



合肥大學  
HEFEI UNIVERSITY



# Programming with Python

## 12. None

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

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

应用优化研究所  
人工智能与大数据学院  
合肥大学  
中国安徽省合肥市

# Programming with Python



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

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



# Outline



1. Einleitung
2. Ausprobieren
3. Zusammenfassung





# Einleitung



# Einleitung



- Der letzte einfache Datentyp, den wir besprechen werden, ist `NoneType` and und sein einziger Wert, `None`.



# Einleitung



- Der letzte einfache Datentyp, den wir besprechen werden, ist `NoneType` and und sein einziger Wert, `None`.
- Wir kennen bereits den Datentyp `bool`, der nur zwei Werte annehmen kann, `True` und `False`.





# Einleitung



- Der letzte einfache Datentyp, den wir besprechen werden, ist `NoneType` and und sein einziger Wert, `None`.
- Wir kennen bereits den Datentyp `bool`, der nur zwei Werte annehmen kann, `True` und `False`.
- Wir haben auch gelernt, dass der Datentyp `float` einen besonderen Wert hat, nämlich „Not a Number“, welcher als `nan` geschrieben wird.

# Einleitung



- Der letzte einfache Datentyp, den wir besprechen werden, ist `NoneType` and und sein einziger Wert, `None`.
- Wir kennen bereits den Datentyp `bool`, der nur zwei Werte annehmen kann, `True` und `False`.
- Wir haben auch gelernt, dass der Datentyp `float` einen besonderen Wert hat, nämlich „Not a Number“, welcher als `nan` geschrieben wird.
- `None` wird in Situationen genutzt, in denen wir spezifizieren wollen, dass etwas keinen Wert hat.



# Einleitung



- Der letzte einfache Datentyp, den wir besprechen werden, ist `NoneType` and und sein einziger Wert, `None`.
- Wir kennen bereits den Datentyp `bool`, der nur zwei Werte annehmen kann, `True` und `False`.
- Wir haben auch gelernt, dass der Datentyp `float` einen besonderen Wert hat, nämlich „Not a Number“, welcher als `nan` geschrieben wird.
- `None` wird in Situationen genutzt, in denen wir spezifizieren wollen, dass etwas keinen Wert hat.
- Es ist kein `int`, `float`, `str` oder `bool`.

# Einleitung



- Der letzte einfache Datentyp, den wir besprechen werden, ist `NoneType` and und sein einziger Wert, `None`.
- Wir kennen bereits den Datentyp `bool`, der nur zwei Werte annehmen kann, `True` und `False`.
- Wir haben auch gelernt, dass der Datentyp `float` einen besonderen Wert hat, nämlich „Not a Number“, welcher als `nan` geschrieben wird.
- `None` wird in Situationen genutzt, in denen wir spezifizieren wollen, dass etwas keinen Wert hat.
- Es ist kein `int`, `float`, `str` oder `bool`.
- `None` ist nicht das selbe wie `0`, es ist nicht das selbe wie `nan`, und es entspricht auch nicht dem leeren String `""`.

# Einleitung



- Der letzte einfache Datentyp, den wir besprechen werden, ist `NoneType` and und sein einziger Wert, `None`.
- Wir kennen bereits den Datentyp `bool`, der nur zwei Werte annehmen kann, `True` und `False`.
- Wir haben auch gelernt, dass der Datentyp `float` einen besonderen Wert hat, nämlich „Not a Number“, welcher als `nan` geschrieben wird.
- `None` wird in Situationen genutzt, in denen wir spezifizieren wollen, dass etwas keinen Wert hat.
- Es ist kein `int`, `float`, `str` oder `bool`.
- `None` ist nicht das selbe wie `0`, es ist nicht das selbe wie `nan`, und es entspricht auch nicht dem leeren String `""`.
- Es ist einfach `nichts`.



Ausprobieren



# Probieren wir das mal aus

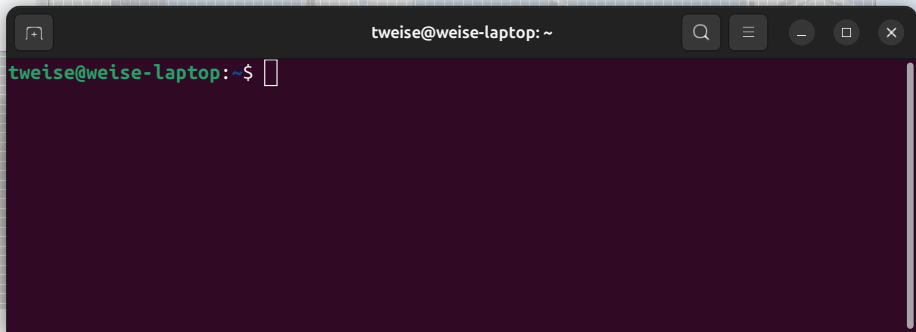
- Probieren wir das mal aus.



# Probieren wir das mal aus



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

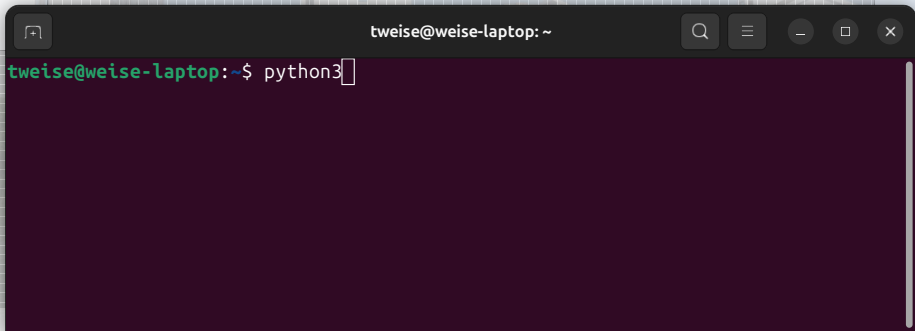




# Probieren wir das mal aus



- Wir schreiben `python3` und drücken .



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



# Probieren wir das mal aus

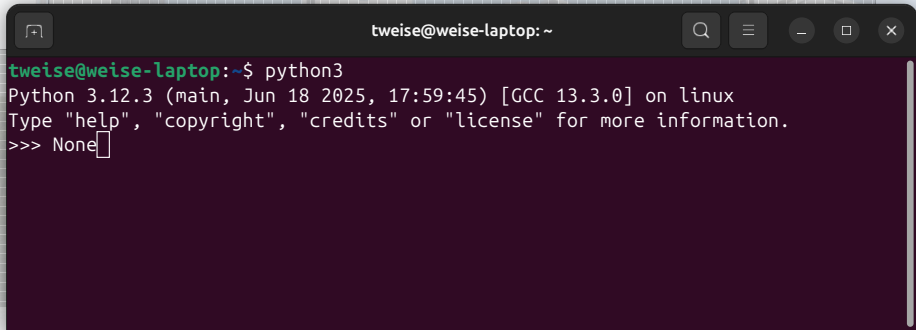


- Der Python-Interpreter startet.

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



# Probieren wir das mal aus

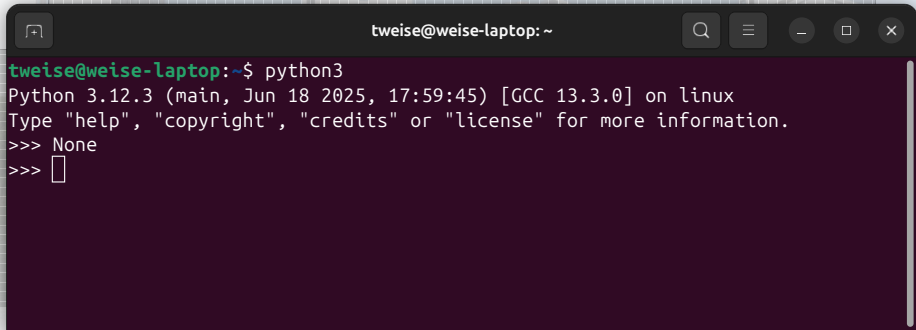
- Normalerweise, wenn wir einen Wert (z. B. `3`) in die Python-Konsole schreiben und  drücken, dann wird der Wert uns wieder ausgegeben. Schreiben wir dagegen `None` in die Python-Konsole und drücken , dann...



```
twaise@weise-laptop: ~  
twaise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> None
```

# Probieren wir das mal aus

- Normalerweise, wenn wir einen Wert (z. B. `3`) in die Python-Konsole schreiben und  drücken, dann wird der Wert uns wieder ausgegeben. Schreiben wir dagegen `None` in die Python-Konsole und drücken , dann passiert gar nichts.



```
tweise@weise-laptop: ~  
tweise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> None  
>>> 
```

# Probieren wir das mal aus



- Wollen wir `None` ausgeben, dann müssen wir explizit `print(None)` schreiben.

```
tweise@weise-laptop: ~  
twaise@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> None  
>>> print(None)
```

# Probieren wir das mal aus



- Wollen wir `None` ausgeben, dann müssen wir explizit `print(None)` schreiben. Dann wird es wirklich ausgegeben.

```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> None  
>>> print(None)  
None  
>>> 
```



# Probieren wir das mal aus



- Lassen Sie uns jetzt prüfen, ob etwas `None` ist oder nicht. Normalerweise würden wir dafür den `==` Operator verwenden, aber das soll man nicht, wenn `None` vorkommen kann<sup>11</sup>.

```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> None  
>>> print(None)  
None  
>>> 1 == None
```

# Probieren wir das mal aus



- Lassen Sie uns jetzt prüfen, ob etwas `None` ist oder nicht. Normalerweise würden wir dafür den `==` Operator verwenden, aber das soll man nicht, wenn `None` vorkommen kann<sup>11</sup>. Wir machen es trotzdem. Nur spaßeshalber. Und es funktioniert trotzdem wie erwartet.

```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> None  
>>> print(None)  
None  
>>> 1 == None  
False  
>>> □
```

# Probieren wir das mal aus



- Ist ein String gleich `None`?

```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> None  
>>> print(None)  
None  
>>> 1 == None  
False  
>>> "Hello World!" == None
```

# Probieren wir das mal aus



- Ist ein String gleich `None`? Nein.

```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> None  
>>> print(None)  
None  
>>> 1 == None  
False  
>>> "Hello World!" == None  
False  
>>> 
```

# Probieren wir das mal aus



- Ist `None` gleich `None`?

```
tweise@weise-laptop: ~  
twiese@weise-laptop:~$ python3  
Python 3.12.3 (main, Jun 18 2025, 17:59:45) [GCC 13.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> None  
>>> print(None)  
None  
>>> 1 == None  
False  
>>> "Hello World!" == None  
False  
>>> None == None
```

# Probieren wir das mal aus



- Ist `None` gleich `None`? Ja. Das ist interessant, weil wir ja wissen, dass `nan == nan` `False` ergibt. Aber `nan` ist ja auch „undefiniert“ und `None` ist „Nichts“.

```
tweise@weise-laptop: ~  
Type "help", "copyright", "credits" or "license" for more information.  
>>> None  
>>> print(None)  
None  
>>> 1 == None  
False  
>>> "Hello World!" == None  
False  
>>> None == None  
True  
>>> 
```



# Probieren wir das mal aus



- Der Operator `is` prüft die Identität von Objekten: `a is b` ist `True`, wenn `a` und `b` das selbe Objekt sind (nicht nur das gleiche). Testen wir mal `1 is None`.

```
tweise@weise-laptop: ~  
Type "help", "copyright", "credits" or "license" for more information.  
>>> None  
>>> print(None)  
None  
>>> 1 == None  
False  
>>> "Hello World!" == None  
False  
>>> None == None  
True  
>>> 1 is None
```

# Probieren wir das mal aus



- Der Operator `is` prüft die Identität von Objekten: `a is b` ist `True`, wenn `a` und `b` das selbe Objekt sind (nicht nur das gleiche). Testen wir mal `1 is None`. Das stimmt natürlich nicht. (Und wir bekommen auch eine Warnung, dass die Frage an sich schon sinnlos ist.)

```
tweise@weise-laptop: ~  
None  
>>> 1 == None  
False  
>>> "Hello World!" == None  
False  
>>> None == None  
True  
>>> 1 is None  
<stdin>:1: SyntaxWarning: "is" with 'int' literal. Did you mean "=="?  
False  
>>> 
```

# Probieren wir das mal aus



- Der Operator `is` prüft die Identität von Objekten: `a is b` ist `True`, wenn `a` und `b` das selbe Objekt sind (nicht nur das gleiche). Testen wir mal `"Hello World!" is None`.

```
tweise@weise-laptop: ~  
None  
>>> 1 == None  
False  
>>> "Hello World!" == None  
False  
>>> None == None  
True  
>>> 1 is None  
<stdin>:1: SyntaxWarning: "is" with 'int' literal. Did you mean "=="?  
False  
>>> "Hello World!" is None
```

# Probieren wir das mal aus



- Der Operator `is` prüft die Identität von Objekten: `a is b` ist `True`, wenn `a` und `b` das selbe Objekt sind (nicht nur das gleiche). Testen wir mal `"Hello World!" is None`. Das stimmt natürlich nicht. (Und wir bekommen auch eine Warnung, dass die Frage an sich schon sinnlos ist.)

```
tweise@weise-laptop: ~  
>>> "Hello World!" == None  
False  
>>> None == None  
True  
>>> 1 is None  
<stdin>:1: SyntaxWarning: "is" with 'int' literal. Did you mean "=="?  
False  
>>> "Hello World!" is None  
<stdin>:1: SyntaxWarning: "is" with 'str' literal. Did you mean "=="?  
False  
>>> 
```

# Probieren wir das mal aus



- Der Operator `is` prüft die Identität von Objekten: `a is b` ist `True`, wenn `a` und `b` das selbe Objekt sind (nicht nur das gleiche). Testen wir nun `None is None`.

```
tweise@weise-laptop: ~  
>>> "Hello World!" == None  
False  
>>> None == None  
True  
>>> 1 is None  
<stdin>:1: SyntaxWarning: "is" with 'int' literal. Did you mean "=="?  
False  
>>> "Hello World!" is None  
<stdin>:1: SyntaxWarning: "is" with 'str' literal. Did you mean "=="?  
False  
>>> None is None
```

# Probieren wir das mal aus



- Der Operator `is` prüft die Identität von Objekten: `a is b` ist `True`, wenn `a` und `b` das selbe Objekt sind (nicht nur das gleiche). Testen wir nun `None is None`. Das stimmt natürlich.

```
tweise@weise-laptop: ~  
>>> None == None  
True  
>>> 1 is None  
<stdin>:1: SyntaxWarning: "is" with 'int' literal. Did you mean "=="?  
False  
>>> "Hello World!" is None  
<stdin>:1: SyntaxWarning: "is" with 'str' literal. Did you mean "=="?  
False  
>>> None is None  
True  
>>> 
```



# Probieren wir das mal aus



- Der Operator `is` prüft die Identität von Objekten: `a is b` ist `True`, wenn `a` und `b` das selbe Objekt sind (nicht nur das gleiche). Testen wir mal `1 is None`. Das stimmt natürlich nicht. (Und wir bekommen auch eine Warnung, dass die Frage an sich schon sinnlos ist.)
- Testen wir mal `"Hello World!" is None`. Das stimmt natürlich nicht. (Und wir bekommen auch eine Warnung, dass die Frage an sich schon sinnlos ist.)
- Testen wir nun `None is None`. Das stimmt natürlich.

## Gute Praxis

Vergleiche mit Singletons wie `None` müssen immer mit dem `is` oder dem `is not` Operator gemacht werden, niemals mit Gleichheitsoperatoren wie `Python==` oder `!=`<sup>11</sup>.

# Probieren wir das mal aus



- Wir haben bereits viele Funktionen in Python kennengelernt, die Werte zurückliefern. So gibt uns `sin` z. B. einen `float`-Wert zurück. Was aber liefern Funktionen wie `print` zurück, die keinen Rückgabewert haben?

```
tweise@weise-laptop: ~  
>>> None == None  
True  
>>> 1 is None  
<stdin>:1: SyntaxWarning: "is" with 'int' literal. Did you mean "=="?  
False  
>>> "Hello World!" is None  
<stdin>:1: SyntaxWarning: "is" with 'str' literal. Did you mean "=="?  
False  
>>> None is None  
True  
>>> print(print("Hello World!"))
```

# Probieren wir das mal aus



- Wir haben bereits viele Funktionen in Python kennengelernt, die Werte zurückliefern. So gibt uns `sin` z. B. einen `float`-Wert zurück. Was aber liefern Funktionen wie `print` zurück, die keinen Rückgabewert haben? `None`. Die liefern `None` zurück, weil `None` nämlich „Nichts“ ist.

```
tweise@weise-laptop: ~  
<stdin>:1: SyntaxWarning: "is" with 'int' literal. Did you mean "=="?  
False  
>>> "Hello World!" is None  
<stdin>:1: SyntaxWarning: "is" with 'str' literal. Did you mean "=="?  
False  
>>> None is None  
True  
>>> print(print("Hello World!"))  
Hello World!  
None  
>>> 
```

# Probieren wir das mal aus



- Was ist der Datentyp von `None`?

```
tweise@weise-laptop: ~  
<stdin>:1: SyntaxWarning: "is" with 'int' literal. Did you mean "=="?  
False  
>>> "Hello World!" is None  
<stdin>:1: SyntaxWarning: "is" with 'str' literal. Did you mean "=="?  
False  
>>> None is None  
True  
>>> print(print("Hello World!"))  
Hello World!  
None  
>>> type(None) 
```

# Probieren wir das mal aus



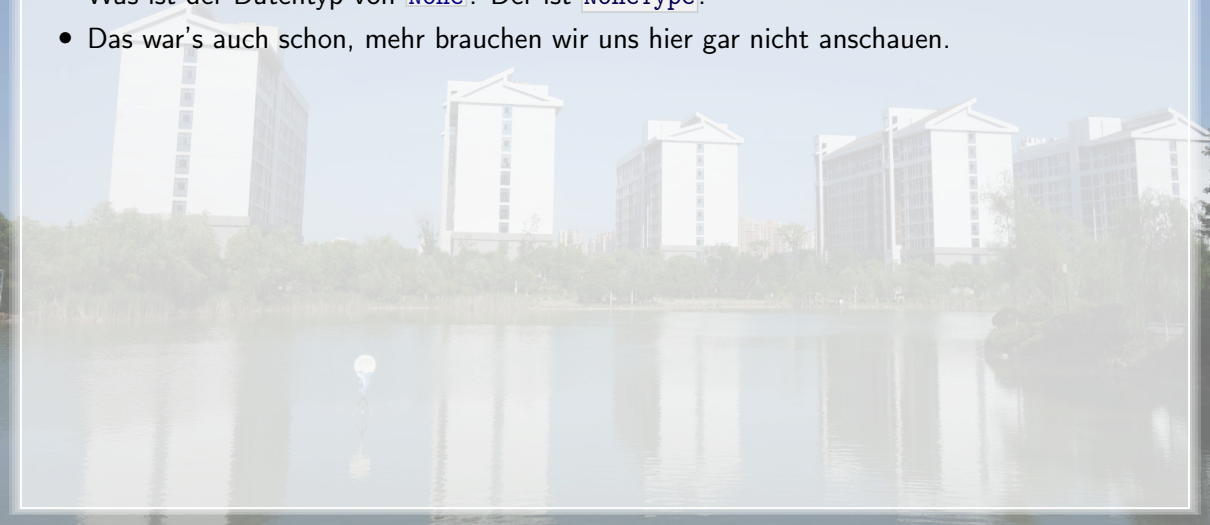
- Was ist der Datentyp von `None`? Der ist `NoneType`.

```
tweise@weise-laptop: ~  
>>> "Hello World!" is None  
<stdin>:1: SyntaxWarning: "is" with 'str' literal. Did you mean "=="?  
False  
>>> None is None  
True  
>>> print(print("Hello World!"))  
Hello World!  
None  
>>> type(None)  
<class 'NoneType'>  
>>> 
```

# Probieren wir das mal aus



- Probieren wir das mal aus.
- Was ist der Datentyp von `None`? Der ist `NoneType`.
- Das war's auch schon, mehr brauchen wir uns hier gar nicht anschauen.



# Probieren wir das mal aus



- Probieren wir das mal aus.
- Was ist der Datentyp von `None`? Der ist `NoneType`.
- Das war's auch schon, mehr brauchen wir uns hier gar nicht anschauen.
- `None` steht für „Nichts“.

# Probieren wir das mal aus



- Probieren wir das mal aus.
- Was ist der Datentyp von `None`? Der ist `NoneType`.
- Das war's auch schon, mehr brauchen wir uns hier gar nicht anschauen.
- `None` steht für „Nichts“.
- Es ist kein Wert und keine Zahl.



# Probieren wir das mal aus



- Probieren wir das mal aus.
- Was ist der Datentyp von `None`? Der ist `NoneType`.
- Das war's auch schon, mehr brauchen wir uns hier gar nicht anschauen.
- `None` steht für „Nichts“.
- Es ist kein Wert und keine Zahl.
- Funktionen, die nichts zurückliefern, liefern `None` zurück.

# Probieren wir das mal aus



- Probieren wir das mal aus.
- Was ist der Datentyp von `None`? Der ist `NoneType`.
- Das war's auch schon, mehr brauchen wir uns hier gar nicht anschauen.
- `None` steht für „Nichts“.
- Es ist kein Wert und keine Zahl.
- Funktionen, die nichts zurückliefern, liefern `None` zurück.
- Wenn wir wissen wollen, ob `x` `None` ist, dann schreiben wir `x is None`.



# Zusammenfassung



# Zusammenfassung



- Das war ein sehr kurzes Kapitel. Trotzdem haben wir einiges gelernt.



# Zusammenfassung



- Das war ein sehr kurzes Kapitel. Trotzdem haben wir einiges gelernt.
- `None` steht für „Nichts“.



# Zusammenfassung



- Das war ein sehr kurzes Kapitel. Trotzdem haben wir einiges gelernt.
- `None` steht für „Nichts“.
- Wofür braucht man das?



# Zusammenfassung



- Das war ein sehr kurzes Kapitel. Trotzdem haben wir einiges gelernt.
- `None` steht für „Nichts“.
- Wofür braucht man das?
  1. Funktionen, die keine Ergebnisse zurückliefern, liefern `None` zurück.



# Zusammenfassung



- Das war ein sehr kurzes Kapitel. Trotzdem haben wir einiges gelernt.
- `None` steht für „Nichts“.
- Wofür braucht man das?
  1. Funktionen, die keine Ergebnisse zurückliefern, liefern `None` zurück.
  2. Manchmal speichert man mehrere Werte während einer Berechnung. Man kann Variablen (kommt später) mit `None` initialisieren um auszudrücken, dass sie noch keinen Wert haben. Das ist besser als mit `0` oder `nan`...



# Zusammenfassung



- Das war ein sehr kurzes Kapitel. Trotzdem haben wir einiges gelernt.
- `None` steht für „Nichts“.
- Wofür braucht man das?
  1. Funktionen, die keine Ergebnisse zurückliefern, liefern `None` zurück.
  2. Manchmal speichert man mehrere Werte während einer Berechnung. Man kann Variablen (kommt später) mit `None` initialisieren um auszudrücken, dass sie noch keinen Wert haben. Das ist besser als mit `0` oder `nan`...
  3. Manche Funktionen haben optionale Parameter (kommt später) und man nimmt gerne `None` als Standardwert für diese.

# Zusammenfassung



- Das war ein sehr kurzes Kapitel. Trotzdem haben wir einiges gelernt.
- `None` steht für „Nichts“.
- Wofür braucht man das?
  1. Funktionen, die keine Ergebnisse zurückliefern, liefern `None` zurück.
  2. Manchmal speichert man mehrere Werte während einer Berechnung. Man kann Variablen (kommt später) mit `None` initialisieren um auszudrücken, dass sie noch keinen Wert haben. Das ist besser als mit `0` oder `nan`...
  3. Manche Funktionen haben optionale Parameter (kommt später) und man nimmt gerne `None` als Standardwert für diese.
- Der Operator `a is b` prüft, ob zwei Werte `a` und `b` das selbe Objekt sind (das gucken wir uns irgendwann viel später mal genauer an).

# Zusammenfassung



- Das war ein sehr kurzes Kapitel. Trotzdem haben wir einiges gelernt.
- `None` steht für „Nichts“.
- Wofür braucht man das?
  1. Funktionen, die keine Ergebnisse zurückliefern, liefern `None` zurück.
  2. Manchmal speichert man mehrere Werte während einer Berechnung. Man kann Variablen (kommt später) mit `None` initialisieren um auszudrücken, dass sie noch keinen Wert haben. Das ist besser als mit `0` oder `nan`...
  3. Manche Funktionen haben optionale Parameter (kommt später) und man nimmt gerne `None` als Standardwert für diese.
- Der Operator `a is b` prüft, ob zwei Werte `a` und `b` das selbe Objekt sind (das gucken wir uns irgendwann viel später mal genauer an).
- Anders als für `nan`, wo ja `nan == nan` `False` ergibt, gilt `None is None` (und auch `None == None`)



谢谢你们！  
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. 55).
- [2] Ed Bott. *Windows 11 Inside Out*. Hoboken, NJ, USA: Microsoft Press, Pearson Education, Inc., Feb. 2023. ISBN: 978-0-13-769132-6 (siehe S. 55).
- [3] 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. 55).
- [4] Michael Hausenblas. *Learning Modern Linux*. Sebastopol, CA, USA: O'Reilly Media, Inc., Apr. 2022. ISBN: 978-1-0981-0894-6 (siehe S. 55).
- [5] 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. 55).
- [6] 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. 55).
- [7] 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. 55).
- [8] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 55).
- [9] 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. 55).
- [10] 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. 55).
- [11] Guido van Rossum, Barry Warsaw und Alyssa Coghlan. *Style Guide for Python Code*. Python Enhancement Proposal (PEP) 8. Beaverton, OR, USA: Python Software Foundation (PSF), 5. Juli 2001. URL: <https://peps.python.org/pep-0008> (besucht am 2024-07-27) (siehe S. 13–22, 33).



# References II



- [12] Sander van Vugt. *Linux Fundamentals*. 2. Aufl. Hoboken, NJ, USA: Pearson IT Certification, Juni 2022. ISBN: 978-0-13-792931-3 (siehe S. 55).
- [13] 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. 55).



# Glossary (in English) I



IT information technology

**Linux** is the leading open source operating system, i.e., a free alternative for Microsoft Windows<sup>1,4,9,10,12</sup>. We recommend using it for this course, for software development, and for research. Learn more at <https://www.linux.org>. Its variant Ubuntu is particularly easy to use and install.

**Microsoft Windows** is a commercial proprietary operating system<sup>2</sup>. It is widely spread, but we recommend using a Linux variant such as Ubuntu for software development and for our course. Learn more at <https://www.microsoft.com/windows>.

**Python** The Python programming language<sup>6-8,13</sup>, i.e., what you will learn about in our book<sup>13</sup>. Learn more at <https://python.org>.

**Ubuntu** is a variant of the open source operating system Linux<sup>3,5</sup>. We recommend that you use this operating system to follow this class, for software development, and for research. Learn more at <https://ubuntu.com>. If you are in China, you can download it from <https://mirrors.ustc.edu.cn/ubuntu-releases>.