Einleitung

Die Programmiersprache Python besitzt einzigartige Stärken und Vorteile, die nicht immer ganz einfach zu verstehen sind. Mit anderen Sprachen vertraute Programmierer nähern sich Python oftmals nur zögerlich, anstatt die enorme Ausdrucksfähigkeit willkommen zu heißen. Andere übertreiben in entgegengesetzter Richtung und überstrapazieren Python-Funktionalitäten, was später große Probleme verursachen kann.

Dieses Buch möchte Ihnen einen Einblick geben in das, was im Englischen als *pythonic* (»Python-artig«) bezeichnet wird: die beste Art und Weise, Python zu verwenden. Dieser Programmierstil setzt grundlegende Kenntnisse der Sprache voraus, über die Sie, wie ich annehme, bereits verfügen. Programmieranfänger lernen die bewährten Vorgehensweisen zur Nutzung von Pythons Fähigkeiten kennen. Erfahrene Programmierer hingegen erfahren, wie sie souverän mit den Eigenheiten dieses neuen Werkzeugs zurechtkommen.

Ich möchte Sie darauf vorbereiten, mit Python große Wirkung zu erzielen.

Inhalt des Buchs

In den einzelnen Kapiteln des Buchs sind verschiedene Punkte aufgeführt, die jeweils weit gefasste, aber doch verwandte Themen betreffen. Es steht Ihnen frei, ganz nach Ihrem Interesse im Buch herumzublättern. Jeder Punkt enthält kompakte und präzise Anleitungen, die Ihnen dabei helfen sollen, effektivere Python-Programme zu schreiben. Sie finden dort Ratschläge, was zu tun und was besser zu lassen ist, Hinweise zum Eingehen von Kompromissen und Erklärungen, warum das eine oder das andere die bessere Wahl ist. Es gibt Querverweise, die Ihnen beim Lesen das Verständnis erleichtern sollen.

Die zweite Ausgabe dieses Buchs konzentriert sich vollständig auf Python 3 bis einschließlich Version 3.8 (siehe Punkt 1: *Kenntnis der Python-Version*). Die meisten Punkte der ersten Ausgabe wurden überarbeitet und sind nach wie vor vorhanden, bei vielen hat es jedoch beträchtliche Aktualisierungen gegeben. Bei einigen Punkten hat sich mein Rat im Vergleich zur ersten Ausgabe aufgrund der Best Practices, die sich bei der Weiterentwicklung von Python ergeben haben, sogar völlig verändert. Sollten Sie trotz der endgültigen Abkündigung zum 1. Januar 2020 noch immer vornehmlich Python 2 verwenden, dürfte die erste Ausgabe dieses Buchs für Sie nützlicher sein.

Python verfolgt hinsichtlich der Standardbibliothek die Philosophie, dass sozusagen die Batterien im Lieferumfang enthalten sein sollten. Viele andere Sprachen werden mit nur einigen wenigen gängigen Paketen ausgeliefert und Sie müssen sich selbst auf die Suche begeben, wenn Sie zusätzliche wichtige Funktionalitäten benötigen. Allerdings sind viele der Standardmodule so eng mit den typischen Spracheigenheiten Pythons verflochten, dass sie sehr wohl auch Bestandteil der Sprachspezifikation sein könnten. Viele dieser Pakete sind so eng mit typischem Python-Code verwoben, dass sie eigentlich auch Teil des Sprachumfangs sein könnten. Die Anzahl der Standardmodule ist zu groß, um sie in diesem Buch behandeln zu können. Dennoch habe ich diejenigen aufgenommen, deren Kenntnis und Nutzung ich für unverzichtbar halte.

Kapitel 1: Pythons Sicht der Dinge

In der Python-Community dient das englische Adjektiv *pythonic* zur Beschreibung von Code, der einem bestimmten Programmierstil folgt. Dieser Stil hat sich im Laufe der Zeit durch Erfahrung im Umgang mit der Sprache und die Zusammenarbeit der Community allmählich entwickelt. Dieses Kapitel beschreibt die beste Art und Weise, die gängigsten Aufgaben in Python zu erledigen.

Kapitel 2: Listen und Dictionaries

In Python ist die häufigste Methode zur Organisation von Informationen das Speichern einer Reihe von Werten in einer Liste. Die naheliegende Ergänzung dazu ist das Dictionary, das den Werten zugeordnete Schlüssel speichert. Dieses Kapitel beschreibt, wie sich Programme mit diesen vielfältigen Bausteinen erstellen lassen.

Kapitel 3: Funktionen

In Python besitzen Funktionen eine Vielzahl von Merkmalen, die einem Programmierer das Leben erleichtern. Einige sind den Fähigkeiten anderer Programmiersprachen ähnlich, viele gibt es jedoch nur in Python. Dieses Kapitel hat zum Thema, wie man Funktionen dazu verwendet, ihren Zweck zu verdeutlichen, ihre Wiederverwendung zu ermöglichen und Bugs zu vermeiden.

Kapitel 4: Listen-Abstraktionen und Generatoren

In Python gibt es eine spezielle Syntax, um Listen, Dictionaries und Sets schnell durchlaufen zu können und davon abgeleitete Datenstrukturen zu erzeugen. Ebenso ist es möglich, dass eine Funktion die durchlaufenen Werte der Reihe nach zurückgibt. Dieses Kapitel erläutert, wie diese Features eine bessere Leistung erbringen können, weniger Speicher benötigen und besser verständlich eingesetzt werden.

Kapitel 5: Klassen und Schnittstellen

Python ist eine objektorientierte Programmiersprache. Um Aufgaben in Python zu erledigen, ist es oft erforderlich, neue Klassen anzulegen und zu definieren, wie sie mittels ihrer Schnittstellen und der Klassenhierarchie interagieren. Dieses Kapitel erläutert, wie man Klassen zur Modellierung des beabsichtigten Verhaltens durch Objekte nutzt.

Kapitel 6: Metaklassen und Attribute

Metaklassen und dynamische Attribute sind leistungsfähige Python-Merkmale, die es allerdings auch ermöglichen, äußerst merkwürdiges und unerwartetes Verhalten zu implementieren. Dieses Kapitel stellt die üblichen Verwendungsweisen dieser Mechanismen vor, damit Sie dem *Prinzip der geringsten Überraschung* folgen können.

Kapitel 7: Nebenläufigkeit und parallele Ausführung

In Python können schnell und einfach Programme geschrieben werden, die scheinbar mehrere Aufgaben gleichzeitig erledigen. Python kann außerdem durch Systemaufrufe, Subprozesse oder C-Erweiterungen Code parallel ausführen. Dieses Kapitel führt vor, wie sich diese leicht unterschiedlichen Verfahren am besten einsetzen lassen.

Kapitel 8: Robustheit und Performance

Python bringt eine Reihe integrierter Features und Module mit, die dabei helfen, stabile und verlässliche Programme zu erstellen. Darüber hinaus bietet Python Tools, die es ermöglichen, mit minimalem Aufwand eine höhere Leistung zu erzielen. Dieses Kapitel beschreibt, wie Sie Python nutzen können, um die Stabilität und Effizienz Ihrer Programme zu maximieren.

Kapitel 9: Testen und Debuggen

In welcher Sprache Sie programmieren, spielt keine Rolle; Sie sollten Ihren Code stets testen. Pythons dynamische Features können allerdings das Risiko erhöhen, dass es zu Laufzeitfehlern kommt. Erfreulicherweise vereinfachen sie es jedoch auch, Tests zu schreiben und nicht richtig funktionierende Programme zu untersuchen. Dieses Kapitel befasst sich mit Pythons integrierten Werkzeugen zum Testen und Debuggen.

Kapitel 10: Zusammenarbeit

Bei der gemeinsamen Entwicklung von Python-Programmen muss dem Programmierstil Beachtung geschenkt werden. Selbst wenn Sie der einzige Programmierer sind, müssen Sie doch wissen, wie die von anderen Entwicklern erstellten Module

verwendet werden. Dieses Kapitel behandelt die Standardwerkzeuge und bewährte Vorgehensweisen, die es ermöglichen, gemeinsam an Python-Programmen zu arbeiten.

Konventionen dieses Buchs

Die Codebeispiele in diesem Buch sind in einer `nicht-proportionalen` Schrift gedruckt. Wenn lange Zeilen umbrochen werden müssen, verwende ich zur Markierung das Zeichen »\«. Weggelassene Codeabschnitte sind durch Auslassungszeichen (`...`) gekennzeichnet. Sie weisen darauf hin, dass weiterer Code vorhanden ist, der im gegebenen Zusammenhang jedoch nicht von Bedeutung ist. Sie müssen sich den vollständigen Beispielcode herunterladen (siehe unten), um die gekürzten Codeschnipsel ausführen zu können.

Ich habe mir hinsichtlich der üblichen Python-Stilregeln eine gewisse »künstlerische Freiheit« genommen, um die Darstellung an die Gegebenheiten eines Buchs anzupassen oder die wichtigsten Teile hervorzuheben. Außerdem habe ich aus Platzgründen die Docstrings entfernt. Das sollten Sie in Ihren eigenen Projekten allerdings nicht tun – folgen Sie besser den üblichen Stilregeln (siehe Punkt 2: *Stilregeln gemäß PEP 8*) und dokumentieren Sie Ihren Code (siehe Punkt 84: *Docstrings für sämtliche Funktionen, Klassen und Module*).

Viele Codeabschnitte enthalten auch die bei der Ausführung des Codes erfolgenden Ausgaben. Damit sind die Ausgaben auf der Konsole oder im Terminalfenster gemeint, die erscheinen, wenn das Programm im interaktiven Python-Interpreter ausgeführt wird. Vor den in `nicht-proportionaler` Schrift gedruckten Ausgaben steht eine Zeile mit den Zeichen `>>>` (die interaktive Python-Eingabeaufforderung). Dem liegt der Gedanke zugrunde, dass Sie die Beispiele in einer Python-Shell eingeben und die Ausgaben reproduzieren können.

Schließlich gibt es noch weitere Abschnitte in `nicht-proportionaler` Schrift ohne führende `>>>`-Zeile. Sie stellen Ausgaben des laufenden Programms dar (nicht des Interpreters). Am Anfang der Beispiele steht oftmals ein Dollarzeichen (`$`), das darauf hinweist, dass ich Programme von einer Shell wie Bash aus starte. Falls Sie die Befehle unter Windows oder einem anderen System ausführen, müssen Sie die Programmnamen und Argumente womöglich entsprechend anpassen.

Beispielcode herunterladen

Manche Beispiele in diesem Buch sollten besser als komplette Programme ohne die eingestreuten Texte betrachtet werden. Sie können sich den Quellcode auf der englischen Website zum Buch (`http://www.effectivepython.com`) herunterladen. Sie haben dann auch die Möglichkeit, mit dem Code herumzuexperimentieren und die Funktionsweise besser zu verstehen.

Über den Autor

Brett Slatkin ist bei Google als Führungskraft in der Softwareentwicklung tätig. Er ist leitender Ingenieur und Mitbegründer des Projekts Google Surveys, ist Miterfinder des PubSubHubbub-Protokolls und war an der Entwicklung von Googles erstem Cloud-Computing-Produkt (App Engine) beteiligt. Vor 14 Jahren sammelte er bei der Verwaltung von Googles riesigem Serverbestand erste Erfahrungen mit Python.

Neben der alltäglichen Arbeit spielt er gern Klavier und surft (beides mehr schlecht als recht). Auf seiner privaten Website (`http://www.onebigfluke.com`) schreibt er über Programmierung und damit verwandte Themen. Er erwarb seinen Bachelor of Science in technischer Informatik an der Columbia-Universität in New York und lebt in San Francisco.

Danksagungen

Ohne die Hilfe, Unterstützung und den Zuspruch vieler Menschen würde es dieses Buch nicht geben.

Dank an Scott Meyers für die Buchreihe *Effective Software Development*. Als ich 15 Jahre alt war, habe ich erstmals *Effective C++* gelesen und es war um mich geschehen. Es besteht kein Zweifel daran, dass Scotts Bücher zu meiner akademischen Laufbahn und meiner ersten Anstellung geführt haben. Ich bin begeistert, dass ich die Möglichkeit hatte, dieses Buch zu schreiben.

Dank an die technischen Korrekturleser für ihre ausführlichen und gründlichen Rückmeldungen zur zweiten Ausgabe dieses Buchs: Andy Chu, Nick Cohron, Andrew Dolan, Asher Mancinelli und Alex Martelli. Dank an meine Kollegen bei Google für die Durchsicht des Buchs. Ohne eure Hilfe wäre dieses Buch nicht vorstellbar.