



# Python Intro

---

November  
2024

# LIZENZ

Die Charts & das Kursmaterial wurden – soweit nicht anders angegeben – von Ralf Bendig erstellt.  
Lizenz CC BY 4.0.

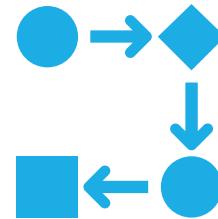
Titelseite: Bild mit ideogram.ai erstellt.

# INHALT

- Intro
- Colab & Jupyter Notebooks
- Python Grundlagen
- Coding & KI
- Entwicklungsprozess
- Integrierte Entwicklungsumgebung
- Style Guide
- Git & GitHub
- Anhang

# INTRO

# KURSORGANISATION



## ZEITPLANUNG

- 5 Tage, Mo - Fr
- Start: 09:00 Uhr
- Ende: 16:30 Uhr
- Pause nach 90 Min

## VORGEHEN

- Grundlagen/Basiswissen
- Beispiele
- Training & Übungen
- Ergebnisse stellen  
Teilnehmer vor

## VERSCHIEDENES

- WLAN
- Pinboard

# KURSINHALT



- Standard-Datentypen: Integer, Float, String, Listen, Dictionary
- Zuweisungen/Variablen, Globale/lokale Variablen
- Rechenoperatoren, Vergleichsoperatoren, Logische Operatoren
- Kontrollstrukturen: Verzweigungen, Schleifen
- Standard-Funktionen und -Methoden, eigene Funktionen,
- Ein-/Ausgaben, Dateien lesen/schreiben, Formatierung
- Nutzung Python-Bibliotheken/Module, eigene Module
- Umgang mit Programmfehlern
- Objektorientierte Programmierung
- Graphische Benutzeroberfläche
- Direkt ausführbare Python-Programme (EXE-Dateien), Konvertierung Notizbücher in .py Dateien

Bild mit ideogram.ai erstellt

# HINWEISE ZUM SKRIPT & DEN JUPYTER NOTEBOOKS



Bild von [Susanne Weitzhofer](#) auf [Pixabay](#)

- Das Skript ist eine **strukturierte Sammlung** von Themen zu Coding.
- Die wesentlichen Themenfelder, wie Datentypen, Kontrollstrukturen, Funktionen und objektorientierte Programmierung, werden dargestellt.
- Das Skript ist als **Begleitmaterial** zum **Kurs** konzipiert und kein klassisches Lehrbuch.
- Hinweise zu **Lehrbüchern** für Coding finden sich in der Literaturliste im Anhang.
- Die Notebooks beinhalten Beispiele & Trainingsmaterial und Lösungen.
- Die Charts & Notebooks sind teilweise **optional**.

# PINBOARD

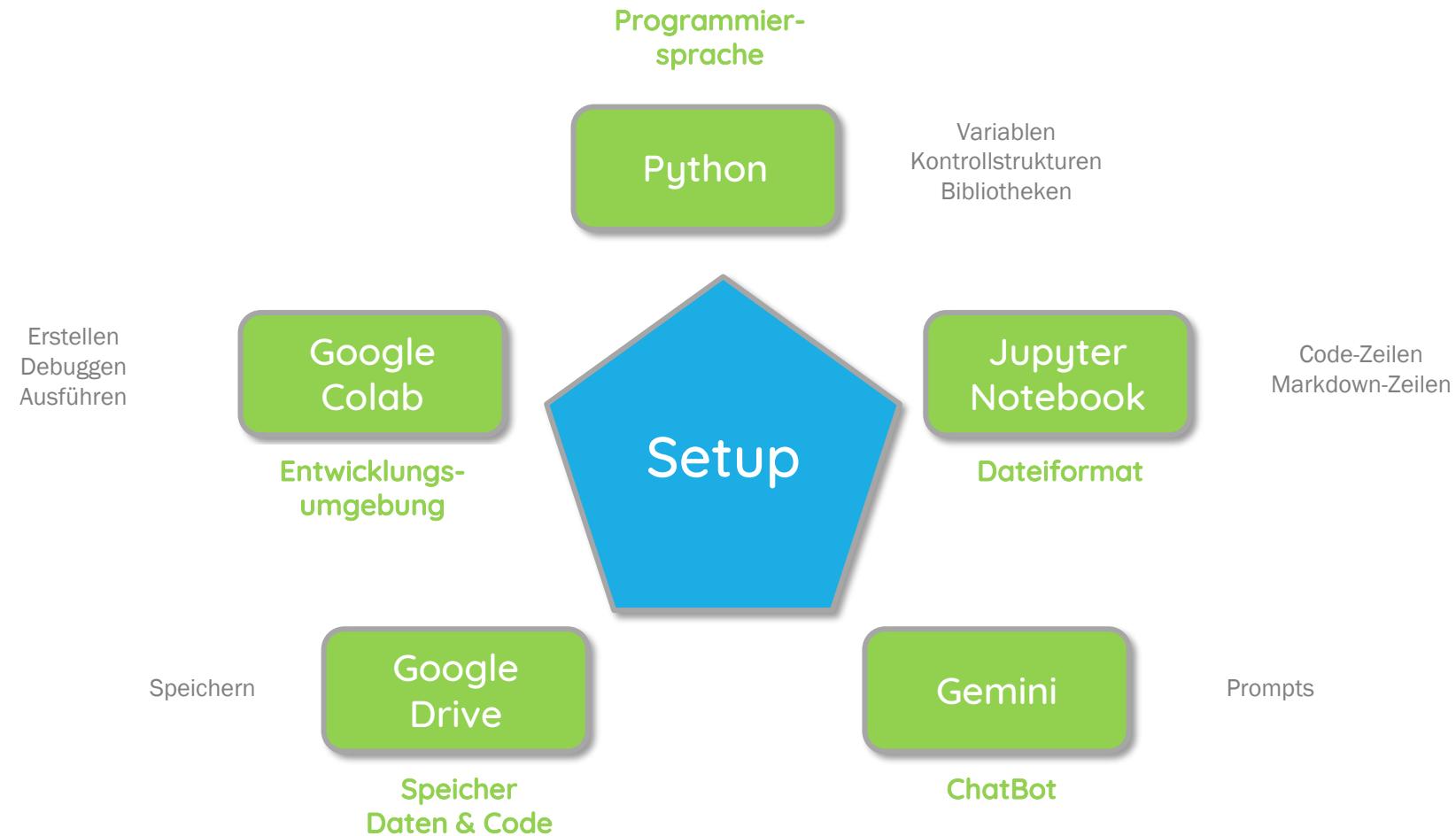
[bit.ly/3S06vX6](https://bit.ly/3S06vX6)





# COLAB & JUPYTER NOTEBOOKS

# IT-SETUP





## GOOGLE COLAB(ORATORY)

- Google Colaboratory, kurz Colab, ist eine **kostenlose Entwicklungsumgebung**, die vollständig in der Cloud arbeiten wird.
- In Colab können Jupyter-Notebooks erstellt, bearbeiten und ausgeführt werden.
- Colab unterstützt viele beliebte **Machine-Learning**-Module, die einfach in ein Notebook geladen werden können.
- Colab erlaubt es unterschiedliche Laufzeitumgebungen zu definieren in denen man neben einer CPU auch **GPUs** und **TPUs** verwenden kann.
- Colab hat mit **Gemini** eine integrierte GenKI für Coding.

CPU = Central Processing Unit, GPU = Graphics Processing Unit, TPU = Tensor Processing Unit

# EXKURS: CPU/GPU/TPU

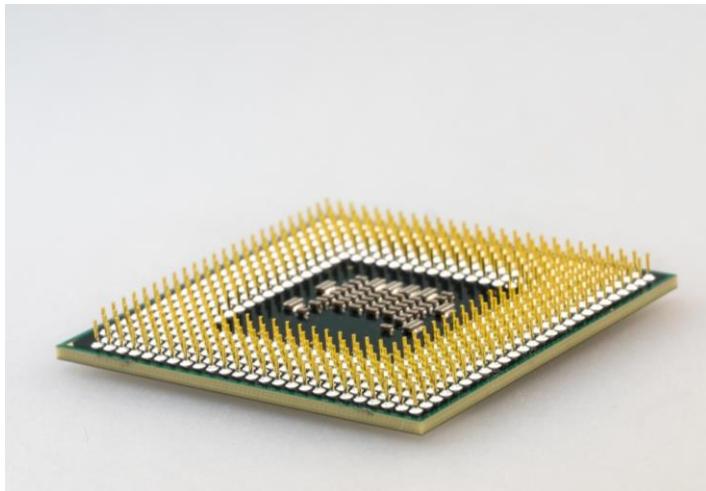


Bild von [Michael Schwarzenberger](#) auf [Pixabay](#)

- CPU (Central Processing Unit) ist ein **Allzweckprozessor**, der die meisten Verarbeitungsaufgaben in einem Computer ausführt. Es kann eine Vielzahl von Funktionen verarbeiten, ist jedoch nicht für bestimmte Aufgaben wie Grafikverarbeitung oder Matrizenberechnungen optimiert.
- GPU (Graphics Processing Unit) wurde speziell für die **Grafik- und Videoverarbeitung** entwickelt. Es verfügt über viele kleine Kerne, die für die Parallelverarbeitung optimiert sind, wodurch es für Aufgaben wie das Rendern von 3D-Bildern, das Abspielen von Videos und das Ausführen komplexer Simulationen viel schneller als eine CPU ist.
- TPU (Tensor Processing Unit) ist ein benutzerdefinierter Chip, der speziell für Aufgaben des **maschinellen Lernens** entwickelt wurde, insbesondere mit dem TensorFlow-Framework. Es ist für Matrixberechnungen und andere Operationen optimiert, die häufig im Deep Learning verwendet werden, wodurch es viel schneller und effizienter für das Training großer neuronaler Netze ist.
- Zusammenfassend lässt sich sagen, dass die CPU ein Allzweckprozessor ist, die GPU auf die Grafikverarbeitung spezialisiert ist und die TPU auf maschinelles Lernen spezialisiert ist.

# JUPYTER



- Die Jupyter App ist eine **Entwicklungsumgebung**, die das Bearbeiten und Ausführen von Programmiersprachen, u.a. Python, über einen Webbrowser ermöglicht.
- Der Name **Jupyter** bezieht sich auf die drei wesentlichen Programmiersprachen **Julia**, **Python** und **R** und ist auch eine Hommage an Galileos Notizbucheinträge zur Entdeckung der Jupitermonde.
- Mit der Jupyter App kann man **Notizbücher** erstellen. Diese Notizbücher (Dateiendung `.ipynb`) enthalten:
  - **Programmcode**, der ausgeführt werden kann,
  - **Markdown-Zeilen**, das sind Textzeilen mit Formatierungsangaben.
  - Die Jupyter-Notebooks werden v.a. für interaktive wissenschaftliche Analysen und Berechnungen, z.B. Data Analytics und Machine Learning, verwendet.

# JUPYTER-NOTEBOOKS: EIN INTERAKTIVES TOOL



Bild mit ideogram.ai erstellt

Inhalte und Struktur der Notebooks:

- **Grundlagen** und **Vertiefung**: Jedes Thema wird durch detaillierte Erläuterungen eingeführt, gefolgt von Code-Beispielen.
- Praktische **Übungen**: Zu den meisten Themen gibt es Übungen, die in Schwierigkeitsgraden gestaffelt sind. Dies ermöglicht eine individuelle Anpassung an das eigene Lerntempo.
- Lösungen und Hinweise: Für jede Übung steht ein **Lösungsvorschlag** bereit. Zusätzliche **Lösungshinweise** unterstützen beim eigenständigen Lösen der Aufgaben.
- Präsentation und Diskussion: Die erarbeiteten Lösungen können innerhalb der Lerngruppe präsentiert und diskutiert werden, was den Lernprozess fördert und vertieft.



# GOOGLE COLAB - EINSTIEG

The screenshot shows the Google Colab interface in a web browser. The main area displays a welcome message and links to test the Gemini API. Below this, there's a video thumbnail for a tutorial on cool features. The sidebar on the left lists various notebooks and sections like 'Erste Schritte' and 'Data Science'. The bottom of the screen shows a code cell with some initial text.

Willkommen bei Colab!

(Neu) Gemini API testen

- [Generate a Gemini API key](#)
- [Talk to Gemini with the Speech-to-Text API](#)
- [Gemini API: Quickstart with Python](#)
- [Gemini API code sample](#)
- [Compare Gemini with ChatGPT](#)
- [More notebooks](#)

Wenn Sie Colab schon kennen, erfahren Sie in diesem Video mehr zu interaktiven Tabellen, zur Verlaufsansicht für ausgeführten Code und zur Befehlspalette.

3 Cool Google Colab Features

Was ist Colab?

Mit Colab oder „Colaboratory“ können Sie Python-Code in Ihrem Browser schreiben und ausführen. Sie können Folgendes tun:

- Keine Konfiguration erforderlich
- Kostenloser Zugriff auf GPUs
- Einfache Freigabe

Egal, ob Sie **Student**, **Data Scientist** oder **AI-Forscher** sind – Colab erleichtert Ihnen die Arbeit. Im Video [Einführung in Colab](#) erhalten Sie weitere

## GEMINI IN COLAB



### ■ Code Zeile



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

- Code-Generierung direkt im Jupyter Notebook
- Code-Vervollständigung

### ■ Gemini Register

◆ Gemini

- Code optimieren
- Code kommentieren/dokumentieren
- Code erklären/erläutern
- allgemeine Prompts

# TASTATUREINSTELLUNGEN COLAB

## Tastatureinstellungen

Editor-Tastaturbelegungen  
default  
 Mit der Eingabetaste werden Vorschläge akzeptiert

### Tastenkombinationen

Wenn Sie eine Tastenkombination hinzufügen oder ändern möchten, klicken Sie auf die entsprechende Tastenkombination und geben Sie dann die neue Kombination ein. Beachten Sie, dass Ctrl+M als Präfix für Tastenkombinationen mit mehreren Tasten verwendet werden kann.

Tastenkombination ipynb herunterladen

### Standardeinstellungen wiederherstellen

Ctrl+M . Laufzeit neu starten

Tastenkombination py herunterladen

Aktuelle Zeile kommentieren

Tastenkombination Alle Ausgaben löschen

Tastenkombination Alle Laufzeiten auf Werkseinstellungen zurücksetzen

Ctrl+Shift+A Alle Zellen auswählen

Ctrl+F9 Alle Zellen in Notebook ausführen

Ctrl+H Alle suchen/ersetzen

Ctrl+[ Alle/ausgewählte Abschnitte maximieren

Ctrl+] Alle/ausgewählte Abschnitte minimieren

Ctrl+M - An Cursorposition teilen

Tastenkombination Ansicht mit einem Tab ansehen

Tastenkombination Auf die zuletzt ausgeführte Zelle fokussieren

Ctrl+M I Ausführung unterbrechen

Ctrl+M O Ausgabe einblenden/ausblenden

Tastenkombination Ausgabevollbild anzeigen

## Tastatureinstellungen

Tastenkombination Ausgewählte Ausgaben löschen

Ctrl+M K Ausgewählte Zellen nach oben verschieben

Ctrl+M J Ausgewählte Zellen nach unten verschieben

Ctrl+F10 Ausgewählte Zellen und alle folgenden Zellen ausführen

Tastenkombination Ausgewählte Zellen zusammenführen

Ctrl+Shift+Enter Auswahl ausführen

Shift+Down Auswahl erweitern, um nächste Zelle einzuschließen

Shift+Up Auswahl erweitern, um vorherige Zelle einzuschließen

Tastenkombination Automatische Ausführung der ersten Zelle oder des ersten Abschnitts bei jeder Ausführung aktivieren

Ctrl+Space oder Tab Automatische Vervollständigung

Ctrl+Shift+P Befehlspalette anzeigen

Tastenkombination Code ein-/ausblenden

Ctrl+Alt+P Code-Snippets-Bereich anzeigen

Tab Code-docstring-Hilfe aktivieren/deaktivieren

?

Tastenkombination Codezelle hinzufügen

Ctrl+M A Codezelle oben einfügen

Ctrl+M B Codezelle unten einfügen

Tastenkombination Colab Enterprise öffnen

Tastenkombination Notebook freigeben

Tastenkombination Notebook hochladen

Tastenkombination Notebook in Drive verschieben

Tastenkombination Notebook in Google Drive markieren/Markierung aufheben

Tastenkombination Notebook in Papierkorb verschieben

Ctrl+S Notebook speichern

Ctrl+O Notebook öffnen

Tastenkombination Notebook-Einstellungen öffnen

Tastenkombination Notebook-Quelle anzeigen

Shift+Up Auswahl erweitern, um vorherige Zelle einzuschließen

Tastenkombination Notebook-Vergleich

Ctrl+M N Nächste Zelle

Ctrl+Shift+] Nächsten Tab hervorheben

Ctrl+G Nächstes Element suchen

Tastenkombination Ressourcen ansehen

Ctrl+Alt+N Scratchpad-Codezelle öffnen

Tastenkombination Sarterleiste für Kommentare

Tastenkombination Sichtbarkeit des Headers aktivieren/deaktivieren

Tastenkombination Sitzungen verwalten

Tastenkombination Statusleiste einblenden/ausblenden

Tastenkombination Tab in den nächsten Bereich verschieben

Tastenkombination Tab in den vorherigen Bereich verschieben

Tastenkombination Tabs in zwei Spalten anzeigen

## Tastatureinstellungen

Tastenkombination Dateibrowser anzeigen

Tastenkombination Drive bereitstellen

Tastenkombination Drive trennen

Tastenkombination Editor-Einstellungen öffnen

Einstellungen öffnen

Shift+Tab Einzug für aktuelle Zeile entfernen

Tastenkombination Fokus mit Tabulatoraste verschieben

aktivieren/deaktivieren

Tastenkombination Formular hinzufügen

Ctrl+M F Formularansicht ändern

Tastenkombination Formularattribute bearbeiten

Tastenkombination Formularfeld hinzufügen

Tastenkombination Geplante Notebooks anzeigen

Ctrl+Enter Hervorgehobene Zelle ausführen

Ctrl+Shift+S Hervorgehobene Zelle auswählen

Tastenkombination Hervorgehobene Zelle mit nächster Zelle zusammenführen

Tastenkombination Hervorgehobene Zelle mit vorheriger Zelle zusammenführen

Esc Hervorheben der Zelle aufheben

Tastenkombination Im Playground-Modus öffnen

Tastenkombination In Google Drive suchen

Tastenkombination In Scratchpad-Zelle kopieren

Shift+Ctrl+H In aktueller Zelle alle ersetzen

Tastenkombination Informationen zu Notebook-Datei anzeigen

Tastenkombination Inhaltsverzeichnis anzeigen

Tastenkombination Tabs in zwei Zeilen anzeigen

Ctrl+M H Tastenkombinationen anzeigen

Tastenkombination Terminal anzeigen

Tastenkombination Textzeile hinzufügen

Tastenkombination Variablenübersicht anzeigen

Tastenkombination Verbindung mit gehosteter Laufzeit herstellen

Tastenkombination Verbindung trennen und Laufzeit löschen

Tastenkombination Formular hinzufügen

Ctrl+M P Vorherige Zelle

Ctrl+Shift+[ Vorherigen Tab hervorheben

Tastenkombination Geplante Notebooks anzeigen

Ctrl+Shift+G Vorheriges Element suchen

Tastenkombination Wieder verbinden

Ctrl+M L Zellennummern aktivieren/deaktivieren

Ctrl+Shift+Y Zellaktion wiederholen

Ctrl+Click Zellauswahl aktivieren/deaktivieren

Alt+Enter Zelle ausführen und neue Zelle einfügen

Shift+Enter Zelle ausführen und nächste Zelle auswählen

Tastenkombination Zelle in Tab spiegeln

Tastenkombination Zelle mit Abschnittsüberschrift hinzufügen

Tastenkombination Zelle oder Auswahl ausschneiden

## Tastatureinstellungen

Tastenkombination JSON-Notebook anzeigen

Ctrl+Alt+M Kommentar hinzufügen

Tastenkombination Kommentarbereich öffnen

Tastenkombination Kopie als GitHub Gist speichern

Tastenkombination Kopie in Drive speichern

Tastenkombination Kopie in GitHub speichern

Tastenkombination Zu Codezelle konvertieren

Ctrl+M M Zu Textzelle konvertieren

Tastenkombination Zu einer konkreten Zelle

Ctrl+M S Überarbeitung speichern und anpinnen

Tastenkombination Überarbeitungsverlauf anzeigen

# PYTHON GRUNDLAGEN

# BASICS

# WARUM BRAUCHT MAN CODE?

“

„Der Grund, warum wir Code schreiben und keine natürliche Sprache, ist, dass natürliche Sprache mehrdeutig ist. Diese Mehrdeutigkeit kann man nicht wirklich umgehen.“

**Ada Morse**

Codecademy Curriculum Developer in Data Science

”

Quelle: [Can ChatGPT Teach Me How To Code Better Than Courses? \(codecademy.com\)](#)



## PYTHON – EINE VON VIELEN

- Python ist eine **universelle, höhere** Programmiersprache.
- Die Sprache wurde Anfang der 1990er Jahre von Guido van Rossum entwickelt.
- Sie fördert einen **gut lesbaren, knappen** Programmierstil.
- Python unterstützt mehrere Programmierparadigmen, z. B. die objektorientierte und funktionale Programmierung.
- Zu Beginn der Entwicklung/Implementierung von Python begann, las Guido van Rossum die veröffentlichten Drehbücher aus "**Monty Python's Flying Circus**", einer BBC-Comedy-Serie aus den 1970er Jahren.
- Van Rossum dachte, er brauche einen Namen, der kurz, einzigartig und leicht mysteriös sei, also beschloss er, die Sprache Python zu nennen.

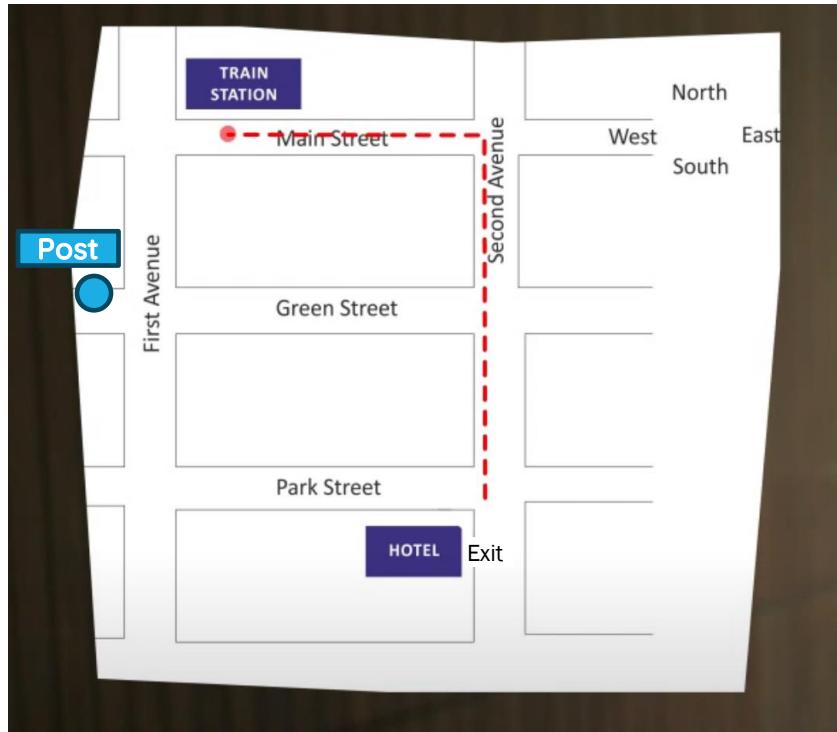
# MERKMALE VON PYTHON

Python ist eine **Skriptsprache**, die interpretiert wird. Das bedeutet, dass der Code Zeile für Zeile von einem Interpreter ausgeführt wird, anstatt vorher komplett in Maschinencode übersetzt zu werden.

Merkmale sind:

- implizit deklarierte Variablen (erfolgt durch Wertzuweisung),
- dynamische Typisierung (Typisierung erfolgt zum Zeitpunkt der Nutzung),
- automatische Speicherverwaltung (Belegung und Freigabe von Arbeitsspeicher),
- unmittelbare Ausführung durch Interpretation des Quelltextes ohne getrennte Übersetzungsphase.

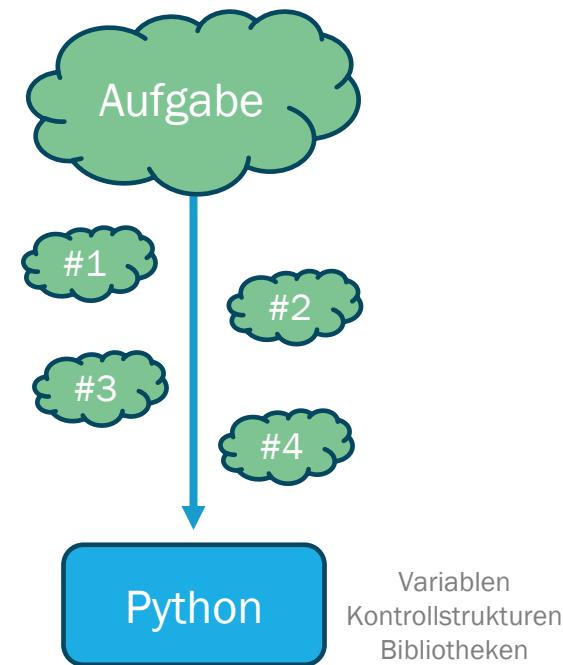
# EINFACHE ABLÄUFE / SEQUENZEN



- Ein Python Programm besteht aus mehreren Anweisungen.
- Die einzelnen Anweisungen repräsentieren die Schritte vom Start zum Ziel.
- Die Anweisungen werden im einfachsten Fall sequenziell (nacheinander) ausgeführt.
- Einfache Programme werden häufig nach dem EVA-Prinzip gestaltet (Eingabe – Verarbeitung – Ausgabe).

Quelle: Abgefragt 30.08.2021 [1.1.2 - CODE YOURSELF - Representation of Algorithms - YouTube](#)

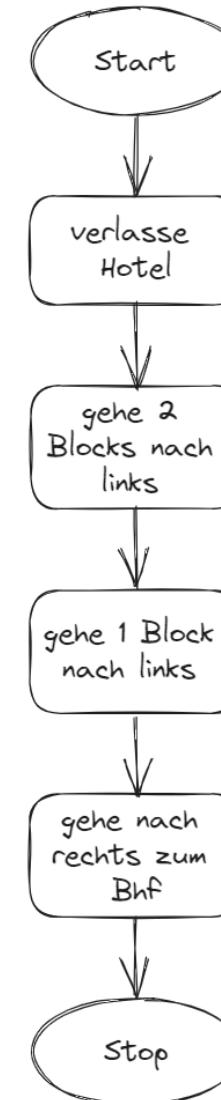
# DER 1. SCHRITT: PROBLEMZERLEGUNG



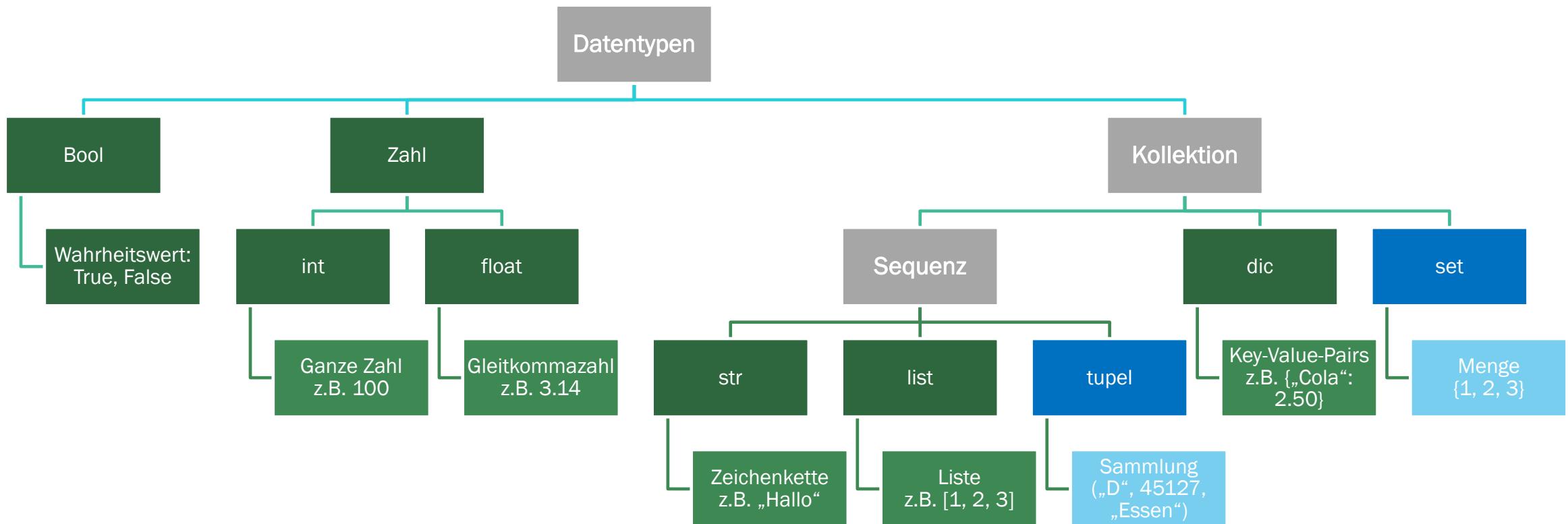
# PROBLEMZERLEGUNG – PROGRAMMABLAUFPLAN

- Ein Programmablaufplan (PAP) ist ein Ablaufdiagramm für ein Computerprogramm.
- Es ist eine grafische Darstellung und beschreibt die Folge von Aktionen bzw. Operationen zur Lösung einer Aufgabe.
- Über Symbole wird der Programmablauf bzw. der Datenfluss dargestellt.

Beispiel: Weg zum Bahnhof



# WESENTLICHE DATENTYPEN

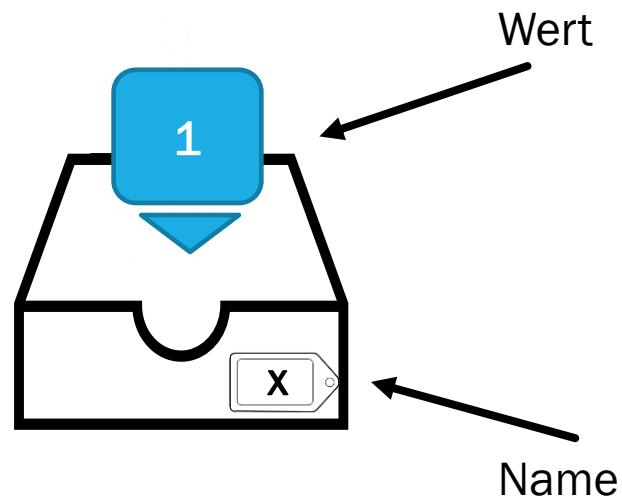


# RECHEN- UND STRINGOPERATOREN

Symbol	Rechenoperatoren
+ -	Vorzeichen
+ - * /	Grundrechenarten
//	Ganzzahlige Division
%	Rest der ganzzahligen Division
**	Exponentialfunktion
=	Zuweisung
+= oder -=	Zuweisen & Addition oder Subtraktion
*= oder /=	Zuweisen & Multiplikation o. Division

Symbol	String Operatoren
+	String verbinden
*	String vervielfachen

# ZUWEISUNGEN - VARIABLEN



- Zuweisungen sind die häufigsten Befehle in einem Programm.
- Die einfachste Form besteht aus einem Zuweisen von einem Wert zu einem Namen  
$$\text{Name} \leftarrow \text{Wert}$$

Beispiel: `x = 1`
- Eine Variable ist ein „Behälter“, in dem man einen Wert aufbewahrt (speichert).
- Über dem Namen der Variablen kann man auf den Wert zugreifen.

# FUNKTIONEN

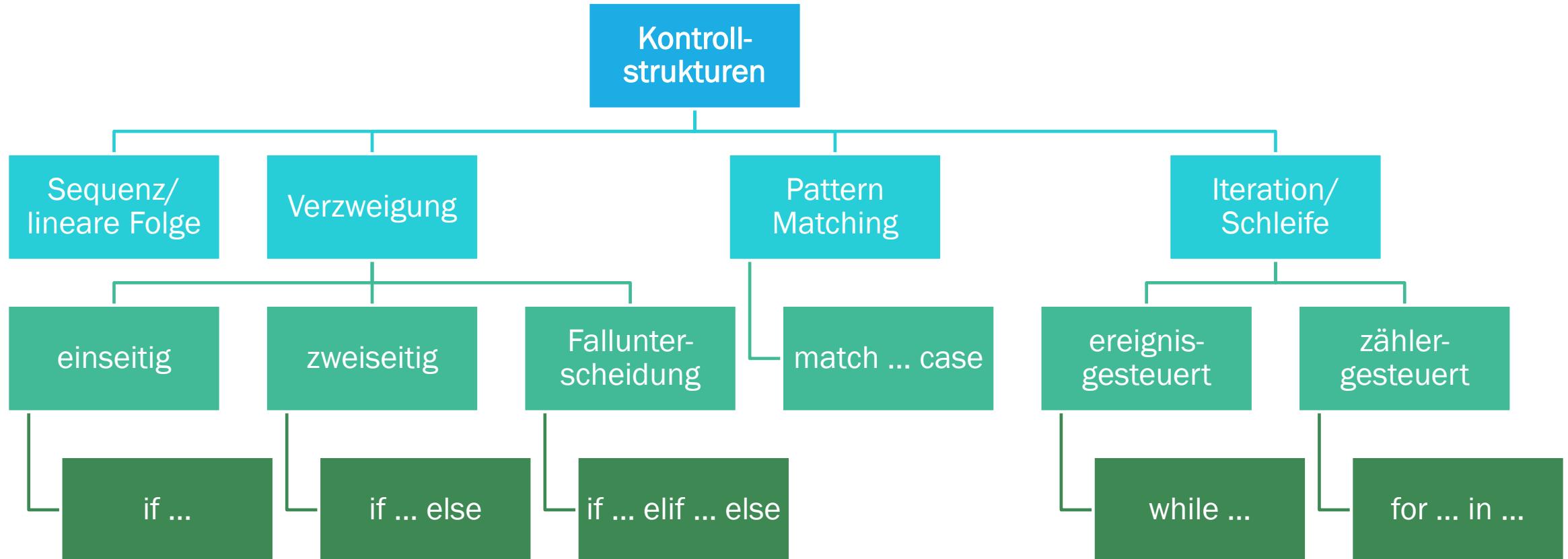
- Funktionen sind neben den Datentypen, Operatoren und Kontrollstrukturen die Bausteine einer jeden Programmiersprache.
- Python stellt vordefinierte Funktionen/**Standardfunktionen** zur Verfügung.
- Eine Funktion liefert ein vordefiniertes Ergebnis, sie löst eine Teilaufgabe in einem Programm.
- Sie werden für einfache und komplexe Aufgaben eingesetzt.
- Beispiele:

```
In [1]: 1 print("Hello Python")  
Hello Python
```

```
In [2]: 1 len("Hello Python")  
Out[2]: 12
```

genereller Aufbau:  
**Funktionsname(Argumente)**

# KONTROLLSTRUKTUREN



# VERGLEICHS- UND LOGIKOPERATOREN

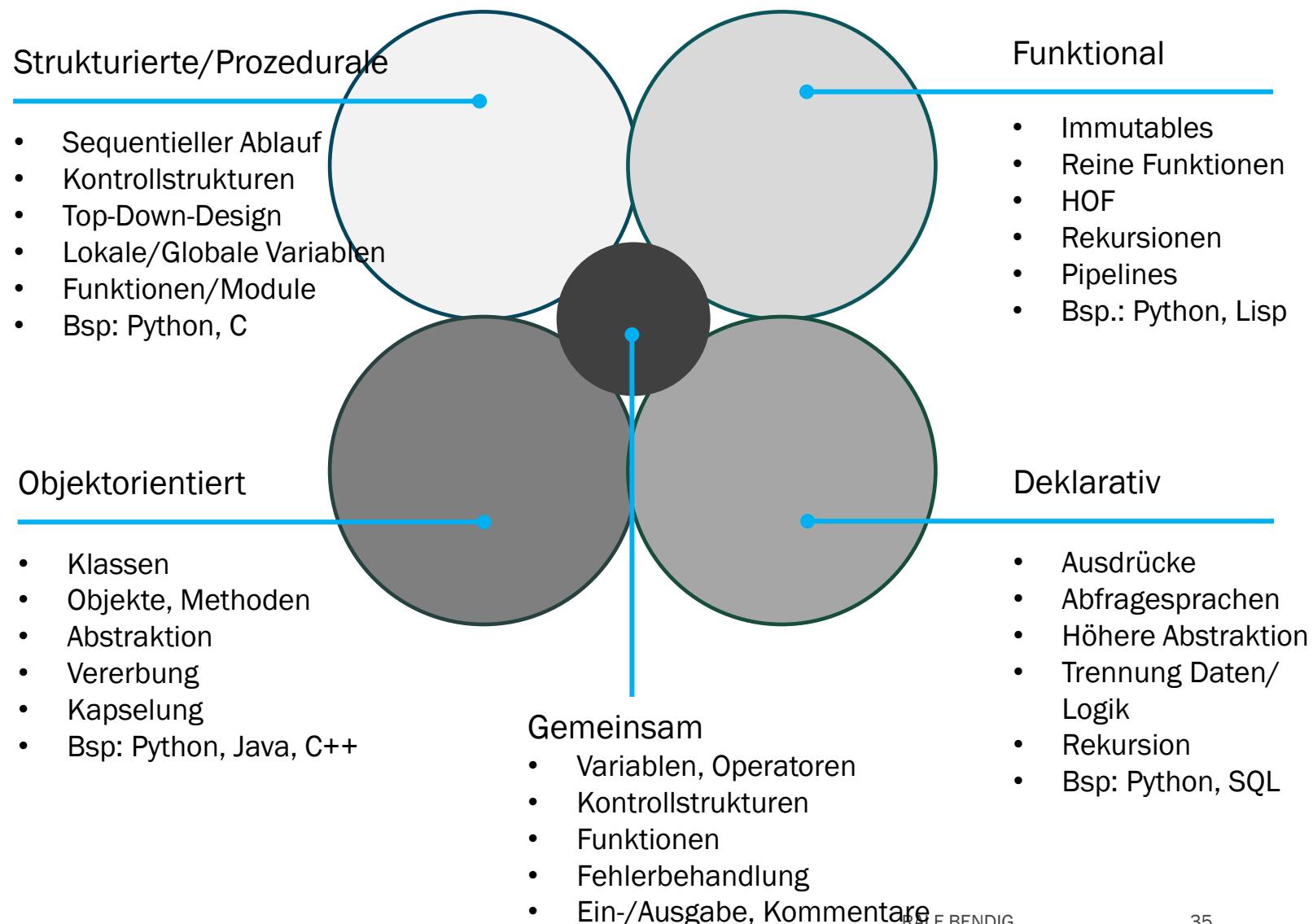
Operator	Vergleichsoperatoren
<code>==</code>	Gleichheit testen
<code>!=</code>	Ungleichheit testen
<code>&lt; &gt;</code>	kleiner, größer
<code>&lt;= &gt;=</code>	kleiner-gleich, größer-gleich
<code>in</code>	testen, ob in Aufzählung enthalten

Operator	Logikoperatoren
<code>or</code>	logisches Oder
<code>and</code>	logisches Und
<code>not</code>	logisches Nicht

# OBJEKTORIENTIERTE PROGRAMMIERUNG (OOP)

# PROGRAMMIER- PARADIGMEN

Programmierparadigmen sind **grundlegende Stilrichtungen** oder Philosophien, die die Art und Weise definieren, wie in der Softwareentwicklung **Probleme strukturiert und gelöst** werden. Jedes Paradigma bietet einen eigenen Rahmen für das Verständnis, wie die Bausteine eines Programms interagieren und wie die Programmlogik konzipiert wird. Python ist eine Multi-Paradigmensprache



# OOP - GRUNDIDEE

- Die objektorientierte Programmierung (kurz OOP) ist ein **Programmierparadigma** (grundlegende Denkweise).
- Die Grundidee besteht darin, die Architektur einer Software an den **Grundstrukturen** desjenigen Bereichs der **Wirklichkeit** auszurichten, der die gegebene Anwendung betrifft.
- Ein **Modell** dieser Strukturen wird in der Entwurfsphase aufgestellt. Es enthält Informationen über die auftretenden Objekte und deren Verallgemeinerungen.
- Die Umsetzung dieser Denkweise erfordert die Einführung verschiedener Konzepte, insbesondere **Klassen, Methoden und Vererbung**.

# GRUNDBEGRIFFE OOP

**Methode** – sind  
Fähigkeiten der Objekte.  
(~Funktion)

## Methoden:

- Fahren
- Parken
- Tanken
- ...



**Objekte**: BMW X1, : Audi A3, ...

**Objekt/Instanz** –  
konkrete Ausprägung  
(~ konkrete Variable)

**Klasse** – legt die  
prinzipielle Gestalt  
(Attribute) und  
Fähigkeiten (Methoden)  
der Objekte fest.  
(~Datentyp, ~Bauplan)

**Klasse**: Auto

## Attribute:

- Hersteller
- Modell
- Leistung
- Baujahr
- ...

**Attribut** – kennzeichnet die  
Eigenschaften und somit die  
Unterschiede zwischen den  
Objekten.  
(~Merkmale)

# KERNKONZEPTE OOP

## Kernkonzepte

- Abstraktion
- Vererbung
- Kapselung
- Polymorphie

- **Abstraktion:** Die Abstraktion ist der Prozess, bei dem komplexe Systeme oder Prozesse auf ihre **wesentlichen Merkmale reduziert** werden. In der objektorientierten Programmierung werden Klassen verwendet, um Abstraktionen zu erstellen, die die gemeinsamen Merkmale und Verhaltensweisen von Objekten darstellen.
- **Vererbung:** Vererbung ist ein Konzept, das es ermöglicht, eine neue Klasse auf der Grundlage einer vorhandenen Klasse zu erstellen. Die neue Klasse **erbt die Eigenschaften** und Methoden der vorhandenen Klasse und kann diese nach Belieben erweitern oder ändern. Dies ermöglicht es, effektiveren und wiederverwendbaren Code zu schreiben.
- **Kapselung:** Die Kapselung bezieht sich auf die Idee, dass Objekte bestimmte **Informationen** vor der Außenwelt **verbergen** können und nur ausgewählte Methoden und Eigenschaften für den Zugriff durch andere Objekte freigeben. Durch Kapselung wird die Interaktion mit Objekten auf eine definierte und kontrollierte Weise durchgeführt.
- **Polymorphie:** Polymorphie ermöglicht es, dass ein Objekt **unterschiedliche Formen** oder **Verhaltensweisen** annimmt, basierend auf dem Kontext, in dem es verwendet wird. Das bedeutet, dass ein Objekt in der Lage ist, verschiedene Methoden oder Eigenschaften bereitzustellen, je nachdem, wie es verwendet wird.

## \_\_INIT\_\_



```
1 class Auto:  
2     def __init__(self, hersteller, modell, leistung):  
3         self.hersteller = hersteller  
4         self.modell = modell  
5         self.leistung = leistung  
6  
7     def __str__(self):  
8         return f'{self.hersteller}, {self.modell}, {self.leistung}'
```

- `__init__` eine spezielle Methode in objektorientierten Programmiersprachen, die automatisch aufgerufen wird, wenn ein Objekt/Instanz einer Klasse erstellt wird.
- Der Hauptzweck besteht darin, die Anfangswerte für die Attribute des Objekts festzulegen und jegliche erforderliche Initialisierungsaktivitäten durchzuführen.
- Es liefert damit den Bauplan, nach dem das Objekt bzw. die Instanz erstellt wird.

# STR



```
1 class Auto:  
2     def __init__(self, hersteller, modell, leistung):  
3         self.hersteller = hersteller  
4         self.modell = modell  
5         self.leistung = leistung  
6  
7     def __str__(self):  
8         return f"{self.hersteller}, {self.modell}, {self.leistung}"
```

- Die `__str__`-Methode in Python ist eine spezielle Methode, die verwendet wird, um einen lesbaren String einer Instanz einer Klasse zu erstellen.
- Wenn eine Instanz einer Klasse in Python als String gedruckt wird, ruft Python automatisch die `__str__`-Methode auf, um den String der Instanz zu erzeugen.
- Ohne `__str__`-Methode erfolgt die Ausgabe einer Referenz: `<__main__.Person object at 0x7f946bbf9dc0>`

# SELF



```
1 class Auto:  
2     def __init__(self, hersteller, modell, leistung):  
3         self.hersteller = hersteller  
4         self.modell = modell  
5         self.leistung = leistung  
6  
7     def __str__(self):  
8         return f"{self.hersteller}, {self.modell}, {self.leistung}"
```



- **self** ist ein Verweis auf das aktuelle Objekt/Instanz und wird verwendet, um auf Attribute oder Methoden dieses Objekts zuzugreifen.
- Es ist das erste Argument, das jeder Methode in einer Klasse in Python übergeben wird, einschließlich `__init__`.
- Es ermöglicht, zwischen Klassenattributen (die für alle Objekte der Klasse gemeinsam sind) und Instanzattributen (die für jedes Objekt einzigartig sind) zu unterscheiden.
- Ganz einfach: Mit der Klasse, die oben definiert wird, wird der allgemeine Bauplan für ein Objekt vorgegeben. Von dieser Klasse können wir aber beliebig viele Objekte/Instanzen erzeugen.
- Beim Aufruf der `init`-Methode wird die Referenz auf das aktuelle Objekt/Instanz mitübergeben.

# FUNKTIONEN UND METHODEN

- Funktionen können direkt über ihren Namen aufgerufen werden – Methoden benötigen zusätzlich immer ihr Objekt.
- Die Schreibweise:
  - *Funktionen:* *funktionsname()*
  - *Methoden:* *objekt.methode()*
- **Funktionen:** Unabhängige Codeblöcke, die eine bestimmte Aufgabe ausführen, Parameter akzeptieren und optional einen Wert zurückgeben können.
- **Methoden:** Funktionen, die **innerhalb** einer Klasse definiert sind. Sie können, genau wie Funktionen, Werte zurückgeben und den Zustand eines Objekts verändern oder einfach Informationen darüber abrufen.

Beispiel:

```
[ ] 1 liste = ["apple", "banana", "kiwi", "grapefruit"]
```

```
[ ] 1 # Funktion
2 liste_sortiert = sorted(liste, key=len)
3 liste, liste_sortiert
```

```
(['apple', 'banana', 'kiwi', 'grapefruit'],
 ['kiwi', 'apple', 'banana', 'grapefruit'])
```

```
[ ] 1 # Methode
2 liste.sort(key=len)
3 liste
```

```
['kiwi', 'apple', 'banana', 'grapefruit']
```

# VERGLEICH FUNKTION VS METHODE

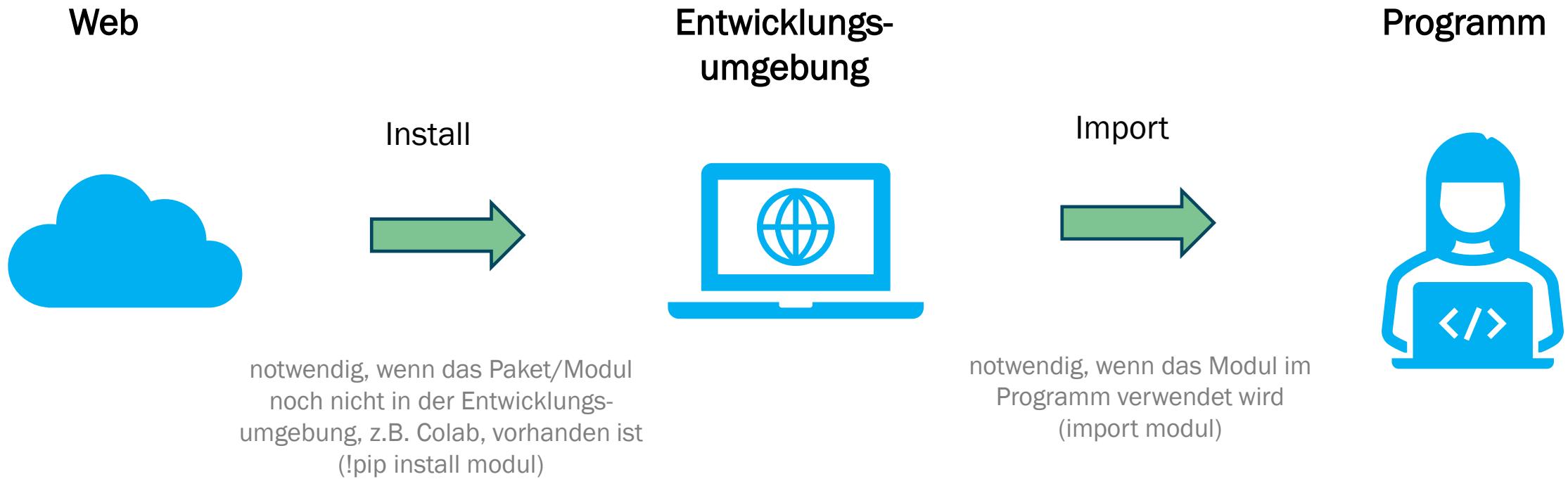
Eigenschaften	Funktion	Methode
Definition	Ein unabhängiger Codeblock, der eine spezifische Aktion ausführt.	Eine Funktion, die zu einer Klasse gehört und auf deren Attribute zugreifen kann.
Aufruf	Unabhängig von einem Objekt.	Immer bezogen auf ein Objekt oder eine Klasse.
Erster Parameter	Keine festgelegten Konventionen.	Für Instanzmethoden ist es das Objekt selbst (self).
Definitionsort	Kann überall im Code definiert werden.	Wird innerhalb einer Klasse definiert.
Zugriff auf Objekt-Attribute	Hat keinen impliziten Zugriff auf das umgebende Objekt.	Hat Zugriff auf das Objekt und seine Attribute.

# PAKETE & MODULE

# MODUL

- In Python ist ein Modul eine Datei, die Funktionen, Klassen und Variablen enthält.
- **Module** werden verwendet, um den Code in wiederverwendbare Teile aufzuteilen, was dazu beitragen kann, den Code effizienter zu gestalten und ihn einfacher zu warten.
- Ein Modul kann über das **Import-Statement** in einem anderen Python-Skript oder Modul geladen werden, um auf dessen Funktionen, Klassen und Variablen zuzugreifen.
- Das Import-Statement wird in der Regel am Anfang des Skripts oder Moduls verwendet.
- Python verfügt über eine Vielzahl von Standardmodulen, die für eine Vielzahl von Aufgaben verwendet werden können.
- Es gibt auch eine große Anzahl von Drittanbietermodulen, die von der Python-Community entwickelt wurden und für spezielle Aufgaben oder Anwendungen nützlich sein können.

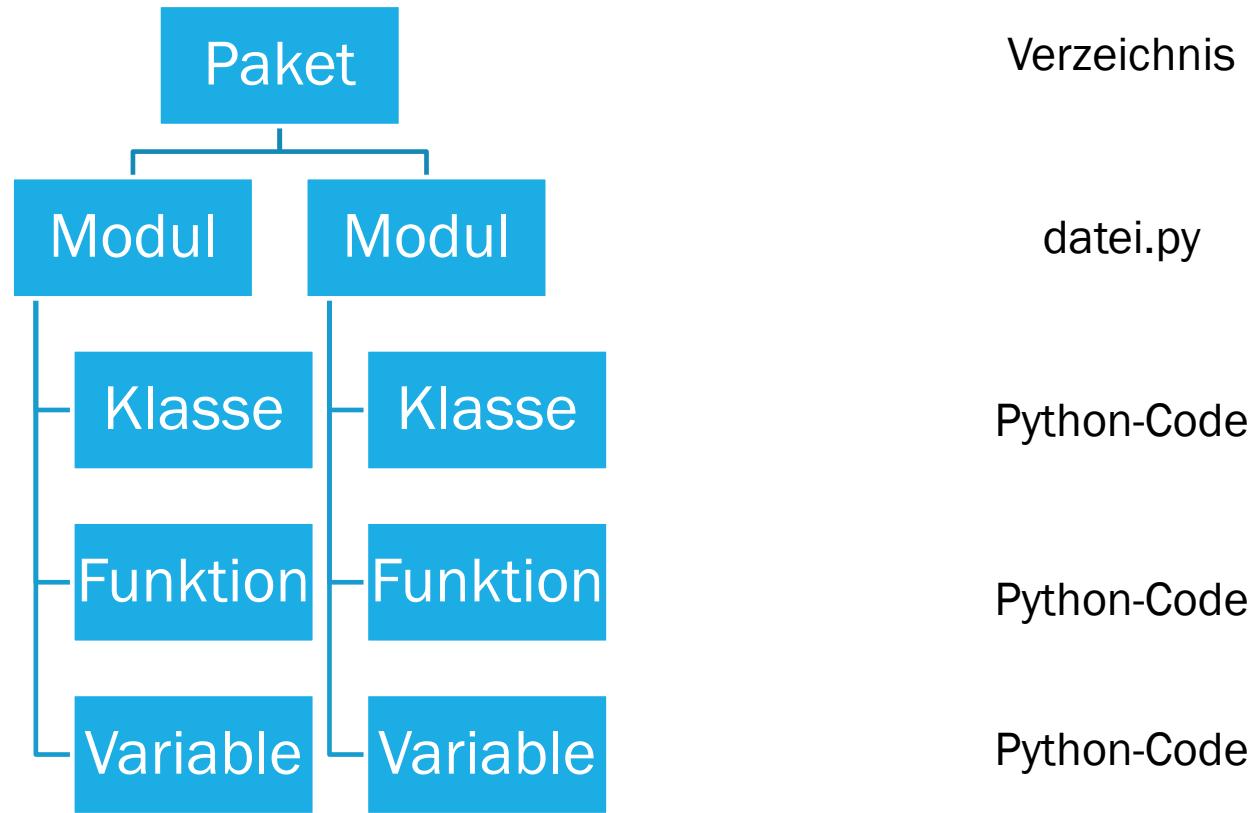
# INSTALL & IMPORT



# PYTHON PACKAGE INDEX (PYPI)

- Bei der Verwendung von `pip install` zum Installieren von Python-Bibliotheken werden die Pakete standardmäßig von Python Package Index (PyPI) heruntergeladen. PyPI ist eine **Software-Repository** für Python, die eine umfangreiche Sammlung von **Open-Source-Paketen** für die Python-Programmiersprache bereitstellt.
- PyPI agiert als zentrale Anlaufstelle für Python-Entwickler, um ihre Software zu teilen, damit andere sie leicht finden und installieren können. Wenn `pip install <paketname>` ausgeführt wird, sucht pip nach dem Paketnamen in PyPI, lädt das Paket und seine Abhängigkeiten herunter und installiert sie dann in der Python-Umgebung.
- Man kann auch Pakete von anderen Quellen als PyPI installieren, indem man die URL der Paketdatei oder eines Git-Repositories angibt. Zum Beispiel:
  - Um ein Paket direkt von einer URL zu installieren: `pip install <Paket-URL>` verwenden.
  - Um ein Paket von einem Git-Repository zu installieren: `pip install git+<Repository-URL>`.
  - Es ist auch möglich, pip so zu konfigurieren, dass es Pakete von einem anderen Index als PyPI sucht, indem man die Option `--index-url` verwendet.
- Zur Qualität von Bibliotheken sie u.a. [How to Evaluate the Quality of Python Packages – Real Python](#)

# PAKETE & MODULE



# CODING & KI

# BASICS

# CHATGPT – EINER VON VIELEN



- ChatGPT ist ein von OpenAI entwickeltes künstliches Intelligenzmodell.
- Es ist ein **Large Language Model** (LLM), dass darauf trainiert wurde, menschliche Sprache zu verstehen und zu generieren.
- Es errechnet die **Wahrscheinlichkeit** der nächsten **Wortsequenz**, um menschenähnliche Inhalte zu imitieren.
- Dieses Modell wurde mit **Milliarden** von Wörtern trainiert & kann Texte generieren, Fragen beantworten und viele Aufgaben durchführen, die mit Sprache zu tun haben.
- Es nutzt Deep Learning und speziell entworfene neuronale Netze, um **menschenähnliche Textantworten** zu produzieren.
- ChatGPT basiert auf der **GPT** (Generative Pre-trained Transformer) **Architektur**, die eine spezielle Art von Transformer-Modell ist, und ist darauf ausgelegt, Sprache in einem gegebenen Kontext zu simulieren.

Gilt analog  
für Gemini

# CHATGPT



Wortsequenz-  
wahrscheinlichkeits-  
rechenmaschine

Version	Jahr	Anzahl Parameter
GPT-1	2018	117 Millionen
GPT-2	2019	1,5 Milliarden
GPT-3	2020	175 Milliarden
GPT-4	2023	100 Billionen (geschätzt)

Bild mit DALL-E erstellt

FT  
FINANCIAL  
TIMES

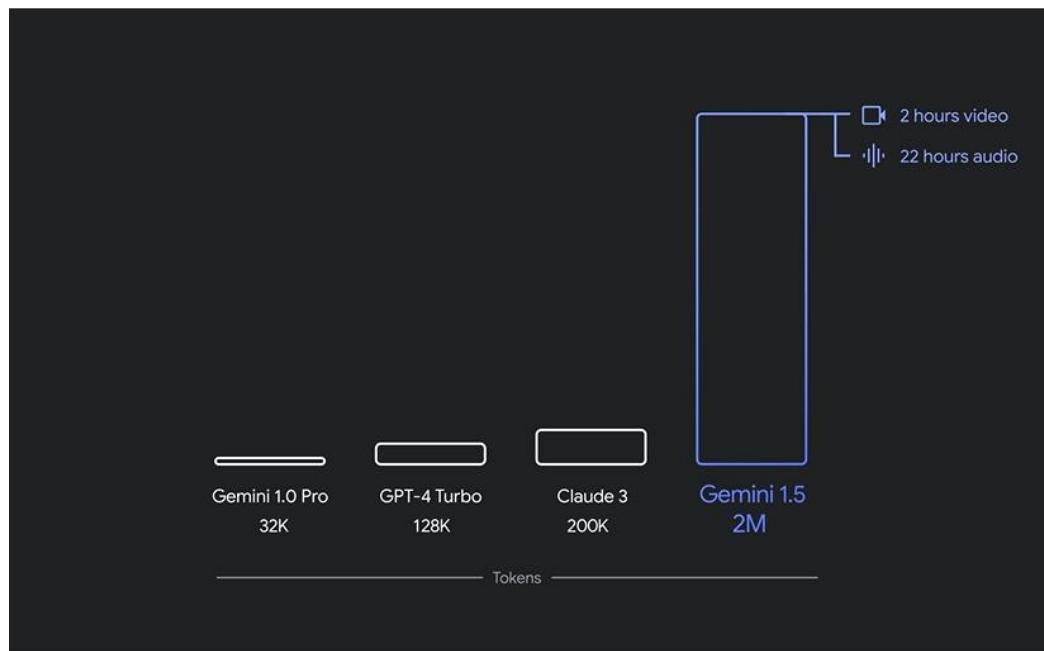


Bild: [Gemini: schnelleres Modell, größeres Kontextfenster und KI-Agenten \(blog.google\)](#)

- Gemini ist ein von Google AI entwickeltes KI-Sprachmodell.
- Es basiert auf einer innovativen neuen Architektur und bietet eine Reihe von Funktionen, die es von anderen Modellen abheben:
- Effizient: Gemini kann dank seiner einzigartigen Architektur auf energieeffiziente Weise lernen und arbeiten.
- Flexibel: Es kann eine große Vielfalt an Aufgaben bewältigen, von der Textgenerierung über die Codegenerierung bis hin zur Beantwortung von Fragen.
- Skalierbar: Die Architektur von Gemini ermöglicht es, es auf einer breiten Palette von Geräten einzusetzen, von Smartphones bis hin zu Rechenzentren.

# ROLLE DES MENSCHEN: INTERAKTOR



Bild von [Peggy und Marco Lachmann-Anke](#) auf [Pixabay](#)

Rollen des Menschen als Interaktor ...

- Fragensteller(in)
- Lernende/Lernender
- Lehrer(in) und Korrektor(in)
- Bewerter(in)
- Ideen- und Anregungsgeber(in)
- Empfänger(in) von Unterstützung
- Prüfer(in)
- ...

Interaktor →  
wechselseitige  
Beeinflussung

# PROMPT-STRUKTUR

Baustein	Beschreibung	Beispiel
Instruktion	Rolle ChatBot	Name und Rollenbeschreibung
	Rolle Nutzer	Name und Rollenbeschreibung
	Anweisungen	Allg. Anweisungen, Rahmenvorgaben
	Kontext	Information zur Situation
	Tools	Vom ChatBot zu verwendenden Tools
Aufgabe	Aufgabe	Frage/Beschreibung der Aufgabe
	Ergebnis	Was soll das Ergebnis sein
	Tonalität	Stimmung oder Charakter der Antwort
	Format	Formate des Ergebnisses
	Umfang	Umfang des Ergebnisses

# KI VERÄNDERT CODING

# KI FÜR CODING?

---

“

„Der Grund, warum wir Code schreiben und keine natürliche Sprache, ist, dass natürliche Sprache mehrdeutig ist. Diese Mehrdeutigkeit kann man nicht wirklich umgehen.“

**Ada Morse**

Codecademy Curriculum Developer in Data Science

---

”

Quelle: [Can ChatGPT Teach Me How To Code Better Than Courses? \(codecademy.com\)](https://www.codecademy.com/)

# AS TIME GOES BY ...



Quelle: [Papst-Momente: Bilder zeigen Vergleich zwischen 2005 und 2013 - DER SPIEGEL](#)

# PRESSE

Künstliche Intelligenz

## So werden ChatGPT-Nutzer auch zu Programmierern

Auf der neuen GPT-Plattform können Nutzer eigene Apps bauen, ohne eine Zeile Code zu schreiben. Das bringt das Geschäftsmodell vieler neuer KI-Start-ups in Gefahr.

Lina Knees  
27.11.2023 - 09:33 Uhr



Mit dem GPT Builder kann jeder eigene KI-Apps erstellen ohne zu Programmieren. (KI-generiertes Bild mit Dall-E)

Künstliche Intelligenz

## Läutet KI das Ende des Programmierens ein?

Künstliche Intelligenz macht Programmierer schneller und besser. Software wird so zum Allgemeingut. Auch kleine Unternehmen oder Mittelständler könnten bald mit Tech-Giganten mithalten.

Stephan Scheuer  
13.11.2023 - 10:48 Uhr



Der Einfluss von KI auf das Programmieren könnte die Gewichte im Technologiesektor verschieben. Foto: Getty Images

KÜNSTLICHE INTELLIGENZ

## KI wird Ihnen Ihren Programmierjob nicht wegnehmen (aber sie wird ihn verändern)

Die KI-Codierungsrevolution: Wird Devin alles verändern?

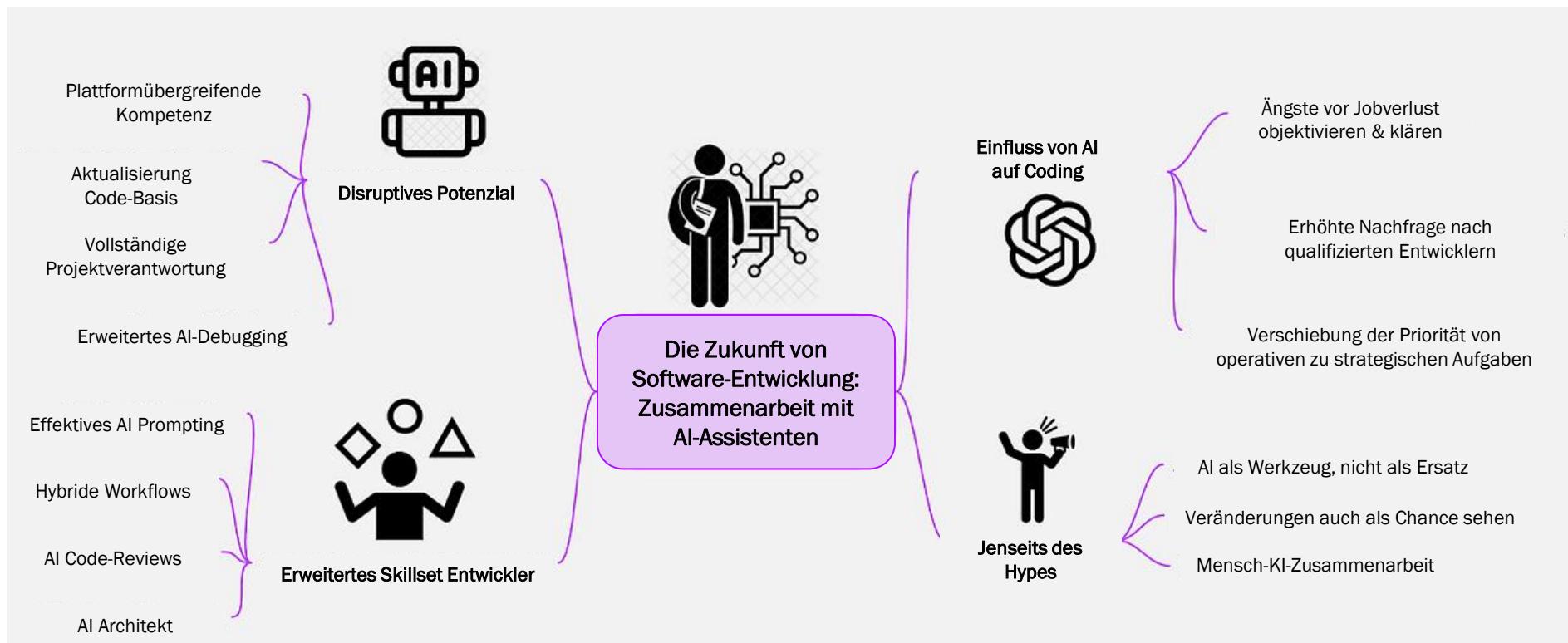


Amine Raji · Folgen

Veröffentlicht in Auf dem Weg zur Datenwissenschaft · 8 min Lesezeit · Vor 2 Tagen

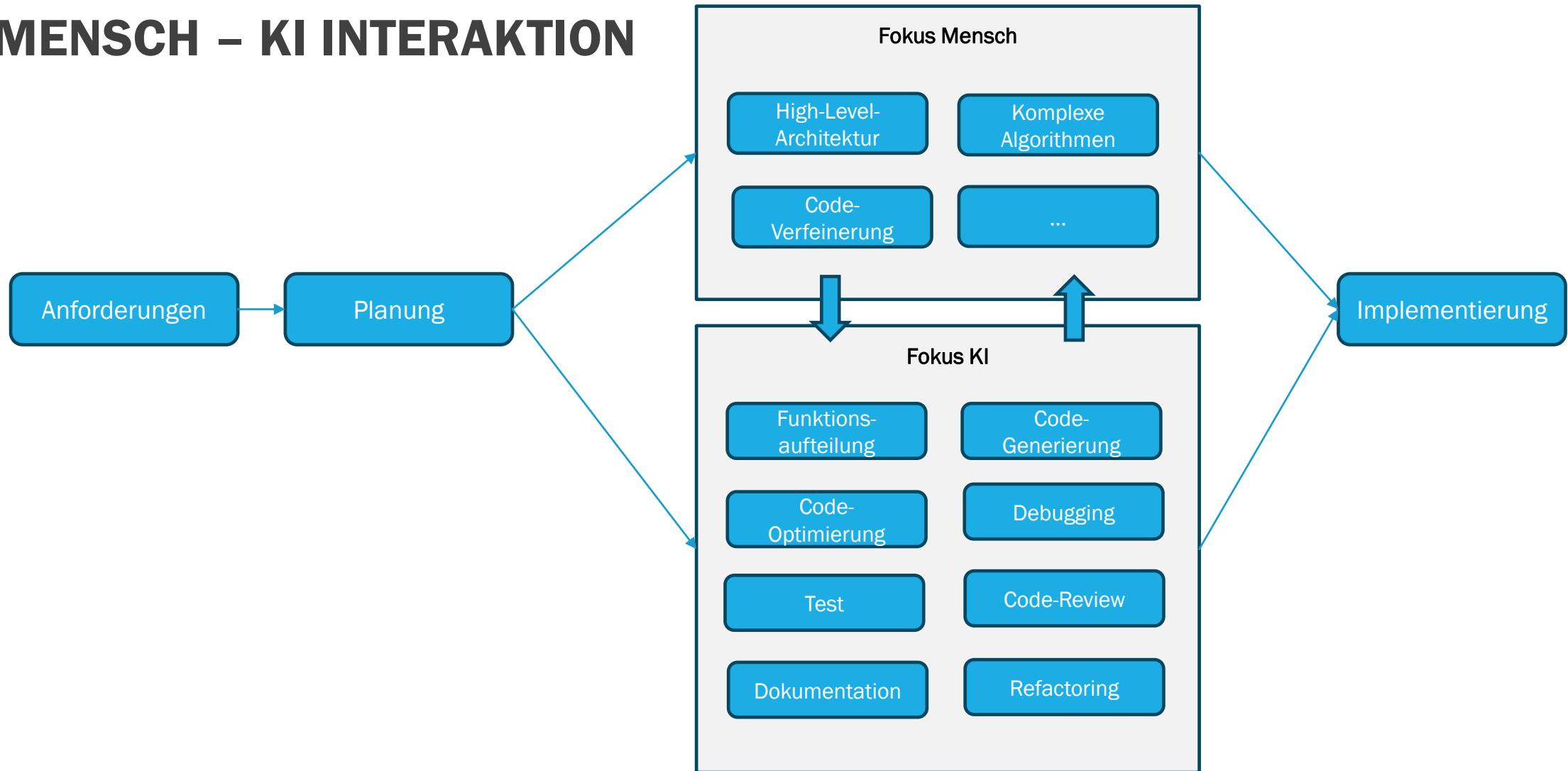


# KI VERÄNDERT CODING – EINE PERSPEKTIVE



Nach: [AI Won't Steal Your Coding Job \(But It Will Change It\) | by Amine Raji | Mar, 2024 | Towards Data Science \(medium.com\)](https://towardsdatascience.com/ai-wont-steal-your-coding-job-but-it-will-change-it-by-amine-raji-mar-2024)

# MENSCH – KI INTERAKTION



# ZEITAUFWAND ENTWICKLER & ENTWICKLUNGSTEAMS

## What development teams spend most of their time doing

Top 3 ranked responses, top responses shown, N=500



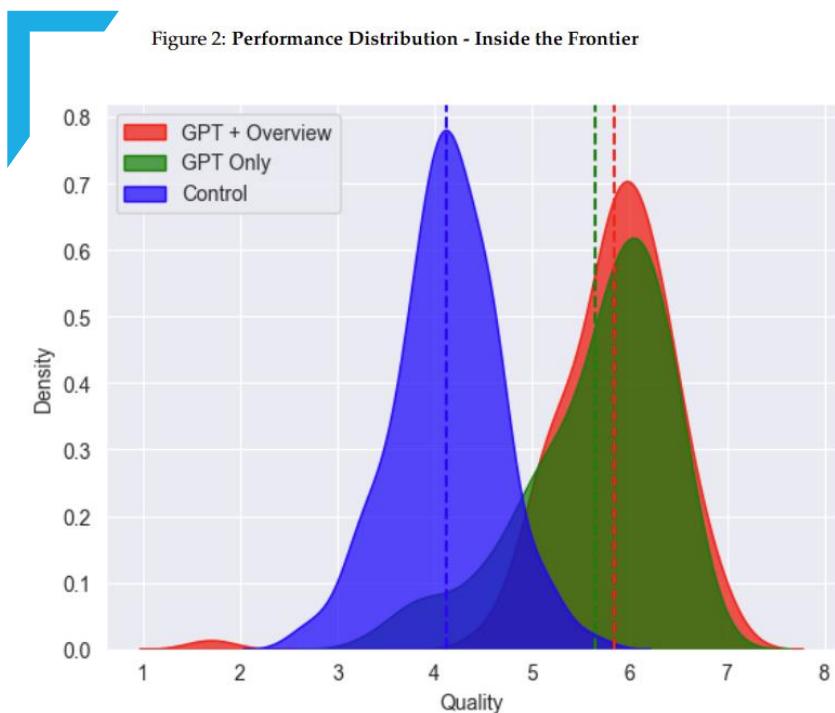
## What developers spend the most time on daily

Top 3 ranked responses, top responses shown, N=500



Quelle: [Survey reveals AI's impact on the developer experience - The GitHub Blog](#)

# PERFORMANCESTEIGERUNG



Quelle: Dell'Acqua et al. "Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality." Harvard Business School Working Paper, No. 24-013, September 2023

**Coding:** Laut der Studie "GenAI at Work" wurden Performanceverbesserung durch den Einsatz von AI-Tools erzielt. Die Studie ergab, dass der Zugang zu generativer KI die Produktivität im Durchschnitt um **14%** steigerte, wobei insbesondere weniger erfahrene und qualifizierte Mitarbeiter von einer Verbesserung um **34%** profitierten.

Quelle: Erik Brynjolfsson et. Al. GENERATIVE AI AT WORK, Working Paper 31161, <http://www.nber.org/papers/w31161>, NATIONAL BUREAU OF ECONOMIC RESEARCH. April 2023, revised November 2023

# CODING ASSISTANT TOOLS



**GitHub**  
Copilot



**Devin**

**Futurepedia**  
AI Tool Categories | Deals | Submit A Tool | Advertise | Resources  
[In-Depth Examples and Industry Applications ↓](#)  
[Exploring High-Impact Use Cases ↓](#)  
[Future Horizons: AI Code Assistants' Expanding Role ↓](#)

**171 AI Code Assistant Tools to Write Accurate Code Faster**

Featured | Popular | New | Filters | Verified | View

Tool	Type	Rating	Reviews	Description	Tags	Visit
Codeium	Free	★★★★★ (10)	3151	Elevate coding with AI-driven code completion, chat assistance, and...	#code assistant #education #low-code/no-code	<a href="#">Visit</a>
MarsX	Free	★★★★★ (0)	1293	Unleash rapid app development with AI, NoCode, and MicroApps...	#code assistant #website builders	<a href="#">Visit</a>
Safurai	Free	★★★★★ (0)	1234	Transform the way you use ChatGPT for coding	#code assistant #low-code/no-code	<a href="#">Visit</a>
Merlin	Free	★★★★★ (1)	1144	Enhance web productivity with AI: write, summarize, code, translate...	#ai chatbots #email assistant #summarizer	<a href="#">Visit</a>
Perplexity	Freemium	★★★★★ (2)	741	Unlock the power of AI to transform your search experience...	#writing generators #paraphrasing #copywriting	<a href="#">Visit</a>
Phind	Free	★★★★★ (2)	659	Unlock visual data insights with intuitive AI-powered image search.	#search engine #code assistant #image to image	<a href="#">Visit</a>
Replit	Freemium	★★★★★ (0)	692	Streamline coding, collaborate in real-time, deploy instantly from...	#code assistant #education	<a href="#">Visit</a>
GitFluence	Free	★★★★★ (1)	652	AI-powered instant Git command suggestions for streamlined...	#code assistant #github #low-code/no-code	<a href="#">Visit</a>
GPT-3 Playground	Free Trial	★★★★★ (1)	510	Unleash AI-driven text generation with unparalleled versatility and...	#writing generators #education #code assistant	<a href="#">Visit</a>
GitHub Copilot	Free Trial	★★★★★ (1)	489	AI-powered coding assistant offering real-time suggestions...	#code assistant #github #low-code/no-code	<a href="#">Visit</a>
Cron AI	Free	★★★★★ (0)	351	Transform plain text into precise cron expressions effortlessly...	#code assistant #workflows	<a href="#">Visit</a>
Tabnine	Freemium Free Trial	★★★★★ (0)	324	Unlock rapid, secure coding with AI-driven personalized...	#code assistant #education #workflows	<a href="#">Visit</a>

< Previous | 1 | 2 | 3 | 4 | 5 | Next >

# GRENZEN VON KI

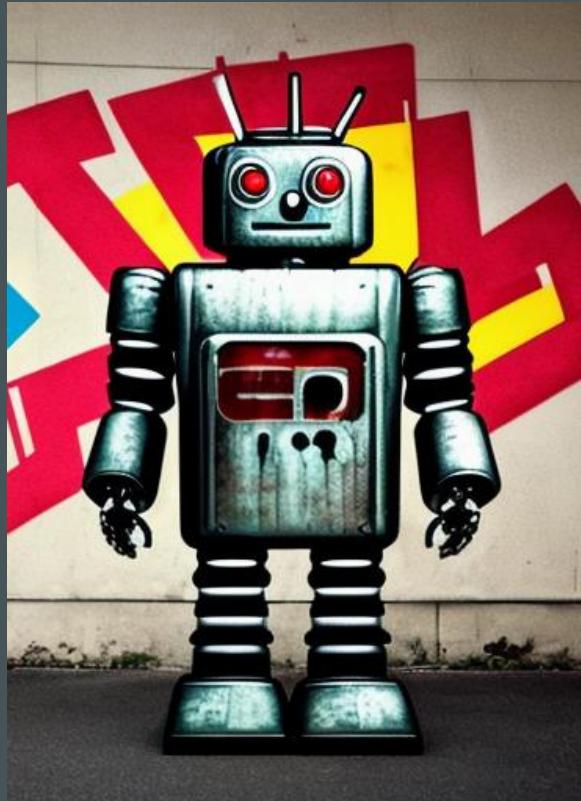


Bild mit DALL-E erstellt

Können wir uns  
immer auf den  
KI-Assistenten  
verlassen?

# KI-HALLUZINATIONEN



Bild von [Mario](#) auf [Pixabay](#)

- Für welches KI-Modell man sich auch entscheidet, es ist wichtig zu beachten, dass KI-Modelle (oft) „**halluzinieren**“ und Informationen geben, die sachlich falsch sind.
- Dieses Verhalten ist darauf zurückzuführen, dass z.B. LLMs auf riesigen Datensätzen mit **logischen** Fehlern und Unsinn trainiert werden.
- Es gibt Antworten, die keinen Bezug zur Realität haben. Wenn Sie „**lügen**“, **verzerren** Sie die Realität. Wenn Sie die Wahrheit sagen, beschreiben sie **ihre Darstellung** der Realität. Aber wenn sie „**Bullshit**“ machen, erfinden sie Dinge, ohne Rücksicht darauf, wie die Realität aussehen könnte.
- Sprachmodelle sind höchst plausible „**Bullshitter**“, die oft auf der Wahrheit landen. Man kann ihnen jedoch nicht völlig vertrauen.

# KI-BIAS

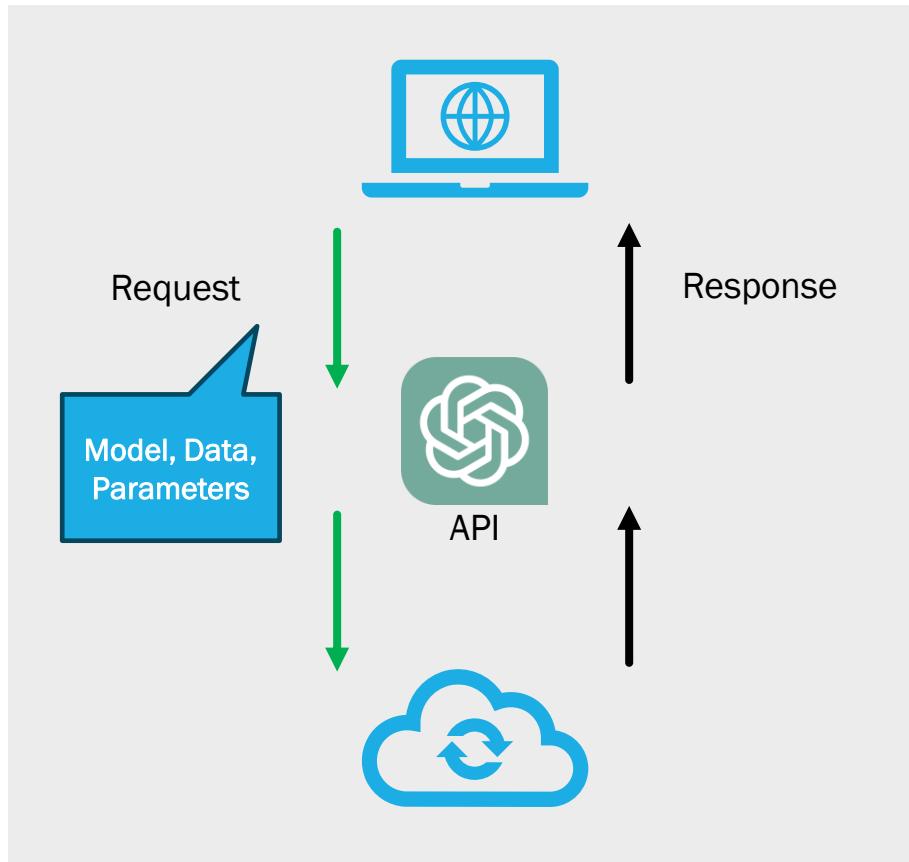


Bild von [Kawita Chitprathak](#) auf [Pixabay](#)

- KI-Bias bezeichnet systematische, oft **daten- und/oder Algorithmus bedingte Verzerrungen**, die einschränkende Resultate, z.B. hinsichtlich Rasse, Geschlecht oder Technologie, bewirken können.
- Hauptursachen sind **unausgewogene Trainingsdaten**, die bestehende Vorurteile spiegeln, und die Programmierung der Algorithmen.
- KI-Bias verstärkt Ungleichheiten und kann zu fehlerhaften Lösungen führen.
- Für die **Minimierung** von KI-Bias sind diverse, repräsentative Daten und ethische KI-Entwicklungsstandards essenziell.

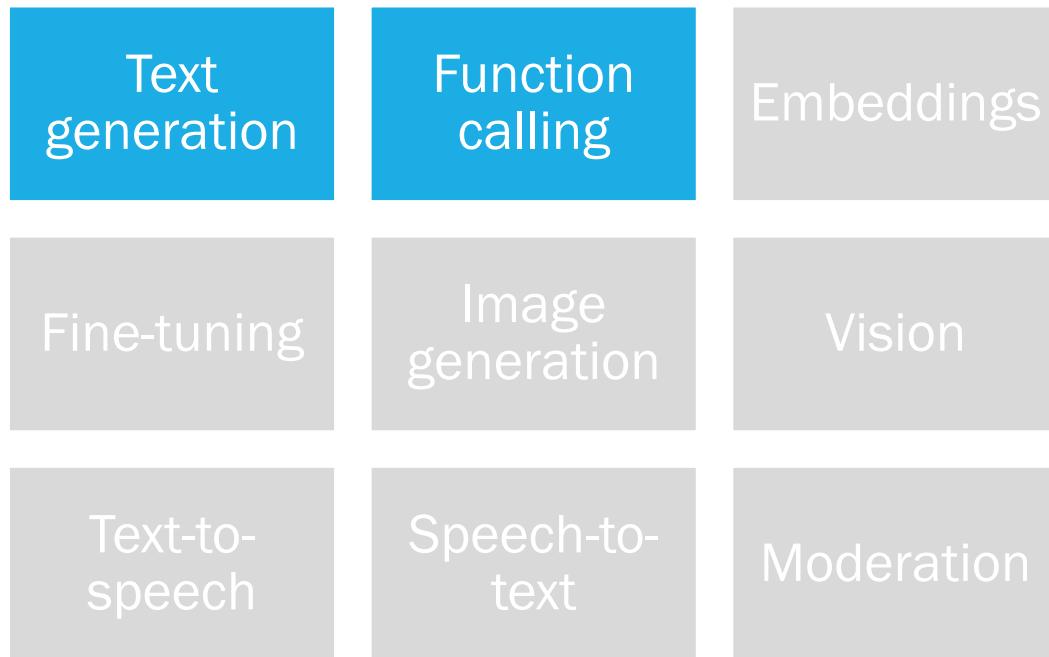
# OPENAI API

# OPENAI API - FUNKTIONSWEISE



- Eine **API** (Application Programming Interface) ist eine Sammlung von Regeln und Spezifikationen, die es Softwareanwendungen ermöglicht, miteinander zu kommunizieren (Schnittstelle).
- Die **OpenAI API** ermöglicht es Entwicklern, Zugang zu fortschrittlicher KI-Technologie zu erhalten, insbesondere zu den Sprachmodellen von OpenAI wie GPT (Generative Pre-trained Transformer).
- Diese API bietet eine **breite Palette von Funktionen**, darunter Textgenerierung, Textverständnis, Übersetzungen, Zusammenfassungen und viele andere sprachbasierte Aufgaben.
- Die API ist so gestaltet, dass sie **leicht zugänglich** und **benutzerfreundlich** ist, wodurch sie für ein breites Spektrum von Anwendungen, von kleinen Projekten bis hin zu groß angelegten Unternehmungen, geeignet ist.

# OPENAI CAPABILITIES



- OpenAI bietet eine Vielzahl von **Fähigkeiten** (Capabilities) über seine API an, die künstliche Intelligenz in verschiedenen Formen für Entwickler zugänglich macht.
- Diese Fähigkeiten umfassen die Generierung von Text, das **Verstehen** und **Beantworten** von Fragen, die **Übersetzung** von Texten zwischen verschiedenen Sprachen, die Erstellung von Zusammenfassungen, die Erzeugung von Code und vieles mehr.

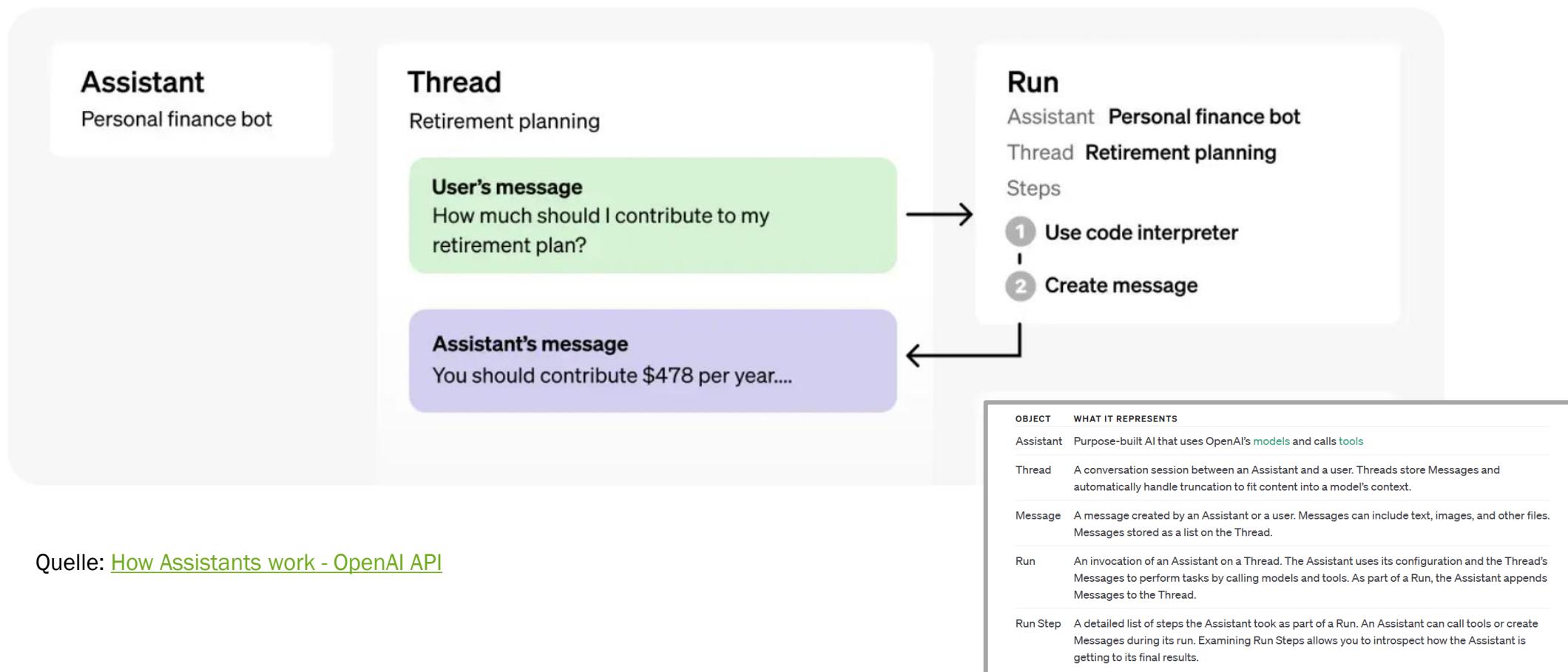
# OPENAI ASSISTENT



Bild mit ideogram.ai erstellt

- **Assistenten** können die Modelle von OpenAI aufrufen und ihre Instruktionen und Fähigkeiten nutzen.
- Assistenten können parallel auf mehrere **Tools** zugreifen, sowohl von OpenAI gehostete Tools (z.B. Code-Interpreter & Retrieval) als auch Tools, die Sie erstellen/hosten.
- Assistenten können auf persistente **Threads** (Chat-/Ausführungsstrang) zugreifen. Threads vereinfachen die Entwicklung von KI-Anwendungen, indem sie den Nachrichtenverlauf speichern.
- Assistenten können auf Dateien in verschiedenen **Formaten** zugreifen – entweder als Teil ihrer Erstellung oder als Teil von Threads zwischen Assistenten und Benutzern (z. B. Bilder, Tabellenkalkulationen usw.) - und Dateien zitieren, auf die sie in erstellten Nachrichten verweisen.

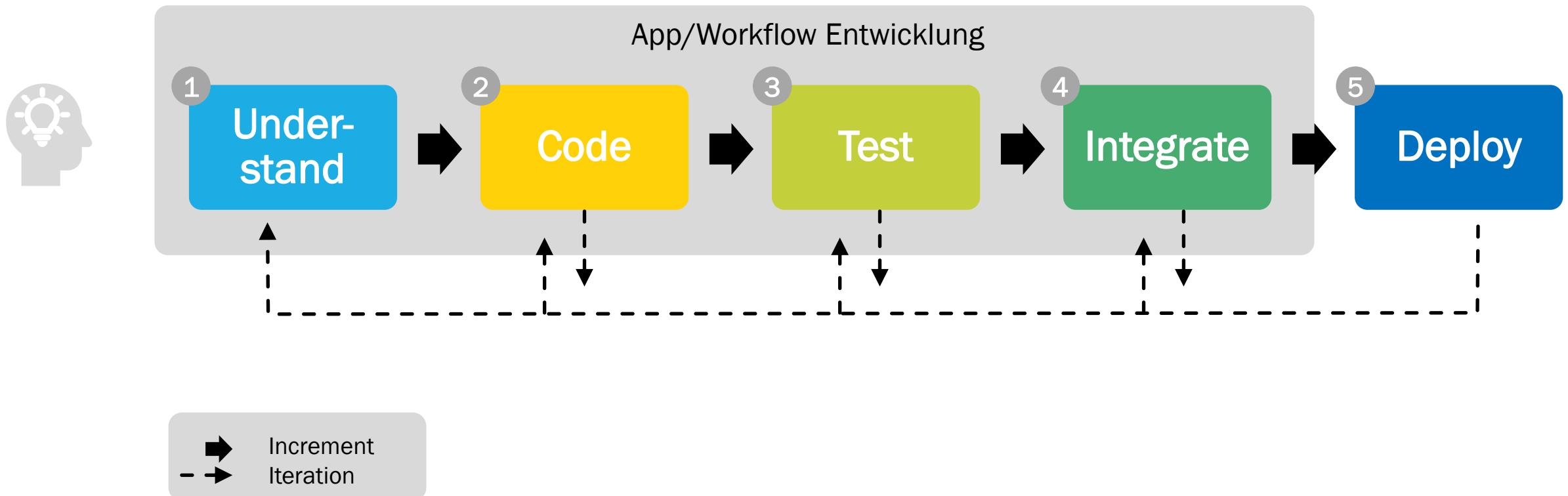
# WIE ARBEITET EIN OPENAI ASSISTENT



Quelle: [How Assistants work - OpenAI API](#)

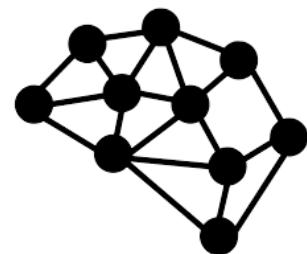
# ENTWICKLUNGSPROZESS

# PROZESSMODELL



Performance  
steigern

# KI ALS SUPPORTER / ASSISTANT



- Code-Schnipsel: Sie können KI bitten, Ihnen Beispiele oder kleine Code-Schnipsel zu geben, die bestimmte Funktionalitäten demonstrieren.
- Code-Verständnis: Wenn Sie Schwierigkeiten haben, einen bestimmten Code-Schnipsel oder eine Funktion zu verstehen, können Sie KI um Erklärungen bitten.
- Fehlerbehebung: Haben Sie einen Fehler in Ihrem Code, den Sie nicht beheben können? Sie können den Fehler und den relevanten Code an KI weitergeben, um Hinweise oder Lösungsvorschläge zu erhalten.
- Algorithmus-Erklärungen: Wenn Sie einen bestimmten Algorithmus oder eine Datenstruktur nicht verstehen, kann KI Erklärungen oder Pseudocode bereitstellen.
- Tools und Libraries: Fragen Sie nach Empfehlungen für Tools, Libraries oder Frameworks, die für Ihre Aufgabe geeignet sind.
- Lernressourcen: Erhalten Sie Empfehlungen für Bücher, Online-Kurse oder Tutorials zu spezifischen Programmierthemen. KI kann auch direkt als interaktives Tutorial genutzt werden.
- ...

Gute  
Option

## RE-USE



- Finde einen vorhandenen, ähnlichen Workflow.
- Nehme den einfachsten Workflow, der zu finden ist.
- Analysiere diesen Workflow und baue ein Verständnis auf.
- Nehme die erforderlichen Anpassungen vor.
- Bringe den angepassten/veränderten Workflow zum Laufen.

Bild von [Shirley Hirst](#) auf [Pixabay](#)

Gute  
Option

# CODE-SNIPPETS



Bild von [Hans](#) auf [Pixabay](#)

- Code-Snippets sind vorgefertigte Codefragmente, die in der Programmierung wiederverwendet werden können.
- Sie können in verschiedenen Entwicklungsumgebungen, Texteditoren oder IDEs (Integrated Development Environments) eingesetzt werden, um Entwicklungszeit zu sparen und die Effizienz zu steigern.
- Einsatzmöglichkeiten u.a.:
  - Wiederverwendung von Code: Häufig wiederkehrende Codeblöcke, die in verschiedenen Projekten oder sogar innerhalb desselben Projekts verwendet werden sollen. Das spart Zeit und vermeidet Tippfehler.
  - Schnelle Implementierung: Code-Snippets können helfen, häufig verwendete Funktionen oder Algorithmen schnell zu implementieren.
  - ...

Gute  
Option

# CHECKLISTEN



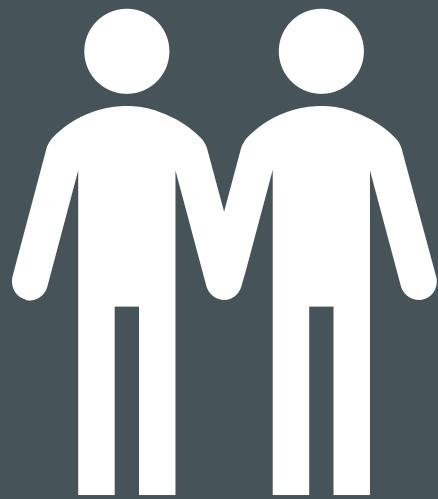
Bild von [Gerry](#) auf [Pixabay](#)

In der Fliegerei werden Checklisten verwendet, um sicherzustellen, dass wichtige Schritte und Verfahren ordnungsgemäß durchgeführt werden und keine wichtigen Details übersehen werden.

- **Sicherheit:** Checklisten helfen dabei, sicherheitsrelevante Aufgaben und Verfahren zu standardisieren und sicherzustellen, dass sie in einer sicheren und konsistenten Art und Weise durchgeführt werden. Dies minimiert das Risiko menschlicher Fehler und reduziert potenzielle Gefahren.
- **Komplexität:** Es gibt eine Vielzahl von Systemen, Verfahren und Protokollen, die beachtet werden müssen. Checklisten helfen den Piloten und dem Flugpersonal dabei, diese Komplexität zu bewältigen, indem sie sicherstellen, dass keine wichtigen Aufgaben oder Überprüfungen vergessen werden.
- **Standardisierung:** Checklisten ermöglichen eine Standardisierung der Abläufe und Verfahren. Dies ist besonders wichtig, wenn mehrere Piloten oder Crewmitglieder zusammenarbeiten, da es sicherstellt, dass alle nach dem gleichen Prozess arbeiten und keine wichtigen Schritte auslassen.
- **Vermeidung von Gedächtnisfehlern:** Durch das Arbeiten mit Checklisten wird das Risiko von Gedächtnisfehlern minimiert. Gerade in Stresssituationen oder unter Zeitdruck kann das Gedächtnis versagen oder wichtige Details vergessen werden. Checklisten bieten eine klare Anleitung und erinnern die Piloten und das Flugpersonal an alle notwendigen Schritte.

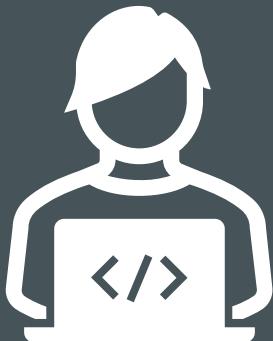
## PAIR PROGRAMMING

---



- Beim Pair Programming arbeiten zwei Entwickler:innen gleichberechtigt (an einem Rechner) an einer gemeinsamen Aufgabe.
- Die zwei Entwickler:innen nehmen unterschiedliche Rollen ein, welche oft mit „Pilot“ und „Navigator“ bezeichnet werden.
- Der „Pilot“ schreibt den Code, während der „Navigator“ die Korrektheit des Codes und des Lösungsansatzes überwacht und parallel über Verbesserungen am Design nachdenkt.
- Der „Pilot“ wechselt alle paar Minuten zum „Navigator“ und der „Navigator“ wird zum „Piloten“.
- Welche Vorteile kann Pair Programming bieten?
  - Qualität der Software wird verbessert
  - Fehler werden früh erkannt
  - Mehr Spaß an der Arbeit
  - Wissen verbreitet sich im gesamten Team
  - Kommunikation im Team verbessert sich

## PAIR PROGRAMMING IN GOOGLE COLAB



- Rolle Pilot & Navigator wechselt von Aufgabe zu Aufgabe.
- Pilot gibt seinen Bildschirm frei und erstellt den Code.
- Navigator gibt Hinweise zu Lösungsansatz, Korrektheit, Verbesserungen, ...
- Teilen des Notebooks erfolgt nach Fertigstellung der Aufgabe.

Oder

- Pilot erstellt Jupyter-Notebook und speichert es auf ihrem/seinem Google Drive.
- Mit Menü-Punkt Teilen wird das Notizbuch mit einem anderen Google Mailaccount geteilt.
- Pilot & Navigator haben Zugriff auf das Notebook.
- Echtzeitcollaboration erfolgt mit leichtem Zeitversatz.

# **INTEGRIERTE ENTWICKLUNGSUMGEBUNG**

# PYTHON ENTWICKLUNGSUMGEBUNGEN (IDE)

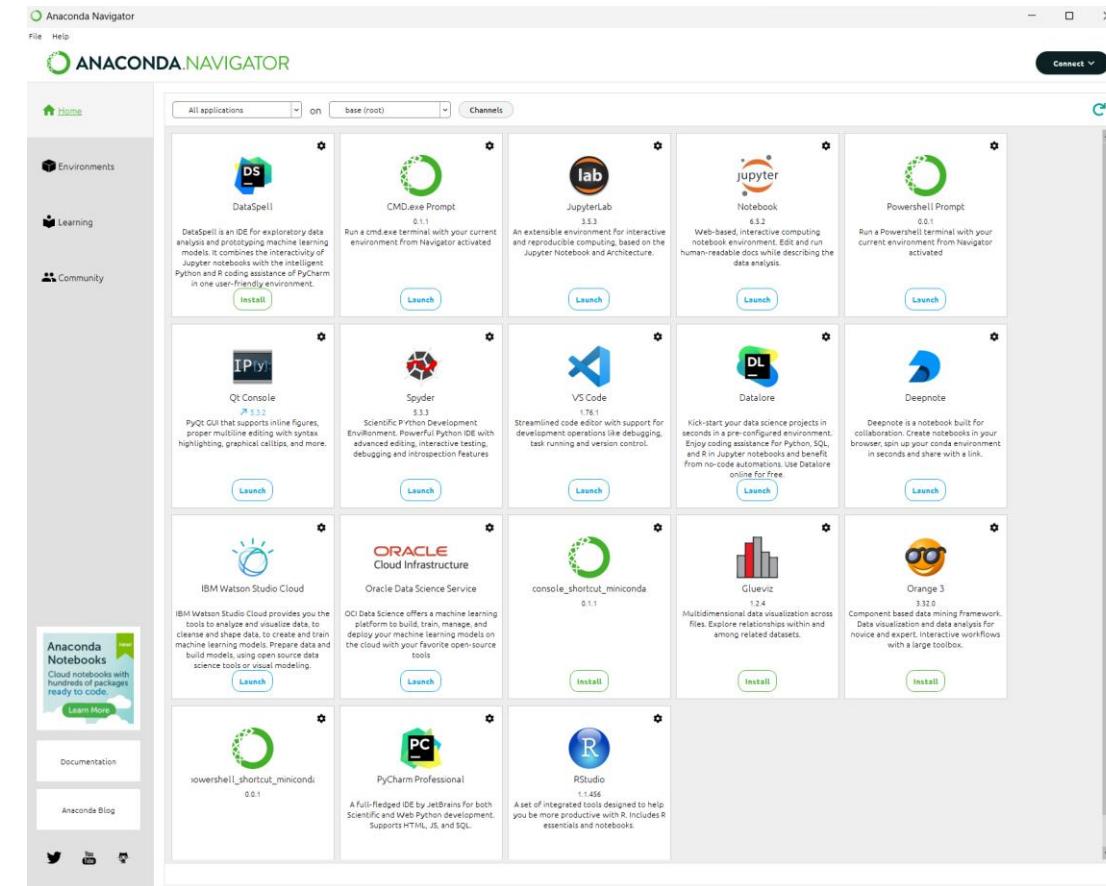
- Eine integrierte Entwicklungsumgebung (IDE) ist eine Sammlung von Apps/Funktionen, mit denen die Aufgaben der Softwareentwicklung möglichst ohne Medienbrüche bearbeitet werden können.
- IDEs stellen hilfreiche Werkzeuge bereit, die Softwareentwicklern häufig wiederkehrende Aufgaben unterstützt, z.B. Arbeits(zwischen)ergebnisse verwaltet, Code erstellen/ändern, Code dokumentieren, Code testen, etc. Entwickler werden dadurch von formalen Arbeiten entlastet und können ihre eigentliche Aufgabe, das Entwickeln/Programmieren von Software, mit Systemunterstützung effizient ausführen.
- Bekannte Python-IDE's sind neben Google Colab:
  - Visual Studio Code
  - Jupyter
  - PyCharm
  - Spyder

Siehe z.B. auch: [10 Beste Python-IDE für Supercharge-Entwicklung und -Debugging \(geekflare.com\)](https://geekflare.com/best-python-ide/)

# ANACONDA - DISTRIBUTION

Anaconda ist eine Distribution für die Programmiersprachen Python und R, die unter anderem die Entwicklungsumgebung Visual Studio Code und Jupyter Notebook/Lab enthält.

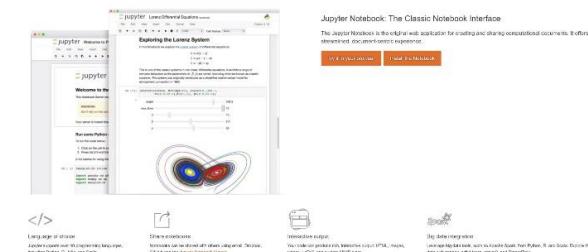
Das Ziel der Distribution ist die Vereinfachung von Paketmanagement und Softwareverteilung.



# PROJEKT JUPYTER

Das Projekt Jupyter ist der Herausgeber von Softwareprodukten für interaktive wissenschaftliche Datenauswertung und wissenschaftliche Berechnungen. Das Project Jupyter hat die Produkte Jupyter Notebook, JupyterHub und JupyterLab entwickelt.

[Link](#)



# COMPILING UND EXE-UMWANDLUNG

Python EXE Datei erstellen



Dauer: 3:38

[\(49\) Python Datei in EXE umwandeln - YouTube](#)

## Compiling & Decompiling Python Scripts

Dauer: 9 Min

[\(49\) Compiling & Decompiling Python Scripts – YouTube](#)

## Convert Python To Exe Files

Dauer: 8 Min

[\(49\) Convert Python To Exe Files - YouTube](#)

# UNIT-TEST - BEISPIEL

```
▶ 1 import unittest  
2  
3 def addiere(a, b):  
4     return a + b  
5  
6 class TestAddition(unittest.TestCase):  
7     def test_addiere_positive_zahlen(self):  
8         self.assertEqual(addiere(1, 2), 3)  
9  
10    def test_addiere_negative_zahlen(self):  
11        self.assertEqual(addiere(-1, -2), -1)  
12  
13 if __name__ == '__main__':  
14     unittest.main(argv=['first-arg-is-ignored'], exit=False)  
  
☞ F.  
=====FAIL: test_addiere_negative_zahlen (__main__.TestAddition)-----  
Traceback (most recent call last):  
  File "<ipython-input-6-2f9658d0ed86>", line 11, in test_addiere_negative_zahlen  
    self.assertEqual(addiere(-1, -2), -1)  
AssertionError: -3 != -1-----  
Ran 2 tests in 0.157s  
FAILED (failures=1)
```

- Die Python-Bibliothek unittest wird verwendet, um Unit-Tests für Python-Programme zu schreiben und auszuführen. Unit-Tests sind Tests, die einzelne Komponenten oder Funktionen eines Programms überprüfen, um sicherzustellen, dass sie wie erwartet funktionieren. Unit-Tests helfen, Fehler im Code frühzeitig zu finden und zu beheben, die Codequalität zu verbessern und die Softwareentwicklung zu beschleunigen.
- Die unittest-Bibliothek bietet eine Reihe von Funktionen und Klassen, um Unit-Tests zu erstellen, zu organisieren und auszuführen. Zum Beispiel
  - Die Klasse **Testcase** repräsentiert eine einzelne Testeinheit, die eine oder mehrere Testmethoden enthält.
  - Die Klasse **Testsuite** repräsentiert eine Sammlung von Testfällen oder Testsuiten, die zusammen ausgeführt werden sollen.
  - Die Klasse **TestRunner** steuert die Ausführung der Tests und liefert das Ergebnis an den Benutzer zurück.
  - Die Funktion **assert** überprüft, ob eine Bedingung wahr ist, und löst eine Ausnahme aus, wenn sie falsch ist.

# STYLE GUIDE

# BENENNUNGSKONVENTIONEN FÜR VARIABLEN

Konvention	Beschreibung	Beispiel
PascalCase	Jedes Wort beginnt mit einem Großbuchstaben ohne Unterstriche.	MeineVariable
camelCase	Erstes Wort klein geschrieben, jedes folgende Wort groß beginnend.	meineVariable
snake_case	Wörter klein geschrieben, durch Unterstriche getrennt.	meine_variable
UPPER_SNAKE_CASE	Wörter groß geschrieben, durch Unterstriche getrennt.	MEINE_KONSTANTE

# 10 REGELN ZUR GESTALTUNG VON GUT LESBAREM CODE

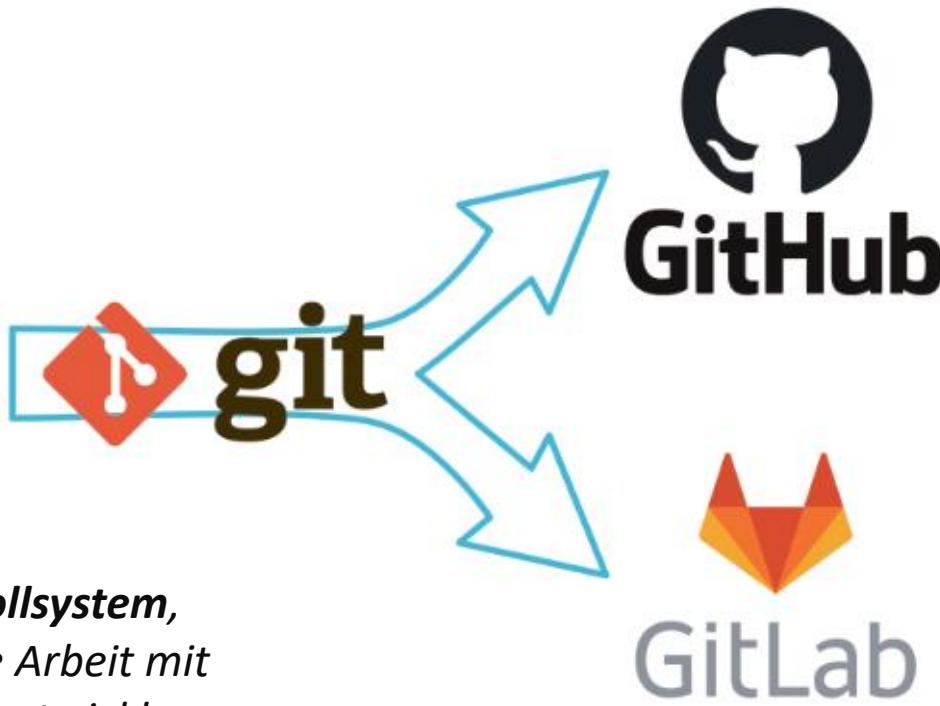
1. Klare und konsistente Benennung: Verwende aussagekräftige Namen für Variablen, Funktionen und Klassen, die deren Zweck widerspiegeln. Halte dich an eine Namenskonvention (z.B. `snake_case` für Variablen und Funktionen in Python, `PascalCase` für Klassen).
2. Vermeidung langer Funktionen und Klassen: Zerlege deinen Code in kleinere, wiederverwendbare Funktionen und Klassen, die jeweils nur eine spezifische Aufgabe erfüllen. Dies macht den Code leichter zu verstehen und zu testen.
3. Kommentare und Dokumentation: Kommentiere deinen Code, wo es notwendig ist, um zu erklären, warum etwas auf eine bestimmte Weise gemacht wird. Verwende Docstrings (‘‘‘ `Doc-String` ‘‘‘), um Funktionen, Klassen und Module in Python zu dokumentieren.
4. Einhalten von Code-Standards und Style-Guides: Folge den PEP 8-Richtlinien für Python, um Konsistenz im Code-Stil zu gewährleisten. Dies umfasst Empfehlungen zur Formatierung, zum Zeilenabstand und zu anderen Stilfragen.
5. Verwendung von Versionierungstools: Nutze Versionierungstools wie Git, um Änderungen am Code zu verfolgen. Dies erleichtert die Zusammenarbeit und hilft, den Überblick über verschiedene Versionen des Projekts zu behalten.

# 10 REGELN ZUR GESTALTUNG VON GUT LESBAREM CODE

6. Schreiben von Tests: Implementiere Unit-Tests, um die Korrektheit deines Codes zu überprüfen. Tests helfen dabei, Fehler frühzeitig zu entdecken und sicherzustellen, dass Änderungen am Code die bestehende Funktionalität nicht beeinträchtigen.
7. Vermeidung von globalen Variablen: Beschränke die Verwendung globaler Variablen, da sie den Code schwerer zu verstehen und zu debuggen machen können. Verwende stattdessen lokale Variablen oder übergib Variablen an Funktionen.
8. Vermeidung von magischen Zahlen und Strings: Definiere Konstanten für Zahlen und Strings, die eine spezifische Bedeutung haben, anstatt sie direkt in den Code einzufügen. Dies macht den Code lesbarer und erleichtert Änderungen an diesen Werten.
9. Beachtung der Prinzipien der Softwareentwicklung: Prinzipien wie DRY (Don't Repeat Yourself) und KISS (Keep It Simple, Stupid) können dabei helfen, Redundanzen zu vermeiden und die Komplexität des Codes zu minimieren.
10. Regelmäßige Code-Reviews: Lasse deinen Code regelmäßig von anderen überprüfen. Code-Reviews können helfen, Fehler, schlechte Praktiken und Verbesserungsmöglichkeiten zu identifizieren.

# GIT & GITHUB

# GIT - GITHUB - GITLAB



*Git ist ein **Versionskontrollsystem**, das die Grundlage für die Arbeit mit Codeversionierung und -entwicklung legt*

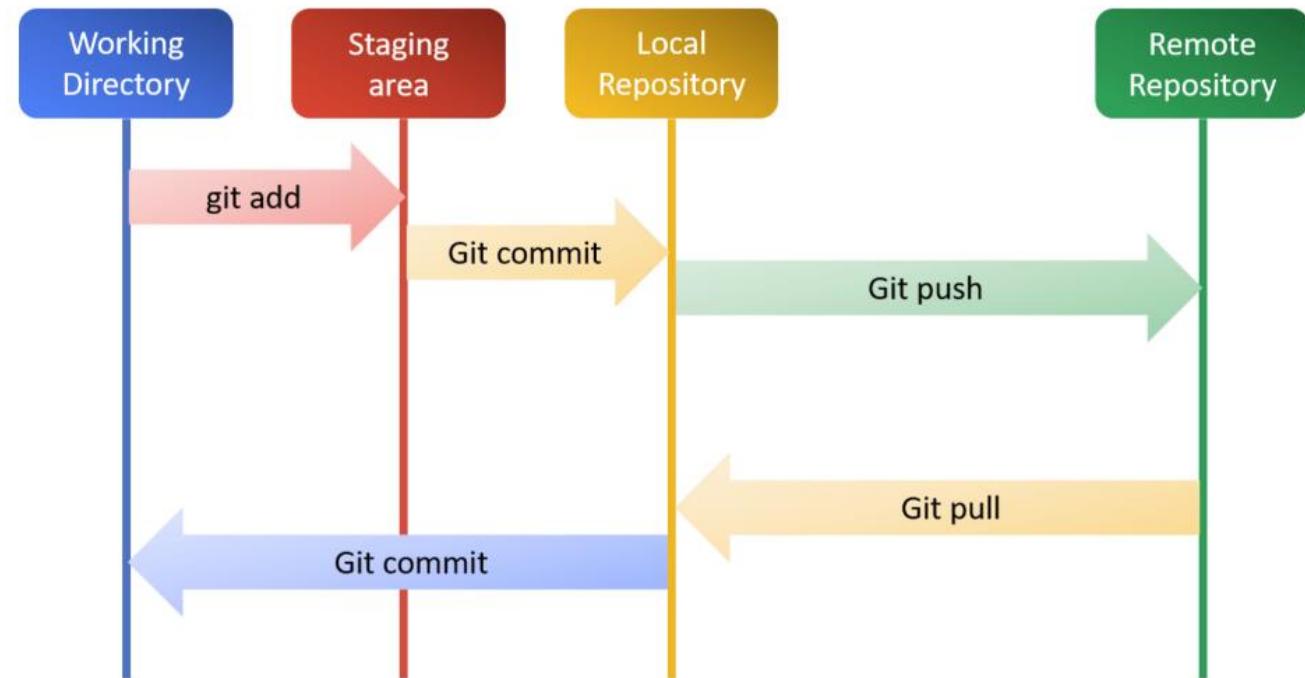
Quelle: [Github or Gitlab: Which is the Best for Your Code in 2024? \(atonce.com\)](https://atonce.com/github-or-gitlab-best-for-code-in-2024/)

*GitHub und GitLab sind Plattformen, die Git-Hosting-Dienste anbieten und durch zusätzliche Tools und Funktionen die Zusammenarbeit und das Projektmanagement in Softwareentwicklungsprojekten erleichtern.*

*Während GitHub besonders populär für Open-Source-Projekte und seine Community ist, bietet GitLab eine integrierte Lösung, die sich besonders für umfassende DevOps-Anforderungen und private Projekte eignet.*

# FUNKTIONSWEISE VON GIT

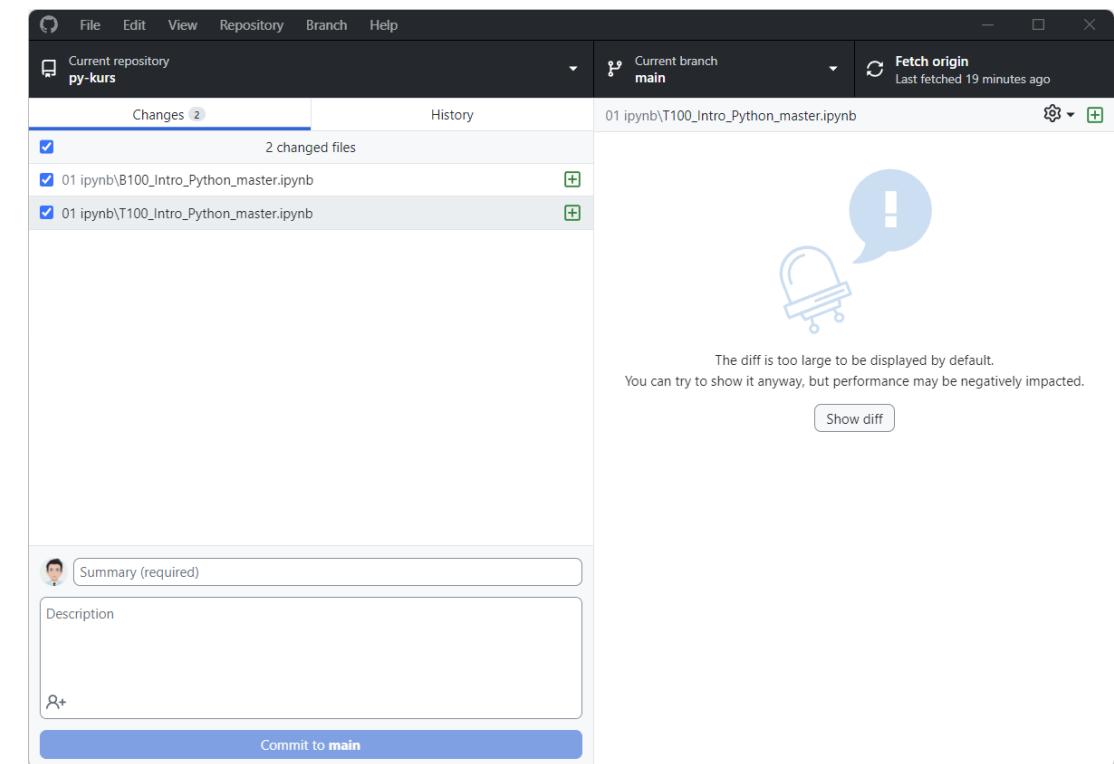
- Im **Arbeitsbereich** bearbeiten Sie Ihre Dateien. Es ist einfach das Verzeichnis auf Ihrem lokalen Computer, in dem Ihre Projektdateien gespeichert sind.
- Alle Änderungen an den Dateien im Arbeitsbereich, werden von Git erst verfolgt, wenn sie dem **Staging-Bereich** hinzufügt werden.
- Der Staging-Bereich, auch Index genannt, ist ein temporärer Speicherbereich, in dem Git Änderungen an Dateien verfolgt, die zum Festschreiben bereit sind.
- Sobald Änderungen festgeschrieben werden sollen, erfolgt ein commit, um sie dauerhaft im **local Repository** zu speichern.
- Im Repository speichert Git die Projektdateien und deren vollständigen Verlauf dauerhaft.
- Mit einem push werden diese Änderungen in das **Remote Repository** übertragen.



Quelle: [What is GitHub? A Beginner's Introduction to Git and GitHub - WPBlog \(whooshpro.com\)](http://What%20is%20GitHub?%20A%20Beginner's%20Introduction%20to%20Git%20and%20GitHub%20-%20WPBlog%20(whooshpro.com))

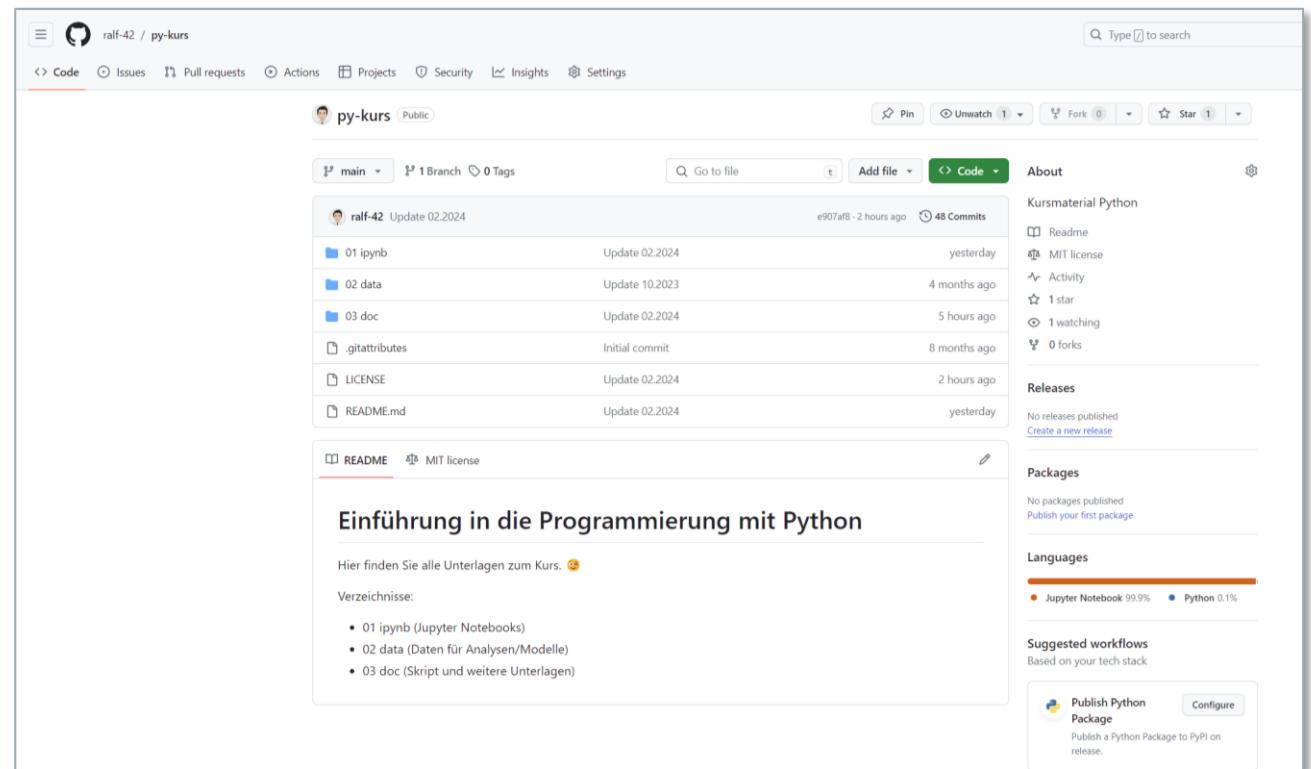
# GITHUB DESKTOP

- Änderungen im **Working Directory** werden in der linken Seitenleiste aufgelistet, wo die Dateien und die Art der Änderungen (hinzugefügt, geändert, gelöscht) gezeigt werden.
- Zum **Staging** hinzufügen erfolgt durch Auswahl der Dateien, die du commiten möchtest. GitHub Desktop ermöglicht es, spezifische Änderungen auszuwählen, die in das nächsten Commit aufgenommen werden.
- **Commit und Local Repository:** Nach der Auswahl Änderungen kann ein Commit durchgeführt werden. Dies aktualisiert das lokale Repository mit den neuen Änderungen.
- **Push:** Schließlich werden die Commits von einem lokalen Repository an das entfernte Remote Repository auf GitHub gepusht, um die Änderungen verfügbar zu machen.



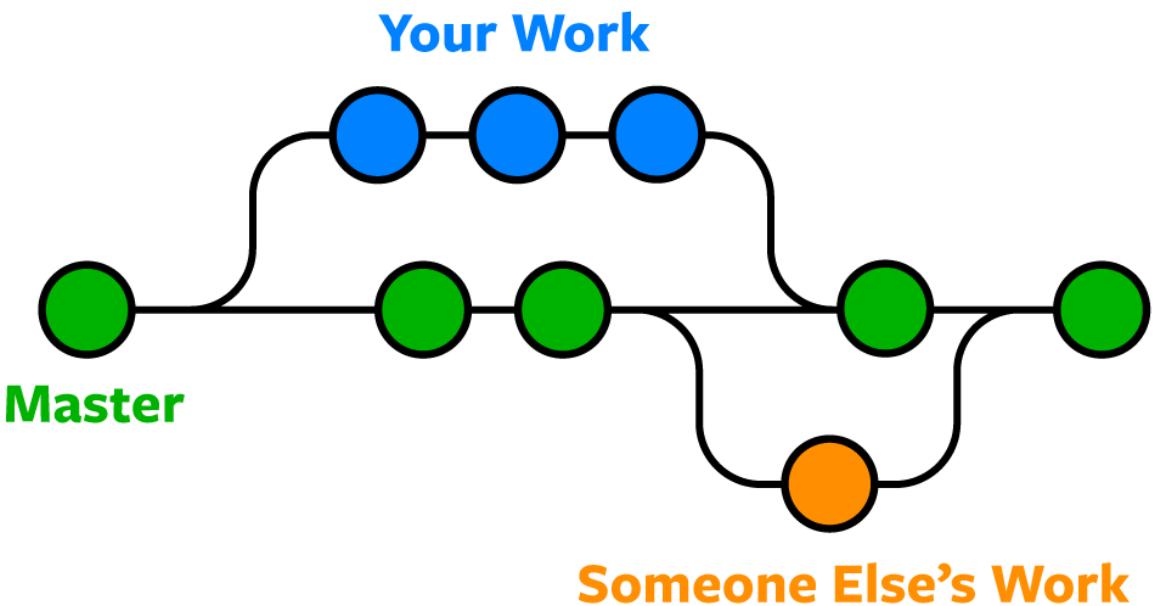
# GITHUB - REPOSITORY

- Ein GitHub-Repository, oft einfach als Repo bezeichnet, ist ein grundlegendes Element auf der GitHub-Plattform, das verwendet wird, um ein Projekt zu speichern.
- Es enthält alle Projektdateien, einschließlich Dokumentation, Code, Konfigurationsdateien und mehr, sowie die Versionsgeschichte aller Dateien.
- Repositories auf GitHub können verwendet werden, um Code zu speichern, zu teilen und zu verwalten, und sie bieten eine Reihe von Tools zur Zusammenarbeit an Softwareprojekten.



# BRANCHES - VERZWEIGUNGEN

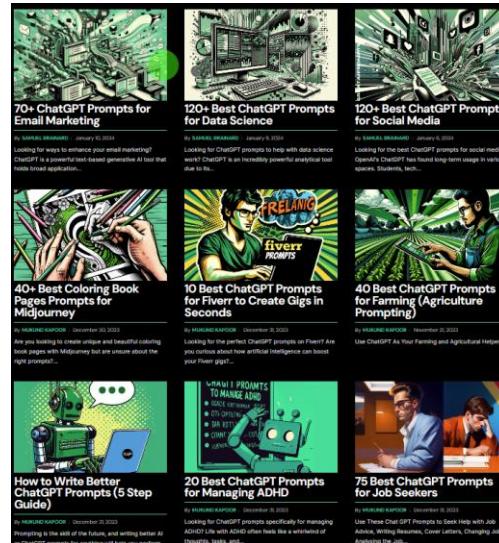
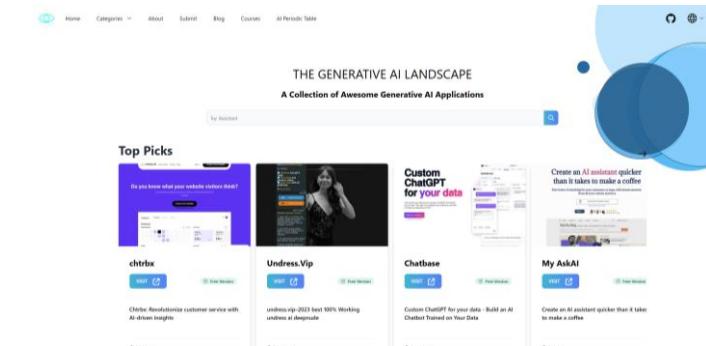
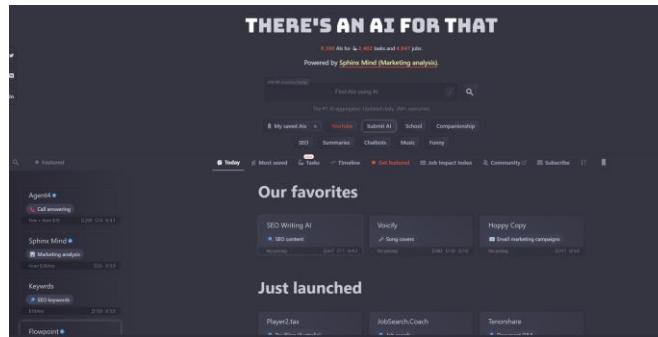
- Nahezu jedes Versionskontrollsysteem (VCS) unterstützt eine Form von Branching.
- Branching bedeutet, dass man von der Hauptlinie (Main, Master) der Entwicklung abzweigen und die Arbeit fortsetzen, ohne die Hauptlinie zu stören.
- Die Art und Weise, wie Git Branches anlegt ist unglaublich leichtgewichtig.
- Branch-Operationen werden nahezu verzögerungsfrei ausgeführt und auch das Hin- und Herschalten zwischen einzelnen Entwicklungszweigen läuft meist genauso schnell ab.
- Sobald Änderungen genehmigt wurden, kann die Entwicklung mithilfe von Git mit dem ursprünglichen Projekt (Main, Master) zusammenführen werden.



Quelle: [What is GitHub? A Beginner's Introduction to Git and GitHub - WPBlog \(whooshpro.com\)](http://What%20is%20GitHub?%20A%20Beginner's%20Introduction%20to%20Git%20and%20GitHub%20-%20WPBlog%20(whooshpro.com))

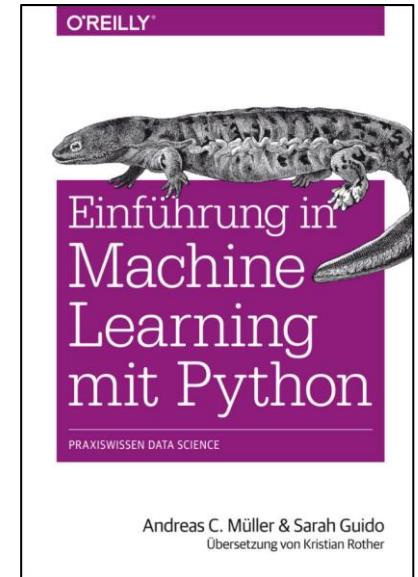
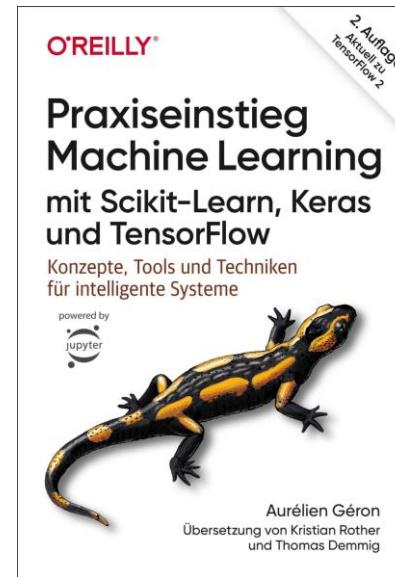
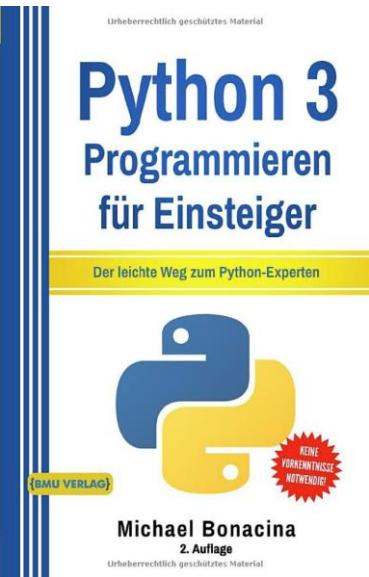
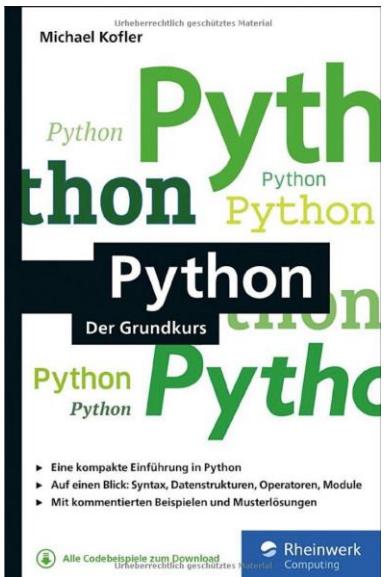
# ANHANG

# AI-COLLECTIONS

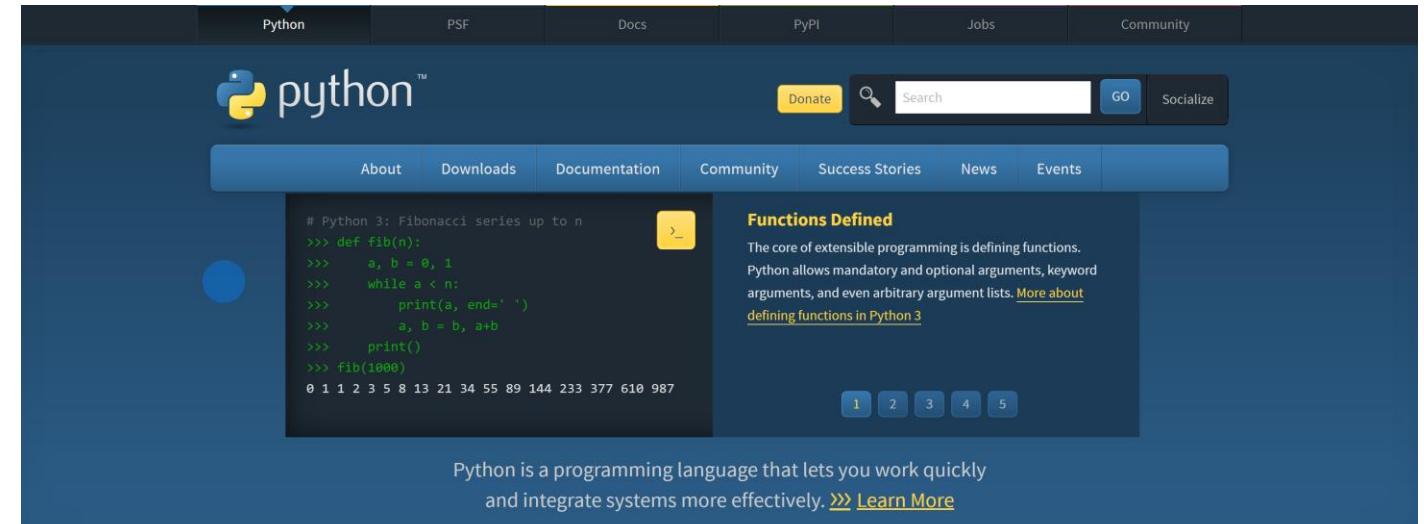


## Hugging Face

# BÜCHER



# PYTHON.ORG



The screenshot shows the Python.org homepage with a dark blue header featuring the Python logo and the word "python™". The header includes navigation links for Python, PSF, Docs, PyPI, Jobs, and Community, along with a search bar and social sharing buttons. Below the header, there's a main content area with a dark background containing a code snippet and a sidebar with information about functions. At the bottom, there's a call-to-action message and footer links for Get Started, Download, Docs, and Jobs.

Python

python™

Donate Search GO Socialize

About Downloads Documentation Community Success Stories News Events

```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Functions Defined

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. [More about defining functions in Python 3](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. [» Learn More](#)

---

Get Started

Whether you're new to programming or an experienced developer, it's easy to learn and use Python.

Start with our Beginner's Guide

---

Download

Python source code and installers are available for download for all versions!

Latest: Python 3.9.7

---

Docs

Documentation for Python's standard library, along with tutorials and guides, are available online.

[docs.python.org](https://docs.python.org)

---

Jobs

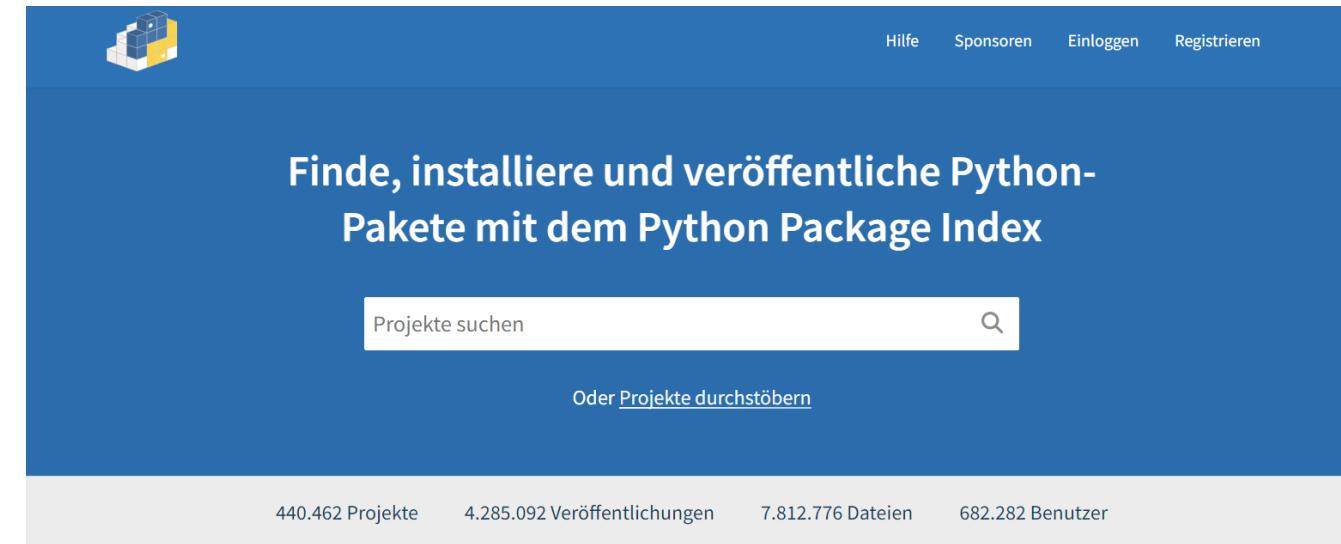
Looking for work or have a Python related position that you're trying to hire for? Our [relaunched community-run job board](#) is the place to go.

[jobs.python.org](https://jobs.python.org)

# PYPI.ORG

Python-Paketindex

[\[Mehr Details\]](#)



The screenshot shows the PyPI homepage with a dark blue header. On the left is the Python logo, and on the right are links for Hilfe, Sponsoren, Einloggen, and Registrieren. The main title "Finde, installiere und veröffentliche Python-Pakete mit dem Python Package Index" is centered. Below it is a search bar with the placeholder "Projekte suchen" and a magnifying glass icon. A link "Oder Projekte durchstöbern" is also present. At the bottom of the header, there are statistics: 440.462 Projekte, 4.285.092 Veröffentlichungen, 7.812.776 Dateien, and 682.282 Benutzer.



Der Python Package Index (PyPI) ist ein Software-Verzeichnis der Programmiersprache Python.

PyPI hilft dabei, von der Python-Community entwickelte und geteilte Software zu finden und zu installieren. [Learn about installing packages ↗](#).

Paket-Entwickler benutzen PyPI, um ihre Software zu veröffentlichen. [Erfahren sie hier wie sie ihren Python-Code für PyPI vorbereiten. ↗](#)



Hilfe

Über PyPI

Mitwirken bei PyPI

PyPI verwenden



## LINKS (1/2)

Eine Liste mit Links zu ...

- Python
- Lernplattform
- Fehlerhilfe
- Google Colab
- Anaconda

Thema	Link
Python	<a href="#">Welcome to Python.org</a>
	<a href="#">The Python Standard Library — Python 3.9.6 documentation</a>
	<a href="#">Our Documentation   Python.org</a>
Lernplattform	<a href="#">Dashboard   HackerRank</a>
	<a href="#">The Python Tutorial — Python 3.9.6 documentation</a>
	<a href="#">Python Tutorial (w3schools.com)</a>
Google Colab	<a href="#">Python Tutorials – Real Python</a>
	<a href="#">Willkommen bei Colaboratory - Colaboratory (google.com)</a>
	<a href="#">Google Colab: Alles, was Sie wissen müssen - Geekflare</a>
Anaconda	<a href="#">Google Colab Tutorial for Beginners   Get Started with Google Colab</a>
	<a href="#">Google Colab Introduction. Colab Tutorial. Colab for Beginners. Colab Explained.</a>
	<a href="#">Anaconda   The World's Most Popular Data Science Platform</a>

VIELE  
TUTORIALS  
AUF  
YOUTUBE



## LINKS (2/2)

Eine Liste mit Links zu ...

- Markdown
- Style Guide
- Jupyter Cheat Sheet
- Git/GitHub

Thema	Link
Markdown	<a href="#">Basic Syntax   Markdown Guide</a>
Style Guide	<a href="#">PEP 8 – Style Guide for Python Code   Python.org</a>
	<a href="#">styleguide   Style guides for Google-originated open-source projects</a>
	<a href="#">PEP 8: The Style Guide for Python Code</a>
Cheat Sheet	<a href="#">Jupyter Notebook CheatSheet (edureka.co)</a>
Git/GitHub	<a href="#">(3) Git And GitHub in ~30 Minutes - YouTube</a>
JupyterLab	
Debugger	<a href="#">(6) Visual Debugger for Jupyter Lab/IPython Notebooks   Installation, Code Examples &amp; Debugging - YouTube</a>