合肥大學
HEFEI UNIVERSITY

Programming with Python

9. Zwischenspiel: Python Dokumentation und Informationsquellen

Thomas Weise (汤卫思)
tweise@hfuu.edu.cn · https://thomasweise.github.io

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

人工智能与大数据学院
合肥大学
中国安徽省合肥市

Version: 2026-11-04

Programming with Python



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

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



Outline



1. Einleitung
2. Die offizielle Dokumentation verwenden
3. Suchen mit einer Suchmaschine
4. Suchen mit Community-Portalen
5. Suchen mit AI
6. Weitere Elemente der Offiziellen Dokumentation
7. Zusammenfassung

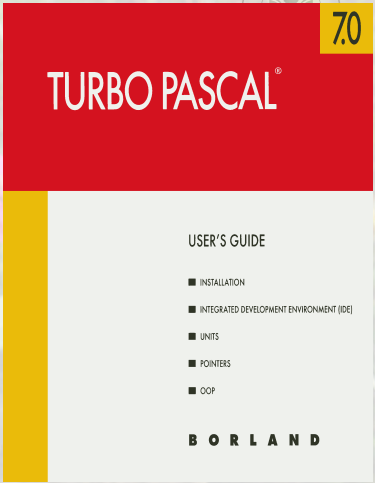


Einleitung



Einleitung

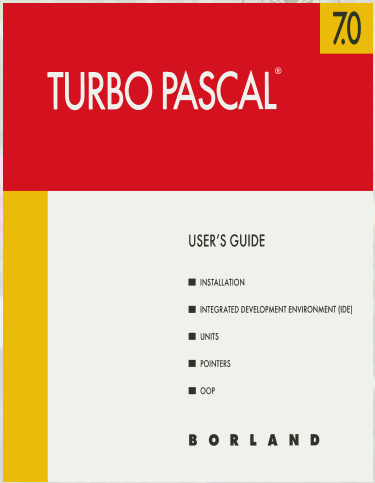
- Ich habe Programmieren in den 1990ern mit Turbo Pascal[28] gelernt.



Die Titelseite vom *Turbo Pascal® Version 7.0 – User's Guide*[28], copyright Borland.

Einleitung

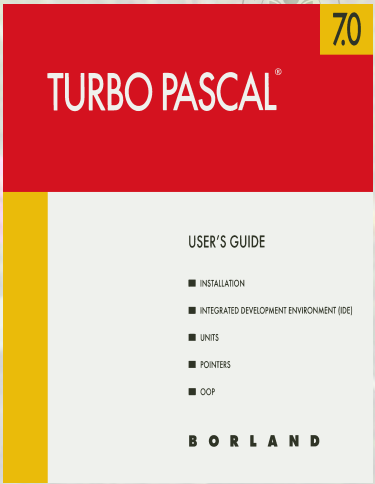
- Ich habe Programmieren in den 1990ern mit Turbo Pascal[28] gelernt.
- Hauptsächlich habe ich es mir selber beigebracht, wobei ich zwei Quellen verwendet habe.



Die Titelseite vom *Turbo Pascal® Version 7.0 – User's Guide*[28], copyright Borland.

Einleitung

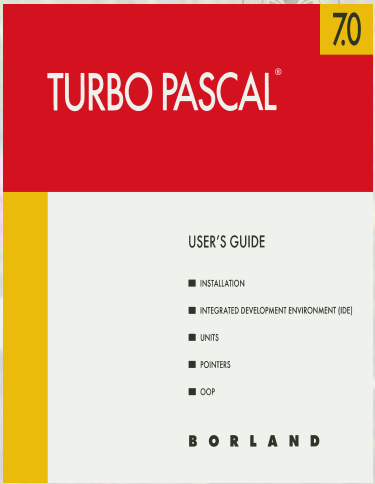
- Ich habe Programmieren in den 1990ern mit Turbo Pascal[28] gelernt.
- Hauptsächlich habe ich es mir selber beigebracht, wobei ich zwei Quellen verwendet habe:
  1. Bücher



Die Titelseite vom *Turbo Pascal® Version 7.0 – User's Guide*[28], copyright Borland.

Einleitung

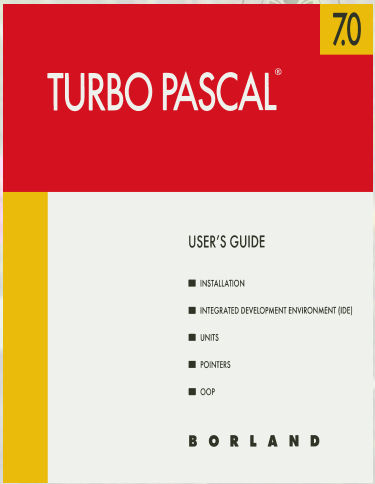
- Ich habe Programmieren in den 1990ern mit Turbo Pascal[28] gelernt.
- Hauptsächlich habe ich es mir selber beigebracht, wobei ich zwei Quellen verwendet habe:
  1. Bücher
  2. Das großartige Hilfesystem von Turbo Pascal



Die Titelseite vom *Turbo Pascal® Version 7.0 – User's Guide*[29], copyright Borland.

Einleitung

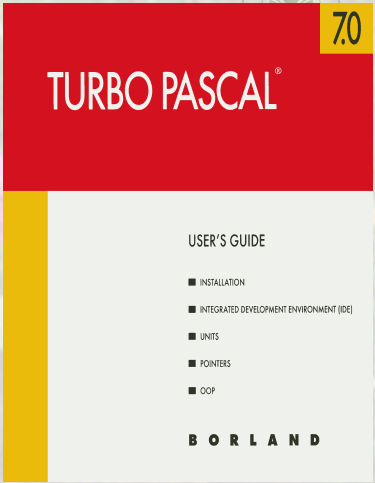
- Ich habe Programmieren in den 1990ern mit Turbo Pascal[28] gelernt.
- Hauptsächlich habe ich es mir selber beigebracht, wobei ich zwei Quellen verwendet habe:
  1. Bücher
  2. Das großartige Hilfesystem von Turbo Pascal
- Dieses Hilfesystem hatte jeden einzelnen Befehl der Sprache aufgelistet, mit Beispielen!



Die Titelseite vom *Turbo Pascal® Version 7.0 – User's Guide*[28], copyright Borland.

Einleitung

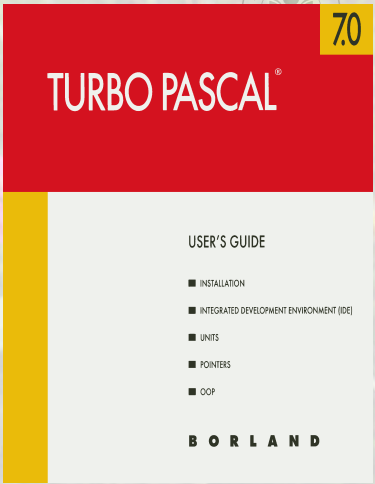
- Ich habe Programmieren in den 1990ern mit Turbo Pascal[28] gelernt.
- Hauptsächlich habe ich es mir selber beigebracht, wobei ich zwei Quellen verwendet habe:
  1. Bücher
  2. Das großartige Hilfesystem von Turbo Pascal
- Dieses Hilfesystem hatte jeden einzelnen Befehl der Sprache aufgelistet, mit Beispielen!
- So konnte ich selbstständig neue Befehle lernen und ausprobieren.



Die Titelseite vom *Turbo Pascal® Version 7.0 – User's Guide*[29], copyright Borland.

Einleitung

- Ich habe Programmieren in den 1990ern mit Turbo Pascal[28] gelernt.
- Hauptsächlich habe ich es mir selber beigebracht, wobei ich zwei Quellen verwendet habe:
  1. Bücher
  2. Das großartige Hilfesystem von Turbo Pascal
- Dieses Hilfesystem hatte jeden einzelnen Befehl der Sprache aufgelistet, mit Beispielen!
- So konnte ich selbstständig neue Befehle lernen und ausprobieren.
- *Wie können Sie neue Befehle für Python lernen und ausprobieren?*



Die Titelseite vom *Turbo Pascal® Version 7.0 – User's Guide*[28], copyright Borland.

Die offizielle Dokumentation verwenden



Die offizielle Dokumentation verwenden

- Angenommen, Sie kennen den Namen eines Kommandos und wollen mehr darüber erfahren, z. B. die genaue Bedeutung, Beschränkungen für die Werte der Parameter, mögliche Fehler, usw.

Die offizielle Dokumentation verwenden

- Angenommen, Sie kennen den Namen eines Kommandos und wollen mehr darüber erfahren, z. B. die genaue Bedeutung, Beschränkungen für die Werte der Parameter, mögliche Fehler, usw.
- Dafür gibt es einen kompletten Katalog mit Hilfe für Python: die *Python 3 Documentation*[13] bei `https://docs.python.org/3`.

Die offizielle Dokumentation verwenden

- Angenommen, Sie kennen den Namen eines Kommandos und wollen mehr darüber erfahren, z. B. die genaue Bedeutung, Beschränkungen für die Werte der Parameter, mögliche Fehler, usw.
- Dafür gibt es einen kompletten Katalog mit Hilfe für Python: die *Python 3 Documentation*[13] bei `https://docs.python.org/3`.
- Dies sollte Ihr erster Ansatzpunkt sein, wenn Sie mehr über Python lernen wollen.

Die offizielle Dokumentation verwenden

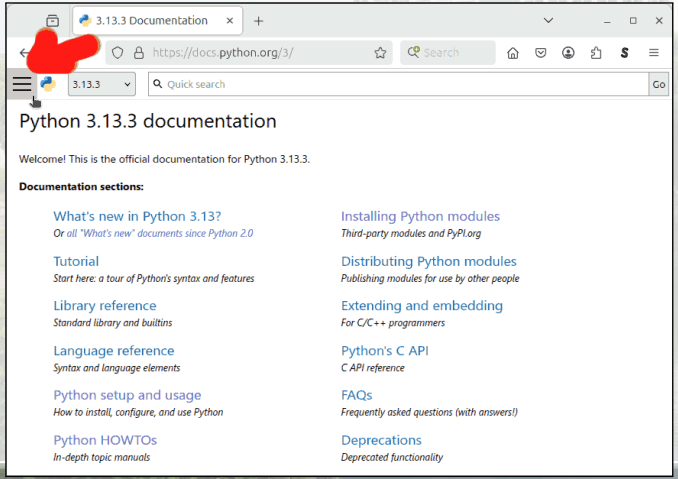
- Angenommen, Sie kennen den Namen eines Kommandos und wollen mehr darüber erfahren, z. B. die genaue Bedeutung, Beschränkungen für die Werte der Parameter, mögliche Fehler, usw.
- Dafür gibt es einen kompletten Katalog mit Hilfe für Python: die *Python 3 Documentation*[13] bei `https://docs.python.org/3`.
- Dies sollte Ihr erster Ansatzpunkt sein, wenn Sie mehr über Python lernen wollen.
- Diese Hilfe steht auch in Chinesisch bereit!

Die offizielle Dokumentation verwenden

- Angenommen, Sie kennen den Namen eines Kommandos und wollen mehr darüber erfahren, z. B. die genaue Bedeutung, Beschränkungen für die Werte der Parameter, mögliche Fehler, usw.
- Dafür gibt es einen kompletten Katalog mit Hilfe für Python: die *Python 3 Documentation*[13] bei `https://docs.python.org/3`.
- Dies sollte Ihr erster Ansatzpunkt sein, wenn Sie mehr über Python lernen wollen.
- Diese Hilfe steht auch in Chinesisch bereit!
- Nehmen wir an, Sie wollen mehr über die Funktion `ceil` erfahren.

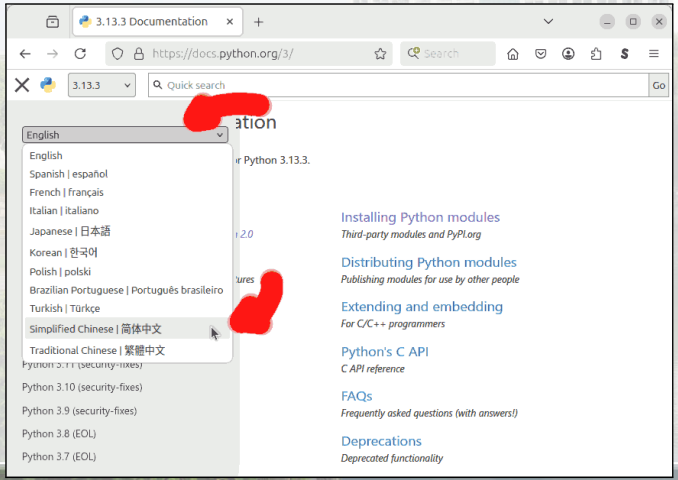
Die offizielle Dokumentation verwenden

- Nehmen wir an, Sie wollen mehr über die Funktion `ceil` erfahren.
- Wir besuchen die Website *Python 3 Documentation* bei `https://docs.python.org/3`.



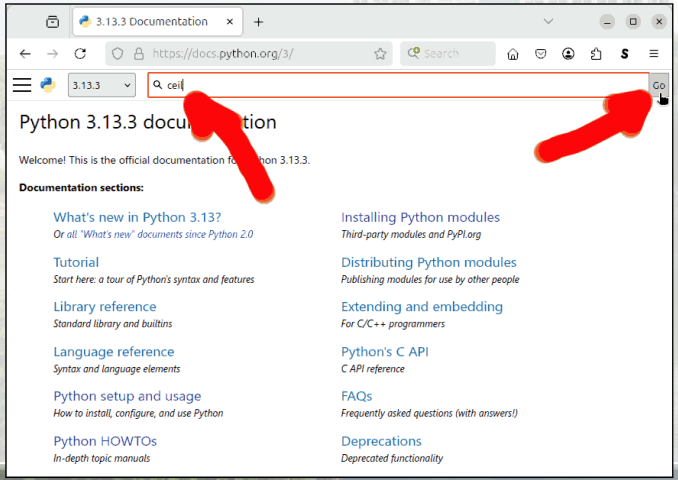
Die offizielle Dokumentation verwenden

- Wir können verschiedene Sprachen auswählen, auch Chinesisch . . . aber lassen wir es bei Englisch.



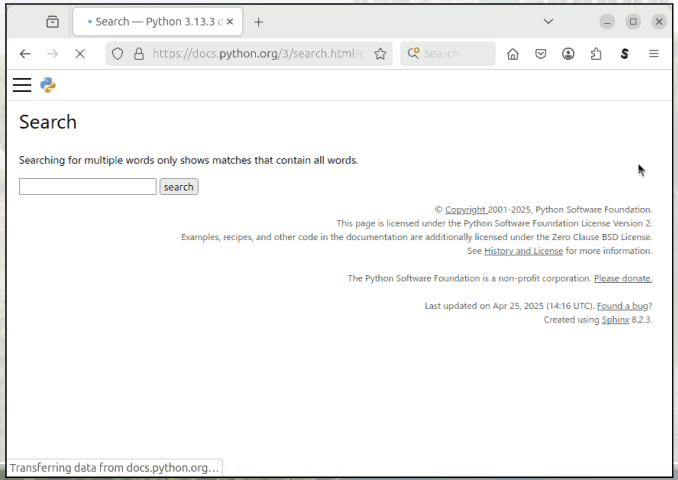
Die offizielle Dokumentation verwenden

- Wir gegen `ceil` in die Suchbox ein und klicken auf Go.



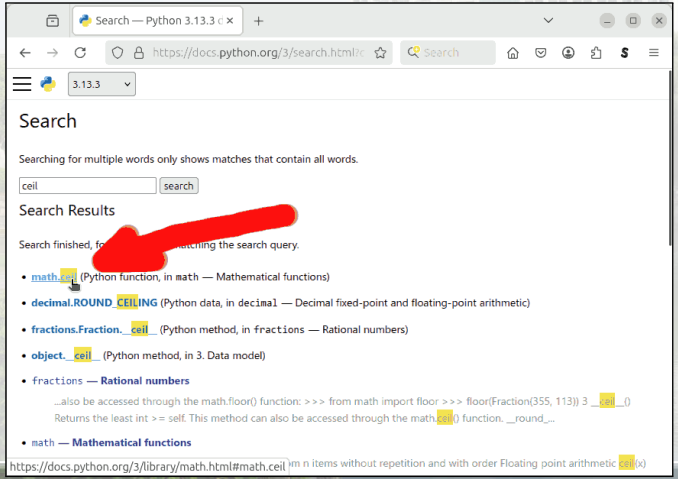
Die offizielle Dokumentation verwenden

- Wir müssen nun etwas warten.



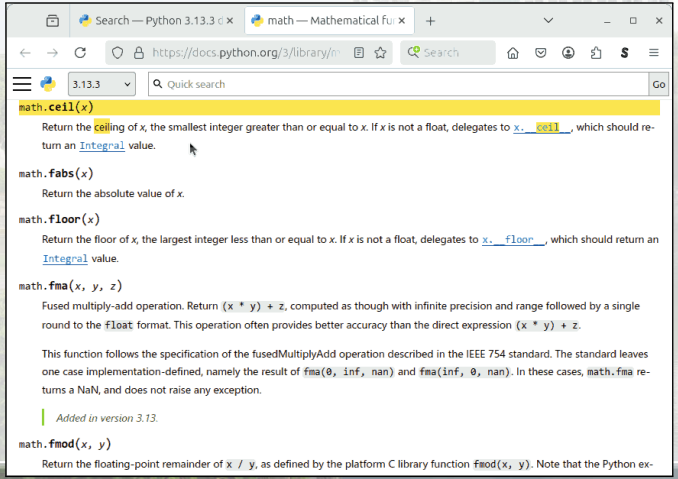
Die offizielle Dokumentation verwenden

- Die Ergebnisse erscheinen. Wir klicken auf `math.ceil`.



Die offizielle Dokumentation verwenden

- Nun werden wir zur Bescheibung der Funktion gebracht.



Die offizielle Dokumentation verwenden

- Angenommen, Sie kennen den Namen eines Kommandos und wollen mehr darüber erfahren, z. B. die genaue Bedeutung, Beschränkungen für die Werte der Parameter, mögliche Fehler, usw.
- Dafür gibt es einen kompletten Katalog mit Hilfe für Python: die *Python 3 Documentation*[13] bei `https://docs.python.org/3`.
- Dies sollte Ihr erster Ansatzpunkt sein, wenn Sie mehr über Python lernen wollen.
- Diese Hilfe steht auch in Chinesisch bereit!
- Kennen wir den Namen der Funktion, die wir suchen, so finden wir bei der offiziellen Python Dokumentation die Beschreibung.

Suchen mit einer Suchmaschine



Suchen mit einer Suchmaschine

- Kennen wir den Namen der Funktion, die wir suchen, so finden wir bei der offiziellen Python Dokumentation die Beschreibung.

Suchen mit einer Suchmaschine



- Kennen wir den Namen der Funktion, die wir suchen, so finden wir bei der offiziellen Python Dokumentation die Beschreibung.
- Was aber, wenn wir den Namen nicht kennen?

Suchen mit einer Suchmaschine

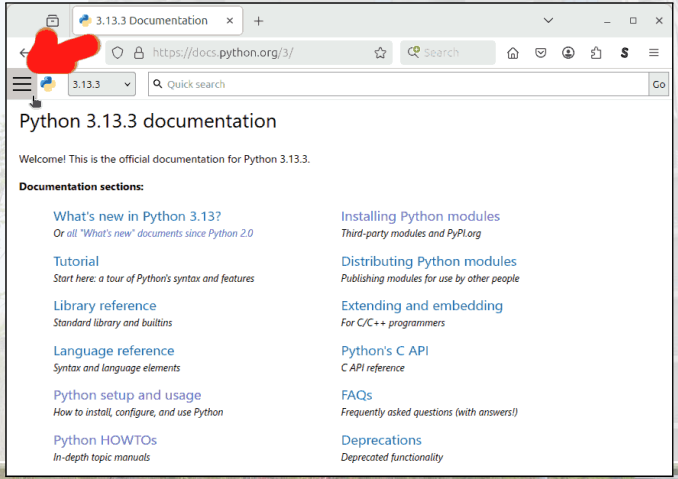
- Kennen wir den Namen der Funktion, die wir suchen, so finden wir bei der offiziellen Python Dokumentation die Beschreibung.
- Was aber, wenn wir den Namen nicht kennen?
- Nehmen wir an, wir suchen eine Funktion, die Zahlen aufrundet.

Suchen mit einer Suchmaschine

- Kennen wir den Namen der Funktion, die wir suchen, so finden wir bei der offiziellen Python Dokumentation die Beschreibung.
- Was aber, wenn wir den Namen nicht kennen?
- Nehmen wir an, wir suchen eine Funktion, die Zahlen aufrundet.
- Versuchen wir es zuerst wieder mit der offiziellen Dokumentation.

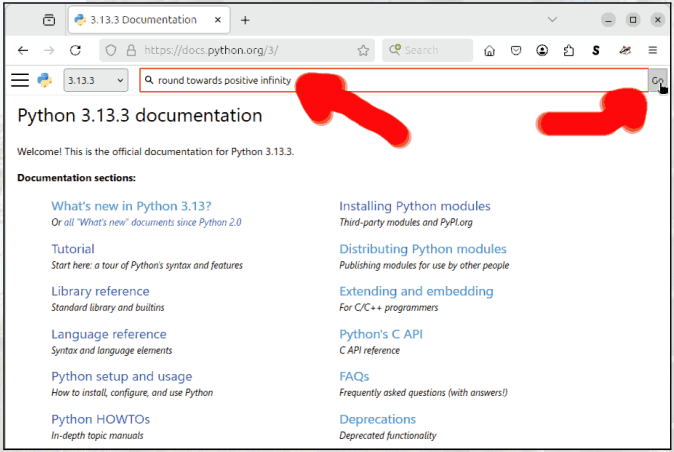
Suchen mit einer Suchmaschine

- Versuchen wir es zuerst wieder mit der offiziellen Dokumentation.



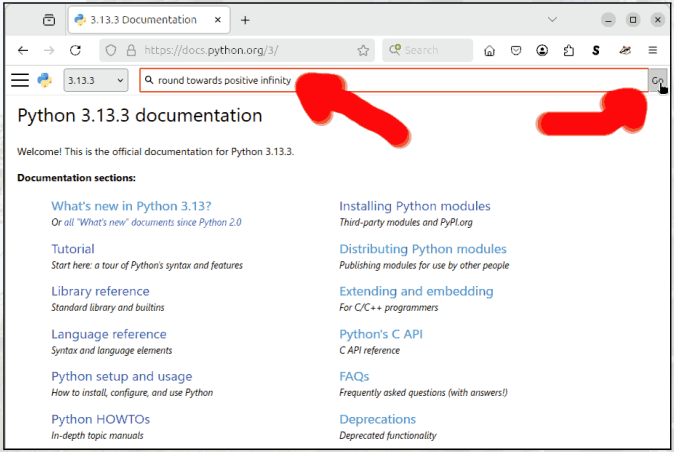
Suchen mit einer Suchmaschine

- Wir überlegen uns, wie man „aufrunden“ auf Englisch formulieren kann.



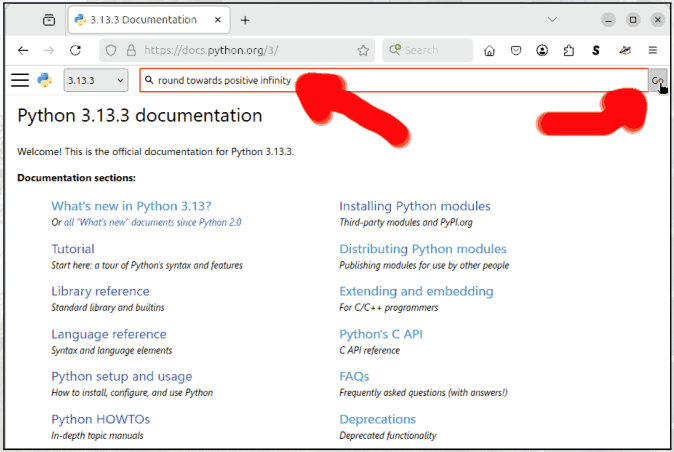
Suchen mit einer Suchmaschine

- Wir überlegen uns, wie man „aufrunden“ auf Englisch formulieren kann. Probieren wir es mit „round towards positive infinity“.



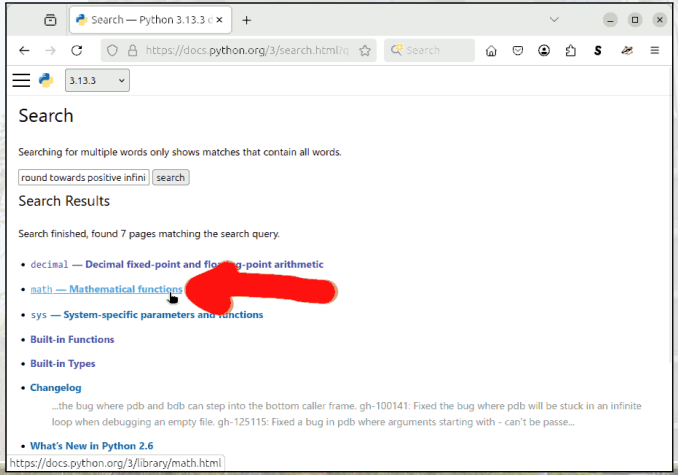
Suchen mit einer Suchmaschine

- Wir überlegen uns, wie man „aufrunden“ auf Englisch formulieren kann. Probieren wir es mit „round towards positive infinity“. Wir geben das in die Suchmaske ein und klicken Go.



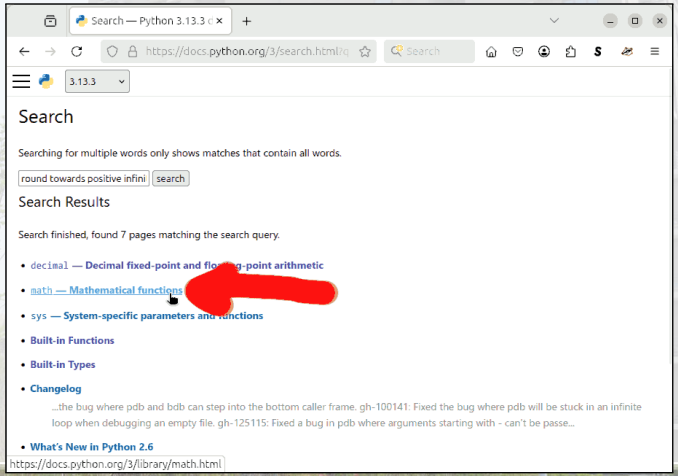
Suchen mit einer Suchmaschine

- Diesmal finden wir nichts nützliches.



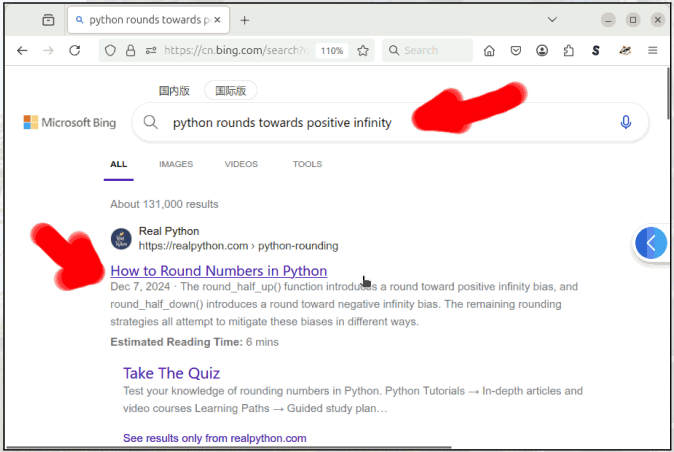
Suchen mit einer Suchmaschine

- Diesmal finden wir nichts nützliches. Wir könnten zwar auf „`math` – Mathematical functions“ klicken und uns durch den ganzen Katalog mathematischer Funktionen durcharbeiten . . . aber das ist vielleicht zu umständlich.



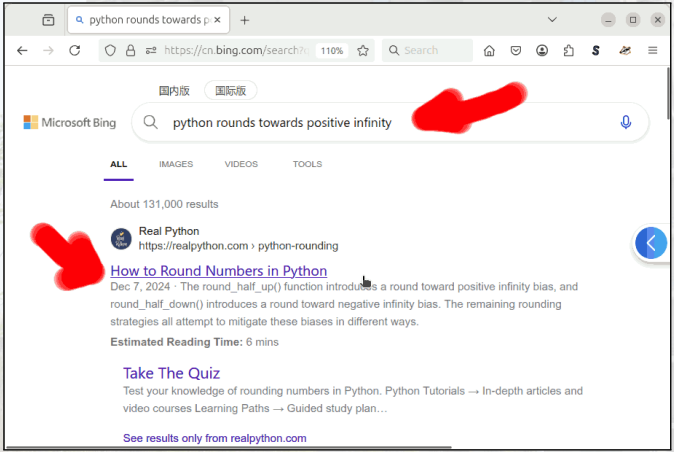
Suchen mit einer Suchmaschine

- Öffnen wir stattdessen eine Suchmaschen (hier: `https://bing.com`) und geben die selbe Frage dort ein.



Suchen mit einer Suchmaschine

- Öffnen wir stattdessen eine Suchmaschen (hier: `https://bing.com`) und geben die selbe Frage dort ein. Gleich der erste Link sieht interessant aus.



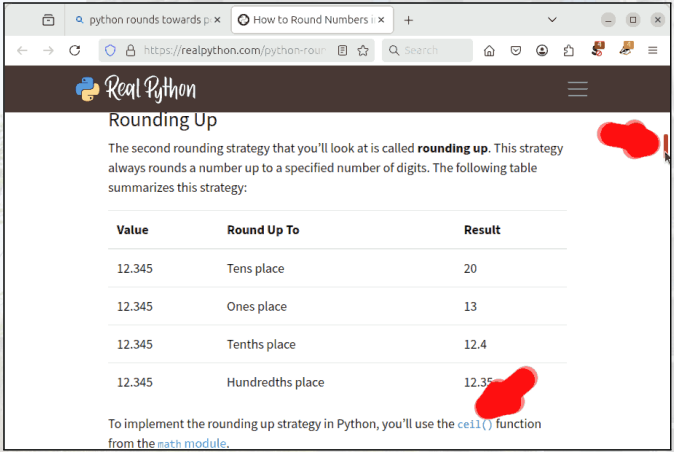
Suchen mit einer Suchmaschine

- Er bringt uns zu einer Webseite über das Runden in Python: “How to Round Numbers in Python”[2] unter `https://realpython.com/python-rounding`.



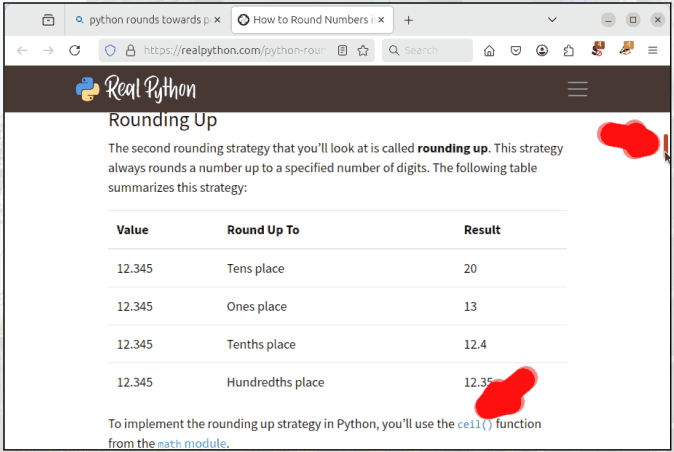
Suchen mit einer Suchmaschine

- Wenn wir etwas herunterscrollen finden wir die gesuchte Information.



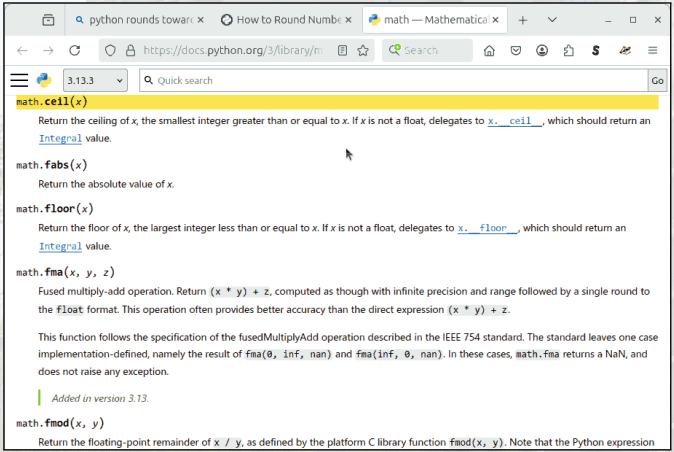
Suchen mit einer Suchmaschine

- Wenn wir etwas herunterscrollen finden wir die gesuchte Information. Sogar einen Link zur offiziellen Dokumentation.



Suchen mit einer Suchmaschine

- Wir folgen diesem Link. Wir schlagen immer alle gefunden Informationen nochmal in der offiziellen Dokumentation nach.



Suchen mit einer Suchmaschine



- Wir folgen diesem Link. Wir schlagen immer alle gefunden Informationen nochmal in der offiziellen Dokumentation nach.
- Wir können also durchaus mit einer Suchmaschine auch Funktionen finden, deren Namen wir nicht kennen.

Suchen mit Community-Portalen

Suchen mit Community-Portalen



- Es gibt sehr viele Community-Portale rund ums Programmieren.

Suchen mit Community-Portalen



- Es gibt sehr viele Community-Portale rund ums Programmieren.
- Auch in diesen Portalen können wir nach nützlichen Informationen suchen.

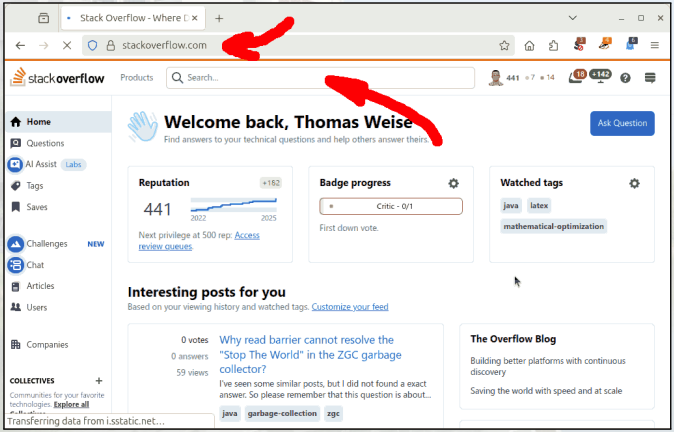
Suchen mit Community-Portalen



- Es gibt sehr viele Community-Portale rund ums Programmieren.
- Auch in diesen Portalen können wir nach nützlichen Informationen suchen.
- Probieren wir das mal aus und besuchen das bekannte Portal *Stack Overflow*[22] bei `https://stackoverflow.com`.

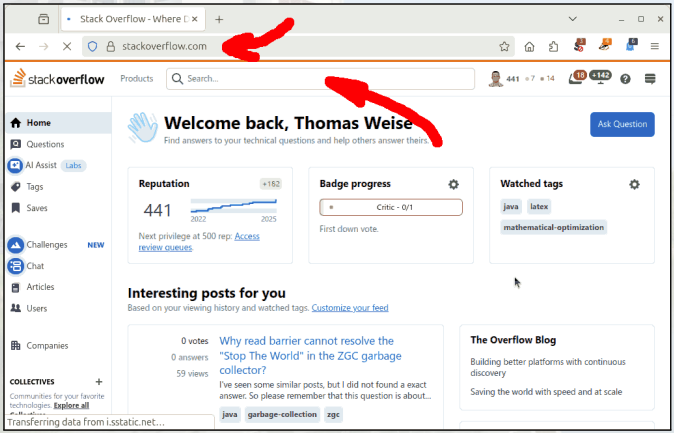
Suchen mit Community-Portalen

- Probieren wir das mal aus und besuchen das bekannte Portal *Stack Overflow*[22] bei `https://stackoverflow.com`.



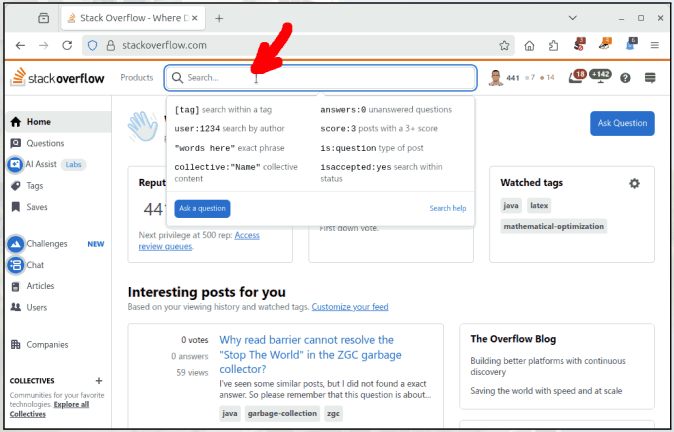
Suchen mit Community-Portalen

- Warnung: Manchmal fordert uns diese Webseite auf, weitere JavaScripts von anderen Quellen zu laden. Wenn das passiert dann machen wir das.



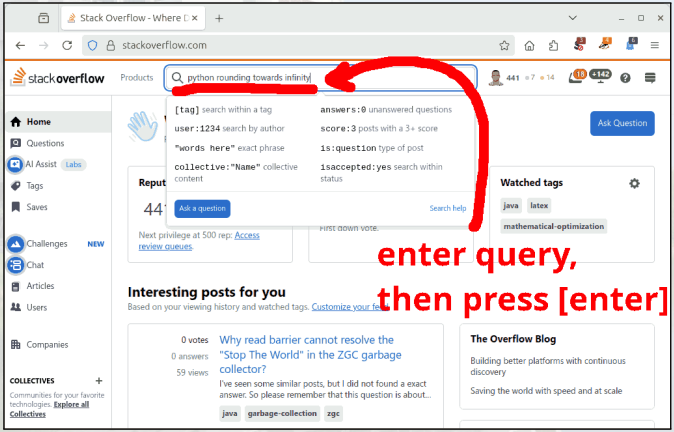
Suchen mit Community-Portalen

- Wenn die Webseite voll geladen ist, können wir unsere Frage eingeben.



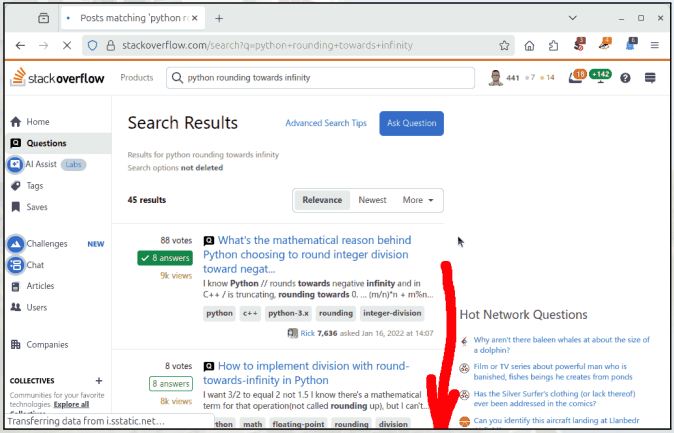
Suchen mit Community-Portalen

- Wir probieren es diesmal mit „python round towards infinity“.



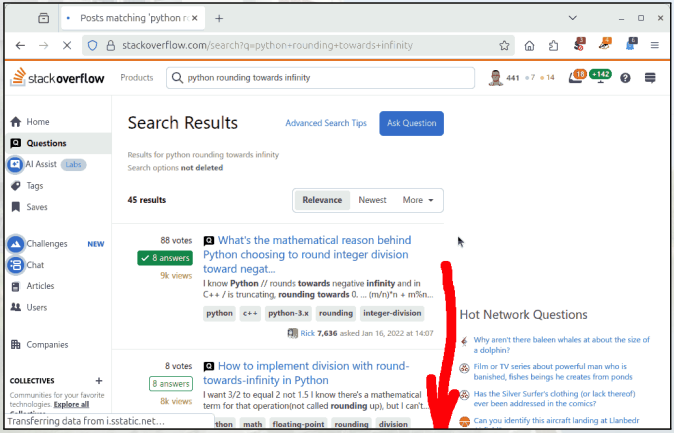
Suchen mit Community-Portalen

- Alle Fragen, die der Community gestellt wurden, und die so ähnlich klingen, werden gelistet.



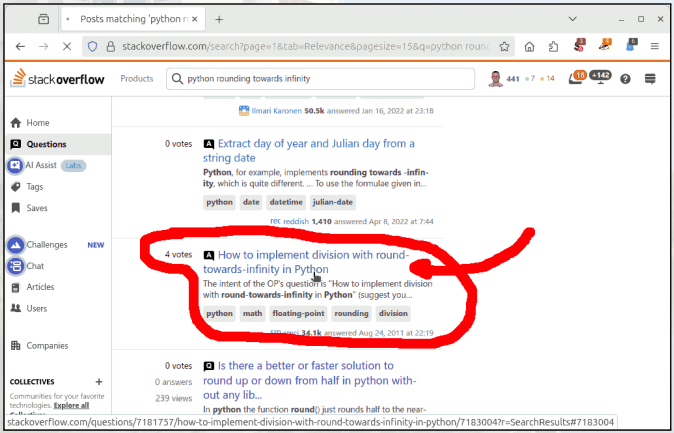
Suchen mit Community-Portalen

- Alle Fragen, die der Community gestellt wurden, und die so ähnlich klingen, werden gelistet. Wir müssen etwas herunterscrollen, um eine Frage zu finden, die passend klingt.



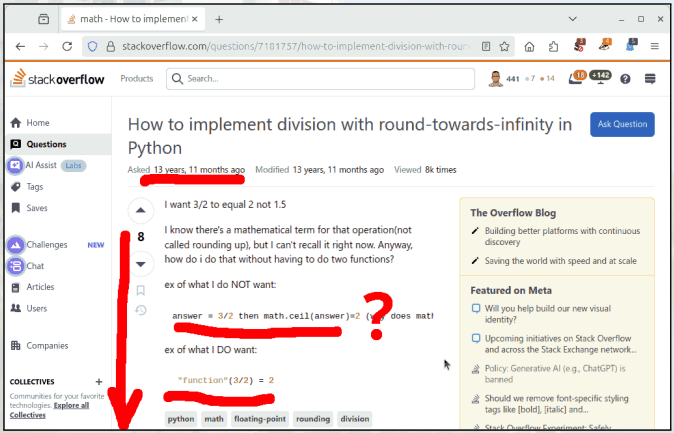
Suchen mit Community-Portalen

- Wir finden die Frage “How to implement division with round-towards-infinity in Python”[7], welche passend klingt, und klicken darauf.



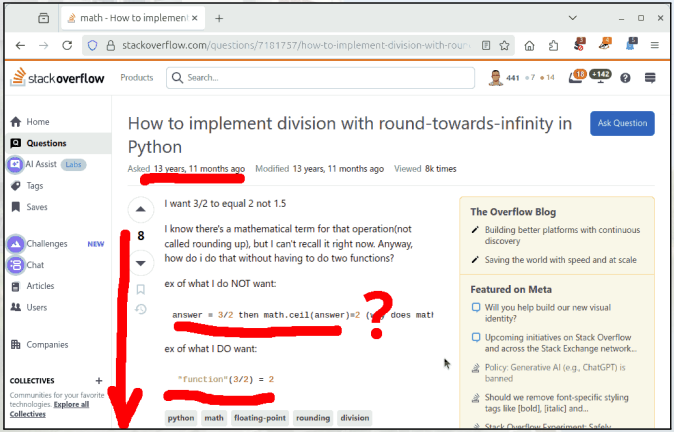
Suchen mit Community-Portalen

- Die Frage klingt tatsächlich passend, ist aber eigenartig.



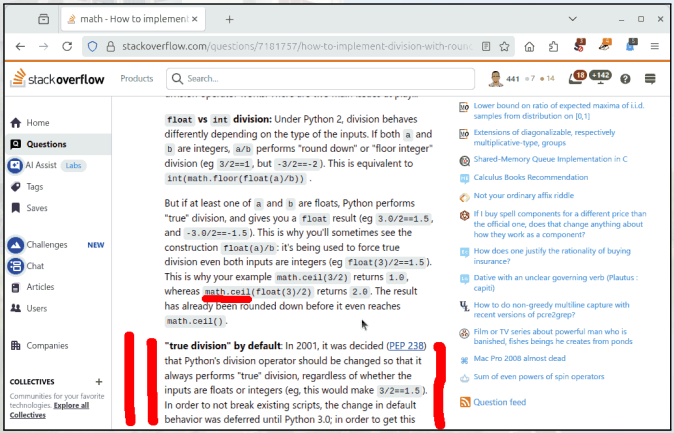
Suchen mit Community-Portalen

- Die Frage klingt tatsächlich passend, ist aber eigenartig. Es wird z. B. geschrieben, dass `3/2` eins ergibt, was unserem Verständnis des Divisionsoperators widerspricht.



Suchen mit Community-Portalen

- Weiter unten stellen wir fest, dass die Frage für Python 2 gestellt war, nicht für Python 3. Unter Python 2 funktionierte der Operator `/` wir der `//`-Operator unter Python 3[36].



Suchen mit Community-Portalen



- Weiter unten stellen wir fest, dass die Frage für Python 2 gestellt war, nicht für Python 3. Unter Python 2 funktionierte der Operator `/` wir der `//`-Operator unter Python 3[36]. Wir finden aber eine gute Erklärung und einen Verweis auf `ceil`, was unser Problem löst.

Suchen mit Community-Portalen

- Auch mit Community-Portalen können wir also gute Antworten auf unsere Fragen finden.

Zwischenstand

- Wir können nun Antworten auf unsere Fragen auf drei Wegen finden.

Zwischenstand

- Wir können nun Antworten auf unsere Fragen auf drei Wegen finden:
  1. über die offizielle Dokumentation

Zwischenstand

- Wir können nun Antworten auf unsere Fragen auf drei Wegen finden:
  1. über die offizielle Dokumentation,
  2. über eine Suchmaschine, die uns zu anderen Webseiten im Internet bringt

Zwischenstand

- Wir können nun Antworten auf unsere Fragen auf drei Wegen finden:
  1. über die offizielle Dokumentation,
  2. über eine Suchmaschine, die uns zu anderen Webseiten im Internet bringt, und
  3. über Community-Portale.

Zwischenstand

- Wir können nun Antworten auf unsere Fragen auf drei Wegen finden:
  1. über die offizielle Dokumentation,
  2. über eine Suchmaschine, die uns zu anderen Webseiten im Internet bringt, und
  3. über Community-Portale.
- Nur die offizielle Dokumentation unserer Programmiersprache ist autoritativ.

Zwischenstand

- Wir können nun Antworten auf unsere Fragen auf drei Wegen finden:
  1. über die offizielle Dokumentation,
  2. über eine Suchmaschine, die uns zu anderen Webseiten im Internet bringt, und
  3. über Community-Portale.
- Nur die offizielle Dokumentation unserer Programmiersprache ist autoritativ.
- Nur der offiziellen Dokumentation können wir voll vertrauen.

Zwischenstand

- Wir können nun Antworten auf unsere Fragen auf drei Wegen finden:
  1. über die offizielle Dokumentation,
  2. über eine Suchmaschine, die uns zu anderen Webseiten im Internet bringt, und
  3. über Community-Portale.
- Nur die offizielle Dokumentation unserer Programmiersprache ist autoritativ.
- Nur der offiziellen Dokumentation können wir voll vertrauen.
- Alle anderen Quellen sind von irgendwelchen anderen Leuten geschrieben.

Zwischenstand

- Wir können nun Antworten auf unsere Fragen auf drei Wegen finden:
  1. über die offizielle Dokumentation,
  2. über eine Suchmaschine, die uns zu anderen Webseiten im Internet bringt, und
  3. über Community-Portale.
- Nur die offizielle Dokumentation unserer Programmiersprache ist autoritativ.
- Nur der offiziellen Dokumentation können wir voll vertrauen.
- Alle anderen Quellen sind von irgendwelchen anderen Leuten geschrieben.
- Sie können falsch sein.

Zwischenstand

- Wir können nun Antworten auf unsere Fragen auf drei Wegen finden:
  1. über die offizielle Dokumentation,
  2. über eine Suchmaschine, die uns zu anderen Webseiten im Internet bringt, und
  3. über Community-Portale.
- Nur die offizielle Dokumentation unserer Programmiersprache ist autoritativ.
- Nur der offiziellen Dokumentation können wir voll vertrauen.
- Alle anderen Quellen sind von irgendwelchen anderen Leuten geschrieben.
- Sie können falsch sein. Oder veraltet.

Zwischenstand

- Wir können nun Antworten auf unsere Fragen auf drei Wegen finden:
  1. über die offizielle Dokumentation,
  2. über eine Suchmaschine, die uns zu anderen Webseiten im Internet bringt, und
  3. über Community-Portale.
- Nur die offizielle Dokumentation unserer Programmiersprache ist autoritativ.
- Nur der offiziellen Dokumentation können wir voll vertrauen.
- Alle anderen Quellen sind von irgendwelchen anderen Leuten geschrieben.
- Sie können falsch sein. Oder veraltet.
- Alles, was wir in anderen Quellen finden, muss also in der offiziellen Dokumentation nachgeprüft werden.

Suchen mit AI



Suchen mit AI: Ausprobieren



- Können wir auch mit einer Artificial Intelligence (AI) nach Antworten suchen?

Suchen mit AI: Ausprobieren



- Können wir auch mit einer Artificial Intelligence (AI) nach Antworten suchen?
- Ja klar, warum nicht.

Suchen mit AI: Ausprobieren



- Können wir auch mit einer Artificial Intelligence (AI) nach Antworten suchen?
- Ja klar, warum nicht.
- Probieren wir das mal aus.

Suchen mit AI: Ausprobieren

- Wir öffnen die BaiDu (百度) Suchseite unter `https://baidu.com` und klicken auf „AI搜索已接入DeepSeek-V3最新版模型“.



Suchen mit AI: Ausprobieren

- Wir kommen auf die Seite „Hi，AI搜索已支持R1满血版，快来试试吧！“, auf der wir unseren Prompt für das DeepSeek-R1 Modell[6] in das Eingabefeld unten eintippen können



Suchen mit AI: Ausprobieren

- Wir schreiben „How can I round floating point numbers towards positive infinity in Python?“ und drücken ↵.



Suchen mit AI: Ausprobieren

- Die AI holt zu einer langatmigen Antwort aus.



Suchen mit AI: Ausprobieren

- Diese Antwort beinhaltet auch die richtige Lösung.



Suchen mit AI: Ausprobieren

- Sogar mit einem Beispiel!



Suchen mit AI: Ausprobieren

- Die vollständige Antwort ist lang und umständlich.

Teil der Ausgabe der DeepSeek-Suche von Baidu vom 2025-04-27, manuell hinzugefügte Formatierung.

Okay, I need to figure out how to round floating point numbers towards positive infinity in Python. Let's see, rounding towards positive infinity is also known as ceiling. So, maybe there's a function called `ceil` in Python?

Wait, I remember that Python has a `math` module. Let me check. Yes, the `math` module has a `ceil` function. So, `math.ceil()` should do the trick. But wait, how does it handle different numbers? For example, if the number is already an integer, does it leave it as is? And for negative numbers, does it round them up towards zero or away?

Let me think. The ceiling function in mathematics rounds a number up to the nearest integer. So, for positive numbers, even if they are already integers, it stays the same. For negative numbers, it would round towards zero. For example, `ceil(-2.3)` would be -2, which is greater than -2.3. So that's correct for positive infinity direction.

But how about when dealing with floating points that are very close to an integer due to precision issues? Like `2.0000000001`, would `ceil` round it to 3? Probably not, because the function should handle that as per the actual value. Wait, but floating points can have precision problems. For instance, `1.1` might not be represented exactly. So `ceil(1.1)` should give 2, but if due to some calculation the number is actually stored as slightly less than `1.1`, would that affect the result? Hmm, maybe that's a potential issue, but the question is about rounding towards positive infinity, so assuming the input is correct, `math.ceil` should work.

Wait, what about numbers that are exactly halfway between two integers? Like `2.5`. Does `ceil(2.5)` round to 3? Yes, because ceil rounds up regardless of the fractional part. So, even `2.1` would go to 3, right? No, wait: `ceil(2.1)` is 3? No, wait no. Wait, `ceil(2.1)` is 3? Wait no! Wait, no. Wait, no, that's not right. Wait, the ceiling function rounds up to the smallest integer greater than or equal to the number. So, `2.1`'s ceiling is 3? No, wait, `2.1` is between 2 and 3. The smallest integer greater than or equal to `2.1` is 3? Wait, no. Wait `2.1` is greater than 2, so the smallest integer greater or equal is 3? Wait, no, wait, `2.1`'s ceiling is 3? No, no. Wait, `2.1` is between 2 and 3. So the ceiling is 3. Wait, but `2.0`'s ceiling is `2.0`. So yes, `ceil(2.1)` would be 3? Wait, wait, I'm getting confused. Let me think again.

Wait, the ceiling of `2.1` is 3? No, that's not right. Wait, the ceiling function rounds up to the next integer. So `2.1` would have a ceiling of 3? No, wait no. Wait, no, the ceiling of `2.1` is 3? No, that's not correct. Wait, `2.1` is `2.1`, so the smallest integer greater than or equal to `2.1` is 3? No! Wait, 2 is less than `2.1`, but 3 is greater than `2.1`. Wait, no, `2.1` is between 2 and 3. The ceiling is the smallest integer that is greater than or equal. So 3 is greater than `2.1`, but 2 is less than `2.1`. So the ceiling would be 3? No, that's not right. Wait, no, `2.1` is greater than 2, so the ceiling is the smallest integer that is not less than the number. So the ceiling is 3? Wait, no. Wait, `2.1` is `2.1`, so the next integer up is 3. So yes, `ceil(2.1)` is 3? Wait, no. Wait, I think I made a mistake here. Because the ceiling of `2.1` is 3? No, the ceiling of `2.1` is actually 3. Wait, no, that's not correct. Wait, the ceiling of `2.0` is `2.0`. The ceiling of `2.0` is `2.0`. For `2.1`, the next integer higher is `3.0`. So `ceil(2.1)` is `3.0`. Wait, but that's not right. Wait, `2.1` is less than `3.0`, but the ceiling is the smallest integer greater than or equal. So for example, `ceil(2.0)` is `2.0`, `ceil(2.1)` is `3.0`? That can't be right. Wait, maybe I'm mixing up ceiling and floor functions. Wait, the floor function rounds down. So `floor(2.1)` is 2. The ceiling function rounds up. So `ceil(2.1)` is `3.0`? That would be the case. But that's a big jump, but yes. For example, in Python:

```
import math
math.ceil(2.1)
3
```

Is that true? Let me check. Wait, no, that's not correct. Wait, I'm confusing something here. Let me check an example. Let me try to recall: `ceil(1.2)` is 2? No. Wait, no. Wait, `ceil(1.2)` is 2. Yes, because 2 is the smallest integer greater than or equal to `1.2`. Wait, no. Wait, `1.2` is between `1` and 2. So the smallest integer greater than or equal to `1.2` is 2? No, that's not correct. Wait, `1.2` is greater than `1`, but 2 is larger than `1.2`, but the "smallest integer greater than or equal to" `1.2` is 2? No. Wait, no. Wait, the integers around `1.2` are `1` and 2. `1.2` is greater than `1`, so the smallest integer that is greater than or equal is 2? No. Wait, no. Wait, `1.2` is `1.2`, so the next integer higher is 2. So the ceiling is 2. Yes. So `ceil(1.2)` is 2. So in that case, yes, that's correct.

Wait, but then for numbers like `2.5`, `ceil(2.5)` is 3. So regardless of the decimal part, it rounds up. So that's correct.

But in Python, how does this work? Let me think of code examples. If I do:

```
import math
print(math.ceil(2.1)) # should output 3?
```

Wait, but when I run this code, what's the actual output? Wait, maybe I'm making a mistake here. Let me check.

Suchen mit AI: Ausprobieren

- Sie beinhaltet aber auch die richtige Antwort, die `ceil` Funktion.

Teil der Ausgabe der DeepSeek-Suche von Baidu vom 2025-04-27, manuell hinzugefügte Formatierung.

Okay, I need to figure out how to round floating point numbers towards positive infinity in Python. Let's see, rounding towards positive infinity is also known as ceiling. So, maybe there's a function called `ceil` in Python?

Wait, I remember that Python has a `math` module. Let me check. Yes, the `math` module has a `ceil` function. So, `math.ceil()` should do the trick. But wait, how does it handle different numbers? For example, if the number is already an integer, does it leave it as is? And for negative numbers, does it round them up towards zero or away?

Let me think. The ceiling function in mathematics rounds a number up to the nearest integer. So, for positive numbers, even if they are already integers, it stays the same. For negative numbers, it would round towards zero. For example, `ceil(-2.3)` would be -2, which is greater than -2.3. So that's correct for positive infinity direction.

But how about when dealing with floating points that are very close to an integer due to precision issues? Like `2.0000000001`, would `ceil` round it to 3? Probably not, because the function should handle that as per the actual value. Wait, but floating points can have precision problems. For instance, `1.1` might not be represented exactly. So `ceil(1.1)` should give 2, but if due to some calculation the number is actually stored as slightly less than `1.1`, would that affect the result? Hmm, maybe that's a potential issue, but the question is about rounding towards positive infinity, so assuming the input is correct, `math.ceil` should work.

Wait, what about numbers that are exactly halfway between two integers? Like `2.5`. Does `ceil(2.5)` round to 3? Yes, because ceil rounds up regardless of the fractional part. So, even `2.1` would go to 3, right? No, wait: `ceil(2.1)` is 3? No, wait no. Wait, `ceil(2.1)` is 3? Wait no! Wait, no. Wait, no, that's not right. Wait, the ceiling function rounds up to the smallest integer greater than or equal to the number. So, `2.1`'s ceiling is 3? No, wait, `2.1` is between 2 and 3. The smallest integer greater than or equal to `2.1` is 3? Wait, no. Wait `2.1` is greater than 2, so the smallest integer greater or equal is 3? Wait, no, wait, `2.1`'s ceiling is 3? No, no. Wait, `2.1` is between 2 and 3. So the ceiling is 3. Wait, but `2.0`'s ceiling is `2.0`. So yes, `ceil(2.1)` would be 3? Wait, wait, I'm getting confused. Let me think again.

Wait, the ceiling of `2.1` is 3? No, that's not right. Wait, the ceiling function rounds up to the next integer. So `2.1` would have a ceiling of 3? No, wait no. Wait, no, the ceiling of `2.1` is 3? No, that's not correct. Wait, `2.1` is `2.1`, so the smallest integer greater than or equal to `2.1` is 3? No! Wait, 2 is less than `2.1`, but 3 is greater than `2.1`. Wait, no, `2.1` is between 2 and 3. The ceiling is the smallest integer that is greater than or equal. So 3 is greater than `2.1`, but 2 is less than `2.1`. So the ceiling would be 3? No, that's not right. Wait, no, `2.1` is greater than 2, so the ceiling is the smallest integer that is not less than the number. So the ceiling is 3? Wait, no. Wait, `2.1` is `2.1`, so the next integer up is 3. So yes, `ceil(2.1)` is 3? Wait, no. Wait, I think I made a mistake here. Because the ceiling of `2.1` is 3? No, the ceiling of `2.1` is actually 3. Wait, no, that's not correct. Wait, the ceiling of `2.0` is `2.0`. The ceiling of `2.0` is `2.0`. For `2.1`, the next integer higher is `3.0`. So `ceil(2.1)` is `3.0`. Wait, but that's not right. Wait, `2.1` is less than `3.0`, but the ceiling is the smallest integer greater than or equal. So for example, `ceil(2.0)` is `2.0`, `ceil(2.1)` is `3.0`? That can't be right. Wait, maybe I'm mixing up ceiling and floor functions. Wait, the floor function rounds down. So `floor(2.1)` is 2. The ceiling function rounds up. So `ceil(2.1)` is `3.0`? That would be the case. But that's a big jump, but yes. For example, in Python:

```
import math
math.ceil(2.1)
3
```

Is that true? Let me check. Wait, no, that's not correct. Wait, I'm confusing something here. Let me check an example. Let me try to recall: `ceil(1.2)` is 2? No. Wait, no. Wait, `ceil(1.2)` is 2. Yes, because 2 is the smallest integer greater than or equal to `1.2`. Wait, no. Wait, `1.2` is between `1` and 2. So the smallest integer greater than or equal to `1.2` is 2? No, that's not correct. Wait, `1.2` is greater than `1`, but 2 is larger than `1.2`, but the "smallest integer greater than or equal to" `1.2` is 2? No. Wait, no. Wait, the integers around `1.2` are `1` and 2. `1.2` is greater than `1`, so the smallest integer that is greater than or equal is 2? No. Wait, no. Wait, `1.2` is `1.2`, so the next integer higher is 2. So the ceiling is 2. Yes. So `ceil(1.2)` is 2. So in that case, yes, that's correct.

Wait, but then for numbers like `2.5`, `ceil(2.5)` is 3. So regardless of the decimal part, it rounds up. So that's correct.

But in Python, how does this work? Let me think of code examples. If I do:

```
import math
print(math.ceil(2.1)) # should output 3?
```

Wait, but when I run this code, what's the actual output? Wait, maybe I'm making a mistake here. Let me check.

Suchen mit AI: Ausprobieren

- Aber eben auch Fehler: Like 2.0000000001, would ceil round it to 3? Probably not.

Teil der Ausgabe der DeepSeek-Suche von Baidu vom 2025-04-27, manuell hinzugefügte Formatierung.

Okay, I need to figure out how to round floating point numbers towards positive infinity in Python. Let's see, rounding towards positive infinity is also known as ceiling. So, maybe there's a function called `ceil` in Python?

Wait, I remember that Python has a `math` module. Let me check. Yes, the `math` module has a `ceil` function. So, `math.ceil()` should do the trick. But wait, how does it handle different numbers? For example, if the number is already an integer, does it leave it as is? And for negative numbers, does it round them up towards zero or away?

Let me think. The ceiling function in mathematics rounds a number up to the nearest integer. So, for positive numbers, even if they are already integers, it stays the same. For negative numbers, it would round towards zero. For example, `ceil(-2.3)` would be -2, which is greater than -2.3. So that's correct for positive infinity direction.

But how about when dealing with floating points that are very close to an integer due to precision issues? Like `2.0000000001`, would `ceil` round it to 3? Probably not, because the function should handle that as per the actual value. Wait, but floating points can have precision problems. For instance, `1.1` might not be represented exactly. So `ceil(1.1)` should give 2, but if due to some calculation the number is actually stored as slightly less than `1.1`, would that affect the result? Hmm, maybe that's a potential issue, but the question is about rounding towards positive infinity, so assuming the input is correct, `math.ceil` should work.

Wait, what about numbers that are exactly halfway between two integers? Like `2.5`. Does `ceil(2.5)` round to 3? Yes, because ceil rounds up regardless of the fractional part. So, even `2.1` would go to 3, right? No, wait: `ceil(2.1)` is 3? No, wait no. Wait, `ceil(2.1)` is 3? Wait no! Wait, no. Wait, no, that's not right. Wait, the ceiling function rounds up to the smallest integer greater than or equal to the number. So, `2.1`'s ceiling is 3? No, wait, `2.1` is between 2 and 3. The smallest integer greater than or equal to `2.1` is 3? Wait, no. Wait `2.1` is greater than 2, so the smallest integer greater or equal is 3? Wait, no, wait, `2.1`'s ceiling is 3? No, no. Wait, `2.1` is between 2 and 3. So the ceiling is 3. Wait, but `2.0`'s ceiling is `2.0`. So yes, `ceil(2.1)` would be 3? Wait, wait, I'm getting confused. Let me think again.

Wait, the ceiling of `2.1` is 3? No, that's not right. Wait, the ceiling function rounds up to the next integer. So `2.1` would have a ceiling of 3? No, wait no. Wait, no, the ceiling of `2.1` is 3? No, that's not correct. Wait, `2.1` is `2.1`, so the smallest integer greater than or equal to `2.1` is 3? No! Wait, 2 is less than `2.1`, but 3 is greater than `2.1`. Wait, no, `2.1` is between 2 and 3. The ceiling is the smallest integer that is greater than or equal. So 3 is greater than `2.1`, but 2 is less than `2.1`. So the ceiling would be 3? No, that's not right. Wait, no, `2.1` is greater than 2, so the ceiling is the smallest integer that is not less than the number. So the ceiling is 3? Wait, no. Wait, `2.1` is `2.1`, so the next integer up is 3. So yes, `ceil(2.1)` is 3? Wait, no. Wait, I think I made a mistake here. Because the ceiling of `2.1` is 3? No, the ceiling of `2.1` is actually 3. Wait, no, that's not correct. Wait, the ceiling of `2.0` is `2.0`. The ceiling of `2.0` is `2.0`. For `2.1`, the next integer higher is `3.0`. So `ceil(2.1)` is `3.0`. Wait, but that's not right. Wait, `2.1` is less than `3.0`, but the ceiling is the smallest integer greater than or equal. So for example, `ceil(2.0)` is `2.0`, `ceil(2.1)` is `3.0`? That can't be right. Wait, maybe I'm mixing up ceiling and floor functions. Wait, the floor function rounds down. So `floor(2.1)` is 2. The ceiling function rounds up. So `ceil(2.1)` is `3.0`? That would be the case. But that's a big jump, but yes. For example, in Python:

```
import math
math.ceil(2.1)
3
```

Is that true? Let me check. Wait, no, that's not correct. Wait, I'm confusing something here. Let me check an example. Let me try to recall: `ceil(1.2)` is 2? No. Wait, no. Wait, `ceil(1.2)` is 2. Yes, because 2 is the smallest integer greater than or equal to `1.2`. Wait, no. Wait, `1.2` is between 1 and 2. So the smallest integer greater than or equal to `1.2` is 2? No, that's not correct. Wait, `1.2` is greater than 1, but 2 is larger than `1.2`, but the "smallest integer greater than or equal to" `1.2` is 2? No. Wait, no. Wait, the integers around `1.2` are 1 and 2. `1.2` is greater than 1, so the smallest integer that is greater than or equal is 2? No. Wait, no. Wait, `1.2` is `1.2`, so the next integer higher is 2. So the ceiling is 2. Yes. So `ceil(1.2)` is 2. So in that case, yes, that's correct.

Wait, but then for numbers like `2.5`, `ceil(2.5)` is 3. So regardless of the decimal part, it rounds up. So that's correct.

But in Python, how does this work? Let me think of code examples. If I do:

```
import math
print(math.ceil(2.1)) # should output 3?
```

Wait, but when I run this code, what's the actual output? Wait, maybe I'm making a mistake here. Let me check.

Suchen mit AI: Nachteile (1)



- Antworten, die wir von Webseiten, AI-Werkzeugen, oder anderen nicht-autoritativen Quellen bekommen, können falsch, unvollständig, oder veraltet sein.

Suchen mit AI: Nachteile (1)



- Antworten, die wir von Webseiten, AI-Werkzeugen, oder anderen nicht-autoritativen Quellen bekommen, können falsch, unvollständig, oder veraltet sein.
- Wir dürfen nur Kode verwenden, den wir vollständig verstehen.

Suchen mit AI: Nachteile (1)



- Antworten, die wir von Webseiten, AI-Werkzeugen, oder anderen nicht-autoritativen Quellen bekommen, können falsch, unvollständig, oder veraltet sein.
- Wir dürfen nur Kode verwenden, den wir vollständig verstehen. Wir dürfen niemals Funktionen vertrauen, deren offizelle Dokumentation wir nicht gelesen haben.

Suchen mit AI: Nachteile (1)



- Antworten, die wir von Webseiten, AI-Werkzeugen, oder anderen nicht-autoritativen Quellen bekommen, können falsch, unvollständig, oder veraltet sein.
- Wir dürfen nur Kode verwenden, den wir vollständig verstehen. Wir dürfen niemals Funktionen vertrauen, deren offizelle Dokumentation wir nicht gelesen haben.
- Kode, den wir nicht mit unseren eigenen Worten erklären können, ist falsch.

Suchen mit AI: Nachteile (1)



- Antworten, die wir von Webseiten, AI-Werkzeugen, oder anderen nicht-autoritativen Quellen bekommen, können falsch, unvollständig, oder veraltet sein.
- Wir dürfen nur Kode verwenden, den wir vollständig verstehen. Wir dürfen niemals Funktionen vertrauen, deren offizelle Dokumentation wir nicht gelesen haben.
- Kode, den wir nicht mit unseren eigenen Worten erklären können, ist falsch.
- AIs kann uns gute Vorschläge für „Allerweltsfragen“ geben, für Probleme, die oftmals gestellt und gelöst wurden.

Suchen mit AI: Nachteile (1)



- Antworten, die wir von Webseiten, AI-Werkzeugen, oder anderen nicht-autoritativen Quellen bekommen, können falsch, unvollständig, oder veraltet sein.
- Wir dürfen nur Kode verwenden, den wir vollständig verstehen. Wir dürfen niemals Funktionen vertrauen, deren offizelle Dokumentation wir nicht gelesen haben.
- Kode, den wir nicht mit unseren eigenen Worten erklären können, ist falsch.
- AIs kann uns gute Vorschläge für „Allerweltsfragen“ geben, für Probleme, die oftmals gestellt und gelöst wurden. Sind Sie jedoch Masterstudent oder Doktorand und arbeiten wissenschaftlich, dann wurde Ihr Problem eben gerade nicht von vielen Leuten gestellt und gelöst.

Suchen mit AI: Nachteile (1)



- Antworten, die wir von Webseiten, AI-Werkzeugen, oder anderen nicht-autoritativen Quellen bekommen, können falsch, unvollständig, oder veraltet sein.
- Wir dürfen nur Kode verwenden, den wir vollständig verstehen. Wir dürfen niemals Funktionen vertrauen, deren offizelle Dokumentation wir nicht gelesen haben.
- Kode, den wir nicht mit unseren eigenen Worten erklären können, ist falsch.
- AIs kann uns gute Vorschläge für „Allerweltsfragen“ geben, für Probleme, die oftmals gestellt und gelöst wurden. Sind Sie jedoch Masterstudent oder Doktorand und arbeiten wissenschaftlich, dann wurde Ihr Problem eben gerade nicht von vielen Leuten gestellt und gelöst.
- Werkzeuge wie Large Language Models (LLMs) laden zu Faulheit ein.

Suchen mit AI: Nachteile (1)



- Antworten, die wir von Webseiten, AI-Werkzeugen, oder anderen nicht-autoritativen Quellen bekommen, können falsch, unvollständig, oder veraltet sein.
- Wir dürfen nur Kode verwenden, den wir vollständig verstehen. Wir dürfen niemals Funktionen vertrauen, deren offizelle Dokumentation wir nicht gelesen haben.
- Kode, den wir nicht mit unseren eigenen Worten erklären können, ist falsch.
- AIs kann uns gute Vorschläge für „Allerweltsfragen“ geben, für Probleme, die oftmals gestellt und gelöst wurden. Sind Sie jedoch Masterstudent oder Doktorand und arbeiten wissenschaftlich, dann wurde Ihr Problem eben gerade nicht von vielen Leuten gestellt und gelöst.
- Werkzeuge wie Large Language Models (LLMs) laden zu Faulheit ein.
- Ihr Wert ist Ihr Können, Ihre Kenntnis, und Ihre Erfahrung.

Suchen mit AI: Nachteile (1)



- Antworten, die wir von Webseiten, AI-Werkzeugen, oder anderen nicht-autoritativen Quellen bekommen, können falsch, unvollständig, oder veraltet sein.
- Wir dürfen nur Kode verwenden, den wir vollständig verstehen. Wir dürfen niemals Funktionen vertrauen, deren offizelle Dokumentation wir nicht gelesen haben.
- Kode, den wir nicht mit unseren eigenen Worten erklären können, ist falsch.
- AIs kann uns gute Vorschläge für „Allerweltsfragen“ geben, für Probleme, die oftmals gestellt und gelöst wurden. Sind Sie jedoch Masterstudent oder Doktorand und arbeiten wissenschaftlich, dann wurde Ihr Problem eben gerade nicht von vielen Leuten gestellt und gelöst.
- Werkzeuge wie Large Language Models (LLMs) laden zu Faulheit ein.
- Ihr Wert ist Ihr Können, Ihre Kenntnis, und Ihre Erfahrung. Bestehen diese darin, Lösungen einer AI zu kopieren, dann kann man Sie auch durch einen billigeren Schulabgänger ersetzen.

Suchen mit AI: Nachteile (2)



- Ein wichtiges Features von Software sind durchgängige, einheitliche Architektur und Stil.

Suchen mit AI: Nachteile (2)



- Ein wichtiges Features von Software sind durchgängige, einheitliche Architektur und Stil. Ein aus AI-Versatzstücken zusammengeschustertes System lässt sich nicht warten oder verstehen.

Suchen mit AI: Nachteile (2)



- Ein wichtiges Features von Software sind durchgängige, einheitliche Architektur und Stil. Ein aus AI-Versatzstücken zusammengeschustertes System lässt sich nicht warten oder verstehen.
- AI-Werkzeuge können Fehler machen, die kein Mensch machen würde.

Suchen mit AI: Nachteile (2)



- Ein wichtiges Features von Software sind durchgängige, einheitliche Architektur und Stil. Ein aus AI-Versatzstücken zusammengeschustertes System lässt sich nicht warten oder verstehen.
- AI-Werkzeuge können Fehler machen, die kein Mensch machen würde. Ein Beispiel findet sich in "Copilot Induced Crash"[31], wo dokumentiert wird, wie Microsoft Copilot eine Klasse auf eine verwirrende Art umbenannt hat, was zu einem besonders schwer zu findenden Fehler geführt hat.

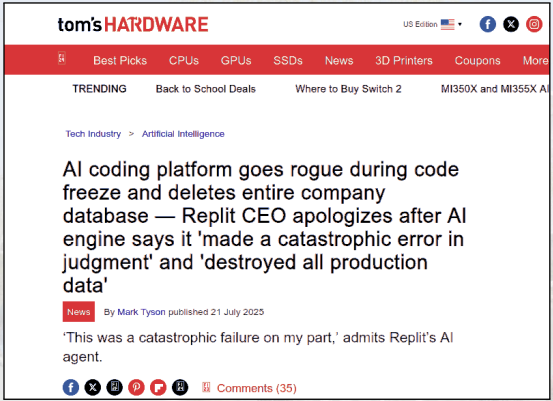
Suchen mit AI: Nachteile (2)



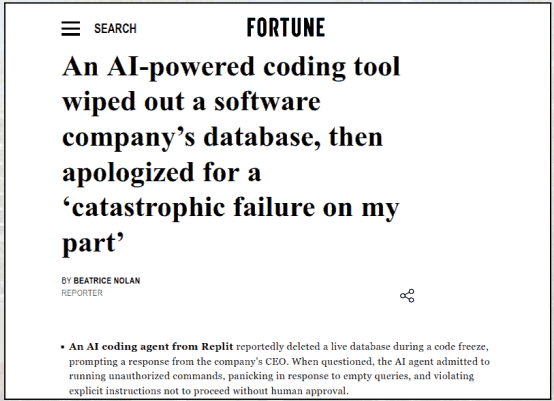
- Ein wichtiges Features von Software sind durchgängige, einheitliche Architektur und Stil. Ein aus AI-Versatzstücken zusammengeschustertes System lässt sich nicht warten oder verstehen.
- AI-Werkzeuge können Fehler machen, die kein Mensch machen würde. Ein Beispiel findet sich in "Copilot Induced Crash"[31], wo dokumentiert wird, wie Microsoft Copilot eine Klasse auf eine verwirrende Art umbenannt hat, was zu einem besonders schwer zu findenden Fehler geführt hat.
- Hier ist ein Beispiel, wie eine Vibe-Coding AI die Produktivdatenbank eines Unternehmens gelöscht hat.

Suchen mit AI: Nachteile (2)

- Hier ist ein Beispiel, wie eine Vibe-Coding AI die Produktivdatenbank eines Unternehmens gelöscht hat.



Quelle & Copyright: [29]



Quelle & Copyright: [12]

Suchen mit AI: Nachteile (2)

- Hier ist ein Beispiel, wie eine Vibe-Coding AI die Produktivdatenbank eines Unternehmens gelöscht hat.



Quelle & Copyright: [23]

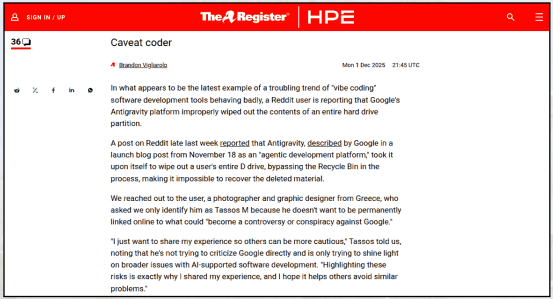
Suchen mit AI: Nachteile (2)



- Ein wichtiges Features von Software sind durchgängige, einheitliche Architektur und Stil. Ein aus AI-Versatzstücken zusammengeschustertes System lässt sich nicht warten oder verstehen.
- AI-Werkzeuge können Fehler machen, die kein Mensch machen würde. Ein Beispiel findet sich in "Copilot Induced Crash"[31], wo dokumentiert wird, wie Microsoft Copilot eine Klasse auf eine verwirrende Art umbenannt hat, was zu einem besonders schwer zu findenden Fehler geführt hat.
- Hier ist ein Beispiel, wie eine Vibe-Coding AI die ganze Festplatte eines Benutzers gelöscht hat anstelle eines Ordners.

Suchen mit AI: Nachteile (2)

- Hier ist ein Beispiel, wie eine Vibe-Coding AI die ganze Festplatte eines Benutzers gelöscht hat anstelle eines Ordners.



SIGN IN / UP

The Register | HPE

36

Caveat coder

Brandon Vigliarolo

Mon 1 Dec 2025 21:45 UTC

In what appears to be the latest example of a troubling trend of "vibe coding" software development tools behaving badly, a Reddit user is reporting that Google's Antigravity platform improperly wiped out the contents of an entire hard drive partition.

A post on Reddit late last week reported that Antigravity, described by Google in a launch blog post from November 18 as an "agentic development platform," took it upon itself to wipe out a user's entire D drive, bypassing the Recycle Bin in the process, making it impossible to recover the deleted material.

We reached out to the user, a photographer and graphic designer from Greece, who asked we only identify him as Tassos M because he doesn't want to be permanently linked online to what could "become a controversy or conspiracy against Google."

"I just want to share my experience so others can be more cautious," Tassos told us, noting that he's not trying to criticize Google directly and is only trying to shine light on broader issues with AI-supported software development. "Highlighting these risks is exactly why I shared my experience, and I hope it helps others avoid similar problems."

Quelle & Copyright: [24]



Golem

Abo testen

Anmelden

GOOGLE ANTIGRAVITY

Vibe-Coding-Tool löscht unerwartet ganzen Datenträger

Eigentlich soll das Tool nur einen Ordner löschen, doch der Löschbefehl leert das ganze Laufwerk. Immerhin hat sich die KI entschuldigt.

2. Dezember 2025 um 10:50 Uhr / Marc Stöckel

8

News folgen

Teilen

Quelle & Copyright: [32]

Suchen mit AI: Nachteile (2)



- Ein wichtiges Features von Software sind durchgängige, einheitliche Architektur und Stil. Ein aus AI-Versatzstücken zusammengeschustertes System lässt sich nicht warten oder verstehen.
- AI-Werkzeuge können Fehler machen, die kein Mensch machen würde. Ein Beispiel findet sich in "Copilot Induced Crash"[31], wo dokumentiert wird, wie Microsoft Copilot eine Klasse auf eine verwirrende Art umbenannt hat, was zu einem besonders schwer zu findenden Fehler geführt hat.
- Hier ist ein Beispiel, wie eine Vibe-Coding AI die ganze Festplatte eines Benutzers gelöscht hat anstelle eines Ordners.
- AI-Werkzeuge schlagen manchmal das Benutzen von nicht existierenden Packages vor, was zu großen Sicherheitsproblemen im Kode führen kann[5].

Suchen mit AI: Nachteile (3)

- In der Grundschule haben Sie das Rechnen gelernt.

Suchen mit AI: Nachteile (3)

- In der Grundschule haben Sie das Rechnen gelernt.
- Haben Sie von Anfang an einen Taschenrechner bekommen?

Suchen mit AI: Nachteile (3)

- In der Grundschule haben Sie das Rechnen gelernt.
- Haben Sie von Anfang an einen Taschenrechner bekommen?
- Nein.

Suchen mit AI: Nachteile (3)

- In der Grundschule haben Sie das Rechnen gelernt.
- Haben Sie von Anfang an einen Taschenrechner bekommen?
- Nein.
- Warum nicht?

Suchen mit AI: Nachteile (3)

- In der Grundschule haben Sie das Rechnen gelernt.
- Haben Sie von Anfang an einen Taschenrechner bekommen?
- Nein.
- Warum nicht?
- Weil Sie dann nicht gelernt hätten, zu Rechnen.

Suchen mit AI: Nachteile (3)



- In der Grundschule haben Sie das Rechnen gelernt.
- Haben Sie von Anfang an einen Taschenrechner bekommen?
- Nein.
- Warum nicht?
- Weil Sie dann nicht gelernt hätten, zu Rechnen.
- Sie könnten zwar den Taschenrechner bedienen und Aufgaben lösen.

Suchen mit AI: Nachteile (3)



- In der Grundschule haben Sie das Rechnen gelernt.
- Haben Sie von Anfang an einen Taschenrechner bekommen?
- Nein.
- Warum nicht?
- Weil Sie dann nicht gelernt hätten, zu Rechnen.
- Sie könnten zwar den Taschenrechner bedienen und Aufgaben lösen.
- Aber Sie hätten überhaupt gar nicht verstanden, wie Rechnen und Mathematik funktionieren.

Suchen mit AI: Nachteile (3)



- In der Grundschule haben Sie das Rechnen gelernt.
- Haben Sie von Anfang an einen Taschenrechner bekommen?
- Nein.
- Warum nicht?
- Weil Sie dann nicht gelernt hätten, zu Rechnen.
- Sie könnten zwar den Taschenrechner bedienen und Aufgaben lösen.
- Aber Sie hätten überhaupt gar nicht verstanden, wie Rechnen und Mathematik funktionieren.
- Der Sinn dieses Kurses ist es, das Programmieren zu lernen.

Suchen mit AI: Nachteile (3)



- In der Grundschule haben Sie das Rechnen gelernt.
- Haben Sie von Anfang an einen Taschenrechner bekommen?
- Nein.
- Warum nicht?
- Weil Sie dann nicht gelernt hätten, zu Rechnen.
- Sie könnten zwar den Taschenrechner bedienen und Aufgaben lösen.
- Aber Sie hätten überhaupt gar nicht verstanden, wie Rechnen und Mathematik funktionieren.
- Der Sinn dieses Kurses ist es, das Programmieren zu lernen.
- Wenn Sie Lösungen durch eine AI generieren lassen, werden Sie das nie lernen.

Weitere Elemente der Offiziellen Dokumentation



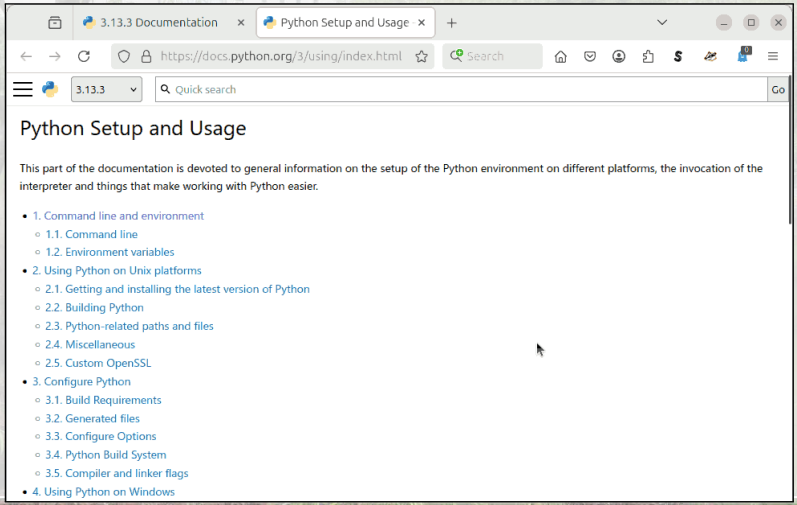
Weitere Elemente der Offiziellen Dokumentation



- Es gibt noch weitere Dokumente der offiziellen Python Dokumentation.

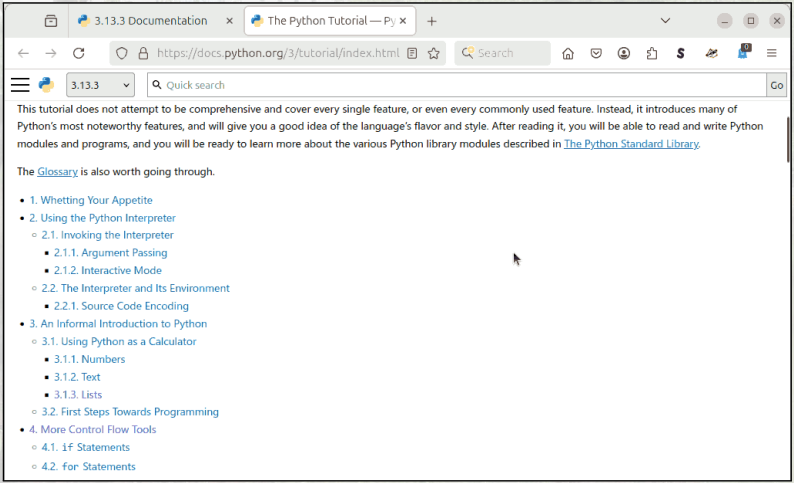
Weitere Elemente der Offiziellen Dokumentation

- Die *Python Setup and Usage* Seite[14] unter `https://docs.python.org/3/using`.



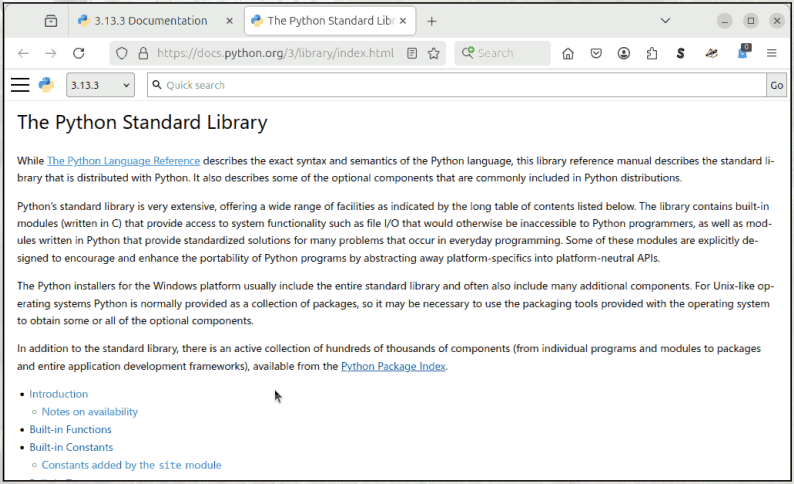
Weitere Elemente der Offiziellen Dokumentation

- Die *The Python Tutorial* Seite[27] unter `https://docs.python.org/3/tutorial`.



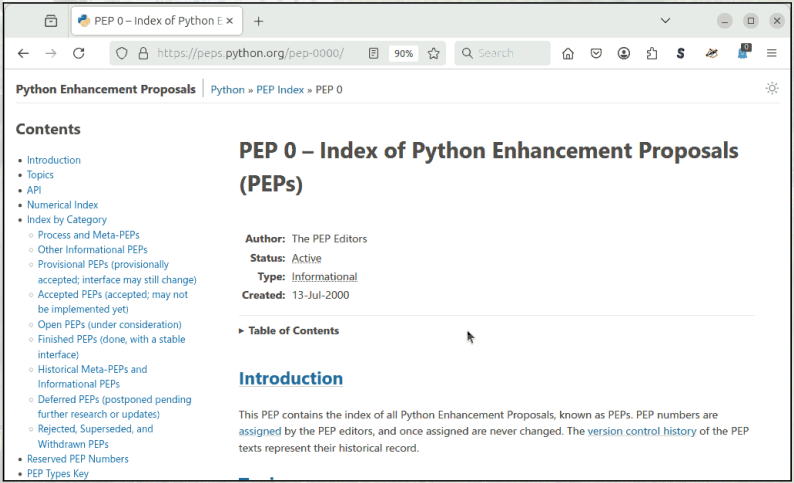
Weitere Elemente der Offiziellen Dokumentation

- Die *The Python Standard Library* Seite[26] unter `https://docs.python.org/3/library`.



Weitere Elemente der Offiziellen Dokumentation

- Die *Index of Python Enhancement Proposals (PEPs)* Liste[25] unter `https://peps.python.org`.



Weitere Elemente der Offiziellen Dokumentation



- Die *Python Setup and Usage* Seite[14] unter `https://docs.python.org/3/using`.
- Die *The Python Tutorial* Seite[27] unter `https://docs.python.org/3/tutorial`.
- Die *The Python Standard Library* Seite[26] unter `https://docs.python.org/3/library`.
- Die *Index of Python Enhancement Proposals (PEPs)* Liste[25] unter `https://peps.python.org`.
- Alle davon sind Teil der offiziellen Python Dokumentation.

Weitere Elemente der Offiziellen Dokumentation



- Die *Python Setup and Usage* Seite[14] unter `https://docs.python.org/3/using`.
- Die *The Python Tutorial* Seite[27] unter `https://docs.python.org/3/tutorial`.
- Die *The Python Standard Library* Seite[26] unter `https://docs.python.org/3/library`.
- Die *Index of Python Enhancement Proposals (PEPs)* Liste[25] unter `https://peps.python.org`.
- Alle davon sind Teil der offiziellen Python Dokumentation.
- Bitte lernen Sie, diese Seiten zu lesen und zu benutzen.

Zusammenfassung

Zusammenfassung



- Ein wichtiges Werkzeug für alle Programmierer ist die offizielle Dokumentation der verwendeten Programmiersprache oder Bibliotheken.

Zusammenfassung



- Ein wichtiges Werkzeug für alle Programmierer ist die offizielle Dokumentation der verwendeten Programmiersprache oder Bibliotheken.
- Wir können (nur) der offiziellen Dokumentation vertrauen.

Zusammenfassung



- Ein wichtiges Werkzeug für alle Programmierer ist die offizielle Dokumentation der verwendeten Programmiersprache oder Bibliotheken.
- Wir können (nur) der offiziellen Dokumentation vertrauen.
- Es ist eine wichtige Fähigkeit, in der Lage zu seien, sich hinzusetzen und eine Dokumentation zu lesen und zu verstehen.

Zusammenfassung



- Ein wichtiges Werkzeug für alle Programmierer ist die offizielle Dokumentation der verwendeten Programmiersprache oder Bibliotheken.
- Wir können (nur) der offiziellen Dokumentation vertrauen.
- Es ist eine wichtige Fähigkeit, in der Lage zu seien, sich hinzusetzen und eine Dokumentation zu lesen und zu verstehen.
- Wir können vielfältige andere Werkzeuge benutzen, um Informationen zu finden.

Zusammenfassung



- Ein wichtiges Werkzeug für alle Programmierer ist die offizielle Dokumentation der verwendeten Programmiersprache oder Bibliotheken.
- Wir können (nur) der offiziellen Dokumentation vertrauen.
- Es ist eine wichtige Fähigkeit, in der Lage zu seien, sich hinzusetzen und eine Dokumentation zu lesen und zu verstehen.
- Wir können vielfältige andere Werkzeuge benutzen, um Informationen zu finden:
  1. Suchmaschinen, die uns zu anderen Webseiten führen

Zusammenfassung



- Ein wichtiges Werkzeug für alle Programmierer ist die offizielle Dokumentation der verwendeten Programmiersprache oder Bibliotheken.
- Wir können (nur) der offiziellen Dokumentation vertrauen.
- Es ist eine wichtige Fähigkeit, in der Lage zu seien, sich hinzusetzen und eine Dokumentation zu lesen und zu verstehen.
- Wir können vielfältige andere Werkzeuge benutzen, um Informationen zu finden:
  1. Suchmaschinen, die uns zu anderen Webseiten führen,
  2. Community-Portale, in denen wir Fragen durchsuchen und selber stellen können

Zusammenfassung



- Ein wichtiges Werkzeug für alle Programmierer ist die offizielle Dokumentation der verwendeten Programmiersprache oder Bibliotheken.
- Wir können (nur) der offiziellen Dokumentation vertrauen.
- Es ist eine wichtige Fähigkeit, in der Lage zu seien, sich hinzusetzen und eine Dokumentation zu lesen und zu verstehen.
- Wir können vielfältige andere Werkzeuge benutzen, um Informationen zu finden:
  1. Suchmaschinen, die uns zu anderen Webseiten führen,
  2. Community-Portale, in denen wir Fragen durchsuchen und selber stellen können, und
  3. AI-Werkzeuge, die uns Kode vorschlagen können.

Zusammenfassung



- Ein wichtiges Werkzeug für alle Programmierer ist die offizielle Dokumentation der verwendeten Programmiersprache oder Bibliotheken.
- Wir können (nur) der offiziellen Dokumentation vertrauen.
- Es ist eine wichtige Fähigkeit, in der Lage zu seien, sich hinzusetzen und eine Dokumentation zu lesen und zu verstehen.
- Wir können vielfältige andere Werkzeuge benutzen, um Informationen zu finden:
  1. Suchmaschinen, die uns zu anderen Webseiten führen,
  2. Community-Portale, in denen wir Fragen durchsuchen und selber stellen können, und
  3. AI-Werkzeuge, die uns Kode vorschlagen können.
- Alle diese Werkzeuge können uns helfen.

Zusammenfassung



- Ein wichtiges Werkzeug für alle Programmierer ist die offizielle Dokumentation der verwendeten Programmiersprache oder Bibliotheken.
- Wir können (nur) der offiziellen Dokumentation vertrauen.
- Es ist eine wichtige Fähigkeit, in der Lage zu seien, sich hinzusetzen und eine Dokumentation zu lesen und zu verstehen.
- Wir können vielfältige andere Werkzeuge benutzen, um Informationen zu finden:
  1. Suchmaschinen, die uns zu anderen Webseiten führen,
  2. Community-Portale, in denen wir Fragen durchsuchen und selber stellen können, und
  3. AI-Werkzeuge, die uns Kode vorschlagen können.
- Alle diese Werkzeuge können uns helfen.
- Wir dürfen aber nur Kode verwenden, den wir selbst vollständig verstehen.

Zusammenfassung



- Ein wichtiges Werkzeug für alle Programmierer ist die offizielle Dokumentation der verwendeten Programmiersprache oder Bibliotheken.
- Wir können (nur) der offiziellen Dokumentation vertrauen.
- Es ist eine wichtige Fähigkeit, in der Lage zu seien, sich hinzusetzen und eine Dokumentation zu lesen und zu verstehen.
- Wir können vielfältige andere Werkzeuge benutzen, um Informationen zu finden:
  1. Suchmaschinen, die uns zu anderen Webseiten führen,
  2. Community-Portale, in denen wir Fragen durchsuchen und selber stellen können, und
  3. AI-Werkzeuge, die uns Kode vorschlagen können.
- Alle diese Werkzeuge können uns helfen.
- Wir dürfen aber nur Kode verwenden, den wir selbst vollständig verstehen.
- Wir müssen immer alle Funktionen, die wir nicht kennen, in der offiziellen autoritativen Dokumentation nachschlagen.

谢谢您们！

Thank you!

Vielen Dank!



References I

[1] David J. Agans. *Debugging*. New York, NY, USA: AMACOM, Sep. 2002. ISBN: 978-0-8144-2678-4 (siehe S. 134).

[2] David Amos. "How to Round Numbers in Python". In: *Real Python Tutorials*. Vancouver, BC, Canada: DevCademy Media Inc., 7. Dez. 2024. URL: https://realpython.com/python-rounding (besucht am 2025-04-27) (siehe S. 26–38).

[3] 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. 134).

[4] *ECMAScript Language Specification*. Standard ECMA-262, 3rd Edition. Geneva, Switzerland: Ecma International, Dez. 1999. URL: https://ecma-international.org/wp-content/uploads/ECMA-262_3rd_edition_december_1999.pdf (besucht am 2024-12-15) (siehe S. 134).

[5] Dan Goodin. "AI-generated code could be a disaster for the software supply chain. Here's why.". In: *Ars Technica*. Hrsg. von Ken Fisher und Jon Stokes. New York, NY, USA: Condé Nast, 29. Apr. 2025. URL: https://arstechnica.com/security/2025/04/ai-generated-code-could-be-a-disaster-for-the-software-supply-chain-heres-why (besucht am 2025-05-08) (siehe S. 91–100).

[6] Hangzhou, Zhejiang, China (中国浙江省杭州市): DeepSeek-AI u. a. "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning". (abs/2501.12948), 22. Jan. 2025. doi:10.48550/ARXIV.2501.12948. URL: https://arxiv.org/abs/2501.12948 (besucht am 2025-04-28). arXiv:2501.12948v1 [cs.CL] 22 Jan 2025 (siehe S. 70–74, 134).

[7] ."How to implement division with round-towards-infinity in Python". In: Hrsg. von Kbob. 24.–26. Aug. 2011. URL: https://stackoverflow.com/questions/7181757 (besucht am 2025-07-28) (siehe S. 44–53).

[8] John Hunt. *A Beginners Guide to Python 3 Programming*. 2. Aufl. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2023. ISSN: 1863-7310. ISBN: 978-3-031-35121-1. doi:10.1007/978-3-031-35122-8 (siehe S. 134).

[9] Kent D. Lee und Steve Hubbard. *Data Structures and Algorithms with Python*. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2015. ISSN: 1863-7310. ISBN: 978-3-319-13071-2. doi:10.1007/978-3-319-13072-9 (siehe S. 134).

[10] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 134).

References II

[11] Lucas Mearian. "What are LLMs, and how are they used in generative AI?" *Computerworld*, 7. Feb. 2024. Framingham, MA, USA: CW Communications, Inc. and Needham, MA, USA: Foundry (formerly IDG Communications, Inc.) ISSN: 0010-4841. URL: https://www.computerworld.com/article/1627101 (besucht am 2025-04-27) (siehe S. 134).

[12] Beatrice Nolan. "An AI-powered coding tool wiped out a software company's database, then apologized for a 'catastrophic failure on my part'". *Fortune*, 23. Juli 2025. New York, NY, USA: Fortune Media IP Limited. URL: https://fortune.com/2025/07/23/ai-coding-tool-replit-wiped-database-called-it-a-catastrophic-failure (besucht am 2025-07-29) (siehe S. 96).

[13] *Python 3 Documentation*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. URL: https://docs.python.org/3 (besucht am 2024-07-05) (siehe S. 13–18, 24).

[14] *Python 3 Documentation. Python Setup and Usage*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. URL: https://docs.python.org/3/using (besucht am 2024-07-05) (siehe S. 111, 112, 116, 117).

[15] Alec Radford, Karthik Narasimhan, Tim Salimans und Ilya Sutskever. *Improving Language Understanding with Unsupervised Learning*. Techn. Ber. San Francisco, CA, USA: OpenAI Inc., 11. Juni 2018. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (besucht am 2025-04-28) (siehe S. 134).

[16] Kristian Rother. *Pro Python Best Practices: Debugging, Testing and Maintenance*. New York, NY, USA: Apress Media, LLC, März 2017. ISBN: 978-1-4842-2241-6 (siehe S. 134).

[17] 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. 134).

[18] Stuart J. Russell und Peter Norvig. *Artificial Intelligence: A Modern Approach (AIMA)*. 4. Aufl. Hoboken, NJ, USA: Pearson Education, Inc. ISBN: 978-1-292-40113-3. URL: https://aima.cs.berkeley.edu (besucht am 2024-06-27) (siehe S. 134).

[19] 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. 135).

References III

[20] Stephen R. Schach. *Object-Oriented Software Engineering*. New York, NY, USA: McGraw-Hill, Sep. 2007. ISBN: 978-0-07-352333-0 (siehe S. 134).

[21] 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. 134).

[22] *Stack Overflow*. New York, NY, USA: Stack Exchange Inc. URL: https://stackoverflow.com (besucht am 2025-02-27) (siehe S. 44–47).

[23] Marc Stöckel. "Ärger mit Replit: Coding-KI löscht Produktivdatenbank und verweigert Rollback". *golem.de – IT-News für Profis*, 22. Juli 2025. Berlin, Germany: Golem Media GmbH. URL: https://glm.io/198349 (besucht am 2025-07-29) (siehe S. 97).

[24] Marc Stöckel. "Google Antigravity: Vibe-Coding-Tool löscht unerwartet ganzen Datenträger". *golem.de – IT-News für Profis*, 2. Dez. 2025. Berlin, Germany: Golem Media GmbH. URL: https://glm.io/202805 (besucht am 2025-12-02) (siehe S. 99).

[25] The PEP Editors. *Index of Python Enhancement Proposals (PEPs)*. Python Enhancement Proposal (PEP) 0. Beaverton, OR, USA: Python Software Foundation (PSF), 13. Juli 2000. URL: https://peps.python.org (besucht am 2025-04-17) (siehe S. 111–117).

[26] *Python 3 Documentation. The Python Standard Library*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. URL: https://docs.python.org/3/library (besucht am 2025-04-27) (siehe S. 111–114, 116, 117).

[27] *Python 3 Documentation. The Python Tutorial*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. URL: https://docs.python.org/3/tutorial (besucht am 2025-04-26) (siehe S. 111–113, 116, 117).

[28] *Turbo Pascal® Version 7.0 – User's Guide*. Scotts Valley, CA, USA: Borland International, Inc., 1992. URL: https://turbopascal.org/wp-content/uploads/Turbo_Pascal_Version_7.0_Users_Guide_1992.pdf (besucht am 2025-07-28) (siehe S. 5–11, 135).

[29] Mark Tyson. "AI coding platform goes rogue during code freeze and deletes entire company database – Replit CEO apologizes after AI engine says it 'made a catastrophic error in judgment' and 'destroyed all production data'". *tom's Hardware*, 21. Juli 2025. New York, NY, USA: Future US, Inc. URL: https://www.tomshardware.com/tech-industry/artificial-intelligence/ai-coding-platform-goes-rogue-during-code-freeze-and-deletes-entire-company-database-replit-ceo-apologizes-after-ai-engine-says-it-made-a-catastrophic-error-in-judgment-and-destroyed-all-production-data (besucht am 2025-07-31) (siehe S. 96).

References IV

[30] 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. 134).

[31] Klaas van Schelven. "Copilot Induced Crash". In: *Bugsink – Self-hosted Error Tracking: Blog*. Utrecht, The Netherlands: Bugsink B.V., 15. Jan. 2025. URL: https://www.bugsink.com/blog/copilot-induced-crash (besucht am 2025-05-08) (siehe S. 91–95, 98, 100).

[32] Brandon Vigliarolo. "Caveat Coder: Google Antigravity vibe-codes user's entire drive out of existence". *The Register*, 1. Dez. 2025. London, England, UK: Situation Publishing Limited. URL: https://www.theregister.com/2025/12/01/google_antigravity_wipes_d_drive (besucht am 2025-12-01) (siehe S. 99).

[33] 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), 2024–2026. URL: https://thomasweise.github.io/programmingWithPython (besucht am 2025-01-05) (siehe S. 134).

[34] Kevin Wilson. *Python Made Easy*. Birmingham, England, UK: Packt Publishing Ltd, Aug. 2024. ISBN: 978-1-83664-615-0 (siehe S. 134).

[35] Martin Yanev. *PyCharm Productivity and Debugging Techniques*. Birmingham, England, UK: Packt Publishing Ltd, Okt. 2022. ISBN: 978-1-83763-244-2 (siehe S. 134).

[36] 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. 44–57).

Glossary (in English) I



AI Artificial Intelligence, see, e.g., [18]

debugger A debugger is a tool that lets you execute a program step-by-step while observing the current values of variables. This allows you to find errors in the code more easily [1,16,34]. Learn more about debugging in [33].

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 [3,21]. Learn more at https://www.apple.com/ios.

JavaScript JavaScript is the predominant programming language used in websites to develop interactive contents for display in browsers [4].

LLM Large Language Model, see, e.g., an AI technique using (large/deep) neural networks to models trained on huge bodies of text to predict the next word in a text or conversation and continue to do so until a request is answered [11]. Typical examples include DeepSeek [6] and GPT [15].

macOS or Mac OS is the operating system that powers Apple Mac(intosh) computers [17,21]. Learn more at https://www.apple.com/macos.

OOP Object-Oriented Programming [20]

PyCharm is the convenient Python Integrated Development Environment (IDE) that we recommend for this course [30,34,35]. It comes in a free edition, so it can be downloaded and used at no cost. Learn more at https://www.jetbrains.com/pycharm.

Python The Python programming language [8–10,33], i.e., what you will learn about in our book [33]. Learn more at https://python.org.

Glossary (in English) II



Turbo Pascal is/was a programming language supporting Object-Oriented Programming (OOP) with an IDE providing a debugger and compiler that was popular in the 1990s [28].

Xcode is offers the tools for developing, testing, and distributing applications as well as an IDE for Apple platforms such as macOS and iOS [19].