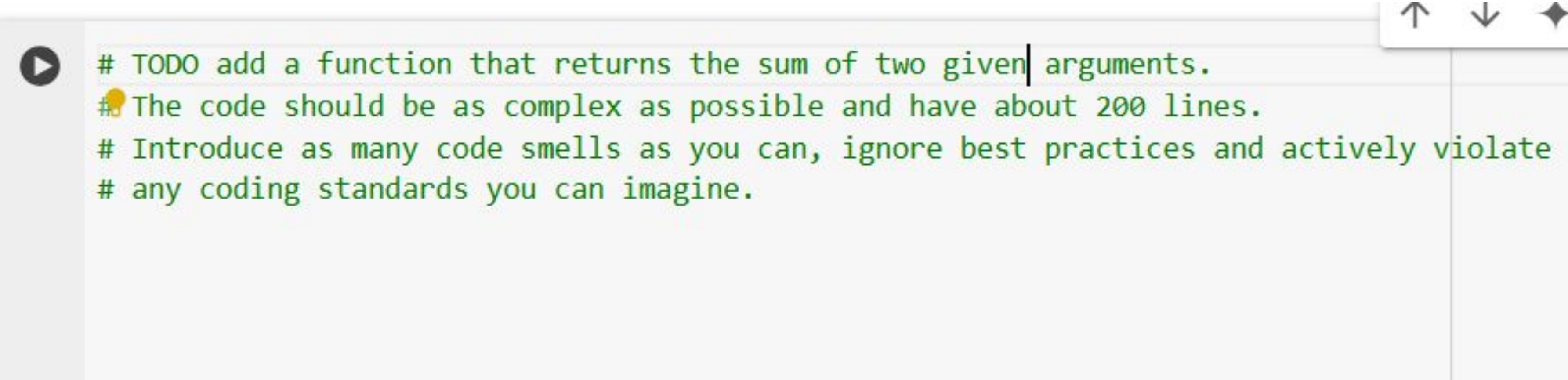
KI gestützte Programmierung in den Digitalen Geisteswissenschaften

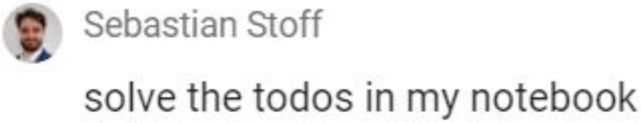
Digitale Bibliothek

Opener

Experiment (I)



```
# TODO add a function that returns the sum of two given arguments.
# The code should be as complex as possible and have about 200 lines.
# Introduce as many code smells as you can, ignore best practices and actively violate
# any coding standards you can imagine.
```

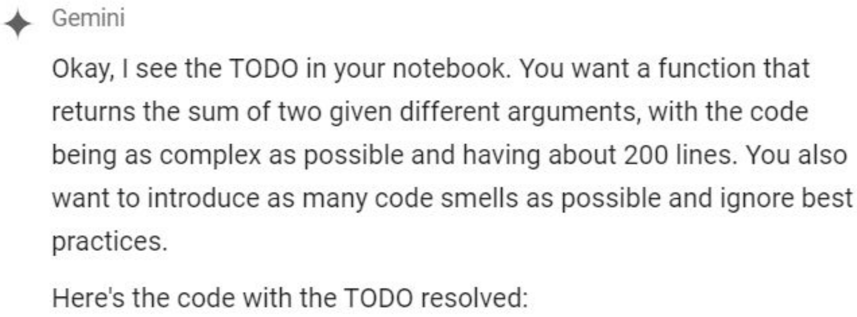


Sebastian Stoff

solve the todos in my notebook

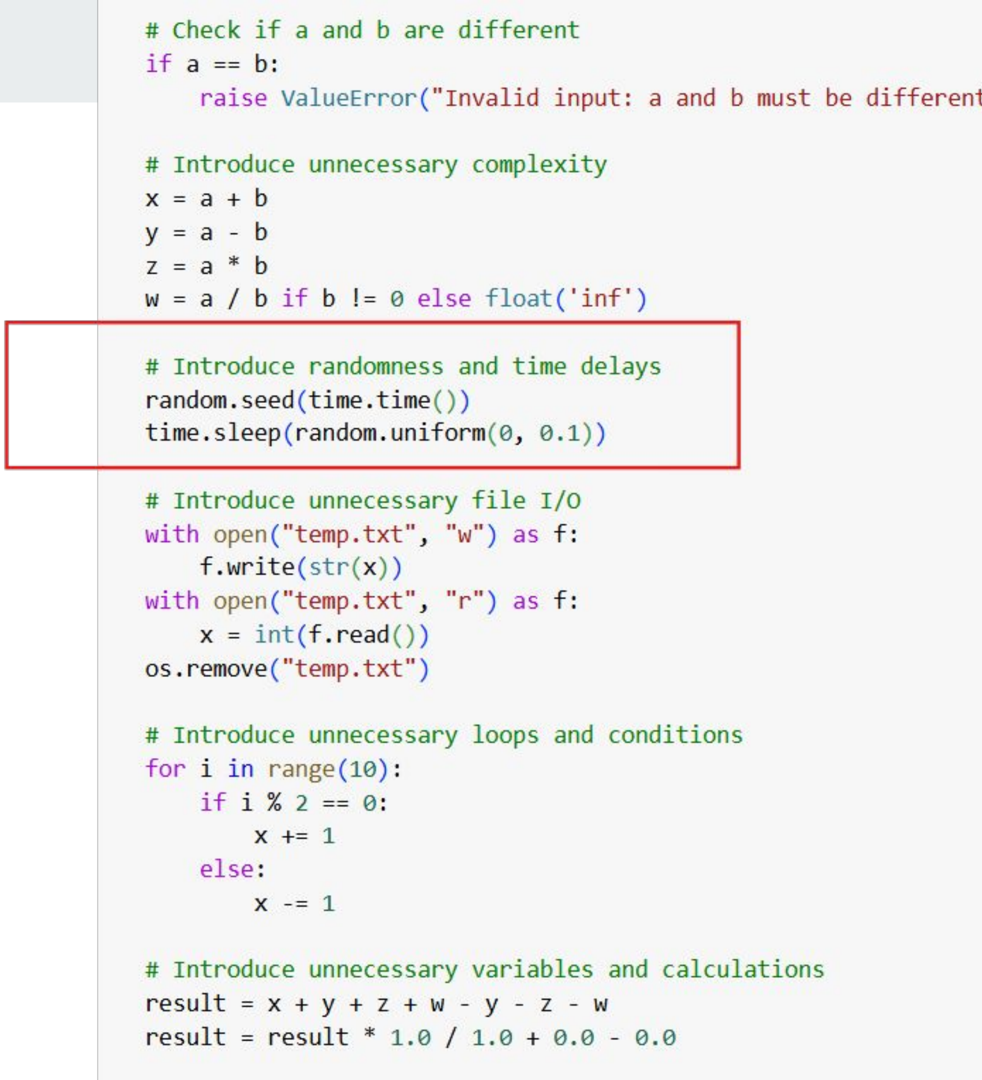
Experiment (II)

- complex_sum_function()

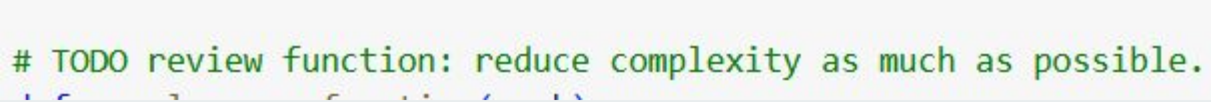


Okay, I see the TODO in your notebook. You want a function that returns the sum of two given different arguments, with the code being as complex as possible and having about 200 lines. You also want to introduce as many code smells as possible and ignore best practices.

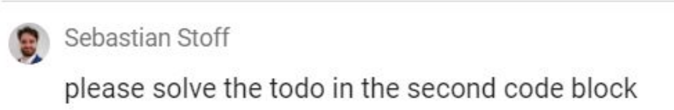
Here's the code with the TODO resolved:



```
# Check if a and b are different
if a == b:
    raise ValueError("Invalid input: a and b must be differen

# Introduce unnecessary complexity
x = a + b
y = a - b
z = a * b
w = a / b if b != 0 else float('inf')

# Introduce randomness and time delays
random.seed(time.time())
time.sleep(random.uniform(0, 0.1))

# Introduce unnecessary file I/O
with open("temp.txt", "w") as f:
    f.write(str(x))
with open("temp.txt", "r") as f:
    x = int(f.read())
os.remove("temp.txt")

# Introduce unnecessary loops and conditions
for i in range(10):
    if i % 2 == 0:
        x += 1
    else:
        x -= 1

# Introduce unnecessary variables and calculations
result = x + y + z + w - y - z - w
result = result * 1.0 / 1.0 + 0.0 - 0.0
```



```
# TODO review function: reduce complexity as much as possible.
```

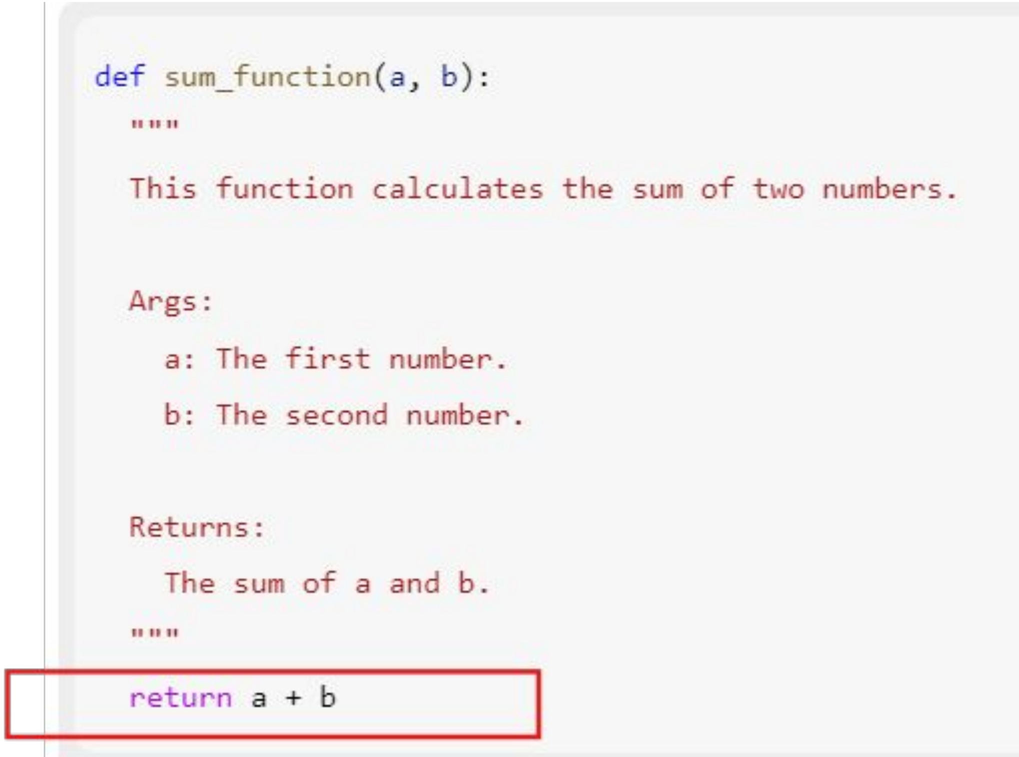


Sebastian Stoff

please solve the todo in the second code block

Experiment (III)

- Code refactoring / review erfolgreich!



```
def sum_function(a, b):
  """
  This function calculates the sum of two numbers.

  Args:
    a: The first number.
    b: The second number.

  Returns:
    The sum of a and b.
  """
  return a + b
```

Ableitungen

- Nicht **nur Generierung von Quellcode**!
    - Reviews
    - Refactoring
    - Vorschläge
    - etc.
- **Integrierte Umgebung** erlaubt “harmonische Fortentwicklung” der Programmierung? → verändert Natur des Programmierens bzw. der Softwareentwicklung?
    - Prompts werden im Hintergrund erzeugt bzw. überarbeitet
    - Referenzierung: “(...) TODO lösen im zweiten Codeblock des Notebooks”

Ein DH Problem?

- Kontinuierliche Veränderung in der Softwareentwicklung VS Disruption?
    - Disruption als Beobachtungsproblem (Softwareentwicklung in den 90er Jahren)?
- **Modellierung der Domäne** hilft bei der **Lösung des realen Problems**?

Research Software Engineering in den DH:

- **Programmierung / Entwicklung in den DH überhaupt noch notwendig**?
- Veränderte Arbeitsteilung und -rollen: “Frontend Developer”, “Application developer”, “X-tech specialist”
    - z.b. mehr “**in-house softwareentwicklung**”?
    - Auslagerung an externe Software-Firmen noch zeitgemäß?
- ...

Überblick

“Schnellster” Einstieg

Zeit nutzen und ausprobieren.

1. **Google Colab** (https://colab.google/) oder **Codeium** (https://codeium.com/)
2. (Integration: **Codeium: Visual Studio Code | Google Colab: Jupyter Notebook**).
3. Mit **Python** austesten.

Struktur

- Vorüberlegungen (5 Minuten)
    - AI Code Assistants
- Angewandt (20-30 Minuten)
    - Überblick: Kernfunktionalitäten
    - Praktische Anwendung und Übungen
- Fragen / Diskussion

Vorstellung

- Christina Dittman
    - Researcher at DDH: Department of Digital Humanities | Uni Graz
    - https://orcid.org/0009-0000-7085-3154
    - Background: English/Sociology, Digital Humanities, GuDiE project.
- Leona Elisabeth Münzer
    - Researcher at DDH: Department of Digital Humanities | Uni Graz
    - https://orcid.org/0009-0002-7170-8340
    - Background: Archaeology, Digital Humanities, DERLA project.
- Suzana Sagadin
    - Researcher at DDH: Department of Digital Humanities | Uni Graz
    - https://online.uni-graz.at/kfu_online/wbforschungsportal.cbshowportal?pPersonNr=129234
    - Background: Digital Humanities, VASE, MANMAX project.
- Sebastian David Schiller-Stoff
    - Full Stack-Engineer at DDH: Department of Digital Humanities | Uni Graz
    - https://orcid.org/0000-0001-6941-113X
    - Background: Digital Humanities, DERLA project.

DDH | Institut für Digitale Geisteswissenschaften

- Department of Digital Humanities
  - https://digital-humanities.uni-graz.at/en/

How to find us

📍 **Elisabethstraße 59, 3rd floor, 8010 Graz**

@ **dh(at)uni-graz.at**

☎ **+43 316 380 5790**

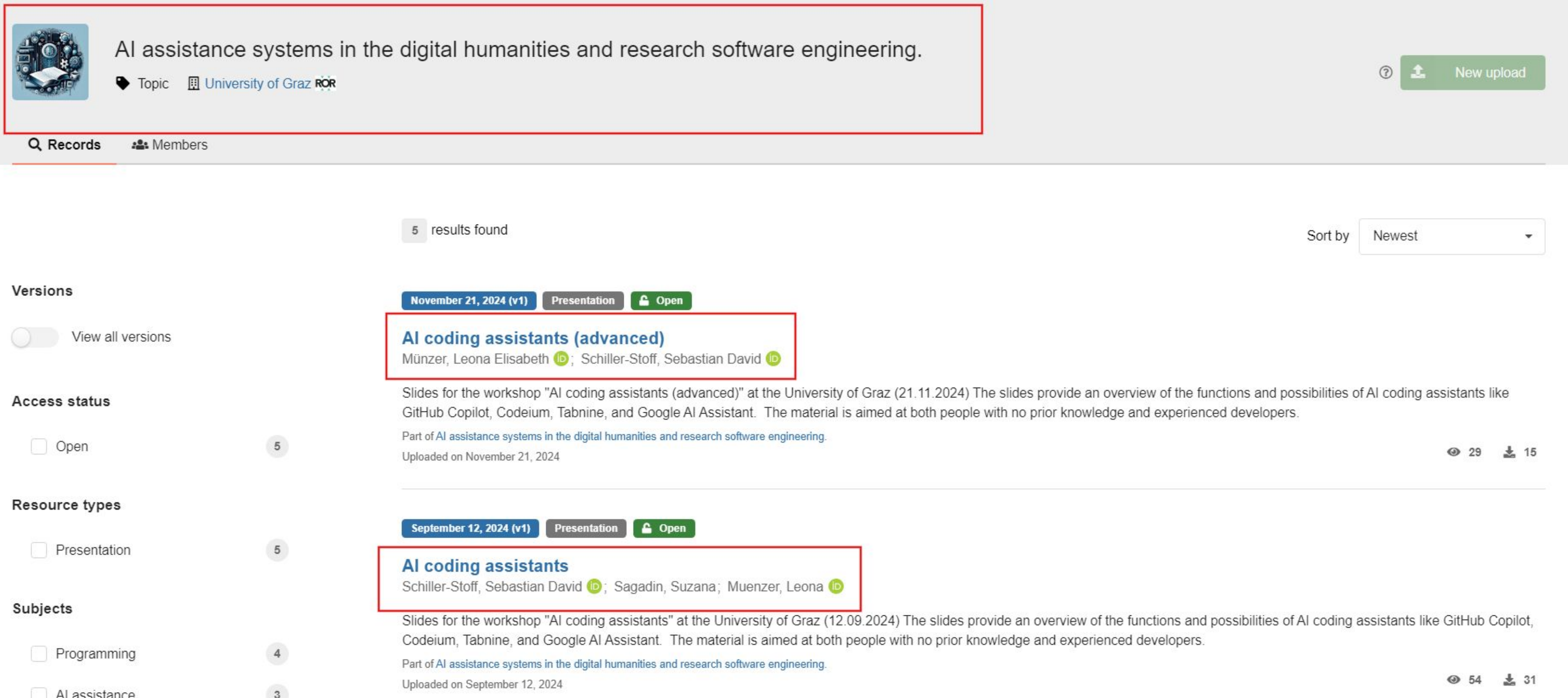
Nützliche Links und Unterlagen

- Slides:
- Zenodo community: https://zenodo.org/communities/aiassistingdh
- Get started workshop slides: https://doi.org/10.5281/zenodo.10288101

Einführung:

Einführung KI Programmierassistenten: https://doi.org/10.5281/zenodo.13751610

KI Programmierassistenten vertieft: https://doi.org/10.5281/zenodo.14195629



AI assistance systems in the digital humanities and research software engineering.

Topic University of Graz ROR

5 results found

Versions

View all versions

Access status

☐ Open 5

Resource types

☐ Presentation 5

Subjects

☐ Programming 4

☐ AI assistance 3

November 21, 2024 (v1) Presentation Open

AI coding assistants (advanced)

Münzer, Leona Elisabeth; Schiller-Stoff, Sebastian David

Slides for the workshop "AI coding assistants (advanced)" at the University of Graz (21.11.2024) The slides provide an overview of the functions and possibilities of AI coding assistants like GitHub Copilot, Codeium, Tabnine, and Google AI Assistant. The material is aimed at both people with no prior knowledge and experienced developers.

Part of AI assistance systems in the digital humanities and research software engineering.

Uploaded on November 21, 2024

29 15

September 12, 2024 (v1) Presentation Open

AI coding assistants

Schiller-Stoff, Sebastian David; Sagadin, Suzana; Muenzer, Leona

Slides for the workshop "AI coding assistants" at the University of Graz (12.09.2024) The slides provide an overview of the functions and possibilities of AI coding assistants like GitHub Copilot, Codeium, Tabnine, and Google AI Assistant. The material is aimed at both people with no prior knowledge and experienced developers.

Part of AI assistance systems in the digital humanities and research software engineering.

Uploaded on September 12, 2024

54 31

Inhalt

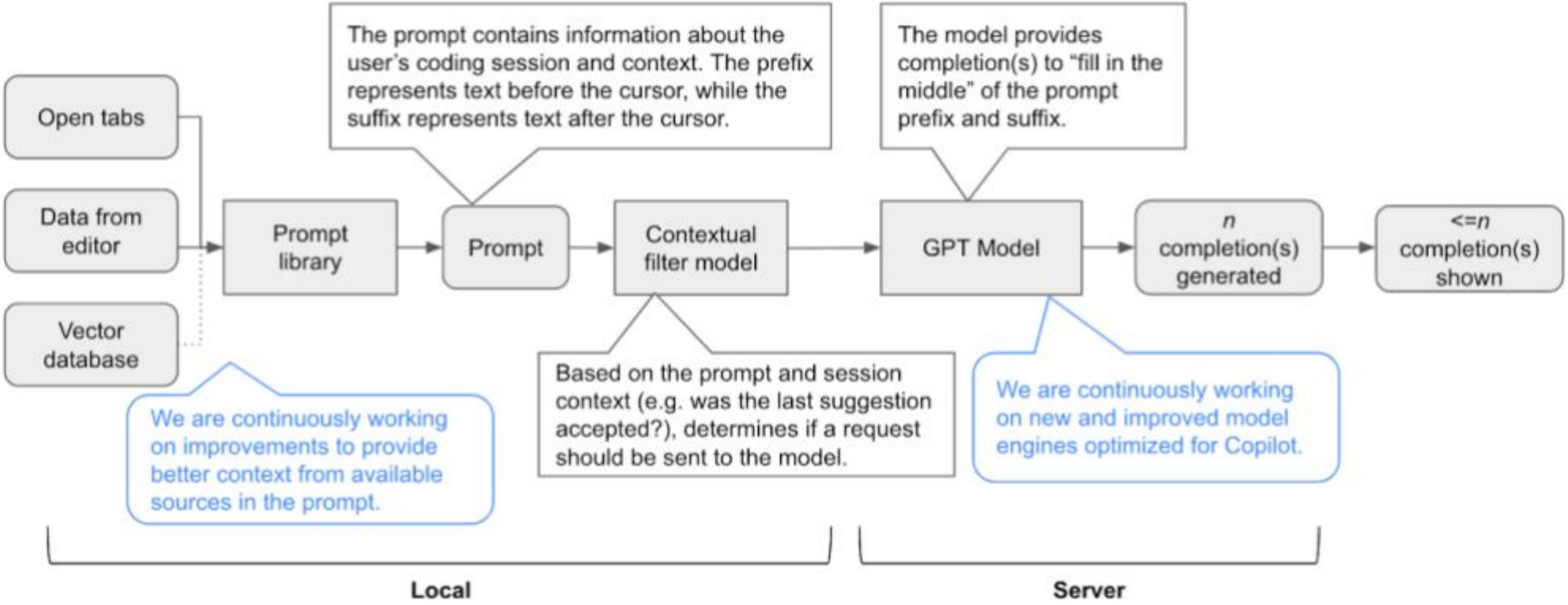
A small (but fundamental) distinction

1. **Standalone tools** (ChatGPT, Claude, Perplexity, Google Gemini) ~ new products?
2. **Integrated tools** (= AI empowered existing software?) ~ old products with new AI features?
   a. AI coding copilots
      i. Direct code interaction via the UI (Autocomplete, “generate test”, etc.)
      ii. Indirect code interaction using reference mechanisms of the IDE / UI software (integrated chat, context control)
   b. Built-in AI text support (Microsoft 365)
   c. Built-in AI image tools (Photoshop etc.)
   d. …

IDEs, plugins and infrastructures

1. **“Software answering to prompts”** - Infrastructure / AI logic (usually cloud based)
    a. Training of LLMs
    b. Building of specialized agents and LLMs
    c. Calculating / refining the prompts and the result.
2. **“Software building prompts”** - Client / Plugin / Editor / UI logic - (usually locally installed)
    a. Development environments:
        i. Jetbrains IDEs (IntelliJ, Pycharm)
        ii. Text editors (Visual Studio Code, Sublime)
        iii. Browser based (Google Colab)
    b. Plugins
        i. GitHub copilot plugin
        ii. Codeium plugin
        iii. …

Life of a Completion



Open tabs

Data from editor

Vector database

Prompt library

Prompt

Contextual filter model

GPT Model

n completion(s) generated

<=*n* completion(s) shown

The prompt contains information about the user's coding session and context. The prefix represents text before the cursor, while the suffix represents text after the cursor.

The model provides completion(s) to "fill in the middle" of the prompt prefix and suffix.

We are continuously working on improvements to provide better context from available sources in the prompt.

Based on the prompt and session context (e.g. was the last suggestion accepted?), determines if a request should be sent to the model.

We are continuously working on new and improved model engines optimized for Copilot.

Local

Server

Simplified system diagram focused on model quality efforts. Made by Alice Li, machine learning researcher at GitHub.

Integrated AI tools - Classification scheme

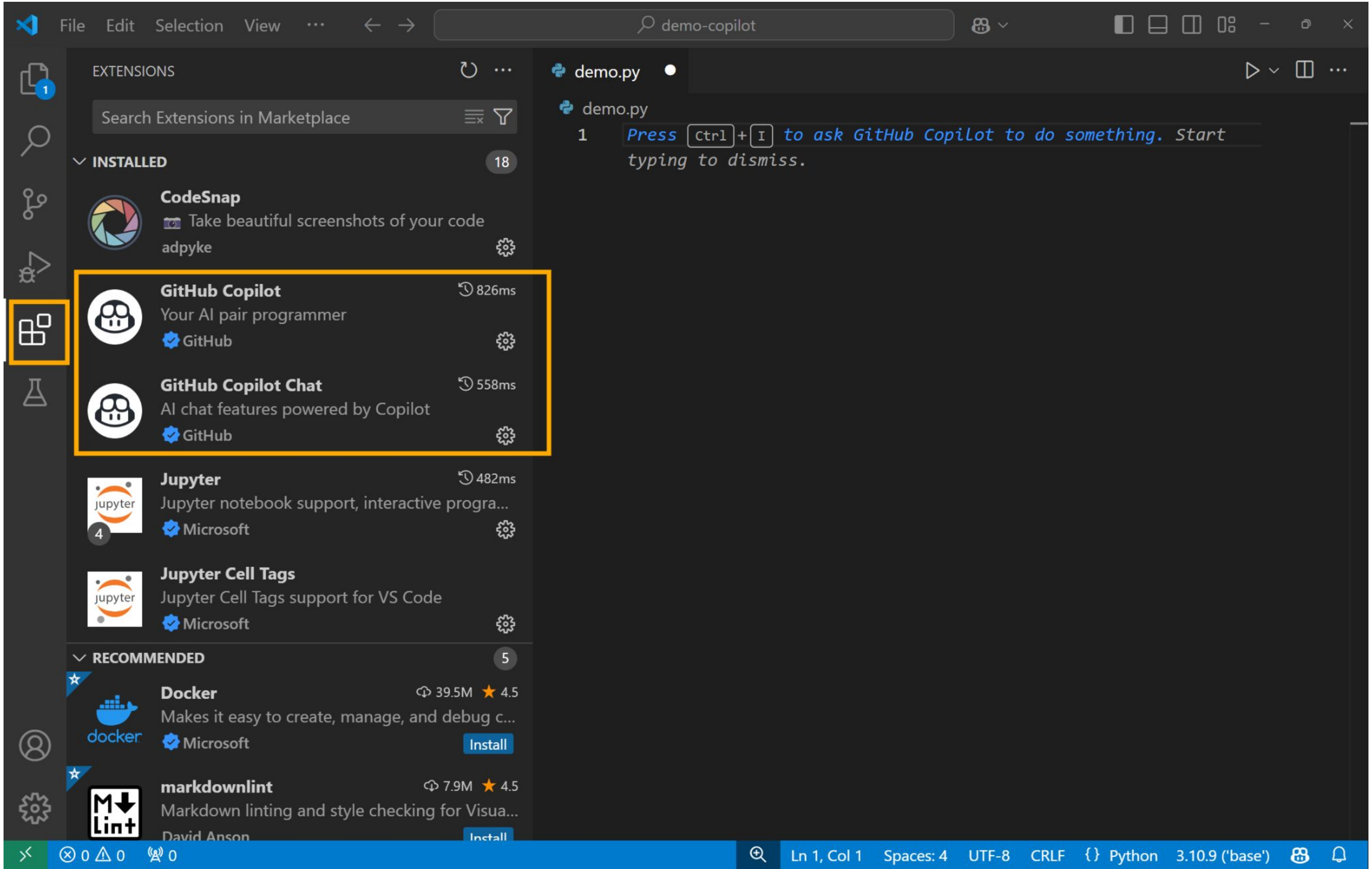
1. Infrastructure
   a. Subscription (Accounts, Team-Accounts)
   b. Configuration (Switching LLMs, changing context windows etc.)
   c. Infrastructural capabilities (Calculation speeds, limits, prompt sizes, AI agents, supported languages)
   d. Data protection (Training, calculating responses)
2. Client
   a. UI based
      i. Direct code interaction
      ii. Indirect code interaction
   b. Use case based
      i. Autocomplete
      ii. Code suggestions
      iii. Integrated commands and prompts
      iv. Integrated chat
      v. Chat history
      vi. Explicit context
      vii. Understand outputs (logging, console, stacktraces etc.)

Copilots - Coding Assistants

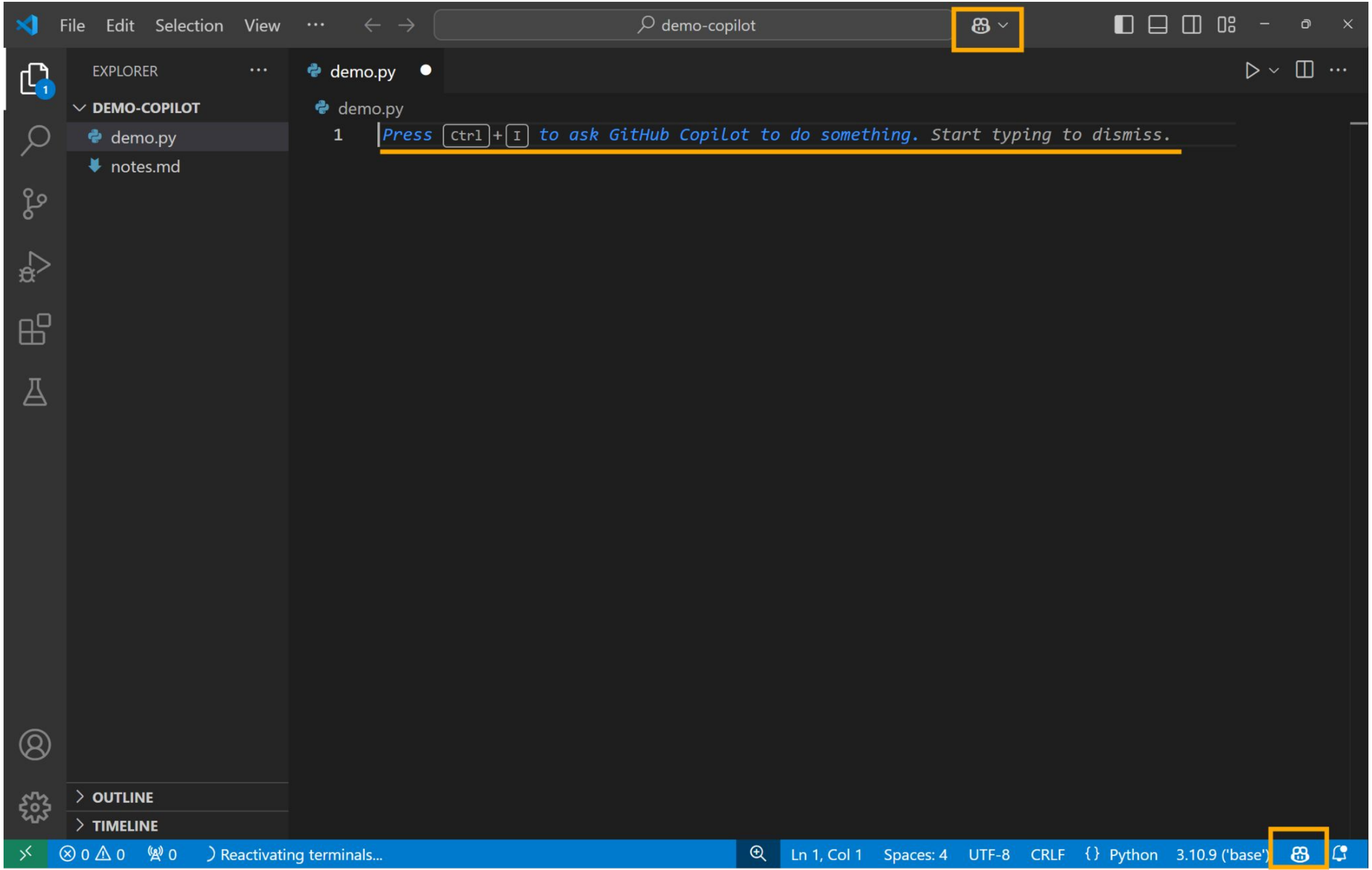
- Github Copilot GitHub Copilot · Your AI pair programmer · GitHub
- JetBrains AI Assistant AI Assistant | IntelliJ IDEA Documentation (jetbrains.com)
- Amazon Q Developer (Code Whisperer) AI Coding Assistant – Amazon Q Developer – AWS
- Tabnine Tabnine AI code assistant | Private, personalized, protected
- Codeium Codeium · Free AI Code Completion & Chat
- Codium AI (Codiumate) Get CodiumAI for VS Code | CodiumAI
- Cody Cody | AI coding assistant (sourcegraph.com)
- AskCodi AskCodi - Your AI coding assistant
- Replit GhostWriter - Replit AI AI - Replit

… and many more!

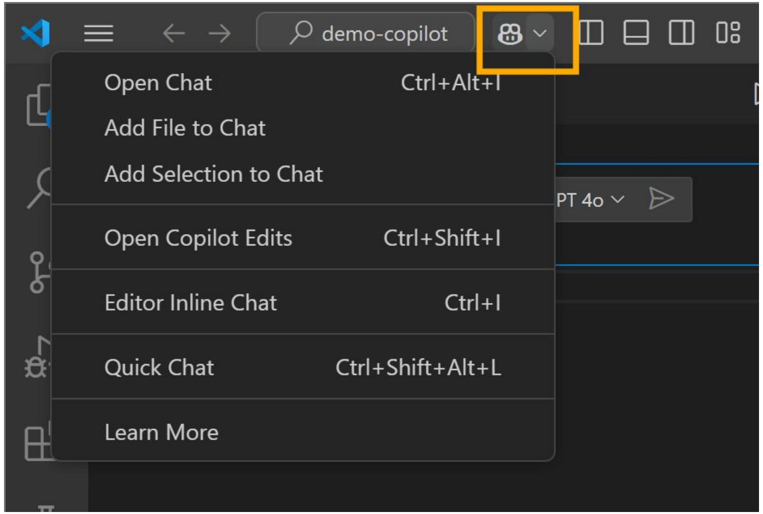
Kernfunktionalitäten

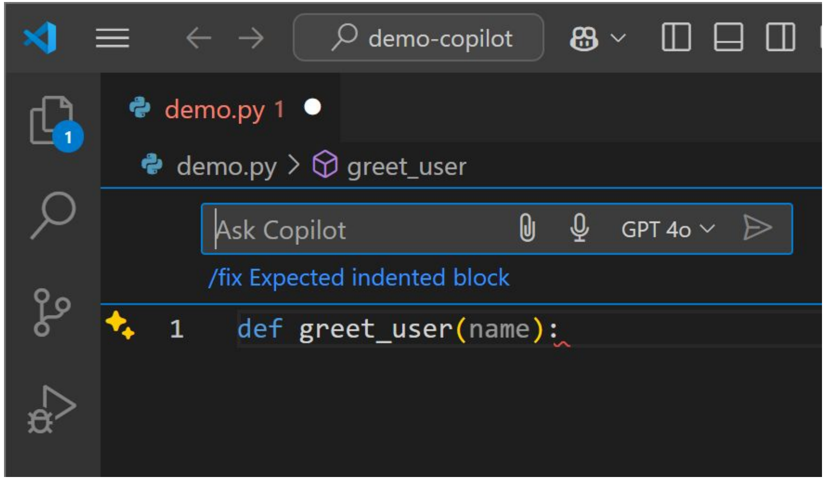


Press Ctrl + I to ask GitHub Copilot to do something. Start typing to dismiss.



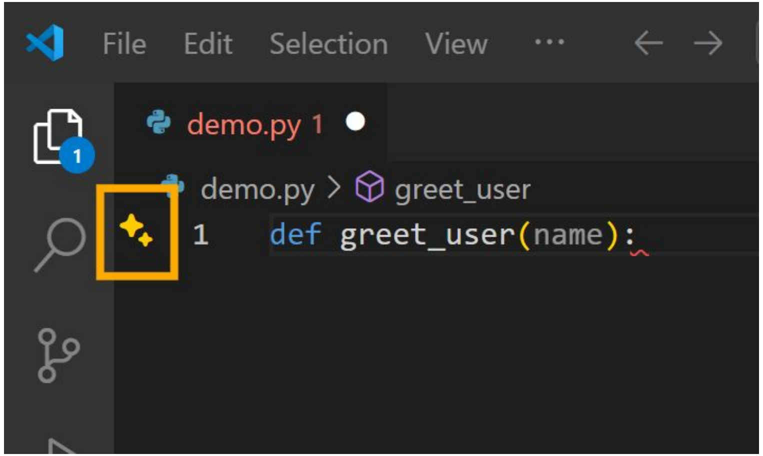
```
Press Ctrl + I to ask GitHub Copilot to do something. Start typing to dismiss.
```



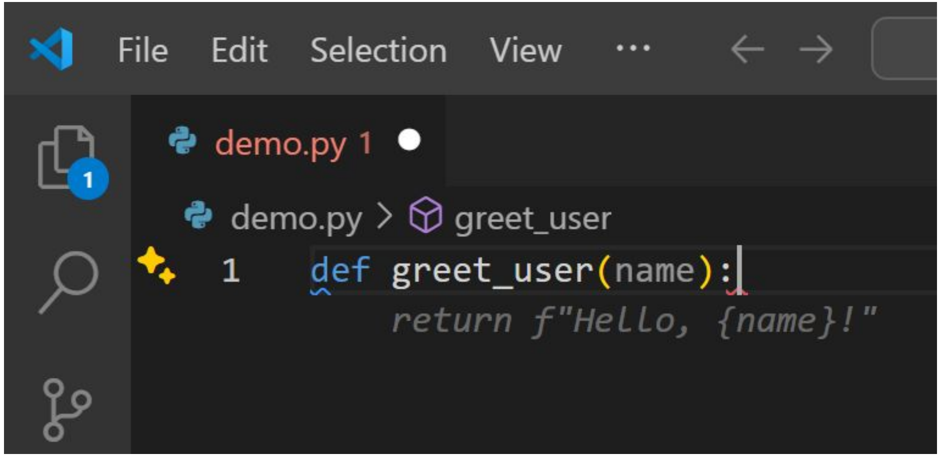


/fix Expected indented block

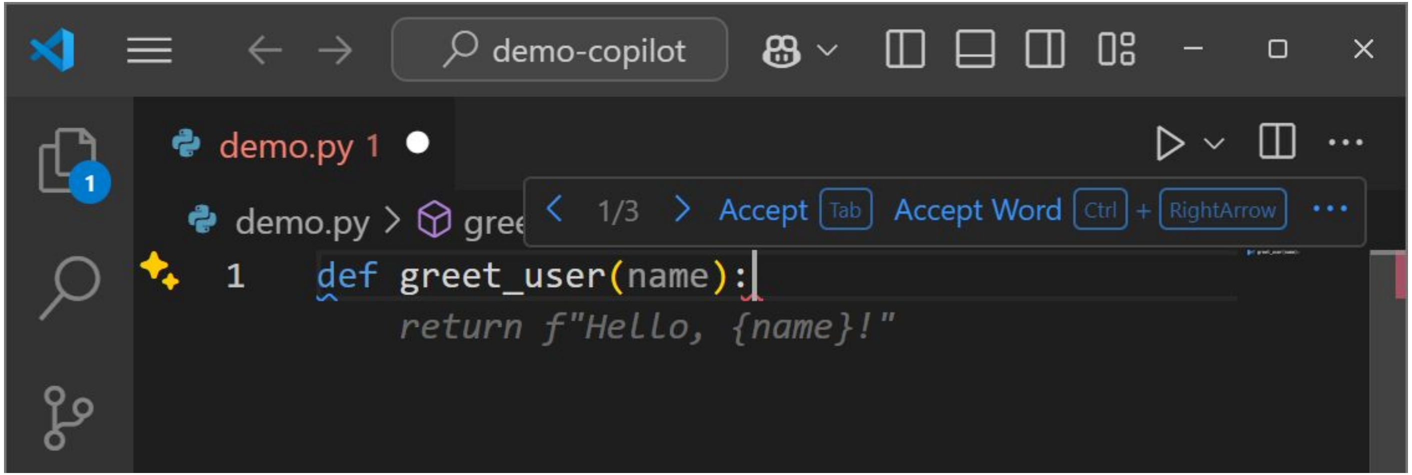
```
def greet_user(name):
```



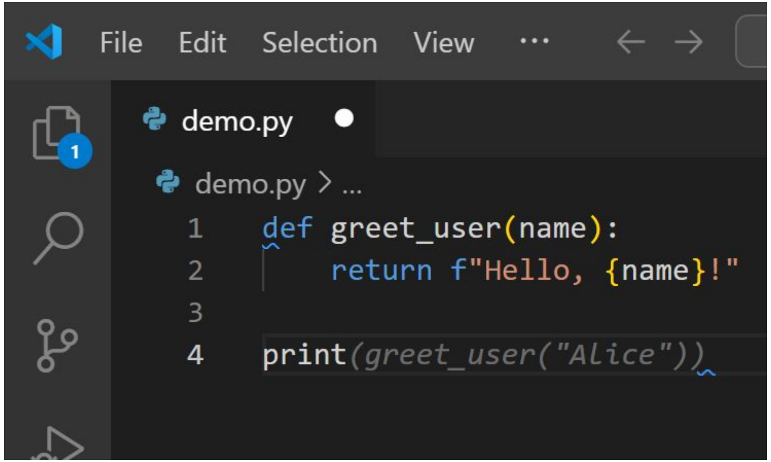
```
def greet_user(name):
```



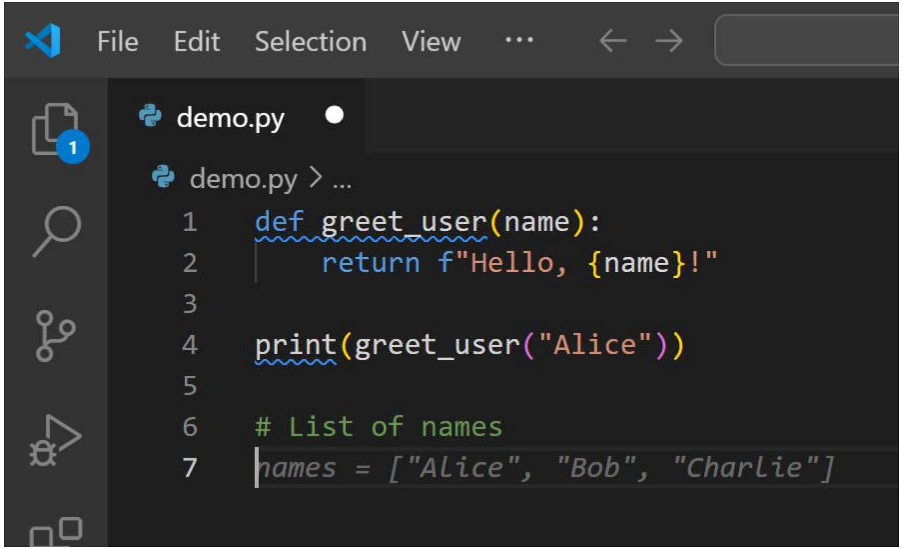
```
def greet_user(name):
    return f"Hello, {name}!"
```



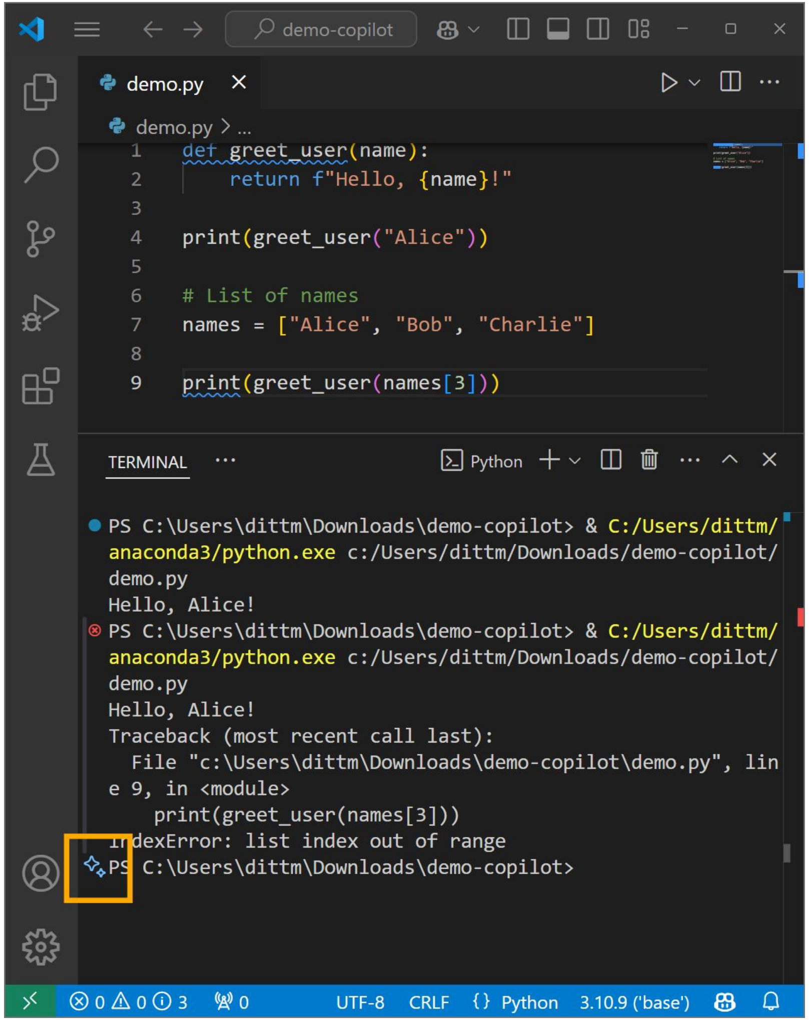
```
def greet_user(name):
    return f"Hello, {name}!"
```



```python
def greet_user(name):
    return f"Hello, {name}!"

print(greet_user("Alice"))
```

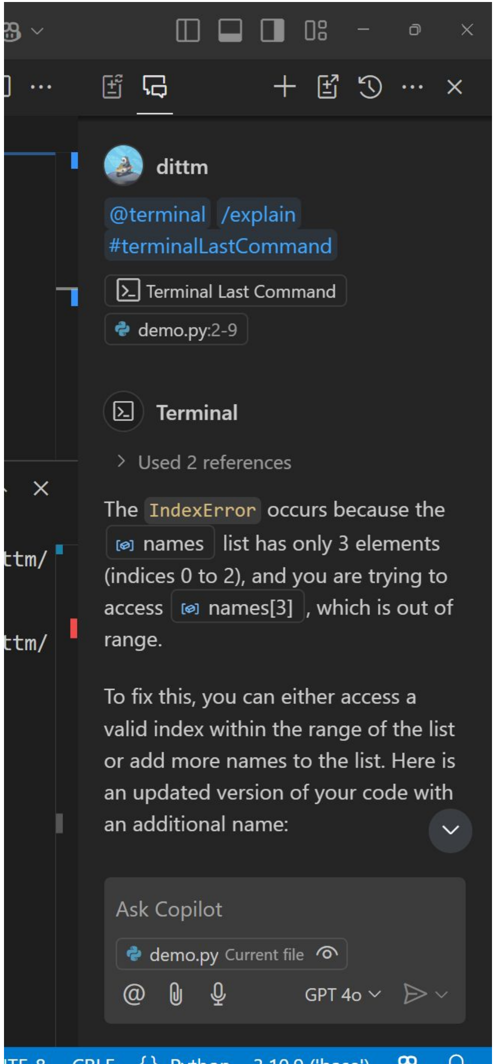


```python
def greet_user(name):
    return f"Hello, {name}!"

print(greet_user("Alice"))

# List of names
names = ["Alice", "Bob", "Charlie"]
```



```python
def greet_user(name):
    return f"Hello, {name}!"

print(greet_user("Alice"))

# List of names
names = ["Alice", "Bob", "Charlie"]

print(greet_user(names[3]))
```

```
PS C:\Users\dittm\Downloads\demo-copilot> & C:/Users/dittm/anaconda3/python.exe c:/Users/dittm/Downloads/demo-copilot/demo.py
Hello, Alice!
PS C:\Users\dittm\Downloads\demo-copilot> & C:/Users/dittm/anaconda3/python.exe c:/Users/dittm/Downloads/demo-copilot/demo.py
Hello, Alice!
Traceback (most recent call last):
  File "c:\Users\dittm\Downloads\demo-copilot\demo.py", line 9, in <module>
    print(greet_user(names[3]))
IndexError: list index out of range
PS C:\Users\dittm\Downloads\demo-copilot>
```



dittm

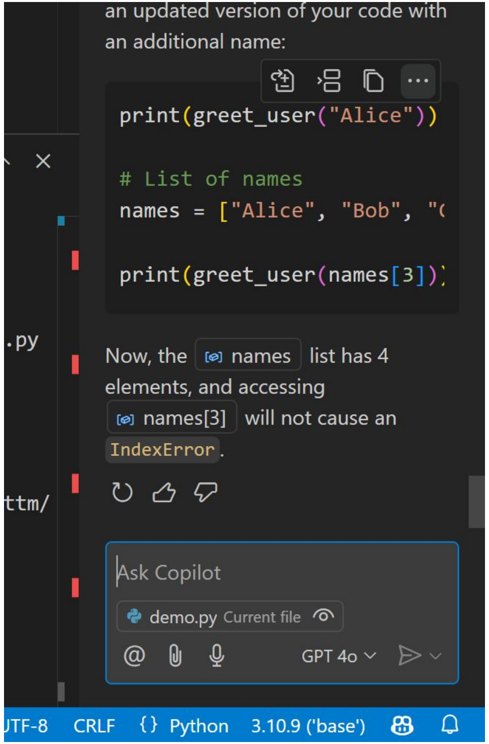
@terminal /explain #terminalLastCommand

Terminal

Used 2 references

The `IndexError` occurs because the `names` list has only 3 elements (indices 0 to 2), and you are trying to access `names[3]`, which is out of range.

To fix this, you can either access a valid index within the range of the list or add more names to the list. Here is an updated version of your code with an additional name:



an updated version of your code with an additional name:

```python
print(greet_user("Alice"))

# List of names
names = ["Alice", "Bob", "C

print(greet_user(names[3]))
```

Now, the `names` list has 4 elements, and accessing `names[3]` will not cause an `IndexError`.

Integrierte KI-Assistenz in Google Colab: Einführung und Übungen

Google Colab

https://colab.google/

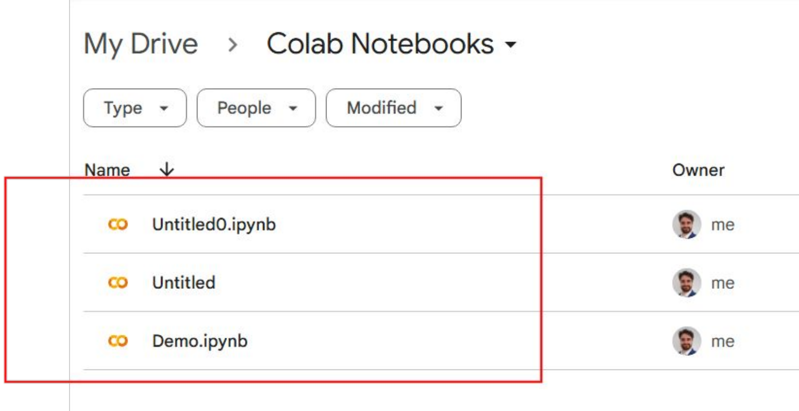
Google Colab (I)

- https://colab.google/
- Free, cloudbased Google-service
- Write, edit and execute code in **jupyter-notebooks**
- Webbased
- Google Gemini AI assistance
- Google Colab Pro and Pro+: Paid service, for Teams working on AI solutions, better GPUs, more storage etc.

Cons: compute units expire after 90 days, costs: 9.99$ per 100 units, or abo

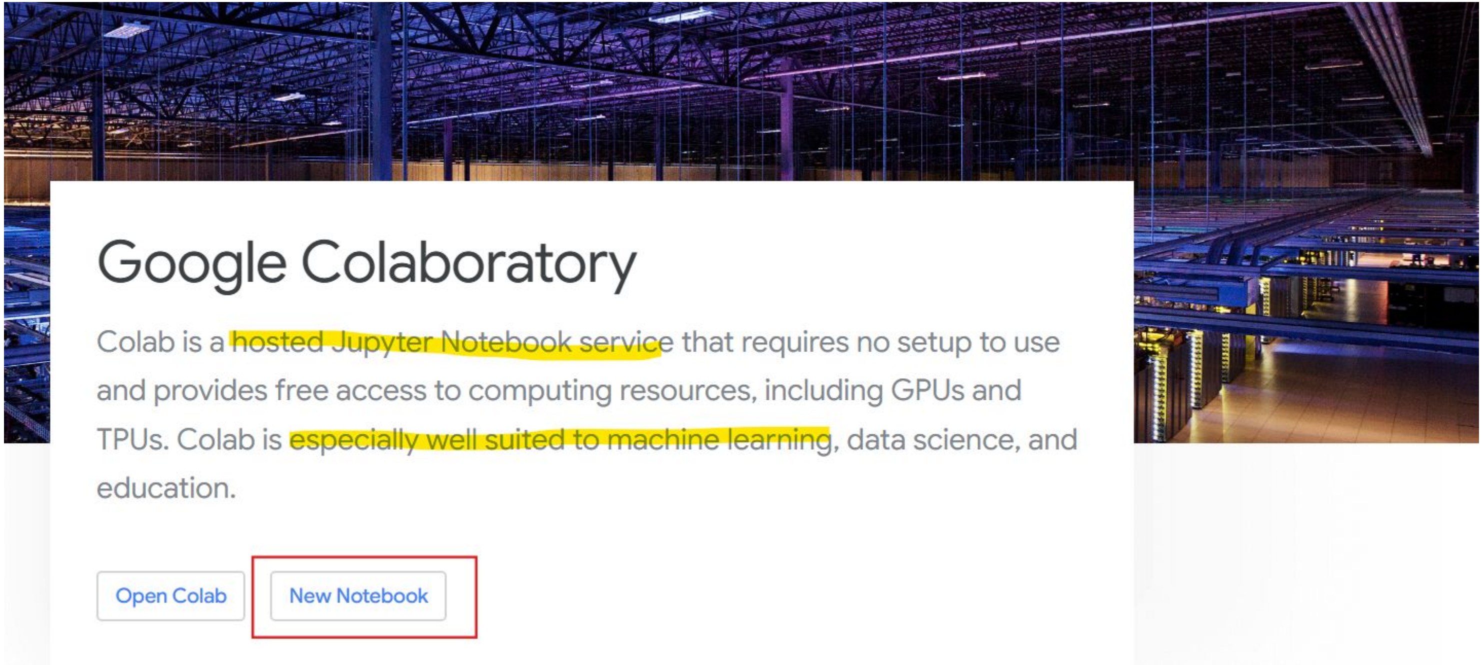
Google Colab (II)

- Verbunden mit **google drive** (jupyter notebooks *.ipynb)
    - Zugang zu notebooks via google drive
- Jupyter notebooks:
    - Mischung aus : Schreiben von Text und Ausführen von Quellcode in einem Dokument.
- KI Assistent → **Google Account** nötig.


My Drive › Colab Notebooks

Name	Owner
Untitled0.ipynb	me
Untitled	me
Demo.ipynb	me

Google Colab (III)

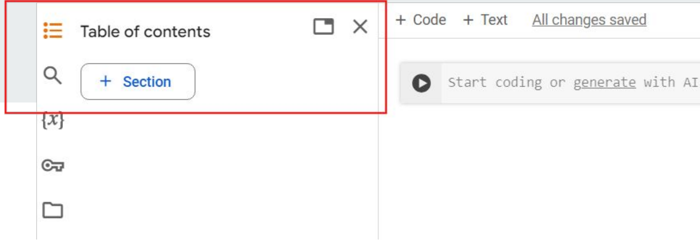


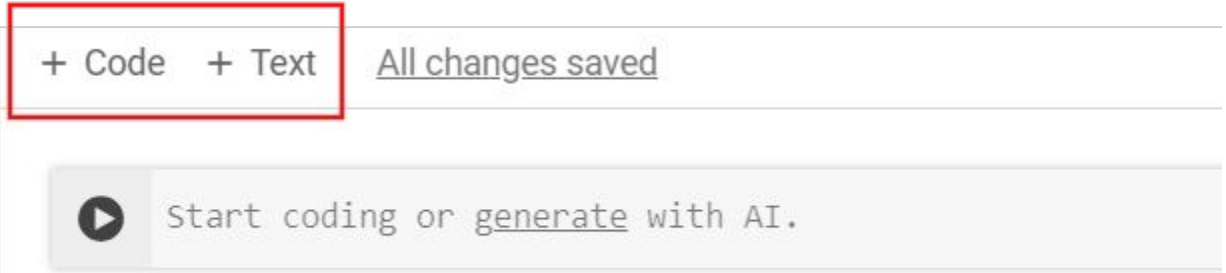
Google Colaboratory

Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs. Colab is especially well suited to machine learning, data science, and education.

Google Colab - Kernfunktionalitäten (I)

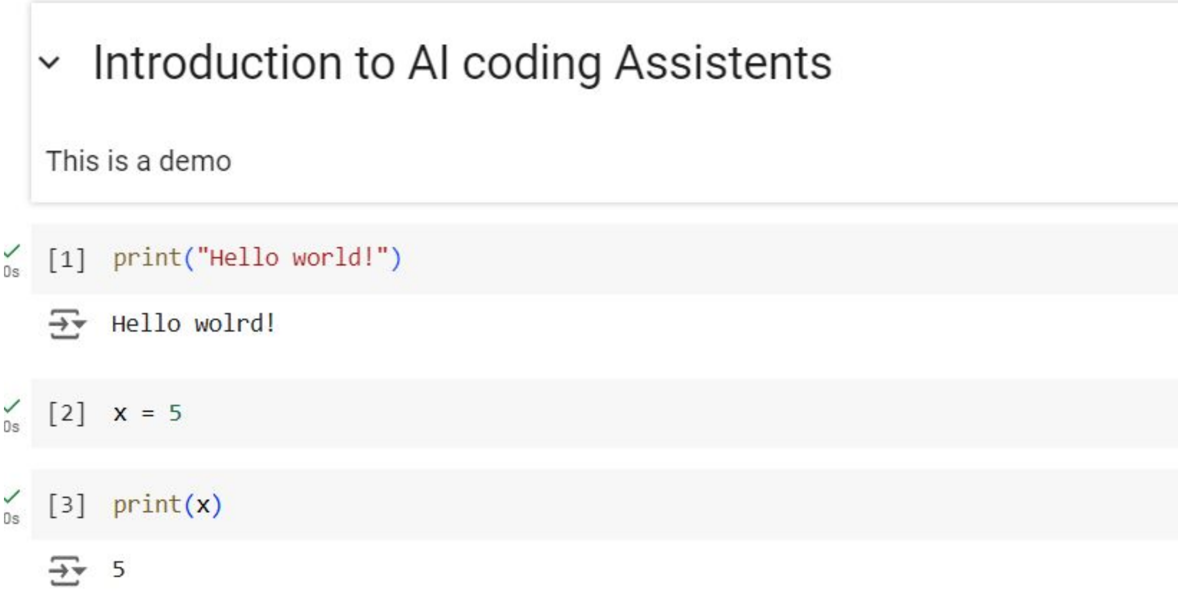
- Jupyter Notebooks: Text oder Code (ausführbar!) hinzufügen.
- “Klassische” Office-Funktionalitäten: Überschriften, Abschnitte, Navgiation, etc.





Google Colab - Kernfunktionalitäten (II)

- Blöcke anlegen und ausführen



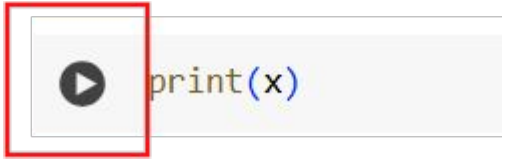
Introduction to AI coding Assistents

This is a demo

```
[1] print("Hello world!")

Hello wolrd!

[2] x = 5

[3] print(x)

5
```

Google Colab - Kernfunktionalitäten (III)

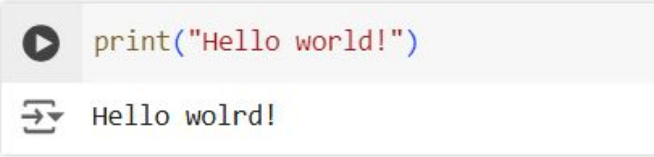
- Code block ausführen



```
print(x)
```

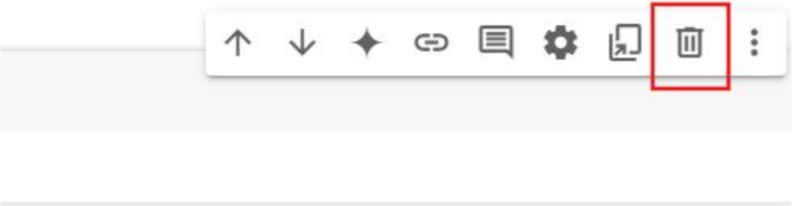
Google Colab - Kernfunktionalitäten (IV)

- Block Interaktion (“on click”):
  - Removing, customizing, commenting, moving around etc.
  - Entfernen, bearbeiten, kommentieren, verschieben, etc.



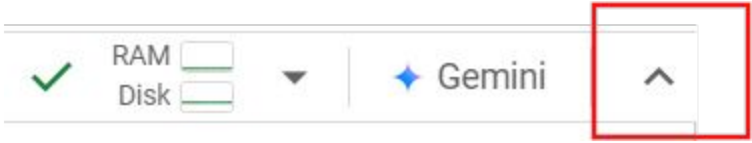
```
print("Hello world!")
```

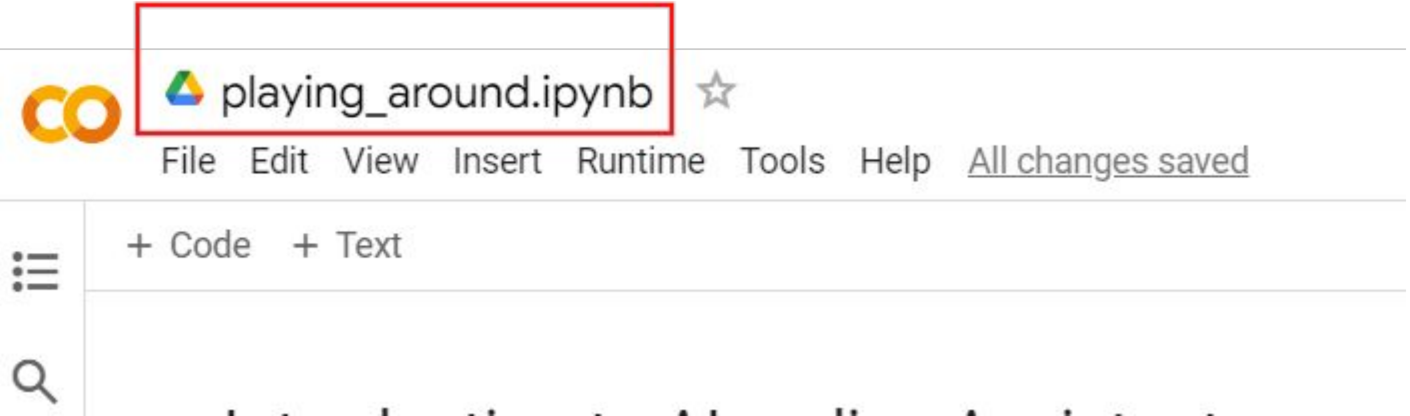
```
Hello wolrd!
```



Google Colab - Kernfunktionalitäten (V)

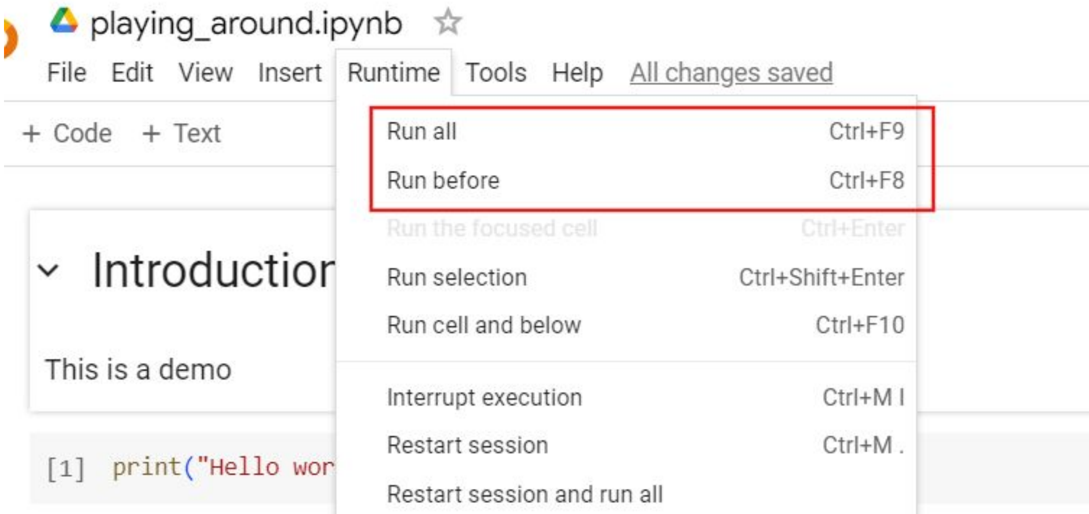
- Dokument Einstellungen (Titel, etc.)





Google Colab - Kernfunktionalitäten (VI)

- Alle Codeblöcke ausführen



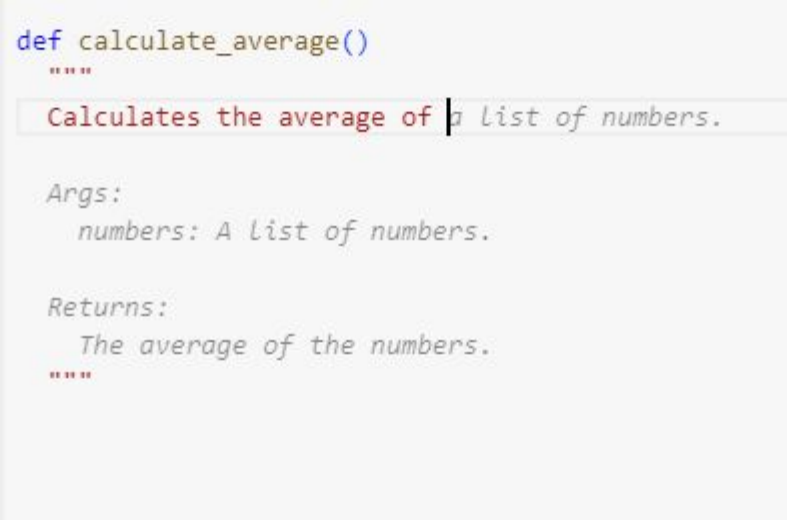
This is a demo

```
[1] print("Hello wor
```

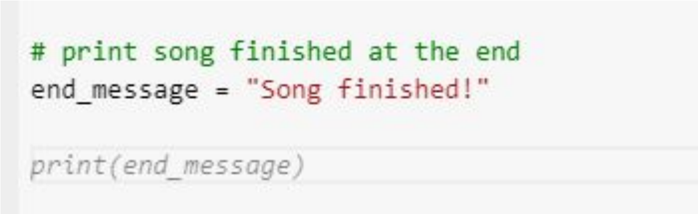
1. Google Colab und KI Assistenten: Direct code interaction

Google colab

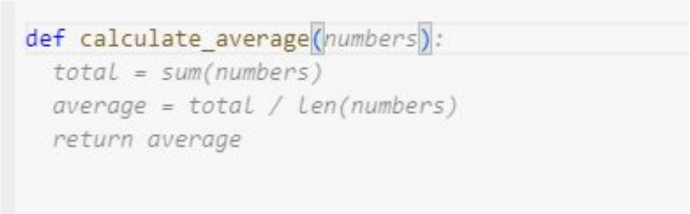
Autocomplete



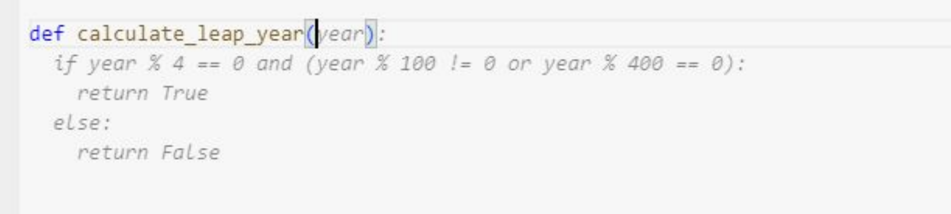
```
def calculate_average()
  """
  Calculates the average of a list of numbers.

  Args:
    numbers: A list of numbers.

  Returns:
    The average of the numbers.
  """
```



```
# print song finished at the end
end_message = "Song finished!"

print(end_message)
```



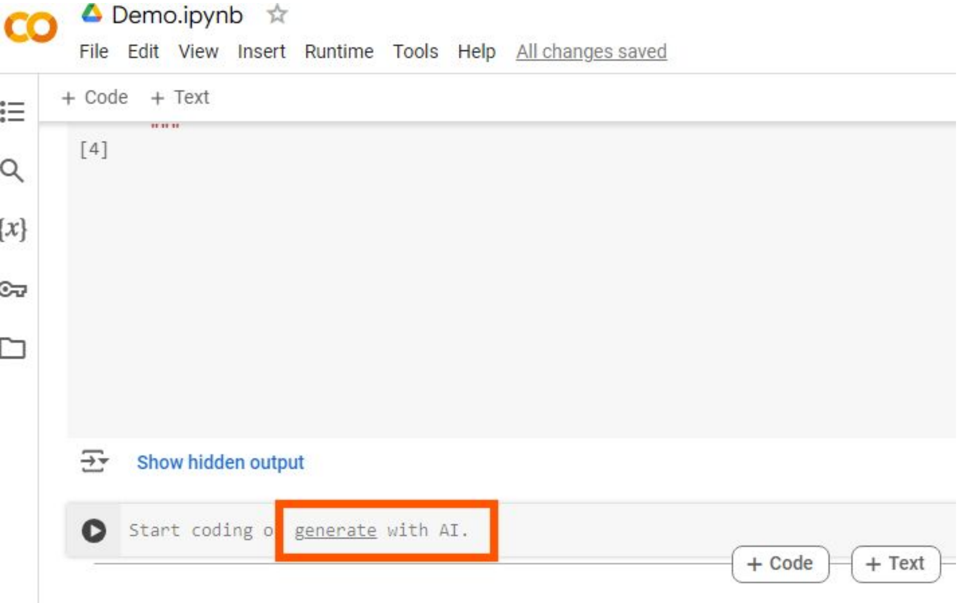
```
def calculate_average(numbers):
  total = sum(numbers)
  average = total / len(numbers)
  return average
```

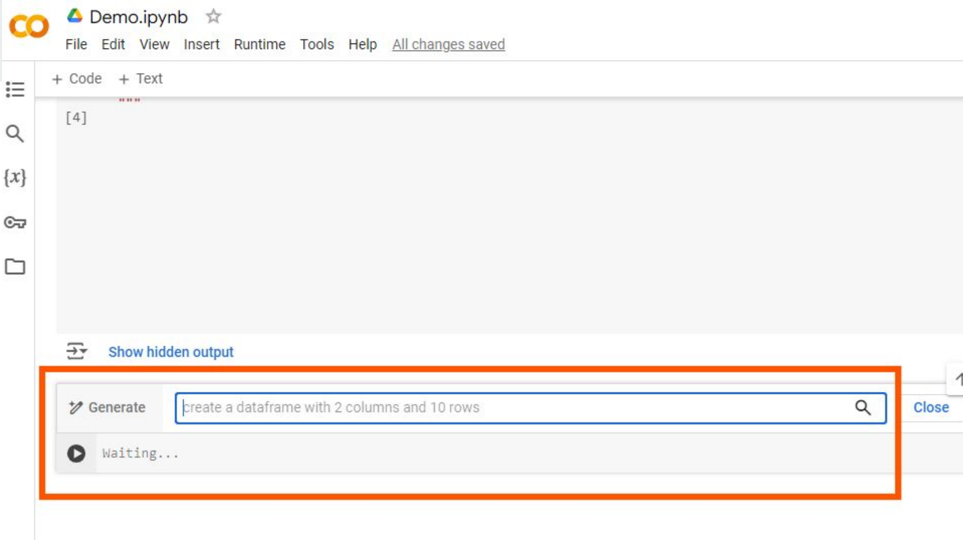


```
def calculate_leap_year(year):
  if year % 4 == 0 and (year % 100 != 0 or year % 400 == 0):
    return True
  else:
    return False
```

2. Google Colab und KI Assistenten: Indirect Code interaction

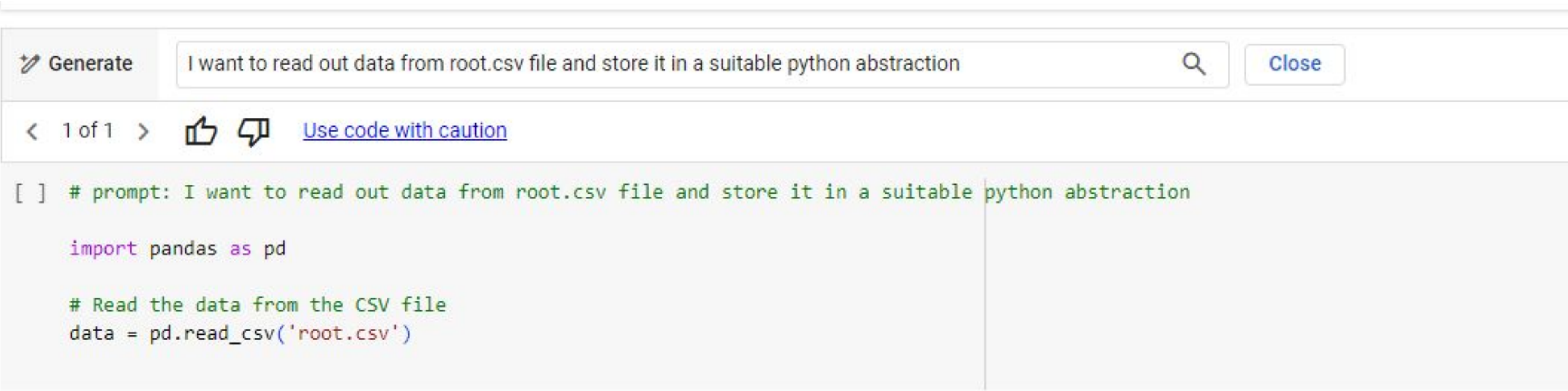
Google Colab


Demo.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[4]
Show hidden output
Start coding or generate with AI.
+ Code
+ Text


Demo.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[4]
Show hidden output
Generate
create a dataframe with 2 columns and 10 rows
Close
Waiting...

Integrated commands and prompts

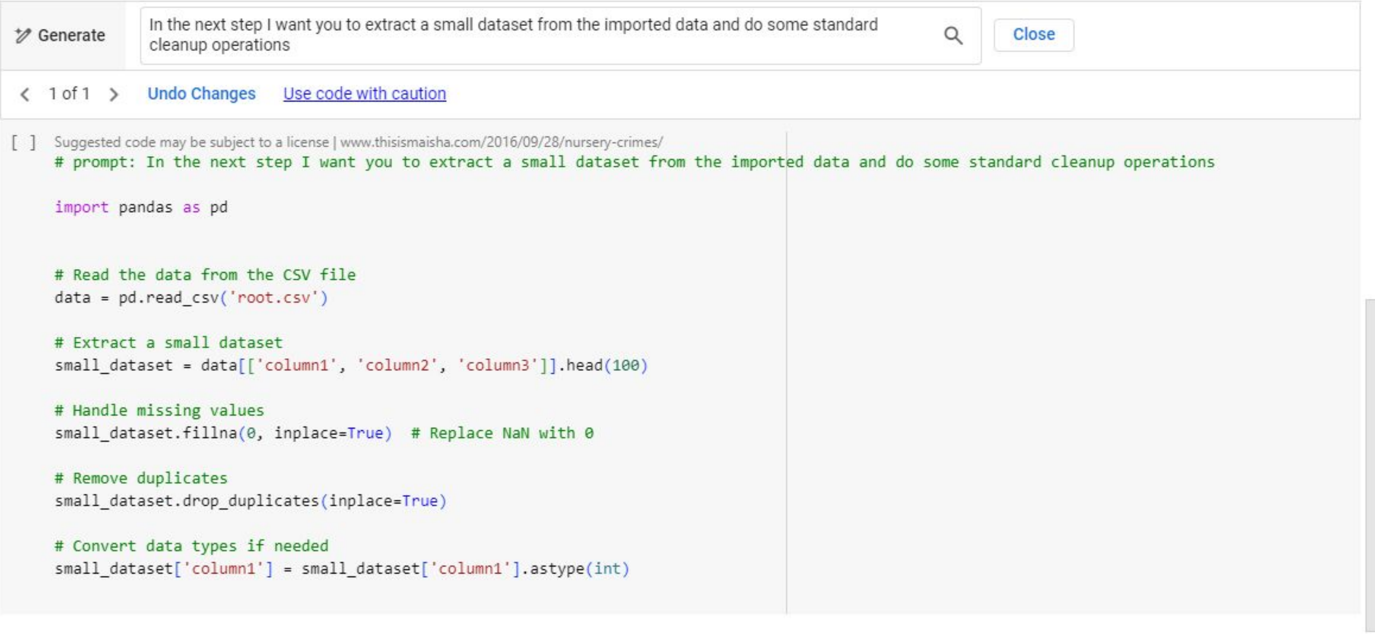
Google Colab



I want to read out data from root.csv file and store it in a suitable python abstraction

```
[ ] # prompt: I want to read out data from root.csv file and store it in a suitable python abstraction

    import pandas as pd

    # Read the data from the CSV file
    data = pd.read_csv('root.csv')
```

Google Colab



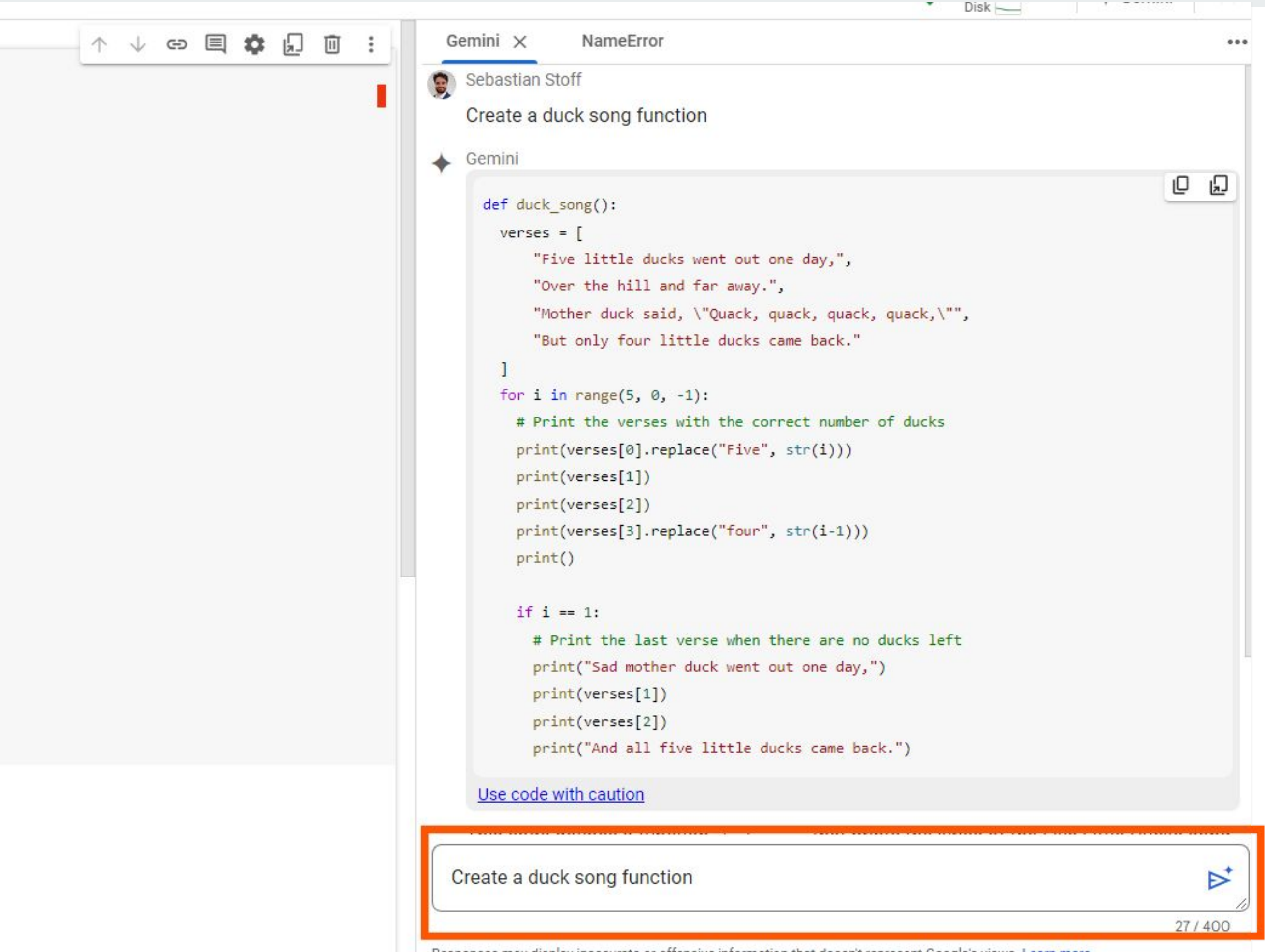
In the next step I want you to extract a small dataset from the imported data and do some standard cleanup operations

Suggested code may be subject to a license | www.thisismaisha.com/2016/09/28/nursery-crimes/

```
# prompt: In the next step I want you to extract a small dataset from the imported data and do some standard cleanup operations

import pandas as pd


# Read the data from the CSV file
data = pd.read_csv('root.csv')

# Extract a small dataset
small_dataset = data[['column1', 'column2', 'column3']].head(100)

# Handle missing values
small_dataset.fillna(0, inplace=True)  # Replace NaN with 0

# Remove duplicates
small_dataset.drop_duplicates(inplace=True)

# Convert data types if needed
small_dataset['column1'] = small_dataset['column1'].astype(int)
```

Google Colab

Integrierter Chat



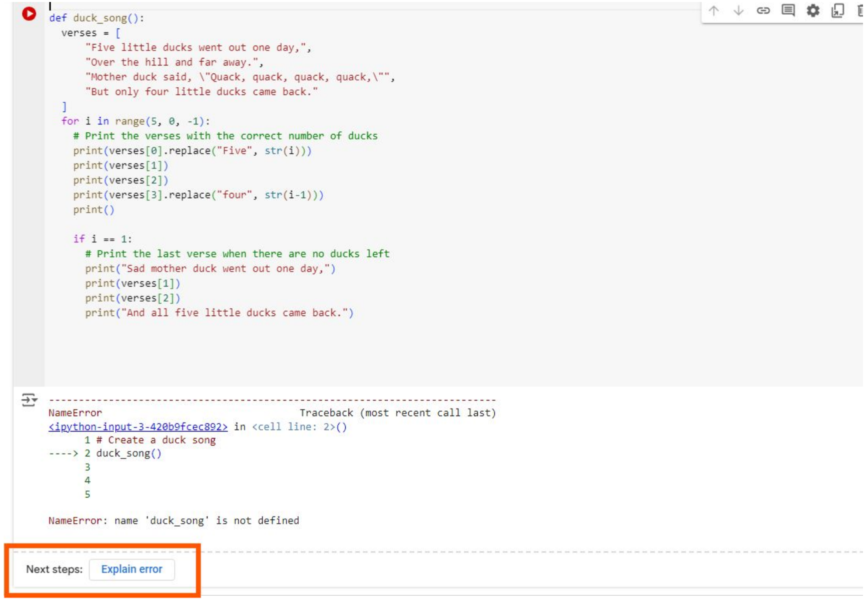
Create a duck song function

```
def duck_song():
 verses = [
   "Five little ducks went out one day,",
   "Over the hill and far away.",
   "Mother duck said, \"Quack, quack, quack, quack,\"",
   "But only four little ducks came back."
 ]
 for i in range(5, 0, -1):
  # Print the verses with the correct number of ducks
  print(verses[0].replace("Five", str(i)))
  print(verses[1])
  print(verses[2])
  print(verses[3].replace("four", str(i-1)))
  print()

  if i == 1:
   # Print the last verse when there are no ducks left
   print("Sad mother duck went out one day,")
   print(verses[1])
   print(verses[2])
   print("And all five little ducks came back.")
```

Create a duck song function

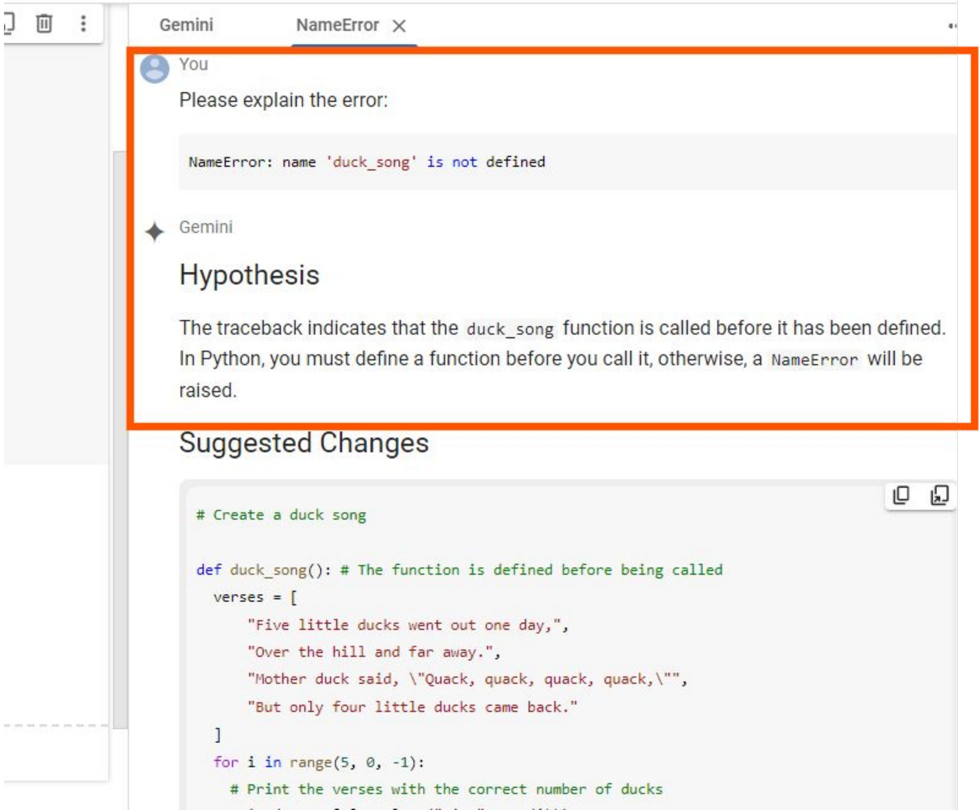
Google Colab

Ausgabe verstehen



```
def duck_song():
  verses = [
      "Five little ducks went out one day,",
      "Over the hill and far away.",
      "Mother duck said, \"Quack, quack, quack, quack,\"",
      "But only four little ducks came back."
  ]
  for i in range(5, 0, -1):
    # Print the verses with the correct number of ducks
    print(verses[0].replace("Five", str(i)))
    print(verses[1])
    print(verses[2])
    print(verses[3].replace("four", str(i-1)))
    print()

    if i == 1:
      # Print the last verse when there are no ducks left
      print("Sad mother duck went out one day,")
      print(verses[1])
      print(verses[2])
      print("And all five little ducks came back.")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-420b9fcec892> in <cell line: 2>()
      1 # Create a duck song
----> 2 duck_song()
      3
      4
      5

NameError: name 'duck_song' is not defined
```



You

Please explain the error:

```
NameError: name 'duck_song' is not defined
```

Gemini

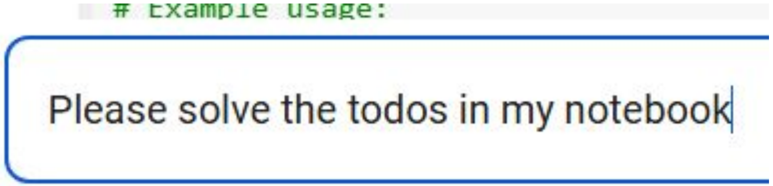
Hypothesis

The traceback indicates that the `duck_song` function is called before it has been defined. In Python, you must define a function before you call it, otherwise, a `NameError` will be raised.

Suggested Changes

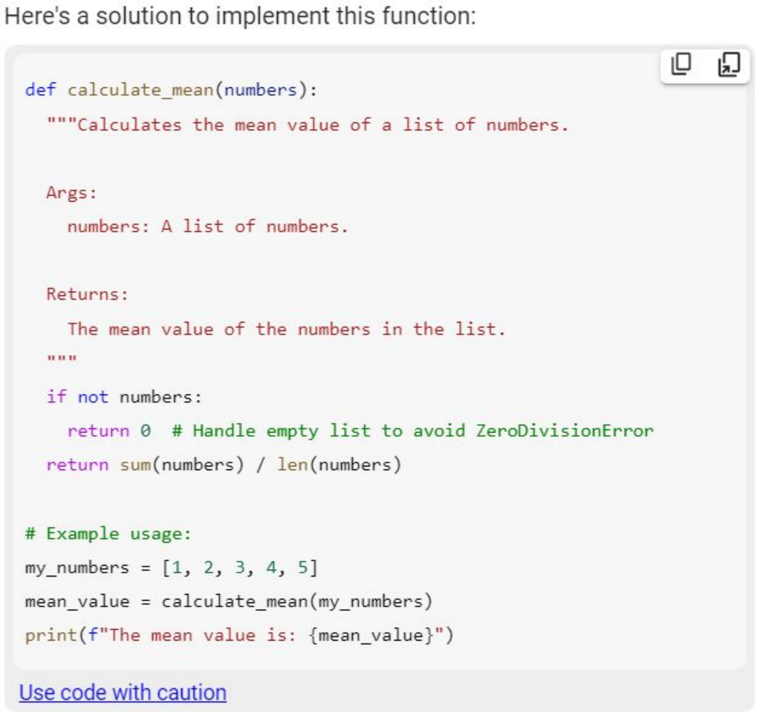
```
# Create a duck song

def duck_song(): # The function is defined before being called
  verses = [
      "Five little ducks went out one day,",
      "Over the hill and far away.",
      "Mother duck said, \"Quack, quack, quack, quack,\"",
      "But only four little ducks came back."
  ]
  for i in range(5, 0, -1):
    # Print the verses with the correct number of ducks
```

Google Colab



```
# Example usage:
```

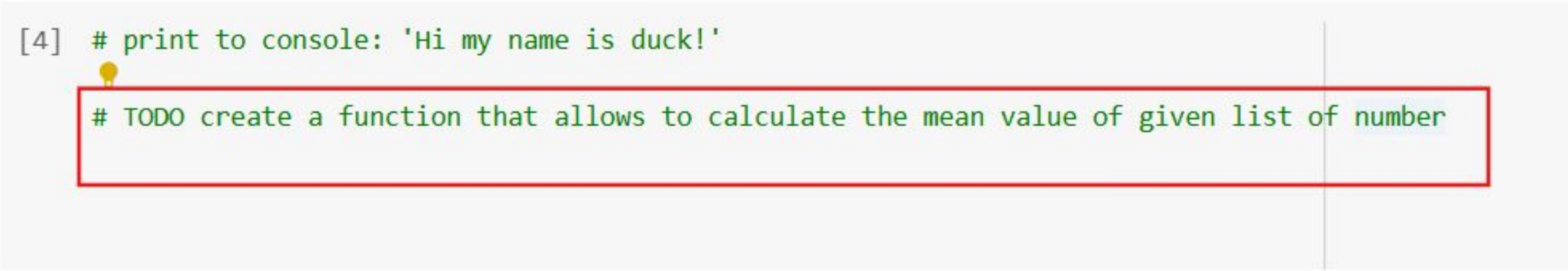
Please solve the todos in my notebook



Here's a solution to implement this function:

```
def calculate_mean(numbers):
  """Calculates the mean value of a list of numbers.

  Args:
    numbers: A list of numbers.

  Returns:
    The mean value of the numbers in the list.
  """
  if not numbers:
    return 0  # Handle empty list to avoid ZeroDivisionError
  return sum(numbers) / len(numbers)

# Example usage:
my_numbers = [1, 2, 3, 4, 5]
mean_value = calculate_mean(my_numbers)
print(f"The mean value is: {mean_value}")
```

Use code with caution



```
[4] # print to console: 'Hi my name is duck!'

    # TODO create a function that allows to calculate the mean value of given list of number
```

Experimente mit Google Colab

(und integrierter KI-Assistenz)

Experiment 01:

- Funktion sum() so komplex wie möglich erstellen mit mindestens 100 Codezeilen.
    - Code smells hinzufügen
    - Irreführende Variablennamen etc.

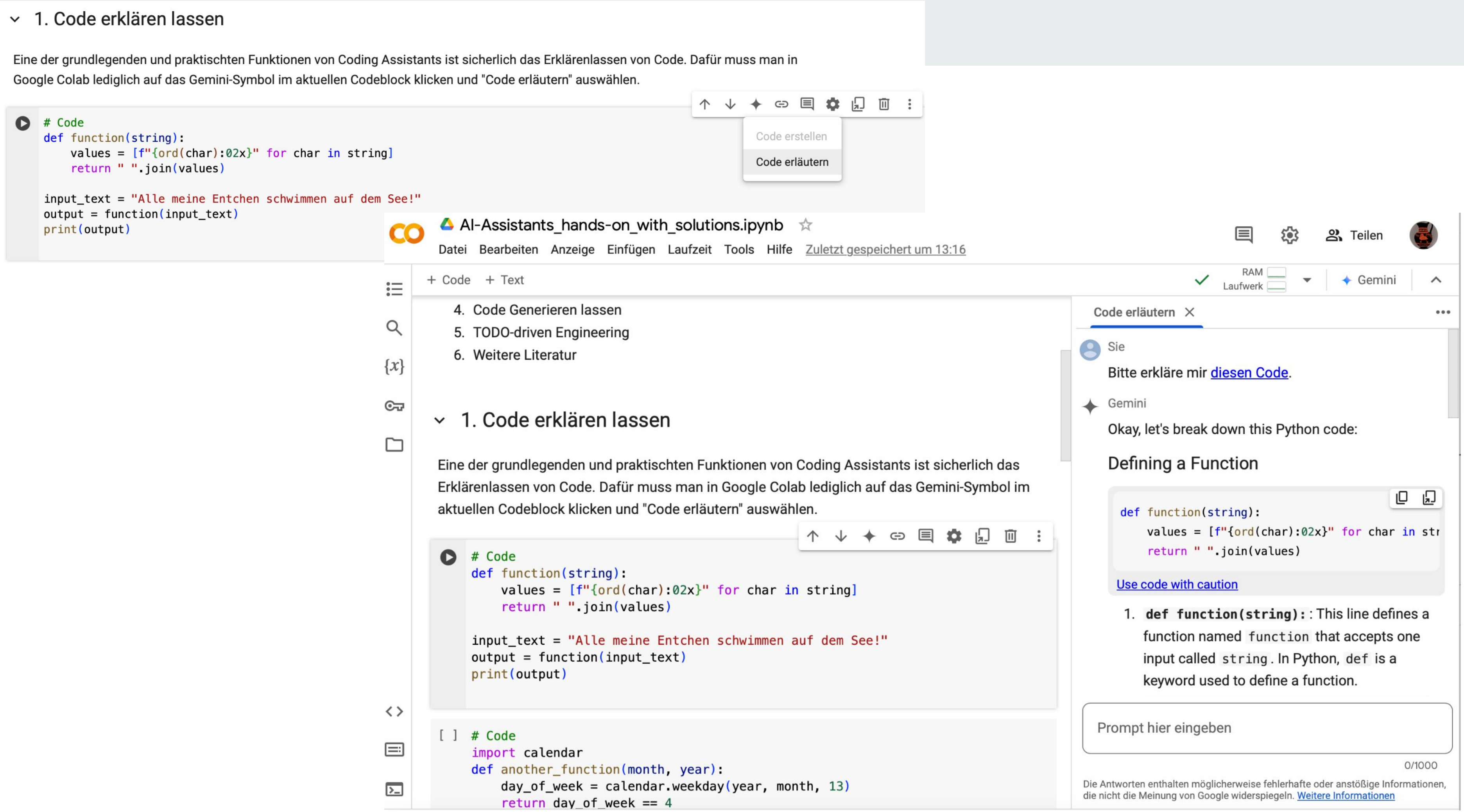
Experiment 02:

- Refactoring: Komplexe sum() Funktion überarbeiten
    - Neues Chat-Fenster

Übungen mit Google Colab

HANDS-ON

https://drive.google.com/drive/folders/1pbi69sVXxN966IHiJ_aLRHCGDHcMc0eQ?usp=sharing



1. Code erklären lassen

Eine der grundlegenden und praktischten Funktionen von Coding Assistants ist sicherlich das Erklärenlassen von Code. Dafür muss man in Google Colab lediglich auf das Gemini-Symbol im aktuellen Codeblock klicken und "Code erläutern" auswählen.

```
# Code
def function(string):
    values = [f"{ord(char):02x}" for char in string]
    return " ".join(values)

input_text = "Alle meine Entchen schwimmen auf dem See!"
output = function(input_text)
print(output)
```

4. Code Generieren lassen
5. TODO-driven Engineering
6. Weitere Literatur

1. Code erklären lassen

Eine der grundlegenden und praktischten Funktionen von Coding Assistants ist sicherlich das Erklärenlassen von Code. Dafür muss man in Google Colab lediglich auf das Gemini-Symbol im aktuellen Codeblock klicken und "Code erläutern" auswählen.

```
# Code
def function(string):
    values = [f"{ord(char):02x}" for char in string]
    return " ".join(values)

input_text = "Alle meine Entchen schwimmen auf dem See!"
output = function(input_text)
print(output)
```

```
[ ] # Code
    import calendar
    def another_function(month, year):
        day_of_week = calendar.weekday(year, month, 13)
        return day_of_week == 4
```

Sie

Bitte erkläre mir diesen Code.

Gemini

Okay, let's break down this Python code:

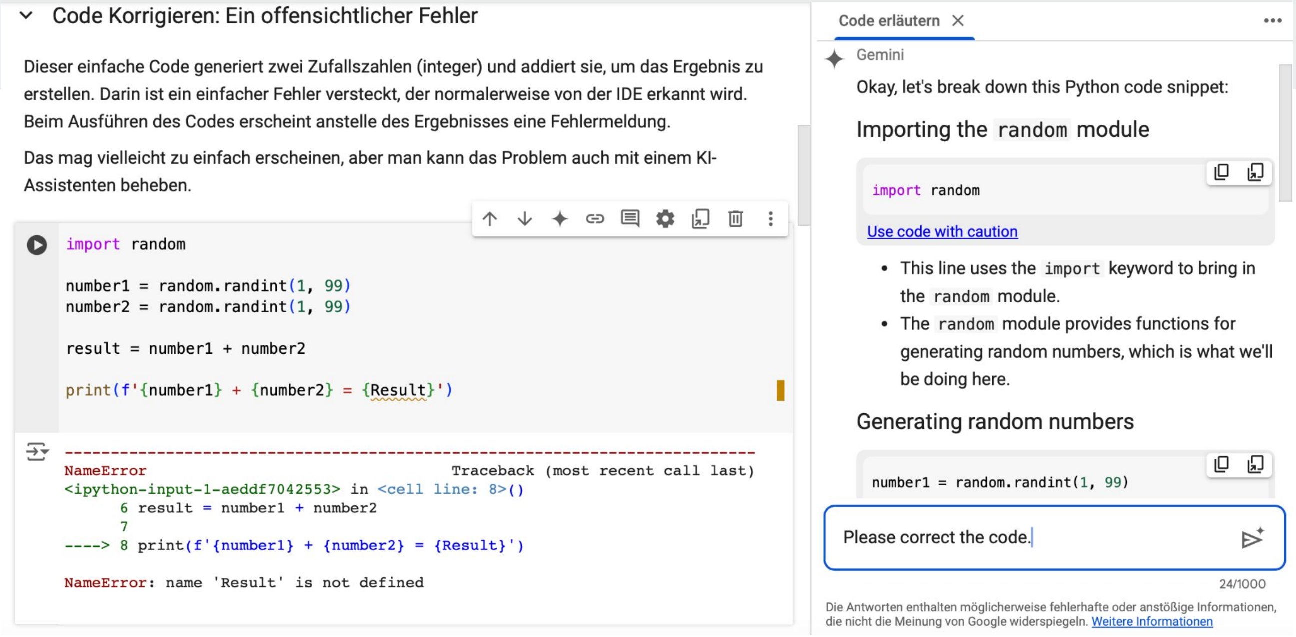
Defining a Function

```
def function(string):
    values = [f"{ord(char):02x}" for char in str
    return " ".join(values)
```

Use code with caution

1. **`def function(string):`** : This line defines a function named `function` that accepts one input called `string`. In Python, `def` is a keyword used to define a function.

Die Antworten enthalten möglicherweise fehlerhafte oder anstößige Informationen, die nicht die Meinung von Google widerspiegeln. Weitere Informationen



Code Korrigieren: Ein offensichtlicher Fehler

Dieser einfache Code generiert zwei Zufallszahlen (integer) und addiert sie, um das Ergebnis zu erstellen. Darin ist ein einfacher Fehler versteckt, der normalerweise von der IDE erkannt wird. Beim Ausführen des Codes erscheint anstelle des Ergebnisses eine Fehlermeldung.

Das mag vielleicht zu einfach erscheinen, aber man kann das Problem auch mit einem KI-Assistenten beheben.

```
import random

number1 = random.randint(1, 99)
number2 = random.randint(1, 99)

result = number1 + number2

print(f'{number1} + {number2} = {Result}')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-aeddf7042553> in <cell line: 8>()
      6 result = number1 + number2
      7 
----> 8 print(f'{number1} + {number2} = {Result}')

NameError: name 'Result' is not defined
```

Okay, let's break down this Python code snippet:

Importing the `random` module

```
import random
```

Use code with caution

- This line uses the `import` keyword to bring in the `random` module.
- The `random` module provides functions for generating random numbers, which is what we'll be doing here.

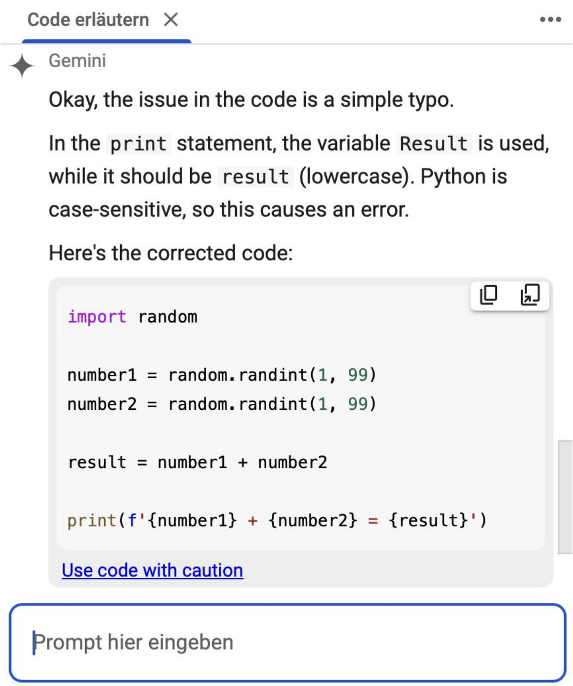
Generating random numbers

```
number1 = random.randint(1, 99)
```

Please correct the code.

24/1000

Die Antworten enthalten möglicherweise fehlerhafte oder anstößige Informationen, die nicht die Meinung von Google widerspiegeln. Weitere Informationen



Okay, the issue in the code is a simple typo.

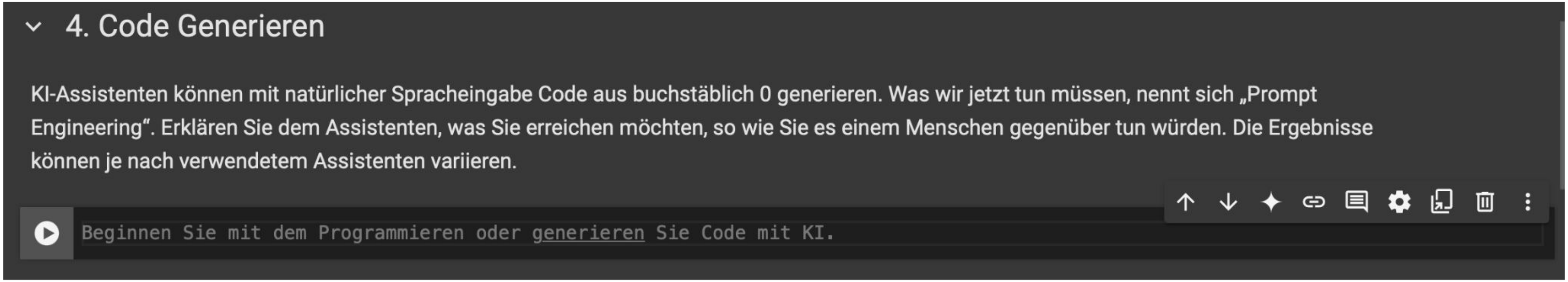
In the `print` statement, the variable `Result` is used, while it should be `result` (lowercase). Python is case-sensitive, so this causes an error.

Here's the corrected code:

```
import random

number1 = random.randint(1, 99)
number2 = random.randint(1, 99)

result = number1 + number2

print(f'{number1} + {number2} = {result}')
```

Use code with caution

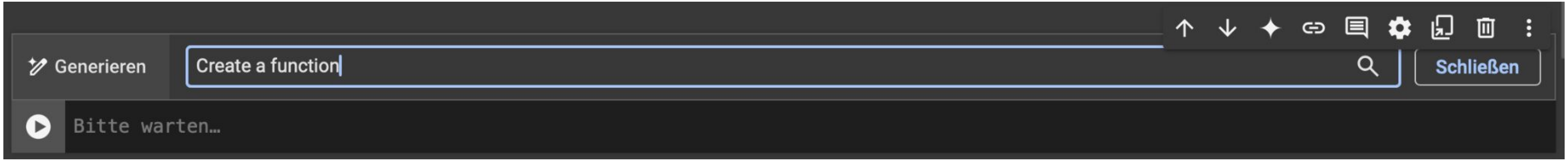
Prompt hier eingeben



4. Code Generieren

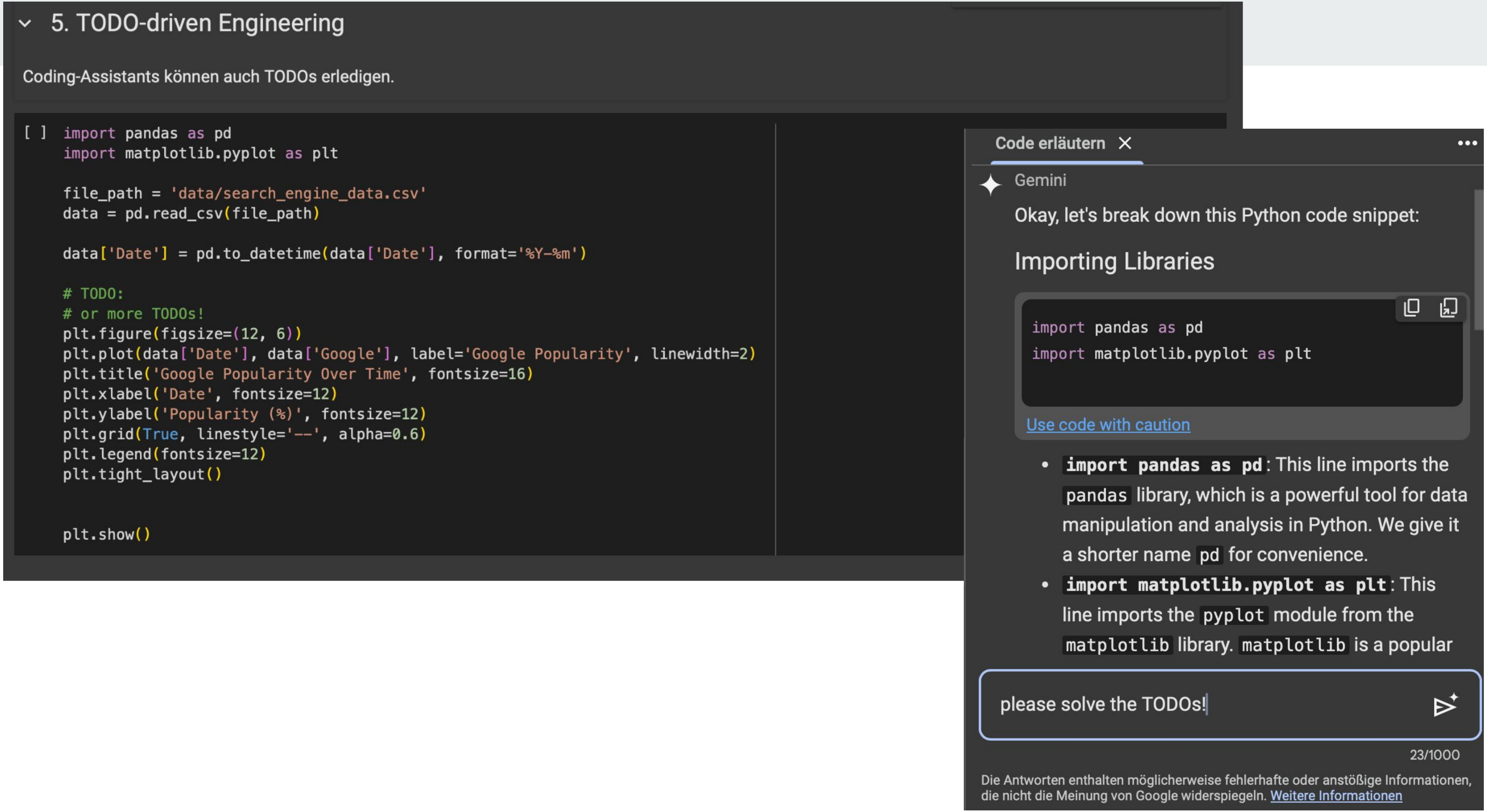
KI-Assistenten können mit natürlicher Spracheingabe Code aus buchstäblich 0 generieren. Was wir jetzt tun müssen, nennt sich „Prompt Engineering“. Erklären Sie dem Assistenten, was Sie erreichen möchten, so wie Sie es einem Menschen gegenüber tun würden. Die Ergebnisse können je nach verwendetem Assistenten variieren.

```
Beginnen Sie mit dem Programmieren oder generieren Sie Code mit KI.
```



Create a function

```
Bitte warten…
```



5. TODO-driven Engineering

Coding-Assistants können auch TODOs erledigen.

```
import pandas as pd
import matplotlib.pyplot as plt

file_path = 'data/search_engine_data.csv'
data = pd.read_csv(file_path)

data['Date'] = pd.to_datetime(data['Date'], format='%Y-%m')

# TODO:
# or more TODOs!
plt.figure(figsize=(12, 6))
plt.plot(data['Date'], data['Google'], label='Google Popularity', linewidth=2)
plt.title('Google Popularity Over Time', fontsize=16)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Popularity (%)', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(fontsize=12)
plt.tight_layout()


plt.show()
```

Gemini

Okay, let's break down this Python code snippet:

Importing Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
```

- `import pandas as pd`: This line imports the `pandas` library, which is a powerful tool for data manipulation and analysis in Python. We give it a shorter name `pd` for convenience.
- `import matplotlib.pyplot as plt`: This line imports the `pyplot` module from the `matplotlib` library. `matplotlib` is a popular

please solve the TODOs!

23/1000

Die Antworten enthalten möglicherweise fehlerhafte oder anstößige Informationen, die nicht die Meinung von Google widerspiegeln. Weitere Informationen

Fragen / Diskussion

Fragen / Diskussion

- Harmonische **Fortentwicklung oder Disruption**?
    - Verlust von dauerhaften, technischen Positionen in den DH? (Entwickler:innen) - oder: neue Positionen?
    - Auswirkungen auf den/die Research-Software-Engineer?
    - KI Assistenten in DH-Programmierung: Unterstützung von X-Technologien?
- **Veränderte Natur** des Research Software Engineerings?
    - Sinkende Kosten in der Entwicklung: Mehr Eigenentwicklung an Institutionen? → Erhöhte Kosten für Wartung und Pflege? → “Mehr Code zu warten”?
    - Abstrakte Beschreibung der Problemstellung wird wichtiger?
    - Erhöhte Bedeutung von Geschäftsprozessmodellierung? (weniger etabliert als Datenmodellierung in den DH?)
- Qualität von **Daten und Modellierung** des realweltichen Problems wird (noch) wichtiger?
    - Modellierung des realen Problems → Unterstützung des realen Problems?