O'REILLY®

Programmieren lernen

von Kopf bis Fuß

Ihr Einstieg in die Programmierung mit Python

Umgehen Sie verbreitete Fallstricke und Fehler

Starten Sie hier Ihre Programmierkarriere

Vermeiden Sie peinliche Syntaxfehler

Trainieren Sie Ihr Gehirn mit 130 Rätseln und Übungen

Erfahren Sie, warum alles, was Ihre Freunde über Informatik wissen, vermutlich falsch ist

Eric Freeman

Deutsche Übersetzung von Jørgen W. Lang

Programmieren lernen
von Kopf bis Fuß

Eric Freeman

Deutsche Übersetzung von
Jørgen W. Lang

O'REILLY®

Beijing • Boston • Farnham • Sebastopol • Tokyo

Eric Freeman

Lektorat: Alexandra Follenius
Übersetzung: Jørgen W. Lang
Korrektorat: Sibylle Feldmann, www.richtiger-text.de
Herstellung: Stefanie Weidner
Umschlaggestaltung: Randy Comer, Michael Oréal, www.oreal.de
Satz: Ulrich Borstelmann, www.borstelmann.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN:
Print 978-3-96009-076-2
PDF 978-3-96010-186-4
ePub 978-3-96010-187-1
mobi 978-3-96010-188-8

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«.
O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

1. Auflage 2018

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Vor KISS hatte ich null Erfahrung darin,
in einer Rockband zu spielen, die Make-up trägt.
— Gene Simmons

Eric wurde von Kathy Sierra, der Mitschöpferin der *Von Kopf bis Fuß*-Reihe, als eine der seltenen Persönlichkeiten beschrieben, die »die Sprache, die Umgangsformen und das Selbstverständnis von so unterschiedlichen Menschen wie Hackern, Managern, Ingenieuren und Wissenschaftlern gleichermaßen verstehen«. Und genau das beschreibt auch seinen Hintergrund ziemlich gut. Eric ist von Haus aus Informatiker. Er hat während seiner Promotion an der Yale University bei der IT-Koryphäe David Gelernter studiert. Beruflich hat Eric viel Zeit in der Führungsetage eines Medienunternehmens verbracht – er war CTO bei Disney.com und der Walt Disney Company. Eric war außerdem bei O'Reilly Media, der NASA und in verschiedenen Start-ups tätig. Seine IP ist lizenziert und auf jedem Mac und PC in Gebrauch. Während der letzten 15 Jahre war Eric einer der meistverkauften technischen Autoren zu Themen wie Web- und anspruchsvoller Softwareentwicklung.

Aktuell ist Eric Leiter von WickedlySmart, LLC, und lebt mit seiner Frau und seiner kleinen Tochter in Austin, Texas.

Schreiben Sie Eric unter *eric@wickedlysmart.com* oder besuchen Sie seine Site unter *http://wickedlysmart.com*.

Inhalt (im Überblick)

Inhalt (jetzt ausführlich)

Einführung

Ihr Gehirn und das Programmieren. *Sie* versuchen, etwas zu *lernen*, und Ihr *Hirn* tut sein Bestes, damit das Gelernte *nicht hängen bleibt*. Es denkt nämlich: »Wir sollten lieber ordentlich Platz für wichtigere Dinge lassen, z. B. für das Wissen darüber, welche Tiere einem gefährlich werden könnten, oder dass es eine ganz schlechte Idee ist, nackt Snowboard zu fahren.« Tja, wie schaffen wir es nun, Ihr Gehirn davon zu überzeugen, dass Ihr Leben davon abhängt, dass Sie programmieren können?

Rechnerisch denken

1 Erste Schritte

Durch die Fähigkeit, rechnerisch zu denken, haben Sie die Kontrolle. Jeder weiß, dass die Welt um einen herum immer vernetzter, konfigurierbarer, programmierbarer und eben **rechnerischer** wird. Entweder Sie bleiben passiv, oder Sie lernen zu programmieren. Sobald Sie es können, sind Sie Regisseur und Schöpfer – Sie sagen dem Computer, was er für Sie tun soll. Sobald Sie es können, bestimmen Sie Ihr Schicksal selbst (oder können zumindest Ihre Rasensprinkleranlage über das Internet steuern). Aber wie lernt man zu programmieren? Zuerst müssen Sie lernen, **rechnerisch** zu denken. Dann schnappen Sie sich eine **Programmiersprache**, um mit Ihrem Computer, Mobilgerät oder eigentlich allem mit einer CPU sprechen zu können. Und was haben Sie davon? Mehr Zeit, mehr Fähigkeiten und mehr kreative Möglichkeiten, die Dinge zu tun, die Sie wirklich tun wollen. Dann mal los …

Einfache Werte, Variablen und Typen

2 Kennen Sie Ihren Wert

Computer können nur zwei Dinge richtig gut: Werte speichern und Operationen daran ausführen. Es scheint, als täten Computer sehr viel mehr, wenn Sie Texte schicken, online einkaufen, Photoshop benutzen oder Ihr Telefon als Navi verwenden. Dennoch kann alles, was Computer tun, in **einfache Operationen** aufgeteilt werden, die an **einfachen Werten** ausgeführt werden. Deshalb besteht ein wichtiger Teil des **rechnerischen Denkens** darin, zu lernen, diese Operationen und Werte benutzen, um etwas Anspruchsvolleres, Komplexeres und Sinnvolleres zu erschaffen. Zuerst sehen wir uns aber an, um welche Werte es geht, welche Operationen Sie daran ausführen können und welche Rolle **Variablen** dabei spielen.

Boolesche Werte, Entscheidungen und Schleifen

3 Entscheidender Code

Finden Sie unsere Programme auch ein wenig langweilig?

Bis jetzt enthält unser Code einfach ein paar Anweisungen, die der Interpreter **von oben nach unten** auswertet. Keine unerwarteten Handlungswendungen, keine Überraschungen, kein unabhängiges Denken. Damit der Code interessanter wird, muss er **Entscheidungen treffen können** – um **sein eigenes Schicksal zu bestimmen** und um Dinge **mehr als einmal** ausführen zu können. Genau das lernen wir in diesem Kapitel. Danach erfahren wir mehr über ein mysteriöses Spiel namens Shoushiling und treffen einen Typ namens Boole. Außerdem sehen wir, warum ein Datentyp mit nur zwei Werten durchaus unsere Aufmerksamkeit verdient. Wir lernen sogar, wie wir die gefürchtete **Endlosschleife** bändigen können. Und los!

Listen und Iterationen

4 Etwas mehr Struktur

Zahlen, Strings und boolesche Werte sind nicht die einzigen Datentypen in Python. Bisher haben Sie beim Schreiben Ihres Codes nur sogenannte **primitive Datentypen** verwendet – Fließkomma- und Integerwerte, Strings und natürlich boolesche Werte mit Werten wie `3.14`, `42`, `„Hey, ich bin dran!"` und `True`. Und damit lässt sich schon eine Menge anfangen. Aber irgendwann müssen Sie Code schreiben, der mit großen Datenmengen umgehen muss, z. B. dem Inhalt eines Warenkorbs, den Namen aller nennenswerten Promis oder mit einem kompletten Produktkatalog. Dafür brauchen wir etwas mehr *Wumms*. In diesem Kapitel lernen wir einen neuen Datentyp kennen: die **Liste**, die eine ganze Sammlung von Werten enthalten kann. Mit Listen können Sie Daten **strukturieren**. Anstatt eine Fantastillion einzelner Variablen im Code zu verwenden, können Sie diese Variablen auch als Einheit behandeln und mithilfe der ***for***-Schleife (die wir aus dem vorigen Kapitel kennen) über die einzelnen Elemente **iterieren**. Wenn Sie dieses Kapitel abgeschlossen haben, können Sie mit Daten umgehen, die wachsen und sich ausdehnen.

Funktionen und Abstraktion

5 Funktional werden

Sie wissen bereits eine Menge. Allein mit Variablen, Datentypen, Bedingungen und Iterationen könnten Sie schon alle Programme schreiben, die Sie jemals brauchen werden. Ein Informatiker würde vermutlich sagen, dass Sie sich damit jedes nur vorstellbare Programm schreiben können. Trotzdem sind wir noch lange nicht am Ende. Im nächsten Schritt des rechnerischen Denkens lernen Sie, Ihren Code zu **abstrahieren**. Auch wenn es kompliziert klingt, wird es Ihr Leben als Programmierer sogar einfacher machen. Durch die Abstraktionen erreichen Sie einen höheren Wirkungsgrad und können komplexere und mächtigere Programme leichter erstellen. Sie können Ihren Code in praktische kleine und vor allem wiederverwendbare Einheiten verpacken. Sie brauchen sich nicht mehr um jedes kleine Detail Ihres Code zu kümmern und können stattdessen anfangen, auf einer höheren Ebene zu denken.

Sortierung und verschachtelte Iteration

4 Teil 2 Ordnung in die Daten bringen

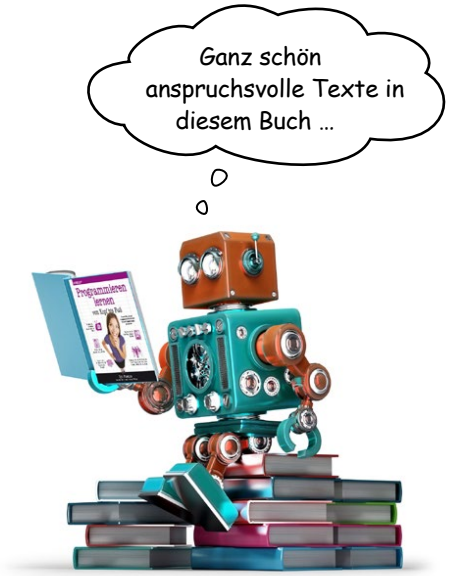
Manchmal reicht es einfach nicht aus, die Daten nur zu ordnen.

Angenommen, Sie haben eine Liste Ihrer Highscores von 80er-Jahre-Arcade-Spielen. Diese sollen alphabetisch nach Spielen sortiert werden. Und dann gibt es die Liste, in der festgehalten ist, wie oft Ihre Mitarbeiter Sie schon betrogen haben. Sie wollen wirklich gerne wissen, wer auf dieser Liste ganz oben steht. Dafür müssen Sie allerdings lernen, Daten zu sortieren. Zu dem Zweck betrachten wir ein paar Algorithmen, die es etwas mehr in sich haben als die bisher gesehenen. Wir werden uns außerdem mit verschachtelten Schleifen beschäftigen und ein wenig über die Effizienz des von Ihnen geschriebenen Codes nachdenken. Dann wollen wir Ihr rechnerisches Denken mal auf das nächste Level heben!

Text, Strings und Heuristiken

6 Die Einzelteile verbinden

Sie haben schon eine Menge Superkräfte erworben. Jetzt geht es darum, sie auch zu benutzen. In diesem Kapitel werden wir alle bisher gelernten Dinge miteinander koppeln, um damit **ziemlich coolen Code** zu schreiben. Außerdem erweitern wir Ihr Wissen und Ihre Programmierfähigkeiten. In diesem Kapitel lernen Sie, Code zu schreiben, der **sich etwas Text schnappt**, ihn in seine Einzelteile zerlegt und dann eine **Datenanalyse** daran durchführt. Außerdem erfahren Sie, was eine **Heuristik** ist. Schnallen Sie sich an, denn dies ein tiefergelegtes, hart getuntes Vollgas-Coding-Kapitel!



Module, Methoden, Klassen und Objekte

7 Die Module spielen verrückt ...

Ihr Code wird umfangreicher und komplexer. Also brauchen Sie bessere Möglichkeiten, ihn zu abstrahieren, zu modularisieren und zu organisieren. Durch Funktionen können mehrere Codezeilen zusammengefasst und wiederverwendet werden. Außerdem wissen Sie inzwischen, dass Sammlungen von Funktionen und Variablen in Modulen gespeichert und dadurch leichter weitergegeben und wiederverwendet werden können. In diesem Kapitel befassen wir uns noch einmal mit Modulen und lernen, sie effizienter zu nutzen (z. B. um den Code direkt mit anderen teilen zu können). Dann kommen wir zur ultimativen Form der Codewiederverwendung: *Objekte*. Sie werden merken, dass Sie quasi von Python-Objekten umgeben sind, die nur darauf warten, von Ihnen benutzt zu werden.

Rekursion und Dictionaries

8 Mehr als Iteration und Indizes

Es ist Zeit, Ihr rechnerisches Denken auf die nächste Ebene zu heben.

Und genau das werden wir in diesem Kapitel tun. Bisher haben wir einen eher iterativen Programmierstil verwendet. Wir haben Datenstrukturen (z. B. Listen, Strings und Zahlenbereiche) erstellt und Code geschrieben, der darüber iteriert, um zu einer Lösung zu kommen. In diesem Kapitel vermitteln wir Ihnen eine neue Weltsicht: zuerst in Bezug auf die Art, Dinge zu berechnen, und dann in Bezug auf Datenstrukturen. Rechnerisch befassen wir uns mit einer Programmiertechnik, die *rekursiven* Code verwendet – Code, der sich selbst aufruft. Wir lernen außerdem eine neue Datenstruktur kennen, die funktioniert wie ein Wörterbuch, also eher wie *assoziative Beziehungen* und weniger wie eine Liste. Danach verbinden wir die beiden neuen Konzepte und verursachen eine Menge Schwierigkeiten. Seien Sie gewarnt: Diese Themen brauchen etwas Zeit, um sich in Ihrem Gehirn festzusetzen. Aber der Aufwand lohnt sich auf jeden Fall.

Dateien speichern und auslesen

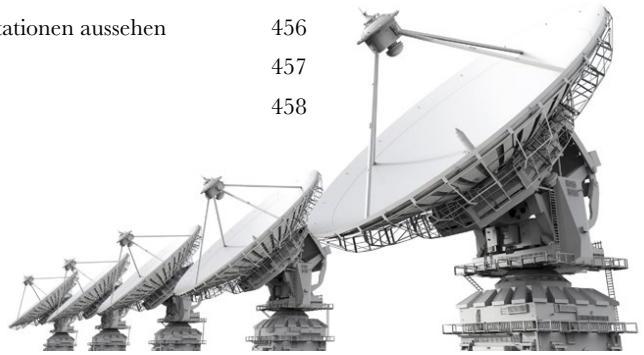
9 Persistenz

Sie wissen, dass Sie Werte in Variablen speichern können. Aber sobald Ihr Programm endet, sind sie für immer verloren. Genau da kommt die *persistente Speicherung* ins Spiel – eine Möglichkeit, Werte und Daten über die Laufzeit des Programms hinaus zu speichern. Die meisten Geräte, auf denen Python läuft, besitzen die eine oder andere Form von persistenter Speicherung, z. B. Festplatten, SD-Karten oder Cloud-Speicher. In diesem Kapitel lernen Sie, Code zu schreiben, der Daten in Dateien speichert und wieder ausliest. Und wann hilft uns das? Zum Beispiel jedes Mal, wenn Sie Benutzereinstellungen oder die Ergebnisse einer Analyse für den Chef speichern wollen, ein Bild zur Bearbeitung einlesen, die E-Mails der letzten zehn Jahre durchsuchen oder bestimmte Daten in einer Tabellenkalkulation berechnen wollen. Wir könnten die Liste noch endlos weiterführen, aber wir fangen wohl besser mit dem Kapitel an.

Web-APIs benutzen

10 Sie sollten wirklich öfter mal ausgehen

Sie haben großartigen Code geschrieben, aber Sie sollten wirklich öfter ausgehen. Im Web wartet die ganze Welt der **Daten** auf Sie: Brauchen Sie Informationen zum Wetter? Oder wollen Sie auf eine riesige Datenbank mit Kochrezepten zugreifen? Oder stehen Sie mehr auf Sportergebnisse? Vielleicht eine Musikdatenbank mit Künstlern, Alben und Titeln? Mit **Web-APIs** brauchen Sie nur zuzugreifen. Um APIs zu benutzen, benötigen Sie lediglich ein paar Informationen über ihre Funktionsweise, die Aussprache des lokalen Dialekts und die Verwendung einiger neuer Python-Module: `requests` und `json`. In diesem Kapitel werden wir Web-APIs erforschen und Ihre Python-Fähigkeiten auf eine neue Ebene heben. Tatsächlich nehmen wir Sie mit in den Weltraum – und wieder zurück.



Widgets, Events und emergentes Verhalten

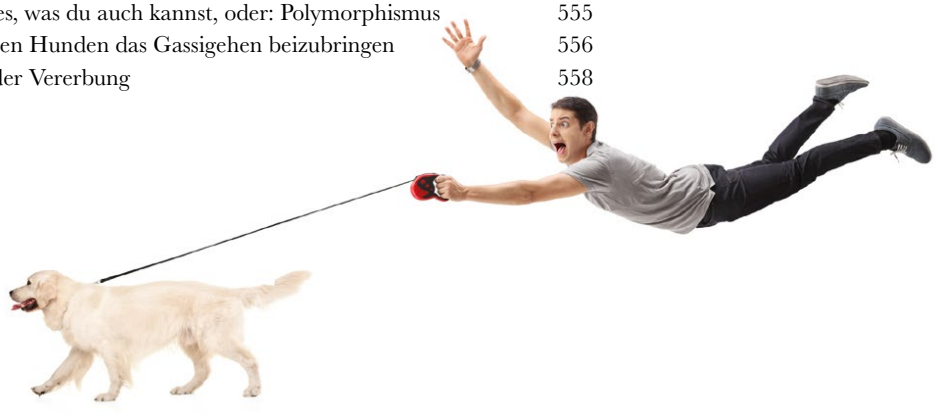
11 Interaktivität

Sie haben inzwischen schon einige grafische Applikationen geschrieben. Eine echte grafische Benutzerschnittstelle fehlt aber noch. Das heißt, Sie haben noch nichts programmiert, mit dessen Hilfe der Benutzer mit einer grafischen Benutzeroberfläche (auch als GUI bezeichnet) interagieren kann. Hierfür brauchen Sie eine neue Denkweise dazu, wie ein Programm ausgeführt wird, sodass es direkt auf die Benutzereingaben **reagiert**. Hat der Benutzer gerade einen Button angeklickt? Dann sollte Ihr Code auch wissen, wie er darauf reagieren muss und was als Nächstes zu tun ist. Die Programmierung von Benutzeroberflächen unterscheidet sich stark von der bisher verwendeten prozeduralen Vorgehensweise. In diesem Kapitel werden Sie Ihr erstes echtes GUI programmieren. Das wird übrigens weder ein einfacher To-do-Listenmanager noch ein Höhe/Breite-Rechner. Wir machen etwas viel Interessanteres. Wir schreiben einen Simulator für künstliches Leben mit emergentem (sich weiterentwickelndem) Verhalten. Und was heißt das? Finden Sie es heraus.

Objektorientierte Programmierung

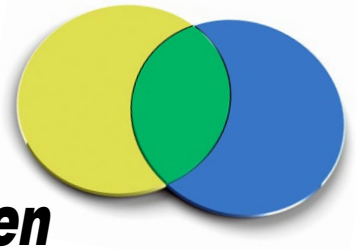
12 Ausflug nach Objekthausen

In diesem Buch haben Sie meist Funktionen benutzt, um Code zu abstrahieren. Bisher haben Sie für die Programmierung einen **prozeduralen Ansatz** verwendet: einfache Anweisungen, Bedingungen und `for`/`while`-Schleifen mit Funktionen. Nichts davon ist wirklich **objektorientiert**. Eigentlich ist es *überhaupt nicht* objektorientiert! Im Code haben wir uns Objekte und ihre Verwendung bereits angesehen. Aber eigene Objekte haben Sie bisher nicht erstellt, und mit dem objektorientierten Design Ihres Code haben Sie sich auch noch nicht beschäftigt. Daher ist es jetzt Zeit, diese langweilige prozedurale Stadt hinter sich zu lassen. In diesem Kapitel finden Sie heraus, warum die Verwendung von Objekten Ihr Leben deutlich erleichtern kann – zumindest bei der **Programmierung** (für Unterstützung in anderen Lebensbereichen reicht dieses Buch einfach nicht aus). Nur eine Warnung: Sobald Sie Objekte entdeckt haben, wollen Sie nicht mehr zurück. Schicken Sie uns eine Postkarte, wenn Sie angekommen sind.

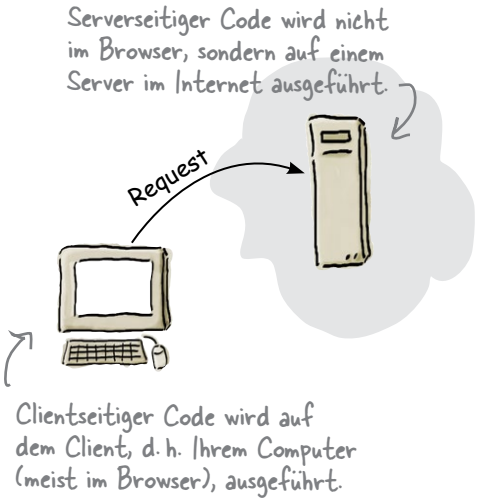


Anhang: Was übrig bleibt

Die Top Ten der Themen, die wir nicht behandelt haben



Wir haben eine Menge Themen behandelt, und Sie haben das Buch fast durchgelesen. Wir werden Sie vermissen. Aber ganz ohne etwas mehr Vorbereitung wollen wir Sie auch nicht in die Welt entlassen. Natürlich passt nicht alles, was Sie wissen müssen, in diesen relativ kurzen Anhang. Ursprünglich hatten wir sogar sämtliche Informationen über die Programmierung mit Python (die in den anderen Kapiteln bisher nicht zur Sprache kamen) für Sie hier hineingepackt, indem wir die Schriftgröße auf 0,00004 Punkt reduziert hatten. Hat alles wunderbar gepasst. Nur konnte es leider niemand mehr lesen. Also haben wir das meiste wieder rausgeschmissen und nur die besten Dinge für diesen Top-Ten-Anhang behalten. Dies ist *wirklich* das Ende des Buchs. Natürlich bis auf den Index (absolut lesenswert!).



Wie man dieses Buch benutzt

Einführung

In diesem Abschnitt beantworten wir die brennende Frage: >>Warum steht SO ETWAS in einem Buch, in dem man programmieren lernen soll?<<

Für wen ist dieses Buch?

Wenn Sie alle folgenden Fragen mit »Ja« beantworten können …

1. Wollen Sie **lernen, verstehen** und **behalten**, wie man programmiert?

2. Bevorzugen Sie eine **anregende Unterhaltung beim Abendessen** oder **trockene, langweilige Vorlesungen?**

… dann ist dieses Buch etwas für Sie.

[Hinweis aus der Marketingabteilung: Dieses Buch ist etwas für jeden mit einer Kreditkarte.]

Dies ist KEIN Nachschlagewerk. *Programmieren lernen von Kopf bis Fuß* ist ein Buch, in dem Sie programmieren *lernen*. Es ist kein Lexikon mit Programmierfakten (dafür gibt es Google, oder?).

Wer sollte eher die Finger von diesem Buch lassen?

Wenn Sie eine der folgenden Fragen mit »Ja« beantworten müssen …

1. **Sind Computer völliges Neuland für Sie?**

   Wenn Sie sich auf Ihrem Computer nicht auskennen, nicht wissen, wie man mit Dateien und Ordnern umgeht, Programme installiert oder mit einem Texteditor arbeitet, sollten Sie diese Dinge zuerst lernen.

2. Sind Sie ein Oberklasse-Programmierer, der nach einem **Referenzbuch** sucht?

3. Haben Sie **Angst, etwas Neues auszuprobieren?** Ist Ihnen eine Wurzelbehandlung lieber, als Streifen mit Karos zu tragen? Glauben Sie, dass ein technisches Buch nicht seriös sein kann, wenn man beim Lernen Spaß hat?

… dann ist dieses Buch nichts für Sie.

Wir wissen, was Sie gerade denken.

»Kann *das* wirklich ein seriöses Buch sein?«

»Was sollen all die Abbildungen?«

»Kann ich das auf diese Weise wirklich *lernen*?«

Und wir wissen, was Ihr *Gehirn* gerade denkt.

Ihr Gehirn lechzt nach Neuem. Es ist ständig dabei, Ihre Umgebung abzusuchen, und es *wartet* auf etwas Ungewöhnliches. So ist es nun einmal gebaut, und es hilft Ihnen zu überleben.

Also, was macht Ihr Gehirn mit all den gewöhnlichen, normalen Routinesachen, denen Sie begegnen? Es tut alles in seiner Macht Stehende, damit es dadurch nicht bei seiner *eigentlichen* Arbeit gestört wird: Dinge zu erfassen, die wirklich *wichtig* sind. Es gibt sich nicht damit ab, die langweiligen Sachen zu speichern, sondern lässt diese gar nicht erst durch den »Dies-ist-offensichtlich-nicht-wichtig«-Filter.

Woher *weiß* Ihr Gehirn denn, was wichtig ist? Nehmen Sie an, Sie machen einen Tagesausflug und ein Tiger springt vor Ihnen aus dem Gebüsch: Was passiert dabei in Ihrem Kopf und Ihrem Körper?

Neuronen feuern. Gefühle werden angekurbelt. *Chemische Substanzen durchfluten Sie.*

Und so weiß Ihr Gehirn:

Dies muss wichtig sein! Vergiss es nicht!

Aber nun stellen Sie sich vor, Sie sind zu Hause oder in einer Bibliothek. In einer sicheren, warmen, tigerfreien Zone. Sie lernen. Bereiten sich auf eine Prüfung vor. Oder Sie versuchen, irgendein schwieriges Thema zu lernen, von dem Ihr Chef glaubt, Sie bräuchten dafür eine Woche oder höchstens zehn Tage.

Da ist nur ein Problem: Ihr Gehirn versucht, Ihnen einen großen Gefallen zu tun. Es versucht, dafür zu sorgen, dass diese *offensichtlich* unwichtigen Inhalte nicht knappe Ressourcen verstopfen. Ressourcen, die besser dafür verwendet würden, die wirklich *wichtigen* Dinge zu speichern. Wie Tiger. Wie die Gefahren des Feuers. Wie die Notwendigkeit, schnell das Browserfenster mit dem YouTube-Video zu einer Alien-Entführung zu verbergen, wenn Ihr Chef die Nase ins Büro steckt.

Und es gibt keine einfache Möglichkeit, Ihrem Gehirn zu sagen: »Hey, Gehirn, vielen Dank, aber egal, wie langweilig dieses Buch auch ist und wie klein der Ausschlag auf meiner emotionalen Richterskala gerade ist, ich *will* wirklich, dass du diesen Kram behältst.«

Wir stellen uns unseren Leser als einen aktiv Lernenden vor.

Also, was ist nötig, damit Sie etwas *lernen*? Erst einmal müssen Sie es *aufnehmen* und dann dafür sorgen, dass Sie es nicht wieder *vergessen*. Es geht nicht darum, Fakten in Ihren Kopf zu schieben. Nach den neuesten Forschungsergebnissen der Kognitionswissenschaft, der Neurobiologie und der Lernpsychologie gehört zum *Lernen* viel mehr als nur Text auf einer Seite. Wir wissen, was Ihr Gehirn anmacht.

Einige der Lernprinzipien dieser Buchreihe:

Bilder einsetzen. An Bilder kann man sich viel besser erinnern als an Worte allein und lernt so viel effektiver (bis zu 89 % Verbesserung bei Abrufbarkeits- und Lerntransferstudien). Außerdem werden die Dinge dadurch verständlicher. **Setzen Sie Text in oder neben die Grafiken,** auf die sie sich beziehen, anstatt darunter oder auf eine andere Seite. Die Leser werden auf den Bildinhalt bezogene Probleme dann mit *doppelt* so hoher Wahrscheinlichkeit lösen können.

Verwenden Sie einen gesprächsorientierten Stil mit persönlicher Ansprache. Nach neueren Untersuchungen haben Studenten nach dem Lernen bei Tests bis zu 40 % besser abgeschnitten, wenn der Inhalt den Leser direkt in der ersten Person und im lockeren Stil angesprochen hat statt in einem formalen Ton. Halten Sie keinen Vortrag, sondern erzählen Sie Geschichten. Benutzen Sie eine zwanglose Sprache. Nehmen Sie sich selbst nicht zu ernst. Würden *Sie* einer anregenden Unterhaltung beim Abendessen mehr Aufmerksamkeit schenken oder einem Vortrag?

Ich glaube wirklich, Sie sollten diesen Code in eine Funktion abstrahieren.

Lernen Sie nicht einfach zu programmieren – lernen Sie, rechnerisch zu denken.

Bringen Sie den Lernenden dazu, intensiver nachzudenken. Mit anderen Worten: Falls Sie nicht aktiv Ihre Neuronen strapazieren, passiert in Ihrem Gehirn nicht viel. Ein Leser muss motiviert, begeistert und neugierig sein und angeregt werden, Probleme zu lösen, Schlüsse zu ziehen und sich neues Wissen anzueignen. Und dafür brauchen Sie Herausforderungen, Übungen, zum Nachdenken anregende Fragen und Tätigkeiten, die beide Seiten des Gehirns und mehrere Sinne einbeziehen.

Jetzt, nachdem ich Ihre Aufmerksamkeit habe, sollten Sie bei der Verwendung globaler Variablen wirklich vorsichtiger sein.

Ziehen Sie die Aufmerksamkeit des Lesers auf sich – und behalten Sie sie. Wir alle haben schon Erfahrungen dieser Art gemacht: »Ich will das wirklich lernen, aber ich kann einfach nicht über Seite 1 hinaus wach bleiben.« Ihr Gehirn passt auf, wenn Dinge ungewöhnlich, interessant, merkwürdig, auffällig, unerwartet sind. Ein neues, schwieriges, technisches Thema zu lernen, muss nicht langweilig sein. Wenn es das nicht ist, lernt Ihr Gehirn viel schneller.

Sprechen Sie Gefühle an. Wir wissen, dass Ihre Fähigkeit, sich an etwas zu erinnern, wesentlich von dessen emotionalem Gehalt abhängt. Sie erinnern sich an das, was Sie *bewegt*. Sie erinnern sich, wenn Sie etwas *fühlen*. Nein, wir erzählen keine herzzerreißenden Geschichten über einen Jungen und seinen Hund. Was wir erzählen, ruft Überraschungs-, Neugier-, Spaß- und Was-soll-das?-Emotionen hervor und dieses Hochgefühl, das Sie beim Lösen eines Puzzles empfinden oder wenn Sie etwas lernen, das alle anderen schwierig finden. Oder wenn Sie merken, dass Sie etwas können, was dieser »Ich-bin-ein-besserer-Techniker-als-du«-Typ aus der Technikabteilung *nicht kann*.

Metakognition: Nachdenken übers Denken

Wenn Sie wirklich lernen möchten, und zwar schneller und nachhaltiger, dann schenken Sie Ihrer Aufmerksamkeit Aufmerksamkeit. Denken Sie darüber nach, wie Sie denken. Lernen Sie, wie Sie lernen.

Die meisten von uns haben in ihrer Jugend keine Kurse in Metakognition oder Lerntheorie gehabt. Es wurde von uns *erwartet,* dass wir lernen, aber nur selten wurde uns auch *beigebracht,* wie man lernt.

Wir nehmen aber an, dass Sie wirklich etwas über das Programmieren lernen möchten, wenn Sie dieses Buch in den Händen halten. Und wahrscheinlich möchten Sie nicht viel Zeit aufwenden. Und Sie wollen sich an das *erinnern,* was Sie lesen, und es anwenden können. Und deshalb müssen Sie es *verstehen.* Wenn Sie so viel wie möglich von diesem Buch profitieren wollen oder von irgendeinem anderen Buch oder einer anderen Lernerfahrung, übernehmen Sie Verantwortung für Ihr Gehirn. Ihr Gehirn im Zusammenhang mit diesem Lernstoff.

Der Trick besteht darin, Ihr Gehirn dazu zu bringen, neuen Lernstoff als etwas wirklich Wichtiges anzusehen. Als entscheidend für Ihr Wohlbefinden. So wichtig wie einen Tiger. Andernfalls stecken Sie in einem dauernden Kampf, in dem Ihr Gehirn sein Bestes gibt, um die neuen Inhalte davon abzuhalten, hängen zu bleiben.

Wie bringen Sie also Ihr Gehirn dazu, programmieren für so wichtig zu halten wie einen Tiger?

Da gibt es den langsamen, ermüdenden Weg oder den schnelleren, effektiveren Weg. Der langsame Weg geht über bloße Wiederholung. Natürlich ist Ihnen klar, dass Sie lernen und sich sogar an die langweiligsten Themen erinnern *können,* wenn Sie sich die gleiche Sache immer wieder einhämmern. Wenn Sie nur oft genug wiederholen, sagt Ihr Gehirn: »Er hat zwar nicht das *Gefühl,* dass das wichtig ist, aber er sieht sich dieselbe Sache *immer und immer wieder* an – dann muss sie wohl wichtig sein.«

Der schnellere Weg besteht darin, **alles zu tun, was die Gehirnaktivität erhöht**, vor allem verschiedene Arten von Gehirnaktivität. Eine wichtige Rolle dabei spielen die auf der vorhergehenden Seite erwähnten Dinge – alles Dinge, die nachweislich dabei helfen, dass Ihr Gehirn *für* Sie arbeitet. So hat sich z. B. in Untersuchungen gezeigt: Wenn Wörter *in* den Abbildungen stehen, die sie beschreiben (und nicht irgendwo anders auf der Seite, z. B. in einer Bildunterschrift oder im Text), versucht Ihr Gehirn, herauszufinden, wie die Wörter und das Bild zusammenhängen, und dadurch feuern mehr Neuronen. Und je mehr Neuronen feuern, umso größer ist die Chance, dass Ihr Gehirn mitbekommt: Bei dieser Sache lohnt es sich, aufzupassen, und vielleicht auch, sich daran zu erinnern.

Ein lockerer Sprachstil hilft, denn Menschen tendieren zu höherer Aufmerksamkeit, wenn ihnen bewusst ist, dass sie ein Gespräch führen – man erwartet dann ja von ihnen, dass sie dem Gespräch folgen und sich beteiligen. Das Erstaunliche daran ist: Es ist Ihrem Gehirn ziemlich egal, dass die »Unterhaltung« zwischen Ihnen und einem Buch stattfindet! Wenn der Schreibstil dagegen formal und trocken ist, hat Ihr Gehirn den gleichen Eindruck wie bei einem Vortrag, bei dem in einem Raum passive Zuhörer sitzen. Nicht nötig, wach zu bleiben.

Aber Abbildungen und ein lockerer Sprachstil sind erst der Anfang.

Das haben WIR getan:

Wir haben ***Bilder*** verwendet, weil Ihr Gehirn auf visuelle Eindrücke eingestellt ist, nicht auf Text. Soweit es Ihr Gehirn betrifft, sagt ein Bild *wirklich* mehr als 1.024 Worte. Und dort, wo Text und Abbildungen zusammenwirken, haben wir den Text *in* die Bilder eingebettet, denn Ihr Gehirn arbeitet besser, wenn der Text *innerhalb* der Sache steht, auf die er sich bezieht, und nicht in einer Bildunterschrift oder irgendwo vergraben im Text.

Wir haben ***Redundanz*** eingesetzt, d. h. dasselbe auf *unterschiedliche* Art und mit verschiedenen Medientypen ausgedrückt, damit Sie es über *mehrere Sinne* aufnehmen. Das erhöht die Chance, dass die Inhalte an mehr als nur einer Stelle in Ihrem Gehirn verankert werden.

Wir haben Konzepte und Bilder in ***unerwarteter*** Weise eingesetzt, weil Ihr Gehirn auf Neuigkeiten programmiert ist. Und wir haben Bilder und Ideen mit zumindest *etwas* ***emotionalem*** *Charakter* verwendet, weil Ihr Gehirn darauf eingestellt ist, auf die Biochemie von Gefühlen zu achten. An alles, was ein *Gefühl* in Ihnen auslöst, können Sie sich mit höherer Wahrscheinlichkeit erinnern, selbst wenn dieses Gefühl nicht mehr ist als ein bisschen ***Belustigung, Überraschung oder Interesse.***

Seien Sie der Python-Interpreter

Wir haben einen ***umgangssprachlichen Stil*** mit direkter Anrede benutzt, denn Ihr Gehirn ist von Natur aus aufmerksamer, wenn es Sie in einer Unterhaltung wähnt als wenn es davon ausgeht, dass Sie passiv einer Präsentation zuhören – sogar dann, wenn Sie *lesen.*

Wir haben mehr als 100 ***Aktivitäten*** für Sie vorgesehen, denn Ihr Gehirn lernt und behält von Natur aus besser, wenn Sie Dinge ***tun,*** als wenn Sie nur darüber *lesen.* Und wir haben die Übungen zwar anspruchsvoll, aber doch lösbar gemacht, denn so ist es den meisten Lesern am liebsten.

Punkt für Punkt

Wir haben ***mehrere unterschiedliche Lernstile*** eingesetzt, denn vielleicht bevorzugen *Sie* ein Schritt-für-Schritt-Vorgehen, während jemand anderes erst einmal den groben Zusammenhang verstehen und ein Dritter einfach nur ein Codebeispiel sehen möchte. Aber ganz abgesehen von den jeweiligen Lernvorlieben profitiert *jeder* davon, wenn er die gleichen Inhalte in unterschiedlicher Form präsentiert bekommt.

Wir liefern Inhalte für ***beide Seiten Ihres Gehirns,*** denn je mehr Sie von Ihrem Gehirn einsetzen, umso wahrscheinlicher werden Sie lernen und behalten, und umso länger bleiben Sie konzentriert. Wenn Sie mit einer Seite des Gehirns arbeiten, bedeutet das häufig, dass sich die andere Seite des Gehirns ausruhen kann; so können Sie über einen längeren Zeitraum produktiver lernen.

Und wir haben ***Geschichten*** und Übungen aufgenommen, die ***mehr als einen Blickwinkel repräsentieren,*** denn Ihr Gehirn lernt von Natur aus intensiver, wenn es gezwungen ist, selbst zu analysieren und zu beurteilen.

Sie begleiten uns.

Wir haben ***Herausforderungen*** eingefügt: in Form von Übungen und indem wir ***Fragen*** stellen, auf die es nicht immer eine eindeutige Antwort gibt, denn Ihr Gehirn ist darauf eingestellt, zu lernen und sich zu erinnern, wenn es an etwas *arbeiten* muss. Überlegen Sie: Ihren *Körper* bekommen Sie ja auch nicht in Form, wenn Sie nur die Leute auf dem Sportplatz *beobachten.* Aber wir haben unser Bestes getan, um dafür zu sorgen, dass Sie – wenn Sie schon hart arbeiten – an den *richtigen* Dingen arbeiten. Dass Sie ***nicht einen einzigen Dendriten darauf verschwenden,*** ein schwer verständliches Beispiel zu verarbeiten oder einen schwierigen, mit Fachbegriffen gespickten oder übermäßig gedrängten Text zu analysieren.

Wir haben ***Menschen*** eingesetzt. In Geschichten, Beispielen, Bildern usw. – denn *Sie sind* ein Mensch. Und Ihr Gehirn schenkt *Menschen* mehr Aufmerksamkeit als *Dingen.*

Wir haben einen ***80/20***-Ansatz benutzt. Wir gehen davon aus, dass dies nicht Ihr einziges Buch sein wird, wenn Sie ein genialer Webentwickler werden wollen. Deshalb besprechen wir nicht *alles.* Nur das, was Sie wirklich *brauchen* werden.

Schneiden Sie dies aus und heften Sie es an Ihren Kühlschrank.

Und das können SIE tun, um sich Ihr Gehirn untertan zu machen

So, wir haben unseren Teil der Arbeit geleistet. Der Rest liegt bei Ihnen. Diese Tipps sind ein Anfang; hören Sie auf Ihr Gehirn und finden Sie heraus, was bei Ihnen funktioniert und was nicht. Probieren Sie neue Wege aus.

1 Immer langsam. Je mehr Sie verstehen, umso weniger müssen Sie auswendig lernen.

Lesen Sie nicht nur. Halten Sie inne und denken Sie nach. Wenn das Buch Sie etwas fragt, springen Sie nicht einfach zur Antwort. Stellen Sie sich vor, dass Sie das wirklich jemand *fragt.* Je gründlicher Sie Ihr Gehirn zum Nachdenken zwingen, umso größer ist die Chance, dass Sie lernen und behalten.

2 Bearbeiten Sie die Übungen. Machen Sie selbst Notizen.

Wir haben sie entworfen, aber wenn wir sie auch für Sie lösen würden, wäre das, als ob jemand anderes Ihr Training für Sie absolviert. Und sehen Sie sich die Übungen *nicht einfach nur an.* **Benutzen Sie einen Bleistift.** Es deutet vieles darauf hin, dass körperliche Aktivität *beim* Lernen den Lernerfolg erhöhen kann.

3 Lesen Sie die Abschnitte »Es gibt keine Dummen Fragen«.

Und zwar alle. Das sind keine Zusatzanmerkungen – ***sie gehören zum Kerninhalt!*** Überspringen Sie sie nicht.

4 Lesen Sie dies als Letztes vor dem Schlafengehen. Oder lesen Sie danach zumindest nichts *Anspruchsvolles* mehr.

Ein Teil des Lernprozesses (vor allem die Übertragung in das Langzeitgedächtnis) findet erst statt, *nachdem* Sie das Buch zur Seite gelegt haben. Ihr Gehirn braucht Zeit für sich, um weitere Verarbeitung zu leisten. Wenn Sie in dieser Zeit etwas Neues aufnehmen, geht ein Teil dessen, was Sie gerade gelernt haben, verloren.

5 Trinken Sie Wasser. Viel.

Ihr Gehirn arbeitet am besten in einem schönen Flüssigkeitsbad. Austrocknung (zu der es schon kommen kann, bevor Sie überhaupt Durst verspüren) beeinträchtigt die kognitive Funktion.

6 Reden Sie drüber. Laut.

Sprechen aktiviert einen anderen Teil des Gehirns. Wenn Sie etwas verstehen oder Ihre Chancen verbessern wollen, sich später daran zu erinnern, sagen Sie es laut. Noch besser: Versuchen Sie, es jemand anderem laut zu erklären. Sie lernen dann schneller und haben vielleicht Ideen, auf die Sie beim bloßen Lesen nie gekommen wären.

7 Hören Sie auf Ihr Gehirn.

Achten Sie darauf, Ihr Gehirn nicht zu überladen. Wenn Sie merken, dass Sie etwas nur noch überfliegen oder dass Sie das gerade erst Gelesene vergessen haben, ist es Zeit für eine Pause. Ab einem bestimmten Punkt lernen Sie nicht mehr schneller, indem Sie mehr hineinzustopfen versuchen; das kann sogar den Lernprozess stören.

8 Aber bitte mit *Gefühl!*

Ihr Gehirn muss wissen, dass es *um etwas Wichtiges geht.* Lassen Sie sich in die Geschichten hineinziehen. Erfinden Sie eigene Bildunterschriften für die Fotos. Über einen schlechten Scherz zu stöhnen, ist *immer noch* besser, als gar nichts zu fühlen.

9 *Erschaffen* Sie etwas!

Wenden Sie das hier Gelernte auf ein Design an, an dem Sie gerade arbeiten, oder gestalten Sie ein älteres Projekt damit um. Tun Sie *irgendetwas,* um neben den Übungen in diesem Buch weitere Erfahrungen zu sammeln. Sie brauchen dazu nur einen Bleistift und ein zu lösendes Problem ... ein Problem, das davon profitieren würde, dass Sie programmieren können.

10 Schlafen Sie ausreichend.

Sie müssen Ihr Gehirn mächtig anstrengen, wenn Sie gut programmieren wollen. Schlafen Sie viel, es wird Ihnen helfen!

Lies mich!

Dieses Buch ist als »Lernerfahrung« konzipiert, nicht als Nachschlagewerk. Wir haben bewusst alles herausgestrichen, was an irgendeiner Stelle des Buchs hinderlich für den Lernprozess sein könnte. Und wenn Sie das Buch zum ersten Mal durcharbeiten, müssen Sie am Anfang beginnen, denn das Buch baut stets darauf auf, was Sie schon gesehen und gelernt haben.

Wir wollen, dass Sie den Denkprozess hinter dem Programmieren lernen.

Manche Leute nennen die Informatik auch »Computerwissenschaft«. Hier ein kleines Geheimnis: Die Computerwissenschaft ist keine Wissenschaft und hat nicht einmal viel mit Computern zu tun (so wie es bei der Astronomie auch nicht um Teleskope geht). Es geht eher um »rechnerisches« Denken, wie man heutzutage sagt. Sobald Sie das gelernt haben, können Sie diese Denkweise leicht auf beliebige andere Probleme, Umgebungen oder Programmiersprachen anwenden.

In diesem Buch verwenden wir Python.

Fahren lernen ohne Auto ist ein bisschen zu theoretisch. Rechnerisch denken lernen ohne Programmiersprache ist eher ein Gedankenexperiment als eine tatsächlich anwendbare Fähigkeit. Daher benutzen wir in diesem Buch die beliebte Programmiersprache Python. Warum diese Sprache besonders gut geeignet ist, erfahren Sie in Kapitel 1. Egal ob Sie das Programmieren nur als Hobby betreiben oder einen sechsstellig dotierten Softwareentwicklerjob landen wollen, Python ist ein guter Startpunkt (und vielleicht ein Endpunkt).

Wir werden nicht jeden Aspekt von Python behandeln.

Nicht einmal im Ansatz. Es gibt eine Menge über Python zu lernen. Dieses Buch ist kein Nachschlagewerk, sondern ein Lehrbuch. Es wird also nicht alles behandelt, was es Wissenswertes zu Python gibt. Wir wollen Ihnen die Grundlagen des Programmierens und des rechnerischen Denkens beibringen, damit Sie ein Buch über jede *beliebige Programmiersprache* zur Hand nehmen können, ohne sich komplett verloren zu fühlen.

Sie können einen Mac, einen PC oder Linux benutzen. Völlig egal.

Da wir in diesem Buch hauptsächlich die plattformunabhängige Programmiersprache Python einsetzen, können Sie einfach das Betriebssystem benutzen, das Ihnen am besten liegt. Die meisten Abbildungen in diesem Buch stammen von einem Mac. Auf Ihrem PC oder Ihrer Linux-Kiste sollte es ähnlich aussehen.

Dieses Buch befürwortet gut strukturierten, lesbaren Code, basierend auf den gängigen »Best Practices«.

Sie wollen Code schreiben, den Sie und andere Leute lesen und verstehen können, Code, der auch noch in kommenden Python-Versionen funktioniert. In diesem Buch bringen wir Ihnen bei, von Anfang an klaren, gut organisierten Code zu schreiben, Code, auf den Sie stolz sein können, Code, den Sie rahmen und an die Wand hängen wollen (aber nicht vergessen, den Rahmen wieder abzuhängen, falls Sie Ihr Rendezvous mit nach Hause bringen). Der einzige Unterschied zu professionell geschriebenem Code ist, dass dieses Buch handgeschriebene Randbemerkungen verwendet, um zu erläutern, was der Code tut. Wir haben gemerkt, dass diese Methode in einem Lehrbuch besser funktioniert als traditionelle Codekommentare. (Es ist nicht schlimm, wenn Sie keinen Schimmer haben, wovon wir hier reden. Nach ein paar Kapiteln werden Sie es wissen.) Aber keine Sorge. Wir bringen Ihnen natürlich bei, wie Sie Ihren Code dokumentieren, und wir zeigen Ihnen Beispiele, wie wir es machen würden. Wir wollen Ihnen auf möglichst direktem Weg zeigen, wie man programmiert, damit Sie den Job erledigen und mit wichtigeren Dingen weitermachen können.

Anmerkungen wie diese.

Programmieren ist eine ernste Sache. Sie werden arbeiten müssen, manchmal auch hart. Programmierer haben eine eigene Denkweise, eine eigene Sicht der Dinge. Manchmal wird Ihnen das Programmieren sehr logisch vorkommen, manchmal aber auch sehr abstrakt oder gar bewusstseinsverändernd. Einige Programmierkonzepte brauchen ihre Zeit, um in Ihr Hirn einzudringen – Sie werden teilweise eine Nacht darüber schlafen müssen, um sie zu kapieren. Aber kein Problem, wir gehen dabei sehr gehirnfreundlich vor. Lassen Sie sich einfach Zeit, die Konzepte auf sich wirken zu lassen, und wiederholen Sie das Material bei Bedarf einfach ein paarmal.

Die Übungen sind NICHT optional.

Die Übungen und sonstigen Aktivitäten in diesem Buch sind *keine* Zugaben, sondern Grundbestandteile dieses Buchs. Manche helfen Ihnen beim Einprägen, andere beim Verständnis und wieder andere bei der Anwendung des Gelernten. Wenn Sie sie überspringen, werden Sie große Teile des Buchs verpassen (und vermutlich ziemlich durcheinanderkommen). Nur die Kreuzworträtsel sind keine Pflicht, obwohl Ihr Gehirn die Wörter dadurch auch einmal in einem anderen Zusammenhang wahrnehmen kann.

Die Redundanz ist beabsichtigt und wichtig.

Ein wesentliches Anliegen eines »Von Kopf bis Fuß«-Buchs ist, dass wir wirklich wollen, dass Sie das Gesagte kapieren und dass Sie sich nach dem Lesen des Buchs an das Gelernte erinnern. Den meisten Referenzwerken geht es nicht ums Behalten und Abrufen, aber in diesem Buch geht es ums Lernen. Daher werden Sie viele Konzepte mehr als einmal sehen.

Die Beispiele sind so knapp wie möglich.

Unsere Leser finden es frustrierend, sich durch 200 Zeilen Beispielcode zu graben, um die zwei Zeilen zu finden, um die es tatsächlich geht. Daher haben die meisten Beispiele in diesem Buch gerade so viel Kontext wie nötig, um den Teil, den Sie lernen sollen, so klar und einfach wie möglich rüberzubringen. Erwarten Sie nicht, dass alle Beispiele robust oder gar voll funktionsfähig sind – sie sind speziell für das Lernen geschrieben. Außerdem haben wir versucht, zumindest die großen Beispiele so zu gestalten, dass sie auch Spaß machen, faszinieren und so richtig cool sind – etwas, das Sie Ihren Freunden und Ihrer Familie gerne zeigen.

Alle Beispieldateien stehen im Web zum Herunterladen bereit. Sie finden sie unter `http://wickedlysmart.com/hflearntocode`.

Unsere Callcentermitarbeiter warten NICHT auf Sie. Aber Sie finden den gesamten Code und die Beispieldateien unter http://wickedlysmart.com/hflearntocod.

Zu den Kopfnuss-Aufgaben gibt es normalerweise keine Lösungen.

Für einige Aufgaben gibt es keine richtige Antwort, bei anderen wiederum ist es ein Teil der Lernerfahrung, dass Sie selbst entscheiden müssen, ob und wann Ihre Antworten richtig sind. In einigen Kopfnuss-Aufgaben finden Sie Hinweise, die Ihnen die richtige Richtung weisen.

Finden Sie Codebeispiele, Hilfe und Diskussionen

Alles, was Sie zu diesem Buch brauchen, finden Sie online unter `http://wickedlysmart.com/hflearntocode`, die Dateien mit dem Beispielcode und zusätzliches englischsprachiges Material eingeschlossen.