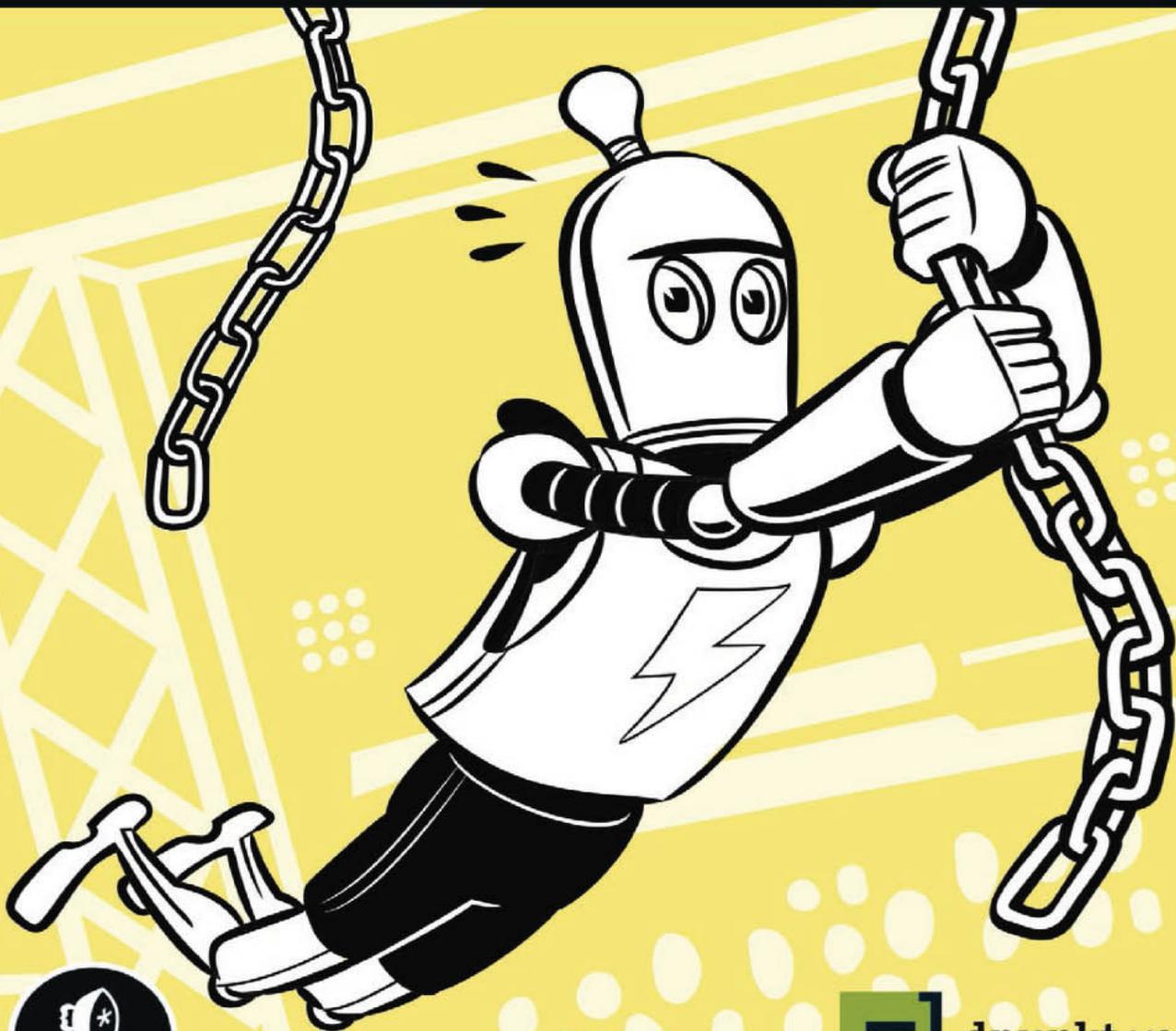


Der Weg zum Python-Profi

Ein Best-Practice-Buch für sauberes Programmieren

Al Sweigart



dpunkt.verlag

Papier
plus⁺
PDF.

Zu diesem Buch – sowie zu vielen weiteren dpunkt.büchern – können Sie auch das entsprechende E-Book im PDF-Format herunterladen. Werden Sie dazu einfach Mitglied bei dpunkt.plus⁺:

www.dpunkt.plus

Al Sweigart

Der Weg zum Python-Profi

**Ein Best-Practice-Buch für sauberes
Programmieren**



dpunkt.verlag

Al Sweigart

Lektorat: Gabriel Neumann

Lektoratsassistentz: Anja Weimer

Übersetzung: Volkmar Gronau, G&U Language & Publishing Services GmbH,
Flensburg, www.GundU.com

Copy-Editing: Claudia Lötschert, www.richtiger-text.de

Satz: Ulrich Borstelmann, www.borstelmann.de

Herstellung: Stefanie Weidner

Umschlaggestaltung: Helmut Kraus, www.exclam.de, nach der Originalvorlage
von No Starch Press

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen
Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über
<http://dnb.d-nb.de> abrufbar.

ISBN:

Print 978-3-86490-874-3

PDF 978-3-96910-677-8

ePub 978-3-96910-678-5

mobi 978-3-96910-679-2

1. Auflage 2022

Translation Copyright für die deutschsprachige Ausgabe © 2022 dpunkt.verlag
GmbH

Wieblinger Weg 17

69123 Heidelberg

Copyright © 2021 by Al Sweigart. Title of English-language original: *Beyond the
Basic Stuff with Python: Best Practices for Writing Clean Code*, ISBN 978-1-
59327-966-0,

published by No Starch Press Inc. 245 8th Street, San Francisco, California
United States 94103.

The German-language edition Copyright © 2022 by dpunkt.verlag under license
by No Starch Press Inc.

All rights reserved.

Hinweis:

Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger
Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die
Einschweißfolie.



Schreiben Sie uns:

Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: hallo@dpunkt.de.

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag noch Übersetzer können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

Inhalt

Der Autor

Der Fachgutachter

Danksagung

Einleitung

Wer dieses Buch lesen sollte und warum

Über dieses Buch

Ihr Weg zur Programmierung

1 Fehlermeldungen und Recherche

Python-Fehlermeldungen verstehen

- Tracebacks untersuchen

- Fehlermeldungen recherchieren

Fehler vermeiden mit Lintern

Um Hilfe bitten

- Geben Sie gleich ausreichend Informationen, um Rückfragen zu vermeiden

- Formulieren Sie Ihre Fragen als Fragen

Stellen Sie Ihre Fragen auf einer geeigneten Website

Geben Sie das Problem in der Überschrift an

Erklären Sie, was der Code tun soll

Geben Sie die vollständige Fehlermeldung an

Teilen Sie Ihren Code vollständig mit

Gestalten Sie Ihren Code durch saubere Formatierung lesbar

Beschreiben Sie, was Sie bereits versucht haben

Beschreiben Sie Ihre Ausstattung

Ein Beispiel für eine gut gestellte Frage

Zusammenfassung

2 Die Einrichtung der Umgebung und die Befehlszeile

Das Dateisystem

Pfade in Python

Das Benutzerverzeichnis

Das aktuelle Arbeitsverzeichnis

Absolute und relative Pfade

Programme und Prozesse

Die Befehlszeile

Ein Terminalfenster öffnen

Programme an der Befehlszeile ausführen

Befehlszeilenargumente

Python-Code mit `-c` an der Befehlszeile ausführen

Python-Programme an der Befehlszeile ausführen

Py.exe

Befehle aus einem Python-Programm heraus ausführen

Tipparbeit durch Tabulatorvervollständigung sparen

Der Befehlsverlauf

Gebräuchliche Befehle

PATH und andere Umgebungsvariablen

Umgebungsvariablen anzeigen

Die Umgebungsvariable PATH

Die Umgebungsvariable PATH in der Befehlszeile ändern

Ordner in Windows dauerhaft zu PATH hinzufügen

Ordner in macOS und Linux dauerhaft zu PATH hinzufügen

Python-Programme außerhalb der Befehlszeile ausführen

Python-Programme in Windows ausführen

Python-Programme in macOS ausführen

Python-Programme in Ubuntu Linux ausführen

Zusammenfassung

3 Codeformatierung mit Black

Wie man Freunde und Kollegen vergrault

PEP 8 und andere Stilrichtlinien

Horizontale Abstände

Leerzeichen zur Einrückung verwenden

Abstände innerhalb einer Zeile

Vertikale Abstände

Beispiel für vertikale Abstände

Empfohlene Vorgehensweisen für vertikale Abstände

Black: Der kompromisslose Codeformatierer

Black installieren

Black an der Befehlszeile ausführen

Black für einzelne Abschnitte Ihres Codes ausschalten

Zusammenfassung

4 Aussagekräftige Namen

Groß- und Kleinschreibung

Namenskonventionen in PEP 8

Namen geeigneter Länge

- Zu kurze Namen

- Zu lange Namen

Leicht zu findende Namen

Scherze, Wortspiele und Anspielungen vermeiden

Integrierte Namen nicht überschreiben

Die allerschlechtesten Variablennamen

Zusammenfassung

5 Codegerüche

Duplizierter Code

Magische Zahlen

Auskommentierter und toter Code

Print-Debugging

Variablen mit numerischen Suffixen

Klassen statt Funktionen oder Module

Verschachtelte Listennotation

Leere except-Blöcke und nichtssagende Fehlermeldungen

Legenden über Code Smells

- Legende: Funktionen sollten nur eine return-Anweisung am Ende aufweisen

- Legende: Funktionen mit einer try-Anweisung dürfen keine anderen Anweisungen enthalten

- Legende: Flag-Argumente sind schlecht

- Legende: Globale Variablen sind schlecht

- Legende: Kommentare sind unnötig

Zusammenfassung

6 Pythonischer Code

Python-Zen

Aussagekräftige Einrückungen

Leistung mit dem Modul `timeit` messen

Häufig falsch angewendete Syntax

- Verwenden Sie `enumerate()` statt `range()`

- Verwenden Sie `with` statt `open()` und `close()`

- Verwenden Sie `is` statt `==` zum Vergleich mit `None`

Stringformatierung

- Verwenden Sie Rohstrings bei einer großen Anzahl von Backslashes

- F-Strings

Flache Kopien von Listen

Pythonischer Umgang mit Dictionaries

- Verwenden Sie `get()` und `setdefault()` für Dictionaries

- Verwenden Sie `collections.defaultdict` für Standardwerte

- Verwenden Sie Dictionaries statt `switch`-Konstruktionen

Bedingte Ausdrücke: Pythons »hässlicher« ternärer Operator

Variablenwerte

- Zuweisungs- und Vergleichsoperatoren verketteten

- Eine Variable auf Gleichheit mit mehreren Werten prüfen

Zusammenfassung

7 Programmierjargon

Definitionen

Die Sprache Python und der Interpreter Python

Garbage Collection

Literale

Schlüsselwörter

Objekte, Werte und Identitäten

Elemente

Veränderbare und unveränderbare Objekte

Indizes, Schlüssel und Hashes

Container-, Folgen-, Maps- und Set-Datentypen

Dunder- oder magische Methoden

Module und Pakete

Aufrufbare Objekte und Objekte erster Klasse

Häufig verwechelte Begriffe

Anweisungen und Ausdrücke

Blöcke, Klauseln und Rümpfe

Variablen und Attribute

Funktionen und Methoden

Iterierbare Objekte und Iteratoren

Syntax-, Laufzeit- und semantische Fehler

Parameter und Argumente

Implizite und explizite Typumwandlung

Eigenschaften und Attribute

Bytecode und Maschinencode

Skripte und Programme, Skriptsprachen und Programmiersprachen

Bibliotheken, Frameworks, SDKs, Engines und APIs

Zusammenfassung

Literatur

8 Häufige Fallstricke in Python

Elemente zu Listen hinzufügen und entfernen

Veränderbare Werte kopieren

Standardargumente

Strings zusammenbauen

Sortierung mit sort()

Genauigkeit von Fließkommazahlen

Verkettung von Ungleichheitsoperatoren

Das Komma in einelementigen Tupeln

Zusammenfassung

9 Exotische Eigenarten von Python

256 ist 256, aber 257 ist nicht 257

String-Interning

Die Bedeutung von ++ und -- in Python

Alles von nichts

Boolesche Werte als Integer

Verkettung unterschiedlicher Operatoren

Schwerelosigkeit in Python

Zusammenfassung

10 Zweckmäßige Funktionen

Funktionsnamen

Der Umfang von Funktionen

Funktionsparameter und -argumente

Standardargumente

Argumente mit * und ** an Funktionen übergeben

Variadische Funktionen mit * erstellen

Variadische Funktionen mit ** erstellen

- Wrapper-Funktionen mit * und ** erstellen
- Funktionale Programmierung
 - Nebenwirkungen
 - Funktionen höherer Ordnung
 - Lambda-Funktionen
 - Zuordnung und Filterung mit Listennotation
- Der Datentyp von Rückgabewert
- Ausnahmen auslösen oder Fehlercodes zurückgeben
- Zusammenfassung

11 Kommentare, Docstrings und Typhinweise

Kommentare

- Formatierung von Kommentaren
- Inline-Kommentare
- Erklärende Kommentare
- Kommentare zur Gliederung
- »Lessons-Learned-Kommentare«
- Rechtliche Hinweise
- Professionelle Formulierung
- Codetags und TODO-Kommentare
- Magische Kommentare und Quelldateicodierung

Docstrings

Typhinweise

- Tools zur statischen Analyse
- Typhinweise für mehrere Typen
- Typhinweise für Listen, Dictionarys u. Ä
- Rückportierung von Typhinweisen mithilfe von Kommentaren

Zusammenfassung

12 Versionssteuerung mit Git

Commits und Repositorys in Git

Neue Python-Projekte mit Cookiecutter erstellen

Git installieren

- Git-Benutzername und E-Mail-Adresse angeben

- GUI-Werkzeuge für Git installieren

Der Arbeitsablauf in Git

- Der Dateistatus in Git

- Wozu gibt es bereitgestellte Dateien?

Ein Git-Repository erstellen

- Zu verfolgende Dateien hinzufügen

- Einzelne Dateien ignorieren

- Änderungen mit Commit bestätigen

- Änderungen mit `git diff` vor dem Commit einsehen

- Änderungen mit `git difftool` in einer GUI-Anwendung einsehen

- Häufigkeit von Commits

- Dateien löschen

- Dateien umbenennen und verschieben

Das Commitprotokoll einsehen

Frühere Versionen wiederherstellen

- Unbestätigte lokale Änderungen rückgängig machen

- Bereitstellung einer Datei aufheben

- Die letzten Commits zurücknehmen

- Zurücksetzen einer einzelnen Datei zu einem bestimmten Commit

- Den Commitverlauf ändern

GitHub und `git push`

- Ein bestehendes Repository auf GitHub übertragen

Ein GitHub-Repository klonen

Zusammenfassung

13 Leistungsmessung und Algorithmusanalyse

Das Modul timeit

Der Profiler cProfile

Komplexitätsanalyse

Ordnungen

Ein Bücherregal als Metapher für Ordnungen

Worst Case und Best Case

Die Ordnung Ihres Codes bestimmen

Warum Terme niedriger Ordnungen und Koeffizienten keine Rolle spielen

Beispiele für die Komplexitätsanalyse

Die Ordnung gängiger Funktionsaufrufe

Komplexitätsanalyse im Überblick

Die Ordnung spielt bei kleinem n keine Rolle - und n ist gewöhnlich klein

Zusammenfassung

14 Praxisprojekte

Turm von Hanoi

Die Ausgabe

Der Quellcode

Den Code schreiben

Vier gewinnt

Die Ausgabe

Der Quellcode

Den Code schreiben

Zusammenfassung

15 Klassen

Formulare als Veranschaulichung

Objekte aus Klassen erstellen

Eine einfache Klasse erstellen: WizCoin

- Methoden, `__init__()` und der Parameter `self`

- Attribute

- Private Attribute und private Methoden

Die Funktion `type()` und das Attribut `__qualname__`

OOP- und Nicht-OOP-Code im Vergleich

Klassen für reale Objekte

Zusammenfassung

16 Vererbung

Wie Vererbung funktioniert

- Methoden überschreiben

- Die Funktion `super()`

- Komposition statt Vererbung

- Nachteile der Vererbung

Die Funktionen `isinstance()` und `issubclass()`

Klassenmethoden

Klassenattribute

Statische Methoden

Wann brauchen Sie Klassenmerkmale und statische Methoden?

Schlagwörter der objektorientierten Programmierung

- Kapselung

- Polymorphismus

Wann Sie die Vererbung nicht nutzen sollten

Mehrfachvererbung

Die Reihenfolge der Methodenauflösung

Zusammenfassung

17 Pythonische OOP: Eigenschaften und Dunder-Methoden

Eigenschaften

Attribute in Eigenschaften umwandeln

Set-Methoden zur Datenvalidierung

Schreibgeschützte Eigenschaften

Wann Sie Eigenschaften verwenden sollten

Dunder-Methoden

Dunder-Methoden zur Stringdarstellung

Numerische Dunder-Methoden

Reflektierte numerische Dunder-Methoden

Direkte Dunder-Methoden für erweiterte

Zuweisungsoperatoren

Dunder-Methoden für Vergleiche

Zusammenfassung

Stichwortverzeichnis

Für meinen Neffen Jack

Der Autor

Al Sweigart ist Softwareentwickler und Fachbuchautor und lebt in Seattle. Python ist seine bevorzugte Programmiersprache, für die er bereits mehrere Open-Source-Module entwickelt hat. Er hat mehrere Programmierbücher für Einsteiger geschrieben, darunter *Routineaufgaben mit Python automatisieren* und *Cooler Spiele mit Scratch 3*, die ebenfalls beim dpunkt.verlag erschienen sind. Die englische Originalversion seiner Bücher steht unter einer Creative-Commons-Lizenz kostenlos auf seiner Website <https://www.inventwithpython.com/> zur Verfügung. Seine Katze Zophie wiegt 11 Pfund.

Der Fachgutachter

Kenneth Love ist Programmierer und Lehrer und richtet Konferenzen aus. Er ist Mitarbeiter von Django und ein PSF Fellow. Zurzeit arbeitet er als technischer Leiter und Softwareingenieur für O'Reilly Media.

Danksagung



Es ist irreführend, dass nur mein Name auf dem Titelbild steht, denn ohne die Bemühungen vieler anderer Personen hätte es dieses Buch nie gegeben. Ich möchte meinem Herausgeber Bill Pollock, meinen Lektoren Frances Saux, Annie Choi, Meg Sneeringer und Jan Cash, Herstellungsleiterin Maureen Forys, Korrektorin Anne Marie Walker und Chefredakteurin Barbara Yien danken. Ein weiteres Dankeschön geht an Josh Ellingson für ein weiteres hervorragendes Titelbild, an meinen Fachgutachter Kenneth Love und an all die großartigen

Freunde, die ich in der Python-Community kennengelernt habe.

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 überlauschte 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 erkannte 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 wechseln, ohne

die typischen Hello-World-Grundlagen wiederzukäuen. In diesem Fall brauchen Sie nicht erst Hunderte von Seiten durcharbeiten, 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 Programmier Techniken 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 gesetzt 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 Feuerwehrschauch 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