

Funktionale Programmierung mit **Python**

» Hier geht's
direkt
zum Buch

DAS VORWORT

1

Vorwort

Willkommen zu unserem Buch über funktionale Programmierung in Python. In einer Welt, die zunehmend von datengetriebenen Entscheidungen und komplexen Systemen geprägt ist, wird die Fähigkeit immer wichtiger, robuste und elegante Software zu entwickeln. Dieses Buch bietet eine fundierte Einführung in die funktionale Programmierung unter Python, ein Paradigma, das eine andere Herangehensweise an das Programmieren bietet und helfen kann, Probleme auf eine klarere und effizientere Weise zu lösen.

Obwohl Python keine rein funktionale Sprache ist, bietet sie eine Vielzahl von funktionalen Werkzeugen und Konzepten, die es wert sind, erkundet zu werden. Dieses Buch ist darauf ausgelegt, die Prinzipien und die Möglichkeiten der funktionalen Programmierung unter Python aufzuzeigen und zu demonstrieren, wie man diese Prinzipien effektiv in der Praxis anwenden kann. Neben theoretischen Erklärungen legen wir großen Wert auf praktische Anwendungen, um das Gelernte direkt umzusetzen. Jedes Kapitel enthält praktische Beispiele und Aufgaben, um die erlernten Fähigkeiten weiterzuentwickeln und die Kenntnisse zu vertiefen.

Dieses Buch richtet sich sowohl an erfahrene Programmierer aus anderen Programmiersprachen, die ihre Kenntnisse in Python und funktionaler Programmierung erweitern möchten, als auch an Anfänger mit geringen Kenntnissen in Python, die ein neues Paradigma kennenlernen wollen.

Viel Erfolg und Freude beim Lesen des Buches und beim Ausprobieren der zahlreichen Beispiele und Aufgaben!

2

Danksagung

Ein Buch entsteht über einen langen Zeitraum und ist das Ergebnis der gemeinsamen Anstrengung vieler engagierter Menschen. Neben den Autoren, die den wesentlichen Beitrag leisten, sind auch zahlreiche andere Personen direkt oder indirekt an der Entstehung beteiligt. Unser besonderer Dank gilt Karola und Melisa, die die zusätzliche Belastung mit großem Verständnis und Geduld getragen haben.

Ein solches Werk wäre ohne die Unterstützung eines gut funktionierenden und anerkannten Verlages nicht möglich. Deshalb möchten wir uns herzlich beim Hanser Verlag bedanken, der uns die Gelegenheit gegeben hat, dieses Buch zu veröffentlichen. Unser besonderer Dank gilt Frau Brigitte Bauer-Schiewek und Frau Kristin Rothe für ihre kontinuierliche und hervorragende Unterstützung sowie ihr unermüdliches Engagement, das dieses Projekt maßgeblich vorangebracht hat.

Ein weiterer Dank gilt den zahlreichen Teilnehmerinnen und Teilnehmern unserer Python-Kurse, deren Fragen, Anregungen und Rückmeldungen uns geholfen haben, unsere didaktischen und fachlichen Fähigkeiten stetig weiterzuentwickeln – Fähigkeiten, die beim Verfassen dieses Buches von unschätzbarem Wert waren. Ein besonderer Dank geht zudem an die Besucherinnen und Besucher unserer Online-Tutorials auf www.python-kurs.eu und www.python-course.eu, insbesondere an diejenigen, die uns durch konstruktive Anmerkungen und wertvolles Feedback unterstützt haben.

Wir möchten auch Herrn Jürgen Dubau für das hervorragende Lektorat danken, das wesentlich zur sprachlichen Klarheit und Präzision dieses Werkes beigetragen hat. Ebenso gebührt unser Dank Herrn Stephan Korell und Frau Irene Weilhart für ihre wertvolle Unterstützung bei der Umsetzung in LaTeX. Ihre Expertise und Sorgfalt haben die technische Umsetzung dieses Buches erheblich erleichtert.

Abschließend möchten wir all jenen danken, die durch ihre kritischen Anmerkungen, ihr Lob und ihre Unterstützung dazu beigetragen haben, dass dieses Buch in seiner jetzigen Form entstehen konnte. Ihre Beiträge sind für uns von unschätzbarem Wert.

Bernd Klein, Singen
Philip Klein, Freiburg

August 2024

3

Einleitung

■ 3.1 Einführung

Dieses Buch widmet sich einem speziellen und faszinierenden Aspekt der Programmierung mit Python: der funktionalen Programmierung. Während Python vor allem für seine Einfachheit und Vielseitigkeit bekannt ist, bietet es auch mächtige Werkzeuge und Paradigmen, die es ermöglichen, Probleme auf eine neue und elegante Weise zu lösen. Eines dieser Paradigmen ist die funktionale Programmierung.

Funktionale Programmierung ist ein Programmierparadigma, das darauf abzielt, Berechnungen als Auswertung mathematischer Funktionen zu behandeln und Zustandsveränderungen zu vermeiden. Dies steht im Gegensatz zum imperativen Paradigma, das den Schwerpunkt auf die Ausführung von Befehlen und die Manipulation von Zuständen legt. Durch die Anwendung der Prinzipien der funktionalen Programmierung können Programme oft klarer, kompakter und weniger fehleranfällig gestaltet werden.

■ 3.2 Zielsetzung des Buches

Dieses Buch ist nicht als umfassende und systematische Einführung in die funktionale Programmierung konzipiert. Der Grund dafür liegt darin, dass Python keine rein funktionale Programmiersprache ist, sondern eine vielseitige Sprache, die sowohl imperative als auch objektorientierte Paradigmen unterstützt. Obwohl Python viele funktionale Prinzipien integriert, bietet es oft Kompromisse zugunsten dieser anderen Paradigmen. Der Schwerpunkt dieses Buches liegt daher auf den Bereichen der funktionalen Programmierung, in denen Python besonders stark ist. Dies wird durch zahlreiche praktische Anwendungen und anschauliche Beispiele verdeutlicht. Jedes Kapitel endet mit einer Reihe von Aufgaben, die das Gelernte vertiefen und erweitern. Zu allen Aufgaben werden ausführliche Lösungen bereitgestellt, um das Verständnis zu fördern und die Lernziele zu erreichen.

■ 3.3 Aufbau des Buches

Das Buch besteht aus zwei Hauptteilen:

1. Python-Grundlagen unter funktionalen Aspekten
2. Funktionale Programmierung

Im ersten Teil des Buches befassen wir uns mit den grundlegenden Datenstrukturen von Python wie Integer, Gleitkommazahlen, Strings, Listen, Tupel und Dictionaries. Darüber hinaus behandeln wir die Kontrollstrukturen, darunter bedingte Anweisungen und Schleifen, wobei wir dabei auf die Besonderheiten im Vergleich zur funktionalen Programmierung bereits ein wenig eingehen.

Das Modul `collections` in Python ist wichtig, weil es erweiterte Datenstrukturen wie `deque`, `defaultdict`, `Counter` und `namedtuple` bietet, die über die Standard-Datenstrukturen hinausgehen und häufige Programmieraufgaben erleichtern. Diese spezialisierten Container ermöglichen effizientere und lesbarere Lösungen für komplexe Aufgabenstellungen. In der funktionalen Programmierung erleichtert das Modul `collections` die Arbeit mit unveränderlichen und strukturierten Daten, indem es z. B. `namedtuple` bereitstellt, um benannte und unveränderliche Datentypen zu erstellen. Außerdem bietet `Counter` eine praktische Möglichkeit, Daten zu aggregieren und zu zählen, was funktionale Techniken wie Map- und Reduce-Operationen unterstützen kann.

Da die funktionale Programmierung bei all diesen Themen nicht im Vordergrund steht, haben wir sie in diesem ersten Teil des Buches zusammengefasst. Dieser Abschnitt richtet sich insbesondere an Personen, die bereits über Programmiererfahrung verfügen, jedoch noch keine Vorkenntnisse in Python besitzen.

Im zweiten Teil geht es dann um die funktionalen Aspekte. Zunächst bemühen wir uns um eine Begriffsbestimmung des funktionalen Programmierstils und einer Abgrenzung zu anderen Programmierparadigmen in [Kapitel 7 \(Begriffsbestimmung\)](#). Das folgende [Kapitel 8 \(Funktionen\)](#) widmet sich ganz den Funktionen: beginnend bei einfachen Funktionen, lambda-Funktionen und rekursiven Funktionen bis hin zum Prinzip der First-Class-Funktionen. Dann sind wir bereit für das Thema Dekoratoren in [Kapitel 9 \(Dekoratoren\)](#). Dekoratoren in Python ermöglichen es, das Verhalten von Funktionen oder Methoden modular und wiederverwendbar zu erweitern oder zu modifizieren, ohne den ursprünglichen Code zu verändern. In diesem Kapitel finden sich viele interessante Anwendungen für Dekoratoren in Form von Beispielen und Aufgaben. Einer der faszinierendsten Anwendungen, nämlich der Memoisation, ist ein eigenes [Kapitel 10 \(Memoisation\)](#) gewidmet.

Die Komposition von Funktionen ist in der funktionalen Programmierung von großer Bedeutung, da sie hilft, sauberen, wartbaren und flexiblen Code zu schreiben. Sie fördert die Modularität, Lesbarkeit und Wiederverwendbarkeit von Funktionen, unterstützt die Vermeidung von Seiteneffekten und ermöglicht eine höhere Abstraktionsebene, die die Entwicklung komplexer Systeme vereinfacht. Dieses Thema behandeln wir deshalb auch in einem eigenen [Kapitel 12 \(Komposition von Funktionen\)](#).

Mit dem Thema „Currying“ beschäftigen wir uns in [Kapitel 13 \(Currying in Python\)](#). Dabei handelt es sich um ein zentrales Konzept der funktionalen Programmierung, das die

Umwandlung von Funktionen mit mehreren Argumenten in eine Serie von Funktionen mit jeweils einem Argument ermöglicht. Dies verbessert die Modularität und Wiederverwendbarkeit des Codes. In Python unterstützt Currying elegante und flexible Code-Strukturen, die den Prinzipien der funktionalen Programmierung entsprechen.

Funktionale Programmierung und objektorientierte Programmierung sind mächtige, aber unterschiedliche Paradigmen. In [Kapitel 14 \(Funktionale Emulation von OOP-Konzepten\)](#) zeigen wir, dass es sich nicht um unvereinbare oder gegensätzliche Paradigmen handelt. Wir zeigen, wie wir Klassendefinitionen in der funktionalen Programmierung durch First-Class-Funktionen unter Wahrung der objektorientierten Ziele realisieren können. Zudem demonstrieren wir, dass sich selbst objektorientierte Konzepte wie Vererbung funktional emulieren lassen.

Iteratoren und Generatoren sind in Python von großer Bedeutung, da sie eine effiziente Speicherverwaltung ermöglichen. Anstatt Daten vollständig auf einmal zu laden, erzeugen und verarbeiten sie diese bei Bedarf. Dies führt zu einer verbesserten Leistung und Skalierbarkeit von Programmen, insbesondere bei der Verarbeitung großer Datenmengen oder endloser Sequenzen. Deshalb haben wir dieser Thematik gleich drei Kapitel gewidmet. In [Kapitel 15 \(Generatoren und Iteratoren\)](#) zeigen wir, wie man mit Generatorfunktionen Iteratoren erzeugt, die Werte schrittweise mit dem Schlüsselwort `yield` zurückgeben, wodurch sie speichereffizient große Datenmengen oder unendliche Sequenzen verarbeiten können. In [Kapitel 16 \(Iteratoren der Standardbibliothek\)](#) lernen wir dann wichtige Iteratoren wie `zip`, `enumerate`, `map`, `filter`, `reduce`, der Standardbibliothek kennen. In den Beispielen demonstrieren wir, wie man mit ihrer Hilfe effizienteren Code schreiben kann. Das effizientere Laufzeitverhalten bei der Verwendung von Iteratoren und Generatoren kann allgemein dadurch erklärt werden, dass sie Daten „on-the-fly“ erzeugen, anstatt ganze Listen oder andere Datenstrukturen im Speicher zu halten.

Besonders intensiv gehen wir in [Kapitel 17 \(Das Modul `itertools`\)](#) auf das Modul `itertools` ein, da es im Zusammenhang mit funktionaler Programmierung von besonderer Wichtigkeit ist, weil es eine Sammlung von effizienten, speichersparenden Iteratoren zur Verfügung stellt, die sich ideal für funktionale Programmierparadigmen eignen. Diese Werkzeuge ermöglichen es, komplexe Iterationen und Kombinationen von Daten in einem deklarativen Stil auszudrücken, ohne explizit Schleifen oder temporäre Datenstrukturen zu verwenden. Dies führt zu sauberem, lesbarem und wartbarem Code, der die Prinzipien der funktionalen Programmierung wie Unveränderlichkeit und First-Class-Funktionen unterstützt.

Die `itertools`-Bibliothek in Python bietet vielseitige Funktionen für die Arbeit mit Iterationen. Zu den wichtigsten gehören unendliche Iteratoren wie `count`, `cycle` und `repeat`, die unendlich lange Sequenzen erzeugen. Endliche Kombinations- und Permutations-Iteratoren wie `chain`, `product`, `permutations` und `combinations` ermöglichen die Erzeugung komplexer Kombinationen und Anordnungen von Daten. Zudem bietet `itertools` leistungsfähige Filter- und Mapping-Tools wie `starmap`, `takewhile`, `dropwhile` und `accumulate`, die es ermöglichen, Sequenzen nach bestimmten Kriterien zu filtern und zu transformieren. Insgesamt erleichtert `itertools` die effiziente und speicherschonende Verarbeitung von Daten.

■ 3.4 Leserschaft

Auch wenn wir im Buch extra einen Abschnitt zu den Grundlagen von Python aufgenommen haben, wäre es dennoch hilfreich, bereits etwas Programmiererfahrung mit Python zu besitzen. Ansonsten richtet sich dieses Buch an Personen mit Programmiererfahrung in Sprachen wie C, C++, C#, JavaScript oder Java, die die funktionale Denkweise und die Programmierung in Python erlernen möchten. Es vermittelt sowohl das notwendige theoretische Wissen als auch praktische Fähigkeiten, um die Prinzipien der funktionalen Programmierung erfolgreich in der Praxis anzuwenden.

Letztlich ist es jedoch immer eine individuelle Entscheidung, ob ein Buch für jemanden geeignet ist oder nicht. Kein Buch kann für jeden „ideal“ sein. Wir sind jedoch überzeugt, dass dieses Buch für viele Programmierende und Studierende neue Perspektiven eröffnen und ihren Programmierhorizont erheblich erweitern kann.

■ 3.5 Zusätzliche Unterlagen

Auf unserer Webseite <https://python-kurs.eu/funktional> stellen wir die Beispiele und Lösungen als Jupyter-Notebooks zur Verfügung. Denn was nutzt das beste Buch, wenn man den Code der Programme nicht zur Verfügung hat und mühsam eintippen muss, um Beispiele und Aufgaben nachvollziehen zu können? Deshalb haben wir uns entschieden, Jupyter-Notebooks mit Programmcode online zur Verfügung zu stellen.