

16.2	Datenvisualisierung mit matplotlib	343
16.2.1	Liniendiagramme	344
16.2.2	Mehrere Linien in einem Diagramm	346
16.2.3	Histogramme	347
16.2.4	Projekt: Würfeln	348
16.2.5	Heatmaps	349
16.3	Projekt: Bewegungsprofil	350
16.4	Google Colaboratory – Colab	354
16.4.1	Ein Colab-Notebook erzeugen	354
16.4.2	Text-Zellen	356
16.4.3	Bilder einfügen	358
16.4.4	Notebooks speichern und öffnen	359
16.5	Projekt: Füchse und Hasen – Simulation eines Räuber-Beute- Systems	360
16.5.1	Notebooks teilen	363
16.6	Rückblick	364
16.7	Übungen	365
16.8	Lösungen zu den Fragen	367
17	Dynamische Webseiten: CGI und WSGI	369
17.1	Dynamische Webseiten mit CGI	370
17.1.1	Projekt: Wie spät ist es?	371
17.1.2	Die Ausgabe eines CGI-Skripts	372
17.1.3	Wie ist ein CGI-Skript aufgebaut?	372
17.1.4	CGI-Skripte unter Windows	373
17.1.5	Aufruf mit dem Webbrowser	373
17.1.6	Ein HTTP-Server	374
17.1.7	Zugriff von einem anderen Computer im lokalen Netz	375
17.2	Interaktive Webseiten	375
17.2.1	Eingabekomponenten in einem HTML-Formular	377
17.2.2	Verarbeitung von Eingabedaten mit FieldStorage	379
17.3	Wie verarbeitet man Umlaute? *	380
17.4	Dynamische Webseiten mit WSGI	382
17.4.1	Das Applikationsobjekt	382
17.4.2	Skripte mit eigenem HTTP-Server – das Modul wsgiref ...	383
17.5	Projekt: Wie spät ist es? (II)	383
17.6	Projekt: Umfrage	386
17.6.1	Die HTML-Schablonen	387
17.6.2	Der algorithmische Teil	388

17.7	Einen Hosting-Dienst nutzen	390
17.7.1	Python Anywhere	390
17.7.2	Das vorgefertigte OWSGI-Programm ausprobieren	390
17.7.3	Projekt: Wie spät ist es? (III)	393
17.7.4	WSGI-Projekte modularisieren	394
17.8	Rückblick	395
17.9	Übungen	395
17.10	Lösung zur Frage: Interaktive Webseite	397
18	Professionelle Software-Entwicklung	399
18.1	Die Laufzeit von Programmen	399
18.1.1	Schnelles Sortieren – Quicksort versus Straight Selection .	399
18.1.2	Performance-Analyse mit dem Profiler	402
18.2	Agile Software-Entwicklung	403
18.2.1	Software Engineering	403
18.2.2	Einige Grundideen der agilen Software-Entwicklung	404
18.3	Projekt: Digitales Notizbuch	405
18.3.1	Stories	405
18.3.2	Erste Iteration	406
18.3.3	Zweite Iterationen	407
18.3.4	Refactoring	408
18.3.5	Neue Stories und Änderbarkeit	412
18.4	Test Driven Development mit doctest	413
18.5	Übung: Ticketbuchung	415
19	Glossar	417
20	Stichwortverzeichnis	425



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!
```