# Einleitung

*Hello again, world!* In den späten 90er-Jahren habe ich als programmierender Teenager und Möchtegernhacker immer die neueste Ausgabe von *2600: The Hacker Quarterly* studiert. Eines Tages fand ich endlich den Mut, an dem von der Zeitschrift organisierten monatlichen Treffen in meiner Heimatstadt teilzunehmen, und war beeindruckt davon, wie viel diese Leute alle wussten! (Später erkannte ich jedoch, dass viele von ihnen über weit mehr Selbstvertrauen als echte Kenntnisse verfügten.) Das ganze Treffen über lauschte ich ehrfürchtig nickend dem, was die anderen zu sagen hatten, und versuchte, den Unterhaltungen zu folgen. Danach war ich entschlossen, jede Gelegenheit zu nutzen, um mehr über Computer, Programmierung und Netzwerksicherheit zu lernen, sodass ich mich beim nächsten Treffen an den Gesprächen beteiligen konnte.

Beim nächsten Treffen hörte ich jedoch weiterhin nur zu und fühlte mich im Vergleich zu allen anderen minderbemittelt. Also fasste ich erneut den Vorsatz, zu lernen und klug genug zu werden, um mitzuhalten. Ich vertiefte mein Wissen Monat für Monat, hatte aber immer das Gefühl, hinterherzuhinken. Schließlich er-

kannte ich, wie umfangreich das Gebiet der Informatik war, und befürchtete, niemals genug wissen zu können.

Ich wusste damals zwar schon mehr als meine Schulfreunde, doch meine Kenntnisse reichten mit Sicherheit nicht aus, um als Softwareentwickler arbeiten zu können. In den 90er-Jahren gab es Google, YouTube und Wikipedia noch nicht, aber ich hätte auch gar nicht gewusst, wie ich sie hätte nutzen sollen. Es war mir nie klar, womit ich mich als Nächstes befassen sollte. Stattdessen lernte ich, Hallo-Welt-Programme in verschiedenen Programmiersprachen zu schreiben. Dabei hatte ich jedoch nie das Gefühl, echte Fortschritte zu machen. Ich wusste nicht, wie ich jemals über die Grundlagen hinauskommen sollte.

Zur Softwareentwicklung gehört weit mehr als Schleifen und Funktionen. Wenn Sie einen Anfängerkurs absolviert oder ein Programmierbuch für Einsteiger gelesen haben und sich nach weiterführenden Informationen umsehen, landen Sie aber leider meistens nur bei weiteren Hallo-Welt-Tutorials. Programmierer nennen diese Phase die *Wüste der Verzweiflung*: Sie bewegen sich ziellos durch unterschiedliche Lernstoffe, ohne das Gefühl zu haben, Fortschritte zu machen. Für Material, das sich an Anfänger richtet, wissen Sie schon zu viel, aber es fehlt Ihnen an Erfahrung, um sich den komplizierteren Themen zu widmen.

Wenn Sie sich in dieser Wüste befinden, kommen Sie sich wie ein Hochstapler vor. Sie haben nicht das Gefühl, ein »richtiger« Programmierer zu sein oder Code so schreiben zu können, wie »richtige« Programmierer es tun. Dieses Buch habe ich für Personen geschrieben, denen es genau so geht. Wenn Sie die Grundlagen von Python kennen, hilft es Ihnen, ein besserer Softwareentwickler zu werden und dieses Gefühl der Verzweiflung zu überwinden.

## Wer dieses Buch lesen sollte und warum

Dieses Buch richtet sich an Leser, die bereits einen Grundkurs in Python absolviert haben und mehr lernen möchten. Bei diesem Grundkurs kann es sich um mein Buch *Routineaufgaben mit Python automatisieren* (dpunkt.verlag, 2020), das Buch *Python 3 Crashkurs* von Eric Matthes (dpunkt.verlag, 2020) oder auch eine Onlineschulung handeln.

Solche Einführungen können Ihre Begeisterung für die Programmierung wecken, allerdings gibt es danach immer noch viel zu lernen. Wenn Sie das Gefühl haben, noch nicht das Niveau eines professionellen Programmierers erreicht zu haben, und nicht wissen, wie Sie dorthin gelangen sollen, ist dies das richtige Buch für Sie.

Vielleicht haben Sie ja auch in einer anderen Sprache zu programmieren gelernt und möchten nun unmittelbar in das Python-Umfeld mit allen seinen Tools wech-

seln, ohne die typischen Hello-World-Grundlagen wiederzukäuen. In diesem Fall brauchen Sie nicht erst Hunderte von Seiten durchzuarbeiten, die die grundlegende Syntax erklären. Es reicht, wenn Sie sich den Artikel »Learn Python in Y Minutes« auf *https://learnxinyminutes.com/docs/python* ansehen oder Eric Matthes' Python-Spickzettel von »Python Crash Course – Cheat Sheet« auf *https://ehmatthes.github.io/pcc/cheatsheets/README.html* herunterladen, bevor Sie dieses Buch in Angriff nehmen.

## Über dieses Buch

In diesem Buch geht es nicht nur um Python-Syntax für Fortgeschrittene, sondern auch um die Verwendung der Befehlszeile und von Tools wie Codeformatierern, Lintern und Versionssteuerung, die auch professionelle Entwickler einsetzen.

Ich erkläre Ihnen, was Code gut lesbar macht und wie Sie sauberen Code schreiben können. Zur Veranschaulichung dieser Prinzipien stelle ich einige Programmierprojekte vor. Dies soll zwar kein Lehrbuch in Informatik werden, aber dennoch werden wir uns auch mit der O-Notation und objektorientierter Entwicklung beschäftigen.

Ein Buch allein reicht nicht aus, um jemanden zu einem professionellen Softwareentwickler zu machen. Ich habe dieses Buch geschrieben, um Ihre Kenntnisse zu vertiefen und Ihnen dadurch auf diesem Weg weiterzuhelfen. Dabei spreche ich auch einige Themen an, die Sie anderenfalls nur nach und nach durch Erfahrung lernen würden. Dieses Buch verschafft Ihnen eine solide Grundlage, sodass Sie gut für neue Herausforderungen gerüstet sind.

Ich rate Ihnen zwar dazu, das Buch von vorn bis hinten zu lesen, aber wenn ein Thema Sie besonders interessiert, können Sie auch gern zu dem betreffenden Kapitel vorblättern.

**Teil I: Erste Schritte**

- **Kapitel 1: Fehlermeldungen und Recherche** Zeigt Ihnen, wie Sie erfolgreich Fragen stellen und selbstständig Lösungen finden können. Außerdem erfahren Sie hier, wie Fehlermeldungen zu lesen sind und welche Verhaltensregeln Sie beachten müssen, wenn Sie online um Hilfe bitten.

- **Kapitel 2: Die Einrichtung der Umgebung und die Befehlszeile** Erklärt, wie Sie sich an der Befehlszeile zurechtfinden und wie Sie Ihre Entwicklungsumgebung sowie die Umgebungsvariable `PATH` einrichten.

**Teil II: Werkzeuge, Techniken und bewährte Vorgehensweisen**

- **Kapitel 3: Codeformatierung mit Black** Beschreibt die Stilrichtlinie PEP 8 und erklärt, wie Sie Ihren Code formatieren sollten, um ihn besser lesbar zu machen. Sie erfahren hier auch, wie Sie diesen Vorgang mit dem Codeformatierungswerkzeug Black automatisieren können.
- **Kapitel 4: Aussagekräftige Namen** Erklärt, wie Sie Variablen und Funktionen benennen sollten, um Ihren Code übersichtlicher zu gestalten.
- **Kapitel 5: Codegerüche** Beschreibt Warnzeichen, die auf mögliche Bugs in Ihrem Code hinweisen.
- **Kapitel 6: Pythonischer Code** Erklärt, was Code *pythonisch* macht, und zeigt Vorgehensweisen auf, um solchen idiomatischen Python-Code zu schreiben.
- **Kapitel 7: Programmierjargon** Erklärt Fachbegriffe, die in der Programmierung verwendet werden, sowie Begriffe, die oft verwechselt werden.
- **Kapitel 8: Häufige Fallstricke in Python** Beschreibt Aspekte von Python, die oft zu Missverständnissen oder zu Bugs führen, und zeigt Korrekturmaßnahmen sowie Möglichkeiten auf, solche Schwierigkeiten zu vermeiden.
- **Kapitel 9: Exotische Eigenarten von Python** Behandelt einige seltsame Eigenheiten von Python wie String-Interning oder das Schwerelosigkeits-Easter-Egg, auf die Sie sonst kaum stoßen würden. Vor allem aber erfahren Sie, warum sich einige Datentypen und Operatoren unerwartet verhalten, und erlangen dadurch ein tieferes Verständnis der Funktionsweise von Python.
- **Kapitel 10: Zweckmäßige Funktionen** Zeigt, wie Sie Funktionen strukturieren sollten, um sie zweckmäßig und übersichtlich zu gestalten. Sie lernen hier die Verwendung von * und ** in der Syntax für Argumente, die Vor- und Nachteile von umfangreichen und kleinen Funktionen sowie funktionale Programmiertechniken wie Lambda-Funktionen kennen.
- **Kapitel 11: Kommentare, Docstrings und Typhinweise** Erklärt, warum die nicht funktionalen Bestandteile von Programmen wichtig sind und wie sie dazu beitragen, dass sich der Code besser pflegen lässt. Sie erfahren hier, wie dicht Kommentare und Docstrings gesät sein sollten und wie Sie sie möglichst informativ gestalten können. Außerdem geht es in diesem Kapitel um Typhinweise und um die Verwendung von statischen Analysatoren wie Mypy zum Aufspüren von Bugs.

- **Kapitel 12: Versionssteuerung mit Git** Erklärt, wie Sie mit dem Versionssteuerungssystem Git den Verlauf von Änderungen am Quellcode aufzeichnen und zu früheren Versionen zurückkehren oder das erste Auftreten eines Bugs aufspüren können. Des Weiteren wird beschrieben, wie Cookiecutter Ihnen hilft, die Dateien für Ihre Codeprojekte zu strukturieren.

- **Kapitel 13: Leistungsmessung und Algorithmusanalyse** Erklärt, wie Sie die Geschwindigkeit Ihres Codes mithilfe der Module `timeit` und `cProfile` objektiv messen können. Darüber hinaus lernen Sie die Algorithmusanalyse mit der O-Notation kennen, mit der Sie vorhersagen können, wie sich die Codeausführung mit zunehmender Datenmenge verlangsamt.

- **Kapitel 14: Praxisprojekte** Hier wenden Sie die in Teil II erlernten Techniken in der Praxis an, indem Sie zwei Befehlszeilenspiele schreiben, nämlich *Turm von Hanoi*, bei dem es darum geht, Scheiben von einem Turm auf einen anderen umzustapeln, und *Vier gewinnt*.

**Teil III: Objektorientiertes Python**

- **Kapitel 15: Klassen** Erläutert die Bedeutung der objektorientierten Programmierung (OOP), da sie oft missverstanden wird. Viele Entwickler wenden OOP-Techniken im Übermaß an, da sie glauben, dies wäre die übliche Vorgehensweise, und machen ihren Code dadurch unnötig kompliziert. In diesem Kapitel erfahren Sie, wie Klassen geschrieben werden, aber vor allem, wann Sie es tun sollten und wann nicht.

- **Kapitel 16: Vererbung** Erklärt die Vererbung von Klassen und deren praktischen Nutzen für die Wiederverwendung von Code.

- **Kapitel 17: Pythonische OOP: Eigenschaften und Dunder-Methoden** Beschreibt Python-spezifische Merkmale für objektorientierte Programmierung wie Eigenschaften, Dunder-Methoden und Operatorüberladung.

## Ihr Weg zur Programmierung

Auf dem Weg vom Anfänger zum erfahrenen Programmierer fühlt man sich oft so, als ob man versucht, aus einem unter hohem Druck stehenden Feuerwehrschlauch zu trinken. Angesichts des großen Angebots an Lernstoff sorgen sich viele, ihre Zeit mit ungeeigneten Programmieranleitungen zu vergeuden.

Nachdem Sie dieses Buch gelesen haben (oder vielleicht sogar schon während der Lektüre), sollten Sie sich die folgenden weiteren einführenden Werke ansehen:

- *Python 3 Crashkurs* (dpunkt.verlag, 2020) von Eric Matthes richtet sich zwar an Anfänger, vermittelt dank seines projektgestützten Lehransatzes aber auch erfahrenen Programmierern eine Vorstellung der Python-Bibliotheken Pygame, Matplotlib und Django.
- *Impractical Python Projects* (No Starch Press, 2018) von Lee Vaughan erweitert Ihre Python-Fertigkeiten anhand von Projekten. Die Programme, die Sie aufgrund der Anleitungen in diesem Buch erstellen, machen Spaß und stellen eine großartige Übung dar.
- *Serious Python* (No Starch Press, 2018) von Julien Danjos erklärt, was Sie tun müssen, um sich von einem Hobbyprogrammierer zu einem erfahrenen Softwareentwickler zu mausern, die empfohlenen Vorgehensweisen der Branche zu befolgen und skalierbaren Code zu schreiben.

Die technischen Aspekte sind aber nur eine Stärke von Python. Rund um diese Programmiersprache ist auch eine bunte Community gewachsen. Sie ist für die gut zugängliche Dokumentation und den Support verantwortlich, die unter Programmiersprachen ihres Gleichen suchen. Es gibt jährliche PyCon-Konferenzen sowie viele regionale Zusammenkünfte mit vielfältigen Vorträgen, die sich an Programmierer mit unterschiedlicher Erfahrung richten. Diese Vorträge können Sie sich auch kostenlos online auf *https://pyvideo.org/* ansehen. Um Vorträge zu Ihren Interessengebieten zu finden, schlagen Sie auf der Seite *Tags* nach.

Um sich intensiver mit den anspruchsvolleren Aspekten der Syntax von Python und der Standardbibliothek zu beschäftigen, empfehle ich Ihnen die Lektüre der folgenden Titel:

- *Effektiv Python programmieren* (mitp, 2020) von Brett Slatkin bietet eine beeindruckende Zusammenstellung von empfohlenen *pythonischen* Vorgehensweisen und Sprachmerkmalen.
- *Python Cookbook* (O'Reilly Media, 2013) von David Beazley und Brian K. Jones wartet mit einer umfangreichen Menge von Codebausteinen auf, mit denen Python-Neulinge ihr Repertoire erweitern können.
- *Fluent Python* (O'Reilly Media, 2021) von Luciano Ramalho ist ein Meisterwerk, das die Feinheiten von Python erklärt. Sein Umfang von fast 800 Seiten mag abschreckend wirken, aber die Lektüre ist mit Sicherheit der Mühe wert.

Viel Glück auf Ihrem Weg zur besseren Programmierung! Fangen wir an!