Einleitung

Danke, dass Sie *Algorithmen in Python* gekauft haben. Python ist eine der populärsten Programmiersprachen der Welt, und Menschen mit verschiedensten Hintergründen werden Python-Programmierer. Einige haben eine formale Informatikausbildung. Andere lernen Python als Hobby. Wieder andere benutzen Python in ihrem beruflichen Umfeld, obwohl Softwareentwicklung nicht ihre Hauptaufgabe ist. Die Aufgaben in diesem Buch haben einen mittleren Schwierigkeitsgrad und helfen so erfahrenen Programmierern, Ideen aus ihrer Informatikausbildung aufzufrischen und dabei einige fortgeschrittene Features der Sprache zu erlernen. Programmierer im Selbststudium werden ihre Informatikausbildung beschleunigen, indem sie lernen, klassische Aufgaben in der Sprache ihrer Wahl zu lösen: Python. Dieses Buch behandelt eine so große Bandbreite von Problemlösungsstrategien, dass wirklich für jeden etwas dabei ist.

Dieses Buch ist keine Einführung in Python. Es gibt diverse hervorragende Bücher von Manning, dem Rheinwerk Verlag und anderen Verlagen, die diese Aufgabe erfüllen.[1] Wenngleich dieses Buch Python 3.7 voraussetzt, wird die Beherrschung jeder Facette der neuesten Version von Python nicht vorausgesetzt. Tatsächlich wurde das Buch mit dem Anspruch geschrieben, als Lernmaterial zu dienen, das Lesern hilft, genau diese Beherrschung zu erreichen. Andererseits ist dieses Buch nicht für Leser geeignet, denen Python vollkommen neu ist.

Warum Python?

Python wird in so unterschiedlichen Aufgabenfeldern wie Datenwissenschaft, Filmproduktion, Informatikausbildung, IT-Management und vielen anderen eingesetzt. Praktisch kein Gebiet der Computeranwendung ist von Python unberührt geblieben (außer vielleicht die Kernel-Entwicklung). Python ist beliebt wegen seiner Flexibilität, seiner schönen und knappen Syntax, seiner objektorientierten Reinheit und seiner lebendigen Community. Die starke Community ist wichtig, weil sie dafür sorgt, dass Python Neueinsteiger willkommen heißt und ein umfangreiches Ökosystem verfügbarer Bibliotheken aufweist, auf die Entwickler aufsetzen können.

1 Wenn Sie Ihre Reise durch Python gerade erst beginnen, sollten Sie sich zuerst The Quick Python Book, 3. Auflage, von Naomi Ceder (Manning, 2018) anschauen, bevor Sie mit diesem Buch anfangen. [deutschsprachige Alternative: »Einstieg in Python«, 6. Auflage, von Thomas Theis, Rheinwerk Verlag 2020]

Aus den vorgenannten Gründen wird Python manchmal als anfängerfreundliche Sprache betrachtet, und diese Charakterisierung ist wahrscheinlich korrekt. Die meisten Leute würden beispielsweise zustimmen, dass Python leichter zu lernen ist als C++, und es ist so gut wie sicher, dass seine Community netter zu Einsteigern ist. In der Folge lernen viele Leute Python, weil es zugänglich ist, und sie beginnen sehr schnell damit, die Programme zu schreiben, die sie schreiben möchten. Aber sie haben womöglich nie eine Informatikausbildung erhalten, in der ihnen die ganze Vielfalt verfügbarer, leistungsfähiger Problemlösungsstrategien beigebracht wurde. Wenn Sie zu den Programmierern gehören, die mit Python, aber nicht mit der Informatik vertraut sind, ist dieses Buch etwas für Sie.

Andere lernen Python als zweite, dritte, vierte oder fünfte Sprache, nachdem sie lange in der Softwareentwicklung gearbeitet haben. Bei ihnen wird es den Python-Lernfortschritt beschleunigen, wenn sie alte Aufgaben wiedersehen, mit denen sie bereits in anderen Sprachen zu tun hatten. Für sie kann dieses Buch der Wiederauffrischung vor einem Vorstellungsgespräch dienen oder sie mit Problemlösungsstrategien vertraut machen, deren Einsatz sie bei ihrer Arbeit zuvor nicht in Betracht gezogen haben. Ich würde ihnen raten, das Inhaltsverzeichnis zu überfliegen, um herauszufinden, ob es in diesem Buch Themen gibt, die sie begeistern.

Was ist eine klassische Informatikaufgabe?

Manche sagen, dass sich Computer zur Informatik verhalten wie Teleskope zur Astronomie. Wenn das der Fall ist, dann entspricht eine Programmiersprache vielleicht einer Teleskoplinse. In jedem Fall wird der Begriff »klassische Informatikaufgaben« hier für »Programmieraufgaben, die typischerweise im Grundstudium der Informatik gelehrt werden« verwendet.

Es gibt bestimmte Programmieraufgaben, die neuen Programmierern zum Lösen vorgelegt werden und verbreitet genug sind, um als Klassiker bezeichnet zu werden, ob im Rahmen des Unterrichts während eines Bachelor-Studiengangs (in Informatik, Software-Engineering und ähnlichen) oder in einem Programmierlehrbuch mittlerer Lernstufe (zum Beispiel einem Einführungsbuch über künstliche Intelligenz oder Algorithmen). Eine Auswahl solcher Aufgaben finden Sie in diesem Buch.

Die Aufgaben reichen von trivialen, die mit ein paar Zeilen Code gelöst werden können, bis hin zu komplexen, die den Aufbau von Systemen über mehrere Kapitel hinweg benötigen. Einige Aufgaben berühren die künstliche Intelligenz, während andere bloß etwas Nachdenken erfordern. Einige Aufgaben sind praktisch, andere eher kurios.

Welche Arten von Aufgaben gibt es in diesem Buch?

Kapitel 1 stellt Problemlösungsstrategien vor, die den meisten Leserinnen und Lesern vertraut vorkommen werden. Memoisation und Bit-Manipulation sind grundlegende Bausteine anderer Verfahren, die in späteren Kapiteln betrachtet werden.

Auf diese behutsame Einführung folgt Kapitel 2, das sich mit Suchaufgaben beschäftigt. Die Suche ist ein so umfangreiches Thema, dass man vermutlich die meisten Aufgaben in diesem Buch unter dieser Überschrift zusammenfassen könnte. Kapitel 2 stellt die grundlegenden Suchalgorithmen vor, darunter binäre Suche, Tiefensuche, Breitensuche und A*. Diese Algorithmen werden im Rest des Buchs wiederverwendet.

In Kapitel 3 schreiben Sie ein Framework zur Lösung einer breiten Palette von Aufgaben, die sich abstrakt durch Variablen mit beschränktem Wertebereich und zwischen ihnen geltenden Bedingungen beschreiben lassen. Dazu gehören Klassiker wie das Acht-Damen-Problem, das Einfärben der Landkarte Australiens oder das kryptoarithmetische Rätsel SEND+MORE=MONEY.

Kapitel 4 erforscht die Welt der Graphenalgorithmen, deren Anwendungsgebiete für Uneingeweihte überraschend vielfältig erscheinen. In diesem Kapitel schreiben Sie eine Graphen-Datenstruktur und verwenden sie zur Lösung diverser klassischer Optimierungsaufgaben.

Kapitel 5 beschäftigt sich mit genetischen Algorithmen, einem Verfahren, das weniger deterministisch ist als die meisten in diesem Buch behandelten, aber manchmal Probleme lösen kann, die traditionelle Algorithmen nicht innerhalb einer vernünftigen Zeitspanne lösen können.

Kapitel 6 behandelt k-Means-Cluster-Algorithmen und ist vielleicht das algorithmenspezifischste Kapitel im Buch. Diese Clustertechnik ist einfach zu implementieren, einfach zu verstehen und auf vielen Gebieten anwendbar.

Kapitel 7 versucht zu erklären, was ein neuronales Netzwerk ist, und gibt Lesern einen Einblick in den Aufbau eines sehr einfachen neuronalen Netzwerks. Das Kapitel strebt nicht danach, dieses faszinierende und sich stetig weiterentwickelnde Wissensgebiet umfassend zu behandeln. In diesem Kapitel schreiben Sie ein neuronales Netzwerk nach einfachen Prinzipien und ohne externe Bibliotheken, damit Sie genau sehen können, wie ein neuronales Netzwerk arbeitet.

In Kapitel 8 geht es um Adversarial Search in Zwei-Spieler-Spielen mit perfekter Information. Sie lernen einen Suchalgorithmus kennen, der Minimax genannt wird und verwendet werden kann, um einen künstlichen Gegner zu entwickeln, der Spiele wie Schach, Dame und Vier Gewinnt gut spielen kann.

Kapitel 9 behandelt schließlich interessante (und Spaß machende) Aufgaben, die nirgendwo anders ins Buch passten.

Für wen ist dieses Buch geeignet?

Dieses Buch eignet sich sowohl für fortgeschrittene als auch für erfahrene Programmierer. Erfahrene Programmierer, die ihre Python-Kenntnisse vertiefen möchten, finden hier hinreichend bekannte Aufgaben aus ihrer Informatik- oder Programmierausbildung. Fortgeschrittene Programmierer lernen diese klassischen Aufgaben in der Sprache ihrer Wahl kennen: Python. Entwickler, die sich auf Coding-Vorstellungsgespräche vorbereiten, finden in diesem Buch hilfreiches Vorbereitungsmaterial.

Neben professionellen Programmierern werden auch Studenten im Grundstudium der Informatik dieses Buch vermutlich hilfreich finden. Es unternimmt jedoch keinen Versuch, eine gründliche Einführung in Datenstrukturen und Algorithmen zu sein. *Dies ist kein Lehrbuch über Datenstrukturen und Algorithmen*. Sie werden auf seinen Seiten keine Beweise oder ausführliche Big-O-Notation finden. Stattdessen handelt es sich um ein zugängliches, praktisches Tutorial für Problemlösungsverfahren, die das Endergebnis von Kursen über Datenstrukturen, Algorithmen und künstliche Intelligenz sein sollten.

Noch einmal: Kenntnisse der Syntax und Semantik von Python werden vorausgesetzt. Einem Leser, der keinerlei Programmiererfahrung hat, bringt dieses Buch wenig, und ein Programmierer ohne jegliche Python-Erfahrung wird sich fast sicher schwertun. *Algorithmen in Python* ist mit anderen Worten ein Buch für aktive Python-Programmierer und Informatikstudenten.

Python-Version, Quellcode-Repository und Type-Hints

Der Quellcode in diesem Buch wurde in Übereinstimmung mit Version 3.7 der Sprache Python geschrieben. Er verwendet Features von Python, die erst seit Python 3.7 verfügbar sind, so dass ein Teil des Codes nicht mit älteren Versionen von Python funktionieren wird. Anstatt sich abzumühen, die Beispiele in einer älteren Version zum Laufen zu bringen, laden Sie bitte einfach die neueste Version von Python herunter, bevor Sie mit dem Buch beginnen.[2]

2 [Anmerkung des Übersetzers: Zum Zeitpunkt der Übersetzung ist die aktuelle Version 3.8, und alle Beispiele funktionieren damit tadellos.]

Dieses Buch verwendet ausschließlich die Python-Standardbibliothek (mit einer kleinen Ausnahme in Kapitel 2, wo das Modul `typing_extensions` installiert wird), so dass sämtlicher Code in diesem Buch auf jeder Plattform laufen sollte, auf der Python unterstützt wird (macOS, Windows, GNU/Linux und so weiter). Der Code in diesem Buch wurde ausschließlich mit CPython getestet (dem von *python.org* verfügbaren Haupt-Python-Interpreter), wird aber wahrscheinlich größtenteils auch in einer Python-3.7-kompatiblen Version eines anderen Python-Interpreters laufen.

Dieses Buch erklärt nicht, wie Sie Python-Tools wie Editoren, IDEs, Debugger oder das Python-REPL verwenden. Der Quellcode des Buches ist online im GitHub-Repository verfügbar: *https://github.com/davecom/ClassicComputerScienceProblemsInPython/*.[3] Der Quellcode ist in Ordnern nach Kapiteln organisiert. Wenn Sie jedes Kapitel lesen, finden Sie den Namen einer Quelldatei in der Überschrift jedes Codelistings. Diese Quelldatei finden Sie in ihrem jeweiligen Ordner im Repository. Sie sollten das Programm ausführen können, indem Sie `python3 filename.py` oder `python filename.py` eingeben, je nach Konfiguration Ihres Computers, was den Namen des Python-3-Interpreters angeht.

Jedes Codelisting in diesem Buch macht Gebrauch von Python-Type-Hints, auch *Typ-Annotationen* genannt. Diese Annotationen sind ein relativ neues Feature für die Sprache Python, und sie mögen auf Python-Programmierer, die sie noch nie zuvor gesehen haben, einschüchternd wirken. Sie werden aus drei Gründen verwendet:

1. Sie bieten Klarheit über die Typen von Variablen, Funktionsparametern und Funktionsrückgabewerten.
2. Als Konsequenz aus Grund 1 dokumentieren sie in gewisser Weise automatisch den Code. Anstatt einen Kommentar oder Docstring zu durchsuchen, um den Rückgabewert einer Funktion zu finden, können Sie einfach ihre Signatur betrachten.
3. Sie ermöglichen es, den Code auf Korrektheit der Datentypen zu prüfen. Ein populärer Python-Typchecker ist mypy.

Nicht jeder ist ein Fan von Type-Hints, und sie im gesamten Buch einzusetzen, war in gewisser Weise ein Wagnis. Ich hoffe, dass sie eine Hilfe und keine Hürde darstellen. Es benötigt etwas mehr Zeit, Python mit Type-Hints zu schreiben, bietet aber mehr Klarheit beim späteren Lesen. Ein interessanter Fakt ist, dass Type-Hints keine Auswirkungen auf die Ausführung des Codes im Python-Interpreter haben. Sie können die Type-Hints aus jedem Code in diesem Buch entfernen, und er sollte immer noch laufen. Wenn Sie noch nie zuvor Type-Hints gesehen haben und glauben, dass Sie eine gründlichere Einfüh-

3 [Der Rheinwerk Verlag stellt den Quellcode auf der Webseite zum Buch unter »Materialien zum Buch« zum Download bereit: *https://www.rheinwerk-verlag.de/5143*]

rung benötigen, bevor Sie sich das Buch vornehmen, lesen Sie bitte Anhang C, der einen Crashkurs in Type-Hints liefert.

Keine Grafik, kein UI-Code, nur die Standardbibliothek

Es gibt in diesem Buch keine Beispiele, die grafische Ausgaben erzeugen oder Gebrauch von einer grafischen Benutzeroberfläche (GUI) machen. Warum? Das Ziel ist, die gestellten Aufgaben durch Code zu lösen, der so kompakt und lesbar wie möglich ist. Oft steht Grafik diesem Ziel im Weg oder macht Lösungen deutlich komplexer, als sie zur Veranschaulichung des jeweiligen Verfahrens oder Algorithmus sein müssten.

Außerdem macht die Entscheidung, keinen Gebrauch von einem GUI-Framework zu machen, jeglichen Code in diesem Buch äußerst gut portierbar. Er kann ebenso gut auf einer Embedded-Distribution von Python unter Linux laufen wie auf einem Desktop unter Windows. Außerdem wurde die bewusste Entscheidung getroffen, nur Pakete aus der Python-Standardbibliothek zu verwenden statt externer Bibliotheken, anders als in den meisten Python-Büchern für Fortgeschrittene. Warum? Das Ziel ist es, Problemlösungsverfahren von ihren Grundprinzipien aus zu lehren, nicht »pip install Lösung« anzubieten. Indem Sie jede Aufgabe von Grund auf durcharbeiten müssen, gewinnen Sie hoffentlich ein Verständnis dafür, wie populäre Bibliotheken hinter den Kulissen arbeiten. In jedem Fall macht die ausschließliche Verwendung der Standardbibliothek den Code in diesem Buch portierbarer und einfacher ausführbar.

Das soll nicht heißen, dass grafische Lösungen einen Algorithmus nicht manchmal besser veranschaulichen können als textbasierte Lösungen. Das war einfach nicht der Fokus dieses Buches. Es würde eine zusätzliche Ebene unnötiger Komplexität hinzufügen.

Teil einer Serie

Dies ist das zweite Buch einer Serie namens *Classic Computer Science Problems*, die von Manning verlegt wird. Das erste Buch war *Classic Computer Science Problems in Swift*, erschienen 2018. In jedem Buch der Serie versuchen wir, sprachspezifische Besonderheiten herauszustellen, während wir (ungefähr) dieselben Informatikaufgaben lehren.

Wenn Ihnen dieses Buch gefällt und Sie beschließen, eine andere in dieser Serie behandelte Sprache zu erlernen, finden Sie auf dem Weg von einem Buch zum nächsten möglicherweise einen einfachen Weg, Ihre Kenntnisse in dieser Sprache zu vertiefen. Im Moment umfasst die Serie nur Swift und Python. Ich habe die ersten beiden Bücher selbst geschrieben, weil ich über große Erfahrung mit beiden Sprachen verfüge, aber wir

diskutieren bereits Pläne für weitere Bücher in der Reihe mit Experten in anderen Sprachen, die als Co-Autoren in Frage kommen. Wenn Ihnen dieses Buch gefällt, ermutige ich Sie, danach Ausschau zu halten. Weitere Informationen über die Serie finden Sie unter *https://classicproblems.com/*.