Vorwort

Zunächst einmal bedanke ich mich bei Ihnen, dass Sie sich für dieses Buch entschieden haben. Hierin finden Sie eine Vielzahl an Übungsaufgaben zu den unterschiedlichsten Themengebieten, die kurzweilige Unterhaltung durch Lösen und Implementieren der Aufgaben bieten und Sie so entweder auf Bewerbungsgespräche einstimmen oder aber einfach Ihre Problemlösungsfähigkeiten verbessern.

Übung macht den Meister

Wir alle kennen das Sprichwort: »Übung macht den Meister.« Im Handwerk und in diversen Bereichen des realen Lebens wird viel geübt und der Ernstfall ist eher selten, etwa im Sport, bei Musikern und in anderen Bereichen. Merkwürdigerweise ist dies bei uns Softwareentwicklern oftmals deutlich anders. Wir entwickeln eigentlich fast die gesamte Zeit und widmen uns dem Üben und Lernen bzw. Einstudieren eher selten, teilweise gar nicht. Wie kommt das?

Vermutlich liegt das neben dem in der Regel vorherrschenden Zeitdruck auch daran, dass nicht so viel geeignetes Übungsmaterial zur Verfügung steht – es gibt zwar Lehrbücher zu Algorithmen sowie Bücher zu Coding, aber meistens sind diese entweder zu theoretisch oder zu Sourcecode-lastig und beinhalten zu wenig Erklärungen der Lösungswege. Das will dieses Buch ändern.

Wieso dieses Buch?

Wie kam ich dazu, dieses Buchprojekt in Angriff zu nehmen? Das hat mehrere Gründe. Zum einen wurde ich immer wieder per Mail oder persönlich von Teilnehmern meiner Workshops gefragt, ob es nicht ein Übungsbuch als Ergänzung zu meinem Buch »Der Weg zum Java-Profi« [4] geben würde. Dadurch kam die erste Idee auf.

Doch wirklich ausgelöst hat das Ganze dann, dass ein Google-Recruiter mit einer Jobanfrage recht überraschend auf mich zukam. Als Vorbereitung für die dann bevorstehenden Jobinterviews und zur Auffrischung meiner Kenntnisse machte ich mich auf die Suche nach geeigneter Lektüre und entwickelte selbst schon ein paar Übungsaufgaben. Dabei entdeckte ich das großartige, aber teilweise auch recht anspruchsvolle Buch »Cracking the coding interview« von Gayle Laakmann McDowell [6], das mich weiter inspirierte. Als Folge davon machte ich mich zunächst an ein auf Java ausgerichtetes Buchprojekt namens »Java Challenge«. Im Laufe der Zeit kam die Idee auf, etwas

Entsprechendes auch für Python umzusetzen. Somit basiert diese Python-Ausgabe auf der Java-Version, allerdings wurde das gesamte Buch überarbeitet und pythonifiziert. Dazu habe ich an einigen Stellen Dinge ergänzt, leicht abgewandelt oder teilweise entfernt. Insbesondere zeige ich, da wo es sinnvoll ist, wie man mit Python-Besonderheiten wie List Comprehensions u. Ä. Lösungen prägnanter gestalten kann.

An wen richtet sich dieses Buch?

Dieses Buch ist kein Buch für Programmierneulinge, sondern richtet sich an Leser, die bereits etwas Python-Know-how besitzen und dieses mithilfe von Übungen vertiefen wollen. Anhand kleiner Programmieraufgaben erweitern Sie auf unterhaltsame Weise Ihr Wissen rund um Python, Algorithmen und gutes Design.

Dieses Buch richtet sich im Speziellen an folgende Zielgruppen:

1. Schüler und Studierende – Zunächst sind dies Schüler mit Interesse an Informatik sowie Informatikstudierende, die Python als Sprache schon ganz passabel beherrschen und nun ihr Wissen anhand von Übungen vertiefen wollen.
2. Lehrer und Dozierende – Selbstverständlich können auch Lehrer und Dozierende von diesem Buch und seiner Vielzahl unterschiedlich schwieriger Aufgaben profitieren, entweder als Anregung für den eigenen Unterricht oder als Vorlage für Übungen oder Prüfungen.
3. Hobbyprogrammierer und Berufseinsteiger – Außerdem richtet sich das Buch an engagierte Hobbyprogrammierer, aber auch Berufseinsteiger, die gerne mit Python programmieren und sich weiterentwickeln wollen. Das Lösen der Aufgaben hilft darüber hinaus, für potenzielle Fragen in Jobinterviews gut vorbereitet zu sein.
4. Erfahrene Softwareentwickler und -architekten – Schließlich ist das Buch für erfahrene Softwareentwickler und -architekten bestimmt, die ihr Wissen ergänzen oder auffrischen wollen, um ihre Junior-Kollegen besser unterstützen zu können, und dafür ein paar Anregungen und frische Ideen suchen. Zudem lassen sich diverse Aufgaben auch in Jobinterviews verwenden, mit dem Komfort, die Musterlösungen direkt zum Vergleich parat zu haben. Aber auch für die alten Hasen sollte es zur Lösungsfindung und zu Algorithmen und Datenstrukturen das eine oder andere Aha-Erlebnis geben.

Generell verwende ich die maskuline Form, um den Text leichter lesbar zu halten. Natürlich beziehe ich damit alle Leserinnen mit ein und freue mich über diese ganz besonders.

Was vermittelt dieses Buch?

Dieses Buch enthält einen bunten Mix an Übungsaufgaben zu verschiedenen Themengebieten. Mitunter gibt es auch einige Knobelaufgaben, die zwar nicht direkt für die Praxis wichtig sind, aber indirekt doch, weil Sie Ihre Fähigkeiten zur Kreativität und zur Lösungsfindung verbessern.

Neben Übungsaufgaben und dokumentierten Lösungen war es mir wichtig, dass jeder im Buch behandelte Themenbereich mit einer kurzen Einführung startet, damit auch diejenigen Leser abgeholt werden, die in einigen Gebieten vielleicht noch nicht so viel Know-how aufgebaut haben. Damit können Sie sich dann an die Aufgaben bis etwa zum mittleren Schwierigkeitsgrad wagen. In jedem Themengebiet finden sich immer auch einige leichtere Aufgaben zum Einstieg. Mit etwas Übung sollten Sie sich dann an etwas schwierigere Probleme wagen. Mitunter gibt es herausfordernde Knacknüsse, an denen sich besser Experten versuchen oder solche, die es werden wollen.

Tipps und Hinweise aus der Praxis

Dieses Buch ist mit diversen Praxistipps gespickt. In diesen werden interessante Hintergrundinformationen präsentiert oder es wird auf Fallstricke hingewiesen.

Tipp:

In derart formatierten Kästen finden sich im späteren Verlauf des Buchs immer wieder einige wissenswerte Tipps und ergänzende Hinweise zum eigentlichen Text.

Schwierigkeitsgrad im Überblick

Für ein ausgewogenes, ansprechendes Übungsbuch bedarf es selbstverständlich einer Vielzahl an Aufgaben verschiedener Schwierigkeitsstufen, die Ihnen als Leser die Möglichkeit bieten, sich schrittweise zu steigern und Ihre Kenntnisse auszubauen. Dabei setze ich zwar ein gutes Python-Grundwissen voraus, allerdings erfordern die Lösungen niemals ganz tiefes Wissen über ein Themengebiet oder ganz besondere Sprachfeatures.

Damit der Schwierigkeitsgrad einfach und direkt ersichtlich ist, habe ich die von anderen Bereichen bekannte Sternekategorisierung genutzt, deren Bedeutung in diesem Kontext in nachfolgender Tabelle etwas genauer erläutert wird.

Sterne (Bedeutung)	Einschätzung	Zeitaufwand
★☆☆☆☆ (sehr leicht)	Die Aufgaben sollten ohne große Vorkenntnisse mit einfachem Python-Wissen in wenigen Minuten lösbar sein.	< 15 min
★★☆☆☆ (leicht)	Die Aufgaben erfordern ein wenig Nachdenken, sind aber dann direkt zu lösen.	< 30 min
★★★☆☆ (mittel)	Die Aufgaben sind mit etwas Nachdenken, ein wenig Strategie und manchmal durch die Betrachtung verschiedener Rahmenbedingungen gut schaffbar.	~ 30 – 45 min
★★★★☆ (schwierig)	Erprobte Problemlösungsstrategien, gutes Wissen zu Datenstrukturen und fundierte Python-Kenntnisse sind zur Lösung nötig.	~ 45 – 90 min
★★★★★ (sehr schwierig)	Die Aufgaben sind wirklich knifflig und schwierig zu lösen. Das sind erst dann Kandidaten, nachdem die anderen Aufgaben Ihnen keine größeren Schwierigkeiten mehr bereiten.	> 60 min

Dies sind jeweils nur Einschätzungen von meiner Seite und eher grobe Einordnungen. Bedenken Sie bitte, dass die von jedem Einzelnen wahrgenommene Schwierigkeit auch sehr von seinem Background und Wissensstand abhängt. Ich habe schon erlebt, dass sich Kollegen mit Aufgaben schwergetan haben, die ich als recht einfach empfand. Aber auch das Gegenteil kenne ich: Während andere eine Aufgabe anscheinend spielend lösen, ist man selbst am Verzweifeln, weil der Groschen einfach nicht fällt. Manchmal hilft eine Kaffeepause oder ein kleiner Spaziergang. Lassen Sie sich auf keinen Fall demotivieren – jeder hat irgendwann mit irgendeiner Art von Aufgabe zu kämpfen.

Hinweis: Mögliche Alternativen zu den Musterlösungen

Beachten Sie bitte, dass es für Problemstellungen nahezu immer einige Varianten gibt, die für Sie vielleicht sogar eingängiger sind. Deswegen werde ich ab und an als Denkanstoß interessante Alternativen zur (Muster-)Lösung präsentieren.

Aufbau dieses Buchs

Nachdem Sie eine grobe Vorstellung über den Inhalt dieses Buchs haben, möchte ich die Themen der einzelnen Kapitel kurz vorstellen.

Wie bereits angedeutet, sind die Übungsaufgaben thematisch gruppiert. Dabei bilden die fünf Kapitel nach der Einleitung die Grundlage und die darauffolgenden drei Kapitel behandeln fortgeschrittenere Themengebiete.

Kapitel 1 – Einleitung Dieses Kapitel beschreibt den Aufbau der folgenden Kapitel mit den Abschnitten Einführung, Aufgaben und Lösungen. Zudem wird ein Grundgerüst für die oftmals zur Prüfung der Lösungen genutzten Unit Tests vorgestellt. Abschließend gebe ich Hinweise zum Ausprobieren der Beispiele und Lösungen.

Kapitel 2 – Mathematische Aufgaben Das zweite Kapitel widmet sich mathematischen Operationen sowie Aufgaben zu Primzahlen und dem römischen Zahlensystem. Darüber hinaus präsentiere ich ein paar Ideen zu Zahlenspielereien.

Kapitel 3 – Rekursion Rekursion ist ein wichtiger Basisbaustein bei der Formulierung von Algorithmen. Dieses Kapitel gibt einen kurzen Einstieg und die diversen Übungsaufgaben sollten dabei helfen, Rekursion zu verstehen.

Kapitel 4 – Strings Strings sind bekanntermaßen Zeichenketten, die eine Vielzahl an Funktionen bieten. Ein solides Verständnis ist elementar wichtig, da nahezu kein Programm ohne Strings auskommt. Deswegen werden wir in diesem Kapitel die Verarbeitung von Zeichenketten anhand verschiedener Übungsaufgaben kennenlernen.

Kapitel 5 – Basisdatenstrukturen: Listen, Sets und Dictionaries Python bietet von Haus aus Listen, Mengen (Sets) und Schlüssel-Wert-Abbildungen (Dictionaries). Für den Programmieralltag ist ein sicherer Einsatz aller drei Container von großem Vorteil, was durch die Übungsaufgaben trainiert wird.

Kapitel 6 – Arrays Arrays bilden in vielen Programmiersprachen Grundbausteine. In Python sind Listen gebräuchlicher. Bezüglich Performance und Speicherverbrauch besitzen Arrays aber deutliche Vorteile. Grund genug, sich das Ganze in diesem Kapitel genauer anzuschauen.

Kapitel 7 – Rekursion Advanced Kapitel 3 hat das Thema Rekursion einleitend behandelt. Dieses Kapitel thematisiert fortgeschrittenere Aspekte. Wir starten mit der Optimierungstechnik namens Memoization. Im Anschluss schauen wir uns Backtracking als eine Problemlösungsstrategie an, die auf Versuch und Irrtum beruht und mögliche Lösungswege durchprobiert. Damit lassen sich diverse Algorithmen ziemlich verständlich und elegant formulieren.