

Rainer Hattenhauer



Einfach
einsteigen ohne
Vorkenntnisse

Programmieren lernen mit KI

Coden mit Python, ChatGPT und Co.

- + Programmierung mit KI leicht gemacht:
von ChatGPT bis GitHub Copilot
- + Vibe Coding: schneller zum Ziel durch
natürliche Sprache
- + Mit vielen Projekten: Taschenrechner,
Quiz-App, Wetterstation und mehr



Kapitel 1

Los geht's – dein smarterer Einstieg ins Programmieren

Komplexe Programme erstellen, ohne jahrelang kryptische Programmiersprachen zu pauken oder sich gar stundenlang mit Fehlersuche zu beschäftigen – die KI-Revolution macht's möglich! Tauch ein in eine Welt, in der kreatives Denken wichtiger wird als technische Details, und erlebe, wie sich auch für dich die Tür zu einer neuen Ära des Programmierens öffnet.

Science-Fiction-Fans meiner Generation werden sich an eine Szene aus den Star-Trek-Filmen erinnern: In »Star Trek IV: Zurück in die Gegenwart« aus dem Jahr 1986 spricht ein Mitglied der Enterprise-Crew mit einem Computer aus den 1980er Jahren. In der humorvollen Szene versucht Scotty mit einem (damals hochmodernen) Macintosh-Computer zu sprechen, indem er die Maus wie ein Kommunikationsgerät an seinen Mund hält und »Hello, computer?« sagt. Diese Szene ist deshalb so amüsant, weil sie den Kontrast zwischen der fortschrittlichen Technologie der Star-Trek-Zukunft und den vergleichsweise primitiven Computern der 1980er Jahre hervorhebt.

Seit den 1980er Jahren sind wir der Zukunft ein gutes Stück nähergekommen. Die Kommunikation mit Computern erfolgt heute problemlos über Sprache. Hieß es in den Anfängen der IT: »Computer befreien das Denken vom Rechnen«, so kann man heute einen guten Teil des Denkens an ein neuronales Netz auslagern.

In diesem Buch geht es aber nicht darum, das Denken zu verlernen, ganz im Gegenteil: Ich nehme dich mit auf eine Reise, auf der du Programm(ideen), die in deinem Kopf herumspuken, mithilfe moderner KI¹-Werkzeuge in kürzester Zeit in die Realität umsetzen kannst. Du lernst die Grundlagen des Programmierens und wirst in der Lage sein, viele alltägliche Probleme zu lösen, indem du einfach geschickte Anweisungen in natürlicher Sprache formulierst.

1 Wichtige Fachbegriffe werden im Glossar am Ende des Buches in einfacher Sprache erklärt. Damit wird sichergestellt, dass der Lesefluss nicht durch ein Übermaß an technischen Erklärungen gestört wird. Mehr zum didaktischen Aufbau des Buches siehe Abschnitt 1.4, »Der »Beipackzettel« zum Buch«.

1.1 Warum KI das Programmieren revolutioniert

Die Geschichte der Programmierung ist eine spannende Entwicklung auf immer höheren Abstraktionsebenen. In den Anfängen der Computerprogrammierung mussten die Entwickler*innen noch direkt mit der *Hardware* kommunizieren. Heute hingegen ist es möglich, der künstlichen Intelligenz in natürlicher Sprache zu erklären, was ein Programm tun soll. Die Reise durch die Zeit und die verschiedenen Abstraktionsebenen ist nicht nur faszinierend, sondern zeigt auch, wie der technologische Fortschritt uns immer schneller an die eigentliche Problemlösung bringt – weg von überflüssigem technischem Ballast. Nebenbei lernst du einige der bedeutendsten Programmiersprachen kennen.

1.1.1 Von der Maschinensprache zur natürlichen Sprache

Die Programmierer*innen der 1940er und 1950er Jahre mussten ihre Anweisungen direkt in binärem Maschinencode formulieren – das ist eine für die *CPU* verständliche Abfolge von Nullen und Einsen. »Strom an« und »Strom aus« entsprachen den Signalen »1« und »0«. Eine einfache Addition von zwei Zahlen erfordert mehrere Zeilen dieses unverständlichen Codes. Stell dir vor, du müsstest folgende Zeichenfolge tippen, nur um zwei Zahlen zu addieren:

```
0101 0001 0000 0001
0101 0010 0000 0010
0000 0011 0001 0010
```

Listing 1.1 Programmbeispiel in Maschinencode: Addition zweier Zahlen

Was geschieht hier? Zunächst werden zwei Zahlen, z. B. die 5 und die 7, in das binäre Zahlenformat umgewandelt. In diesem Format bestehen die Zahlen nur aus Nullen und Einsen. Auch der Befehl, die beiden Zahlen zu addieren, ist eine Folge von Nullen und Einsen. Ganz schön mühsam, oder? Ein kleiner Tippfehler und das gesamte Programm funktioniert nicht ...

Mit dem Aufkommen der Assemblersprachen in den 1950er Jahren erfolgte der erste Abstraktionssprung. Statt binärer Zahlenkolonnen konnten Programmierer*innen nun symbolische Befehle wie `MOV`, `ADD` oder `JMP` verwenden, die dann vom Assembler in Maschinencode übersetzt wurden:

```
MOV AX, 5 ; Lade die Zahl 5 in das Register AX
MOV BX, 7 ; Lade die Zahl 7 in das Register BX
ADD AX, BX ; Addiere BX zu AX
```

Listing 1.2 Programmbeispiel in Assembler: Addition zweier Zahlen

Trotz dieser verbesserten Lesbarkeit blieb die Programmierung hardwarenah und erforderte detaillierte Kenntnisse der Prozessorarchitektur.

In den 1960er und 1970er Jahren entstanden höhere Programmiersprachen wie FORTRAN, COBOL und schließlich C, die einen weiteren Abstraktionssprung ermöglichten. Imperative Sprachen wie C ermöglichten die Formulierung komplexerer Anweisungen, die mehrere Maschinenoperationen gleichzeitig darstellten:

```
int summe = 5 + 7;
```

Listing 1.3 Programmbeispiel in C: Addition zweier Zahlen

Eine einzige Zeile in C erzeugt mehrere Assembler-Instruktionen, was damals eine erhebliche Steigerung der Produktivität und Lesbarkeit bedeutete.

Die 1980er und 1990er Jahre brachten die objektorientierte Programmierung (OOP) mit Sprachen wie C++ und Java. Dabei wurde versucht, den Programmcode durch die Einführung von anschaulichen Objekten der menschlichen Denkweise anzupassen. Ein Objekt in diesem Zusammenhang könnte ein bestimmtes Auto sein, z. B. ein Sportwagen. Der Sportwagen hat bestimmte Eigenschaften von einer übergeordneten Klasse (Auto) geerbt, z. B. vier Räder und einen Motor. Dieser Paradigmenwechsel ermöglichte es, Software als ein Zusammenspiel von Objekten zu modellieren, die Daten und Funktionen verkapseln:²

```
class Rechner {
    public int addiere(int a, int b) {
        return a + b;
    }
}
// Verwendung
Rechner meinRechner = new Rechner();
int ergebnis = meinRechner.addiere(5, 7);
```

Listing 1.4 Programmbeispiel in Java: Addition zweier Zahlen

² Du verstehst bei diesem kryptischen Code nur Bahnhof? Keine Sorge, in diesem Buch zeige ich dir, wie du leistungsfähige Programme erstellen kannst, ohne die tiefere Bedeutung der Programmierbefehle zu kennen.

Anstatt sich auf einzelne Anweisungen zu konzentrieren, konnten Programmierer nun in Begriffen wie Klassen, Objekten und deren Beziehungen denken – eine weitere Abstraktionsebene.

Die 2000er und 2010er Jahre brachten mit deklarativen und funktionalen Programmierkonzepten weitere Abstraktionen mit sich. Anstatt zu beschreiben, *wie* etwas berechnet werden sollte, konnten Entwicklerinnen jetzt einfach deklarieren, *was* berechnet werden sollte. Dazu bietet sich die Programmiersprache *Python* an, die wir in diesem Buch hauptsächlich zur Umsetzung von Projekten verwenden werden, denn ganz ohne Code geht es leider auch heute (noch) nicht.

```
# Imperativer Ansatz in Python
zahlen = [1, 2, 3, 4, 5]
quadrate = []
for zahl in zahlen:
    quadrate.append(zahl * zahl)
```

Listing 1.5 Programmbeispiel in Python: Berechnung der Quadratzahlen von 1 bis 5

Oh weh! Ist das alles kompliziert ... Aber halt! Bevor du dieses Buch in hohem Bogen wegwirfst, sollst du im nächsten Abschnitt erfahren, wie die KI beim Programmieren zum echten Gamechanger wird.

1.1.2 Der KI-Quantensprung

Mit der Einführung sogenannter *Large Language Models (LLM)* und darauf aufbauender Chatbots wie ChatGPT, Claude oder Gemini hat die Abstraktionsebene einen echten Quantensprung erfahren. Erstmals in der Geschichte der Programmierung können wir unsere Anforderungen in natürlicher Sprache formulieren:

»Schreibe ein Programm, das die Quadratzahlen von 1 bis 5 berechnet!«

Und das KI-System generiert im Handumdrehen den entsprechenden Code:

```
# Programm zur Berechnung der Quadratzahlen von 1 bis 5
for i in range(1, 6):
    quadrat = i * i
    print(f"Die Quadratzahl von {i} ist {quadrat}")
```

Listing 1.6 Berechnung der Quadratzahlen von 1 bis 5 – diesmal per KI-Prompt generiert. Beachte, dass der Code automatisch in Python generiert wird, falls nichts anderes angegeben wird.

Diese Entwicklung ist revolutionär, denn sie hebt die Kommunikation mit dem Computer auf eine (zwischen-)menschliche Ebene. Wir müssen nicht mehr in der Sprache der Maschine denken, sondern können in unserer eigenen Ausdrucksweise bleiben.

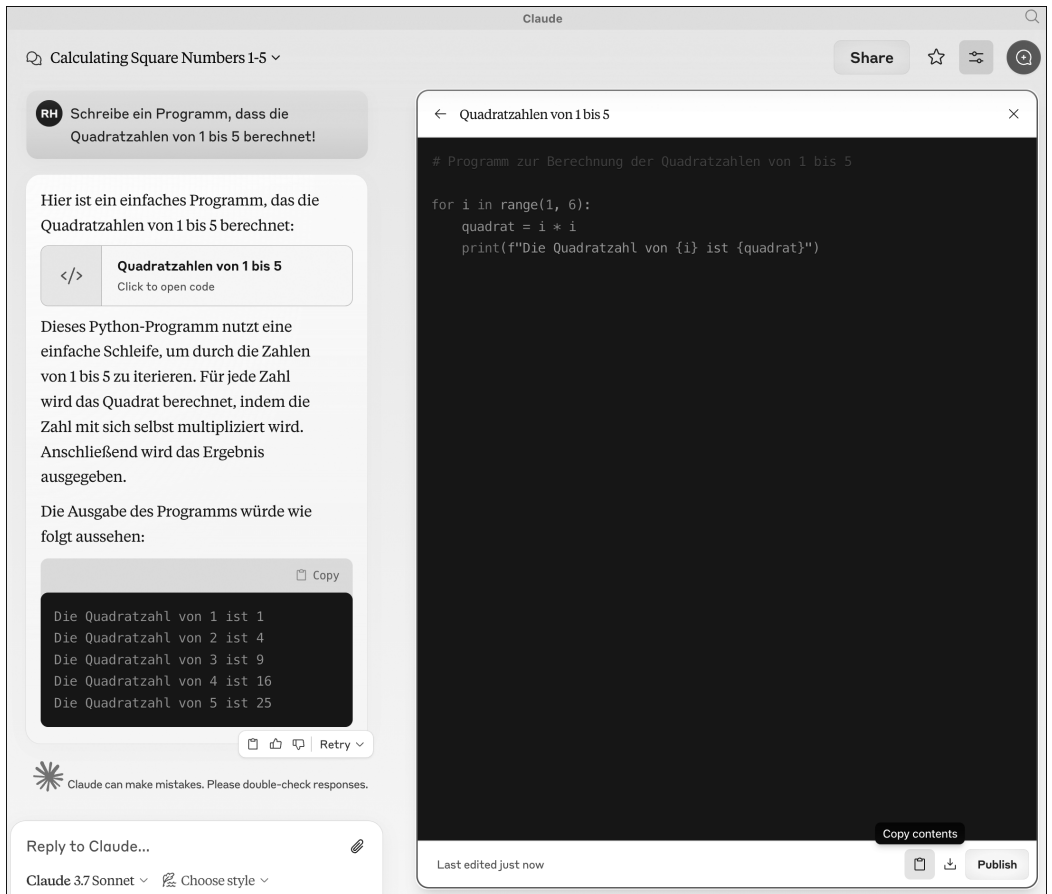


Abbildung 1.1 Ein KI-Chatbot – hier »Claude« von Anthropic – generiert Python-Code auf Zuruf.

Diese KI-gestützte Programmierung ist in mehreren Punkten bahnbrechend:

- **Demokratisierung der Programmierung:** Menschen ohne formale Programmierkenntnisse können nun Software entwickeln, indem sie ihre Anforderungen einfach beschreiben. Die technische Hürde zum Einstieg sinkt dramatisch.

- ▶ **Fokus auf dem Was, nicht dem Wie:** Entwickler*innen können sich auf die eigentliche Problemlösung konzentrieren, anstatt sich mit syntaktischen Details aufzuhalten. Die Arbeit wird konzeptueller und lösungsorientierter.
- ▶ **Effizienzsteigerung:** Routineaufgaben, die früher viel Zeit in Anspruch nahmen, können binnen Sekunden erledigt werden. Die Produktivität steigt dadurch enorm.
- ▶ **Neue Möglichkeiten des Lernens:** Die KI kann nicht nur Code generieren, sondern diesen auch erklären und damit als interaktiver Tutor fungieren. Sie beschleunigt das Verständnis von Programmierkonzepten.

Bedeutet diese Entwicklung, dass traditionelle Programmierer*innen überflüssig werden? Keineswegs! Vielmehr verändert sich ihre Rolle:

- ▶ Anstatt Zeile für Zeile Code zu schreiben, werden sie zu Dirigenten komplexer Systeme.
- ▶ Sie müssen weiterhin verstehen, wie die Software funktioniert, um die Qualität der KI-generierten Programme bewerten zu können.
- ▶ Das Wissen über informatische Konzepte und Algorithmen bleibt unerlässlich – auch in einer KI-gestützten Welt.

KI revolutioniert die Programmierung bei Google

Laut Googles Chief Scientist Jeff Dean generierten KI-Tools Anfang des Jahres 2025 bereits 25 % des Codes in dem großen Tech-Unternehmen.³

Wo stehen wir heute? Die KI-gestützte Programmierung steht noch am Anfang ihrer Entwicklung. Aktuelle KI-Systeme sind beeindruckend, was die Codegenerierung betrifft, haben aber auch ihre Grenzen:

- ▶ Sie verstehen nicht immer perfekt, was wir tatsächlich wollen.
- ▶ Die KI-generierten Programme enthalten manchmal Fehler oder folgen nicht den besten Praktiken, die ein »menschlicher« Programmierer beherrschen würde.
- ▶ Für hochkomplexe, performancekritische oder sicherheitsrelevante Anwendungen ist menschliche Expertise weiterhin unverzichtbar.

Dennoch ist klar: Die Art und Weise, wie wir Software entwickeln, hat sich grundlegend verändert. Das Programmieren wird zugänglicher, effizienter und kreativer – eine wahre Revolution in der Geschichte der Softwareentwicklung.

³ www.indiatoday.in/technology/news/story/googles-jeff-dean-says-ai-doing-25-google-coding-openai-ceo-sam-altman-warns-ai-would-replace-developers-2679509-2025-02-13

In den folgenden Kapiteln wirst du lernen, wie du diese Revolution für dich nutzen kannst, indem du mit KI-Tools programmierst, ohne tiefgreifende Programmierkenntnisse mitbringen zu müssen. Du wirst erleben, wie spannend es ist, deine Ideen in funktionierende Software zu verwandeln – mit der KI als deiner persönlichen Programmierlehrmeisterin. Dabei wünsche ich dir viel Spaß und Erfolg!

1.2 Dein KI-Code-Baukasten – die wichtigsten KI-Assistenten

Wenn du programmieren lernen möchtest, findest du in der Welt der KI-Tools ein wahres Füllhorn an Möglichkeiten. Anstatt dich allein durch die Untiefen kryptischer Programmiersprachen zu kämpfen, kannst du dank moderner KI-Assistenten direkter und intuitiver in die Welt des Programmierens einsteigen. In diesem Abschnitt stelle ich dir einige wichtige Werkzeuge vor, die dir den Einstieg erleichtern und dich auf deinem Lernweg begleiten. Welches der speziellen KI-Tools du letztendlich verwendest, bleibt dir überlassen. Die gute Nachricht ist, dass die meisten dieser Tools in der Basisversion kostenlos sind.

KI-Tools haben die Art und Weise, wie wir programmieren, grundlegend verändert. Sie verstehen die natürliche Sprache und können auf dieser Basis funktionierenden Code in fast allen gängigen Programmiersprachen erzeugen. Tabelle 1.1 gibt einen Überblick über die bekanntesten Vertreter. Alle aufgeführten Bots beherrschen eine Vielzahl von Programmiersprachen. ChatGPT kann einfachen Python-Code direkt im Chatbereich ausführen, Claude kann das Gleiche mit JavaScript tun.

Für Einsteiger sind insbesondere webbasierte KI-Assistenten wie ChatGPT, Claude und Gemini interessant. Sie benötigen keine Installation und bieten einen einfachen Einstieg direkt über den Webbrowser. Sobald du dich mit den Tools im Browser vertraut gemacht hast, kannst du auch die entsprechende spezialisierte Anwendung (= App) eines speziellen Chatbots installieren (siehe Kasten »Der schnelle Weg zum KI-Tool – spezialisierte Apps«). Diese gibt es für viele Betriebssysteme.

ChatGPT hat eine besondere Eigenschaft: Der Bot kann Code nicht nur generieren, sondern diesen auch direkt ausführen. Diese Möglichkeit ist ideal für den Einstieg, wenn du schnell Ergebnisse sehen möchtest. Mit dem eingebauten *Code Interpreter* kannst du somit Python-Programme direkt im Chat ausführen und dir die Ergebnisse anzeigen lassen.

Tool	Beschreibung	Kosten	Link
ChatGPT	Generiert Code in vielen Programmiersprachen, führt Python-Code (mit aktiviertem Code-Interpreter) direkt aus und hilft bei Fehlersuche und Codeerklärungen.	kostenlos (Basis), 22 €/Monat (Plus), 200 €/Monat (Pro)	https://chatgpt.com
Claude	KI-Tool von Anthropic, das natürliche Sprache versteht und Code generieren kann. Kann JavaScript direkt ausführen. Besonders geeignet für komplexe Aufgaben.	kostenlos (Basis), 20 €/Monat (Claude Pro), 30 €/Monat (Teamversion)	https://claude.ai
Gemini	Googles KI-Tool, das Code generieren und optimieren kann. Unterstützt auch Datenanalysen und maschinelles Lernen.	kostenlos (Basis), 22 €/Monat (mit Google-Advanced)	https://gemini.google.com
GitHub Copilot	Bietet intelligente Codevervollständigung und Vorschläge direkt in unterstützten IDEs wie Visual Studio Code.	kostenlos (Basis), 10 €/Monat (Pro-Version für Einzelperson)	https://github.com/features/copilot
Windsurf	Kostenlose, quelloffene Alternative zu GitHub Copilot mit Unterstützung für über 70 Sprachen. Bietet Codevervollständigung und Codegenerierung in der IDE.	kostenlos (Einzelnutzer), 15 €/Monat (Pro-Version)	https://windsurf.com

Tabelle 1.1 KI-Werkzeuge, die als Programmierhilfe verwendet werden können. Die Preise entsprechen dem Stand bei Drucklegung.

Claude von Anthropic zeichnet sich durch die Fähigkeit aus, besonders klaren und gut dokumentierten Code zu erzeugen. Außerdem kann Claude ebenso wie ChatGPT Grafiken und Bilder verstehen – damit kannst du unter anderem ein handgezeichnetes Struktogramm hochladen und in Code umwandeln lassen. Mehr zur Formulierung von Algo-

rithmen in Form von Struktogrammen erfährst du in Abschnitt 3.4, »Der KI-Coach – von der Idee zum Programm«.

Gemini (früher Bard) von Google bietet eine hervorragende Integration in die Google-Dienste. Besonders praktisch ist die Zusammenarbeit mit Google Colab, einem kostenlosen *Jupyter-Notebook*-Dienst, der sich perfekt zum Erlernen von Python eignet. Jupyter Notebook ist ein spezielles Dateiformat, das die Erstellung von Dokumenten ermöglicht, die sowohl Text als auch ausführbaren Programmcode enthalten. So können interaktive Dokumente erstellt werden. Wir beschäftigen uns noch ausführlicher mit Jupyter Notebook in Abschnitt 2.2.

Der schnelle Weg zum KI-Tool – spezialisierte Apps

Fast alle KI-Tools sind auch als eigenständige Apps für die gängigen Betriebssysteme (Windows, macOS, Linux sowie für die mobilen Plattformen iOS und Android) erhältlich. So kannst du den Chatbot nahtlos in das Betriebssystem deiner Wahl integrieren. Such einfach im entsprechenden App Store oder auf der Homepage des Anbieters nach einer App für dein spezielles Betriebssystem.



Abbildung 1.2 Die ChatGPT-App für macOS. Der Python-Code kann direkt in der App ausgeführt werden, sofern er nicht zu komplex ist.

1.3 Dein erstes Programm

Als Programmieranfänger*in wirst du zunächst mit einer alten Tradition vertraut gemacht: Das erste Programm, das du in einer neuen Programmiersprache oder Umgebung schreibst, ist fast immer das sogenannte »Hallo Welt«-Programm. Dieses einfache Programm hat nur eine Aufgabe – den Text »Hallo, Welt!« auf dem Bildschirm auszugeben. So simpel es klingt, so aussagekräftig ist es: Es zeigt dir, ob deine Programmierumgebung richtig eingerichtet ist und du mit ihr später auch komplexere Aufgaben meistern kannst.

Die Tradition des »Hallo Welt«-Programms geht auf das Jahr 1978 zurück, als Brian Kernighan und Dennis Ritchie in ihrem Buch »The C Programming Language« erstmals ein solches Beispielprogramm vorstellten. Seitdem ist es zum Standard geworden, es ist sozusagen dein erster Schritt auf der langen Reise zum Programmierprofi.

1.3.1 Schritt-für-Schritt zu deinem ersten Programm

Beginnen wir mit der Erstellung eines einfachen »Hallo Welt«-Programms mithilfe von KI. Solange du dem Chatbot nichts anderes sagst, werden Programme normalerweise in Python geschrieben. Es kann andererseits nicht schaden, das Schlüsselwort `Python` explizit in den Prompt aufzunehmen. Im Folgenden zeige ich dir, wie verschiedene Chatbots diese Aufgabe lösen und wie du den generierten Code anschließend ausführen kannst.

ChatGPT und Thonny

Um den Vater der generativen KI zum Programmieren zu nutzen, gehst du folgendermaßen vor: Gib die Adresse <https://chatgpt.com> in die Adresszeile deines Browsers ein. Es erscheint ein Fenster mit einer zentralen Prompt-Eingabezeile. Gib darin den folgenden Prompt ein:

Prompt: *Schreibe mir ein einfaches Hallo-Welt-Programm in Python.*

ChatGPT wird daraufhin einen Code ähnlich wie in Abbildung 1.3 generieren.

Nun möchtest du den Code sicher testen und schauen, ob er das richtige Ergebnis liefert. Dazu benötigst du eine einfache Python-Programmierumgebung. Hier hat sich *Thonny* (<https://thonny.org>) im Bereich von Schulung und Ausbildung bewährt. Du findest im Download-Bereich auf der Thonny-Website Installationspakete für alle prominenten Betriebssysteme. Nach der Installation kannst du den vom KI-Bot generierten Code per `[Strg]+[C]` und `[Strg]+[V]` direkt in den Editorbereich kopieren und ihn durch Anklicken der Play-Schaltfläche testen.

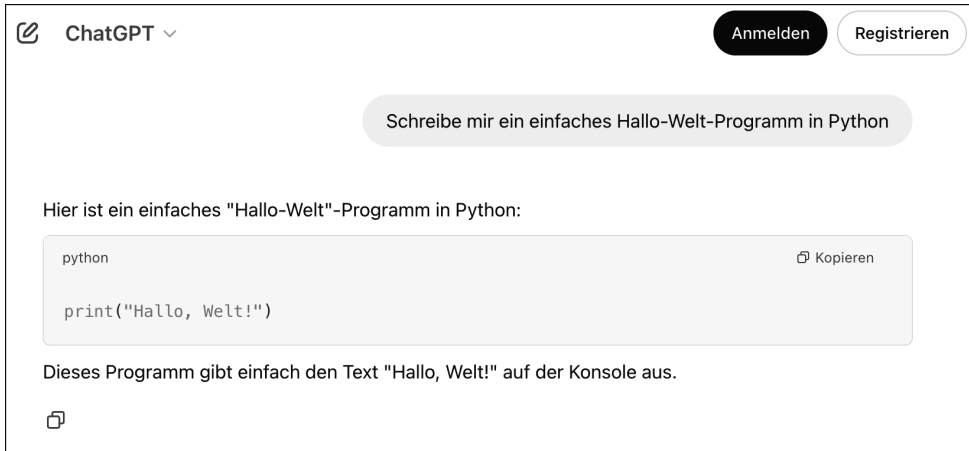


Abbildung 1.3 ChatGPT erzeugt bereits in der kostenlosen Version ohne Registrierung korrekten Code.

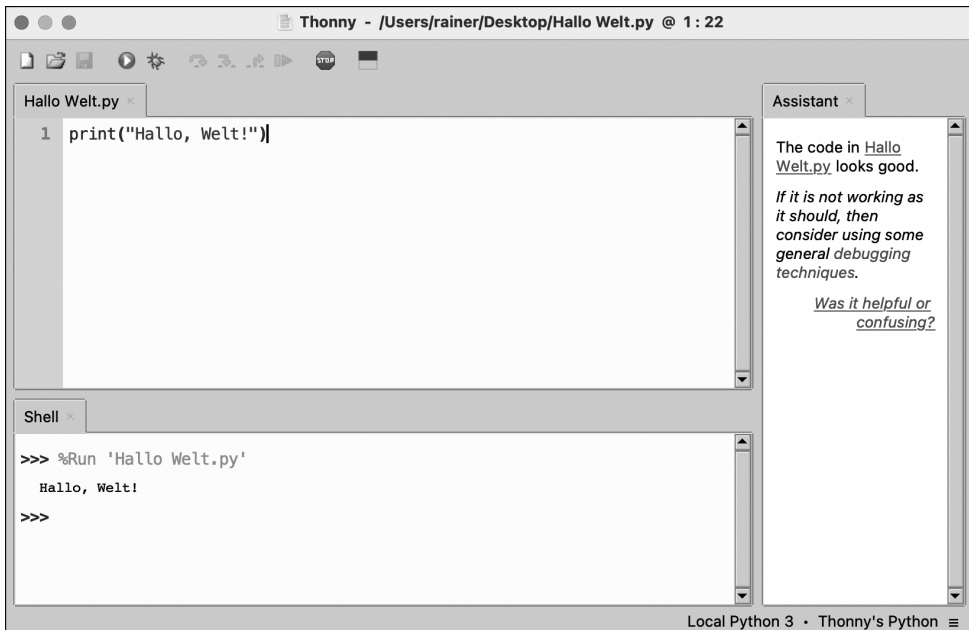


Abbildung 1.4 Der von ChatGPT erzeugte Code wird per Copy and Paste in Thonny eingefügt und kann dort ausgeführt werden.

Herzlichen Glückwunsch! Du hast soeben dein erstes Programm erstellt – und das, ohne eine einzige Zeile Code selbst geschrieben zu haben.

Möchtest du auf deine Chats bzw. Programme in ChatGPT später noch einmal zugreifen, dann empfiehlt es sich, ein (kostenloses) Konto bei OpenAI zu erstellen. Das erreichst du über die Schaltfläche REGISTRIEREN (siehe Abbildung 1.3). Maximalen Nutzen kannst du mit Abschluss eines ChatGPT-Abos erzielen: In der Plus-Version kannst du einfachen Code auch direkt im Chatfenster ausführen lassen (siehe Abbildung 1.5). Dazu musst du lediglich auf die AUSFÜHREN-Schaltfläche im Canvas-Modus klicken. Das Wort »Canvas« steht für »Leinwand« und bedeutet, dass du hier mit dem vorhandenen Code im Dialog mit der KI spielen kannst – genauso wie du es mit Farbe auf einer Leinwand tust. Den Canvas-Modus öffnest du (falls nicht bereits automatisch geschehen) mithilfe des folgenden Prompts:

Prompt: *Öffne den Code in Canvas.*

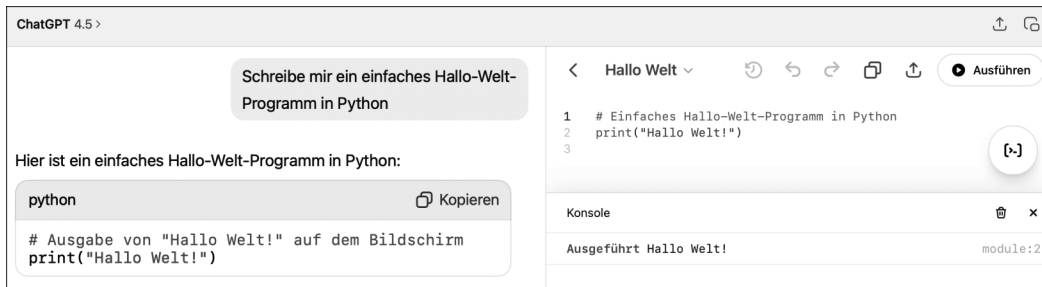


Abbildung 1.5 In der kostenpflichtigen Plus-Version kannst du den Code direkt im Chatbereich ausführen. Dazu muss allerdings zunächst der sogenannte Canvas-Modus aktiviert werden.

Claude und Thonny

Schauen wir uns nun an, wie Claude von Anthropic den Prompt umsetzt. Claude ist wie ChatGPT in der Grundversion kostenlos, eine Registrierung ist jedoch zwingend erforderlich. Begib dich dazu auf die Website <https://claude.ai> und melde dich entweder mit einem Google-Konto oder mit deiner E-Mail-Adresse an. Im letzteren Fall erhältst du einen Link an deine E-Mail-Adresse. Klicke auf diesen Link, um deine Anmeldung zu bestätigen. Anschließend ist es erforderlich, eine Telefonnummer und dein Alter zur Verifizierung anzugeben.

Gib nun in die Claude-Prompt-Zeile den gleichen Prompt wie zuvor bei ChatGPT ein. Das Resultat sollte ähnlich sein (siehe Abbildung 1.6).

Beachte: Bei einigen Chatbots gibt es keine eingebaute Ausführungsfunktion für Code. In diesem Fall musst du den Code, wie oben beschrieben, kopieren und in einer Python-Umgebung deiner Wahl ausführen, z. B. in Thonny.



Abbildung 1.6 »Hallo, Welt!« – umgesetzt mit Claude

Google Colab mit Gemini

Google Colab stellt in Verbindung mit der Google-eigenen KI Gemini eine weitere Alternative für KI-generierten Code dar. Dabei handelt es sich um ein Rundum-sorglos-Paket: Du kannst in Colab Code per KI sowohl generieren als auch ausführen. Colab orientiert sich an der bekannten Jupyter-Notebook-Umgebung. Um Colab optimal nutzen zu können, benötigst du ein Google-Konto. Logg dich damit auf der Seite <https://colab.research.google.com> ein und erstelle dort durch Anklicken der gleichnamigen Schaltfläche ein neues Notebook. Klicke im Notebook rechts oben auf die Schaltfläche GEMINI ANZEIGEN und gib den nun schon oft genutzten Prompt zum Schreiben eines »Hallo Welt«-Programms im Gemini-Chatfenster ein. Anschließend kannst du den generierten Code direkt durch Anklicken des Symbols CODEZEILE HINZUFÜGEN in das Notebook auf der linken Seite übernehmen. Der Code wird schließlich durch Anklicken der Play-Schaltfläche ausgeführt. Das Ergebnis zeigt Abbildung 1.7.

Die Ergebnisse, die du erhältst, sind bemerkenswert konsistent. Alle drei KI-Modelle generieren praktisch identischen Code für diese einfache Aufgabe, was zeigt, dass alle grundlegend als Programmierhilfe geeignet sind. Ich werde alle drei oben vorgestellten Chatbots in verschiedenen Projekten im vorliegenden Buch abwechselnd verwenden. Da die Sprachmodelle in einem stetigen Wettstreit miteinander stehen, kann ich dir an dieser Stelle keinen »Universal Tipp« für *die* beste KI geben. Probiere am besten selbst aus, welche KI dein spezielles Projekt am besten umsetzt.

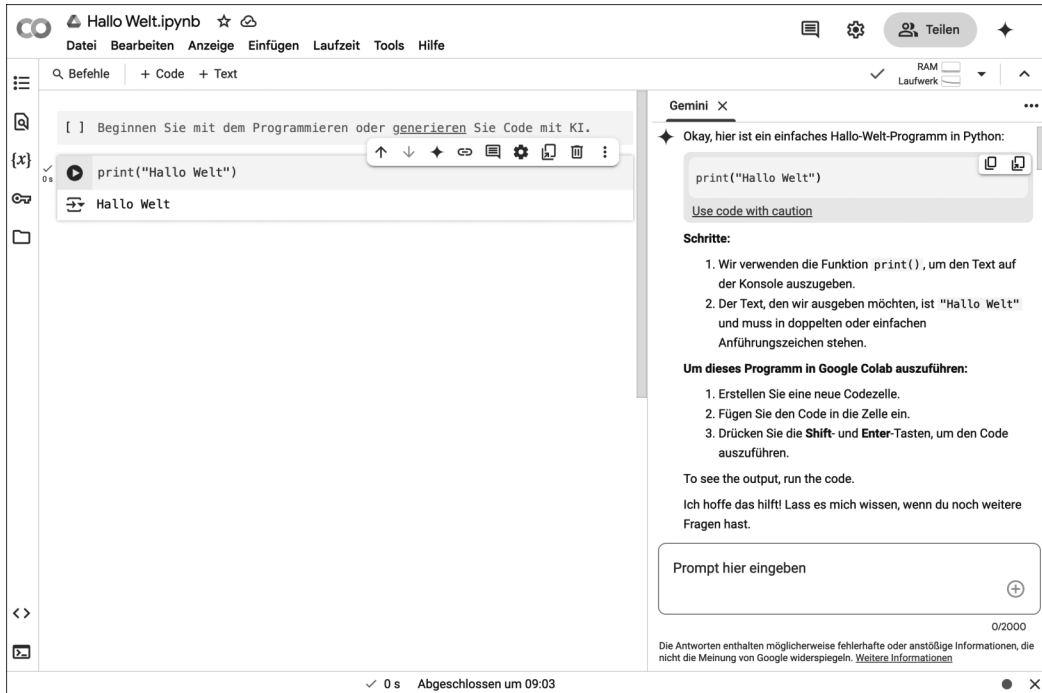


Abbildung 1.7 In der Kombination Google Colab und Gemini steht dir das beliebte Jupyter-Notebook-Format zur Verfügung.

1.3.2 Ist eine KI für »Hallo Welt« überqualifiziert?

Eine berechtigte Frage: Warum sollte man für ein so einfaches Programm eine hoch entwickelte KI einsetzen? Wird da nicht, wie man so schön sagt, »mit Kanonen auf Spatzen geschossen«?

Die Antwort liegt in der Skalierbarkeit. Während das »Hallo Welt«-Programm in seiner Urfassung natürlich äußerst einfach ist, zeigt sich die wahre Stärke textgenerativer KI, wenn die Anforderungen komplexer werden. Schauen wir uns das gleich an einem Beispiel an.

Angenommen, du möchtest »Hallo Welt« nicht einfach nur als Text ausgeben, sondern in Form von *ASCII-Art*. Die Idee ist dabei, eine Zeichenfolge zu erstellen, aus der jeder Buchstabe aus vielen kleinen Versionen dieses Buchstabens dargestellt wird. Das H soll also aus lauter kleinen H's bestehen, das a aus lauter kleineren a's usw.

Probieren wir es mit ChatGPT und dem folgenden Prompt:

Prompt: *Erstelle ein Python-Programm, das »Hallo Welt« als ASCII-Art ausgibt. Jeder Buchstabe der Ausgabe soll aus demselben Buchstaben bestehen, also das H aus H's, das a aus a's usw.*

ChatGPT erzeugt daraufhin ein deutlich komplexeres Programm (siehe Abbildung 1.8).⁴

```

1  def print_hallo_welt():
2      ascii_art = {
3          'H': ["H H", "H H", "HHHHH", "H H", "H H"],
4          'a': ["aaa", "a a", "aaaaa", "a a", "a a"],
5          'l': ["l", "l", "l", "l", "lllll"],
6          'o': ["ooo", "o o", "o o", "o o", "ooo"],
7          'W': ["W W", "W W", "W W W", "WW WW", "W W"],
8          'e': ["eeee", "e", "eeee", "e", "eeee"],
9          't': ["ttttt", "t", "t", "t", "t"]
10     }
11
12     text = "Hallo Welt"
13
14     lines = ['' for _ in range(5)]
15
16     for char in text:
17         if char == ' ':
18             for i in range(5):
19                 lines[i] += ' '

```

```

>>> %Run 'Hallo ASCII.py'
H H aaa l l ooo W W eeee l ttttt
H H a a l l o o W W e l t
HHHHH aaaaa l l o o W W W eeee l t
H H a a l l o o WW WW e l t
H H a a lllll llllll ooo W W eeeee llllll t
>>>

```

Local Python 3 • Thonny's Python

Abbildung 1.8 »Hallo Welt« – dargestellt in ASCII-Art

⁴ Die Quellcodes sämtlicher Beispiele im Buch findest du in den Begleitmaterialien zum Buch (www.rheinwerk-verlag.de/programmieren-lernen-mit-ki/).

Wenn du diesen Code ausführst, siehst du eine beeindruckende ASCII-Art-Version von »Hallo Welt«, bei der jeder Buchstabe aus sich selbst besteht.

Das ist etwas, das zumindest Einsteiger nicht ohne Weiteres programmieren könnten – und genau hier liegt der Wert der KI-gestützten Programmierung. Du kannst komplexe Ideen mit deinen eigenen Worten ganz einfach beschreiben, und die KI übernimmt die technische Umsetzung.

1.3.3 Grüße aus Babylon – »Hallo Welt« in anderen Programmiersprachen

Die Vielseitigkeit der KI-Tools zeigt sich auch darin, dass sie nicht auf eine bestimmte Programmiersprache beschränkt sind. Lassen wir ChatGPT ein animiertes »Hallo Welt« in Form einer *HTML*-Datei erstellen, die *JavaScript* enthält:

Prompt: *Erstelle ein animiertes »Hallo Welt« in HTML mit JavaScript, bei dem die Buchstaben nacheinander erscheinen und in verschiedenen Farben blinken.*

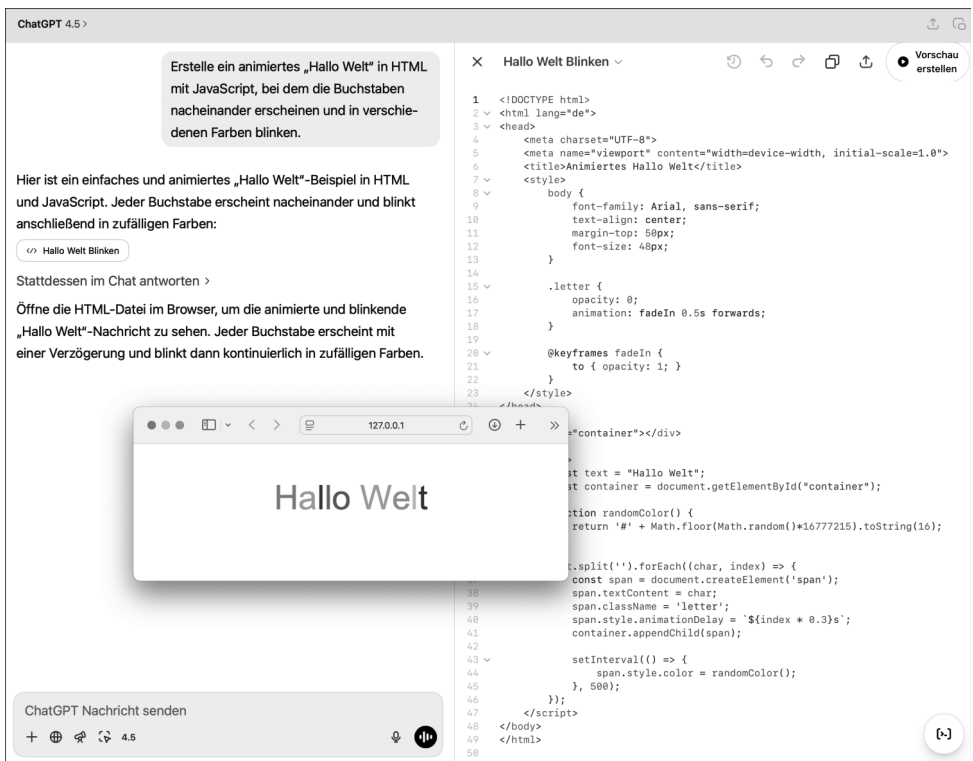


Abbildung 1.9 Von null auf Webentwickler*in innerhalb weniger Sekunden

ChatGPT generiert daraufhin HTML- und JavaScript-Code. Der Code erzeugt eine Webseite, auf der die Buchstaben von »Hallo Welt« nacheinander erscheinen und dann in verschiedenen Farben blinken. Du kannst diesen Code in einen beliebigen Editor kopieren, anschließend in einer Datei mit der Endung *.html* speichern und dann mit einem Webbrowser deiner Wahl öffnen, um das Ergebnis zu sehen.

Was früher nur mit HTML-, CSS- und JavaScript-Kenntnissen möglich war, hast du jetzt mit einem einfachen Prompt umgesetzt. Das zeigt erneut die Stärke der KI-gestützten Programmierung: Sie überbrückt die Kluft zwischen deiner Idee und ihrer technischen Umsetzung, unabhängig von der Programmiersprache.

1.3.4 Codeverstehender – die KI erklärt dir den Code

Ein weiterer Vorteil der KI-gestützten Programmierung ist, dass die KI nicht nur Code erzeugen, sondern ihn dir auch erklären kann. Das ist besonders wertvoll, wenn du den Code verstehen und eventuell selbst anpassen möchtest.

Bitten wir ChatGPT, uns den HTML- und JavaScript-Code aus dem letzten Beispiel zu erklären:⁵

Prompt: *Kannst du mir den HTML/JavaScript-Code für das animierte »Hallo Welt« erklären?*

Ausgabe: Natürlich! Hier erkläre ich dir den HTML/JavaScript-Code Schritt für Schritt:

1. HTML-Struktur:

```
<div id="hello"></div>
```

Dies ist ein leerer Container, in den das animierte »Hallo Welt« eingefügt wird.

2. CSS-Styling und Animationen:

```
body {
```

```
[...]
```

Zusammenfassung der Funktionsweise:

- ▶ Der Text wird in einzelne Buchstaben zerlegt.
- ▶ Jeder Buchstabe erscheint mit einer Verzögerung nacheinander (appear-Animation).

⁵ Codebeispiele und Promptausgaben im Buch werden stets geeignet gekürzt. Kürzungen erkennst du an den Auslassungszeichen [...]. Den vollständigen Code oder ausführliche Antworten auf Prompts findest du in den Begleitmaterialien zum Buch (www.rheinwerk-verlag.de/programmieren-lernen-mit-ki/).

- ▶ Gleichzeitig blinkt jeder Buchstabe dauerhaft in verschiedenen Farben (blink-Animation).
- ▶ Das Ergebnis ist eine animierte Begrüßung, die Aufmerksamkeit erregt und visuell attraktiv wirkt.

Dadurch erhältst du einen Einblick in die Funktionsweise des Codes. Das hilft dir, ein besseres Verständnis für Programmierkonzepte zu entwickeln, unabhängig von der Programmiersprache, die du verwendest.

Ob ein einfaches »Hallo Welt« in Python oder eine komplexe Animation mit HTML und JavaScript – das Erfolgserlebnis, ein funktionierendes Programm binnen kürzester Zeit zu erstellen, ist unbezahlbar. Mit KI-Tools wie ChatGPT, Claude oder Gemini ist dieser Erfolg für alle erreichbar, unabhängig vom Vorwissen.

In diesem Abschnitt hast du gesehen, wie du mit einfachen Befehlen Programme nach deinen Vorstellungen erstellen kannst. Du hast verschiedene Komplexitätsstufen erforscht, von der einfachen Textausgabe bis hin zu Animationen. Und das Beste daran ist, dass du dafür keine Programmiersprache von Grund auf lernen musstest.

Das ist natürlich erst der Anfang. In den folgenden Kapiteln wirst du lernen, komplexere Programme zu erstellen, sie in verschiedenen Umgebungen auszuführen und die Möglichkeiten der KI-gestützten Programmierung voll auszuschöpfen.

Beachte: Der Schlüssel zum erfolgreichen KI-gestützten Programmieren liegt in der klaren Formulierung deiner Ideen. Je präziser deine Anweisungen sind, desto besser wird das Ergebnis sein. Die KI ist dein Werkzeug – und du bist der kreative Kopf dahinter!

1.4 Der »Beipackzettel« zum Buch

Bevor wir nun richtig loslegen, möchte ich dir einen kurzen »Beipackzettel« mit auf den Weg geben. Du kennst das von Medikamenten, diese kleinen, gefalteten Zettel mit wichtigen Informationen zur Anwendung. Hier erfährst du, wie du den maximalen Nutzen aus dem vorliegenden Buch ziehst und die Nebenwirkungen (im Bereich von KI-Tools sind das die gefürchteten Halluzinationen und Redundanzen) möglichst gering halten kannst.

1.4.1 Das pädagogische Konzept

Dieses Buch folgt einem einfachen, aber effektiven Konzept: Du sollst die Grundprinzipien der Programmierung verstehen, ohne dich in den Untiefen der Syntax einzelner Programmiersprachen zu verlieren. Das Ziel ist, dir eine praktische Anleitung an die

Hand zu geben, mit der du zukünftige Probleme, die ein Verständnis von Programmierung und Informatik erfordern, eigenständig lösen kannst. Du wirst zum strategischen Kopf, der die Richtung vorgibt, während die KI die technische Umsetzung übernimmt. Mehr noch, die KI kann dir komplexe theoretische Sachverhalte in einfacher Weise erklären und wird so zu deiner persönlichen Lehrmeisterin. Sämtliche Informationen von ca. 50 Jahren Programmierkunst finden sich im Trainingsdatenbestand der aktuellen großen KI-Sprachmodelle.

Anders ausgedrückt ist dieses Buch kein klassisches Programmierlehrbuch, sondern eher ein Reiseführer, der dich an die schönsten Orte der modernen KI-gestützten Softwareentwicklung bringt. Es zeigt dir, wie du dich binnen kürzester Zeit in unbekannte Softwaretools und *Frameworks* einarbeiten kannst, während du ganz nebenbei die wichtigsten Konzepte der Informatik kennenlernst.

Vibe Coding – die Zukunft des Programmierens?

Wenn du in letzter Zeit die Schlagzeilen der gängigen IT-Nachrichtenportale verfolgt hast, dann bist du sicherlich über das Modewort *Vibe Coding* gestolpert. Was hat es damit auf sich? Vibe Coding (sinngemäß übersetzt »stimmungsbasiertes Programmieren«) ist »Programmieren aus dem Bauch heraus«. Du beschreibst einer KI ein Programmierproblem in Umgangssprache und diese liefert dir sofort den Code in einer Programmiersprache deiner Wahl. Zum Schluss kopierst du den Code in die Entwicklungsumgebung deiner Wahl, und fertig ist dein Programm. Aber ist es wirklich so einfach?

Hier müssen wir kritisch sein: Ohne grundlegende Kenntnisse informatischer Prozesse und Algorithmen wird Programmieren zum reinen Glücksspiel. Der Leitspruch einiger Expert*innen »2 Minuten Coden – 20 Stunden Fehlersuche« hat sich leider viel zu oft als wahr erwiesen. In diesem Buch versuche ich daher, den goldenen Mittelweg zu gehen: Ich werde dich mit den wichtigsten Konzepten der Informatik vertraut machen und du wirst lernen, anfängerfreundlichen, selbsterklärenden Code zu erzeugen, ohne dass du zu sehr in die Feinheiten der jeweiligen Programmiersprache eintauchen musst.

Du hast in diesem Kapitel bereits erfahren, dass es ganz ohne Code auch in dieser neuen Ära (noch) nicht gehen wird. Für die praktische Umsetzung der Programme verwenden wir einige Programmierwerkzeuge, die ich dir im nächsten Kapitel vorstellen werde:

- ▶ **Python** als Hauptprogrammiersprache, da sie relativ einfach zu lesen ist und von allen aktuellen KI-Tools hervorragend unterstützt wird
- ▶ **Visual Studio Code** als Entwicklungsumgebung für größere Projekte (wie in Abschnitt 2.1 beschrieben)
- ▶ **Jupyter Notebook** bzw. **Google Colab** für anspruchsvollere Programmieraufgaben (siehe Abschnitt 2.2)

Du wirst den Code nicht von Grund auf selbst schreiben müssen – die KI wird das für dich tun. Aber du solltest in der Lage sein, diesen Code zu verstehen, zu modifizieren und in den oben genannten Umgebungen auszuführen. Weiterhin findest du sämtliche Programmbeispiele auf der dieses Buch begleitenden Webseite (www.rheinwerk-verlag.de/programmieren-lernen-mit-ki/).

1.4.2 Der menschliche Vorteil: Kreativität

Während KI-Tools immer besser darin werden, Code zu generieren, bleibt die Kreativität die ureigene Domäne des Menschen. Dein Vorteil liegt darin, kreative Ideen zu entwickeln und Probleme so zu formulieren, dass die KI sie versteht.

Die KI sorgt dann für die professionelle Umsetzung deiner Ideen, ohne dass du dich in technischen Details verlieren musst. Du konzentrierst dich auf das Was und Warum, die KI kümmert sich um das Wie. Diese Kombination aus menschlicher Kreativität und KI-gestützter Umsetzung ist die Zukunft der Softwareentwicklung – und hier lernst du, wie du dieses Potenzial voll ausschöpfen kannst.

Apropos KI-Unterstützung: Auch dieses Buch wurde unter Zuhilfenahme verschiedener KI-Tools erstellt. Unter anderem habe ich die folgenden Tools verwendet:

- ▶ **Claude** von Anthropic und **ChatGPT** von OpenAI zur Erstellung von Gliederungen und ersten Entwürfen für Teilkapitel. Die beiden Werkzeuge haben meine Grundideen geordnet und mit eigenen Inhalten ergänzt.
- ▶ **Gemini** von Google zur Erstellung von komplexen Programmbeispielen aus dem Bereich der Hardware
- ▶ **DeepL Write** für den orthografischen und grammatikalischen Feinschliff des Manuskripts
- ▶ **ChatGPT** und **Claude**, um komplexe Sachverhalte zu erklären, das Glossar am Ende des Buches zu erstellen und Fehler im Programmcode zu analysieren
- ▶ **Perplexity** – das »bessere Google« für die Suche nach aktuellen Techniken, Themen und Innovationen im Bereich der Programmierung mit KI-Tools

Ich habe also selbst viel praktische Erfahrung mit dem Prinzip, das ich in diesem Buch vermittele: die Kombination von menschlicher Kreativität und KI-Unterstützung. Die konkrete Umsetzung der Texte habe ich jedoch immer selbst in die Hand genommen, um die größtmögliche Kontrolle über die Inhalte zu behalten.

1.4.3 Die Inhalte – ein kurzer Überblick

Im Folgenden bekommst du einen kurzen Überblick, was dich in den nächsten Kapiteln erwartet:

Kapitel 2: Die perfekte Basis – deine Entwicklungsumgebung

Hier lernst du die wichtigsten Programmierumgebungen kennen und einrichten. Du erfährst, wie du Visual Studio Code installierst und die Python-Unterstützung hinzufügst. Außerdem lernst du die Vorteile von Jupyter Notebook und Google Colab für große, interaktive Projekte kennen und erfährst, wie du KI-Tools direkt in deine Entwicklungsumgebung integrieren kannst.

Kapitel 3: Einfach(e) Programme mit KI schreiben (lassen)

Dieses Kapitel führt dich in die Grundlagen der Programmierung ein, vom einfachen Taschenrechner bis hin zum Text-Adventure. Du lernst die Bedeutung von Variablen, Verzweigungen und Schleifen kennen, ohne dich in der Syntax zu verlieren. Du erfährst auch, wie du KI als Coach nutzen kannst, um von der Idee zum fertigen Programm zu gelangen.

Kapitel 4: Programmieren mit System

Hier tauchst du tiefer in die Welt der strukturierten Programmierung ein. Du lernst, wie du mit Listen und komplexeren Datenstrukturen umgehst, wie du mit Funktionen und Prozeduren deinen Code strukturierst und wie du grafische Benutzeroberflächen erstellst. Das Kapitel zeigt dir auch, wie du mit der Programmierung von Computergrafiken faszinierende visuelle Ergebnisse erzielst.

Kapitel 5: Daten und ihre Visualisierung

In diesem Kapitel lernst du, wie KI dir bei der Analyse und Visualisierung von Daten helfen kann. Du lernst, wie du KI für Excel-Aufgaben einsetzen kannst, wie du mit Python beeindruckende Diagramme erstellst und wie du Webdaten für deine Projekte nutzen kannst. Praktische Beispiele wie die Abfrage von Aktienkursen und Wettervorhersagen machen die Konzepte greifbar.

Kapitel 6: Hardware und Sensoren erforschen

Hier verlässt du die rein digitale Welt und verknüpfst deine Programmierkenntnisse mit echter Hardware. Du lernst den Calliope mini und den Arduino kennen, programmierst

Sensoren und Aktoren und realisierst spannende physikalische Projekte wie Wetterstationen oder Mini-Alarmanlagen. Die KI hilft dir dabei, den nötigen Code zu generieren.

Kapitel 7: Künstliche Intelligenz selbst programmieren

Hier dreht sich alles um die KI selbst. Du lernst die Grundlagen des maschinellen Lernens (ML) kennen, erstellst deine eigenen neuronalen Netze und trainierst sogar deine eigenen KI-Modelle. Vom einfachen Bildklassifikator bis hin zum eigenen Chatbot erlebst du, dass KI nicht nur ein Werkzeug, sondern auch ein faszinierendes Forschungsfeld ist.

Kapitel 8: Fortgeschrittene Techniken

Dieses Kapitel führt dich in komplexere Konzepte wie objektorientierte Programmierung, Datenbanken und Anwendungsentwicklung ein. Mithilfe der KI wirst du in der Lage sein, anspruchsvollere Projekte umzusetzen. Von der Datenverwaltung bis hin zur Entwicklung mobiler Apps – die KI nimmt dir die kniffligen technischen Details ab.

Kapitel 9: Tipps und Tricks

Im letzten Kapitel erhältst du wertvolle Tipps zur Fehlersuche und zum Optimieren mit KI. Du erfährst, wie du den bestehenden Code analysieren und verbessern kannst, wie du Code in andere Programmiersprachen übersetzen kannst und wo du weitere Ressourcen findest, um dein Wissen zu vertiefen.

Jetzt, wo du weißt, was auf dich zukommt, kann die eigentliche Reise beginnen. Lass uns in die spannende Welt der KI-gestützten Programmierung eintauchen!

... und schließlich noch eine Danksagung

Mein besonderer Dank gilt Erik Lipperts vom Rheinwerk Verlag, der nicht nur an das Projekt geglaubt, sondern mit mir gemeinsam das Konzept entwickelt hat. Sein Gespür dafür, was Einsteiger*innen wirklich brauchen, hat das Buch entscheidend geprägt.

Michael Kofler danke ich für das sorgfältige fachliche Lektorat und die vielen klugen Anmerkungen zum Manuskript. Es ist wohl keine Übertreibung zu sagen, dass Michael Koflers Bücher mein Wissen über IT nachhaltig geprägt haben.

wirst du am Ende dieses Kapitels in der Lage sein, eigene kleine Programme zu entwickeln.

3.1 Taschenrechner & Co.

Es ist an der Zeit, etwas Spannenderes zu programmieren, als nur die Worte »Hallo Welt« auf dem Bildschirm auszugeben. In diesem Kapitel tauchen wir tiefer in die Grundlagen der Programmierung ein und du erstellst deine ersten praktischen Programme: einen BMI-Rechner und einen Taschenrechner. Dabei lernst du wichtige Konzepte wie Variablen, Datentypen und Benutzerinteraktionen kennen. Prinzipiell gehst du dabei so vor, dass du mit dem Chatbot in einen Dialog trittst und dieser dir dann bestimmte Konzepte erklärt.

3.1.1 Vom Umgang mit Variablen

Fragen wir doch dazu einfach den Chatbot, was es mit diesen mysteriösen Variablen auf sich hat:¹

Prompt: *Wozu verwendet man Variablen beim Programmieren und wofür sind sie gut?*

Ausgabe: Variablen sind eines der fundamentalsten Konzepte in der Programmierung. Stell dir eine Variable wie eine beschriftete Box vor, in der du einen Wert speichern kannst. Diese Box hat einen Namen (den Variablennamen) und du kannst den Inhalt dieser Box jederzeit ändern oder abrufen.

In der Mathematik kennst du Variablen vielleicht als x oder y in Gleichungen wie $y = 2x + 3$. In der Programmierung sind Variablen noch vielseitiger: Sie können nicht nur Zahlen, sondern auch Text, Listen oder komplexere Datenstrukturen enthalten.

Wofür sind Variablen gut? Sie ermöglichen es dir:

- ▶ Werte zu speichern und wiederzuverwenden
- ▶ Mit Werten zu rechnen
- ▶ Werte zu verändern
- ▶ Programmabläufe zu steuern

¹ Ein grundlegendes Prinzip in diesem Buch wird es sein, sich bestimmte Sachverhalte aus dem Bereich der Informatik direkt vom Chatbot erklären zu lassen. Auf diese Weise lernst du, wie du dir später auch in anderen Bereichen selbstständig Wissen aneignen kannst. Die Antworten des Chatbots wurden in den Beispielen 1 : 1 übernommen. Dabei wurden eventuell auftretende Rechtschreib- oder Grammatikfehler nicht korrigiert.

Aha! Das klingt aufschlussreich. Bohren wir weiter nach:

Prompt: *Kannst du mir ein Beispiel für die Verwendung von Variablen in Python geben?*

Ausgabe: Gerne! Betrachte das folgende Beispiel:

```
# Eine Variable namens 'alter' mit dem Wert 25 erstellen
alter = 25
# Den Wert der Variable ausgeben
print("Mein Alter ist:", alter)
# Den Wert der Variable ändern
alter = 26
# Den neuen Wert ausgeben
print("Nach meinem Geburtstag bin ich", alter, "Jahre alt")
```

In diesem Beispiel haben wir eine Variable `alter` erstellt und ihr zunächst den Wert 25 zugewiesen. Später haben wir den Wert auf 26 geändert. Die Variable fungiert als Container, der einen Wert speichert und es uns ermöglicht, diesen Wert an verschiedenen Stellen im Programm zu verwenden und zu ändern.

Prompt: *Ok. Gibt es eigentlich irgendwelche konkreten Regeln, wie Variablen in Python zu benennen sind?*

Ausgabe: Variablennamen in Python sollten einigen Regeln folgen:

- ▶ Sie dürfen nur Buchstaben, Ziffern und Unterstriche enthalten
- ▶ Sie dürfen nicht mit einer Ziffer beginnen
- ▶ Sie unterscheiden zwischen Groß- und Kleinschreibung (alter und Alter wären zwei verschiedene Variablen)
- ▶ Sie sollten keine reservierten Wörter sein (wie if, for, while etc.)

Eine gute Praxis ist es, Variablennamen zu wählen, die aussagekräftig sind und den Zweck der Variable beschreiben. Also lieber `benutzer_alter` statt nur `a`.

Übung

Nun bist du an der Reihe: Erstelle mithilfe eines KI-Bots ein Python-Programm, das den Anwender nach Alter, Körpergröße und Gewicht fragt. Welche Befehle werden im Programm verwendet, um die entsprechenden Daten vom Benutzer einzulesen? Lass dir diese Befehle von der KI erklären.

Eventuell stößt du bereits an dieser Stelle auf ein Problem, das mit der Nutzung von text-generativen KI-Tools zur Programmierung einhergeht: Der Code ist für Einsteiger*innen oft recht kompliziert. Das kannst du dadurch verhindern, indem du den Bot von vornherein bittest, einfachen Code zu erzeugen, z. B. mit folgendem Prompt:

Prompt: *Erstelle ein sehr einfaches Python-Programm, das den Anwender nach Alter, Körpergröße und Gewicht fragt und diese Daten anschließend ausgibt.*

Ausgabe:

```
# Einfaches Programm zur Abfrage persönlicher Daten
# Eingabe der Daten vom Benutzer
alter = input("Bitte gib dein Alter ein: ")
körpergröße = input("Bitte gib deine Körpergröße in cm ein: ")
gewicht = input("Bitte gib dein Gewicht in kg ein: ")
# Ausgabe der eingegebenen Daten
print("\nDeine eingegebenen Daten:")
print(f"Alter: {alter} Jahre")
print(f"Körpergröße: {körpergröße} cm")
print(f"Gewicht: {gewicht} kg")
```

Lassen wir uns schließlich noch die Befehle zur Ein- und Ausgabe etwas genauer erläutern:

Prompt: *Erläutere mir den prinzipiellen Aufbau der Eingabe- und Ausgabebefehle.*

Ausgabe: Gerne erläutere ich dir den prinzipiellen Aufbau der Eingabe- und Ausgabebefehle in Python:

Eingabebefehl: `input()`

Der `input()`-Befehl hat folgenden Aufbau:

```
variable = input("Aufforderungstext: ")
```

Elemente:

1. `variable`: Ein selbst gewählter Name, unter dem die Eingabe gespeichert wird
2. `input()`: Die Funktion, die eine Benutzereingabe erwartet
3. "Aufforderungstext": Ein optionaler Text in Anführungszeichen, der dem Benutzer anzeigt, was eingegeben werden soll

4. Nach Ausführung wartet das Programm auf eine Eingabe, die mit der Enter-Taste abgeschlossen wird

[...]²

3.1.2 Was sind Variablentypen?

In der realen Welt gibt es verschiedene Arten von Daten: Zahlen, Text, Ja/Nein-Werte und vieles mehr. In der Programmierung spiegelt sich diese Vielfalt in den verschiedenen Datentypen wider. Jede Variable in Python hat einen bestimmten Datentyp, der festlegt, welche Art von Daten sie speichern kann und welche Operationen mit ihr durchgeführt werden können.

Auch hier kann dir ein Chatbot helfen, einen Überblick über die Variablentypen unter Python zu erhalten:

Prompt: *Welche Variablentypen gibt es in Python?*

Ausgabe: In Python gibt es folgende Variablentypen:

1. Numerische Typen:
 - int: Ganzzahlen (z. B. 42, -7)
 - float: Gleitkommazahlen (z. B. 3.14, -2.5)
 - complex: Komplexe Zahlen (z. B. 1+2j)
2. Sequenzen:
 - str: Zeichenketten/Strings (z. B. »Hallo«)
 - list: Listen, veränderbar (z. B. [1, 2, 3])
 - tuple: Tupel, unveränderbar (z. B. (1, 2, 3))
 - [...]
 - None: Repräsentiert den Wert »Nichts« oder »Keine Wert«

In Python kann man den Typ einer Variable mit der Funktion `type()` überprüfen.

Bezüglich der Variablentypen gibt es eine kleine Entwarnung: Du musst dir – anders als in C oder Java – in der Regel am Anfang eines Programms keine großen Gedanken über die Deklaration und Zuweisung von Variablentypen machen. Das weiß auch der KI-Bot:

² Die Auslassungspunkte zeigen an, dass die Antwort des Chatbots im Buch nicht vollständig wiedergegeben wurde. Die vollständigen Antworten finden sich im Begleitmaterial in der Datei, die alle Prompts des Buches und die entsprechenden Antworten des Chatbots zum Zeitpunkt der Drucklegung enthält.

Ausgabe: Python ist eine dynamisch typisierte Sprache. Das bedeutet, dass du den Datentyp einer Variable nicht explizit angeben musst – Python erkennt den Typ automatisch anhand des zugewiesenen Werts. Außerdem kann sich der Typ einer Variable während der Programmausführung ändern:

```
x = 10          # x ist jetzt ein Integer
x = "Hallo"    # x ist jetzt ein String
```

Andererseits ist die Kenntnis von Variablentypen bei (Rechen-)Operationen wichtig:

Ausgabe: Die Kenntnis der Datentypen ist wichtig, weil verschiedene Operationen nur mit bestimmten Typen funktionieren. Zum Beispiel kannst du mathematische Operationen nur mit numerischen Typen (`int`, `float`) durchführen, während String-Methoden nur auf Strings anwendbar sind.

3.1.3 Wie verarbeitet man Zahlen und Texte?

Nachdem du nun weißt, was Variablen und Datentypen sind, schauen wir uns an, wie du mit Zahlen und Texten in Python arbeiten kannst. Beginnen wir mit einfachen Rechenoperationen:³

Prompt: *Wie kann ich mit Variablen in Python rechnen?*

Ausgabe: Python unterstützt alle üblichen mathematischen Operationen:

```
# Addition
summe = 5 + 3 # summe ist 8
# Subtraktion
differenz = 10 - 4 # differenz ist 6
# Multiplikation
produkt = 6 * 7 # produkt ist 42
# Division (ergibt immer eine Fließkommazahl)
quotient = 20 / 4 # quotient ist 5.0
# Ganzzahlige Division (ergibt immer eine ganze Zahl)
ganzzahl_quotient = 20 // 3 # ganzzahl_quotient ist 6
# Modulo (Rest bei Division)
```

³ Beachte: Die Hashtag-Zeichen im Listing markieren in Python den Anfang eines Kommentars.

```
rest = 20 % 3 # rest ist 2
# Potenz
potenz = 2 ** 3 # potenz ist 8 (2 hoch 3)
```

Du kannst auch komplexere mathematische Ausdrücke bilden, wobei die üblichen Vorrangregeln gelten (Klammern, Potenz, Multiplikation/Division, Addition/Subtraktion):

```
ergebnis = (10 + 5) * 2 / 5 # ergebnis ist 6.0
```

Für speziellere mathematische Funktionen wie Wurzeln, Logarithmen oder trigonometrische Funktionen kannst du das `math`-Modul verwenden:

```
import math
# Quadratwurzel
wurzel = math.sqrt(16) # wurzel ist 4.0
# Sinus (im Bogenmaß)
sinus = math.sin(math.pi / 2) # sinus ist 1.0
# Runden
gerundet = round(3.7) # gerundet ist 4
```

Und schließlich schauen wir uns noch an, wie Python mit Zeichenkettenfolgen umgeht:

Prompt: *Erkläre mir, welche Operationen in Python mit Zeichenketten möglich sind.*

Ausgabe: Strings (Texte) werden in Python in einfachen oder doppelten Anführungszeichen geschrieben. Hier sind einige Operationen, die du mit Strings durchführen kannst:

```
# Verkettung (Concatenation)
vorname = "Max"
nachname = "Mustermann"
voller_name = vorname + " " + nachname # "Max Mustermann"
# Wiederholung
wiederholung = "Ha" * 3 # "HaHaHa"
# Länge eines Strings
länge = len(voller_name) # 14
# Zugriff auf einzelne Zeichen (Indizierung beginnt bei 0)
erster_buchstabe = voller_name[0] # "M"
letzter_buchstabe = voller_name[-1] # "n"
# Teilstring (Slice)
teil = voller_name[0:3] # "Max"
# String-Methoden
```

```
großbuchstaben = voller_name.upper() # "MAX MUSTERMANN"  
kleinbuchstaben = voller_name.lower() # "max mustermann"  
ersetzung = voller_name.replace("Max", "Moritz") # "Moritz Mustermann"
```

Ein besonders nützliches Feature in Python sind f-Strings, die es dir ermöglichen, Variablen direkt in Strings einzubetten:

```
name = "Anna"  
alter = 30  
begrüßung = f"Hallo {name}, du bist {alter} Jahre alt."  
# "Hallo Anna, du bist 30 Jahre alt."
```

Python und die Umlaute

Um den Einstieg zu erleichtern, haben wir im obigen Beispiel zugelassen, dass Variablen alle Zeichen des deutschen Zeichensatzes enthalten können, also auch Umlaute wie ä, ö, ü oder das »ß«. Nach einer allgemeinen Konvention in der Programmierung werde ich im Folgenden jedoch Umlaute vermeiden. Aus der Variablen `länge` wird dann `laenge`, aus `begrüßung` wird `begrueessung` usw. Du kannst den Chatbot deiner Wahl jederzeit anweisen, genauso zu verfahren, indem du z. B. folgenden Prompt hinzufügst:

»Verwende bei Codebeispielen niemals Umlaute wie ä, ö oder ü, aber auch ß nicht in Variablenbezeichnern. Beispiel: Verwende statt »größe« den Bezeichner »grosse« usw.«

Damit du diese Anweisung nicht stets von Neuem eingeben musst, kannst du sie im Bereich der persönlichen Konfiguration des Chatbots hinterlegen. Bei Claude findest du diese Option im PROFIL, bei ChatGPT im Bereich PERSONALISIERUNG • CHATGPT INDIVIDUELL KONFIGURIEREN.

3.1.4 Lineare Algorithmen und Benutzerinteraktion

Ein linearer Algorithmus ist eine Folge von Anweisungen, die nacheinander ausgeführt werden, ohne Verzweigungen oder Wiederholungen. Dies ist die einfachste Form eines Algorithmus und eignet sich gut für unsere ersten Programme.

Um interaktive Programme schreiben zu können, müssen wir Benutzereingaben entgegennehmen können – das EVA-Prinzip hast du bereits in Kapitel 2, »Die perfekte Basis – deine Entwicklungsumgebung«, kennengelernt. In Python verwendest du für Benutzereingaben die Funktion `input()`, die du bereits aus Abschnitt 3.1.1, »Vom Umgang mit Variablen«, kennst:

```
name = input("Wie heißt du? ")
print(f"Hallo, {name}! Schön, dich kennenzulernen.")
```

Die `input()`-Funktion zeigt den Text in der Klammer an und wartet dann auf eine Eingabe des Benutzers. Die Eingabe wird als String zurückgegeben, den du in einer Variablen speichern kannst.

Wichtig ist nun: Wenn du numerische Eingaben verarbeiten willst, musst du den eingegebenen String in den entsprechenden Zahlentyp umwandeln:

```
alter_text = input("Wie alt bist du? ")
alter = int(alter_text) # Umwandlung in Integer
groesse_text = input("Wie groß bist du (in Metern)? ")
groesse = float(groesse_text) # Umwandlung in Float
```

Alternativ kannst du die Umwandlung auch direkt in einer Zeile durchführen:

```
alter = int(input("Wie alt bist du? "))
groesse = float(input("Wie groß bist du (in Metern)? "))
```

Nun wollen wir unser Wissen anwenden und einen BMI-Rechner programmieren. Der Body Mass Index (BMI) ist ein Maß für das Körpergewicht im Verhältnis zur Körpergröße. Er wird berechnet, indem das Körpergewicht in Kilogramm durch das Quadrat der Körpergröße in Metern geteilt wird. Die Formel lautet somit:

$$\text{BMI} = \frac{\text{Gewicht (kg)}}{(\text{Größe (m)})^2}$$

Ein Prompt zur Berechnung des BMI könnte folgendermaßen aussehen:

Prompt: *Erstelle ein einfaches Python-Programm, das nach Eingabe von Körpergröße und Gewicht den BMI berechnet. Das Programm soll ein einfacher linearer Algorithmus ohne Unterrouninen und Verzweigungen sein.*

Der letzte Satz ist wichtig, um den erzeugten Code möglichst einsteigerfreundlich zu gestalten. Das Ergebnis ist leicht zu verstehen:

Ausgabe:

```
# Einfaches Python-Programm zur Berechnung des BMI
# Eingabe der Körpergröße in Metern
koerpergroesse =
```

```
float(input("Bitte Körpergröße in Metern eingeben (z. B. 1.75): "))
# Eingabe des Gewichts in Kilogramm
gewicht = float(input("Bitte Gewicht in Kilogramm eingeben (z. B. 70): "))
# BMI berechnen
bmi = gewicht / (koerpergroesse ** 2)
# BMI ausgeben
print("Ihr BMI ist:", round(bmi, 2))
```

Das Programm fragt zuerst nach der Körpergröße und dann nach dem Gewicht. Anschließend berechnet und gibt es den BMI aus, auf zwei Nachkommastellen gerundet.

Ohne die explizite Anweisung, einsteigerfreundlichen Code zu generieren, wäre das Programm deutlich komplexer geworden.

Das Struktogramm eines linearen Algorithmus

Ein *Struktogramm* (auch *Nassi-Shneiderman-Diagramm* genannt) ist die grafische Darstellung eines Algorithmus. Für einen linearen Algorithmus sieht es einfach aus wie eine Folge von Rechtecken, die von oben nach unten gelesen werden. Für das vorliegende Beispiel des BMI-Rechners kann auch ein Struktogramm durch eine einfache Abfrage erzeugt werden:

»Erstelle aus dem Code ein zugehöriges Struktogramm (Nassi-Shneiderman-Diagramm).«



Abbildung 3.1 Nassi-Shneiderman-Diagramm zum BMI-Rechner – erstellt von ChatGPT

Dieses einfache Struktogramm zeigt einen Algorithmus, der zwei Werte einliest, diese über eine Formel verknüpft und das Ergebnis ausgibt. Solche Diagramme helfen uns, den Ablauf eines Programms zu visualisieren, bevor wir es schreiben.

Hinweis: Eindeutigkeit des Ablaufdiagramms

Die Form des von ChatGPT erzeugten Ablaufdiagramms ist nicht eindeutig, teilweise wird das Diagramm auch durch Pfeile in der Antwort auf den Prompt realisiert. Die gegebene Antwort kann von Chat zu Chat variieren, es gibt bei Chatbots nicht »die eine, immer wiederkehrende Antwort«. Das sollte man sich immer bewusst machen!

Übung

An die Stelle des BMI ist mittlerweile das WHtR (Waist-to-Height-Ratio) getreten. Führe zunächst eine KI-gestützte Recherche durch, was sich dahinter verbirgt. Schreibe anschließend ein Programm, das das WHtR nach geeigneten Fragen an den Anwender berechnet.

3.1.5 Einfache Verzweigungen

Wir möchten der Nutzerin bzw. dem Nutzer nicht nur den BMI ausgeben, sondern auch eine Interpretation des Wertes. Eine vereinfachte Einstufung sieht folgendermaßen aus:

- ▶ Unter 18,5: Untergewicht
- ▶ 18,5 bis 24,9: Normalgewicht
- ▶ 25 bis 29,9: Übergewicht
- ▶ 30 und mehr: Adipositas (Fettleibigkeit)

Zur Umsetzung müssen erstmals Verzweigungen verwendet werden. Lassen wir uns einen BMI-Rechner mit der beschriebenen Einordnung in Klassen per Chatbot generieren. Ein abgewandelter Prompt könnte so aussehen:

Prompt: *Erstelle einen BMI-Rechner in Python. Das Programm soll den Benutzer nach seiner Größe in Metern und seinem Gewicht in Kilogramm fragen, den BMI berechnen und dann den Wert mit einer Interpretation ausgeben (z. B. Untergewicht, Normalgewicht etc.). Erstelle ein sehr einfaches Programm, das zwar Verzweigungen, aber keine Unterrouinen verwendet.*

ChatGPT liefert uns folgenden Code (der Codeanfang ist identisch zum oben umgesetzten linearen Algorithmus):

```
# Interpretation ausgeben
if bmi < 18.5:
    print("Untergewicht")
elif bmi < 25:
    print("Normalgewicht")
elif bmi < 30:
    print("Übergewicht")
else:
    print("Adipositas")
```

Benötigst du mehr Informationen zum Code, dann kannst du dir vom Chatbot erklären lassen, wie Verzweigungen in Python aufgebaut sind:

Prompt: *Erkläre anhand des vorliegenden Beispiels den Aufbau von Verzweigungen in Python.*

Ausgabe: Grundprinzip von Verzweigungen

Verzweigungen dienen dazu, Entscheidungen im Programm zu treffen, indem abhängig von bestimmten Bedingungen verschiedene Anweisungen ausgeführt werden.

[...]⁴

Auch zu dieser Aufgabe kannst du dir wieder ein Struktogramm erstellen lassen, das hilft, die Struktur des Programms zu verstehen (siehe Abbildung 3.2). Du musst dir immer vor Augen halten, dass die Ergebnisse, die du beim Zeichnen von technischen Diagrammen mithilfe von KI erhältst, noch weit davon entfernt sind, »perfekt« zu sein. Die logischen Grundlagen des behandelten Problems werden jedoch in der Regel bereits gut wiedergegeben.

Übung

Führe eine ähnliche Kategorisierung für den WHtR durch (siehe Übung in Abschnitt 3.1.4, »Lineare Algorithmen und Benutzerinteraktion«). Recherchiere dazu zunächst nach entsprechenden Kategorien, die für die Klassifizierung verwendet werden können.

4 Die vollständige Antwort des Chatbots findest du in den Begleitmaterialien zum Buch.

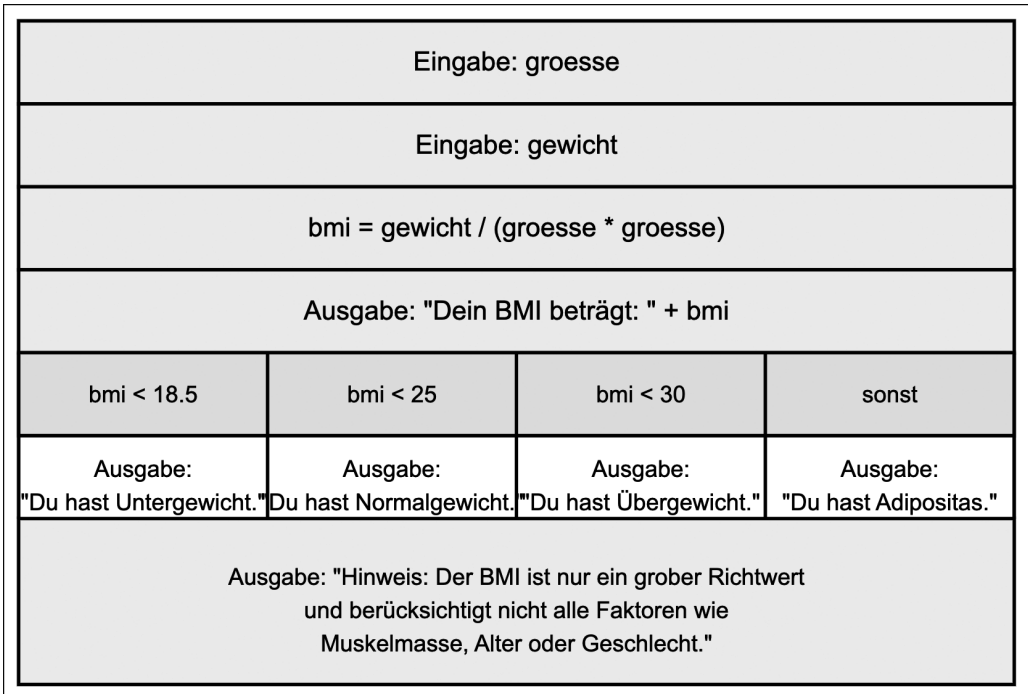


Abbildung 3.2 Das Struktogramm zum BMI-Rechner wurde mit Claude generiert. Es ist zwar nicht zu 100 % normgerecht, hilft aber, die Logik des Programms zu verstehen.

3.1.6 Einen einfachen Taschenrechner programmieren

Wir wollen nun einen einfachen Taschenrechner programmieren, der die vier Grundrechenarten beherrscht: Addition, Subtraktion, Multiplikation und Division. Unser Taschenrechner soll Folgendes tun:

- ▶ zwei Zahlen vom Benutzer einlesen
- ▶ nach der gewünschten Operation fragen
- ▶ das Ergebnis berechnen und ausgeben
- ▶ Fehler abfangen (z. B. eine Division durch null)

Ein möglicher Prompt für den Chatbot könnte wie folgt aussehen:

Prompt: *Programmiere einen einfachen Taschenrechner in Python, der zwei Zahlen und eine Operation (Addition = +, Subtraktion = -, Multiplikation = * und Division = /) vom Benutzer einliest und das Ergebnis ausgibt. Fange dabei auch Fehler wie Division durch null ab. Erstelle dafür sehr einfachen Code ohne Unterroutinen.*

Ein mögliches Ergebnis wäre:

```
try:
    zahl1 = float(input("Geben Sie die erste Zahl ein: "))
    operator = input("Geben Sie den Operator (+, -, *, /) ein: ")
    zahl2 = float(input("Geben Sie die zweite Zahl ein: "))
    if operator == '+':
        ergebnis = zahl1 + zahl2
    elif operator == '-':
        ergebnis = zahl1 - zahl2
    elif operator == '*':
        ergebnis = zahl1 * zahl2
    elif operator == '/':
        if zahl2 == 0:
            print("Fehler: Division durch null ist nicht erlaubt!")
            exit()
        else:
            ergebnis = zahl1 / zahl2
    else:
        print("Fehler: Ungültiger Operator!")
        exit()
    print("Ergebnis:", ergebnis)
except ValueError:
    print("Fehler: Bitte geben Sie gültige Zahlen ein!")
```

Listing 3.1 Ein einfacher Taschenrechner, umgesetzt in Python

Einiges wird dir aus den vorangegangenen Abschnitten bekannt vorkommen. Der Vorteil eines KI-Chatbots liegt aber gerade darin, dass man sich den generierten Code noch einmal genau erklären lassen kann:

Prompt: *Erkläre mir ausführlich den oben erzeugten Code in reiner Textform.*

Ausgabe: Der Code implementiert einen einfachen Taschenrechner in Python, der zwei Zahlen und einen Operator vom Benutzer einliest und dann das entsprechende Ergebnis berechnet. Zunächst wird ein try-Block verwendet, um Fehler abzufangen. Falls bei der Eingabe ungültige Werte (zum Beispiel Buchstaben statt Zahlen) eingegeben werden, fängt der except-Block den ValueError ab und gibt eine Fehlermeldung aus.

Im nächsten Schritt werden vom Benutzer drei Eingaben vorgenommen: die erste Zahl, der gewünschte Operator und die zweite Zahl. Dabei werden die Zahlen mit der

`float()`-Funktion in Gleitkommazahlen umgewandelt. Wird hierbei eine ungültige Eingabe gemacht, resultiert das in einem `ValueError`, der dann abgefangen wird.

[...]

Am Ende wird das berechnete Ergebnis mit der `print()`-Funktion ausgegeben. Der Code ist bewusst sehr einfach gehalten und kommt ohne zusätzliche Funktionen oder Unter-routinen aus, was ihn leicht verständlich und direkt macht.

Dieser Taschenrechner tut, was er soll, bietet aber noch Raum für Verbesserungen. Wir könnten etwa eine Schleife einbauen, die es dem Benutzer ermöglicht, mehrere Berechnungen durchzuführen, ohne das Programm neu starten zu müssen. Du wirst solche Konzepte in späteren Kapiteln kennenlernen.

3.1.7 Ein professioneller Taschenrechner mit echten Tasten

Vielleicht bist du ein wenig enttäuscht, dass der im letzten Abschnitt programmierte Taschenrechner in Textform daherkommt – mit der Hochglanzoptik der Taschenrechner-App auf deinem Smartphone kann er sicher nicht mithalten. Hier schlägt nun die große Stunde des Programmierens mit moderner KI-Technologie. Sofern du dir (wie im letzten Abschnitt geschehen) die Grundstruktur des Projekts in einfachen Schritten verdeutlicht bzw. erarbeitet hast, steht dir die Möglichkeit offen, das Projekt nach Belieben zu perfektionieren. Dies soll im Folgenden geschehen. Der verbesserte Rechner soll folgende Anforderungen erfüllen:

- ▶ Er soll in einer grafischen Oberfläche realisiert werden.
- ▶ Die Zahlen werden über Ziffernschaltflächen eingegeben.
- ▶ Als Rechenoperationen stehen `+`, `-`, `*` und `/` zur Verfügung.
- ▶ Das Ergebnis der Rechenoperation soll durch Anklicken der `=`-Schaltfläche ausgegeben werden.

Fragen wir zunächst den Chatbot, welche Grafikbibliotheken sich für die Darstellung eines Taschenrechners anbieten:

Prompt: *Ich möchte mit Visual Studio Code einen Taschenrechner in Python programmieren. Der Taschenrechner soll in grafischer Form realisiert werden, d. h. Tasten für Ziffern und Rechenoperationen besitzen. Welche Python-Bibliotheken bieten sich hierfür an? Erstelle noch kein Programm!*

Ausgabe: Für einen grafischen Taschenrechner in Python gibt es mehrere gute Bibliotheken, die sich anbieten. Hier sind die wichtigsten Optionen:

1. Tkinter: Eine in Python integrierte GUI-Bibliothek, die einfach zu erlernen und zu verwenden ist.
 - Vorteile: Bereits in Python enthalten, keine zusätzliche Installation nötig, einfacher Einstieg
 - Nachteile: Weniger moderne Optik im Vergleich zu anderen Bibliotheken
2. PyQt5/PySide6: Umfangreiche und mächtige GUI-Bibliotheken basierend auf dem Qt-Framework
 - Vorteile: Professionelles Aussehen, umfangreiche Funktionen, Cross-Plattform
 - Nachteile: Etwas komplexer zu erlernen, benötigt separate Installation
 - [...]

Für einen Anfänger oder für ein einfaches Projekt wie einen Taschenrechner empfehle ich **Tkinter**, da es bereits in Python enthalten ist und eine relativ einfache Syntax bietet. PyQt5 wäre eine gute Alternative, wenn du ein moderneres Aussehen bevorzugst und bereit bist, eine zusätzliche Bibliothek zu installieren.

Wenn schon, denn schon! Wir entscheiden uns für die modernere Variante über die Bibliothek PyQt5. Lass dir zunächst erklären, wie die benötigte Bibliothek installiert wird. Idealerweise gibst du an der Eingabeaufforderung dein Betriebssystem an:

Prompt: *Wie installiere ich die Bibliothek PyQt5 in Visual Studio Code auf einem Mac⁵?*

Ausgabe: Um PyQt5 in Visual Studio Code auf einem Mac zu installieren, folge diesen Schritten:

- ▶ Öffne das Terminal auf deinem Mac
- ▶ Führe folgenden Befehl aus, um PyQt5 über pip zu installieren⁶:

```
pip install pyqt5
```

Nachdem du nun die Grundvoraussetzungen für die Arbeit mit einer grafischen Umgebung (GUI) geschaffen hast, kannst du der KI schließlich den Auftrag geben, das Programm umzusetzen:

⁵ Ersetze an dieser Stelle »Mac« durch das Betriebssystem, das du verwendest (Windows, Linux, ...).

⁶ Für Projekte, die die Installation von Bibliotheken in einer lokalen VSC-Umgebung erfordern, wird die Arbeit mit einer virtuellen Python-Umgebung (venv) empfohlen, siehe Abschnitt 2.2.7, »Jupyter Notebook lokal installieren«.

Prompt: Ich möchte einen grafischen Taschenrechner in Python mithilfe der Bibliothek PyQt5 realisieren. Der Rechner soll wie ein richtiger Taschenrechner virtuelle Tasten für die Ziffern 0 bis 9 haben sowie Tasten für die Rechenoperationen +, -, * und /. Die Funktionsweise soll dann so sein, dass ich durch Anklicken der Ziffern die erste Zahl eingebe, dann die Rechenoperation, dann die zweite Zahl. Wenn ich nun auf die Schaltfläche = klicke, soll das Ergebnis in einem Anzeigebereich erscheinen. Außerdem soll es eine Schaltfläche C zum Löschen der Anzeige geben. Schreibe mir dazu ein passendes Programm!

Beachte, dass wir im Gegensatz zu den vorherigen Beispielen den Chatbot nicht explizit anweisen, möglichst einfachen Code zu generieren. Das Ergebnis wird für uns im Prinzip eine Blackbox sein. Wir testen lediglich, ob das Programm das macht, was wir wollen. Kopiere den vom Chatbot generierten Code in Visual Studio Code und teste ihn! Ein mögliches Ergebnis ist in Abbildung 3.3 zu sehen. Eine mögliche Umsetzung des Projekts findest du auch in den Begleitmaterialien zum Buch.



Abbildung 3.3 Da staunen selbst Profis: Ein Taschenrechner mit ansprechender grafischer Oberfläche kann auch von Laien binnen kürzester Zeit umgesetzt werden.

Auf einen Blick

1	Los geht's – dein smarterer Einstieg ins Programmieren	11
2	Die perfekte Basis – deine Entwicklungsumgebung	33
3	Einfach(e) Programme mit KI schreiben (lassen)	57
4	Programmieren mit System	121
5	Daten und ihre Visualisierung	189
6	Hardware und Sensoren erforschen	247
7	Künstliche Intelligenz selbst programmieren	313
8	Fortgeschrittene Techniken	345
9	Tipps und Tricks	415

Inhalt

1	Los geht's – dein smarterer Einstieg ins Programmieren	11
1.1	Warum KI das Programmieren revolutioniert	12
1.1.1	Von der Maschinensprache zur natürlichen Sprache	12
1.1.2	Der KI-Quantensprung	14
1.2	Dein KI-Code-Baukasten – die wichtigsten KI-Assistenten	17
1.3	Dein erstes Programm	20
1.3.1	Schritt-für-Schritt zu deinem ersten Programm	20
1.3.2	Ist eine KI für »Hallo Welt« überqualifiziert?	24
1.3.3	Grüße aus Babylon – »Hallo Welt« in anderen Programmiersprachen ...	26
1.3.4	Codeversteh'er – die KI erklärt dir den Code	27
1.4	Der »Beipackzettel« zum Buch	28
1.4.1	Das pädagogische Konzept	28
1.4.2	Der menschliche Vorteil: Kreativität	30
1.4.3	Die Inhalte – ein kurzer Überblick	31
2	Die perfekte Basis – deine Entwicklungsumgebung	33
2.1	Visual Studio Code	34
2.1.1	Installation unter Windows, macOS und Linux	34
2.1.2	VSC auf Deutsch umstellen und Python einrichten	36
2.1.3	Ein erstes Python-Projekt in VSC erstellen	38
2.1.4	Integration von KI-Tools in Visual Studio Code	40
2.2	Jupyter Notebook und Google Colab	44
2.2.1	Jupyter Notebook – der Standard für Python-Projekte	44
2.2.2	Google Colab – Jupyter Notebook in der Cloud	45
2.2.3	Anmeldung und Einrichtung von Google Colab	46
2.2.4	Dein erstes Colab Notebook	47
2.2.5	Installation und Verwaltung von Python-Bibliotheken	50

2.2.6	Gemini-KI in Google Colab	51
2.2.7	Jupyter Notebook lokal installieren	52
3	Einfach(e) Programme mit KI schreiben (lassen)	57
3.1	Taschenrechner & Co.	58
3.1.1	Vom Umgang mit Variablen	58
3.1.2	Was sind Variablentypen?	61
3.1.3	Wie verarbeitet man Zahlen und Texte?	62
3.1.4	Lineare Algorithmen und Benutzerinteraktion	64
3.1.5	Einfache Verzweigungen	67
3.1.6	Einen einfachen Taschenrechner programmieren	69
3.1.7	Ein professioneller Taschenrechner mit echten Tasten	71
3.2	Das Abenteuer der Verzweigung	74
3.2.1	Verzweigungen verstehen – der Währungsrechner	74
3.2.2	Ein Mini-Textabenteuer – der virtuelle Escape-Room	76
3.2.3	Im nächtlichen Dschungel der Mathematik	79
3.3	Jetzt geht's rund – Schleifen	81
3.3.1	Das Ziel: Wiederholungen automatisieren	82
3.3.2	Schleifentypen im Überblick	83
3.3.3	Der Passwort-Checker	84
3.4	Der KI-Coach – von der Idee zum Programm	87
3.4.1	Der Chatbot als Programmier-Tutor	87
3.4.2	Vom Konzept zum Code – einen Countdown-Timer entwickeln	88
3.4.3	Bibliotheken für spezielle Aufgaben entdecken	93
3.4.4	Von der Idee über das Struktogramm bis zum Programm	95
3.5	Die KI als Coding-Assistentin	100
3.5.1	Pair Programming – Definition und Einsatzbereiche	100
3.5.2	Explizit anfängertauglichen Code generieren	105
3.5.3	Pair Programming in verschiedenen Umgebungen	108
3.5.4	ChatGPT »Work with Apps«	114
3.5.5	Ein eigenes GPT oder Projekt für die didaktische Programmierung bauen	116

4	Programmieren mit System	121
4.1	Datenverarbeitung	121
4.1.1	Listen und Tabellen als Datenstruktur	122
4.1.2	Listen verarbeiten – die wichtigsten Operationen	125
4.1.3	Praktische Beispiele zum Einsatz von Listen	126
4.1.4	Komplexere Datenstrukturen – Stacks, Queues, Bäume und Graphen	129
4.1.5	Daten einlesen, verarbeiten und speichern	135
4.1.6	Der Vokabeltrainer	140
4.1.7	Arrays und Matrizen	141
4.2	Funktionen definieren	146
4.2.1	Funktionen als Strukturierungselement	146
4.2.2	Parameter und Rückgabewerte	147
4.2.3	Praxisbeispiel: Das Mehrspielerquiz	149
4.3	Bibliotheken verwenden	152
4.3.1	Bibliotheken importieren	152
4.3.2	Lösung linearer Gleichungssysteme mit NumPy	155
4.3.3	Web Scraping mit BeautifulSoup	157
4.4	Grafische Benutzeroberflächen	159
4.4.1	Was ist ein GUI?	159
4.4.2	GUI-Frameworks für Python	160
4.4.3	Fenster und Schaltflächen mit Tkinter erstellen	161
4.4.4	Ereignisse verarbeiten	163
4.4.5	Projekt: Grafikfähiger Taschenrechner mit Tkinter	165
4.4.6	Projekt: To-do-Liste mit PyQt5	167
4.5	Computergrafik	169
4.5.1	Erste Schritte mit Turtle-Grafik	169
4.5.2	Projekt: Das Haus vom Nikolaus	174
4.5.3	3D-Grafik	177
4.5.4	Projekt: Die Mandelbrotmenge (Apfelmännchen)	180
4.5.5	Spieledesign für Einsteiger*innen	182

5	Daten und ihre Visualisierung	189
5.1	KI und Excel	189
5.1.1	KI als Excel-Tutor	191
5.1.2	KI-Integration in Excel & Co.	194
5.1.3	Daten in das Excel-Format konvertieren	198
5.1.4	Excel-Daten per KI verarbeiten	200
5.1.5	Excel um VBA erweitern	205
5.2	Diagramme, Plots und Grafiken erstellen	209
5.2.1	Datensätze aus freien Quellen beschaffen	209
5.2.2	Datenbereinigung und -vorbereitung	212
5.2.3	Die KI als Ideengeber zur Datenanalyse nutzen	214
5.2.4	Visualisierungsbibliotheken im Überblick	218
5.2.5	Projekt: Das Solar-Dashboard	221
5.3	Webdaten nutzen	228
5.3.1	Am Anfang war die API	228
5.3.2	Wie wird das Wetter?	230
5.3.3	Der Börsenticker	234
5.3.4	Projekt: Ein Web Scraper für Medien	238
5.3.5	Einen KI-Chatbot per API nutzen	241
5.3.6	Projekt: Der KI-gestützte Webseitenkompressor	242
5.3.7	Projekt: KI-Bilder via API generieren	245
6	Hardware und Sensoren erforschen	247
6.1	Erste Schritte	247
6.1.1	Der Calliope – ein Mikrocontroller für alle	248
6.1.2	Den Calliope mini einrichten und programmieren	249
6.1.3	Die Alternative – der Funduino Cube	256
6.2	Ich fühle was, was du nicht fühlst – Sensoren	264
6.2.1	Sensoren – ein Überblick	265
6.2.2	Die Umgebungstemperatur messen	266
6.2.3	Die elektronische Wasserwaage	271
6.2.4	Der elektronische Kompass	276

6.3	... und Action! Aktoren nutzen	282
6.3.1	Erste Schritte: LEDs und Sound steuern	282
6.3.2	Motoren steuern	284
6.3.3	Ein Fahrzeug steuern	287
6.4	Komplexe Projekte realisieren	292
6.4.1	Dein persönlicher Hardwareassistent	292
6.4.2	Projekt: Die intelligente Wetterstation auf dem Cube	299
6.4.3	Projekt: Ein Navigationssystem für den Funduino Calle	304
6.4.4	It's Game Time – »Snake« auf dem Calliope	306
7	Künstliche Intelligenz selbst programmieren	313
7.1	Grundlagen des maschinellen Lernens	313
7.1.1	KI, maschinelles Lernen und neuronale Netze	313
7.1.2	Projekt: Deine erste Bildklassifikation	320
7.1.3	Die Grenzen maschinellen Lernens	329
7.2	Neuronale Netze programmieren	330
7.2.1	Grundlagen neuronaler Netze	330
7.2.2	TensorFlow kennenlernen	334
7.2.3	Projekt: Handschrifterkennung	336
8	Fortgeschrittene Techniken	345
8.1	Objektorientierte Programmierung	345
8.1.1	Warum objektorientierte Programmierung?	346
8.1.2	Grundkonzepte der objektorientierten Programmierung	347
8.1.3	Praxisbeispiel: Tierbestand im Zoo verwalten	350
8.1.4	Tipps zur objektorientierten Programmierung	357
8.2	Datenbanken	358
8.2.1	Das Prinzip einer relationalen Datenbank	358
8.2.2	SQLite – der einfache Einstieg	360
8.2.3	MySQL und Docker	367

8.3	App-Entwicklung für Smartphones	379
8.3.1	Apps für alle entwickeln	379
8.3.2	Von der Desktop-App zur mobilen Anwendung	381
8.3.3	Eine plattformübergreifende App mit Flutter bauen	384
8.4	Tools für Profis – Agenten und MCP	391
8.4.1	Agenten	392
8.4.2	Das Model Context Protocol (MCP)	401
8.5	Versionsverwaltung mit GitHub	405
8.5.1	Was ist GitHub und wozu braucht man es?	406
8.5.2	Dein erstes GitHub Repository einrichten	407
8.5.3	GitHub und Visual Studio Code – das perfekte Team	409
9	Tipps und Tricks	415
<hr/>		
9.1	Troubleshooting mit KI	415
9.1.1	Wenn nichts funktioniert – Probleme aus der Praxis	416
9.1.2	Systematisches Debugging mit KI	419
9.1.3	Codeoptimierung	422
9.2	Refaktorisierung und Dokumentation	426
9.2.1	Was ist Refactoring?	426
9.2.2	Code zwischen Programmiersprachen übersetzen	430
9.2.3	Dokumentation	431
9.3	Weiterführende Ressourcen	435
9.3.1	Prompt-Sammlungen nicht nur für Programmierer*innen	435
9.3.2	YouTube-Kanäle und Onlineressourcen	435
9.3.3	Bücher	436
9.3.4	Herausforderungen auf Coding-Plattformen	437
9.4	Fazit und Ausblick	438
	Glossar	441
	Index	449

Programmieren lernen mit KI

» Programmieren lernen völlig neu gedacht «

Michael Kofler, Fachgutachter

Von der Idee zum funktionierenden Code! Dieses Buch zeigt dir, wie du mithilfe von ChatGPT & Co. ganz einfach programmieren lernst, auch ohne Vorkenntnisse. Von leichten Projekten bis zu komplexen Anwendungen erlernst du Schritt für Schritt alle Grundlagen.



Dein Einstieg ins Programmieren mit KI

Du erhältst eine gründliche Einführung in die wichtigsten KI-Tools wie ChatGPT und lernst deine Entwicklungsumgebung kennen. Dabei entwickelst du deine ersten Programme und vertiefst dein Wissen mit praktischen Übungen.

Schritt für Schritt zur eigenen Software

Alle Themen werden an konkreten Projekten erklärt. Von einfachen Programmen über Datenverarbeitung bis zu grafischen Oberflächen – die KI hilft dir stets dabei.

Faszinierende Hardware-Projekte

Lerne die Hardware-Programmierung mit dem Calliope mini und dem Funduino Cube kennen. Von der intelligenten Wetterstation bis zum fahrenden Roboter – spannende Projekte warten auf dich.



Alle Beispielprompts und -ausgaben stehen zum Download bereit.



Rainer Hattenhauer ist promovierter Physiker und unterrichtet Mathematik, Physik und Informatik an einem Gymnasium. Der Technik-Nerd ist seit mehr als zwei Jahrzehnten Autor aus Leidenschaft. Das zeigt sich auch in seiner Begeisterung für KI-Themen.

Erste Schritte

- + KI-Tools kennenlernen
- + Entwicklungsumgebung einrichten
- + Erste Programme mit KI-Unterstützung

Projekte und Anwendungen

- + Taschenrechner und einfache Tools
- + Datenverarbeitung und -analyse
- + Grafische Benutzeroberflächen
- + Objektorientierte Programmierung
- + Datenbanken verwenden

So machst du weiter

- + Hardware-Projekte
- + App-Entwicklung
- + Einführung in Machine Learning

