



# artificial intelligence

## PROGRAMMIEREN MIT UNTERSTÜTZUNG VON KI

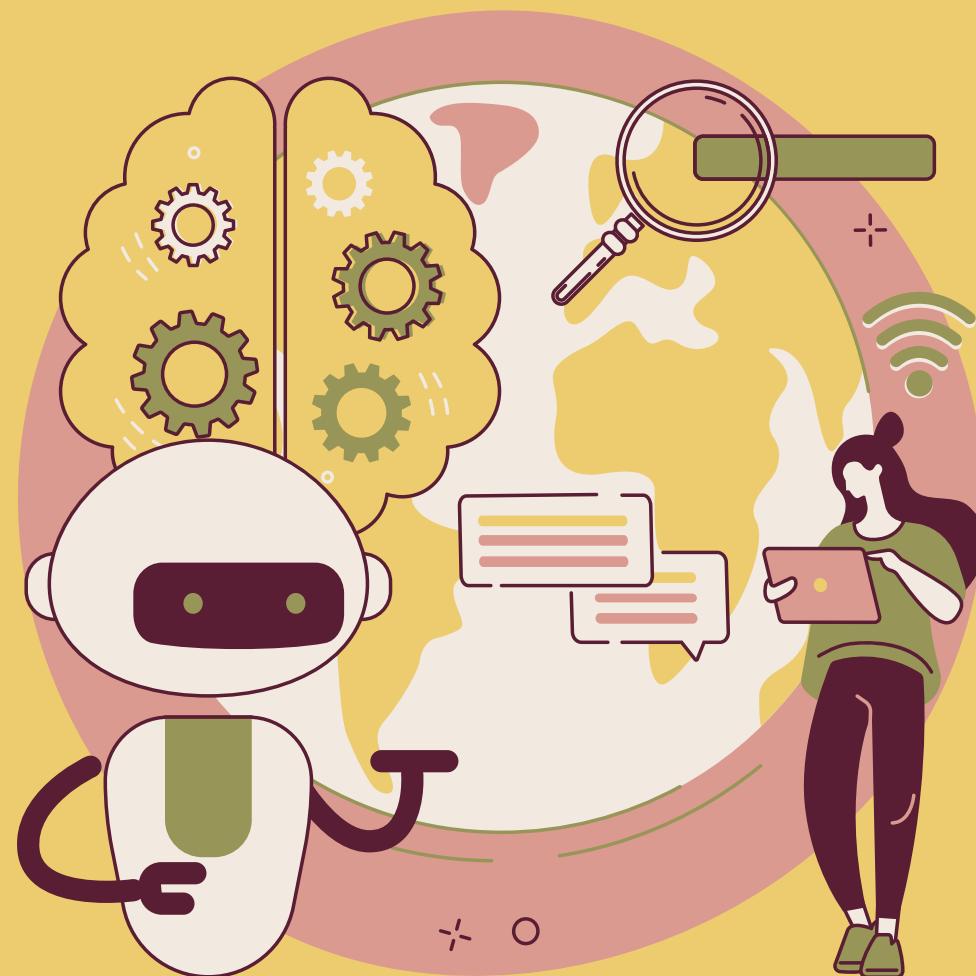
# Disclaimer

- Ich bin kein Softwareentwickler
- Ich bin kein KI Experte
- Ich programmiere lediglich als Hobby
- Der Vortrag beleuchtet ausschliesslich den Aspekt der Softwareentwicklung und basiert auf meinen Erfahrungen als Anregung für eigene Arbeiten

Anmerkung:

Naturgemäss lassen sich KI Ergebnisse nicht 1:1 reproduzieren, zudem 'weiss' die KI was innerhalb des Accounts schon generiert wurde. Aus diesen Gründen wurden die Demos im voraus aufgezeichnet, sind aber zu 100% authentisch.

# Inhalt

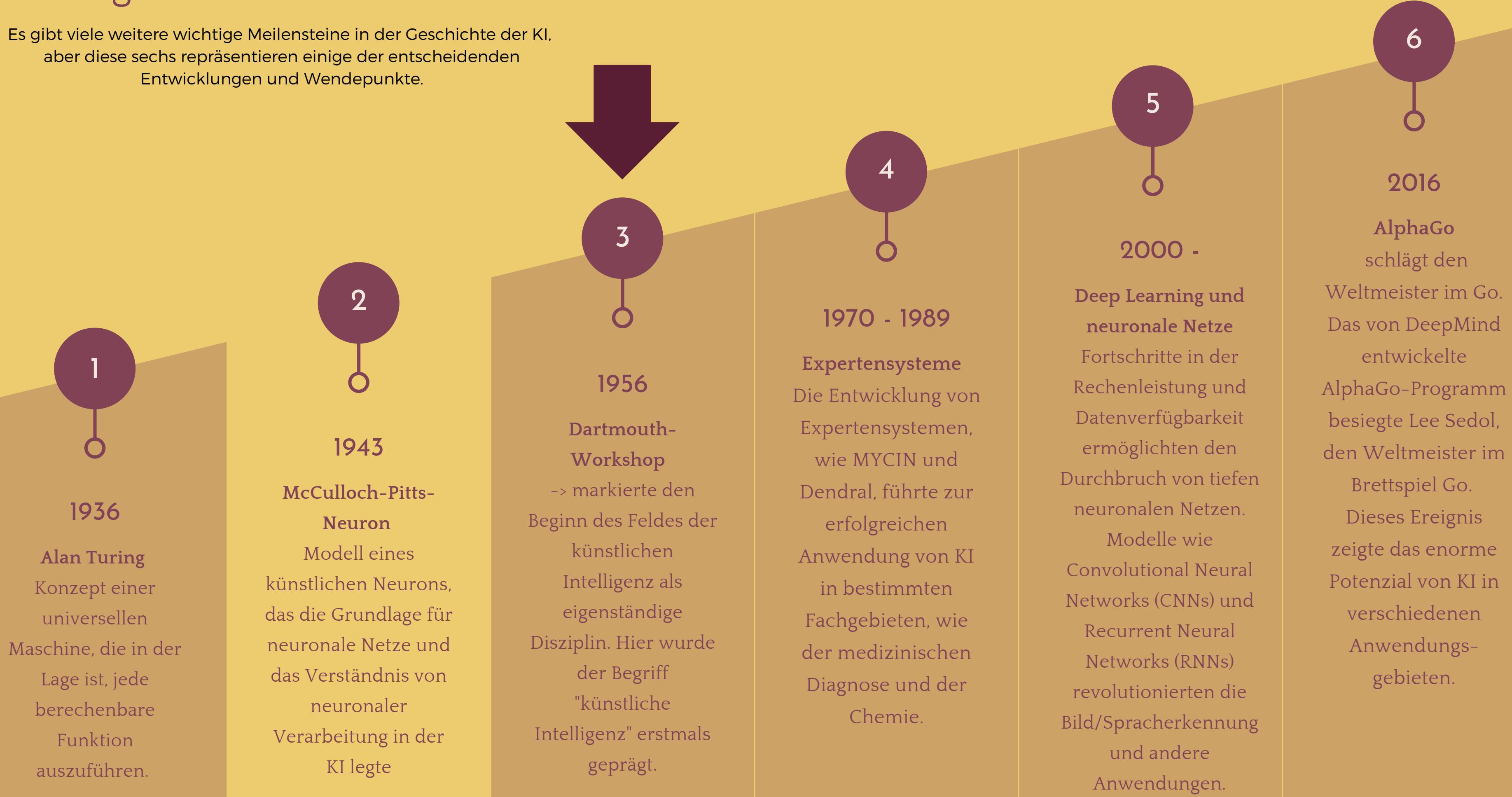


- 1 Einführung
- 2 Wir LASSEN ein Spiel programmieren
- 3 Programmieren mit KI Assistenz
- 4 Code improvement
- 5 Fazit

Oder wieso du Marokko meiden solltest und wie Du eine Million \$ verdienen kannst.

# Einige Meilensteine der KI

Es gibt viele weitere wichtige Meilensteine in der Geschichte der KI, aber diese sechs repräsentieren einige der entscheidenden Entwicklungen und Wendepunkte.



# Hype Cycle for Artificial Intelligence, 2023



gartner.com

Source: Gartner  
© 2023 Gartner, Inc. and/or its affiliates. All rights reserved. 2079794

Gartner®



# Datenhalluzination

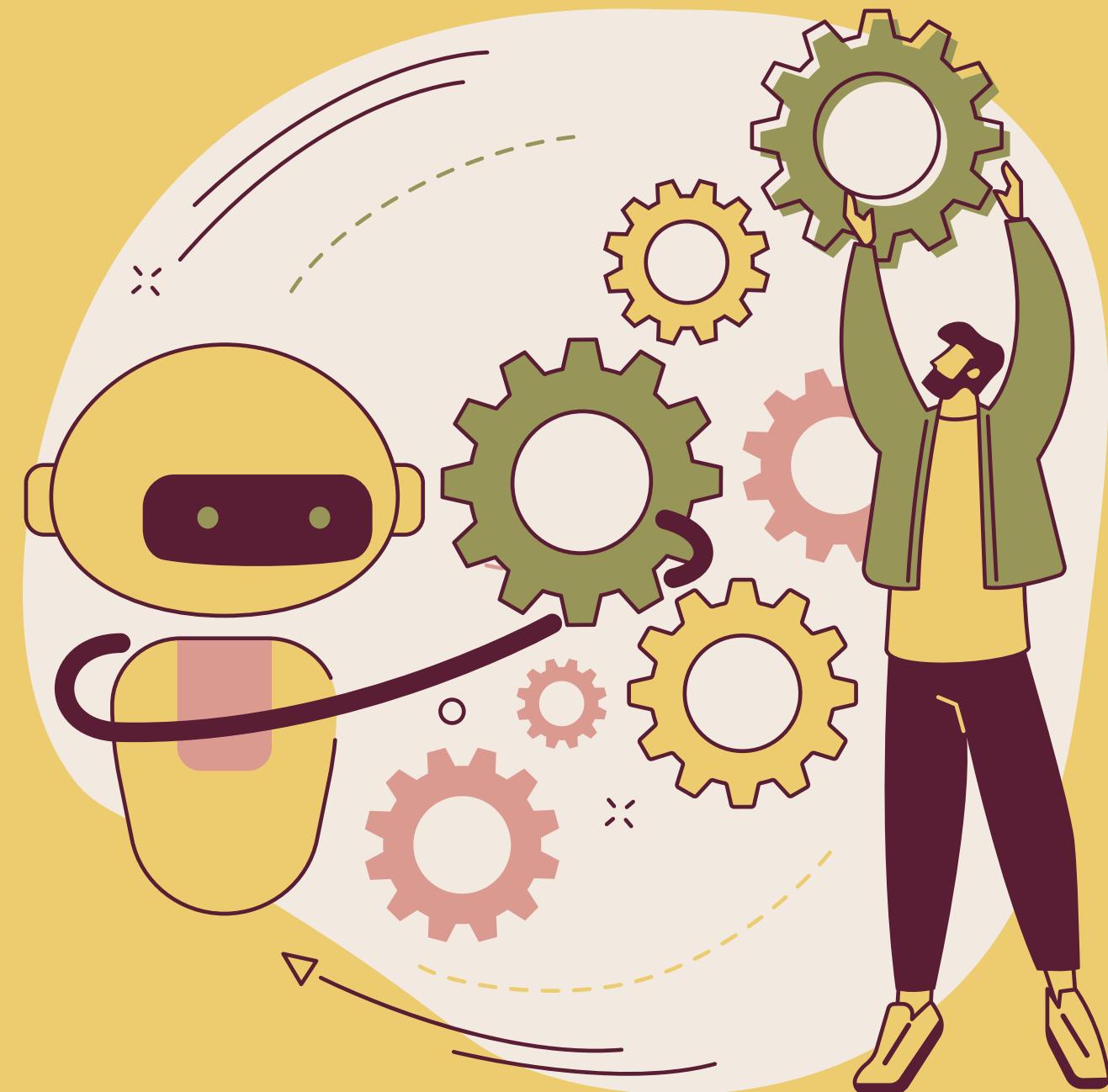
Datenhalluzination bezieht sich auf den Prozess, bei dem ein maschinelles Lernsystem auf der Grundlage von Daten, die es zuvor gesehen hat, etwas erzeugt, das es als wahrscheinlich empfindet, obwohl es möglicherweise tatsächlich nicht existiert.

Dies kann beispielsweise dazu führen, dass ein neuronales Netzwerk, das für die Erkennung von Objekten trainiert wurde, ein Objekt generiert, das es nie zuvor gesehen hat, aber das es trotzdem als real empfindet.

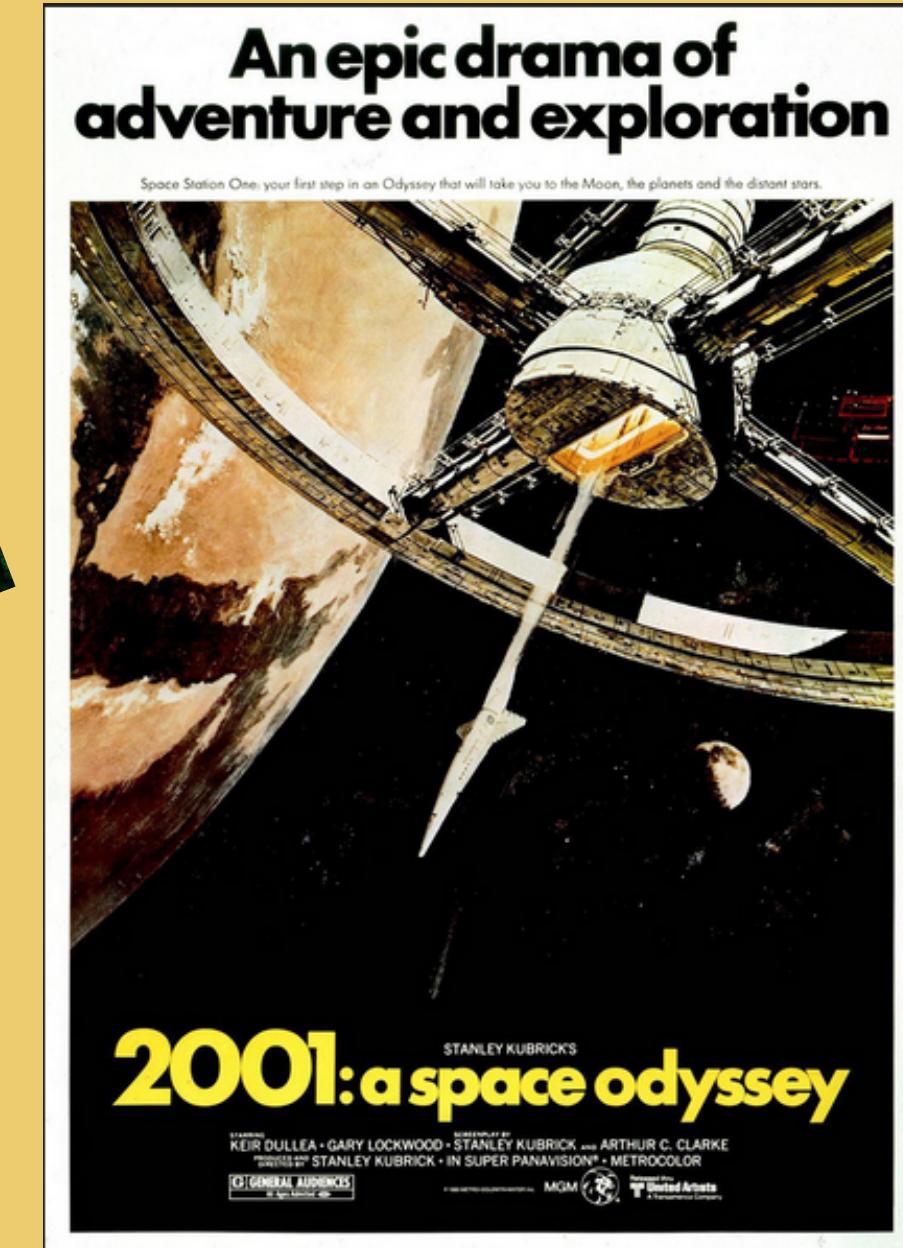
Einer KI beizubringen, dass sie "nicht in Halluzinationen verfällt" ist "leider" nicht so einfach. Künstliche Intelligenz funktioniert nun einmal komplett anders als ein menschliches Gehirn, gerade was Konzeptdenken anbelangt, und das wird sicher noch lange so bleiben. Generative AI wie chatGPT wird deshalb sowohl über- wie auch unterschätzt...

# Prompts bestimmen das Ergebnis!

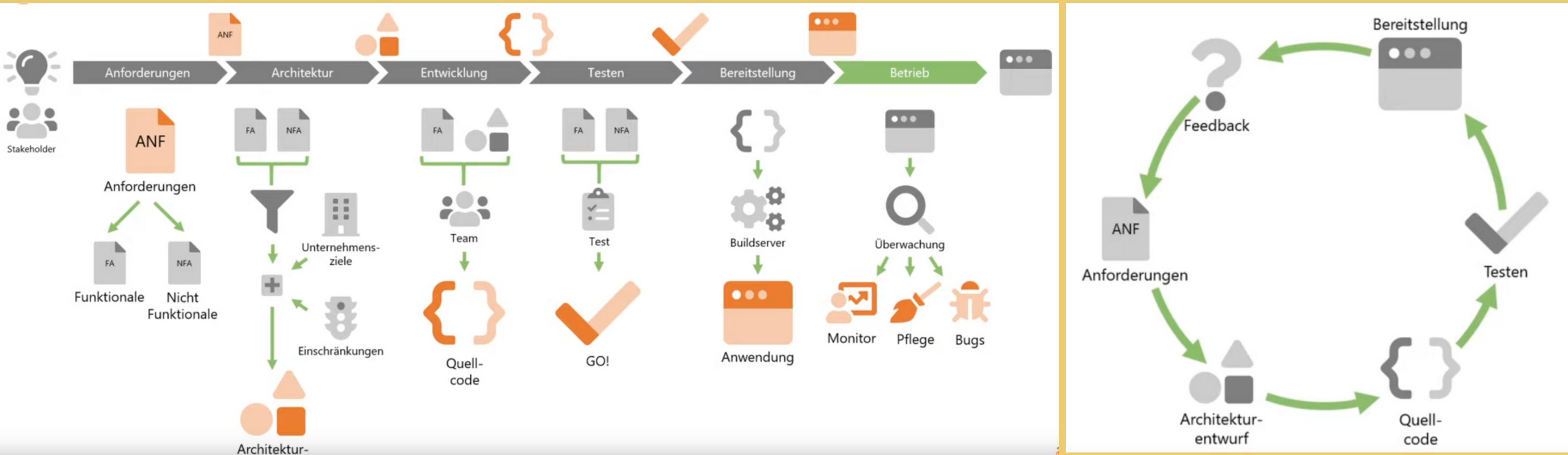
- Anweisungen - Inhalt
- Spezifische Prompts
- Formatvorgaben
- Zero Shot - Few Shot
- Keine Füllwörter
- Kurz fassen
- Positive Prompts
- Anfang vorgeben



# Filme (Auswahl)



# Softwareentwicklung (Aufbau / Ablauf) und mögliche Ansätze durch KI



- Zusammenfassung
- Konsolidierung
- Konsistenzprüfung
- Ergänzung



- Code generieren
- Code Improve
- Fehlersuche
- Dokumentation



- Testmodule generieren
- Dokumentieren
- Fehlersuche
- Anforderungsnachweis



- Expertensystem
- Mustererkennung
- Monitoring
- Reports

Aus meiner Sicht wird eine durchgängige KI Lösung noch eine Weile dauern!?

-> Die Arbeitsplätze dürften noch längere Zeit gesichert sein (Produktivität vs Komplexität)



Den Code zu den Demos findet ihr hier:  
[https://github.com/Wale56/Makertalk\\_231013](https://github.com/Wale56/Makertalk_231013)

# Aufgabe 1: Turtle Crossing

Schreibe ein Skript für ein einfaches Spiel 'Turtle Crossing' in Python



## Optionen:

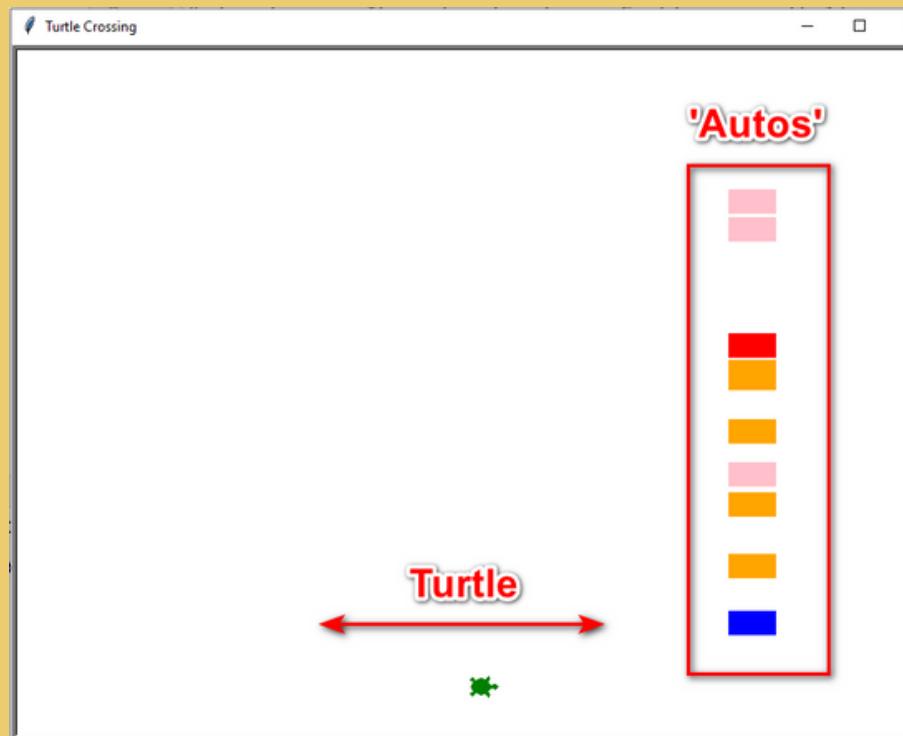
- Rundenzähler
- 'Game Over'

- 1 Der Spieler steuert die Schildkröte
- 2 Schildkröte kann nur vorwärts gehen
- 3 Die Autos fahren horizontal
- 4 Game Over bei Streifung
- 5 Ausgangsposition nach erfolgreichem überqueren und Autos beschleunigen

# Darstellung der wichtigsten Spielstände

Quellcode: [https://github.com/Wale56/Makertalk\\_231013](https://github.com/Wale56/Makertalk_231013)

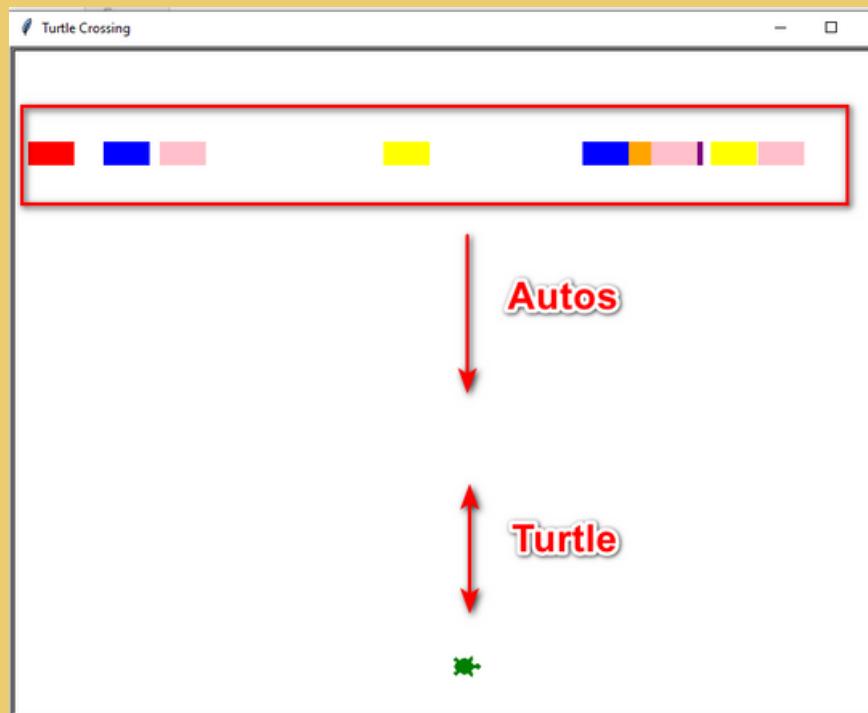
turtle\_0\_start.py



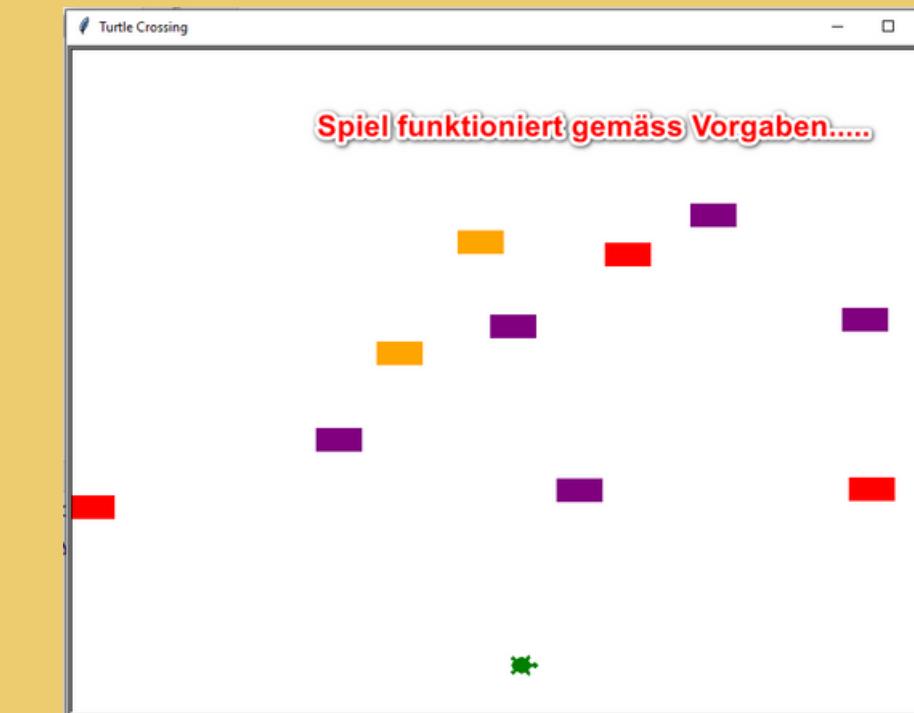
turtle\_2.py



turtle\_1.py



turtle\_6.py (Endstand)



Das fertige Spiel inkl. Optionen

Quellcode: [https://github.com/Wale56/Makertalk\\_231013](https://github.com/Wale56/Makertalk_231013)

turtle\_8\_Game\_over.py



# 2a: Erkläre dieses Programm

```
import ...

# Evaluate data df in the time spread n (months)
1 usage
def get_ret(df, n):
    previous_prices = df[:df.index[-1] -
        DateOffset(months=n)].tail(1).squeeze()
    recent_prices = df.loc[df.index[-1]]
    ret_df = recent_prices / previous_prices - 1
    return previous_prices.name, ret_df

# Read the tickerlist
tickers = pd.read_html('https://en.wikipedia.org/wiki/'
    'List_of_S%26P_500_companies')[0].Symbol
tickers = tickers.to_list()

# Read the data only once!
1 usage
@st.cache_data
def getdata():
    df = yf.download(tickers, start='2020-01-01')
    df = df['Close']
    return df
```

```
df = getdata()

st.title('Index component performance of the S&P500')
# st.title('How cool ist that!?')

n = st.number_input('Please provide the performance horizon in months:',
                    min_value=1, max_value=24)

date, ret_df = get_ret(df, n)
winners, losers = ret_df.nlargest(10), ret_df.nsmallest(10)
winners.name, losers.name = 'winners', 'losers'

st.table(winners)
st.table(losers)

winnerpick = st.selectbox('Pick a winner to visualize:', winners.index)
st.line_chart(df[winnerpick][date:])
loserpick = st.selectbox('Pick a loser to visualize:', losers.index)
st.line_chart(df[loserpick][date:])
```

# 2a: Lösung

## ChatGPT

Dieses Programm lädt den Aktienverlauf von allen S&P500-Komponenten seit dem 1. Januar 2020 und zeigt dann eine interaktive Analyse der besten und schlechtesten Aktien in einem bestimmten Zeitrahmen.

Das Programm beginnt mit dem Laden der Liste aller S&P500-Komponenten von Wikipedia. Die yfinance-Bibliothek wird verwendet, um den Aktienverlauf aller Komponenten herunterzuladen, und die Daten werden dann in einer Pandas DataFrame gespeichert.

Dann wird der Benutzer aufgefordert, einen Zeitrahmen von 1 bis 24 Monaten auszuwählen. Das Programm berechnet dann die Gesamtrendite jeder Aktie im ausgewählten Zeitraum und zeigt die Top 10 Gewinner und Verlierer in einer Tabelle an.

Der Benutzer kann eine der Gewinner- oder Verlierer-Aktien auswählen, um den Aktienverlauf in einem interaktiven Liniendiagramm zu visualisieren.

Der st.cache\_data-Dekorator wird verwendet, um die Daten beim erneuten Laden der Seite nicht erneut zu laden, um die App schneller zu machen.

Das Ergebnis ist eine interaktive App, die es dem Benutzer ermöglicht, den Aktienverlauf der S&P500-Komponenten in einem bestimmten Zeitraum zu analysieren und die Top-Performer und Underperformer zu identifizieren.

# 2b: Erkläre diese Codezeile

```
processed_file = docx2txt.process(uploaded_file)
```

## CodeGPT

Die ausgewählte Zeile verwendet die Funktion `process` aus dem Modul `docx2txt`, um eine hochgeladene Datei zu verarbeiten.

Hier ist eine Erklärung, was diese Zeile bewirkt:

`uploaded_file`: Dies ist eine Variable, die die hochgeladene Datei repräsentiert. Sie wird als Argument für die Funktion `process` übergeben.

`docx2txt`: Dies ist ein Modul, das Funktionen zur Verarbeitung von DOCX-Dateien in Text bereitstellt.

`process`: Dies ist eine Funktion aus dem Modul `docx2txt`, die dazu dient, eine DOCX-Datei zu öffnen und den Inhalt als Text zurückzugeben.

`processed_file`: Dies ist eine Variable, die den zurückgegebenen Text aus der Funktion `process` speichert. Der Text repräsentiert den Inhalt der verarbeiteten Datei.

**Zusammenfassend liest der Code eine hochgeladene DOCX-Datei mit Hilfe der Funktion `process` aus dem Modul `docx2txt` ein und speichert den Textinhalt der Datei in der Variablen `processed_file`.**

# 2c: Erkläre diese Codezeile

```
fb['Direction'] = [1 if fb['PriceDiff'].loc[ei] > 0 else 0 for ei in fb.index]
```

## ChatGPT

Hier ist, was dieser Ausdruck im Detail tut:

- `[1 if fb['PriceDiff'].loc[ei]>0 else 0 for ei in fb.index]` ist eine List Comprehension, die über alle Indizes (ei) im Index von `fb` (wahrscheinlich die Zeilenindizes) iteriert.
- Für jeden Index `ei` wird überprüft, ob der Wert in der 'PriceDiff'-Spalte (in der Zeile mit diesem Index) größer als 0 ist (`fb['PriceDiff'].loc[ei] > 0`). Wenn dies wahr ist, wird 1 in die Liste aufgenommen, andernfalls wird 0 hinzugefügt.
- Am Ende ergibt sich eine Liste von 1en und 0en, die der Länge des DataFrame `fb` entspricht. Diese Liste wird dann der 'Direction'-Spalte des DataFrames `fb` zugewiesen.

Insgesamt dient diese Zeile also dazu, basierend auf den Werten in der 'PriceDiff'-Spalte des DataFrames `fb`, eine neue 'Direction'-Spalte zu erstellen, die 1 enthält, wenn der Wert in 'PriceDiff' größer als 0 ist, und 0, wenn er kleiner oder gleich 0 ist. Dies kann beispielsweise in der Finanzanalyse verwendet werden, um die Richtung von Preisänderungen zu kennzeichnen (steigend oder fallend).

CodeGPT liefert ein analoges, aber konkreteres Ergebnis, weil der Kontext innerhalb PyCharm bekannt ist.

# 3: Transfer MATHLAB-> Python

<https://github.com/gklrajan/shenanigans/blob/main/systematicBealTest.m>

%% einige Hintergrundinformationen

% Die Beal-Vermutung ist eine mathematische "Hypothese", die der Geschäftsmann

% Andrew Beal im Jahr 1993 vorschlug. Sie besagt, dass für drei beliebige positive ganze Zahlen A, B,

% und C, und beliebige positive ganze Zahlen x, y und z (größer als 2), wenn  $A^x +$

%  $B^y = C^z$ , dann müssen A, B und C einen gemeinsamen Faktor größer als 1 haben. Die

% Vermutung bleibt unbewiesen, und es wurde kein Gegenbeispiel gefunden, was sie zu einem der

% ungelösten Probleme der Zahlentheorie macht.

% Andrew Beal hat einen Preis in Höhe von 1 Million Dollar für denjenigen ausgesetzt, der einen

% gültigen Beweis liefern oder die Vermutung widerlegen kann.

## Aufgabenstellung

- Transferiere das Programm nach Python
- Erkläre das Programm
- Begründe, wieso es möglicherweise keine Lösung gibt

## Lösung

-> Demo

-> siehe auch chatGPT July (1. Eintrag)

# 4: Programmentwicklung mit KI Unterstützung



From Stable Diffusion: "A whimsical auto accident, where the cars are driven by robots, in the style of Dali"

Man hört viel über automatisierte Fahrzeuge - welche Probleme sie haben, und das Versprechen, dass sie sicherer sein werden als von Menschen gesteuerte Autos. Dazu machen wir einige Untersuchungen:

- Welche Länder haben im Zeitraum 2017–2022 im Durchschnitt die höchste Zahl an Verkehrstoten, gemessen pro 1 Mio. Fahrzeuge, welche am wenigsten ?
- In welchen fünf Ländern gab es im Zeitraum 2017–2022 durchschnittlich die meisten Verkehrstoten, gemessen pro 1 Mio. Einwohner und wo am wenigsten?
- Stelle ausgewählte Länder in einem Liniendiagramm dar.

-> Demo

# 5: Code verbessern (Improvement)



Simple Employee management system

-> Demo 5\_Codeimprovement.

```
# Findings ChatGPT
# Improvements made:
#
# Removed unnecessary try-except blocks.
# Improved code readability by simplifying and reorganizing the code.
# Used list comprehensions for finding employees by role.
# Renamed some methods and variables for better clarity.
# Added type hints for function arguments and return values.
# Modified the way holidays are paid out to be more informative.
# Added the calculate_payment method for HourlyEmployee to calculate payment based on hourly rate and
hours worked.
#
# This code should be more maintainable and easier to understand.
```

# Mein Fazit (1):

## Vorteile:

- Personalisierte Lösungsfindung
- Abnahme von Routinearbeiten
- Neue Ideen

## Nachteile:

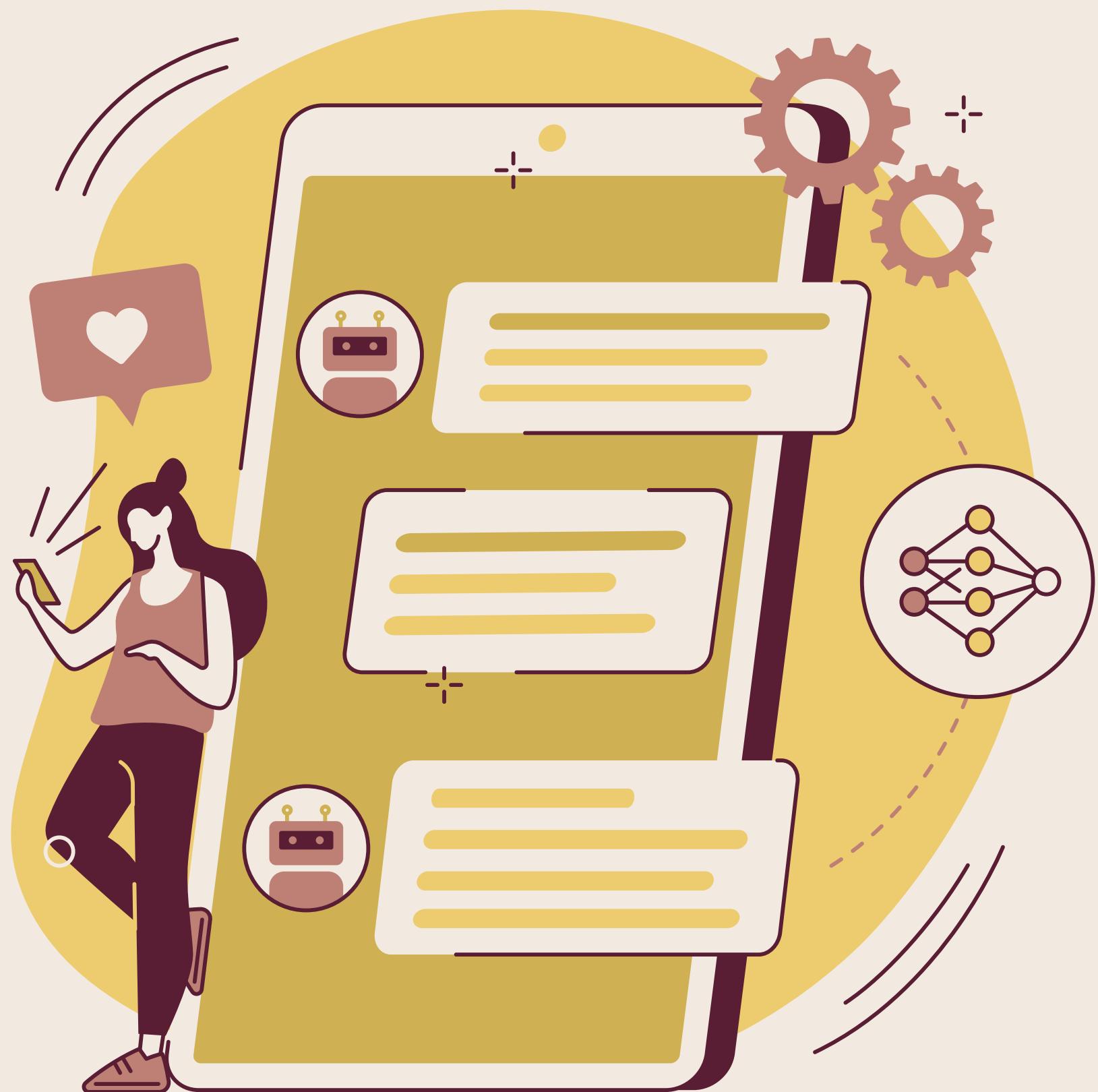
- Mangelhafte Reproduzierbarkeit
- Plötzliche Verschlechterung der Ergebnisse!?
- Privacy (z.B. Firmengeheimnisse)

## Mein Fazit (2):

**KI ist vergleichbar mit den Assistenzsystemen im Auto:**

- Hilfreich, aber nicht perfekt
- Kenntnis der Materie ist zwingend!
- Ergebnisse müssen geprüft werden

- Die häufig propagierte Produktivitätssteigerung um den Faktor zwei ist nach meiner Erfahrung realistisch.
- Der Produktivitätsgewinn wird jedoch infolge der zunehmenden Komplexität und steigenden Anforderungen zum grossen Teil wieder egalisiert.



thank you  
for listening!

Walter Gantenbein  
[gantenbeinw@bluewin.ch](mailto:gantenbeinw@bluewin.ch)