rechnen, wobei Python die Operatorenrangfolge mathematisch korrekt beachtet.
- Zu beachten sind besonders die Divisionsoperatoren `/` und `//`: `//` führt eine Ganzzahldivision durch, wobei Nachkommastellen wegfallen. `/` hingegen liefert immer Fließkommazahlen zurück (lernen wir später) und niemals `int`s[33].
- Wir können auch mit der binären Repräsentation dieser Zahlen rechnen und sie in verschiedene Zahlensysteme, die Sie sicher aus Grundlagenvorlesungen kennen, umrechnen.
- Und wieder sind wir einen Schritt weiter.

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. 232, 233).

[2] Kent L. Beck. *JUnit Pocket Guide*. Sebastopol, CA, USA: O'Reilly Media, Inc., Sep. 2004. ISBN: 978-0-596-00743-0 (siehe S. 233).

[3] Joshua Bloch. *Effective Java*. Reading, MA, USA: Addison-Wesley Professional, Mai 2008. ISBN: 978-0-321-35668-0 (siehe S. 232).

[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. 232).

[5] Ron Brash und Ganesh Naik. *Bash Cookbook*. Birmingham, England, UK: Packt Publishing Ltd, Juli 2018. ISBN: 978-1-78862-936-2 (siehe S. 232).

[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. 232).

[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. 233).

[8] Slobodan Dmitrović. *Modern C for Absolute Beginners: A Friendly Introduction to the C Programming Language*. New York, NY, USA: Apress Media, LLC, März 2024. ISBN: 979-8-8688-0224-9 (siehe S. 232).

[9] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (siehe S. 232).

[10] 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. 233).

[11] 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. 233).

References II

[12] ."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. 13–21).

[13] 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. 233).

[14] Marc Loy, Patrick Niemeyer und Daniel Leuck. *Learning Java*. 5. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2020. ISBN: 978-1-4920-5627-0 (siehe S. 232).

[15] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 233).

[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. 232).

[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. 233).

[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. 233).

[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. 233).

[20] *Programming Languages – C, Working Document of SC22/WG14*. International Standard ISO/IEC9899:2017 C17 Ballot N2176. Geneva, Switzerland: International Organization for Standardization (ISO) und International Electrotechnical Commission (IEC), Nov. 2017. URL: https://files.lhmouse.com/standards/ISO%20C%20N2176.pdf (besucht am 2024-06-29) (siehe S. 30–39, 232).

References III

[21] 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. 232).

[22] 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. 233).

[23] 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. 233).

[24] 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. 232).

[25] 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. 232).

[26] 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. 233).

[27] 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. 232).

[28] 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. 233).

[29] Sander van Vugt. *Linux Fundamentals*. 2. Aufl. Hoboken, NJ, USA: Pearson IT Certification, Juni 2022. ISBN: 978-0-13-792931-3 (siehe S. 232).

[30] 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. 233).

[31] Kevin Wilson. *Python Made Easy*. Birmingham, England, UK: Packt Publishing Ltd, Aug. 2024. ISBN: 978-1-83664-615-0 (siehe S. 233).

References IV

[32] Martin Yanev. *PyCharm Productivity and Debugging Techniques*. Birmingham, England, UK: Packt Publishing Ltd, Okt. 2022. ISBN: 978-1-83763-244-2 (siehe S. 233).

[33] Moshe Zadka und Guido van Rossum. *Changing the Division Operator*. Python Enhancement Proposal (PEP) 238. Beaverton, OR, USA: Python Software Foundation (PSF), 11. März–27. Juli 2001. URL: https://peps.python.org/pep-0238 (besucht am 2025-07-28) (siehe S. 40–91, 219–226).

[34] Giorgio Zarrelli. *Mastering Bash*. Birmingham, England, UK: Packt Publishing Ltd, Juni 2017. ISBN: 978-1-78439-687-9 (siehe S. 232).

Glossary (in English) I

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[5,16,34]. Learn more at https://www.gnu.org/software/bash.

C is a programming language, which is very successful in system programming situations[8,20].

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[6,25]. Learn more at https://www.apple.com/ios.

iPadOS is the operating system that powers Apple iPads[6]. Learn more at https://www.apple.com/ipados.

IT information technology

Java is another very successful programming language, with roots in the C family of languages[3,14].

Linux is the leading open source operating system, i.e., a free alternative for Microsoft Windows[1,9,24,27,29]. 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[21,25]. Learn more at https://www.apple.com/macos.

Microsoft Windows is a commercial proprietary operating system[4]. 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.

modulo division is, in Python, done by the operator `%` that computes the remainder of a division. `15 % 6` gives us `3`.

Glossary (in English) II

PyCharm is the convenient Python IDE that we recommend for this course[28,31,32]. 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[11,13,15,30], i.e., what you will learn about in our book[30]. Learn more at https://python.org.

terminal A terminal is a text-based window where you can enter commands and execute them[1,7]. 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 ⊞+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[7,10]. 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.

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[2,17–19,22,26]. 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 iOS[23].

Glossary (in English) III

$i..j$ with $i, j \in \mathbb{Z}$ and $i \leq j$ is the set that contains all integer numbers in the inclusive range from i to j. For example, $5..9$ is equivalent to $\{5, 6, 7, 8, 9\}$

$\mathbb{R}$ the set of the real numbers.

$\mathbb{Z}$ the set of the integers numbers including positive and negative numbers and 0, i.e., . . . , -3, -2, -1, 0, 1, 2, 3, . . . , and so on. It holds that $\mathbb{Z} \subset \mathbb{R}$.