

Michael Weigend

# PYTHON 3

## FÜR STUDIUM UND AUSBILDUNG

Einfach lernen  
und professionell anwenden



Michael Weigend

# PYTHON 3

FÜR STUDIUM UND AUSBILDUNG

Einfach lernen  
und professionell anwenden





## **Hinweis des Verlages zum Urheberrecht und Digitalen Rechtemanagement (DRM)**

Liebe Leserinnen und Leser,

dieses E-Book, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Mit dem Kauf räumen wir Ihnen das Recht ein, die Inhalte im Rahmen des geltenden Urheberrechts zu nutzen. Jede Verwertung außerhalb dieser Grenzen ist ohne unsere Zustimmung unzulässig und strafbar. Das gilt besonders für Vervielfältigungen, Übersetzungen sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Je nachdem wo Sie Ihr E-Book gekauft haben, kann dieser Shop das E-Book vor Missbrauch durch ein digitales Rechtemanagement schützen. Häufig erfolgt dies in Form eines nicht sichtbaren digitalen Wasserzeichens, das dann individuell pro Nutzer signiert ist. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

Beim Kauf des E-Books in unserem Verlagsshop ist Ihr E-Book DRM-frei.

Viele Grüße und viel Spaß beim Lesen,

*Ihr mitp-Verlagsteam*





**Michael Weigend**

# **Python 3 für Studium und Ausbildung**

Einfach lernen und professionell anwenden



# Impressum

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über

<http://dnb.dnb.de> abrufbar.

ISBN 978-3-7475-0436-9

1. Auflage 2022

[www.mitp.de](http://www.mitp.de)

E-Mail: [mitp-verlag@sigloch.de](mailto:mitp-verlag@sigloch.de)

Telefon: +49 7953 / 7189 - 079

Telefax: +49 7953 / 7189 - 082

© 2022 mitp Verlags GmbH & Co. KG

Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

*»Python« and the Python logos are trademarks or registered trademarks of the Python Software Foundation, used by mitp Verlags GmbH & Co. KG with permission from the Foundation.*

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Lektorat: Janina Bahlmann

Sprachkorrektur: Christine Hofmeister

Covergestaltung: Christian Kalkert

Bildnachweis: © Gstudio / stock.adobe.com, © ylivdesign / stock.adobe.com

Kastenicons: Tanja Wehr, sketchnotelovers

Abbildungen & Grafiken: Michael Weigend

Satz: Petra Kleinwegen

Dieses Ebook verwendet das ePub-Format und ist optimiert für die Nutzung mit dem iBooks-reader auf dem iPad von Apple. Bei der Verwendung anderer Reader kann es zu Darstellungsproblemen kommen.

Hinweis des Verlages zum Urheberrecht und Digitalen Rechtemanagement (DRM)

Der Verlag räumt Ihnen mit dem Kauf des ebooks das Recht ein, die Inhalte im Rahmen des geltenden Urheberrechts zu nutzen. Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen,

Übersetzungen, Mikroverfilmungen und Einspeicherung und Verarbeitung in elektronischen Systemen.

Der Verlag schützt seine ebooks vor Missbrauch des Urheberrechts durch ein digitales Rechtemanagement. Bei Kauf im Webshop des Verlages werden die ebooks mit einem nicht sichtbaren digitalen Wasserzeichen individuell pro Nutzer signiert.

Bei Kauf in anderen ebook-Webshops erfolgt die Signatur durch die Shopbetreiber. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

*Für Lars Jonah*

# Inhalt

## Impressum

## Einleitung

Python in Studium und Ausbildung

Der Aufbau des Buchs

Achten Sie auf den Schrifttyp!

Programmtexte und Lösungen zum Download

## 1 Willkommen zu Python!

1.1 Die Programmiersprache Python

1.2 Was ist ein Algorithmus?

1.3 Syntax und Semantik

1.4 Interpreter und Compiler

1.5 Python installieren

1.6 Python im interaktiven Modus

1.7 Die Entwicklungsumgebung IDLE

1.8 Hotkeys für die IDLE-Shell

1.9 Anweisungen

1.9.1 Ausdruck

1.9.2 Funktionsaufruf

1.9.3 Zuweisung

1.9.4 Erweiterte Zuweisungen

1.10 Zahlen verarbeiten - Python als Taschenrechner

- 1.10.1 Operatoren
- 1.10.2 Variablen verwenden
- 1.11 Eine weitere Entwicklungsumgebung: Thonny
- 1.12 Notebooks mit Jupyter und CoLab
- 1.13 Rückblick
- 1.14 Übungen
- 1.15 Lösung der Frage: Semantik im Alltag

## **2 Datentypen - die Python-Typ-Hierarchie**

- 2.1 Literale und die Funktion type()
- 2.2 Die Python-Typ-Hierarchie
- 2.3 Standard-Typen
  - 2.3.1 Ganze Zahl (int)
  - 2.3.2 Gleitkommazahl (float)
  - 2.3.3 Komplexe Zahlen (complex)
  - 2.3.4 Zeichenkette (str)
  - 2.3.5 Tupel (tuple)
  - 2.3.6 Liste (list)
  - 2.3.7 Menge (set)
  - 2.3.8 Dictionary (dict)
  - 2.3.9 Wahrheitswerte - der Datentyp bool
  - 2.3.10 NoneType
- 2.4 Gemeinsame Operationen für Kollektionen
  - 2.4.1 Kollektion
  - 2.4.2 Sequenz
- 2.5 Objekte eines Typs erzeugen - Casting

2.6 Dynamische Typisierung

2.7 Rückblick

2.8 Übung: Anweisungen

### **3 Interaktive Programme**

3.1 Das erste Python-Skript

3.2 Das EVA-Prinzip

3.3 Kommentare

3.4 Projekt: Volumenberechnung

3.4.1 Kürzerer Programmtext durch verschachtelte Funktionsaufrufe

3.4.2 Aufruf der Funktion print() mit mehreren Argumenten

3.5 Python-Programme starten

3.5.1 Ausführung auf der Kommandozeile

3.5.2 Start durch Anklicken des Programm-Icons unter Windows

3.5.3 Python-Programme unter Linux – die Shebang-Zeile

3.5.4 Starten im Finder von macOS

3.6 Fehler finden

3.6.1 Syntaxfehler

3.6.2 Laufzeitfehler

3.6.3 Semantische Fehler

3.6.4 Tipps zum Fehlerfinden

3.7 Rückblick

3.8 Übungen

## 3.9 Lösungen zu den Fragen

# 4 Kontrollstrukturen

## 4.1 Programmverzweigungen

4.1.1 Einseitige Verzweigung (if)

4.1.2 Projekt: Passwort

4.1.3 Zweiseitige Verzweigung (if...else)

4.1.4 Projekt: Kinokarte

4.1.5 Fallunterscheidung (if...elif...else)

4.1.6 Projekt: Auskunftautomat

## 4.2 Das Layout von Python-Programmen: Zeilen und Blöcke

4.2.1 Block

4.2.2 Zeilenstruktur

## 4.3 Bedingungen konstruieren

4.3.1 Boolesche Werte

4.3.2 Boolesche Operatoren

4.3.3 Vergleichsketten

4.3.4 Projekt: Früchte erkennen

## 4.4 Bedingte Wiederholung - while

4.4.1 Projekt: Aufsummieren

4.4.2 Projekt: Planeten

4.4.3 Projekt: Wurzelberechnung (Mathematik)

4.4.4 Endloswiederholung

## 4.5 Iterationen - for

4.5.1 Wiederholungen mit range()

- 4.6 Rückblick
- 4.7 Übungen
- 4.8 Lösungen zu den Fragen

## **5 Funktionen**

- 5.1 Warum definiert man Funktionen?
- 5.2 Definition und Aufruf einer Funktion
  - 5.2.1 Projekt: Fallhöhe berechnen
- 5.3 Optionale Parameter und voreingestellte Werte
  - 5.3.1 Erweiterung des Projekts: Fallhöhe auf unterschiedlichen Himmelskörpern
- 5.4 Eine Funktion in der Shell testen
- 5.5 Die return-Anweisung
  - 5.5.1 Prozeduren
  - 5.5.2 Wirkungen der return-Anweisung
- 5.6 Positionsargumente und Schlüsselwortargumente
- 5.7 Guter Programmierstil
  - 5.7.1 Funktionsname
  - 5.7.2 Funktionsannotationen
  - 5.7.3 Docstring
  - 5.7.4 Signatur
- 5.8 Die print()-Funktion unter der Lupe
- 5.9 Globale und lokale Namen
- 5.10 Rekursive Funktionen
  - 5.10.1 Projekt: Die Berechnung der Fakultät
- 5.11 Lambda-Funktionen \*

5.12 Funktionen als Argumente: map() und filter() \*

5.12.1 Mapping

5.12.2 Filtern

5.13 Rückblick

5.14 Übungen

5.15 Lösungen zu den Fragen

## **6 Mit Modulen arbeiten**

6.1 Importanweisungen

6.1.1 Ein Modul importieren

6.1.2 Funktionen aus einem Modul importieren

6.2 Mathematische Funktionen: Das Modul math

6.3 Zufallsfunktionen: Das Modul random

6.3.1 Projekt: Würfeln

6.3.2 Projekt: Wer ist der Nächste?

6.4 Datum und Zeit

6.4.1 Projekt: Uhrzeit

6.4.2 Projekt: Rechentrainer

6.5 Ein eigenes Modul erstellen

6.5.1 Projekt: Ein Modul zur Volumenberechnung

6.5.2 Welchen Vorteil haben Module?

6.6 Module aus dem Python Package Index (PyPI)

6.7 Rückblick

6.8 Übungen

## **7 Mit Kollektionen modellieren**

7.1 Sequenzen

- 7.1.1 Listen
- 7.1.2 Tupel
- 7.1.3 Komplexe Sequenzen
- 7.1.4 Iteration über eine Liste aus Tupeln
- 7.1.5 Gemeinsame Operationen für Sequenzen
- 7.1.6 Spezielle Operationen für Listen
- 7.1.7 Sortieren
- 7.1.8 Eine Liste erzeugen
- 7.2 Projekt: Telefonliste
- 7.3 Dictionaries
  - 7.3.1 Operationen für Dictionaries
  - 7.3.2 Ein Dictionary ändern
- 7.4 Projekt: Vokabeltrainer
- 7.5 Projekt: Routenplaner
  - 7.5.1 Verkehrswege und Graphen
  - 7.5.2 Programmierung
- 7.6 Rückblick
- 7.7 Übungen
- 7.8 Lösungen zu den Fragen

## **8 Daten speichern**

- 8.1 Wie werden Daten gespeichert?
  - 8.1.1 Dateien öffnen
  - 8.1.2 Stream-Methoden
  - 8.1.3 Texte speichern und laden
  - 8.1.4 Binärdateien und Bytestrings

- 8.1.5 Pfade im Verzeichnisbaum
- 8.2 Laufzeitfehler abfangen
  - 8.2.1 try...except
  - 8.2.2 try...except...finally
- 8.3 with-Anweisungen
- 8.4 Projekt: Logbuch
- 8.5 Datenstrukturen speichern und laden: Das Modul pickle
  - 8.5.1 Speichern
  - 8.5.2 Laden
- 8.6 Projekt: Digitaler Planer
- 8.7 Daten im JSON-Format speichern
  - 8.7.1 Aufbau eines JSON-Texts
  - 8.7.2 Die Grenzen von JSON
- 8.8 Projekt: Temperaturdaten
  - 8.8.1 Writer
  - 8.8.2 Reader
- 8.9 Daten aus dem Internet
- 8.10 Projekt: Digitale Bibliothek
- 8.11 Rückblick
- 8.12 Übung: News-Check
- 8.13 Lösungen zu den Fragen

## **9 Textverarbeitung**

- 9.1 Unicode-Nummern für Zeichen
- 9.2 Escape-Sequenzen

9.3 Stringmethoden

9.4 Projekt: Goethes Wortschatz

9.5 Projekt: Wie warm wird es heute?

9.6 Ausblick: Reguläre Ausdrücke \*

9.6.1 Was ist ein regulärer Ausdruck?

9.6.2 Aufbau eines regulären Ausdrucks

9.6.3 Textpassagen finden mit findall()

9.6.4 Platzhalter für Zeichen aus einer Zeichenmenge

9.6.5 Reguläre Ausdrücke verknüpfen

9.6.6 Quantoren

9.6.7 Sonderzeichen maskieren

9.6.8 Gieriges oder nicht gieriges Finden

9.6.9 Webscraping mit regulären Ausdrücken

9.7 Texte mit variablen Teilen

9.7.1 Formatierung

9.7.2 Platzhalter mit Namen

9.7.3 Formatangaben für Platzhalter

9.8 Projekt: Textanalyse

9.9 Projekt: Storytelling

9.10 Rückblick

9.11 Übungen

9.12 Lösungen zu den Fragen

## **10 Zugriff auf die Systemumgebung**

10.1 Schnittstelle zum Betriebssystem: Das Modul os

10.2 Suchen und Eigenschaften ermitteln

- 10.2.1 Unterverzeichnisse ausgeben
- 10.2.2 Verzeichnisbaum durchlaufen
- 10.3 Dateien und Verzeichnisse anlegen und umbenennen
  - 10.3.1 Projekt: Bilddateien umbenennen
- 10.4 Das Modul sys - die Schnittstelle zum Laufzeitsystem
  - 10.4.1 Informationen über die aktuelle Systemumgebung abfragen
  - 10.4.2 Kommandozeilenargumente abfragen
  - 10.4.3 Blick hinter die Kulissen: Speicherverwaltung \*
  - 10.4.4 Zugriff auf Module
  - 10.4.5 Die Standardausgabe in eine Datei umleiten
- 10.5 Rückblick
- 10.6 Übungen
- 10.7 Lösung zu der Frage: Welcher Kommentar passt?

## **11 Grafische Benutzungsoberflächen**

- 11.1 Widgets
- 11.2 Das Anwendungsfenster Tk
- 11.3 Ein Widget einfügen
- 11.4 Das Aussehen der Widgets gestalten
  - 11.4.1 Die Größe eines Widgets
- 11.5 Gemeinsame Methoden der Widgets
- 11.6 Schaltflächen und Eventhandler
  - 11.6.1 Projekt: Motivator
- 11.7 Das Layout verfeinern

11.8 Raster-Layout

11.9 Projekt: 25 Farben – ein automatisches Farbfelder-Bild

11.10 Widgets zur Texteingabe

11.10.1 Einzeilige Eingabe: Das Entry-Widget

11.10.2 Mehrzeilige Eingabe: Das Text-Widget

11.10.3 Projekt: Reimen mit Goethe

11.11 Radiobuttons

11.11.1 Projekt: Währungsrechner

11.12 Dialogboxen

11.12.1 Projekt: Texteditor

11.13 Parallele Abläufe: Threads

11.13.1 Ein Experiment: Countdown

11.13.2 Eine Funktion in einem eigenen Thread ausführen

11.14 Rückblick

11.15 Übungen

11.16 Lösungen zu den Fragen

## **12 Grafik programmieren**

12.1 Bilder auf Schaltflächen und Labels

12.1.1 Projekt: Würfelspiel

12.1.2 Bilder verändern

12.1.3 Projekt: Graustufen

12.2 Canvas

12.2.1 Flächen gestalten

12.2.2 Linien gestalten

- 12.2.3 ID-Nummern: Elemente löschen oder bewegen
- 12.3 Projekt: Creative Coding
- 12.4 Die Python Imaging Library (PIL)
  - 12.4.1 Ein Image-Objekt aus einer Datei gewinnen
  - 12.4.2 Ein Image-Objekt ohne Datei erzeugen
  - 12.4.3 Attribute und Methoden von Image-Objekten
  - 12.4.4 Bilder über Listen verarbeiten
  - 12.4.5 Bilder einfügen
  - 12.4.6 Projekt: Greenscreen
  - 12.4.7 PIL.Image-Objekte in tkinter-Anwendungen
  - 12.4.8 Projekt: Webcam-Viewer
- 12.5 Rückblick
- 12.6 Übungen

## **13 Fehler finden und vermeiden**

- 13.1 Zusicherungen
- 13.2 Tracing
  - 13.2.1 Beispiel: Quicksort
- 13.3 Debugging mit IDLE
  - 13.3.1 Der Debugger der Python-Shell
  - 13.3.2 Das Programm schrittweise durchlaufen
  - 13.3.3 Haltepunkte setzen
- 13.4 Debugging mit Thonny
- 13.5 Rückblick
- 13.6 Lösungen zu den Fragen

## **14 Objektorientierte Programmierung**

## 14.1 Klassen und Objekte

14.1.1 Was ist Objektorientierung?

14.1.2 Klassen entwerfen und grafisch darstellen – UML

14.1.3 Definition einer Klasse

14.1.4 Objekte einer Klasse erzeugen: Instanziierung

14.1.5 Auf Attribute zugreifen

14.1.6 Methoden aufrufen

14.1.7 Objekte mit variablen Anfangswerten

14.1.8 Metaphern in der Programmierung

## 14.2 Projekt: Geld

14.2.1 Mit Geld-Objekten rechnen

14.2.2 Klassenattribute

14.2.3 Operatoren überladen – Polymorphie

## 14.3 Magische Methoden

## 14.4 Projekt: Abrechnung

## 14.5 Vererbung

## 14.6 Projekt: Farbtester

## 14.7 Projekt: Zahlenregen

## 14.8 Rückblick

## 14.9 Übungen

## 14.10 Lösungen zu den Fragen

# **15 Datenbanktechnik**

15.1 Was ist ein Datenbanksystem?

15.2 Eine Datenbank entwerfen –  
das Entity-Relationship-Diagramm (ER)

15.3 Relationale Datenbanken

15.4 Relationen mit Python darstellen \*

15.4.1 Menge von Tupeln

15.4.2 Benannte Tupel (named tuples)

15.5 Das Modul sqlite3 – Schnittstelle zu einer SQL-Datenbank

15.5.1 Mit SQL Tabellen anlegen und Tupel eintragen

15.5.2 Mit sqlite3 eine SQLite-Datenbank aufbauen

15.5.3 Formulierung von Anfragen (Queries) mit SQL

15.5.4 Datenbankanfragen in Python-Programmen

15.5.5 SQL-Anweisungen mit variablen Teilen

15.6 Projekt: Zitatesammlung

15.6.1 ER-Diagramm

15.6.2 Tabellen (Beispiel)

15.6.3 Administration der Zitatesammlung

15.6.4 Nach Zitaten suchen

15.7 Rückblick

15.8 Übungen

15.9 Lösungen zu den Fragen

## **16 Wissenschaftliche Projekte**

16.1 NumPy – Rechnen mit Arrays

16.1.1 Arrays

16.1.2 Indizieren

16.1.3 Slicing

16.1.4 Arrays verändern

- 16.1.5 Arithmetische Operationen
- 16.1.6 Funktionen, die elementweise ausgeführt werden
- 16.1.7 Matrizenmultiplikation mit dot()
- 16.1.8 Array-Funktionen und Achsen
- 16.2 Datenvisualisierung mit matplotlib
  - 16.2.1 Liniendiagramme
  - 16.2.2 Mehrere Linien in einem Diagramm
  - 16.2.3 Histogramme
  - 16.2.4 Projekt: Würfeln
  - 16.2.5 Heatmaps
- 16.3 Projekt: Bewegungsprofil
- 16.4 Google Colaboratory - Colab
  - 16.4.1 Ein Colab-Notebook erzeugen
  - 16.4.2 Text-Zellen
  - 16.4.3 Bilder einfügen
  - 16.4.4 Notebooks speichern und öffnen
- 16.5 Projekt: Füchse und Hasen - Simulation eines Räuber-Beute-Systems
  - 16.5.1 Notebooks teilen
- 16.6 Rückblick
- 16.7 Übungen
- 16.8 Lösungen zu den Fragen

## **17 Dynamische Webseiten: CGI und WSGI**

- 17.1 Dynamische Webseiten mit CGI
  - 17.1.1 Projekt: Wie spät ist es?

- 17.1.2 Die Ausgabe eines CGI-Skripts
- 17.1.3 Wie ist ein CGI-Skript aufgebaut?
- 17.1.4 CGI-Skripte unter Windows
- 17.1.5 Aufruf mit dem Webbrowser
- 17.1.6 Ein HTTP-Server
- 17.1.7 Zugriff von einem anderen Computer im lokalen Netz
- 17.2 Interaktive Webseiten
  - 17.2.1 Eingabekomponenten in einem HTML-Formular
  - 17.2.2 Verarbeitung von Eingabedaten mit FieldStorage
- 17.3 Wie verarbeitet man Umlaute? \*
- 17.4 Dynamische Webseiten mit WSGI
  - 17.4.1 Das Applikationsobjekt
  - 17.4.2 Skripte mit eigenem HTTP-Server – das Modul wsgiref
- 17.5 Projekt: Wie spät ist es? (II)
- 17.6 Projekt: Umfrage
  - 17.6.1 Die HTML-Schablonen
  - 17.6.2 Der algorithmische Teil
- 17.7 Einen Hosting-Dienst nutzen
  - 17.7.1 Python Anywhere
  - 17.7.2 Das vorgefertigte OWSGI-Programm ausprobieren
  - 17.7.3 Projekt: Wie spät ist es? (III)
  - 17.7.4 WSGI-Projekte modularisieren
- 17.8 Rückblick

17.9 Übungen

17.10 Lösung zur Frage: Interaktive Webseite

## **18 Professionelle Software-Entwicklung**

18.1 Die Laufzeit von Programmen

18.1.1 Schnelles Sortieren – Quicksort versus Straight Selection

18.1.2 Performance-Analyse mit dem Profiler

18.2 Agile Software-Entwicklung

18.2.1 Software Engineering

18.2.2 Einige Grundideen der agilen Software-Entwicklung

18.3 Projekt: Digitales Notizbuch

18.3.1 Stories

18.3.2 Erste Iteration

18.3.3 Zweite Iterationen

18.3.4 Refactoring

18.3.5 Neue Stories und Änderbarkeit

18.4 Test Driven Development mit doctest

18.5 Übung: Ticketbuchung

## **Glossar**



# Einleitung

## Python in Studium und Ausbildung

In vielen Berufen – auch außerhalb der Informationstechnik – werden heute Programmierkenntnisse als Basiskompetenz vorausgesetzt. Selbst wenn Ihr Schwerpunkt nicht die professionelle Softwareentwicklung ist, werden Sie in Rahmen von wissenschaftlichen Projekten oder in der Berufspraxis Computerprogramme schreiben oder an Entwicklungen beteiligt sein. Darüber hinaus schult das Programmieren das logische Denken. Wer programmieren kann, ist besser in der Lage, Probleme zu analysieren und Lösungen zu finden.

Dieses Buch wendet sich vor allem an Menschen, die im Rahmen eines Studiums oder einer beruflichen Ausbildung einen Einstieg in die Programmierung mit Python suchen. Es lässt sich sowohl als Materialgrundlage für einen Programmierkurs als auch für das eigenständige Lernen einsetzen.

Alle Erklärungen sind leicht verständlich formuliert und setzen keine Vorkenntnisse voraus. Am besten lesen Sie das Buch neben der Computer-Tastatur und probieren die Programmbeispiele gleich aus. Zahlreiche praktische Programmier-Übungen helfen Ihnen, Ihr neues Wissen zu verinnerlichen. Sie werden schnell erste Erfolge erzielen und Freude an der Programmierung finden.

## Der Aufbau des Buchs

Das Buch beginnt mit den Grundlagen: Installation von Python, Nutzung der Entwicklungsumgebung und Formulierung einfacher Anweisungen. Sie lernen Schritt für Schritt, wie man Daten lädt, verarbeitet und speichert, und erhalten eine Einführung in die Verwendung von Funktionen und Modulen, objektorientierte Programmierung und die Gestaltung von grafischen Benutzungsoberflächen.

In den hinteren Kapiteln wenden Sie die gelernten Python-Konzepte in wichtigen und spannenden Gebieten der Informatik an: Datenbanktechnik, Bildverarbeitung, wissenschaftliches Rechnen mit NumPy, Visualisierung von Daten mit Matplotlib und Internetprogrammierung.

Abschnitte, die mit einem Sternchen \* versehen sind, können Sie überspringen, wenn Sie das Thema nicht interessiert. Sie behandeln sehr spezielle Inhalte, die für das Verständnis des nachfolgenden Texts nicht benötigt werden.

Das letzte Kapitel schließlich gibt einen Einblick in fortgeschrittene Techniken (z.B. das Aufspüren von Schwachstellen im Programm mit einer Performance-Analyse) und zeigt Ihnen einige Ideen des agilen Programmierens, die helfen können, ein größeres Softwareprojekt erfolgreich zu planen und durchzuführen.

Gelegentlich stoßen Sie auf Zwischenfragen. Sie sind als kleine Lernaktivierung gedacht und werden am Ende des Kapitels beantwortet. Jedes Kapitel schließt mit praktischen Programmier-Übungen, in denen Sie Ihr neu gewonnenes Wissen vertiefen können. Mit Sternchen \* wird der Schwierigkeitsgrad der Aufgaben gekennzeichnet. Je mehr Sternchen, desto schwieriger. Die Lösungen zu diesen Übungen, die meist viel Programmtext enthalten, stehen in einem Online-Kapitel zum Download zur Verfügung. Mehr dazu im übernächsten Abschnitt.

Am Ende des Buchs finden Sie ein Glossar mit den wichtigsten Fachbegriffen sowie ein Stichwortverzeichnis, das Ihnen hilft, bestimmte Themen im Buch schneller zu finden.

## Achten Sie auf den Schrifttyp!

In diesem Buch hat der Schrifttyp eine Bedeutung. Das soll das Lesen erleichtern. Alle Programmtexte oder Teile von Programmtexten (wie z.B. Variablennamen) sind in einer nicht proportionalen Schrift (Monotype-Schrift) gesetzt.

**Beispiel:** Die Variable `name` hat den Wert `'Jessy'`.

In einigen Passagen der Programmtexte kommen *kursiv* gesetzte Monotype-Texte vor, die als Platzhalter gemeint sind. In einem Programm würde man den Platzhalter durch einen anderen, in den Zusammenhang passenden Text ersetzen.

**Beispiel:** Bei

```
stream = open(dateiname)
```

sind *stream* und *dateiname* Platzhalter.

In Textpassagen, die einen Dialog mit dem Computer wiedergeben, ist der Text, den ein Mensch eingegeben hat, etwas heller gesetzt als der Text, den der Computer ausgibt.

**Beispiel:**

```
Dein Name: Helena  
Guten Morgen Helena!
```

## Programmtexte und Lösungen zum Download

Das Buch enthält viele kleine Beispielprogramme. Sie sind als »Starterprojekte« gedacht und sollen Sie ermuntern, den Code weiterzuentwickeln und eigene Ideen umzusetzen.

Alle Programmtexte sowie die Lösungen zu den Übungen stehen Ihnen auf der Webseite des Verlags unter <https://www.mitp.de/0434> zum Download zur Verfügung. Dort finden Sie gegebenenfalls auch eine Errata-Liste. Wenn Sie einen Fehler finden, würde ich mich über eine Rückmeldung freuen. Am besten schreiben Sie eine E-Mail an [lektorat@mitp.de](mailto:lektorat@mitp.de).

Ich wünsche Ihnen viel Erfolg und Spaß bei der Programmierung mit Python!

*Michael Weigend*