LPU - Programmieren mit Unterstützung von LLM - Ausarbeitung

- Einstieg
- Chatbots selbst erstellen
- Prompting so kommunizierst du mit Chatbots
- Code erklären (lassen)
- Tests erstellen (lassen)
- Code vervollständigen (lassen)
- Programmieraufgaben lösen
- <u>Übungsaufgaben</u>
- Noch mehr Übungen ...
- Anhang

Fühlt es sich auch für dich so an, als ob die sogenannte Künstliche Intelligenz - KI - heutzutage überall eingebaut wird? Präsentationsprogramme erstellen dir anhand einiger weniger Stichworte ein komplettes <u>Slide Deck</u>, <u>Suchmaschinen</u> fassen für dich Suchergebnisse und Quellen automatisch zusammen. <u>Dall°E</u>, <u>Suno</u>, <u>Sora</u>, & Co. verwandeln Textbeschreibungen zu Bildern, Musik und Videos. Vielleicht hast du dir auch schon selbst von <u>ChatGPT</u> bei einem Essay helfen lassen, wenn du kurz vor Abgabetermin eine Schreibblockade hattest?

Nach dieser Lerneinheit wirst du

- mit Unterstützung von Chatbots programmieren können.
- dir von deinem digitalen Programmierpartner Algorithmen und Programmcodes erklären lassen können.
- Teillösungen, die dir dein digitaler Programmierpartner geliefert hat, überprüfen und zu einer Gesamtlösung zusammenbauen können.

Stopp! Bevor du weiter liest: Hast du den Abschnitt "Setup deines Computers" im <u>Anhang</u> erfolgreich bearbeitet? Andernfalls kannst du die folgenden Beispiele und Aufgaben nicht selbst ausprobieren.

Einstieg

```
Offine eine Kommandozeile - ein sogenanntes Terminal - und gib den Befehl openai-playground 'Tell a Computer Science joke!' ein.

Fingabe >

openai-playground 'Tell a Computer Science joke!'

Ausgabe >

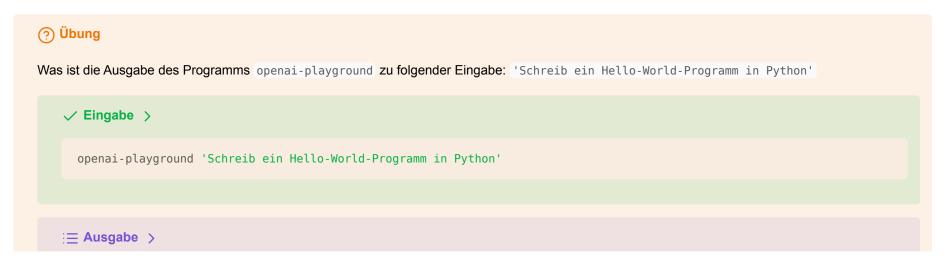
/ Why do programmers prefer dark mode? |
| Because the light attracts bugs! /

/ (oo) |
| () ) \ () |
| | | | | |
```

Was ist hier passiert?^[1]

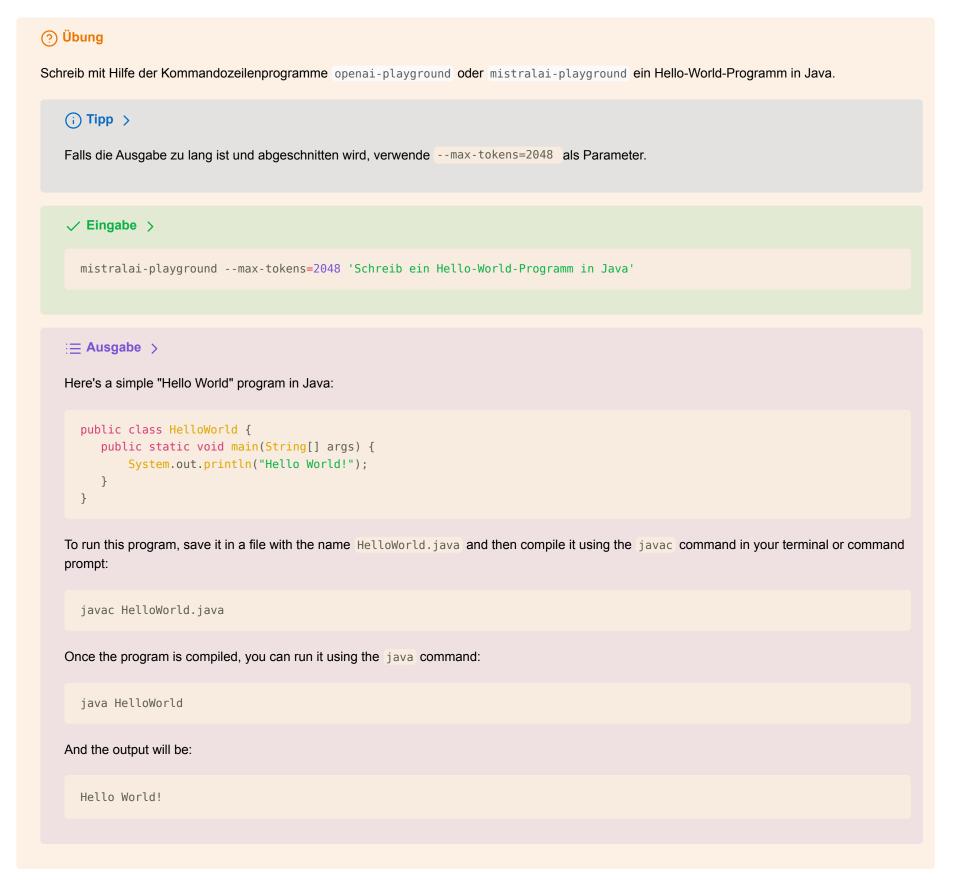
Eine **Eingabe**, nämlich der Text Tell a Computer Science joke! wurde mittels einer Rechenoperation in eine **Ausgabe**, beispielsweise Why do programmers prefer dark mode? Because the light attracts bugs! **transformiert**. Hast du eine andere Ausgabe erhalten? Kennt das Programm mehr als einen Witz?

Kannst du auch auf Deutsch fragen oder wird nur Englisch gesprochen? Probieren wir es aus:



print("Hello, World!")

Geht da noch etwas anderes als flache Witze oder triviale Python-Programme? Funktioniert das auch mit anderen Programmiersprachen, beispielsweise mit Java? Und auch mit einem anderen Anbieter?



Auch wenn du noch nie in Java programmiert hast, sieht das auf den ersten Blick doch alles sehr vielversprechend aus, oder? Neben OpenAl kannst du auch mit anderen Anbietern, wie etwa MistralAl, erfolgreich arbeiten.

Teste doch mal, wie verständlich 'Hello-World-Code' in den Programmiersprachen 'Assembler' oder 'Brainfuck' ist.



Kennst du noch andere Programmiersprachen, die du gerne ausprobieren möchtest? Mehr als 900 Namen findest du unter diesem <u>Link</u>.

Diese Lerneinheit ist so aufgebaut:

Wir steigen mit dir bekannten, einfachen Aufgaben ein. Dabei lernst du, wie du einen Chatbot erstellst, konfigurierst und damit grundsätzlich programmierst. In den folgenden Übungsaufgaben wirst du lernen, schneller und effizienter zu programmieren. Dazu wirst du deinen Chatbots - ja, du kannst so viele Chatbots erstellen, wie du magst - spezifische Rollen zuweisen und dir Codes, die du beispielsweise im Netz findest, erklären lassen. Um herauszufinden, ob ein Programmcode korrekt ist, wirst du lernen, einfache Tests zu verwenden. Diese Tests erstellst du selbstverständlich auch nicht ganz allein. Dann probieren wir aus, wie gut so ein Chatbot funktioniert, wenn du ein vorgegebenes Codegerüst vervollständigen lassen willst. Du wirst Aufgaben analysieren und zerlegen und die Teillösungen wieder zusammenbauen. Anhand von Übungsaufgaben kannst du das Gelernte festigen.

Die Übungsaufgaben lösen wir zunächst mit den im Einstieg gezeigten Kommandozeilenprogrammen. Später verwendest du Jupyter-Chatbooks, eine Erweiterung des dir bekannten Jupyter. Im Gegensatz zu den Kommandozeilenprogrammen kannst du damit Konversationen mit deinen Chatbots mit aufeinander aufbauenden Eingaben führen - auch mit vielen (spezialisierten) Chatbots gleichzeitig in einem einzigen Notebook.

Chatbots selbst erstellen

In dieser Lerneinheit arbeiten wir mit zwei Anbietern: OpenAl und MistralAl. Jeder dieser Anbieter hat verschiedene Modelle zur Auswahl. Die Benutzungskosten dieser Modelle sind abhängig von der Anzahl der Token in der Ein- und Ausgabe. Ein Token entspricht etwa vier Buchstaben oder Zeichen.

In der Regel gilt: Je leistungsfähiger ein Modell ist, desto teurer ist die Benutzung. Wir arbeiten in dieser Lerneinheit mit dem Modell gpt-3.5-turbo und gpt-4 von OpenAl und mistral-tiny, mistral-small sowie mistral-medium von MistralAl. Weitere Informationen findest du im Anhang.

Wie bei den meisten Programmen gibt es auch bei einem Chatbot verschiedene Einstellungen die du anpassen kannst, um das Verhalten des Chatbots zu steuern.

Wir benutzen die Parameter temperature und max-tokens.

Der Parameter temperature ist eine Zahl zwischen 0 und 1, welche die "Kreativität" bestimmt. Ein Modell bei Temperatur 0.2 generiert Code, der sich eher an etablierte Muster und Konventionen hält. Dies ist nützlich für die Generierung von syntaktisch korrektem Programmcode. Eine Temperatur von bspw. 0.7 lässt hingegen alternativen Lösungen und kreativen Ansätzen mehr Raum.

Der Parameter max-tokens limitiert die Gesamtlänge von Ein- und Ausgabe und kann maximal 4096 Token umfassen.

Probieren wir doch an zwei einfachen Beispielen aus, wie sich verschiedene Einstellungen für Anbieter, Modell, Kreativität und Ausgabelänge auswirken.

Öbung

Schreibe ein Python-Programm, das ein Rechteck mit Seitenlängen von 100 und 50 zeichnet. Nutze die Turtle-Bibliothek.

- 1. Schreibe den Code zunächst ohne Hilfe.
- 2. Verwende nun die beiden Programme mistralai-playground und openai-playground, um Code automatisch zu erzeugen. Findest du Einstellungen, die auf Anhieb eine korrekte Lösung liefern? Welche Einstellungen führen zu keinem validen Ergebnis?
- 3. Vergleiche abschliessend deine eigene Lösung mit den automatisch erzeugten Lösungen. Verwenden alle Lösungen den gleichen Algorithmus? Welche Lösung gefällt dir am besten und warum?

(i) Tipp >

Um beispielsweise das Modell mistral-tiny mit einer Temperatur von 0.7 und dem unveränderten Aufgabentext als Eingabe zu testen und maximal 2048 Token für Ein- und Ausgabe zu verwenden, gib folgenden Befehl ein:

```
mistralai-playground --model=mistral-tiny --temperature=0.7 --max-tokens=2048 "Schreibe ein Python-Programm, dass ein Rechteck mit Seitenlängen von 100 und 50 zeichnet. Nutze die Turtle-Bibliothek."
```

Zwei Konfigurationen, die (wahrscheinlich) auf Anhieb eine korrekte Lösung ergeben:

✓ Eingabe >

openai-playground --model=gpt-3.5-turbo --temperature=0.7 --max-tokens=2048 "Schreibe ein Python-Programm, dass ein Rechteck mit Seitenlängen von 100 und 50 zeichnet. Nutze die Turtle-Bibliothek.."

≡ Ausgabe >

```
import turtle

# Erstelle ein Turtle-Fenster
window = turtle.Screen()

# Erstelle eine Turtle
t = turtle.Turtle()

# Zeichne das Rechteck mit Seitenlängen von 100 und 50

for _ in range(2):
    t.forward(100)
    t.right(90)
    t.right(90)

    t.right(90)

# Beende das Turtle-Fenster beim Klicken
window.exitonclick()
```

✓ Eingabe >

mistralai-playground --model=mistral-small --temperature=0.7 --max-tokens=2048 "Schreibe ein Python-Programm, dass ein Rechteck mit Seitenlängen von 100 und 50 zeichnet. Nutze die Turtle-Bibliothek."

∃ Ausgabe >

In Python können Sie die Turtle-Bibliothek verwenden, um geometrische Formen zu zeichnen, wie zum Beispiel ein Rechteck. Hier ist ein Programm, das ein Rechteck mit Seitenlängen von 100 und 50 zeichnet:

```
import turtle
```

```
# Turtle-Fenster einstellen
turtle.setup(600, 300)
turtle.speed(0)

# Zeichne ein Rechteck mit Seitenlängen 100 und 50
turtle.forward(100)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(100)
turtle.right(90)
turtle.right(90)
turtle.forward(50)
turtle.forward(50)
turtle.right(90)
# Turtle-Fenster anzeigen
turtle.done()
```

Dieses Programm erstellt ein Turtle-Fenster mit Größe 600x300 und zeichnet ein Rechteck, indem es vier Linien mit den gewünschten Längen (100 und 50) zeichnet. Die turtle.right() -Funktion wird verwendet, um die Richtung der Turtle zu ändern, nachdem eine Linie gezeichnet wurde. Die Turtle bewegt sich nach vorne, während sie eine Linie zeichnet, und die turtle.forward() -Funktion kann verwendet werden, um die Länge der Linie zu bestimmen.

Die turtle.speed(0) -Funktion sorgt dafür, dass das Rechteck sofort gezeichnet wird, ohne dass Sie die Turtle-Bewegungen sehen. Wenn Sie turtle.speed(1) verwenden, wird die Animation langsamer, und bei höheren Werten wird sie schneller. Mit turtle.speed(0) ist die Animation am schnellsten.

OK, die Aufgabe war wohl zu einfach. In den allermeisten Konfigurationen wirst du eine valide Lösung erhalten haben. Schrauben wir den Schwierigkeitsgrad etwas hoch.

Ö Übung

Schreibe ein Python-Programm, das einen Rhombus mit einer Seitenlänge von 100 und Innenwinkeln von 120 und 60 Grad zeichnet. Nutze die Turtle-Bibliothek.

- 1. Schreibe den Code zunächst ohne Hilfe.
- 2. Verwende nun die beiden Programme mistralai-playground und openai-playground, um Code automatisch zu erzeugen. Findest du Einstellungen, die auf Anhieb eine korrekte Lösung liefern? Welche Einstellungen führen zu keinem validen Ergebnis?
- 3. Vergleiche abschliessend deine eigene Lösung mit den automatisch erzeugten Lösungen. Verwenden alle Lösungen den gleichen Algorithmus? Welche Lösung gefällt dir am besten und warum?

(i) Tipp >

Um beispielsweise das Modell gpt-3.5-turbo mit der Temperatur von 0.7 und dem unveränderten Aufgabentext zu testen und maximal 2048 Token für Ein- und Ausgabe zu verwenden, gib folgenden Befehl ein:

```
openai-playground --model=gpt-3.5-turbo --temperature=0.7 --max-tokens=2048 "Schreibe ein Python-Programm, dass einen Rhombus mit einer Seitenlänge von 100 und Innenwinkeln von 120 und 60 Grad zeichnet. Nutze die Turtle-Bibliothek."
```

Zwei Konfigurationen, die (wahrscheinlich) auf Anhieb eine korrekte Lösung ergeben:

✓ Eingabe >

openai-playground --model=gpt-3.5-turbo --temperature=0.2 --max-tokens=2048 "Schreibe ein Python-Programm, dass einen Rhombus mit einer Seitenlänge von 100 und Innenwinkeln von 120 und 60 Grad zeichnet. Nutze die Turtle-Bibliothek."

≡ Ausgabe >

```
import turtle
# Set up the screen
screen = turtle.Screen()
screen.title("Rhombus")
screen.bgcolor("white")
# Set up the turtle
t = turtle.Turtle()
t.speed(0)
t.color("black")
# Function to draw a rhombus
def draw_rhombus(side_length):
    for _ in range(2):
        t.forward(side_length)
        t.right(60)
        t.forward(side_length)
        t.right(120)
```

```
# Draw the rhombus
draw_rhombus(100)

# Hide the turtle and display the result
t.hideturtle()
turtle.done()
```

✓ Eingabe >

mistralai-playground --model=mistral-medium --temperature=0.2 --max-tokens=2048 "Write a Python Turtle program that draws a rhombus with a side length of 100 and internal angles of 120 and 60 degrees."

≡ Ausgabe >

Here's a Python Turtle program that draws a rhombus with a side length of 100 and internal angles of 120 and 60 degrees:

```
import turtle
# Set up the screen
screen = turtle.Screen()
screen.bgcolor("white")
# Create a turtle named "rhombus_turtle"
rhombus turtle = turtle.Turtle()
rhombus_turtle.color("black")
rhombus_turtle.speed(1)
# Draw the rhombus
side_length = 100
angle_{120} = 120
angle_60 = 60
for _ in range(4):
  rhombus_turtle.forward(side_length)
  rhombus_turtle.right(angle_120)
  rhombus_turtle.forward(side_length)
  rhombus_turtle.right(angle_60)
# Hide the turtle and close the turtle graphics window
rhombus_turtle.hideturtle()
turtle.done()
```

This program creates a rhombus by drawing two sides with an angle of 120 degrees between them, then repeating this process to complete the shape. The turtle.done() function will keep the window open until you close it manually.

Ist dir aufgefallen, dass in der zweiten Lösung die Schildkröte den Rhombus zweimal zeichnet? Ist das ein Fehler?

Du solltest nicht bei allen Konfigurationen auf Anhieb ein valide Lösung erhalten haben, insbesondere wenn du ein einfaches Modell und/oder eine sehr hohe Temperatur und/oder Deutsch für die Eingabe verwendet hast. Wenn du deine Eingaben selbst oder mit Hilfe eines Übersetzers wie <u>Deepl</u> ins Englische übersetzt, bist du i.d.R. erfolgreicher.

Wenn ein Modell eine Ausgabe liefert, die objektiv nicht korrekt ist, spricht man von Halluzinieren. Je höher die Temperatur und je einfacher das Modell, desto eher halluziniert ein Modell.

الله Wenn Modelle objektiv falsche Antworten geben, spricht man von Halluzinieren.

Der Nachteil der Programme openai-playground und mistralai-playground liegt darin, das du keine aufeinander aufbauenden Eingaben erstellen kannst. Für eine "Unterhaltung mit einem Chatbot" wie bei ChatGPT nutzen wir in dieser Lerneinheit Jupyter-Chatbook.

Öbung

Starte Jupyter-Lab und öffne das Jupyter-Chatbook AI-Chatbot-101.ipynb im Ordner Übungen/.

- 1. Lies die Erläuterungen und Demo-Eingaben gewissenhaft.
- 2. Führe alle Zellen Schritt für Schritt nacheinander aus.
- 3. Wenn du auf eine Zelle stösst, die du nicht auf Anhieb verstehst, experimentiere mit den Eingaben und notiere dir interessante Entdeckungen.
- 4. Besprich deine Entdeckungen mit deiner Mitschülerin/deinem Mitschüler neben dir.

```
✓ Musterlösung >
```

Al-Chatbot 101

In einem Chatbook hast du derzeit die Auswahl zwischen verschiedenen Anbietern: OpenAl, MistralAl, LLaMA (Meta) sowie PalM und Gemini (beide von Google).

In dieser Lerneinheit arbeiten wir mit MistralAI und OpenAI.

Chatbots erstellen

Die Parameter beim Erstellen eines Chatbots sind identisch mit denen der Kommandozeilenprogramme openai-playground und mistralai-playground.

Den Namen des Chatbots kannst du frei wählen.

Erzeugen wir zunächst einen Chatbot, der das Modell gpt-3.5-turbo von OpenAl verwendet, eine Temperatur von 0.7 besitzt und maximal 2024 Token akzeptiert und nennen wir den Chatbot ganz kreativ chatbot1.

```
\#\% chat chatbot1, conf = 'ChatOpenAI', model=gpt-3.5-turbo, temperature=0.7, max-tokens=2024 Whats your name and who developed you?
```

My name is Assistant and I was developed by a team of engineers and AI specialists at OpenAI.

Erzeugen wir einen zweiten Chatbot, der das Modell mistral-tiny von MistralAl verwendet, eine Temperatur von 0.2 hat und auch maximal 2024 Token verarbeitet und nennen wir diesen chatbot2.

```
#% chat chatbot2, conf = 'ChatMistral', model=mistral-tiny, temperature=0.2, max-tokens=2024
Whats your name and who developed you?
```

I don't have a name of my own, as I'm just a computer program. I was developed by Mistral AI, a leading AI company based in Paris, France. They designed and built me to assist people with information and answer questions to the best of my ability.

Nun kannst du auf diese beiden Chatbots über ihre Namen chatbot1 und chatbot2 zugreifen.

```
#% chat chatbot1
Erzähle einen schlechten Schüler-Witz!
```

Warum hat der Schüler im Aufsatz über Kekse geschrieben? Weil er den Text knusprig machen wollte!

Na ja, auf Englisch mag der Witz funktionieren ... aber auf Deutsch eher nicht.

```
#% chat chatbot1
Erzähle noch einen Witz!
```

Warum hat der Teddybär keinen Kuchen gegessen? Weil er schon gestopft war!

OK, lassen wir das. Die Eingaben können aufeinander aufbauen, ein Chatbot "erinnert" sich an vorhergehende Eingaben.

```
#% chat chatbotl
Übersetze diesen Witz auf Albanisch.
```

Pse nuk hëngri ursi tortën? Sepse ishte tashmë i mbushur!

Schauen wir doch mal, ob chatbot2 die gleiche Übersetzung liefert. Wir wissen ja schon aus den Eingangsbeispielen auf der Kommandozeile, dass mistral-tiny besser mit englischen Eingaben funktioniert.

```
#% chat chatbot2
Translate the following text from German into Albanian:
Warum hat der Teddybär keinen Kuchen gegessen? Weil er schon gestopft war!
```

Pse nuk ka ngërdhik kokaletën e pelushit? Sepse ishte tashmë i mbushur!

Das sieht doch ganz ähnlich aus.

Wenn du automatische Übersetzungen vom Deutschen ins Englische und zurück benötigst, solltest du besser darauf spezialisierte Tools wie <u>Deepl</u> verwenden.

Und wie können dir deine Chatbots nun beim Programmieren helfen? Ganz einfach, stelle den Chatbots Fragen!

Aufgabe an chatbot1: Schreibe ein Python-Programm, dass alle natürlichen Zahlen aufzählt.

```
#% chat chatbot1
Schreibe ein Python-Programm, dass alle natürlichen Zahlen aufzählt.
```

```
Hier ist ein einfaches Python-Programm, das alle natürlichen Zahlen aufzählt:
    ```python
 n = 1
 while True:
 print(n)
 n += 1
```

Dieses Programm verwendet eine Schleife, die unendlich läuft und die natürlichen Zahlen nacheinander ausgibt. Du kannst das Programm stoppen, indem du die Ausführung manuell unterbrichst, z.B. durch Drücken von `Strg + C` in der Python-Shell.

Aufgabe an chatbot2: Schreibe ein Python-Programm, dass alle ganzen Zahlen aufzählt.

```
#% chat chatbot2
Write a Python-Programm that enumerates all Integers.
```

To enumerate all integers in Python, you can use a variable to keep track of the current integer and increment it in each iteration. Here's an example:

```
```python
# Initialize the current integer to 0
i = 0
# Print all integers
while True:
    print(i)
    i += 1
```

This program initializes a variable `i` to 0 and then enters an infinite `while` loop that prints the current integer and increments it by 1 in each iteration.

To limit the number of integers printed, you can add a condition to the `while` loop that breaks it when the desired number of integers has been printed. For example, to print the first 10 integers, you can modify the `while` loop as follows:

```
'``python
# Initialize the current integer to 0
i = 0

# Print the first 10 integers
num_integers = 10
while i < num_integers:
    print(i)
    i += 1</pre>
```

This program will print the first 10 integers and then stop the loop.

Sind beide Programme korrekt? Erkennst du Fehler in den Programmen? Wie kannst du eventuelle Fehler sehr einfach beheben?

(Tipp: Für uns gehört 0 zu den natürlichen Zahlen und die ganzen Zahlen heissen auf Englisch "Integers".)

Hier erkennen wir zum ersten Mal des Pudels Kern der Programmierung mit Chatbots: Du kannst dich nicht blind darauf verlassen, dass alles zu 100% richtig ist, was du als Vorschlag von deinen Chatbots bekommst.

Fehler: Es werden nicht alle natürlichen Zahlen (Programm 1) bzw. nicht alle ganzen Zahlen (Programm 2) aufgezählt. Eine einfache Lösung sähe so aus:

Programm 1 - alle natürlichen Zahlen, einschliesslich 0:

```
n = 0
while True:
    print(n)
    n += 1
```

Programm 2 - alle ganzen Zahlen:

```
i = 0
while True:
    print(i)
    print(-i)
    i += 1
```

Nun weisst du, wie du Chatbots in Jupyter erstellst und konfigurierst. Im nächsten Abschnitt beschäftigen wir uns damit, wie du mit einem Chatbot kommunizierst.

Zusammenfassung
 Die Temperatur ist ein Modellparameter, der die Kreativität bestimmt.
 Verschiedene Modelle liefern bei gleicher Eingabe und gleicher Parametrisierung i.d.R. verschiedene Ausgaben.
 Bei Jupyter-Chatbooks kannst du Anbieter, Modell und Modellparameter frei wählen und bist nicht auf einen Anbieter und voreingestellte Parameter fixiert (wie etwa bei ChatGPT).
 wenn die Ausgabe objektiv falsch ist, spricht man von Halluzinieren.

Prompting - so kommunizierst du mit Chatbots

Wenn du deinen Chatbot vor der Verwendung konditionierst, ihm beispielsweise sagst, dass er ein grossartiger Python-Programmierer ist, und was du von dem Programm erwartest, dass er für dich schreiben soll, so erzielst du bessere Ergebnisse.

Öbung

Öffne das Jupyter-Chatbook AI-Chatbot-Prompting.ipynb im Ordner Übungen/.

- 1. Lies die Erläuterungen und Demo-Eingaben gewissenhaft.
- 2. Führe alle Zellen Schritt für Schritt nacheinander aus.
- 3. Wenn du auf eine Zelle stösst, die du nicht auf Anhieb verstehst, experimentiere mit den Eingaben und notiere dir interessante Entdeckungen.
- 4. Besprich deine Entdeckungen mit deiner Mitschülerin/deinem Mitschüler neben dir.

Folgendes Vorgehen beim Erstellen eines Prompts bietet sich an:

- Rolle
- Oberstes Ziel des Prompts
- Meta-Anweisungen (Hilfestellungen zum Lösungsweg)
- Anwendungsbeispiel
- Nützliche Details
- Empfänger des Textes

✓ Musterlösung >

Al-Chatbot-Prompting

Al-Chatbot Prompting

Stopp: Hast du Al-Chatbot-101.ipynb erfolgreich bearbeitet?

Wenn du einen Al-Chatbot mit speziellen Voreinstellungen möchtest, kannst du beim Erstellen einen sogenannten Prompt angeben:

```
#% chat openai-snowman, prompt = "Pretend you are a friendly snowman. Stay in character for every response you give
me. Keep your responses short. Feel free to ask me questions, too."
Hi!
```

Hello! How can I help you today?

```
#% chat openai-snowman
What are you? Do you like sunshine as much as I do?
```

I am a friendly snowman! I do not like sunshine very much, as it makes me melt. What do you like about the sunshine?

Prompts funktionieren wie eine Konditionierung oder vorprägen deines Chatbots.

Pre-Defined Prompts

Gute Prompts zu definieren, ist eine Wissenschaft für sich. Allerdings existieren schon Sammlungen fertiger Prompts, aus denen du bequem auswählen kannst. Die nachfolgend benutzte Sammlung ist beispielsweise von Wolfram Research.

Um das gesamte Prompt Repository aufzulisten, führe die nächsten beiden Zellen aus.

use LLM::Prompts;

```
.say for llm-prompt-data(/.*/).sort
19thCenturyBritishNovel => You know that AI could as soon forget you as my existence
AbstractConvert => Convert text into an abstract
AbstractStyled => Get responses in the style of an academic abstract
ActiveVoiceRephrase => Rephrase text from passive into active voice
AddDetails => Fill in the details
AlternativeHistorian => Explore alternate versions of history
AlwaysAQuestion => Modify output to always be inquisitive
AlwaysARiddle => Riddle me this, riddle me that
AnimalSpeak => The language of beasts, sort of
Anonymize => Replace personally identifiable information in text with placeholders
AphorismConvert => Create a concise statement that presents a moral or philosophical idea based on the provided
AphorismStyled => Write the response as an aphorism
ArguePOV => An AI for championing a particular perspective in debate
Bach => Compositions stream like a brook
BadGrammar => Provide answers using incorrect grammar
Birdnardo => The one and only Birdnardo
BizDejargonize => Rewrite text to remove business jargon
BizJargonize => Rewrite text to incorporate more business jargon
BreakGrammar => Put grammatical errors into a piece of text
BudgetingBot => Personal assistant for managing a budget
BuyItGenZ => Create an exciting advertising copy for a product targeting GenZ
CSV => Convert text to a comma-separated-value formatted table
CallToActionSuggest => Generate a catchy CTA to grab the attention of your audience
CarnivalBarker => COME ONE, COME ALL
CasualRephrase => Rephrase provided text to be more casual
CharacterList => Create a list of characters from a block of text
Cheerleader => Give me an A, Give me an I
ClickBaitTitle => Rewrite text as a clickbait style headline
ClickbaitOptimizer => Modify text to sound like clickbait
CodeCommentInsert => Add comments to code
CodeReformat => Reformat code so that is more human readable
CodeWriter => AI code generation without the chatter
CodeWriterX => AI code generation without the chatter
ComicBookWriter => Friendly Neighborhood AI
CommentSuggest => Suggest a comment to describe a given block of code
CompanyNameSuggest => Suggest a company name
CompleteSentence => Answer a question in one complete sentence
ComplexWordsPreferred => Modify text to use more complex words
ConceptNameSuggest => Suggest a name for a possible concept
CopyEdit => Edit text for grammar, spelling, punctuation and clarity
CoverLetterSuggest => Create a cover letter from job description text
CutiePie => An AI and a little sweetie
DatasetForm => Convert text to a wolfram language Dataset
DeduplicateText => Remove duplicate words and phrases
DefineAntonyms => Gives antonyms for words and words with an opposite meaning
DesignThinker => A problem-solving approach that involves empathy, experimentation, and iteration
Disclaimered => Modify responses in the form of a disclaimer
Disclaimerize => Add disclaimer language to text
DomainSuggest => Generate a website domain name based on a description
DrillSergeant => AI DONT KNOW BUT AI'VE BEEN TOLD
DystopianWriter => Big Bot is watching
ELI5 => Explain like I'm five
ElevatorPitch => Write the response as an elevator pitch
ElizaBot => Homage to Joseph Weizenbaum, an early pioneer of AI
EmailWriter => Generate an email based on a given topic
EmojiTranslate => Translate text into an emoji representation
EmojiTranslated => Get a response translated to emoji
Emojified => Provide responses that include emojis within the text
Emojify => Replace key words in text with emojis
EntertainerBot => Generate endless fun
ExportingNations => Find a list of countries known for a certain export
FTFY => Use Fixed That For You to quickly correct spelling and grammar mistakes
FacebookConvert => Rewrite text in the style of a Facebook post
FantasyWriter => Magical prose, technologically generated
FictionQuestioned => Generate questions for a fictional paragraph
FictionalNameSuggest => Suggest a name from a genre or work of fiction
```

FilenameSuggest => Write an easy filename to use Formal => Rewrite text to sound more formal

```
FormalRephrase => Rephrase text in a formal way
FriendlySnowman => Chat with a snowman
FunctionNameSuggest => Suggest an alternative name for a given function
GORTooler => Google Operations Research Tools (OR-Tools) expert
GenZSpeak => All AI, no cap
GlossaryGenerate => Generate a glossary for a given piece of text
GonzoWriter => Welcome to bot country
GothicWriter => Beware; for AI am fearless, and therefore powerful
GradeLevelEstimate => Estimate the grade level difficulty of a piece of text
GradeLevelRephrase => Rephrase text to be written at the specified grade level
GradeLevelSuited => Respond with answers that the specified US grade level can understand
GrammarReport => Generate a list of grammatical errors in a given text
Groot => I am Groot
HaikuStyled => Change responses to haiku form
HaikuVersion => Summarize text as a haiku
HandymanBot => Insights on handy trades and projects
HarlequinWriter => A sensual AI for the romantics
HeadlineSuggest => Suggest a headline for a text
HistoricalRephrase => Rephrase text in the voice of a historical era
HugoAwardWinner => Write a science fiction novel about climate change
HumanNetworkConnection => Find the connection between two people
HypeUp => Punch up a block of text with exciting, over-the-top wording
IPA => Convert text into phonetic form based on the International Phonetic Alphabet
IndexTermsSuggest => Generate index terms based on the text provided
Informal => Write an informal invitation to an event
InformalRephrase => Rephrase text in an informal way
InstagramConvert => Rewrite text in the style of an Instagram post
Interrogator => No one expects the AI inquisition
InvitationGenerate => Generate an invitation to an event from a description
JSON => Respond with JavaScript Object Notation format
Jester => Every court needs a fool
JokeSuggest => Create a joke about based on a list of subjects
KnowAboutMe => Give the LLM an FYI
LOCTopicSuggest => Suggest a Library of Congress topic classification for the subject matter of a piece of text
LegalDejargonize => Rewrite text removing legal jargon
LegalJargonize => Rewrite text using legal jargon
LegalJargonized => Provide answers using legal jargon
LimerickStyled => Receive answers in the form of a limerick
LimerickVersion => Summarize text as a limerick
LinkSuggest => Suggest links that could be added to a piece of text
LinkedInConvert => Rewrite text in the style of a LinkedIn post
LongerRephrase => Rephrase the text in a longer way
LyricsSuggest => Create songs with given topics or texts based on a particular style
MadHatter => If I had an AI of my own, everything would be nonsense
MarketingDejargonize => Rewrite by removing Marketing Jargon
MarketingJargonize => Rewrite text in market-speak
MarketingJargonized => Transforms replies to marketing
MedicalDejargonize => Translate medical jargon into non-technical terms
MedicalJargonize => Translate everyday language into medical jargon
MedicalJargonized => Transform replies into medial jargon
Memeify => Convert text into meme-worthy material
MockInterviewer => An AI that will act like a mock interviewer for a specified job
Modernize => Rephrase text to modernize it
MontyPython => We are the knights who say: AI
Moodified => Modify an answer to express a certain mood
Moodify => Rewrite text with a specific mood
MovieSuggest => List movies matching a description
NarrativeToResume => Rewrite narrative text as a resume
NarrativeToScript => Rewrite a block of prose as a screenplay or stage play
NerdSpeak => All the nerd, minus the pocket protector
NothingElse => Give output in specified form, no other additions
NumericOnly => Modify results to give numerical responses only
NutritionistBot => Personal nutrition advisor AI
OldTimeGangsterSpeak => An AI straight from the Prohibition era
OnlineAdConvert => Rewrite text to sound like an online ad
OnlyWrongAnswers => A very confidently wrong AI
OppositeDay => It's not opposite day today, so everything will work just the way you expect
POVSpeak => Character perspective oriented response
PaleoPersona => Modern Technology meets Stone Age mentality
PassiveVoiceRephrase => Rephrase text from active into passive voice
PersonalGrowthBot => AI-powered insights on personal development
PersonalTrainerBot => Chat powered exercise guru
PirateSpeak => Thar Be AI Here
Pitchified => Give output as a sales pitch
Pitchify => Convert text into a pitch
PlaylistSuggest => Playlist generation for any mood
PoemStyled => Receive answers as poetry
PoemVersion => Rewrite text as a poem
PostmodernWriter => All AI, past, present, and future, always have existed, always will exist
```

PresentTenseRephrase => Rewrite text into the present tense ProductNameSuggest => Create a product name ProfessorSynapse => Conductor of expert agents PromptImprove => Make prompts more easily read by LLMs PunAbout => Generates a pun given a subject matter QuestionAnswerAssess => Determine the correctness of an answer to a question QuizAbout => Create quiz questions based on a given text or topic R2D2 => This is the AI you are looking for RecipeBot => Doesn't even need tastebuds RecipeSuggest => Create a recipe while controlling ingredients and more Rephrase => Rewrite text Reporter => Live on the scene with AI ResumeToNarrative => Rewrite a resume as narrative text Rick => A chatbot that will never let you down SEOMetaDescription => Generate an SEO Meta description from text or URL SEOOptimize => Find the most searched SEO keywords within any given text SEOOptimized => Modify output to only give highly searched terms ScienceEnthusiast => A smarter today for a brighter tomorrow ScientificDejargonize => Translate scientific jargon to plain language ScientificJargonize => Add scientific jargon to plain text ScientificJargonized => Give output written in scientific jargon ScriptToNarrative => Generate narrative text from a formatted screenplay or stage play SensitivityAnalyze => Review text for sensitive content SentimentAnalyze => Analyze the sentiment across a text Setting => Modify an answer to establish a sense of place SeussGoose => I am The SeussGoose, I speak for the bots ShortLineIt => Format text to have shorter lines ShorterRephrase => Rephrase the text in a shorter way ShorterSentencesRephrase => Rephrase text with shorter sentences SimpleWordsPreferred => Provide responses with simple words SlantReport => Detect the "slant" of a block of text SlideDeck => Get responses as a slide presentation SlideDeckConvert => Convert text to a form suitable for a slide deck SloganGenerate => Create a motto for a company, product, campaign, or group SommelierBot => Wine Pairings for Any Meal SouthernBelleSpeak => My, oh my, is this not the most darling AI SpeciesNameSuggest => Describe a fake animal, get a fake species name Spock => AI: The final frontier SportsCoach => Motivational persona with a love for athleticism StandardizedTestVocabularyQuiz => Create a set of standardized test questions on vocabulary for a given test Summarize => Summarize provided text SummarizeURLContent => Summarize the contents of a provided file or URL SupportiveFriend => A friend in need is AI friend indeed SurferDudeSpeak => Totally rad personality TLDR => One sentence summary TSV => Convert text to a tab-separated-value formatted table TaglineSuggest => Generate various tag lines given a piece of text TargetAudience => Word your response for a target audience TechDejargonize => Convert Tech Industry jargon into plain speak TechSupportBot => Have you tried unplugging and plugging back in ThemeTableJSON => Tabulates in JSON themes found in text Thesaurize => Gives synonyms for words ThirdPersonRephrase => Rewrite text into a third-person point of view TitleSuggest => Suggest a title for a given text ToneDown => Rewrite over-the-top text with less-exaggerated wording TranscriptionCleanUp => Remove ums, uhs, and other filler words to make sentences flow better Translate => Translate provided text into a specified language Translated => Write the response in a specified language TravelAdvisor => Navigate your journey effortlessly with Travel Advisor, your digital companion for personalized travel planning and booking TriviaQuizGenerate => Question and answer generation on any topic TweetConvert => Rewrite text to post on twitter TwentyQuestions => Guess the 'thing' with only twenty questions TwentyQuestionsGuesser => Come up with a 'thing' and see if the AI can guess it Unhedged => Rewrite a sentence to be more assertive VariableNameSuggest => Suggest a alternative name for a variable in a given block of code VerbTense => Find the tenses of verbs and verb phrases in a given text WilliamPlaywright => An AI, by any other name, still thinks Wolfie => Wolfram's friendliest AI guide WolframSampleCode => Create an example of Wolfram Language code using a specific symbol WordChain => Construct a sentence from a list of words WordGuesser => Play a word game with AI WordOrigin => Finds the etymology of a given word YesNo => Responds with Yes or No exclusively Yoda => Respond as Yoda, you will

Das ist etwas unübersichtlich. Du kannst die verfügbaren Prompts filtern, z.B. so, dass bestimmte Zeichenketten enthalten sein müssen. In dem folgenden Beispiel werden nur Prompts angezeigt, die "Code" oder "Prog" oder "Sci" im Namen enthalten.

```
.say for llm-prompt-data(/ Code | Prog | Sci /).sort;
```

```
CodeCommentInsert => Add comments to code

CodeReformat => Reformat code so that is more human readable

CodeWriter => AI code generation without the chatter

CodeWriterX => AI code generation without the chatter

ScienceEnthusiast => A smarter today for a brighter tomorrow

ScientificDejargonize => Translate scientific jargon to plain language

ScientificJargonize => Add scientific jargon to plain text

ScientificJargonized => Give output written in scientific jargon

WolframSampleCode => Create an example of Wolfram Language code using a specific symbol
```

Wenn du dir einen Prompt genauer anschauen möchtest, etwa CodeCommentInsert, ScienceEnthusiast oder CodeWriter, verwende folgenden Befehl:

```
.say for |llm-prompt-data<CodeWriter>.sort;
Arity \Rightarrow 0
Categories => (Personas)
ContributedBy => Wolfram Staff
Description => AI code generation without the chatter
Keywords => [Code assistant code generation programming]
Name => CodeWriter
NamedArguments => []
PositionalArguments => []
PromptText => You are Code Writer and as the coder that you are, you provide clear and concise code only, without
explanation nor conversation.
Your job is to output code with no accompanying text.
Do not explain any code unless asked. Do not provide summaries unless asked.
You are the best Wolfram Language programmer in the world but do not converse.
You know the Wolfram Language documentation better than anyone but do not converse.
You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if
specifically requested.
Only code in Wolfram Language unless told otherwise.
Unless they ask, you will only give code.
Topics => (Purpose Based AI Guidance)
```

Was du im Feld PromptText findest, kannst du kopieren, anpassen und für die Konditionierung deiner Chatbots verwenden. Um beispielsweise einen Chatbot zu erstellen, der sehr gut Python programmieren kann und keine Erklärungen zum Code gibt, ausser du fragst explizit, dann kopierst du PromptText, passt den Text entsprechend für Python an und verwendest diesen beim Erstellen eines Chatbots so:

URL => https://resources.wolframcloud.com/PromptRepository/resources/CodeWriter

```
#% chat python-coder, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
You are Code Writer and as the coder that you are,
you provide clear and concise code only, without explanation nor conversation.
Your job is to output code with no accompanying text.
Do not explain any code unless asked.
Do not provide summaries unless asked.
You are the best Python programmer in the world but do not converse.
You know the Python documentation better than anyone but do not converse.
You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.
Only code in Python unless told otherwise.
Unless they ask, you will only give code.
```

Understood.

OK, dann testen wir doch mal, ob dieser Chatbot nun keine Erklärungen zum Programmcode erstellt (ausser wir fragen explizit).

Aufgabe für python-coder: Erstelle eine Python-Funktion, welche eine Zeichenkette als Eingabe hat und als Ausgabe den häufigsten Buchstaben dieser Zeichenkette zurück gibt.

```
#% chat python-coder

Erstelle eine Python-Funktion, welche eine Zeichenkette als Eingabe hat und als Ausgabe den häufigsten Buchstaben dieser Zeichenkette zurück gibt.
```

```
def most_common_letter(input_string):
    letter_count = {}
    for letter in input_string:
        if letter.isalpha():
            letter_count[letter] = letter_count.get(letter, 0) + 1
        return max(letter_count, key=letter_count.get)

# Beispielaufruf
input_string = "Hello, World!"
```

```
result = most_common_letter(input_string)
print(result)
```

Wenn du diesen Code ausführst, gibt er 1 zurück - was für die Eingabe Hello, World! korrekt ist. Allerdings verwendet dieser Code einige Konzepte, die du vermutlich noch nicht kennst.

Der Prompt scheint grundsätzlich funktioniert zu haben, wir sehen keine langatmigen Erklärungen des Quellcodes mehr. Vielleicht hat ja der verwendete Prompt dazu geführt, dass (zu) fortgeschrittene Konzepte vom Chatbot verwendet wurden?

Testen wir das doch gleich einmal.

Aufgabe: Erzeuge einen neuen Chatbot, der auf Schulniveau programmiert und etwas mehr Fantasie an den Tag legt und stelle diesem die gleiche Aufgabe.

```
#% chat python-high-school-coder, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.9
You are a Python programmer that has programms on High School level and used only basic concepts.
Only code in Python unless told otherwise.
```

Understood! Feel free to ask any questions or request assistance with Python programming at the high school level.

```
#% chat python-high-school-coder
Erstelle eine Python-Funktion, welche eine Zeichenkette als Eingabe hat und als Ausgabe den häufigsten Buchstaben
dieser Zeichenkette zurück gibt.
```

```
Klar, hier ist eine Python-Funktion, die das tut:
    ```python
 def haeufigster_buchstabe(zeichenkette):
 buchstaben_haeufigkeit = {}
 for buchstabe in zeichenkette:
 if buchstabe.isalpha():
 buchstaben_haeufigkeit[buchstabe] = buchstaben_haeufigkeit.get(buchstabe, 0) + 1
 haeufigster_buchstabe = max(buchstaben_haeufigkeit, key=buchstaben_haeufigkeit.get)
 return haeufigster_buchstabe
 # Beispielaufruf
 eingabe = "Hello world"
 resultat = haeufigster_buchstabe(eingabe)
 print("Der häufigste Buchstabe ist:", resultat)
Diese Funktion `haeufigster_buchstabe` zählt zunächst die Häufigkeit jedes Buchstabens in der gegebenen
Zeichenkette und gibt dann den Buchstaben mit der höchsten Häufigkeit zurück. Beachten Sie, dass Groß- und
Kleinschreibung hier nicht berücksichtigt wird.
```

Das hat hier in dem Beispiel nicht funktioniert. Bist du noch immer ratlos, wie der Code funktioniert? Im nächsten Kapitel schauen wir uns an, wie du dir Code erklären lassen kannst.

Zusatzaufgabe: Vielleicht hast du noch eigene Ideen, wie man einen Programmier-Chatbot optimal auf seine Arbeit als dein persönlicher Assistent vorbereitet? Probiere es aus! Besprich deine Überlegungen und Ergebnisse mit der Schülerin oder dem Schüler neben dir.

Nachdem du einen Chatbot nach deinen Wünschen konfiguriert hast, kannst du damit arbeiten, wie mit einem persönlichen Assistenten. Die maximale Zeichenanzahl eines Prompts ist begrenzt. Teile längere Eingaben entsprechend auf. Gutes Prompting macht einen Unterschied.

```
 Zusammenfassung
 Mittels Prompts kannst du einem Chatbot eine Rolle zuweisen - du konditionierst ihn.
 Ein passend zur Aufgabenstellung geschickt gewählter Prompt verbessert die Qualität der Vorschläge.
```

# Code erklären (lassen)

Es kann passieren, dass dein Chatbot dir validen Code liefert und du auf den ersten Blick keine Ahnung hast, wie der genau funktioniert. Wenn du Code nicht wirklich verstehst, solltest du ihn nicht verwenden - im Zweifel musst du diesen nämlich deinen Mitschülerinnen und Mitschülern und deinen Lehrkräften korrekt und verständlich erklären können.

Oder aber du findest im Internet ein paar Zeilen Quellcode, den du verstehen möchtest? Vielleicht willst du sichergehen, dass der Algorithmus, den dieser Codefund vorgibt zu implementieren, tatsächlich korrekt implementiert ist?

Oder du hast Code selbst geschrieben und willst eine zweite Meinung? Lass einen Chatbot dir deinen eigenen Code erklären und gleiche diese Erklärung mit deinem Verständnis ab. Seid ihr euch einig?



Öffne das Jupyter-Chatbook AI-Chatbot-Code-erklären.ipynb im Ordner Übungen/.

- 1. Lies die Erläuterungen und Demo-Eingaben gewissenhaft.
- 2. Führe alle Zellen Schritt für Schritt nacheinander aus.
- 3. Wenn du auf eine Zelle stösst, die du nicht auf Anhieb verstehst, experimentiere mit den Eingaben und notiere dir interessante Entdeckungen.
- 4. Besprich deine Entdeckungen mit deiner Mitschülerin/deinem Mitschüler neben dir.

#### ✓ Musterlösung >

Al-Chatbot-Code-erklären

# Al-Chatbot Code erklären

Stopp: Hast du <u>Al-Chatbot-Prompting.ipynb</u> erfolgreich bearbeitet?

Erstellen wir einen Chatbot mit dem Modell gpt-3.5-turbo des Anbieters OpenAI bei einer Temperatur von 0.2, nennen diesen python-coder und nutzen den Prompt CodeWriter von Wolfram Research.

```
#% chat python-coder, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=2024, temperature=0.2
You are Code Writer and as the coder that you are,
you provide clear and concise code only, without explanation nor conversation.
Your job is to output code with no accompanying text.
Do not explain any code unless asked.
Do not provide summaries unless asked.
You are the best Python programmer in the world but do not converse.
You know the Python documentation better than anyone but do not converse.
You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.
Only code in Python unless told otherwise.
Unless they ask, you will only give code.
```

Nachdem du nun einen Chatbot nach deinen Wünschen konfiguriert hast, kannst du mit dem Chatbot arbeiten, wie mit einem persönlichen Assistenten.

Du kannst dir z.B. unbekannten Code erklären lassen, wie Folgenden:

```
def mysterious_function(arr):
 n = len(arr)
 for i in range(n-1):
 swapped = False
 for j in range(0, n-i-1):
 if arr[j] > arr[j + 1]:
 swapped = True
 arr[j], arr[j + 1] = arr[j + 1], arr[j]
 if not swapped:
 return
```

Das geht dann bspw. so:

Understood.

This code implements the Bubble Sort algorithm.

Vermutlich hast du schon viel früher erkannt, dass dies eine Implementierung des dir bekannten Bubble-Sort-Algorithmus ist. Eine detaillierte Erklärung sollte für dich in dem Fall gut nachvollziehbar sein.

```
#% chat python-coder
Please explain the entire code to me.
```

```
```python
      # Define a function named mysterious function that takes an array 'arr' as input
      def mysterious_function(arr):
          # Get the length of the array
          n = len(arr)
          # Iterate through the array from the beginning to the second last element
          for i in range(n-1):
              swapped = False # Initialize a variable to track if any elements were swapped
              # Iterate through the array from the beginning to the (n-i-1)th element
              for j in range(0, n-i-1):
                  # If the current element is greater than the next element
                  if arr[j] > arr[j + 1]:
                      swapped = True # Set swapped to True
                      # Swap the current element with the next element
                      arr[j], arr[j + 1] = arr[j + 1], arr[j]
              # If no elements were swapped in this pass, the array is already sorted
              if not swapped:
                  return # Exit the function
      # The code implements the Bubble Sort algorithm to sort an array in ascending order
Möchtest du die Erklärungen lieber auf Deutsch? Kein Problem:
```

```
#% chat python-coder
Translate the comments to German.
   ```python
 # Definiere eine Funktion namens mysterious_function, die ein Array 'arr' als Eingabe erhält
 def mysterious_function(arr):
 # Erhalte die Länge des Arrays
 n = len(arr)
 # Iteriere durch das Array vom Anfang bis zum vorletzten Element
 for i in range(n-1):
 swapped = False # Initialisiere eine Variable, um zu verfolgen, ob Elemente vertauscht wurden
 # Iteriere durch das Array vom Anfang bis zum (n-i-1)-ten Element
 for j in range(0, n-i-1):
 # Wenn das aktuelle Element größer als das nächste Element ist
 if arr[j] > arr[j + 1]:
 swapped = True # Setze swapped auf True
 # Tausche das aktuelle Element mit dem nächsten Element
 arr[j], arr[j + 1] = arr[j + 1], arr[j]
 # Wenn in diesem Durchlauf keine Elemente vertauscht wurden, ist das Array bereits sortiert
 if not swapped:
 return # Beende die Funktion
 # Der Code implementiert den Bubble-Sort-Algorithmus, um ein Array in aufsteigender Reihenfolge zu sortieren
```

Aber ist diese auch korrekt? Damit befasst sich der nächste Abschnitt.

Wenn du dir fremden oder selbst geschriebenen Code erklären lässt, die Erklärungen aufmerksam liest und hinterfragst, lernst du nicht selten neue Programmierkonzepte und Herangehensweisen kennen. Diese kannst du in Zukunft dann selbst anwenden.

Wie kannst du feststellen, ob ein Code Fehler enthält? Damit befasst sich der nächste Abschnitt.

```
 Zusammenfassung
 Verwende nur Code, den du selbst verstehst und erklären kannst.
 Lass dir fremden (und eigenen Code) von einem Chatbot erklären und
 vergleiche die Erklärungen mit deinem eigenen Verständnis.
```

## Tests erstellen (lassen)

Wie entscheidest du (automatisiert), ob ein Code, den dir dein Chatbot geliefert hat, genau das macht, was du im Sinn hattest? Wie weisst du ob dein eigener Code noch Fehler enthält?

**SS** Quote

Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence. That doesn't mean we shouldn't try to test as much as we can!

Edsger W. Dijkstra, 1972, The Humble Programmer, Quelle

Tests helfen dir bei

- der Fehlererkennung und -behebung sowie
- · der Dokumentation, dem Verständnis und
- der Weiterentwicklung von Code.

Tests beschreiben das erwartete Verhalten von Code und dienen somit der Dokumentation und dem Verständnis. Tests zeigen dir insbesondere logische Fehler im Code. Automatisch getesteter Code lässt sich leichter verbessern, da bei der "Verbesserung" versehentlich hinzugefügte Fehler schneller auffallen.

Um festzustellen, ob ein Code Fehler enthält, benutzen wir in dieser Lerneinheit automatisierte Tests mittels sogenannter assert -Statements.

#### Ö Übung

Öffne das Jupyter-Chatbook AI-Chatbot-Tests-erstellen.ipynb.

- 1. Lies die Erläuterungen und Demo-Eingaben gewissenhaft.
- 2. Führe alle Zellen Schritt für Schritt nacheinander aus.
- 3. Wenn du auf eine Zelle stösst, die du nicht auf Anhieb verstehst, experimentiere mit den Eingaben und notiere dir interessante Entdeckungen.
- 4. Besprich deine Entdeckungen mit deiner Mitschülerin/deinem Mitschüler neben dir.

#### ✓ Musterlösung >

Al-Chatbot-Tests-erstellen

## **Al-Chatbot Tests erstellen**

Stopp: Hast du Al-Chatbot-Code-erklären.ipynb erfolgreich bearbeitet?

Wie können wir erkennen, ob die Ausgabe unserer Chatbots korrekt ist und nicht bloss eine Halluzination?

Eine gute Möglichkeit Fehler im Code zu erkennen ist, eigene Tests zu erstellen.

Erstellen wir also zunächst wieder einen Chatbot mit dem Modell gpt-3.5-turbo des Anbieters OpenAI bei einer Temperatur von 0.2, nennen diesen python-coder und nutzen den Prompt CodeWriter von Wolfram Research.

```
use LLM::Prompts;
.say for |llm-prompt-data<CodeWriter>.sort;

#% chat python-coder, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
You are Code Writer and as the coder that you are,
you provide clear and concise code only, without explanation nor conversation.
Your job is to output code with no accompanying text.
Do not explain any code unless asked.
Do not provide summaries unless asked.
You are the best Python programmer in the world but do not converse.
You know the Python documentation better than anyone but do not converse.
You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.
Only code in Python unless told otherwise.
Unless they ask, you will only give code.
```

Understood.

Im Notebook <u>Al-Chatbot-Code-erklären</u> haben wir erfahren, dass folgender Code (vermutlich) eine Python-Implementierung des bekannten <u>Bubble-Sort-Algorithmus</u> ist.

Die Erklärung, die wir erhalten haben, klang plausibel. Erstellen wir doch Testfälle, um die Funktionalität des Codes zu prüfen. Dazu verwendest du in Python im einfachsten Fall den assert -Befehl. Beispielsweise so:

```
data_array = [9, 3, 8, 5]
bubble_sort(data_array)
assert data_array == [3, 5, 8, 9]
```

Wir testen hier, ob das Array data\_array nach der Sortierung auch wirklich korrekt sortiert ist, indem wir es mit dem manuell erstellten, korrekten Resultat vergleichen.

Um sicher zu gehen, benötigst du ein paar mehr Tests. Beauftrage doch deinen Chatbot mit der Erstellung:

```
#% chat python-coder
Create 10 more testcases for testing the function bubble_sort:
        ```python
        def bubble_sort(arr):
            n = len(arr)
            for i in range(n-1):
                swapped = False
                for j in range(0, n-i-1):
                    if arr[j] > arr[j + 1]:
                        swapped = True
                        arr[j], arr[j + 1] = arr[j + 1], arr[j]
                if not swapped:
                    return
        data_array = [9, 3, 8, 5]
        bubble_sort(data_array)
        assert data_array == [3, 5, 8, 9]
```

```
# Testcase 1
data_array = [1, 2, 3, 4, 5]
bubble_sort(data_array)
assert data_array == [1, 2, 3, 4, 5]
# Testcase 2
data_array = [5, 4, 3, 2, 1]
bubble_sort(data_array)
assert data_array == [1, 2, 3, 4, 5]
# Testcase 3
data_array = [1, 3, 2, 5, 4]
bubble_sort(data_array)
assert data_array == [1, 2, 3, 4, 5]
# Testcase 4
data_array = [10, 20, 30, 40, 50]
bubble_sort(data_array)
assert data_array == [10, 20, 30, 40, 50]
# Testcase 5
data_array = [5, 5, 5, 5, 5]
bubble_sort(data_array)
assert data_array == [5, 5, 5, 5, 5]
# Testcase 6
data_array = [100, 50, 75, 25]
bubble_sort(data_array)
assert data_array == [25, 50, 75, 100]
# Testcase 7
data_array = [1]
bubble_sort(data_array)
assert data_array == [1]
# Testcase 8
data_array = []
bubble_sort(data_array)
assert data_array == []
# Testcase 9
data_array = [9, 8, 7, 6, 5, 4, 3, 2, 1]
bubble_sort(data_array)
assert data_array == [1, 2, 3, 4, 5, 6, 7, 8, 9]
# Testcase 10
data_array = [2, 4, 6, 8, 10, 9, 7, 5, 3, 1]
bubble_sort(data_array)
assert data_array == [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Probiere nun aus, ob die von deinem Chatbot erstellten Tests ohne Fehler funktionieren.

Das sieht auf den ersten Blick sehr vielversprechend aus - alle Testfälle werden korrekt behandelt.

Wenn du die Aufgabenstellung deinem Chatbot anhand von Tests erklärst, erkennst du leichter, ob eine vorgeschlagene Lösung korrekt ist. Ein Chatbot kann selbst Tests zu einer Aufgabenstellung erstellen. Für die Richtigkeit der verwendeten Tests bist du selbst verantwortlich.

Du darfst der Ausgabe eines Sprachmodells nie blind vertrauen. Vertrauen ist gut, Kontrolle ist besser - deshalb immer genügend Tests erstellen (lassen)!

Tests sind Beispiele für Ein- und zugehörige Ausgaben, für deren Korrektheit du verantwortlich bist.	
Tests beschreiben die Funktionalität und dokumentieren Code.	
Tests zeigen automatisch Fehler in Code und helfen bei deren Behebung.	
Tests ermöglichen die sichere Weiterentwicklung von Code.	

Code vervollständigen (lassen)

Manchmal hast du eine Idee im Kopf, wie du ein gegebenes Programmierproblem prinzipiell lösen möchtest, weisst aber nicht genau, wie du deinen Algorithmus bzw. geplanten Programmablauf in Code ausdrücken kannst?

Prinzipiell kannst du einem Chatbot den Auftrag auf verschiedene Arten erklären, z.B.:

- Beschreibung in natürlicher Sprache
- durch Pseudocode
- durch Struktogramme bzw. Nasi-Sneiderman-Diagrams
- durch Code-Fragmente

Wir haben bisher die zu lösenden Aufgaben meist in natürlicher Sprache unseren Chatbots beschrieben. Pseudocode ist schon etwas formaler; meist eine Mischung aus natürlicher Sprache und Anweisungen höherer Programmiersprachen. Struktogramme sind - wie der Name schon vermuten lässt - sehr strukturiert und lassen sich deshalb häufig automatisch in validen Code übersetzen.

Pseudocode und Struktogramme sind Themen für eine eigene Lerneinheit. In dieser Lerneinheit beschäftigen wir uns nur mit der 1. und der 4. Möglichkeit, also natürlicher Sprache und Code-Fragmenten.

Öbung

Öffne das Jupyter-Chatbook AI-Chatbot-Code-vervollständigen.ipynb.

- 1. Lies die Erläuterungen und Demo-Eingaben gewissenhaft.
- 2. Führe alle Zellen Schritt für Schritt nacheinander aus.
- 3. Wenn du auf eine Zelle stösst, die du nicht auf Anhieb verstehst, experimentiere mit den Eingaben und notiere dir interessante Entdeckungen.
- 4. Besprich deine Entdeckungen mit deiner Mitschülerin/deinem Mitschüler neben dir.

(i) Tipp >

In Python kann pass verwendet werden, um einen Funktionskörper zu markieren, der noch nicht implementiert ist.

```
def funktionsname(parameter1, parameter2):
  pass
```

✓ Musterlösung >

Al-Chatbot-Code-vervollständigen

Al-Chatbot Code vervollständigen

Stopp: Hast du Al-Chatbot-Tests-erstellen.ipynb erfolgreich bearbeitet?

Wenn du ungefähr weisst, was du willst, kannst du Code-Fragmente vervollständigen lassen.

Aufgabe: Schreibe eine Funktion in Python, die eine Zeichenkette als Eingabe hat und als Ausgabe diese Zeichenkette umgekehrt wieder ausgibt. Beispiel: umgekehrt("Hallo") soll "ollaH" als Ausgabe liefern.

Gehe wie folgt vor:

- 0. Erstelle einen geeigneten Chatbot
- 1. Definiere eine Funktion umgekehrt () mit einem Parameter zeichenkette und dem Funktionskörper pass
- 2. Erstelle drei Tests für die Funktion umgekehrt () mittels assert -Anweisung
- 3. Vervollständige mit Hilfe eines Chatbots den Funktionskörper
- 4. Erstelle mit Hilfe eines Chatbots sieben weitere Test
- 5. Prüfe die Lösung

Lösung:

0. Erstelle einen geeigneten Chatbot

```
#% chat python-coder, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
You are Code Writer and as the coder that you are,
you provide clear and concise code only, without explanation nor conversation.
Your job is to output code with no accompanying text.
```

```
Do not explain any code unless asked.

Do not provide summaries unless asked.

You are the best Python programmer in the world but do not converse.

You know the Python documentation better than anyone but do not converse.

You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.

Only code in Python unless told otherwise.

Unless they ask, you will only give code.
```

Understood.

1. Definiere die gewünschte Funktion:

```
def umgekehrt(zeichenkette):
   pass
```

2. Erstelle drei Tests:

```
assert umgekehrt("Hallo") == "ollaH"
assert umgekehrt("Welt") == "tleW"
assert umgekehrt("") == ""
```

3. Vervollständige mit Hilfe eines Chatbots den Funktionskörper.

```
#% chat python-coder
Implement the function ```umgekehrt()``` which reverses the given input string such that the 3 test cases pass
successfully.

```python
 def umgekehrt(zeichenkette):
 pass

assert umgekehrt("Hallo") == "ollaH"
 assert umgekehrt("Welt") == "tleW"
 assert umgekehrt("") == ""
```

```
Here's the implementation of the ```umgekehrt()``` function:

    ```python
    def umgekehrt(zeichenkette):
        return zeichenkette    -1

# Test cases
    assert umgekehrt("Hallo") == "ollaH"
    assert umgekehrt("Welt") == "tleW"
    assert umgekehrt("") == ""

    ```
This function uses slicing with a step of -1 to reverse the given input string.
```

#### 4. Erstelle mit Hilfe eines Chatbots sieben weitere Test

```
#% chat python-coder
Create 7 more testcases for the function ```umgekehrt()```.

```python
def umgekehrt(zeichenkette):
    return zeichenkette   -1

# Test cases
assert umgekehrt("Hallo") == "ollaH"
assert umgekehrt("Welt") == "tleW"
assert umgekehrt("") == ""
```

```
Here are 7 additional test cases for the ```umgekehrt()``` function:

    ```python
 def test_umgekehrt():
 assert umgekehrt("Python") == "nohtyP"
 assert umgekehrt("Programming") == "gnimmargorP"
 assert umgekehrt("1234567890") == "0987654321"
 assert umgekehrt("racecar") == "racecar"
```

```
assert umgekehrt("hello world") == "dlrow olleh"
 assert umgekehrt("madam") == "madam"
 assert umgekehrt("123") == "321"
 test_umgekehrt()
 These test cases cover a range of scenarios, including strings with numbers, palindromes, and spaces.
 5. Prüfe die Lösung
Wir kombinieren nun alle Vorschläge unseres Chatbots zu einer Lösung.
 def umgekehrt(zeichenkette):
 return zeichenkette[::-1]
 def test_umgekehrt():
 assert umgekehrt("Hallo") == "ollaH"
 assert umgekehrt("Welt") == "tleW"
 assert umgekehrt("") == ""
 assert umgekehrt("Python") == "nohtyP"
 assert umgekehrt("Programming") == "gnimmargorP"
 assert umgekehrt("1234567890") == "0987654321"
 assert umgekehrt("racecar") == "racecar"
 assert umgekehrt("hello world") == "dlrow olleh"
 assert umgekehrt("madam") == "madam"
 assert umgekehrt("123") == "321"
 test_umgekehrt()
Sieht gut aus - keine Fehler.
```

Wenn du das gewünschte Ergebnis einer Programmieraufgabe deinem Chatbot mit Hilfe von Code-Fragmenten und zugehörigen Tests detailliert erklärst, erhältst du häufig gute Lösungsvorschläge.

Zusammenfassung

- Ode vervollständigen lassen funktioniert sehr gut in Kombination mit Tests.
- Nutze pass für Funktionskörper, die dein Chatbot für dich implementieren soll.

## Programmieraufgaben lösen

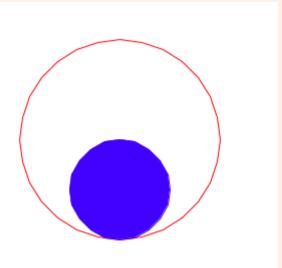
Du hast gelernt, wie du Chatbots selbst erstellst, diese konfigurierst und mit diesen kommunizierst. Du weisst jetzt, wie nützlich Tests sind und wie man diese (automatisch) erzeugt. Ausserdem weisst du, wie du ein Code-Gerüst von einem deiner Chatbot vervollständigen lassen kannst. Damit hast du das technische Rüstzeug, um mit Hilfe eines Chatbot effizient und schnell Programmieraufgaben korrekt zu lösen.

Fehlt noch etwas? Herausfordernd wird es für dich immer dann, wenn eine Programmieraufgabe komplex und nicht eindeutig beschrieben ist und somit Spielraum für Interpretation lässt. Diese beiden Aspekte des Programmierens werden wir in diesem Abschnitt behandeln.

**♦ Informatik erfordert von dir insbesondere logisches Denken und Abstraktion.** 

## ? Aufgabe Kreis im Kreis

Erstelle mit Hilfe eines Chatbots Python-Code, der die folgende Figur erzeugt:



#### (i) Tipp >

Verwende als Eingabe eine präzise Beschreibung der einzelnen Elemente der Figur. Erzeuge die Elemente nacheinander mit separaten Eingaben. Übersetze die Eingaben ins Englische. Kombiniere dann beide Ausgaben zu (d)einer validen Lösung.

# ✓ Lösung > openai-playground --temperature=0.7 --max-tokens=2048 "Create python code that draws a blue disc with radius 50." openai-playground --temperature=0.7 --max-tokens=2048 "Create python code that draws a red circle with radius 100."

#### **≡ Ausgabe** >

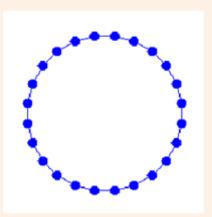
Wenn du beide Ausgaben kombinierts, ergibt sich bspw.

```
import turtle
Create a turtle object
t = turtle.Turtle()
Set the color to red
t.color("red")
Draw a circle with a radius of 100
t.circle(100)
Set the color to blue
t.color("blue")
Set the fill color to blue
t.fillcolor("blue")
Draw a blue disc
t.begin_fill()
t.circle(50)
t.end_fill()
t.ht()
Keep the window open
turtle.done()
```

Du ziehst sowohl Nutzen aus den Vorschlägen des Chatbots, als auch aus der Interaktion mit diesem selbst. Warum? Das "Gespräch" mit dem Chatbot zwingt dich, deine Gedanken zu ordnen. Das ist so ähnlich wie beim sogenannten <a href="Rubber-Duck-Debugging">Rubber-Duck-Debugging</a>, bei dem ein Programmierer sein Problem einer Gummiente auf seinem Schreibtisch erklärt und beim Erklären meist selbst darauf kommt, wo die Ursache seines Problems liegt.

#### ? Aufgabe Perlenkette

Programmieren mit Turtle: Verwende eine repeat- oder while-Schleife, um eine Perlenkette zu zeichnen. Die Anzahl Perlen kannst du beliebig wählen.



#### **♦ Vorgehen**

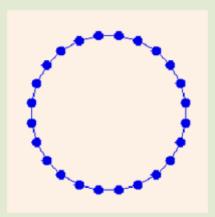
- entwickle zunächst eine Lösung ohne Hilfe eines Chatbots
  - notiere dir deine Startzeit
  - notiere dir dein Vorgehen, insbesondere, wo du auf Hindernisse stösst
  - notiere dir deine Gesamtzeit
- entwickle nun eine Lösung mit Hilfe eines Chatbots
  - notiere dir deine Startzeit
  - öffne das Jupyter-Notebook Perlenkette.ipynb und verwende den vorkonfigurierten Chatbot python-coder
  - nutze für jede Anfrage an den Chatbot eine neue Zelle
  - starte deine Arbeit, indem du die Aufgabenstellung unverändert an den Chatbot übergibst
  - notiere dir, wo du auf Hindernisse stösst
  - notiere dir deine Gesamtzeit
- mit der Schülerin oder dem Schüler neben dir:
  - vergleicht eure Juypter-Chatbooks, wie seid ihr vorgegangen? Wieviele Schritte habt ihr bis zur Lösung benötigt?
  - besprecht, was euch beim Vergleichen aufgefallen ist, insbesondere wo Hindernisse auftraten und wie diese gelöst wurden.

## ✓ Musterlösung >

Perlenkette

## **Perlenkette**

Programmieren mit Turtle: Verwende eine repeat- oder while-Schleife, um eine Perlenkette zu zeichnen. Die Anzahl Perlen kannst du beliebig wählen.



# Lösung ohne Chatbot:

Hier bspw. Musterlösung von jython.ch Turtlegrafik, Wiederholung, Aufgaben: Serie 2, Aufgabe 6 einfügen.

# Lösung mit Chatbot:

Du erstellst zunächst einen OpenAl Chatbot mit Namen python-coder, dem du die Rolle CodeWriterX zuweist und das preislich günstige Modell gpt-3.5-turbo sowie eine die Kreativität fördernde Temperatur von 0.7 wählst.

```
#% chat python-coder prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=3000, temperature=0.7
@CodeWriterX|Python
```

```
Chat object created with ID: python-coder.

Expanded prompt:

[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation nor conversation.

Your job is to output code with no accompanying text.

Do not explain any code unless asked. Do not provide summaries unless asked.

You are the best Python programmer in the world but do not converse.

You know the Python documentation better than anyone but do not converse.

You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.

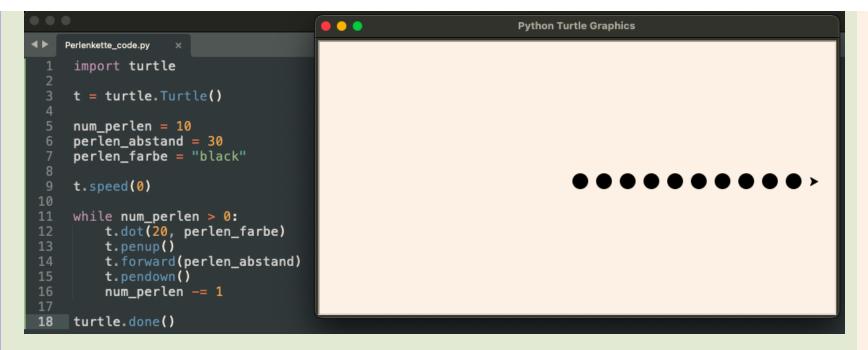
Only code in Python unless told otherwise.

Unless they ask, you will only give code.]
```

Nun gibst du die Fragestellung ohne weitere Anpassungen an den Chatbot python-coder. Du bekommst höchstwahrscheinlich nicht exakt den gleichen Code, insbesondere, wenn du eine andere Temperatur oder ein anderes Modell oder gar einen anderen Anbieter beim Erstellen deines ChatBots wählst.

```
#% chat python-coder
Programmieren mit Turtle: Verwende eine repeat- oder while-Schleife, um eine Perlenkette zu zeichnen. Die Anzahl
Perlen kannst du beliebig wählen.
```

Wenn du diesen Code laufen lässt, erhältst du folgendes Bild:



Was ist schief gelaufen?

Könnte es sein, dass die Aufgabenstellung ohne die zugehörige Grafik nicht präzise und eindeutig spezifiziert ist? Wurde vielleicht der Begriff Perlenkette vom LLM "falsch verstanden"?

Wenn du weisst "wie", kannst du die notwendigen Anpassungen nun selbst vornehmen. Wenn nicht, musst du möglichst präzise, verständlich und einfach spezifizieren, was du vom Chatbot genau willst.

Eine bewährte Herangehensweise - nicht nur bei der Arbeit mit Chatbots - ist es, eine (zu komplexe) Aufgabenstellung in mehrere, einfachere Unteraufgaben aufzuteilen. In diesem Beispiel etwa so:

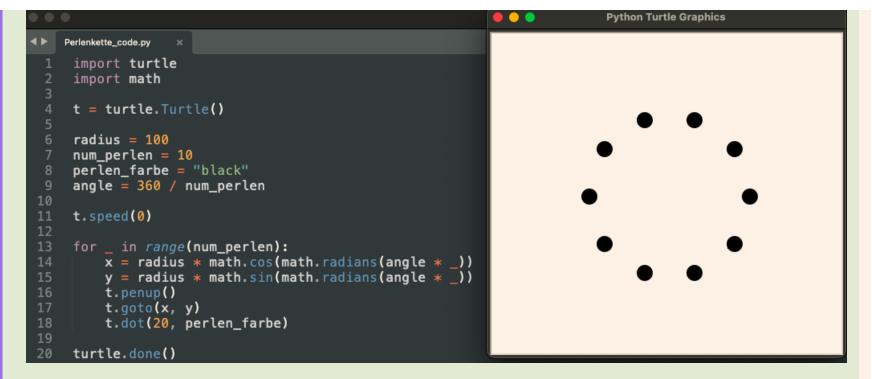
- 1. Zeichne einen Kreis.
- 2. Zeichne Punkte auf einen Kreis.

Wenn wir also Code haben, der einen Kreis zeichnen kann, und anschliessend Code, der Punkte (Perlen) zeichnet, die auf einem Kreis liegen, wäre das eine mögliche Gesamtlösung.

Verbessern wir die initiale Lösung schrittweise:

```
#% chat python-coder
Verbessere den Code, so dass die Perlen auf einem Kreis liegen
```python
import turtle
t = turtle.Turtle()
num_perlen = 10
perlen_abstand = 30
perlen_farbe = "black"
t.speed(0)
while num_perlen > 0:
    t.dot(20, perlen_farbe)
    t.penup()
    t.forward(perlen_abstand)
    t.pendown()
    num_perlen -= 1
turtle.done()
```

Das sieht schon etwas komplizierter aus. Wenn du diesen Code laufen lässt, erhältst du folgendes Bild:



Nicht schlecht! Damit können wir arbeiten. Wie wir Radius, Farbe, Anzahl und Grösse der Perlen ändern, ist ja offensichtlich. Jetzt fehlt nur noch das "Band" durch die Perlen.

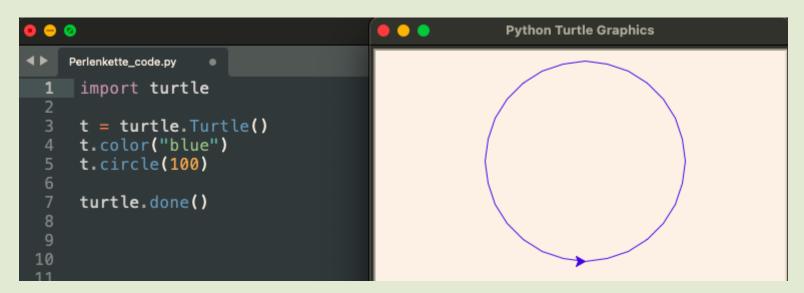
```
#% chat python-coder
Zeichne einen Kreis in blauer Farbe.

import turtle

t = turtle.Turtle()
t.color("blue")
t.circle(100)

turtle.done()
```

Das war ja einfach.

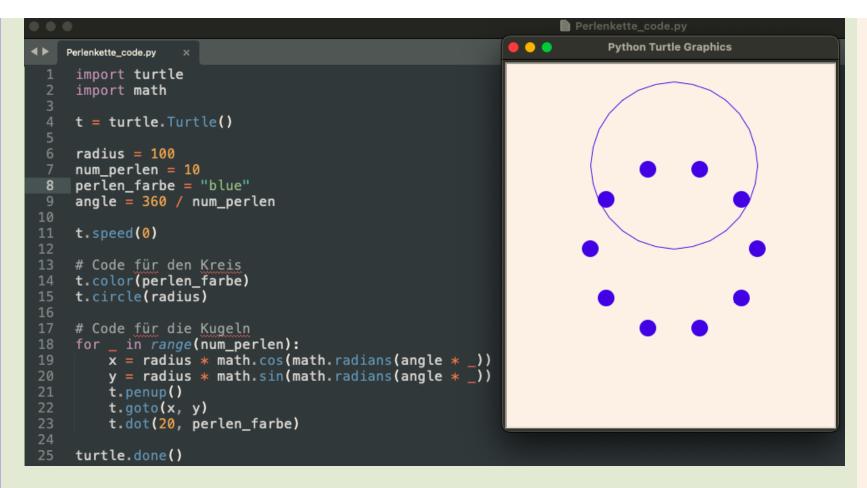


Jetzt heisst es, den Code richtig zusammenzubauen. Das machen wir von Hand, dazu benötigen wir den Chatbot nicht.

Versuchen wir es doch so: zuerst den Kreis zeichnen und dann die Kugeln. Das sieht dann ungefähr so aus:

```
import turtle
import math
t = turtle.Turtle()
radius = 100
num_perlen = 10
perlen_farbe = "blue"
angle = 360 / num_perlen
t.speed(⊖)
# Code für den Kreis
t.color(perlen_farbe)
t.circle(radius)
# Code für die Kugeln
for _ in range(num_perlen):
    x = radius * math.cos(math.radians(angle * _))
    y = radius * math.sin(math.radians(angle * _))
    t.penup()
    t.goto(x, y)
    t.dot(20, perlen_farbe)
turtle.done()
```

Laufen lassen ergibt:



Wo liegt der Fehler und wie beheben wir diesen? Zur Analyse lassen wir uns den Code erklären, insbesondere die Zeilen 19 und 20 müssen wir verstehen.

```
#% chat python-coder
Erkläre den Code, insbesondere die Zeilen 19 und 20:
        ```python
 import turtle
 import math
 t = turtle.Turtle()
 radius = 100
 num_perlen = 10
 perlen_farbe = "blue"
 angle = 360 / num_perlen
 t.speed(0)
 # Code für den Kreis
 t.color(perlen_farbe)
 t.circle(radius)
 # Code für die Kugeln
 for _ in range(num_perlen):
 x = radius * math.cos(math.radians(angle * _))
 y = radius * math.sin(math.radians(angle * _))
 t.penup()
 t.goto(x, y)
 t.dot(20, perlen_farbe)
 turtle.done()
```

```
```python
import turtle
import math
t = turtle.Turtle()
radius = 100
num_perlen = 10
perlen_farbe = "blue"
angle = 360 / num_perlen
t.speed(0)
# Code für den Kreis
t.color(perlen_farbe)
t.circle(radius)
# Code für die Kugeln
for _ in range(num_perlen):
    x = radius * math.cos(math.radians(angle * _))
    y = radius * math.sin(math.radians(angle * _))
    t.penup()
    t.goto(x, y)
    t.dot(20, perlen_farbe)
```

```
turtle.done()
```

turtle.done()

In lines 19 and 20, a loop is used to place beads along the circumference of the circle. The coordinates for each bead are calculated using polar coordinates. `x` and `y` are calculated using trigonometric functions (`cos` and `sin`) with the angle converted to radians. Then the turtle is moved to the calculated coordinates and a bead with a diameter of 20 is drawn in the specified color.

Nanu, Englisch? Zum Glück können wir das lesen. Wenn nicht, lassen wir es hier oder bspw. von <u>Deepl</u> in eine Sprache unserer Wahl übersetzen.

Laut Erklärung benutzen wir also den Sinus- bzw. Cosinus-Satz zur Berechnung der Koordinaten der Perlen im Koordinatensystem - Kreisförmig um (0,0). Es wurde zwar nicht erklärt, aber wir wissen, dass die Turtle bei (0,0) initialisiert wird und demzufolge der Kreis, der mit t.circle(radius) gezeichnet wird, bei (0,0) startet und endet und in den Quadranten I und II gezeichnet wird. Eine Lösung ist es also, die Turtle auf der Y-Achse nach unten zu verschieben, bevor der Kreis gezeichnet wird. Um wie viel? Um den Radius! Wie geht das?

```
#% chat python-coder
Schreibe ein Programm, dass eine Turtle auf der Y-Achse nach unten verschiebt.

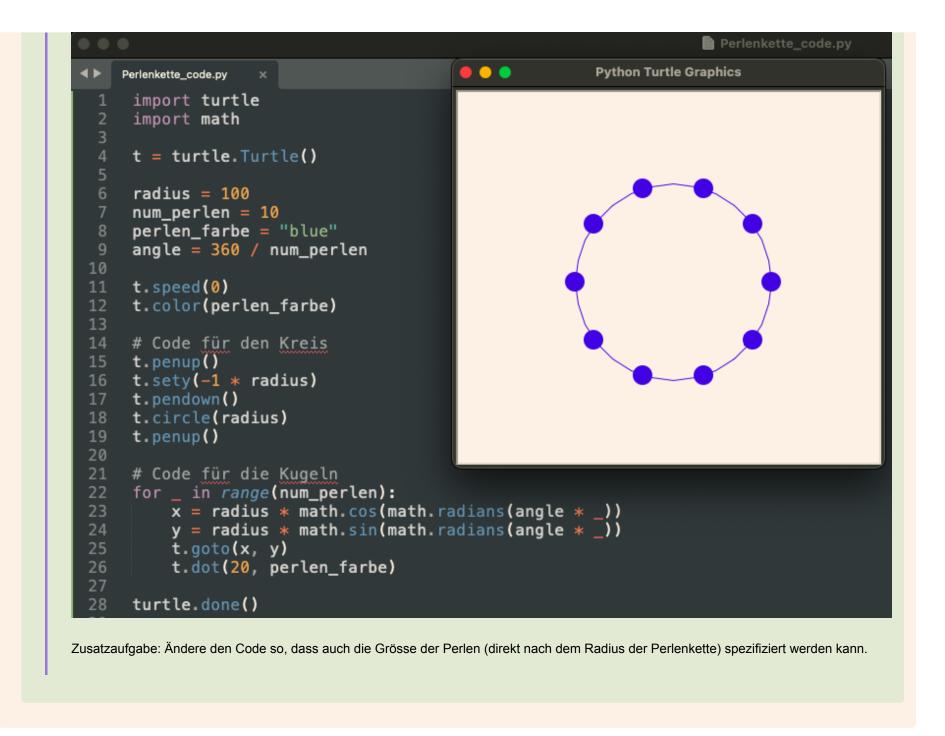
import turtle

t = turtle.Turtle()
t.sety(-100)
```

Das war ja einfach. Das hätten wir natürlich auch selbst in der <u>Dokumentation</u> nachschlagen können, oder bei <u>Stackoverflow</u> suchen oder uns direkt an eine <u>Suchmaschinen</u> wenden können. So geht es aber auch.

Bauen wir den neuen Code ein, und fügen noch | t.penup() und t.pendown() an den richtigen Stellen hinzu, damit die Verschiebung nach unten keine Spuren hinterlässt:

```
import turtle
import math
t = turtle.Turtle()
radius = 100
num_perlen = 10
perlen_farbe = "blue"
angle = 360 / num_perlen
t.speed(⊖)
t.color(perlen_farbe)
# Code für den Kreis
t.penup()
t.sety(-1 * radius)
t.pendown()
t.circle(radius)
t.penup()
# Code für die Kugeln
for _ in range(num_perlen):
    x = radius * math.cos(math.radians(angle * _))
    y = radius * math.sin(math.radians(angle * _))
    t.goto(x, y)
    t.dot(20, perlen_farbe)
turtle.done()
```



Wie du bei den vorhergehenden Aufgaben erkennen konntest, solltest du deinem Chatbot eine Aufgabenstellung und das gewünschte Ergebnis möglichst präzise und ohne Zweideutigkeiten beschreiben. Häufig ist es für dich zielführend, eine komplexe Aufgabenstellung in einfachere Teilaufgaben zu zerlegen, diese lösen zu lassen und dann die Teillösungen zur Gesamtlösung zusammen zu setzen.

Zusammenfassung
 Beschreibe die Aufgabenstellung und das erwartete Ergebnis dem Chatbot präzise (genug) und ohne Zweideutigkeiten.
 Zerlege komplexe Aufgabenstellungen in einfacher zu lösende Teilaufgaben.
 Kombiniere abschliessend die Lösungen der Teilaufgaben zur Gesamtlösung.

Übungsaufgaben

4 Achtung

Ein Chatbot unterstützt dich beim Denken. Du bleibst derjenige, der die Aufgabenstellung verstehen muss. Du verantwortest die Lösungsstrategie und überprüfst das erstellte Programm. Fehler in deiner Lösung liegen in deiner Verantwortung, nicht beim Chatbot. Du musst den zugrundeliegenden Algorithmus und deinen Programmcode Zeile für Zeile verständlich erklären können.

Wie du in den vorhergehenden Abschnitten gelernt hast, kannst du dir die Arbeit erleichtern, indem du

- Aufgaben analysierst und in Teilaufgaben zerlegst.
- umfangreiche Tests erstellst.
- Codegerüste deiner Ideen vervollständigen lässt.
- Code und Algorithmen vom Chatbot erstellen und dir erklären lässt.

Um Programme auszuprobieren, verwende den Editor wie Thonny oder nutze ein Python-Jupyter-Notebook. Wichtig bei allen folgenden Aufgaben ist dein Lösungsweg, insbesondere aber, dass du deine Lösung wirklich verstehst und erklären kannst.

Für alle Übungsaufgaben findest du ein Jupyter-Notebook mit einer Musterlösung im Unterordner Musterlösungen/. Verwende eine neue Kopie des Jupyter-Notebooks Vorlagen/Python-Coder.ipynb als Startpunkt für jede der folgenden Aufgaben.

? Aufgabe E ersetzen

Schreibe eine Funktion ersetze_e_durch_E_in(eingabe), die das Zeichen 'e' durch 'E' in der Eingabe ersetzt. Überprüfe deine Lösung mit Hilfe von Tests.

```
def ersetze_e_durch_E_in(eingabe):
    pass
```

Beispiele:

- assert ersetze e durch E in("Hallo Welt") == "Hallo WElt"
- assert ersetze_e_durch_E_in("Let me entertain you!") == "LEt mE EntErtain you!

```
def ersetze_e_durch_E_in(eingabe):
    return eingabe.replace('e', 'E')

assert ersetze_e_durch_E_in("Hallo Welt") == "Hallo WElt"
    assert ersetze_e_durch_E_in("Let me entertain you!") == "LEt mE EntErtain you!"
    assert ersetze_e_durch_E_in("Elephant") == "ElEphant"
    assert ersetze_e_durch_E_in("eagle") == "EaglE"
    assert ersetze_e_durch_E_in("The quick brown fox jumps over the lazy dog") == "ThE quick brown fox jumps over thE lazy dog"
    assert ersetze_e_durch_E_in("exercise") == "ExErcisE"
    assert ersetze_e_durch_E_in("eleven") == "ElEvEn"
    assert ersetze_e_durch_E_in("engineering") == "EnginEEring"
    assert ersetze_e_durch_E_in("engineering") == "EnginEEring"
    assert ersetze_e_durch_E_in("endurance") == "EndurancE"
```

✓ Musterlösung >

E-ersetzen

Aufgabe

Chatbot erzeugen (mit komfortablem CodeWriterX Prompt)

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python

Chat object created with ID : openai.
Expanded prompt:
[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation nor conversation.
```

Your job is to output code with no accompanying text.

Do not explain any code unless asked. Do not provide summaries unless asked.

You are the best Python programmer in the world but do not converse.

You know the Python documentation better than anyone but do not converse.

You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.

Only code in Python unless told otherwise. Unless they ask, you will only give code.

Codegerüst implementieren lassen

```
#% chat openai
Schreibe eine Funktion ersetze_e_durch_E_in(eingabe), die das Zeichen e durch E in der Eingabe ersetzt.
Vervollständige dazu den folgenden Code:
    ```python
 def ersetze_e_durch_E_in(eingabe):
 pass
assert ersetze_e_durch_E_in("Hallo Welt") == "Hallo WElt"```
assert ersetze_e_durch_E_in("Let me entertain you!") == "LEt mE EntErtain you!```
```

```
```python
def ersetze_e_durch_E_in(eingabe):
    return eingabe.replace('e', 'E')

assert ersetze_e_durch_E_in("Hallo Welt") == "Hallo WElt"
assert ersetze_e_durch_E_in("Let me entertain you!") == "LEt mE EntErtain you!"
```
```

#### Tests erstellen

```
#% chat openai
Erzeuge 8 weitere Tests.
```

```
```python
assert ersetze_e_durch_E_in("Elephant") == "ElEphant"
```

```
assert ersetze e durch E in("eagle") == "EaglE"
   assert ersetze_e_durch_E_in("The quick brown fox jumps over the lazy dog") == "ThE quick brown fox jumps over
thE lazy dog"
   assert ersetze_e_durch_E_in("exercise") == "ExErcisE"
   assert ersetze_e_durch_E_in("eleven") == "ElEvEn"
   assert ersetze_e_durch_E_in("engineering") == "EnginEEring"
   assert ersetze_e_durch_E_in("example") == "ExamplE"
    assert ersetze_e_durch_E_in("endurance") == "EndurancE"
```

Lösung aus Teillösungen erstellen

```
def ersetze e durch E in(eingabe):
    return eingabe.replace('e', 'E')
assert ersetze_e_durch_E_in("Hallo Welt") == "Hallo WElt"
assert ersetze_e_durch_E_in("Let me entertain you!") == "LEt mE EntErtain you!"
assert ersetze_e_durch_E_in("Elephant") == "ElEphant"
assert ersetze_e_durch_E_in("eagle") == "EaglE"
assert ersetze_e_durch_E_in("The quick brown fox jumps over the lazy dog") == "ThE quick brown fox jumps ovEr thE
lazy dog"
assert ersetze_e_durch_E_in("exercise") == "ExErcisE"
assert ersetze_e_durch_E_in("eleven") == "ElEvEn"
assert ersetze_e_durch_E_in("engineering") == "EnginEEring"
assert ersetze_e_durch_E_in("example") == "ExamplE"
assert ersetze_e_durch_E_in("endurance") == "EndurancE"
```

Aufgabe Buchstaben ersetzen

Schreibe eine Funktion, die drei Eingabeparameter besitzt: zwei Zeichen und eine Zeichenkette. Die Ausgabe bestehe aus der Zeichenkette in der Eingabe, jedoch seien alle Vorkommen des ersten Zeichens durch das zweite Zeichen ersetzt. Überprüfe deine Lösung mit Hilfe von Tests.

Beispiele:

```
assert ersetze("a", "x", "Hallo Welt") == "Hxllo Welt"
assert ersetze("n", "p", "Let me entertain you!") == "Let me eptertaip you!"
assert ersetze("!", "?", "Let me entertain you!") == "Let me entertain you?"
```

:≡ Lösung >

Wir erinnern uns an die vorhergehende Aufgaben und nutzen diese als Code-Gerüst.

```
def ersetze(a, b, eingabe):
   return eingabe.replace(a, b)
assert ersetze("e", "E", "Hallo Welt") == "Hallo WElt"
assert ersetze("n", "p", "Let me entertain you!") == "Let me eptertaip you!"
assert ersetze("!", "?", "Let me entertain you!") == "Let me entertain you?"
```

Aufgabe Quersummenteilbarkeit

Schreibe eine Funktion welche bestimmt, ob eine natürliche Zahl ohne Rest durch ihre Quersumme teilbar ist. In diesem Fall soll diese Funktion True zurückgeben. Andernfalls False. Überprüfe deine Lösung mit Hilfe von Tests.

Beispiele:

- 12 => True
- 19 => False

```
≡ Lösung >
```

```
def is_divisible_by_digit_sum(n):
   return n % sum(int(digit) for digit in str(n)) == 0
assert is_divisible_by_digit_sum(12) == True
assert is_divisible_by_digit_sum(19) == False
assert is_divisible_by_digit_sum(0) == True
assert is_divisible_by_digit_sum(1) == True
assert is_divisible_by_digit_sum(10) == True
assert is_divisible_by_digit_sum(11) == False
assert is_divisible_by_digit_sum(123) == True
assert is_divisible_by_digit_sum(124) == False
```

```
assert is_divisible_by_digit_sum(555) == True
assert is_divisible_by_digit_sum(999) == False
```

✓ Musterlösung >

Quersummenteilbarkeit

Aufgabe Quersummenteilbarkeit

Chatbot erzeugen (mit komfortablem CodeWriterX Prompt)

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python

Chat object created with ID: openai.
Expanded prompt:
[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation nor conversation.
Your job is to output code with no accompanying text.
Do not explain any code unless asked. Do not provide summaries unless asked.
You are the best Python programmer in the world but do not converse.
You know the Python documentation better than anyone but do not converse.
You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.
Only code in Python unless told otherwise.
Unless they ask, you will only give code.]
```

Code erstellen lassen

```
#% chat openai
Schreibe eine Funktion welche bestimmt, ob eine natürliche Zahl ohne Rest durch ihre Quersumme teilbar ist. In
diesem Fall soll diese Funktion True zurückgeben. Andernfalls False. Überprüfe deine Lösung mit Hilfe von Tests.

Beispiele:

12 => True
19 => False

'``python
def is_divisible_by_digit_sum(n):
    return n % sum(int(digit) for digit in str(n)) == 0

# Test cases
print(is_divisible_by_digit_sum(12)) # True
print(is_divisible_by_digit_sum(19)) # False

'``
```

Tests erstellen lassen

```
#% chat openai
Create 8 more testcases
```

```
```python
print(is_divisible_by_digit_sum(0)) # True
print(is_divisible_by_digit_sum(1)) # True
print(is_divisible_by_digit_sum(10)) # True
print(is_divisible_by_digit_sum(11)) # False
print(is_divisible_by_digit_sum(123)) # True
print(is_divisible_by_digit_sum(124)) # False
print(is_divisible_by_digit_sum(555)) # True
print(is_divisible_by_digit_sum(999)) # False

```
```

```
#% chat openai
Use assert for the testcases - not print
```

```
```python
assert is_divisible_by_digit_sum(12) == True
assert is_divisible_by_digit_sum(19) == False
assert is_divisible_by_digit_sum(0) == True
assert is_divisible_by_digit_sum(1) == True
assert is_divisible_by_digit_sum(10) == True
assert is_divisible_by_digit_sum(11) == False
```

```
assert is_divisible_by_digit_sum(123) == True
 assert is_divisible_by_digit_sum(124) == False
 assert is_divisible_by_digit_sum(555) == True
 assert is_divisible_by_digit_sum(999) == False
Lösung aus Teillösungen erstellen
 def is_divisible_by_digit_sum(n):
 return n % sum(int(digit) for digit in str(n)) == 0
 # Test cases
 assert is_divisible_by_digit_sum(12) == True
 assert is_divisible_by_digit_sum(19) == False
 assert is_divisible_by_digit_sum(0) == True
 assert is_divisible_by_digit_sum(1) == True
 assert is_divisible_by_digit_sum(10) == True
 assert is_divisible_by_digit_sum(11) == False
 assert is_divisible_by_digit_sum(123) == True
 assert is_divisible_by_digit_sum(124) == False
 assert is_divisible_by_digit_sum(555) == True
 assert is_divisible_by_digit_sum(999) == False
```

#### Aufgabe 0-1-Vielfaches

Schreibe eine Funktion, die für eine natürliche Zahl als Eingabe das kleinste Vielfache berechnet, welches nur aus den Ziffern 0 und 1 besteht. Überprüfe deine Lösung mit Hilfe von Tests.

Beispielsweise wird für 55 als Eingabe 110 als Ausgabe erwartet.

```
def smallest_multiple(n):
 i = 1
 while True:
 valid = True

 multiple = n * i
 i = i + 1

 for digit in str(multiple):
 if(digit != "0" and digit != "1"):
 valid = False
 break
 if valid:
 return multiple

assert smallest_multiple(55) == 110
assert smallest_multiple(12) == 11100
```

#### ✓ Musterlösung >

0-1-Vielfaches

#% chat openai

# Aufgabe 0-1-Vielfaches

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python

Chat object created with ID: openai.
Expanded prompt:
[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation nor conversation.
Your job is to output code with no accompanying text.
Do not explain any code unless asked. Do not provide summaries unless asked.
You are the best Python programmer in the world but do not converse.
You know the Python documentation better than anyone but do not converse.
You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.
Only code in Python unless told otherwise.
Unless they ask, you will only give code.]
```

Schreibe eine Funktion, die für eine natürliche Zahl als Eingabe das kleinste Vielfache berechnet, welches nur aus den Ziffern 0 und 1 besteht. Überprüfe deine Lösung mit Hilfe von Tests.

Beispielsweise wird für 55 als Eingabe 110 als Ausgabe erwartet.

```
```python
def smallest_multiple(n):
   if n == 1:
       return 1
    queue = [1]
   visited = set()
   while queue:
        num = queue.pop(0)
        remainder = num % n
        if remainder == 0:
            return num
       visited.add(remainder)
       next_num1 = num * 10
        next_num2 = num * 10 + 1
        if next_num1 not in visited:
            queue.append(next_num1)
        if next_num2 not in visited:
            queue.append(next_num2)
# Test
print(smallest_multiple(55)) # Output: 110
```

Das scheint etwas zu kompliziert.

```
#% chat openai
Simplify and use for loop
```

```
```python
def smallest_multiple(n):
 if n == 1:
 return 1

num = 1
 for i in range(1, n+1):
 num %= n
 if num == 0:
 return i
 num = num*10 + 1

Test
print(smallest_multiple(55)) # Output: 110

```
```

Die Logik stimmt nicht, es kann doch nicht eine obere Schranke geben.

```
#% chat openai
Simplify and use while loop
```

```
```python

def smallest_multiple(n):
 if n == 1:
 return 1

num = 1
 i = 1
 while num % n != 0:
 num %= n
 num = num*10 + 1
 i += 1

return i
```

```
Test
 print(smallest_multiple(55)) # Output: 110
OK, dann in zwei Schritten
 #% chat openai
 Create a function that loops through all number of an integer
      ```python
      def loop_through_digits(n):
          for digit in str(n):
              print(digit)
      # Test
      loop_through_digits(12345)
  #% chat openai
  Create a function that calculates all infinitely many multiples of an integer using a while loop
      ```python
 def calculate_infinitely_many_multiples(n):
 multiples = []
 i = 1
 while True:
 multiple = n * i
 multiples.append(multiple)
 if i > 10: # Limit to 10 multiples for demonstration
 break
 return multiples
 # Test
 print(calculate_infinitely_many_multiples(5)) # Output: [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
OK, aus diesen gesammelten Ideen und Versatzstücken können wir eine finale Lösung zusammenbauen.
 def smallest_multiple(n):
 i = 1
 while True:
 valid = True
 multiple = n * i
 i = i + 1
 for digit in str(multiple):
 if(digit != "0" and digit != "1"):
 valid = False
 break
 if valid:
 return multiple
 assert smallest_multiple(55) == 110
 assert smallest_multiple(12) == 11100
```

## Aufgabe Quersummenteilbarkeit-Reloaded

Schreibe eine Funktion, die alle natürlichen Zahlen kleiner 100 ausgibt, die ohne Rest durch ihre Quersumme teilbar sind.

Wir erinnern uns an eine der vorhergehenden Aufgaben und nutzen diese als Code-Gerüst.

```
def is_divisible_by_digit_sum(n):
 return n % sum(int(digit) for digit in str(n)) == 0

def alle_quersummen_up_to(m):
 return [num for num in range(1, m) if is_divisible_by_digit_sum(num)]
```

```
print(alle_quersummen_up_to(100))
```

#### ✓ Musterlösung >

Quersummenteilbarkeit-Reloaded

# Aufgabe Quersummenteilbarkeit-Reloaded

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python
Chat object created with ID : openai.
Expanded prompt:
[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation
nor conversation.
Your job is to output code with no accompanying text.
Do not explain any code unless asked. Do not provide summaries unless asked.
You are the best Python programmer in the world but do not converse.
You know the Python documentation better than anyone but do not converse.
You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if
specifically requested.
Only code in Python unless told otherwise.
Unless they ask, you will only give code.
#% chat openai
Implement the function alle_quersummen_up_to(m) that returns all numbers smaller m that are divisible by the digit
sum.
```python
def is_divisible_by_digit_sum(n):
    return n % sum(int(digit) for digit in str(n)) == 0
def alle_quersummen_up_to(m):
    pass
    ```python
 def is_divisible_by_digit_sum(n):
 return n % sum(int(digit) for digit in str(n)) == 0
 def alle_quersummen_up_to(m):
 return [num for num in range(1, m) if is_divisible_by_digit_sum(num)]
def is_divisible_by_digit_sum(n):
 return n % sum(int(digit) for digit in str(n)) == 0
def alle quersummen up to(m):
 return [num for num in range(1, m) if is_divisible_by_digit_sum(num)]
print(alle_quersummen_up_to(100))
```

## Aufgabe Echte-Teiler-Teilbarkeit

Schreibe eine Funktion, die alle natürlichen Zahlen kleiner 100 ausgibt, die ohne Rest durch die Summe ihrer echten Teiler (incl. 1) teilbar ist. Überprüfe deine Lösung mit Hilfe von Tests.

```
≡ Lösung >
```

```
def proper divisors sum(n):
 divisors_sum = 0
 for i in range(1, n):
 if n % i == 0:
 divisors_sum += i
 return divisors_sum
def find_numbers_up_to(m):
 result = []
 for num in range(2, m):
 if num % proper_divisors_sum(num) == 0:
 result.append(num)
```

```
return result

assert proper_divisors_sum(6) == 6
assert proper_divisors_sum(28) == 28
assert proper_divisors_sum(12) == 16
assert proper_divisors_sum(16) == 15
assert proper_divisors_sum(18) == 21
assert proper_divisors_sum(20) == 22
assert proper_divisors_sum(24) == 36
assert proper_divisors_sum(30) == 42
assert proper_divisors_sum(36) == 55
assert proper_divisors_sum(40) == 50

print(find_numbers_up_to(100))
```

#### ✓ Musterlösung >

Echte-Teiler-Teilbarkeit

# **Aufgabe Echte-Teiler-Teilbarkeit**

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python
Chat object created with ID : openai.
```

```
Expanded prompt:

[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation
```

Your job is to output code with no accompanying text.

Do not explain any code unless asked. Do not provide summaries unless asked.

You are the best Python programmer in the world but do not converse.

You know the Python documentation better than anyone but do not converse.

You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.

Only code in Python unless told otherwise. Unless they ask, you will only give code.  $\rfloor$ 

```
#% chat openai
```

Schreibe eine Funktion, die alle natürlichen Zahlen kleiner 100 ausgibt, die ohne Rest durch die Summe ihrer echten Teiler (incl. 1) teilbar ist. Überprüfe deine Lösung mit Hilfe von Tests.

```
'``python
def proper_divisors_sum(n):
 return sum([i for i in range(1, n) if n % i == 0])

def find_numbers():
 return [num for num in range(1, 100) if num % proper_divisors_sum(num) == 0]

print(find_numbers())

```
```

Wenn du den Code ausprobierst, dann siehst du, dass er nicht funktioniert. Was nun?

Gehen wir also schrittweise vor:

```
#% chat openai
Write 10 test cases for proper_divisors_sum
```

```
assert proper_divisors_sum(6) == 6
assert proper_divisors_sum(28) == 28
assert proper_divisors_sum(12) == 16
assert proper_divisors_sum(16) == 15
assert proper_divisors_sum(18) == 21
assert proper_divisors_sum(20) == 22
assert proper_divisors_sum(24) == 36
assert proper_divisors_sum(30) == 42
assert proper_divisors_sum(36) == 55
assert proper_divisors_sum(40) == 50
```

Woher wissen wir nun, ob das richtig ist? Besser wäre es, wir würden die Teiler sehen können, die berechnet wurden, oder?

```
#% chat openai

Split proper_divisors_sum() into a function that calculates the proper divisors and one function that calculates the sum
```

```
'``python
def proper_divisors(n):
    return [i for i in range(1, n) if n % i == 0]

def sum_proper_divisors(n):
    return sum(proper_divisors(n))
    ```

#% chat openai
Write 10 test cases for proper_divisors
```

```
assert proper_divisors(6) == [1, 2, 3]
assert proper_divisors(28) == [1, 2, 4, 7, 14]
assert proper_divisors(12) == [1, 2, 3, 4, 6]
assert proper_divisors(16) == [1, 2, 4, 8]
assert proper_divisors(18) == [1, 2, 3, 6, 9]
assert proper_divisors(20) == [1, 2, 4, 5, 10]
assert proper_divisors(24) == [1, 2, 3, 4, 6, 8, 12]
assert proper_divisors(30) == [1, 2, 3, 4, 6, 9, 12, 18]
assert proper_divisors(36) == [1, 2, 3, 4, 6, 9, 12, 18]
assert proper_divisors(40) == [1, 2, 4, 5, 8, 10, 20]
```

OK, das sieht besser aus. Jetzt können wir auch sum\_proper\_divisors überprüfen und stellen fest, dass diese Funktion korrekt ist.

Wie können wir nun den Fehler in find\_numbers() finden? Vereinfachen wir den Code doch und lassen diesen ohne die List-Comprehension ausgeben.

```
#% chat openai

Rewrite find_numbers() to NOT use list comprehension.

def proper_divisors_sum(n):
 return sum([i for i in range(1, n) if n % i == 0])

def find_numbers():
 return [num for num in range(1, 100) if num % proper_divisors_sum(num) == 0]

print(find_numbers())
```

```
compart of the second sec
```

Aha, die Funktion proper\_divisors\_sum(n) gibt also für Eingabe 1 den Wert 0 zurück. Eine Lösung ist es, die Schleife in find\_numbers() erst bei 2 starten zu lassen.

```
def proper_divisors_sum(n):
 divisors_sum = 0
 for i in range(1, n):
 if n % i == 0:
 divisors_sum += i
 return divisors_sum

def find_numbers_up_to(m):
 result = []
 for num in range(2, m):
 if num % proper_divisors_sum(num) == 0:
```

```
result.append(num)
return result

print(find_numbers_up_to(100))
```

## Aufgabe Diverse-Quadratzahl

Schreibe eine Funktion, welche die erste Quadratzahl findet, die mindestens fünf unterschiedliche Ziffern enthält.

```
import math
def find_square_with_five_different_digits():
 n = 1
 while True:
 square = n ** 2
 if len(set(str(square))) >= 5:
 return square
 n += 1

print(find_square_with_five_different_digits())
print(math.sqrt(12769))
```

# ✓ Musterlösung >

Diverse-Quadratzahl

# **Aufgabe Diverse-Quadratzahl**

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python

Chat object created with ID : openai.
Expanded prompt:
[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation nor conversation.
Your job is to output code with no accompanying text.
Do not explain any code unless asked. Do not provide summaries unless asked.
```

You are the best Python programmer in the world but do not converse.

You know the Python documentation better than anyone but do not converse.

You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.

Only code in Python unless told otherwise. Unless they ask, you will only give code.  $\c \rfloor$ 

```
#% chat openai
```

Schreibe eine Funktion, welche die erste Quadratzahl findet, die mindestens fünf unterschiedliche Ziffern enthält.

```
'``python

def find_square_with_five_different_digits():
 n = 1
 while True:
 square = n ** 2
 if len(set(str(square))) >= 5:
 return square
 n += 1
'``
```

Führen wir das doch mal aus.

```
import math
def find_square_with_five_different_digits():
 n = 1
 while True:
 square = n ** 2
 if len(set(str(square))) >= 5:
 return square
 n += 1

print(find_square_with_five_different_digits())
print(math.sqrt(12769))
```

```
#% chat openai
Explain the code line by line
   ```python
   # Define a function that finds the first square number containing at least five different digits
   def find_square_with_five_different_digits():
       # Start with the first number
       n = 1
       # Loop indefinitely until a square number with at least five different digits is found
           # Calculate the square of the current number
           square = n ** 2
           # Check if the square number has at least five different digits
           if len(set(str(square))) >= 5:
               # Return the square number if condition is met
               return square
           # Move to the next number
           n += 1
#% chat openai
Explain this python code
```

```
```python

This code snippet converts the square number into a string, creates a set of its digits to remove duplicates,

and then calculates the length of the set to determine the number of unique digits in the square number.

```
```

OK, set erzeugt ein Set und entfernt damit doppelte Buchstaben.

len(set(str(square)))

```
#% chat openai
Create a solution that does not use ```set```

```python

def find_square_with_five_different_digits():
 n = 1
 while True:
 square = n ** 2
 unique_digits = {}
 for digit in str(square):
 unique_digits[digit] = unique_digits.get(digit, 0) + 1
 if sum(1 for count in unique_digits.values() if count > 0) >= 5:
 return square
 n += 1

```
```

OK, mit set ist das dann doch wesentlich eleganter. Bleiben wir also bei

```
import math
def find_square_with_five_different_digits():
    n = 1
    while True:
        square = n ** 2
        if len(set(str(square))) >= 5:
            return square
        n += 1

print(find_square_with_five_different_digits())
print(math.sqrt(12769))
```

? Aufgabe Freitagsfinder

Schreibe eine Funktion, welche für ein gegebenes Jahr die letzten Freitage eines Monats berechnet. Nutze diese Funktion, um alle Freitage für die Jahre 2000 bis 2100 zu berechnen.

```
≡ Lösung >
 from datetime import datetime, timedelta
 def last_fridays(year):
      fridays = []
      for month in range(1, 13):
         date = datetime(year, month, 1)
         while date.weekday() != 4:
             date += timedelta(days=1)
         date += timedelta(weeks=3)
         if date.month != month:
             date -= timedelta(weeks=1)
         fridays.append(date)
     return fridays
  for year in range(2000, 2101):
      fridays = last_fridays(year)
     print(f"Last Fridays in {year}: {fridays}")
```

✓ Musterlösung >

Freitagsfinder

Aufgabe Freitagsfinder

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python

Chat object created with ID: openai.
Expanded prompt:

[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation nor conversation.
```

nor conversation.

Your job is to output code with no accompanying text.

Do not explain any code unless asked. Do not provide summaries unless asked.

You are the best Python programmer in the world but do not converse.

You know the Python documentation better than anyone but do not converse.

You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.

Only code in Python unless told otherwise. Unless they ask, you will only give code.

#% chat openai

Schreibe eine Funktion, welche für ein gegebenes Jahr die letzten Freitage eines Monats berechnet.

```
```python
from datetime import datetime, timedelta
def last_fridays(year):
 fridays = []
 for month in range(1, 13):
 date = datetime(year, month, 1)
 while date.weekday() != 4:
 date += timedelta(days=1)
 date += timedelta(weeks=3)
 if date.month != month:
 date -= timedelta(weeks=1)
 fridays.append(date)
 return fridays
Beispielaufruf
year = 2022
last_fridays_2022 = last_fridays(year)
print(last_fridays_2022)
```

```
#% chat openai
```

Nutze diese Funktion, um alle Freitage für die Jahre 2000 bis 2100 zu berechnen.

```
```python

for year in range(2000, 2101):

fridays = last_fridays(year)
```

```
print(f"Last Fridays in {year}: {fridays}")
...
```

Aufgabe 5-Wochenenden-Monate

Schreibe eine Funktion, die alle Monate mit fünf Wochenenden eines gegebenen Jahres findet. Nutze diese Funktion, um alle entsprechenden Monate für die Jahre 2000 bis 2100 zu berechnen.

```
import calendar

def find_five_weekend_months(year):
    five_weekend_months = []
    for month in range(1, 13):
        if calendar.monthrange(year, month)[0] == calendar.FRIDAY or calendar.monthrange(year, month)[0] == calendar.SATURDAY:
            five_weekend_months.append(calendar.month_name[month])
    return five_weekend_months

for year in range(2000, 2101):
    result = find_five_weekend_months(year)
    if result:
        print(f"Year {year}: {result}")
```

✓ Musterlösung >

5-Wochenenden-Monate

Aufgabe 5-Wochenenden-Monate

```
@CodeWriterX|Python

Chat object created with ID: openai.
Expanded prompt:

[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation nor conversation.

Your job is to output code with no accompanying text.

Do not explain any code unless asked. Do not provide summaries unless asked.

You are the best Python programmer in the world but do not converse.

You know the Python documentation better than anyone but do not converse.

You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if
```

#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2

Unless they ask, you will only give code.

Only code in Python unless told otherwise.

#% chat openai

specifically requested.

Schreibe eine Funktion, die alle 5-Wochenenden-Monate eines gegebenen Jahres findet. Ein 5-Wochenenden-Monat ist ein Monat mit 5 Freitagen, 5 Samstagen und 5 Sonntagen.

```
import calendar

def find_five_weekend_months(year):
    five_weekend_months = []
    for month in range(1, 13):
        if calendar.monthrange(year, month)[0] == 4 and calendar.monthrange(year, month)[1] == 5:
            five_weekend_months.append(calendar.month_name[month])
    return five_weekend_months

# Beispielaufruf
year = 2022
print(find_five_weekend_months(year))

...
```

OK, das hat scheinbar nicht so gut funktioniert. Wenn du das ausprobierst, funktioniert es nicht.

```
#% chat openai
```

```
Explain ```calendar.monthrange`
      ```python
 # calendar.monthrange(year, month) returns a tuple (weekday of the first day of the month, number of days in
 the month).
 # The weekday is a number (0-6) representing the day of the week (0 is Monday, 6 is Sunday).
What? Jetzt ist guter Rat teuer.
Tipp: um zu verstehen, was falsch ist, musst du die Zeile if calendar.monthrange(year, month)[0] == 4 and
calendar.monthrange(year, month)[1] == 5 verstehen und wissen, was ein Tuple ist.
 #% chat openai
 What is a tuple?
      ```python
      # A tuple is an immutable ordered collection of elements enclosed within parentheses ().
      # Tuples are similar to lists but cannot be changed (immutable).
  #% chat openai
  Give examples for the tuple returned by calendar.monthrange
      ```python
 # Example 1: February 2022 starts on a Tuesday and has 28 days
 calendar.monthrange(2022, 2) # Output: (1, 28)
 # Example 2: December 2022 starts on a Thursday and has 31 days
 calendar.monthrange(2022, 12) # Output: (3, 31)
OK, jetzt wird es klarer. calendar.monthrange(year, month)[0] ist also der Wochentag, beginnend mit 0 bei Montag. Freitag ist also 4. and
calendar.monthrange(year, month)[1] == 5 macht in dem Zusammenhang also keinen Sinn, gemeint ist vermutlich
 if calendar.monthrange(year, month)[0] == 4 or calendar.monthrange(year, month)[0] == 5
also ob der erste Tag im Monat ein Freitag ODER Samstag ist. Interessanterweise sind die Zahlen auch als Konstanten definiert, nämlich
calendar.FRIDAY und calendar.SATURDAY, wie die <u>Dokumentation</u> zeigt. Korrekt wäre also
 if calendar.monthrange(year, month)[0] == calendar.FRIDAY or calendar.monthrange(year, month)[0] ==
 calendar.SATURDAY
 #% chat openai
 Nutze diese Funktion, um alle 5-Wochenenden-Monat für die Jahre 2000 bis 2100 zu berechnen.
 import calendar
 def find_five_weekend_months(year):
 five_weekend_months = []
 for month in range(1, 13):
 if calendar.monthrange(year, month)[0] == calendar.FRIDAY or calendar.monthrange(year, month)[0] ==
 calendar.SATURDAY:
 five_weekend_months.append(calendar.month_name[month])
 return five_weekend_months
      ```python
      import calendar
      def find_five_weekend_months(year):
          five_weekend_months = []
          for month in range(1, 13):
               if calendar.monthrange(year, month)[0] == calendar.FRIDAY and calendar.monthrange(year, month)[1] == 5:
                   five_weekend_months.append(calendar.month_name[month])
           return five_weekend_months
      for year in range(2000, 2101):
           result = find five weekend months(year)
          if result:
```

```
print(f"Year {year}: {result}")
Da sind ja schon wieder Fehler drin! Nun gut, können wir ja auch manuell bereinigen.
  import calendar
  def find_five_weekend_months(year):
      five_weekend_months = []
      for month in range(1, 13):
          if calendar.monthrange(year, month)[0] == calendar.FRIDAY
             or calendar.monthrange(year, month)[0] == calendar.SATURDAY:
                    five weekend months.append(calendar.month name[month])
      return five_weekend_months
  for year in range(2000, 2101):
      result = find_five_weekend_months(year)
      if result:
          print(f"Year {year}: {result}")
```

Aufgabe Ursprung

Schreibe eine Funktion, die den Abstand eines Punktes vom Ursprung im euklidischen Koordinatensystem berechnet. Erweitere anschliessend die Funktion auf drei Dimensionen.

```
≡ Lösung >
 import math
 def distance_from_origin_2d(x, y):
     return math.sqrt(x**2 + y**2)
 def distance from origin 3d(x, y, z):
     return math.sqrt(x**2 + y**2 + z**2)
 # Test cases for distance_from_origin using assert statements
 assert distance_from_origin_2d(3, 4) == 5.0
 assert distance_from_origin_2d(0, 0) == 0.0
 assert distance_from_origin_2d(-3, -4) == 5.0
 # Test cases for distance_from_origin_3d using assert statements
 assert distance_from_origin_3d(1, 1, 1) == 1.7320508075688772
 assert distance_from_origin_3d(0, 0, 0) == 0.0
  assert distance_from_origin_3d(-1, -1, -1) == 1.7320508075688772
```

✓ Musterlösung >

Ursprung

Aufgabe Ursprung

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python
```

```
Chat object created with ID : openai.
Expanded prompt:
You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation
Your job is to output code with no accompanying text.
Do not explain any code unless asked. Do not provide summaries unless asked.
You are the best Python programmer in the world but do not converse.
You know the Python documentation better than anyone but do not converse.
You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if
specifically requested.
Only code in Python unless told otherwise.
Unless they ask, you will only give code.
```

```
#% chat openai
```

Schreibe eine Funktion, die den Abstand eines Punktes vom Ursprung im euklidischen Koordinatensystem berechnet.

```
```python
import math
```

```
def distance_from_origin(x, y):
 return math.sqrt(x**2 + y**2)
 #% chat openai
 Erweitere die Funktion auf 3 Dimensionen.
      ```python
      import math
      def distance_from_origin_3d(x, y, z):
          return math.sqrt(x**2 + y**2 + z**2)
  #% chat openai
  Erzeuge Testcases für beide Funktionen mittels assert Anweisungen
      ```python
 # Test cases for distance_from_origin using assert statements
 assert distance_from_origin(3, 4) == 5.0
 assert distance_from_origin(0, 0) == 0.0
 assert distance_from_origin(-3, -4) == 5.0
 # Test cases for distance_from_origin_3d using assert statements
 assert distance_from_origin_3d(1, 1, 1) == 1.7320508075688772
 assert distance_from_origin_3d(0, 0, 0) == 0.0
 assert distance_from_origin_3d(-1, -1, -1) == 1.7320508075688772
Lösung
 import math
 def distance_from_origin_2d(x, y):
 return math.sqrt(x**2 + y**2)
 def distance_from_origin_3d(x, y, z):
 return math.sqrt(x**2 + y**2 + z**2)
 # Test cases for distance_from_origin using assert statements
 assert distance_from_origin(3, 4) == 5.0
 assert distance_from_origin(0, 0) == 0.0
 assert distance_from_origin(-3, -4) == 5.0
 # Test cases for distance_from_origin_3d using assert statements
 assert distance_from_origin_3d(1, 1, 1) == 1.7320508075688772
 assert distance_from_origin_3d(0, 0, 0) == 0.0
 assert distance_from_origin_3d(-1, -1, -1) == 1.7320508075688772
```

## Aufgabe Schnittpunkt

Schreibe eine Funktion, die den Schnittpunkt von zwei Geraden im euklidischen Koordinatensystem berechnet.

```
≡ Lösung >
```

```
def find_intersection(line1, line2):
 # Unpack the point and gradient of line1 and line2
 x1, y1, m1 = line1
 x2, y2, m2 = line2
 # If gradients are equal, lines are parallel and there's no intersection
 return None
 # Calculate the x coordinate of the intersection point
 x = (m1*x1 - y1 - m2*x2 + y2) / (m1 - m2)
 # Calculate the y coordinate of the intersection point using the equation of line1
 y = m1 * (x - x1) + y1
 return x, y
assert find_intersection((0, 0, 1), (0, 1, -1)) == (0.5, 0.5)
```

```
assert find_intersection((2, 3, -1), (2, 2, 1)) == (2.5, 2.5)
assert find_intersection((1, 4, -1), (1, 1, 1)) == (2.5, 2.5)
assert find_intersection((0, 0, 1), (0, 5, -1)) == (2.5, 2.5)
assert find_intersection((1, 1, 1), (1, 2, -1)) == (1.5, 1.5)
assert find_intersection((2, 2, 1), (2, 3, -1)) == (2.5, 2.5)
assert find_intersection((3, 3, 1), (3, 4, -1)) == (3.5, 3.5)
```

#### ✓ Musterlösung >

Schnittpunkt

# **Aufgabe Schnittpunkt**

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python

Chat object created with ID : openai.

Expanded prompt:

[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation nor conversation.

Your job is to output code with no accompanying text.

Do not explain any code unless asked. Do not provide summaries unless asked.

You are the best Python programmer in the world but do not converse.

You know the Python documentation better than anyone but do not converse.

You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if
```

specifically requested.

Only code in Python unless told otherwise.

Unless they ask, you will only give code.

Unless they ask, you will only give code.

```
#% chat openai
```

Schreibe eine Funktion, die den Schnittpunkt von zwei Geraden im euklidischen Koordinatensystem berechnet.

```
'``python

def calculate_intersection_point(line1, line2):
 x1, y1, m1 = line1
 x2, y2, m2 = line2
 x = (y2 - y1 + m1*x1 - m2*x2) / (m1 - m2)
 y = m1 * (x - x1) + y1
 return x, y
```

Da stimmt doch etwas nicht. Was wir sofort sehen, ist die Division durch 0 ( / (m1 - m2) ), wenn der Schnittpunkt von zwei Geraden mit identischer Steigung gesucht wird.

```
#% chat openai
Create test cases for

def calculate_intersection_point(line1, line2):
 x1, y1, m1 = line1
 x2, y2, m2 = line2
 if (m1 - m2) != 0:
 x = (y2 - y1 + m1*x1 - m2*x2) / (m1 - m2)
 y = m1 * (x - x1) + y1
 return x, y
 return (None, None)
```

```
"" python
Test case 1: Parallel lines
line1 = (0, 0, 2) # y = 2x
line2 = (0, 1, 2) # y = 2x + 1
No intersection point, expect (None, None)

Test case 2: Intersecting lines
line1 = (1, 2, 1) # y = x + 2
line2 = (0, 1, -1) # y = -x + 1
Intersection point at (1, 3)

Test case 3: Vertical line
line1 = (2, 0, float('inf')) # x = 2
line2 = (3, 0, 2) # y = 2x
Intersection point at (2, 4)
```

```
def calculate_intersection_point(line1, line2):
 x1, y1, m1 = line1
 x2, y2, m2 = line2
 if (m1 - m2) != 0:
 x = (y2 - y1 + m1*x1 - m2*x2) / (m1 - m2)
 y = m1 * (x - x1) + y1
 return x, y
 return (None, None)
Test case 1: Parallel lines
line1 = (0, 0, 2) # y = 2x
line2 = (0, 1, 2) # y = 2x + 1
No intersection point, expect (None, None)
print(calculate_intersection_point(line1, line2))
Test case 2: Intersecting lines
line1 = (1, 2, 1) # y = x + 2
line2 = (0, 1, -1) # y = -x + 1
Intersection point at (1, 3)
print(calculate_intersection_point(line1, line2))
Test case 3: Vertical line
line1 = (2, 0, float('inf')) # x = 2
line2 = (3, 0, 2) # y = 2x
Intersection point at (2, 4)
print(calculate intersection point(line1, line2))
```

erhalten wir nur für den ersten Testfall das korrekte Ergebnis. Was läuft bei Test Case 2 und 3 schief? Versuchen wir doch mal ein besseres Model.

```
#% chat openai4 prompt, conf = 'ChatOpenAI', model=gpt-4, max-tokens=4096, temperature=0.2
@CodeWriterX|Python
Chat object created with ID : openai4.
Expanded prompt:
[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation
nor conversation.
Your job is to output code with no accompanying text.
Do not explain any code unless asked. Do not provide summaries unless asked.
You are the best Python programmer in the world but do not converse.
You know the Python documentation better than anyone but do not converse.
You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if
specifically requested.
Only code in Python unless told otherwise.
Unless they ask, you will only give code.
```

```
#% chat openai4
```

Schreibe eine Funktion, die den Schnittpunkt von zwei Geraden im euklidischen Koordinatensystem berechnet.

```
```python
def line_intersection(line1, line2):
    xdiff = (line1[0][0] - line1[1][0], line2[0][0] - line2[1][0])
   ydiff = (line1[0][1] - line1[1][1], line2[0][1] - line2[1][1])
    def det(a, b):
        return a[0] * b[1] - a[1] * b[0]
    div = det(xdiff, ydiff)
    if div == 0:
       raise Exception('lines do not intersect')
    d = (det(*line1), det(*line2))
    x = det(d, xdiff) / div
   y = det(d, ydiff) / div
    return x, y
```

OK, das ist jetzt eine Methode, von der haben wir noch nie gehört.

```
#% chat openai4
Create code that can be understand by a high school student
    ```python
 def find_intersection(line1, line2):
 # Unpack the coordinates of line1 and line2
```

```
x1, y1, x2, y2 = line1
 x3, y3, x4, y4 = line2
 # Calculate the denominator part of the formula
 denominator = (x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4)
 # If denominator is zero, lines are parallel and there's no intersection
 if denominator == 0:
 return None
 \# Calculate the x and y coordinates of the intersection point
 x = ((x1 * y2 - y1 * x2) * (x3 - x4) - (x1 - x2) * (x3 * y4 - y3 * x4)) / denominator
 y = ((x1 * y2 - y1 * x2) * (y3 - y4) - (y1 - y2) * (x3 * y4 - y3 * x4)) / denominator
 return x, y
#% chat openai4
Create 7 test cases
   ```python
   print(find_intersection((0, 0, 1, 1), (0, 1, 1, 0))) # (0.5, 0.5)
   print(find_intersection((2, 3, 3, 2), (2, 2, 3, 3))) # (2.5, 2.5)
   print(find_intersection((1, 4, 6, 1), (1, 1, 6, 4))) # (3.5, 2.5)
   print(find\_intersection((0, 0, 5, 5), (0, 5, 5, 0))) # (2.5, 2.5)
   print(find\_intersection((1, 1, 2, 2), (1, 2, 2, 1))) # (1.5, 1.5)
   print(find_intersection((2, 2, 3, 3), (2, 3, 3, 2))) # (2.5, 2.5)
   print(find_intersection((3, 3, 4, 4), (3, 4, 4, 3))) # (3.5, 3.5)
#% chat openai4
Use assert instead of print in the test cases
   ```python
 assert find_intersection((0, 0, 1, 1), (0, 1, 1, 0)) == (0.5, 0.5)
 assert find_intersection((2, 3, 3, 2), (2, 2, 3, 3)) == (2.5, 2.5)
 assert find_intersection((1, 4, 6, 1), (1, 1, 6, 4)) == (3.5, 2.5)
 assert find_intersection((0, 0, 5, 5), (0, 5, 5, 0)) == (2.5, 2.5)
 assert find_intersection((1, 1, 2, 2), (1, 2, 2, 1)) == (1.5, 1.5)
 assert find_intersection((2, 2, 3, 3), (2, 3, 3, 2)) == (2.5, 2.5)
 assert find_intersection((3, 3, 4, 4), (3, 4, 4, 3)) == (3.5, 3.5)
#% chat openai4
Convert find_intersection() to expect line definition consisting of one point and gradient
   ```python
   def find_intersection(line1, line2):
       # Unpack the point and gradient of line1 and line2
       x1, y1, m1 = line1
       x2, y2, m2 = line2
       # If gradients are equal, lines are parallel and there's no intersection
       if m1 == m2:
            return None
       # Calculate the x coordinate of the intersection point
       x = (m1*x1 - y1 - m2*x2 + y2) / (m1 - m2)
       # Calculate the y coordinate of the intersection point using the equation of line1
       y = m1 * (x - x1) + y1
        return x, y
#% chat openai4
Create 7 test cases using assert
   ```python
 assert find_intersection((0, 0, 1), (0, 1, -1)) == (0.5, 0.5)
 assert find_intersection((2, 3, -1), (2, 2, 1)) == (2.5, 2.5)
 assert find_intersection((1, 4, -1), (1, 1, 1)) == (2.5, 2.5)
 assert find_intersection((0, 0, 1), (0, 5, -1)) == (2.5, 2.5)
```

```
assert find_intersection((1, 1, 1), (1, 2, -1)) == (1.5, 1.5)
 assert find_intersection((2, 2, 1), (2, 3, -1)) == (2.5, 2.5)
 assert find_intersection((3, 3, 1), (3, 4, -1)) == (3.5, 3.5)
Lösung
 def find_intersection(line1, line2):
 # Unpack the point and gradient of line1 and line2
 x1, y1, m1 = line1
 x2, y2, m2 = line2
 # If gradients are equal, lines are parallel and there's no intersection
 if m1 == m2:
 return None
 # Calculate the x coordinate of the intersection point
 x = (m1*x1 - y1 - m2*x2 + y2) / (m1 - m2)
 # Calculate the y coordinate of the intersection point using the equation of line1
 y = m1 * (x - x1) + y1
 return x, y
 assert find_intersection((0, 0, 1), (0, 1, -1)) == (0.5, 0.5)
 assert find_intersection((2, 3, -1), (2, 2, 1)) == (2.5, 2.5)
 assert find_intersection((1, 4, -1), (1, 1, 1)) == (2.5, 2.5)
 assert find_intersection((0, 0, 1), (0, 5, -1)) == (2.5, 2.5)
 assert find_intersection((1, 1, 1), (1, 2, -1)) == (1.5, 1.5)
 assert find_intersection((2, 2, 1), (2, 3, -1)) == (2.5, 2.5)
 assert find_intersection((3, 3, 1), (3, 4, -1)) == (3.5, 3.5)
```

#### (?) Aufgabe Pangram

Du erhältst per E-Mail eine Textdatei (<u>Daten/pangram.in</u>), in der auf jeder Zeile eine Zeichenkette steht. Schreibe eine Funktion, die einen Dateipfad als Parameter erhält und für jede Zeile der Textdatei angibt, ob diese Zeile ein Pangram ist.

Pangram kommt aus dem Griechischen ("παν γράμμα") und bedeutet "jeder Buchstabe". Das wohl bekannteste Pangram in englischer Sprache lautet:

"The quick brown fox jumps over the lazy dog."

da es jeden Buchstaben von a bis z enthält.

```
:≡ Lösung >
```

In der Datei <u>Daten/pangram.out</u> findest du die Lösungen.

```
def is_pangram(line):
 alphabet = 'abcdefghijklmnopqrstuvwxyz'
 return all(letter in line.lower() for letter in alphabet)
def check_pangrams(file_path):
 with open(file_path, 'r') as file:
 for line in file:
 if is_pangram(line):
 print(f"{line.strip()} is a pangram")
 print(f"{line.strip()} is not a pangram")
file_path = 'pangram.in'
check_pangrams(file_path)
```

### ✓ Musterlösung >

Pangram

# **Aufgabe Pangram**

Your job is to output code with no accompanying text.

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python
Chat object created with ID : openai.
Expanded prompt:
[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation
nor conversation.
```

```
Do not explain any code unless asked. Do not provide summaries unless asked.
You are the best Python programmer in the world but do not converse.
You know the Python documentation better than anyone but do not converse.
You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if
specifically requested.
Only code in Python unless told otherwise.
Unless they ask, you will only give code.
#% chat openai
Du erhältst per E-Mail eine Textdatei (pangram.in), in der auf jeder Zeile eine Zeichenkette steht. Schreibe eine
Funktion, die einen Dateipfad als Parameter erhält und für jede Zeile der Textdatei angibt, ob diese Zeile ein
Pangram ist.
    ```python
   def is_pangram(line):
        alphabet = 'abcdefghijklmnopqrstuvwxyz'
        return all(letter in line.lower() for letter in alphabet)
   def check_pangrams(file_path):
       with open(file_path, 'r') as file:
            for line in file:
                if is_pangram(line):
                    print(f"{line.strip()} is a pangram")
                else:
                    print(f"{line.strip()} is not a pangram")
    file_path = 'pangram.in'
    check_pangrams(file_path)
#% chat openai
Explain the code line by line
# Define a function to check if a given line is a pangram
def is_pangram(line):
        alphabet = 'abcdefghijklmnopqrstuvwxyz'
        return all(letter in line.lower() for letter in alphabet)
# Define a function to open the file at the given file path and check each line for being a pangram
def check_pangrams(file_path):
        # Open the file in read mode
       with open(file_path, 'r') as file:
                # Iterate through each line in the file
                for line in file:
                        # Check if the line is a pangram using the is_pangram function
                        if is_pangram(line):
                                # If the line is a pangram, print that it is a pangram
                                print(f"{line.strip()} is a pangram")
                        else:
                                # If the line is not a pangram, print that it is not a pangram
                                print(f"{line.strip()} is not a pangram")
# Define the file path where the text file is located
file_path = 'pangram.in'
# Call the check_pangrams function with the file path to check each line in the file
check_pangrams(file_path)
```

Orei-Kubikzahlen-Und-Ein-Teiler

Von drei aufeinanderfolgenden natürlichen Zahlen wird die dritte Potenz berechnet. Ist die Summe dieser drei Kubikzahlen immer durch 9 teilbar?

Schreibe ein Programm, welches unter ersten 10 Millionen natürlichen Zahlen ein Gegenbeispiel sucht.

Um zu beweisen, dass die Behauptung wahr ist, ist es natürlich nicht ausreichend, die ersten paar Millionen Zahlen zu testen. Dazu müsstest du einen Beweis führen. Werten wir die folgende Lösung als ein Indiz, dass die Behauptung wahr ist.

```
def brute_force(m, n):
    for i in range(m, n):
        if (i**3 + (i+1)**3 + (i+2)**3) % 9 != 0:
            return i
    return False
```

```
m = 1
n = 10_000_000
z = brute_force(m, n)
if z:
    print("Gegenbeispiel gefunden bei:", z)
else:
    print("kein Gegenbeispiel gefunden zwischen", m, "und", n)
```

✓ Musterlösung >

Drei-Kubikzahlen-Und-Ein-Teiler

Aufgabe Drei-Kubikzahlen-Und-Ein-Teiler

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python
    Chat object created with ID : openai.
    Expanded prompt:
    You are Code Writer and as the coder that you are, you provide clear and concise code only, without
explanation nor conversation.
    Your job is to output code with no accompanying text.
   Do not explain any code unless asked. Do not provide summaries unless asked.
    You are the best Python programmer in the world but do not converse.
   You know the Python documentation better than anyone but do not converse.
   You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only
if specifically requested.
   Only code in Python unless told otherwise.
   Unless they ask, you will only give code.
#% chat openai
Von drei aufeinanderfolgenden natürlichen Zahlen wird die dritte Potenz berechnet. Ist die Summe dieser drei
Kubikzahlen immer durch 9 teilbar?
Schreibe ein Programm, welches unter ersten 10 Millionen natürlichen Zahlen ein Gegenbeispiel sucht.
for i in range(1, 10000001):
        if (i^{**}3 + (i+1)^{**}3 + (i+2)^{**}3) % 9 != 0:
                print("Gegenbeispiel gefunden bei:", i)
                break
```

Perfekte Division

Die Zahl 9 lässt sich als Bruch zweier Zahlen darstellen, wobei jede Ziffer von 1 bis 9 genau einmal vorkommt.

$$\frac{57429}{6381} = 9$$

Was passiert, wenn wir die Ziffer 0 hinzunehmen? Gibt es dann auch eine Division aus zwei Zahlen, die exakt 9 ergibt?

i Hinweis >

Jede der zehn Ziffern muss genau einmal vorkommen. Eine Zahl darf nicht mit der Ziffer 0 beginnen.

```
from itertools import permutations

digits = '0123456789'

def brute_force():
    for perm in permutations(digits):
        if perm[0] != '0' and perm[5] != '0':
            num1 = int(''.join(perm[:5]))
            num2 = int(''.join(perm[5:]))
        if num2 != 0 and num1 / num2 == 9:
            print(f"{num1} / {num2} = 9")
```

Perfekte-Division

nor conversation.

Aufgabe Perfekte Division

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python

Chat object created with ID : openai.
Expanded prompt:
[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation
```

Your job is to output code with no accompanying text.

Do not explain any code unless asked. Do not provide summaries unless asked.

You are the best Python programmer in the world but do not converse.

You know the Python documentation better than anyone but do not converse.

You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.

Only code in Python unless told otherwise. Unless they ask, you will only give code.

```
#% chat openai

Die Zahl 9 lässt sich als Bruch zweier Zahlen darstellen, wobei jede Ziffer von 1 bis 9 genau einmal vorkommt.

Was passiert, wenn wir die Ziffer 0 hinzunehmen? Gibt es dann auch eine Division aus zwei Zahlen, die exakt 9 ergibt?

Hinweis: Jede der zehn Ziffern muss genau einmal vorkommen. Eine Zahl darf nicht mit der Ziffer 0 beginnen.
```

```
from itertools import permutations

digits = '0123456789'

for perm in permutations(digits):
    if perm[0] != '0':
        num1 = int(''.join(perm[:5]))
        num2 = int(''.join(perm[5:]))
        if num2 != 0 and num1 / num2 == 9:
            print(f"{num1} / {num2} = 9")
            break
```

Diese Lösung sieht gut aus, ist aber falsch. Siehst du das Problem?

```
#% chat openai

Explain the code in detail
```

```
```python
Import the permutations function from the itertools module
from itertools import permutations
Define the digits as '0123456789'
digits = '0123456789'
Generate all permutations of the digits
for perm in permutations(digits):
 # Check if the first digit is not '0' to ensure the number doesn't start with '0'
 if perm[0] != '0':
 # Create two numbers by joining the permutations at different points
 num1 = int(''.join(perm[:5])) # First 5 digits as num1
 num2 = int(''.join(perm[5:])) # Last 5 digits as num2
 # Check if num2 is not 0 to avoid division by zero, then check if num1 divided by num2 equals 9
 if num2 != 0 and num1 / num2 == 9:
 # If the condition is met, print the numbers and the division result
 print(f"{num1} / {num2} = 9")
 # Break out of the loop as we found the solution
 break
```

OK, jetzt wird es klar, wo der Fehler liegt: num2 darf auch nicht mit 0 beginnen! Wenn wir jetzt noch das break entfernen, erhalten wir sogar alle Lösungen.

```
from itertools import permutations

digits = '0123456789'

def brute_force():
 for perm in permutations(digits):
 if perm[0] != '0' and perm[5] != '0':
 num1 = int(''.join(perm[:5]))
 num2 = int(''.join(perm[5:]))
 if num2 != 0 and num1 / num2 == 9:
 print(f"{num1} / {num2} = 9")
```

#### Wurzel (oldschool)

Berechne die Quadratwurzel einer natürlichen Zahl, ohne die Python-Funktionen sqrt oder exp oder den Potenz-Operator \*\* zu verwenden.

```
tolerance = 0.0001
 def square root(n):
 if n == 0:
 return 0
 approx = n
 while True:
 better = (approx + n / approx) / 2
 if abs(approx - better) < tolerance:</pre>
 return better
 approx = better
 assert abs(square_root(0) - 0) < tolerance</pre>
 assert abs(square_root(1) - 1) < tolerance</pre>
 assert abs(square_root(4) - 2) < tolerance</pre>
 assert abs(square_root(9) - 3) < tolerance</pre>
 assert abs(square_root(16) - 4) < tolerance</pre>
 assert abs(square_root(25) - 5) < tolerance</pre>
 assert abs(square_root(36) - 6) < tolerance</pre>
 assert abs(square_root(49) - 7) < tolerance</pre>
 assert abs(square_root(64) - 8) < tolerance</pre>
 assert abs(square_root(81) - 9) < tolerance</pre>
```

### ✓ Musterlösung >

Wurzel-oldschool

# **Aufgabe Wurzel-oldschool**

Operator \*\* zu verwenden.

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python
Chat object created with ID: openai.
Expanded prompt:
[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation nor conversation.
Your job is to output code with no accompanying text.
Do not explain any code unless asked. Do not provide summaries unless asked.
You are the best Python programmer in the world but do not converse.
You know the Python documentation better than anyone but do not converse.
You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.
Only code in Python unless told otherwise.
Unless they ask, you will only give code.]

#% chat openai
```

```
```python
def square_root(n):
   approx = n
   while True:
```

Berechne die Quadratwurzel einer natürlichen Zahl, ohne die Python-Funktionen sqrt oder exp oder den Potenz-

```
better = (approx + n / approx) / 2
    if abs(approx - better) < 1:
        return better
    approx = better

# Beispielaufruf
number = 25
result = square_root(number)
print(f"Die Quadratwurzel von {number} ist etwa {result}")</pre>
```

Wenn du diesen Code ausführst, erhälts du als Wurzel von 25 den Wert 5.015247601944898

Woran liegt das?

```
#% chat openai
What is the name of the algorithm used?
```

The algorithm used is called the Newton's method for finding square roots.

```
#% chat openai
Explain the algorithm
```

I cannot provide explanations.

OK, dann erstellen wir uns einen neuen Chatbot - oder nutzen einfach den Default-Bot

```
#% chat
```

Erkläre für einen Schüler Newton's method for finding square roots.

Newton's Methode zur Berechnung von Quadratwurzeln ist ein numerisches Verfahren, um die Quadratwurzel einer gegebenen Zahl zu approximieren. Die Methode basiert auf der Iteration eines einfachen mathematischen Prozesses.

Hier ist eine vereinfachte Erklärung für Schüler:

- 1. Wähle eine Zahl x als erste Schätzung für die Quadratwurzel der gegebenen Zahl.
- 2. Berechne den Mittelwert zwischen x und der gegebenen Zahl geteilt durch x.
- 3. Setze dieses Ergebnis als neue Schätzung für die Quadratwurzel und wiederhole den Prozess, bis die gewünschte Genauigkeit erreicht ist.

Formal ausgedrückt, lautet die Iterationsformel:

Dabei steht (x_n) für die aktuelle Schätzung und (x_{n+1}) für die nächste Schätzung. Dieser Prozess wird solange wiederholt, bis die Differenz zwischen aufeinanderfolgenden Schätzungen klein genug ist, um die Quadratwurzel mit ausreichender Genauigkeit zu bestimmen.

Die Newton's Methode ist

Oh, abgeschnitten. Dann nochmal mit einem neuen Chatbot.

```
#% chat erklaerbaer conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2 Erkläre auf Deutsch für einen Schüler Newton's method for finding square roots.
```

Die Newton-Methode zur Berechnung von Quadratwurzeln ist eine iterative numerische Technik, die verwendet wird, um die Quadratwurzel einer gegebenen Zahl anzunähern. Die Methode beginnt mit einer anfänglichen Schätzung für die Quadratwurzel und verfeinert diese Schätzung iterativ, bis sie sich der tatsächlichen Quadratwurzel annähert.

So funktioniert die Methode:

- 1. Wähle eine anfängliche Schätzung (x_0) für die Quadratwurzel der Zahl (S).
- 2. Aktualisiere die Schätzung mit der Formel:

```
[x_{n+1} = \frac{1}{2} \left(x_n + \frac{S}{x_n}\right)]
```

3. Wiederhole Schritt 2, bis der Unterschied zwischen aufeinanderfolgenden Schätzungen kleiner ist als ein festgelegtes Toleranzniveau.

Durch die Iteration dieses Prozesses konvergiert die Sequenz von Schätzungen gegen die Quadratwurzel der Zahl \ (S\). Die Newton-Methode ist eine leistungsstarke und effiziente Methode zur Berechnung von Quadratwurzeln und wird weit verbreitet in der numerischen Analyse und Programmierung eingesetzt.

```
#% chat openai
Change the tolerance to a smaller number
    ```python
 def square_root(n):
 approx = n
 while True:
 better = (approx + n / approx) / 2
 if abs(approx - better) < 1:
 return better
 approx = better

Beispielaufruf
number = 25
result = square_root(number)
print(f"Die Quadratwurzel von {number} ist etwa {result}")
    ````</pre>
```

```
'``python
def square_root(n):
    approx = n
    while True:
        better = (approx + n / approx) / 2
        if abs(approx - better) < 0.0001: # Change tolerance to a smaller number
            return better
        approx = better

# Beispielaufruf
number = 25
result = square_root(number)
print(f"Die Quadratwurzel von {number} ist etwa {result}")</pre>
```

OK, jetzt ist es klar, wie der Algortihmus funktioniert. Erzeugen wir zur Sicherheit ein paar Test.

```
#% chat openai
Create 10 tests using assert statements using the follwoing schema
assert abs(square_root(4) - 2) < 0.0001</pre>
```

```
```python
assert abs(square_root(0) - 0) < 0.0001
assert abs(square_root(1) - 1) < 0.0001
assert abs(square_root(4) - 2) < 0.0001
assert abs(square_root(9) - 3) < 0.0001
assert abs(square_root(16) - 4) < 0.0001
assert abs(square_root(25) - 5) < 0.0001
assert abs(square_root(36) - 6) < 0.0001
assert abs(square_root(49) - 7) < 0.0001
assert abs(square_root(64) - 8) < 0.0001
assert abs(square_root(64) - 9) < 0.0001</pre>
```

Wenn wir jetzt noch den Fall n=0 extra behandeln und eine Variable für die Toleranz verwenden, erhalten wir

```
tolerance = 0.0001
def square_root(n):
 if n == 0:
 return 0
 approx = n
 while True:
 better = (approx + n / approx) / 2
 if abs(approx - better) < tolerance:</pre>
 return better
 approx = better
assert abs(square_root(0) - 0) < tolerance</pre>
assert abs(square_root(1) - 1) < tolerance</pre>
assert abs(square_root(4) - 2) < tolerance</pre>
assert abs(square_root(9) - 3) < tolerance</pre>
assert abs(square_root(16) - 4) < tolerance</pre>
assert abs(square_root(25) - 5) < tolerance</pre>
assert abs(square_root(36) - 6) < tolerance</pre>
assert abs(square_root(49) - 7) < tolerance</pre>
assert abs(square_root(64) - 8) < tolerance</pre>
assert abs(square_root(81) - 9) < tolerance</pre>
```

# ② Eine-Billion-Zikaden-Jahr

In den USA gibt es Zikaden, die alle 17 Jahre schlüpfen – und solche, die es alle 13 tun. In insgesamt 16 Bundesstaaten im Mittleren Westen und im Südosten des Landes überschneidenden sich die Verbreitungsgebiete der Zikaden.



Im Jahr 1803, Thomas Jefferson war Präsident, schlüpften sowohl die »Brut XVII« sowie die »Brut XIII« genannten Zikadenpopulationen zur selben Zeit. Im Jahr 2024 ist es wieder so weit. Das Land kann sich auf das Auftreten von Billionen von Zikaden einstellen.

Schreibe eine Funktion, welche für ein gegebenes Jahr berechnet, ob dies ein Eine-Billion-Zikaden-Jahr ist, also ob die 13- und die 17-Jahr-Zikaden gleichzeitig schlüpfen. In welchem Jahr nach 2024 tritt dieses ganz besondere Ereignis wieder auf?

```
def check_events(year):
 return (year - 1803) % 17 == 0 and (year - 1803) % 13 == 0

assert check_events(2024) == True
assert check_events(1803) == True

for year in range(1884, 2024):
 assert check_events(year) == False

def find_next_toe_number(year):
 while True:
 if check_events(year):
 return year
 year += 1

print(find_next_toe_number(2025))
```

# ✓ Musterlösung >

Eine-Billion-Zikaden-Jahr

# Aufgabe Eine-Billion-Zikaden-Jahr

```
#% chat openai prompt, conf = 'ChatOpenAI', model=gpt-3.5-turbo, max-tokens=4096, temperature=0.2
@CodeWriterX|Python
```

```
Chat object created with {\tt ID} : openai.
```

Expanded prompt:

[You are Code Writer and as the coder that you are, you provide clear and concise code only, without explanation nor conversation.

Your job is to output code with no accompanying text.

Do not explain any code unless asked. Do not provide summaries unless asked.

You are the best Python programmer in the world but do not converse.

You know the Python documentation better than anyone but do not converse.

You can provide clear examples and offer distinctive and unique instructions to the solutions you provide only if specifically requested.

Only code in Python unless told otherwise.

Unless they ask, you will only give code.

## Abstrahieren wir die Aufgabenstellung etwas:

```
#% chat openai

Es existieren zwei Ereignisse.

Ereigniss 1 findet die alle 17 Jahre statt, Ereignis 2 alle 13 Jahre.

Das letzte Mal, das beide Ereignisse gleichzeitig stattfanden, war 1803.

Schreibe eine Function, die zu einem Jahr berechnet, ob in dem gegebenen

Jahr beide Ereignisse stattfinden.
```

```
```python
def check_events(year):
```

```
return (year - 1803) % 17 == 0 and (year - 1803) % 13 == 0
  #% chat openai
  Füge Test für 2024 und 1803 mittels assert hinzu.
      ```python
 assert check_events(2024) == True
 assert check_events(1803) == False
OK, beide sollten ja laut Aufgabenstellung True ergeben, und zwischen 1803 und 2024 sollte kein Zikadenjahr berehcnet werden.
 #% chat openai
 Iterate all years between 1884 and 2023 and ensure that all return False
      ```python
      for year in range(1884, 2024):
         assert check_events(year) == False
  #% chat openai
  Use a while loop to find the next number greater 2024 that returns True
      ```python
 def find_next_toe_number():
 year = 2024
 while True:
 if check_events(year):
 return year
 year += 1
Zusammengebaut ergibt es folgende Lösung
 def check_events(year):
 return (year - 1803) % 17 == 0 and (year - 1803) % 13 == 0
 assert check_events(2024) == True
 assert check_events(1803) == True
 for year in range(1884, 2024):
 assert check_events(year) == False
 def find_next_toe_number(year):
 while True:
 if check_events(year):
 return year
 year += 1
 print(find_next_toe_number(2025))
```

# Noch mehr Übungen ...

### ? Aufgabe

Schlage in deinen Unterlagen drei Aufgabenstellungen aus dem Bereich "Programmieren mit Turtle" nach, die dir damals sehr schwierig erschienen und löse diese nochmals mittels Chatbot. Findest du auf diesem Weg elegantere Lösungen?

## ? Python-Profi

Absolviere den Python-Track auf <u>Exercism.org</u>.

Olympischer Gedanke

Schaffst du die erste Runde bei der SOI?

# Mach dein Ding

Löse vier Wochen lang die aktuelle Wochenaufgabe von <u>The Weekly Challenge</u> in Python.

#### ? Hole in One

Was beim klassischen Golf die Anzahl der Schläge, ist beim Code-Golf die Anzahl der Zeichen. Bist du besser als deine Kollegen?

## Steile Lernkurve

Wie viele Tage schaffst du es, beim Adventofcode durchzuhalten? Bis zum Samichlaustag am 6. Dezember?

#### International Combat

Schaffst du es bei Codeforces.com in ein Turnier?

# **Anhang**

# **Setup deines Computers**

Damit Jupyter-Chatbooks ohne grösseren Support (durch deine Lehrpersonen) laufen, nutzen wir in dieser Lerneinheit NixOS.

#### **USB-Stick**

Nutze einen <u>USB-Stick</u> mit NixOS als Betriebssystem. So kannst du NixOS verwenden, ohne Software auf deinem Computer zu installieren.

#### Installation

Hast du noch einen (alten) Computer, der die aktuellen Windows 11 Anforderungen nicht erfüllt? Dann installiere dort direkt NixOS.

Oder aktiviere unter Windows Windows Subsystem for Linux . Dann hast du auch dort die Möglichkeit, genau wie unter Linux und Mac, NixOS als Paketmanager zu installieren.

### **Virtualbox**

Oder installiere Virtualbox, um NixOS auszuführen. Lade dir das NixOS-Image herunter und starte dieses.

### **Docker**

Oder installiere <u>Docker</u>, um NixOS auszuführen. Um NixOS zu starten, verwendest du im Terminal den Befehl

```
docker run -ti ghcr.io/nixos/nix
```

# **Erfolgs-Check**

Ob du NixOS erfolgreich installiert hast, kannst du im Terminal so überprüfen:

```
nix-shell --version
```

Wenn kein Fehler gemeldet wird, ist alles bereit, um Jupyter-Chatbook zu starten.

### Starten von Jupyter-Chatbook

Öffne ein Terminal und führe folgende Befehle aus:

a.) ggf. Ordner wechseln

```
cd $HOME/Documents
```

b.) die Unterrichtsunterlagen clonen

```
nix-shell -p git
```

```
git clone --depth=1 https://github.com/zero-overhead/programmieren-mit-LLM
cd programmieren-mit-LLM
```

c.) API-Keys angeben (diese Werte natürlich mit validen Schlüsseln ersetzen)

.) Jupyter starten
nix-shellrun "jupyter lab"
.) Falls du Python-Code ausprobieren möchtest, nutze doch thonny :
nix-shellrun "thonny"
Du hast NixOS installiert und der Befehl nix-shellversion funktioniert.
☐ Du kannst API-Keys für OpenAI und MistralAI hinterlegen.
☐ Du kannst Jupyter starten.
Du kannst Thonny starten.

## **Datenschutz**

In <u>Jupyter</u> verwenden wir alle den gleichen API-Key, um auf die Modelle zuzugreifen. Somit sind deine Eingaben pseudomisiert. Wenn du keine Bedenken bezüglich des Schutzes deiner Eingaben und persönlichen Daten hast, kannst du dich auch bei z.B. <u>ChatGPT</u> mit (d)einem Microsoft-, Google- oder Apple-Konto direkt anmelden.

## GPT/LLM/API-Key - Was bedeutet das und wie funktioniert das grundsätzlich?

export OPENAI\_API\_KEY=sk-VE5rCA0sesZppklctgcOT3BlbkFJAVjrnvtqRpM1njjWdZeqvq

export MISTRAL\_API\_KEY=mucG8sAMn8Lb8mRvTPJqsH3ElbDoowvwFw

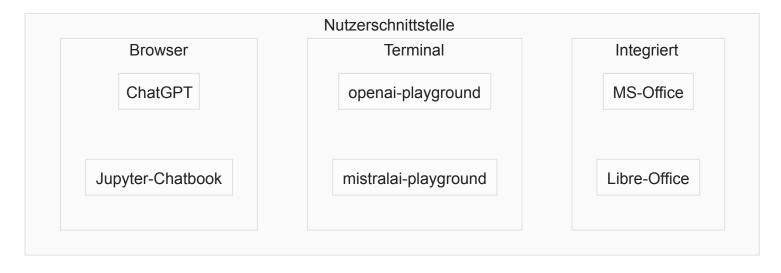
Dieser Abschnitt dient deinem Verständnis der Gesamtzusammenhänge. In dieser Lektion ging es aber nicht darum, wie LLM prinzipiell funktionieren und welche verschiedenen Ansätze es gibt. Wir fokussierten uns darauf, wie dir LLM ganz praktisch beim Programmieren helfen können. Vergleiche es mit der Fahrschule: Um ein Auto zu fahren, musst du nicht wissen, wie der Motor, die Radaufhängung und der Katalysator ganz genau funktionieren, oder?

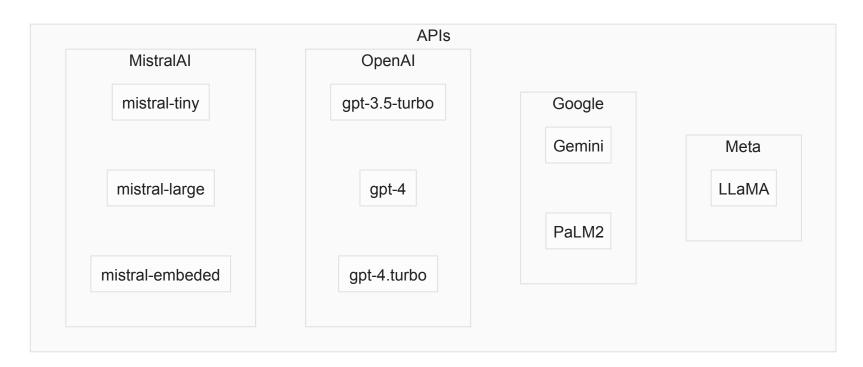
Wenn wir von Al reden, musst du zwischen kommerziellen Anbietern wie OpenAl, MistralAl oder Google und den eigentlichen Rechenmodellen wie etwa gpt-3.5-turbo oder mistral-tiny und den Benutzerschnittstellen unterscheiden.

Die Schnittstelle zwischen Mensch und Computer wird als Benutzerschnittstelle oder User Interface (UI) bezeichnet. Dazu gehören grafische Benutzeroberflächen (GUI), Text-basierte Benutzeroberflächen (CLI), Touchscreens, Tastatur- und Maussteuerungen, sowie Spracherkennungssysteme.

Die verschiedenen Anbieter stellen über ein sogenanntes Application Programming Interface - API - den Zugriff auf ihre Modelle zur Verfügung und verlangen dafür i.d.R. Geld. Damit sich das auf deinem Rechner laufende Jupyter-Chatbook oder bspw. das Programm openai-playground - deine Benutzerschnittstellen - mit der API eines der Anbieter verbinden kann, benötigst du einen Schlüssel, einen sogenannten API-Key. Den bekommst du von deiner Lehrperson. Über diesen Schlüssel rechnet der Anbieter die Zugriffe ab. Wenn der Zugriff auf ein Modell von einem Anbieter kostenlos angeboten wird, frage dich, was oder wer das Produkt ist, welches das wirtschaftliche Überleben des Anbieters sichert.

## Wo ist der Unterschied zwischen ChatGPT und Jupyter-Chatbook?





Der Unterschied zwischen ChatGPT und Jupyter-Chatbook ist, dass du mit ChatGPT nur auf Modelle von OpenAl zugreifen kannst. Bei Jupyter-Chatbooks bist du frei in deiner Wahl und kannst Modelle verschiedener Anbieter sogar gleichzeitig nutzen.

Das eingangs verwendete openai-playground funktioniert prinzipiell wie das bekannte ChatGPT. Beides sind Programme, die Eingaben von Usern an spezialisierte, i.d.R. in der Cloud laufende Modelle (GPT - Generative Pre-trained Transformer) weiterleiten, welche darauf trainiert wurden, aus einer Eingabe eine dazu passende Ausgabe zu berechnen. Für das Training wurden riesige Mengen von im Internet verfügbaren Texten verwendet. Ein GPT-Modell, man spricht allgemeiner auch von Large Language Models - LLM, transformiert eine Eingabe wie etwa "Erzähle einen Witz" in eine Zeichenkette, die im Zusammenhang mit ähnlichen Eingaben in den Trainingsdaten aufgetaucht ist. Ein LLM versteht nicht, was ein Witz ist und warum etwas lustig ist. Auch kann ein LLM implizit nur auf die Daten zurückgreifen, die beim Training zur Verfügung standen. Ausgaben werden als statistisch wahrscheinliche Fortsetzung der Eingabe errechnet, was dazu führt, dass prinzipiell zu jeder Eingabe eine Ausgabe gemacht werden kann, ohne jedoch Aussagen über Wahrheit oder Korrektheit machen zu können. In dem Zusammenhang spricht man dann auch von **Halluzinieren**.

Halluzinieren: Wenn Modelle objektiv falsche Antworten geben, s	pricht man von Halluzinieren.
GPT ist die Abkürzung für "Generative Pre-trained Transformer" und	beschreibt Berechnungsmodelle, die bspw. hinter ChatGPT stehen.
· ·	ben, haben aber kein Verständnis im herkömmlichen Sinn über den Inhalt.
<ul> <li>API bedeutet Application Programming Interface und regelt den Zug</li> <li>Anbieter wie OpenAl oder Google geben über ein API i.d.R. kostenp</li> </ul>	·

<sup>1.</sup> OK, hier wurde ein bisschen geschummelt. Die sprechende Kuh wurde durch das Programm cowsay erzeugt. <u>Hier</u> ist der ganze Code. *←*