

Ein Buch zum Mitmachen und Verstehen

# Programmieren von Kopf bis Fuß



Laden Sie sich  
die wichtigen  
Konzepte direkt in  
Ihr Hirn



Bestücken Sie Ihren  
Werkzeugkasten mit  
Methoden, Funktionen  
und Objekten

Vermeiden  
Sie nervige  
Ein-/Ausgabe-  
Pannen



**Eine allgemeine  
Einführung, deren  
Beispiele in Python  
geschrieben sind**



Verarbeiten Sie  
Ihre Daten wie  
ein echter Profi

Bauen Sie so  
funktionelle wie  
attraktive  
grafische  
Anwendungen



Erfahren Sie, wie sich  
wiederkehrende Aufgaben  
automatisieren lassen

O'REILLY®

Paul Barry & David Griffiths  
Deutsche Übersetzung von Lars Schulten

Die Informationen in diesem Buch wurden mit größter Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Verlag, Autoren und Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für eventuell verbliebene Fehler und deren Folgen. D.h., wenn Sie beispielsweise ein Kernkraftwerk unter Verwendung dieses Buchs betreiben möchten, tun Sie dies auf eigene Gefahr.

Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen. Der Verlag richtet sich im Wesentlichen nach den Schreibweisen der Hersteller. Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Alle Rechte vorbehalten einschließlich der Vervielfältigung, Übersetzung, Mikroverfilmung sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Kommentare und Fragen können Sie gerne an uns richten:

O'Reilly Verlag

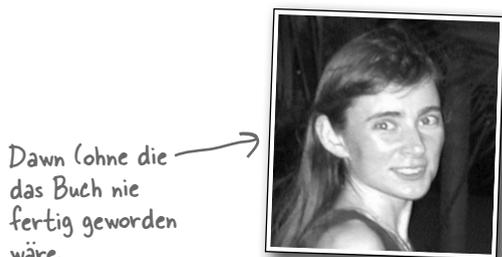
Balthasarstr. 81

50670 Köln

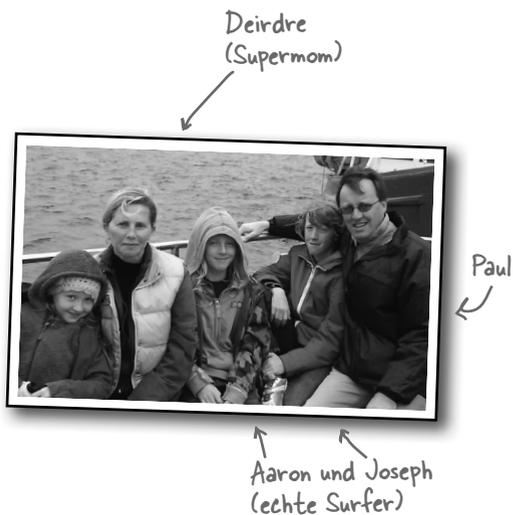
E-Mail: [komentar@oreilly.de](mailto:komentar@oreilly.de)

Copyright der deutschen Ausgabe:

© 2010 by O'Reilly Verlag GmbH & Co. KG



Aideen  
(künftige  
Sängerin) →



Die Originalausgabe erschien 2009 unter dem Titel  
*Head First Programming* bei O'Reilly Media, Inc.

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.

Deutsche Übersetzung und Bearbeitung: Jörg Beyer, Weimar (Lahn)

Lektorat: Christine Haite, Köln

Korrektur: Sibylle Feldmann, Düsseldorf

Satz: Ulrich Borstelmann, Dortmund

Umschlaggestaltung: Karen Montgomery, Boston

Produktion: Andrea Miß, Köln

Belichtung, Druck und buchbinderische Verarbeitung: Media-Print, Paderborn

ISBN 978-3-89721-992-2

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Bei der Produktion dieses Buchs kamen keine Daten zu Schaden.

Wir widmen dieses Buch dem Ersten, der vor einem Computer stand und sich die Frage stellte: »Wie *bringe* ich ihn dazu, das zu tun ... ?«

Und denen, die das Programmieren so komplex machten, dass man ein Buch wie dieses braucht, um es zu lernen.

David: Für Dawn, den klügsten Menschen, der mir je begegnet ist.

Paul: Dieses Buch ist meinem Vater gewidmet, der mich vor 25 Jahren – als ich einen Anstoß brauchte – zur Computerei trieb. Das war ein guter Schubs.

## Die Autoren von Programmieren von Kopf bis Fuß

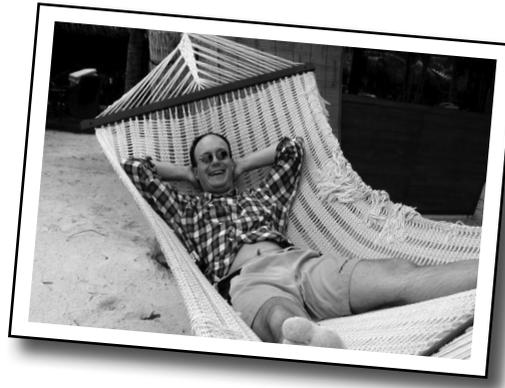
Paul Barry



**Paul Barry** hat vor Kurzem festgestellt, dass er seit fast einem Vierteljahrhundert programmiert. Was ihn nicht wenig schockierte. In dieser Zeit hat er mit vielen verschiedenen Programmiersprachen gearbeitet, in zwei Ländern auf zwei Kontinenten gelebt und gewirkt, geheiratet, drei Kinder bekommen (okay, *bekommen* hat sie eigentlich seine Frau Deirdre, aber irgendwie hat er ja auch seinen Teil geleistet), einen B.Sc. und einen M.Sc. in Computing erworben, neben zwei weiteren Büchern einen Haufen technischer Artikel für das *Linux Journal* geschrieben und es irgendwie geschafft, seine Haare *nicht* zu verlieren ... was sich leider bald ändern könnte.

Als er das erste Mal voller Begeisterung einen Blick auf *HTML mit CSS & XHTML von Kopf bis Fuß* warf, erkannte er sofort, dass dieses Konzept genau der richtige Ansatz war, um Programmieren zu lehren. Er war daher nur zu erfreut, als er gemeinsam mit David die Möglichkeit erhielt, zu zeigen, dass diese Ahnung richtig war.

Normalerweise arbeitet Paul als Lehrer am Institute of Technology, Carlow, in Irland. Als Mitglied der Abteilung für »Computing & Networking« verbringt er seine Tage damit, coole Programmiertechnologien zu erforschen und zu lehren und damit genau das zu tun, was er für den besten Zeitvertreib hält. Seine große Hoffnung ist, dass seine Studenten den Kram, den er sie lehrt, ebenso amüsant finden.



David Griffiths

**David Griffiths** begann im zarten Alter von 12 mit dem Programmieren, nachdem er eine Dokumentation über die Arbeiten von Seymour Papert gesehen hatte. Mit 15 schrieb er eine Implementierung von Paperts Programmiersprache LOGO. Nachdem er an der Universität Mathematik studiert hatte, begann er, Code für Computer und Zeitschriftenartikel für Menschen zu schreiben. Er arbeitete als Trainer für agile Softwareentwicklung, Entwickler und Parkplatzwächter, allerdings nicht in dieser Reihenfolge. Er kann in über zehn Sprachen programmieren, schreiben aber nur in einer. Wenn er nicht schreibt, programmiert oder lehrt, verbringt er seine Zeit auf Reisen mit seiner Frau – und *Von Kopf bis Fuß*-Autorenkollegin – Dawn.

Bevor er *Programmieren von Kopf bis Fuß* schrieb, schrieb er ein anderes Buch namens *Head First Rails*, das ebenfalls sehr lesenswert und ein ausgezeichnetes Geschenk für einen guten Freund oder Verwandten ist.

Sie können ihm auf Twitter folgen:

<http://twitter.com/dgriffiths>

## Über den Übersetzer dieses Buchs

**Lars Schulten** ist freier Übersetzer für IT-Fachliteratur und hat für den O'Reilly Verlag schon unzählige Bücher zu ungefähr allem übersetzt, was man mit Computern so anstellen kann. Eigentlich hat er mal Philosophie studiert, aber mit Computern schlägt er sich schon seit den Zeiten herum, da Windows laufen lernte. Die Liste der Dinge, mit denen er sich beschäftigt, ist ungefähr so lang, launenhaft und heterogen wie die seiner Lieblingsessen und Lieblingsbücher.

Allein tritt er eigentlich nur auf, wenn er mal wieder versucht, den körperlichen Verfall mit sportlicher Betätigung aufzuhalten. Sonst ist er immer in Begleitung eines Buchs, seines Laptops oder Frederics unterwegs. Frederic ist sechs Jahre alt und setzt gern eine sehr kritische Miene auf, wenn Papa die Spielerei mit dem Computer als Arbeit bezeichnet.

## **Verwandte Bücher von O'Reilly**

Einführung in Python

Programmieren mit Python

Python Cookbook

## **Weitere Bücher aus O'Reillys *Von Kopf bis Fuß*-Reihe**

Java™ von Kopf bis Fuß

Objektorientierte Analyse und Design (OOA&D) von Kopf bis Fuß

Softwareentwicklung von Kopf bis Fuß

Entwurfsmuster von Kopf bis Fuß

Servlets und JSP von Kopf bis Fuß

C# von Kopf bis Fuß

SQL von Kopf bis Fuß

PHP & MySQL von Kopf bis Fuß

HTML mit CSS und XHTML von Kopf bis Fuß

JavaScript von Kopf bis Fuß

Datenanalyse von Kopf bis Fuß

Statistik von Kopf bis Fuß

Head First Algebra

Head First Physics

Head First Ajax

Head First Rails

Webdesign von Kopf bis Fuß

Head First PMP

Head First EJB

## Der Inhalt (im Überblick)

|        |  |     |
|--------|--|-----|
|        | Einführung   | xxi |
| 1      | Der Anfang des Programmierens: <i>Orientierungshilfe</i>                       | 1   |
| 2      | Textdaten: <i>Jeder Text zu seiner Zeit</i>                                    | 37  |
| 3      | Funktionen: <i>Ordnung schaffen</i>  | 77  |
| 4      | Daten in Dateien und Listen: <i>Sortieren</i>                                  | 113 |
| 5      | Abbildungen und Datenbanken: <i>Daten ihren Platz zuweisen</i>                 | 145 |
| 6      | Modulare Programmierung: <i>In der Spur bleiben</i>                            | 177 |
| 7      | Grafische Benutzeroberflächen: <i>Visuell-Werdung</i>                          | 215 |
| 8      | GUIs und Daten: <i>Grafisch Dateneingabe</i>                                   | 257 |
| 8 1/2  | Ausnahmen und Dialogfenster: <i>Nachricht angekommen?</i>                      | 293 |
| 9      | Elemente grafischer Oberflächen: <i>Das richtige Werkzeug wählen</i>           | 313 |
| 10     | Eigene Widgets und Klassen: <i>Objekte im Sinn</i>                             | 349 |
| Anhang | Was übrig bleibt: <i>Die Top-Ten der Dinge (die wir nicht behandelt haben)</i> | 385 |
|        | Index  | 397 |

## Der 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, 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, etwas über das Programmieren zu wissen?

|  |        |
|--|--------|
| Für wen ist dieses Buch?               | xxii   |
| Wir wissen, was Sie jetzt denken       | xxiii  |
| Metakognition                          | xxv    |
| So machen Sie sich Ihr Gehirn Untertan | xxvii  |
| Lies mich                              | xxviii |
| Die technischen Gutachter              | xxx    |
| Danksagungen                           | xxxix  |

# Der Anfang des Programmierens

# 1

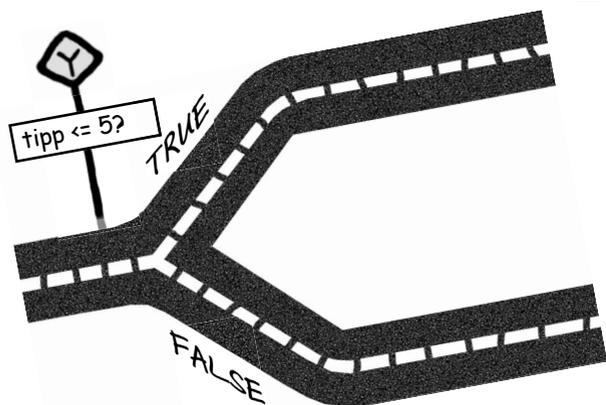
## Orientierungshilfe

### Eigene Programme verleihen Ihnen die Macht, Ihren PC zu steuern.

Fast jeder weiß, wie man mit einem Computer *arbeitet*, aber nur wenige machen den nächsten Schritt und lernen, wie man ihn *steuert*. Wenn Sie Software nutzen, die von anderen entwickelt wurde, sind Sie immer auf das beschränkt, was Sie *deren* Meinung nach tun möchten. Schreiben Sie Ihre eigenen Programme, und nur noch Ihre eigene Vorstellungskraft kann Sie einschränken. Das Programmieren macht Sie kreativer, lässt Sie präziser denken und lehrt Sie, Probleme logisch zu analysieren und zu lösen.

### Wollen Sie lieber programmiert oder Programmierer sein?

|   |    |
|---|----|
| Mit Programmieren können Sie mehr erreichen                 | 2  |
| Und wie führt man das Programm aus?                         | 5  |
| Eine neue Programmdatei erstellen                           | 6  |
| Code vorbereiten und ausführen                              | 7  |
| Ein Programm ist mehr als eine einfache Folge von Befehlen  | 12 |
| Codeville: Ihr Programm ist wie ein Straßennetz             | 13 |
| Verzweigungen sind Codeschnittstellen                       | 14 |
| if/else-Verzweigungen                                       | 15 |
| Der Python-Code braucht verknüpfte Verzweigungen            | 20 |
| In Python werden Pfade durch Einrückung verbunden           | 21 |
| Mit Schleifen können Sie Code immer wieder ausführen lassen | 28 |
| Pythons while-Schleife                                      | 29 |
| Ihr Programmierwerkzeugkasten                               | 35 |



## Textdaten

**Jeder Text zu seiner Zeit**

## 2

**Stellen Sie sich vor, Sie müssten ohne Worte kommunizieren.**

Alle Programme verarbeiten Daten – und eine der wichtigsten Arten von Daten ist **Text**.

In diesem Kapitel werden Sie das Basiswissen zu **Textdaten** erhalten. Sie werden Text

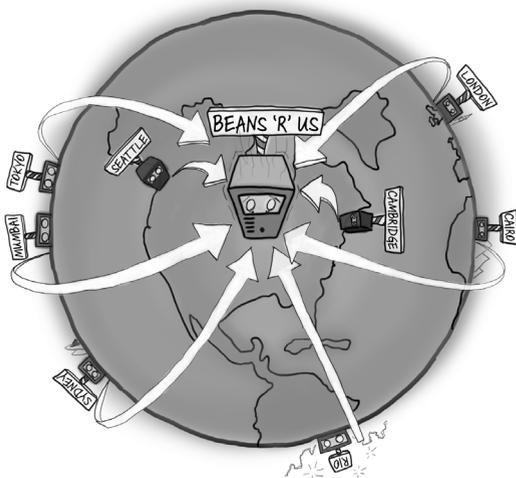
**suchen** und automatisch genau das bekommen, **was Sie suchten**. Und unterwegs

werden Sie grundlegende Konzepte der Programmierung wie **Methoden** aufsammeln

und erfahren, wie Sie sie einsetzen können, **um sich Ihre Daten gefügig zu machen**.

Und schließlich werden Sie Ihren Programmen mithilfe von **Codebibliotheken** im Handumdrehen **Superkräfte geben**.

|  |    |
|--|----|
| Der neue Sternback-Auftrag                         | 38 |
| Hier ist der aktuelle Sternback-Code               | 39 |
| Der Preis ist in das HTML eingebettet              | 41 |
| Ein String ist eine Folge von Zeichen              | 41 |
| Zeichen in Text finden                             | 42 |
| Aber wie erhält man mehr als ein Zeichen?          | 43 |
| Beans'R'Us belohnt Stammkunden                     | 50 |
| Suchen ist nicht leicht                            | 52 |
| Python-Daten sind schlau                           | 54 |
| Strings und Zahlen sind unterschiedlich            | 64 |
| Das Programm hat den Beans'R'Us Server überfordert | 67 |
| Zeit ... hätten wir nur mehr davon                 | 68 |
| Sie nutzen bereits eine Bibliothek                 | 69 |
| Die Ordnung ist wiederhergestellt                  | 74 |
| Ihr Programmierwerkzeugkasten                      | 75 |



## Funktionen

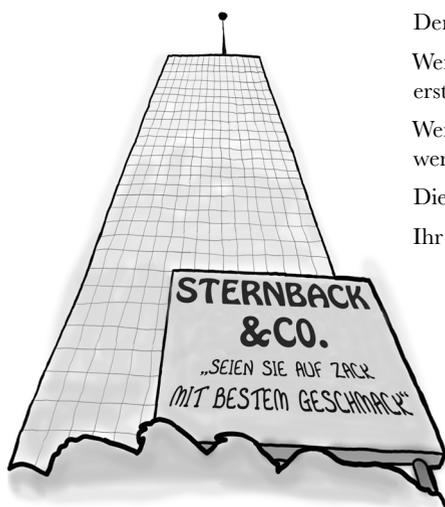
# 3

### Ordnung schaffen

#### Wenn Programme wachsen, wird Code unübersichtlich.

Unübersichtlicher und komplexer Code kann schwer zu lesen und noch schwerer zu warten sein. Eine Möglichkeit, diese Komplexität in den Griff zu bekommen, ist der Einsatz von **Funktionen**. Funktionen sind **Codeeinheiten**, die Sie nach Bedarf in Ihren Programmen einsetzen. Sie ermöglichen Ihnen, **gemeinsam genutzte Aktionen auszulagern**, und das bedeutet, dass Sie Ihren Code **besser lesbar** und **leichter wartbar** machen. In diesem Kapitel werden Sie entdecken, wie Ihnen etwas Wissen zu Funktionen **das Programmiererleben erheblich vereinfachen kann**.

|   |     |
|---|-----|
| Sternback gehen die Bohnen aus!   | 78  |
| Was muss das neue Programm leisten?   | 79  |
| Vermeiden Sie Codeverdopplung ...   | 81  |
| Codewiederverwendung mit Funktionen   | 82  |
| Bringen Sie die Dinge in die richtige Reihenfolge                               | 84  |
| Mit dem Befehl return Daten zurückliefern                                       | 87  |
| Nutze das Web, Luke   | 93  |
| Die Funktion sendet immer die gleiche Nachricht                                 | 94  |
| Funktionsverdopplung mit Parametern vermeiden                                   | 96  |
| Jemand hat an Ihrem Code herumgepfuscht   | 102 |
| Der Rest des Programms kann die Variable password nicht sehen                   | 104 |
| Wenn Sie eine Funktion aufrufen, erstellt der Computer eine neue Variablenliste | 105 |
| Wenn Sie eine Funktion verlassen, werden Ihre Variablen weggeworfen             | 106 |
| Die Sternback-Lager sind gefüllt!   | 110 |
| Ihr Programmierwerkzeugkasten   | 111 |



## Daten in Dateien und Listen

### Sortieren

# 4

## Mit Ihren Programmen entwickeln sich auch die Datenverarbeitungsanforderungen.

Und wenn Sie mit vielen Daten arbeiten müssen, wird es schnell recht mühselig, einzelne Variablen für jedes einzelne Datenelement zu verwenden. Programmierer nutzen deswegen ziemlich beeindruckende Behälter oder Container (die als **Datenstrukturen** bezeichnet werden), um sich die Arbeit mit vielen Daten zu vereinfachen. Häufig kommen alle Daten aus Dateien, die auf der Festplatte gespeichert sind. Aber wie arbeitet man mit Daten in Dateien? Es wird sich herausstellen, dass es ein Kinderspiel ist. Blättern Sie um und erfahren Sie, warum!

|  |     |
|--|-----|
| Starke Brandung in Codeville   | 114 |
| Die höchste Punktzahl in der Ergebnisdatei finden                    | 115 |
| Dateien mit dem Öffnen-Lesen-Schließen-Muster durchlaufen            | 116 |
| Die Datei enthält nicht nur Zahlen ...                               | 120 |
| Die Zeilen beim Lesen spalten  | 121 |
| Die Methode <code>split()</code> zerlegt den String                  | 122 |
| Aber Sie benötigen mehr Ergebnisse                                   | 126 |
| 3 Highscores machen den Code komplexer                               | 127 |
| Eine geordnete Liste macht den Code erheblich einfacher              | 128 |
| Im Speicher sortieren ist einfacher                                  | 129 |
| Sie können unmöglich eine eigene Variable für jede Datenzeile nutzen | 130 |
| Mit Listen können Sie ganze Datenzüge verwalten                      | 131 |
| Listen in Python   | 132 |
| Die Liste vor Anzeige der Ergebnisse sortieren                       | 136 |
| Die Punktzahlen in absteigender Folge sortieren                      | 139 |
| Und wer ist der Sieger?  | 142 |
| Wir haben die Namen der Surfer vergessen                             | 143 |
| Ihr Programmierwerkzeugkasten  | 144 |

Kapitel 4 schon -  
höchste Zeit  
für einen Ritt über  
die Wellen.



## Abbildungen und Datenbanken

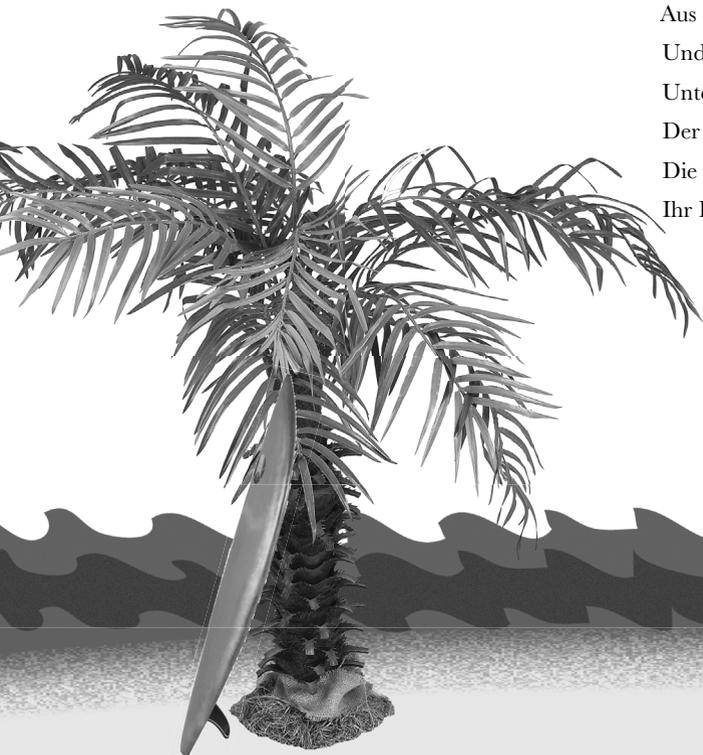
# 5

### Daten ihren Platz zuweisen

#### Listen sind nicht der einzige Datencontainer.

Programmiersprachen bieten weitere Spielzeuge zum Festhalten von Daten, und unser erwähntes Werkzeug, Python, macht da keine Ausnahme. In diesem Kapitel werden Sie Werte mit Namen **verknüpfen** und dazu eine Datenstruktur nutzen, die unter vielen Namen bekannt ist, Hash, Tabelle, **Abbildung**, und unter Python *Dictionary*. Und bei der Speicherung von Daten wollen wir uns jetzt nicht mehr auf Textdaten in Dateien beschränken, sondern auf Daten in einem *externen Datenbanksystem zugreifen*. **Information** ist alles, und da es die ohne Daten nicht gibt, blättern Sie um und beginnen damit, Ihre stetig wachsenden Programmierfertigkeiten auf einige coole Datenverarbeitungsaufgaben anzuwenden.

|  |     |
|--|-----|
| Wer hat den Wettbewerb gewonnen?                 | 146 |
| Punkte und Namen verbinden                       | 150 |
| Einen Schlüssel und einen Wert verbinden         | 153 |
| Abbildungen mit for durchlaufen                  | 154 |
| Die Daten sind nicht sortiert                    | 158 |
| Wenn Daten komplexer werden                      | 160 |
| Aus einer Funktion eine Datenstruktur liefern    | 164 |
| Und hier: Ihr neues Brett!                       | 168 |
| Unterdessen im Studio ...                        | 169 |
| Der Code bleibt gleich; die Funktion ändert sich | 170 |
| Die vKbF-TV-Daten bringen Geld!                  | 174 |
| Ihr Programmierwerkzeugkasten                    | 175 |



## Modulare Programmierung

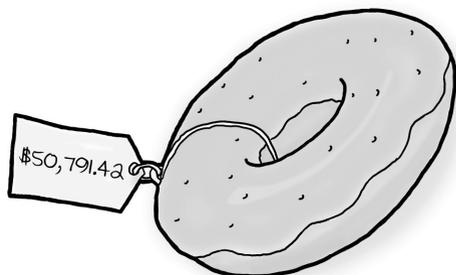
### In der Spur bleiben

# 6

#### Ihr Code geht in viele Programme ein.

Und obgleich **Teilen** eine gute Sache ist, müssen Sie *aufpassen*. Es könnte sein, dass Programmierer Ihren Code nehmen und auf **unerwartete** Weise nutzen oder andere ihn einfach ändern, ohne Sie darüber zu informieren. Eventuell möchten Sie eine Funktion in all Ihren Programmen nutzen, und mit der Zeit **ändert** sich der Code dieser Funktion, damit er weiterhin Ihre Anforderungen erfüllt. Schlaue Programmierer nutzen *modulare Programmiertechniken*, um ihre Arbeitsbelastung unter Kontrolle zu halten. Finden Sie auf den folgenden Seiten heraus, wie man das anstellt ...

|  |     |
|--|-----|
| Fit von Kopf bis Fuß aktualisiert seine Systeme          | 178 |
| Das Programm muss eine Transaktionsdatei erstellen       | 179 |
| Strings mit Strings formatieren                          | 180 |
| Spät in der Nacht schneit eine Mail rein                 | 187 |
| 50000 ... für einen Donut?!                              | 188 |
| Nur die Geschäfte aus Ihrem Programm sind betroffen      | 189 |
| Die neue Bank nutzt ein neues Format                     | 190 |
| Das Programm in der Bar nutzt immer noch das alte Format | 191 |
| Aktualisieren Sie nicht einfach Ihren Code               | 192 |
| Und wie erstellen wir ein Modul?                         | 193 |
| Auch die Transaktionsdatei funktioniert                  | 199 |
| Der Fitnessclub hat einen neuen Wunsch                   | 200 |
| Der Sternback-Code                                       | 205 |
| Die beiden Rabattfunktionen haben den gleichen Namen     | 206 |
| Vollständigqualifizierte Namen verhindern Verwirrungen   | 207 |
| Der Rabatt lässt die Kunden herbeiströmen                | 213 |
| Ihr Programmierwerkzeugkasten                            | 214 |



## Grafische Benutzeroberflächen

# 7

### Augen- und Ohrenschaus

#### Ihre Fertigkeiten als Programmierer nehmen stetig zu.

Aber es ist ein Jammer, dass Ihre Programme nicht *ansehnlicher* sind. Fragen und Antworten auf der Konsole anzuzeigen zu können, ist ja ganz hübsch, aber doch recht retro, oder nicht? Fehlt eigentlich nur noch grüner Text auf schwarzem Hintergrund zum wahren IT-Steinzeitgefühl. Es muss doch eine *bessere* Möglichkeiten geben, mit Benutzern zu kommunizieren – und die gibt es natürlich: **grafische Benutzeroberflächen** oder **GUIs**. Klingt cool und komplex und kann beides sein. Aber keine Angst, mit ein zwei Tricks haben Sie Ihren grafischen Code in null Komma nichts laufen.

|   |     |
|---|-----|
| Von Kopf bis Fuß-TV produziert auch Game-Shows    | 216 |
| pygame ist plattformübergreifend                  | 220 |
| 0... 2... 1... 9... abheben!                      | 230 |
| tkinter schenkt Ihnen die Ereignisschleife        | 234 |
| tkinter hat massenhaft Optionen                   | 235 |
| Das GUI funktioniert, macht aber nichts           | 238 |
| Code mit Button-Ereignissen verbinden             | 239 |
| Jetzt ist das GUI-Programm für ein Casting bereit | 244 |
| Aber noch ist der Moderator nicht zufrieden       | 246 |
| Labeln Sie!                                       | 249 |
| Ihr Programmierwerkzeugkasten                     | 255 |



## GUIs und Daten

## 8

**Grafische Dateneingabe****GUIs verarbeiten nicht nur Ereignisse, sondern auch Daten.**

Fast alle GUI-Anwendungen müssen Benutzerdaten lesen, und die Wahl der richtigen Eingabelemente kann entscheiden, ob Ihre Benutzeroberfläche eine *Dateneingabehölle* oder der siebte *Benutzerhimmel* ist. Widgets können einfachen Text akzeptieren oder nur eine Optionsliste präsentieren. Es gibt viele verschiedene Widgets, und das heißt, dass Sie viele Optionen haben. Welche Sie davon wählen, kann natürlich ganz entscheidende Auswirkungen haben. Es ist an der Zeit, dass Sie Ihre GUI-Programme **weiterentwickeln**.

|  |     |
|--|-----|
| PPD braucht ein neues Versandsystem                              | 258 |
| Es gibt bereits einen Entwurf für die Benutzeroberfläche         | 259 |
| Datenerfassung im GUI  | 260 |
| Mit Entry- und Text-Widgets kann Ihr GUI Textdaten aufnehmen     | 261 |
| Daten in Textfeldern lesen und schreiben                         | 262 |
| Text-Felder sind komplizierter                                   | 263 |
| Eine der Sendungen ging in die Irre                              | 270 |
| In die Felder können beliebige Werte eingegeben werden           | 271 |
| Radiobuttons beschränken die Eingabe                             | 272 |
| Radiobuttons in tkinter erstellen                                | 273 |
| Die Radiobuttons sollten zusammenarbeiten                        | 275 |
| Die Radiobuttons können ein Modell teilen                        | 276 |
| Das System sagt den anderen Widgets, wenn sich das Modell ändert | 277 |
| Wie man in tkinter Modelle nutzt                                 | 278 |
| PPD expandiert   | 282 |
| Es gibt zu viele Depots im GUI                                   | 283 |
| Ein OptionMenu bietet Ihnen so viele Optionen, wie Sie benötigen | 284 |
| Das Modell bleibt gleich   | 285 |
| Ihr Programmierwerkzeugkasten                                    | 292 |

Was Ihr macht, ist mir egal. Ich bin ausgewählt und bleibe das auch.



Cambridge, MA

Ich ebenfalls.



Cambridge, UK

Und ich auch.



Seattle, WA

# 8½

## Ausnahmen und Dialogfenster

### Nachricht angekommen?

**Manchmal geht etwas schief, und Sie müssen sich darum kümmern.**

Es gibt immer Dinge, die Sie nicht im Griff haben. Netzwerke fallen aus. Dateien verschwinden. Schlaue Programmierer wissen, wie sie mit derartigen **Fehlern** umgehen, und bieten in ihren Programmen **Ausweichlösungen**. Gute Software informiert den Nutzer, wenn etwas Übles passiert, und sagt ihm, was erforderlich ist, um die Angelegenheit zu regeln. Wenn Sie lernen, wie Sie **Ausnahmen** und **Dialogfenster** einsetzen, können Sie die Zuverlässigkeit und Qualität Ihrer Software steigern.

|  |     |
|--|-----|
| Was ist das für ein Geruch?  | 294 |
| Jemand hat die Dateiberechtigungen geändert                                      | 295 |
| Als es die Datei nicht schreiben konnte,<br>löste Ihr Programm eine Ausnahme aus | 296 |
| Die Ausnahme abfangen  | 297 |
| Ausnahmen überwachen mit try/except  | 298 |
| Es gibt ein Problem mit der Ausnahmebehandlung                                   | 302 |
| Ein Dialogfenster verlangt Aufmerksamkeit  | 303 |
| Dialogfenster in Python  | 304 |
| Ihr Programmierwerkzeugkasten  | 311 |



## Elemente grafischer Oberflächen

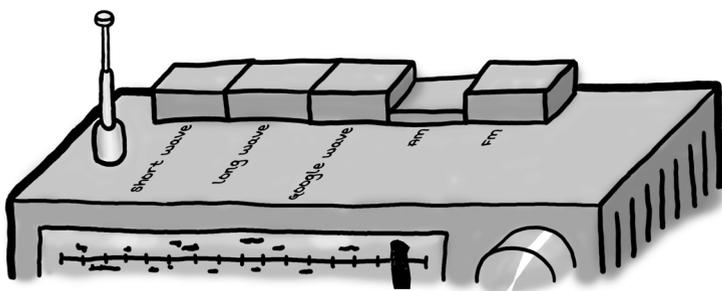
### Das richtige Werkzeug wählen

# 9

#### Gut verwendbare Programme zu schreiben, ist nicht schwer..

Bei GUI-Anwendungen gibt es einen ganz entscheidenden Unterschied zwischen *funktionierenden* Schnittstellen und **nützlichen** sowie **effektiven** Schnittstellen. Wie man das richtige Werkzeug für eine Aufgabe wählt, lernt man durch Erfahrung, und die erwirbt man nur, indem man mit den verfügbaren Werkzeugen arbeitet. In diesem Kapitel werden Sie Ihre Fertigkeiten im Aufbau von GUI-Anwendungen erweitern. Es gibt noch eine Menge nützlicher Widgets, die auf Sie warten. Blättern Sie also um und lassen Sie uns fortfahren.

|   |     |
|---|-----|
| Zeit zu mixen   | 314 |
| Die Musik spielte einfach weiter ...                        | 318 |
| Nicht alle Ereignisse werden von Buttons erzeugt            | 319 |
| Das Ereignis abfangen reicht nicht                          | 326 |
| Zwei Buttons oder nicht zwei Buttons? Das ist die Frage ... | 328 |
| Eine Checkbox ist ein An/Aus-Schalter                       | 331 |
| Checkboxen in tkinter                                       | 332 |
| Die Lautstärke anpassen                                     | 336 |
| Einen Schieber mit Skala modellieren                        | 337 |
| Die Lautstärke mit pygame anpassen                          | 339 |
| Mit tkinter den Rest erledigen                              | 340 |
| Der DJ ist happy!   | 347 |
| Ihr Programmierwerkzeugkasten                               | 348 |



# 10

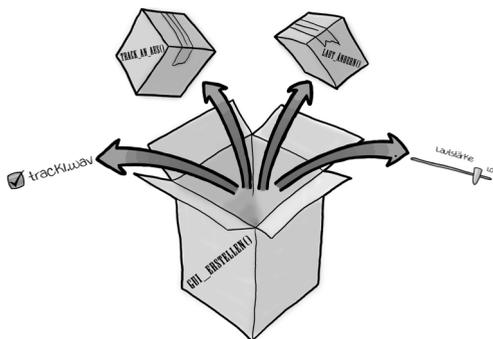
## Eigene Widgets und Klassen

### Objekte im Sinn

**Anforderungen können komplex sein, Programme müssen es nicht sein.**

Indem Sie Objektorientierung nutzen, können Sie Ihre Programme sehr **leistungsfähig** machen, ohne dazu Unmengen an Code zusätzlich schreiben zu müssen. Lesen Sie weiter und erfahren Sie, wie Sie **eigene Widgets** erstellen, die genau das machen, was Sie *wollen*, und Ihnen dazu verhelfen, **Ihre Programmierfertigkeiten weiterzuentwickeln**.

|  |     |
|--|-----|
| Der DJ möchte mehrere Stücke spielen   | 350 |
| Den Code für die Stücke als Funktion speichern                               | 351 |
| Die neue Funktion enthält andere Funktionen                                  | 356 |
| Die Funktion muss Widgets und Ereignis-Handler erstellen                     | 357 |
| Der DJ ist verwirrt  | 362 |
| Widgets gruppieren   | 363 |
| Frame-Widgets enthalten andere Widgets                                       | 364 |
| Klassen sind Maschinen zur Erstellung von Objekten                           | 366 |
| Klassen haben Methoden, die Verhalten definieren                             | 367 |
| Aber wie rufen Objekte Methoden auf?   | 369 |
| Die Klasse MixerPanel hat große Ähnlichkeit mit der Funktion gui_erstellen() | 370 |
| Klassen = Methoden + Daten   | 372 |
| Der DJ hat ein ganzes Verzeichnis voll mit Musik                             | 378 |
| It's party time!   | 382 |
| Ihr Programmierwerkzeugkasten  | 383 |
| Der Aufbruch ...   | 384 |
| Es war nett mit Ihnen hier in Codeville!                                     | 384 |



## Anhang: Was übrig bleibt

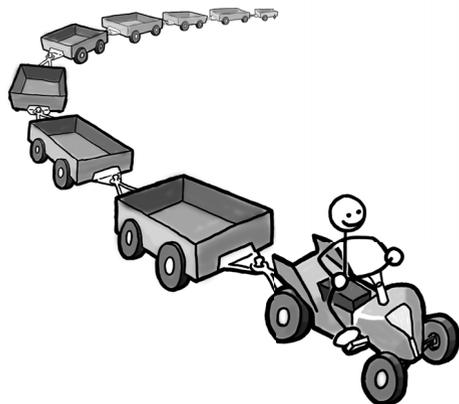


### Die Top-Ten der Dinge (die wir nicht behandelt haben)

#### Sie haben einen weiten Weg zurückgelegt.

Doch programmieren zu lernen ist eine Aufgabe, die nie abgeschlossen ist. Je mehr Sie programmieren, umso wichtiger wird es, dass Sie **neue Wege erforschen, um bestimmte Dinge zu tun**. Sie müssen sich mit **neuen Werkzeugen** und **neuen Techniken** vertraut machen. Leider bietet dieses Buch einfach nicht genügend Raum dafür, Ihnen all das zu zeigen, was eventuell nützlich für Sie werden könnte. Deswegen finden Sie hier eine Liste der zehn wichtigsten Dinge, die wir nicht behandelt haben, die Sie vielleicht aber als Nächstes lernen sollten.

|  |     |
|--|-----|
| 1. »Den Python-Weg« wählen               | 386 |
| 2. Mit Python 2 arbeiten                 | 387 |
| 3. Andere Programmiersprachen            | 388 |
| 4. Automatisiertes Testen                | 389 |
| 5. Debuggen                              | 390 |
| 6. Kommandozeilenausführung              | 391 |
| 7. OOP ist etwas zu kurz gekommen        | 392 |
| 8. Algorithmen                           | 393 |
| 9. Fortgeschrittene Programmierthemen    | 394 |
| 10. Andere IDEs, Shells und Texteditoren | 395 |

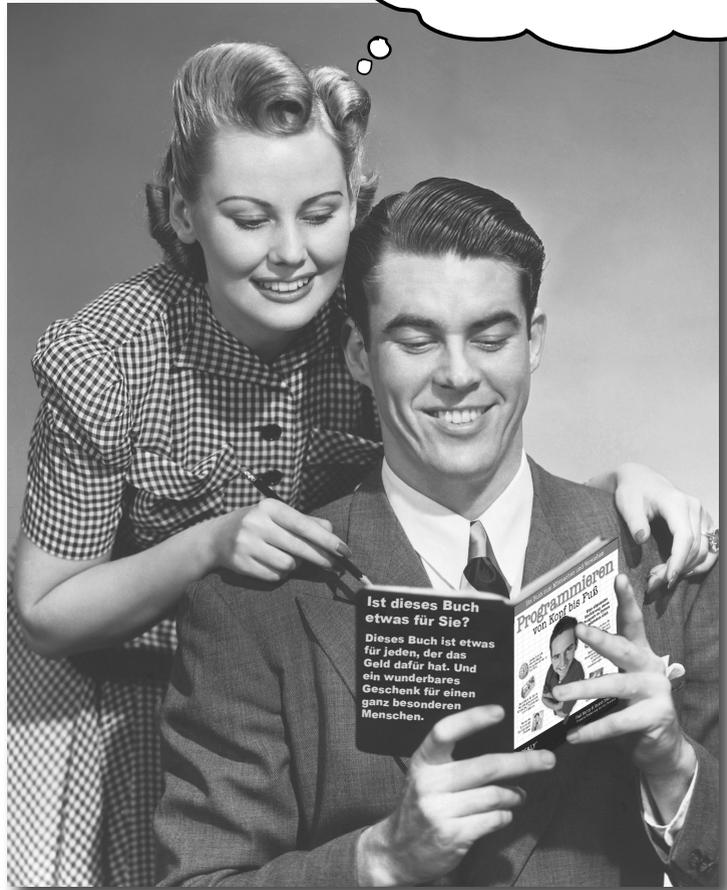




Wie man dieses Buch benutzt

## **Einführung**

Ich kann einfach nicht fassen,  
dass *so etwas* in einer Einführung  
in die Programmierung steht!



In diesem Abschnitt beantworten wir die brennende  
Frage: »Und? Warum STEHT so was in einer  
Einführung in die Programmierung?«

## Für wen ist dieses Buch?

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

- 1 Wüschtent Sie, Sie wüssten, wie Sie Ihren Computer steuern und dazu bringen können, ganz neue Dinge zu tun?
- 2 Möchten Sie programmieren lernen, damit Sie der neue Stern am Softwarehimmel werden, Millionen scheffeln und sich bald auf Ihrer eigenen Insel zur Ruhe setzen können?
- 3 Ziehen Sie es vor, tatsächlich etwas zu tun? Die Dinge anzuwenden, die Sie gelernt haben? Ist das besser, als stundenlang jemandem zuzuhören, der sich in einem endlosen Vortrag über irgendwas ergeht?

← Gut, das ist vielleicht etwas übertrieben, aber alle haben ja mal klein angefangen, oder?

... dann ist dieses Buch etwas für Sie.

## Wer sollte eher die Finger von diesem Buch lassen?

Wenn Sie eine dieser Fragen mit »Ja« beantworten müssen:

- 1 Sind Sie ein erfahrener Programmierer? Wissen Sie bereits, wie man programmiert?
- 2 Suchen Sie nach einer knappen Einführung in oder eine Referenz für Python?
- 3 Würden Sie sich lieber von 15 kreischenden Affen die Zehennägel ziehen lassen, als einmal etwas Neues auszuprobieren? Sind Sie der Ansicht, dass ein Buch zur Programmierung *wirklich alles* behandeln muss und richtig gut eigentlich nur sein kann, wenn es den Leser zu Tode langweilt?

... dann ist dieses Buch **nicht** das richtige für Sie.



[Anmerkung aus dem Marketing: Dieses Buch ist etwas für jeden, der eine Kreditkarte besitzt. Und auch Barzahlung ist möglich.]

# Wir wissen, was Sie gerade denken.

»Wie kann *das* ein ernsthaftes Buch zur Programmierung sein?«

»Was sollen all die Abbildungen?«

»Kann ich 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 sie 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. Die späte Einsicht, dass Sie nie diese »Partybilder« auf Ihre Facebook-Seite hätten stellen dürfen. 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.«



Ihr Gehirn denkt, DAS HIER zu speichern, lohne sich nicht.



## 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 Kognitions-wissenschaft, 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:

**Wir setzen Bilder ein.** 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. **Wir setzen Text in oder neben die Grafiken**, auf die sie sich be-ziehen, anstatt darunter oder auf eine andere Seite. Die Leser werden auf den Bildinhalt bezogene Probleme dann mit *doppelt* so hoher Wahrscheinlichkeit lösen können.

**Wir verwenden 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. Wir halten daher keinen Vortrag, sondern erzählen Geschichten. Wir benutzen eine zwanglose Sprache. Wir nehmen uns selbst nicht zu ernst. Würden Sie einer anregenden Unterhaltung beim Abendessen mehr Aufmerksamkeit schenken oder einem Vortrag?

**Wir bringen 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, begeis-tert 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.

**Wir ziehen die Aufmerksamkeit des Lesers auf uns – und behalten 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.

**Wir sprechen 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 herzerreißenden 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, was alle anderen schwierig finden. Oder wenn Sie merken, dass Sie etwas können, das 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 lernen möchten, wie man programmiert, 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 ein 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, das 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 etwas 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 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**.

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 80 **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.

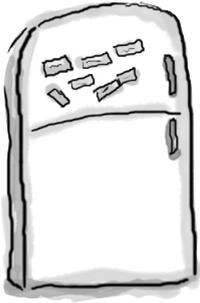
Wir haben **mehrere unterschiedliche Lernstile** eingesetzt, denn vielleicht bevorzugen *Sie* ein Schritt-für-Schritt-Vorgehen, während jemand anders erst einmal den groben Zusammenhang verstehen und ein Dritter einfach nur ein Codebeispiel sehen möchte. Aber ganz abgesehen von den jeweiligen Lernvorlieben 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.

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*.



## 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.

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

### 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 es, als würde jemand anderes Ihr Training für Sie absolvieren. 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 Schlafen. 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 Schreiben Sie viel Code!

Programmieren lernt man nur auf eine Weise: **indem man viel programmiert!** Und das werden Sie in diesem Buch machen. Programmieren ist ein Handwerk. Gut wird man nur durch Übung, und die erhalten Sie hier: Jedes Kapitel enthält eine Menge Übungen, die Sie lösen müssen. Überspringen Sie diese nicht – das Lernen erfolgt zu großen Teilen beim Lösen der Übungen. Zu jeder Übung gibt es eine Lösung – werfen Sie ruhig einen Blick darauf, wenn Sie hängen bleiben! Aber versuchen Sie erst, das Problem zu lösen, bevor Sie sich die Lösung ansehen. Auf alle Fälle sollten Sie die Sache ans Laufen bringen, bevor Sie im Buch weitergehen.

## Lies mich

Dies ist ein Lehrbuch, keine Referenz. Wir haben mit Absicht alles weggelassen, was Ihnen dabei in die Quere kommen könnte, das zu lernen, was wir am jeweiligen Punkt des Buchs behandeln. Beim ersten Lesen sollten Sie unbedingt auch am Anfang des Buchs beginnen. Das Buch geht zu jedem Zeitpunkt davon aus, dass Sie bestimmte Dinge bereits gesehen und gelernt haben.

### **Dies ist nicht Python von Kopf bis Fuß.**

In diesem Buch nutzen wir Version 3 der Programmiersprache Python, aber das macht dieses Buch trotzdem nicht zu *Python von Kopf bis Fuß*. Wir haben Python gewählt, weil es gleichermaßen eine ausgezeichnete Programmiersprache für die ersten Schritte in die Programmierung wie für das Erwachsenwerden als Programmierer ist. Es könnte sogar sein, dass Sie nie eine andere Programmiersprache als Python lernen und nutzen müssen (obgleich Ihr Arbeitgeber da eventuell anderer Meinung sein könnte). Aber irgendwo muss man anfangen. Und uns fällt keine geeignetere Programmiersprache als Python ein, wenn es darum geht, das Programmieren zu lernen. Dennoch soll dieses Buch keine Einführung in Python sein. Es ist als Einführung in die *Programmierung* gedacht. Die meisten Dinge, die wir Ihnen hier vorstellen, sollen also die *Programmierkonzepte veranschaulichen*, nicht die Python-Funktionalitäten.

### **Sie müssen Python 3 auf Ihrem Rechner installieren.**

Damit Sie die Programme in diesem Buch ausführen können, müssen Sie Python 3 herunterladen und auf Ihrem Rechner installieren. Das ist nicht so schwer, wie es klingt. Statten Sie der Python-Download-Seite einen Besuch ab und wählen Sie die Option, die für Ihr System am besten geeignet ist. Achten Sie allerdings darauf, dass Sie Version 3 von Python nehmen, *nicht* Version 2: ***<http://www.python.org/download>***.

### **Wir beginnen mit den grundlegenden Konzepten der Programmierung, dann lassen wir die Programmierung unmittelbar praktisch werden.**

In Kapitel 1 behandeln wir die Grundlagen der Programmierung. Damit können Sie, schon wenn Sie es zu Kapitel 2 geschafft haben, Programme erstellen, die etwas Richtiges, Nützliches und – ja! – Vergnügliches tun. Vermutlich werden Sie erstaunt sein, wie viel man mit den kaum ein Dutzend Codezeilen in Kapitel 2 erreichen kann. Der Rest des Buchs baut auf Ihren kontinuierlich wachsenden Programmierfertigkeiten auf und verwandelt Sie im Handumdrehen vom *Programmieranfänger* zum *Programmierexperten*.

### **Die Aktivitäten sind NICHT optional.**

Die Übungen und Aktivitäten sind kein Beiwerk. Sie sind ein wesentlicher Bestandteil des Buchs. Einige von ihnen sollen Ihnen beim Merken helfen, andere beim Verstehen, und einige helfen Ihnen, das Gelernte anzuwenden. *Überspringen Sie nichts.*

### **Die Redundanz ist beabsichtigt und wichtig.**

Eins der Dinge, das ein Von Kopf bis Fuß-Buch grundlegend anders macht, ist, dass wir möchten, dass Sie die Sache *wirklich* verstehen. Und wir möchten, dass Sie nach Beendigung des Buchs behalten, was Sie gelernt haben. Einprägen und erinnern sind bei den meisten Lehr- oder Referenzbüchern nicht unbedingt das erste Ziel – in diesem Buch schon. Deswegen werden Ihnen hier einige Konzepte mehrfach begegnen.

### **Die Codebeispiele sind so schlank wie möglich.**

Unsere Leser sagen uns, dass sie es frustrierend finden, sich in 200 Zeilen Code graben zu müssen, um die beiden Zeilen zu finden, die sie wirklich verstehen müssen. Die meisten Beispiele in diesem Buch werden mit so wenig Kontext wie möglich gezeigt, damit der Teil, den Sie lernen sollen, klar und einfach ist. Sie dürfen nicht erwarten, dass der Code robust oder gar vollständig ist. Die Beispiele in diesem Buch wurden speziell für Lehrzwecke geschrieben und sind nicht immer vollständig arbeitsfähig.

Wir haben viele der Codebeispiele in diesem Buch ins Web gestellt, damit Sie die Teile, die Sie benötigen, nicht von Hand eingeben müssen. Die Originale finden Sie an zwei Stellen:

<http://www.headfirstlabs.com/books/hfprog/> und

<http://programming.itcarlow.ie>

Der für die deutsche Ausgabe lokalisierte Code liegt unter

[http://examples.oreilly.de/german\\_examples/hfprogrammierer/](http://examples.oreilly.de/german_examples/hfprogrammierer/)

### **Zu den Kopfnuss-Übungen gibt es keine Lösungen.**

Bei einigen Übungen gibt es keine richtige Antwort, und bei anderen gehört es zum Lernprozess, den die Kopfnuss-Übungen anstoßen sollen, dass Sie selbst entscheiden müssen, ob und wann Ihre Antworten richtig sind. Manchmal finden Sie auch Hinweise, die Sie in die richtige Richtung lenken.

## Die technischen Gutachter

Doug Hellman



Ted Leung



Jeremy Jones



Bill Mietelski



### Technische Gutachter:

**Doug Hellmann** ist Senior Software Engineer bei Racemi und war Cheflektor des *Python Magazine*. Mit Python arbeitet er seit Version 1.4. Zuvor arbeitete er hauptsächlich mit C auf einer Vielzahl von Unix- und Nicht-Unix-Systemen. Die Projekte, an denen er mitgearbeitet hat, reichen von Mapping-Anwendungen bis zur Produktion von Medizinnachrichten, etwas Banking zum Ausgleich eingestreut. In seiner Freizeit arbeitet er an einer Reihe von Open Source-Projekten mit, liest Science-Fiction, Geschichtsbücher und Biografien und schreibt die Blog-Reihe »Python Module of the Week«.

**Jeremy Jones** ist Koautor von *Python for Unix and Linux System Administration* und arbeitet seit 2001 aktiv mit Python. Er war Entwickler, Systemadministrator, arbeitete in der Qualitätssicherung und im technischen Support. All diese Aufgaben boten Lohn und Herausforderung, doch seine herausforderndste und lohnenswerteste Aufgabe ist sein Dasein als Ehemann und Vater.

**Ted Leung** programmiert seit 2002 mit Python und ist zurzeit »Principal Software Engineer« bei Sun Microsystems. Er hat einen Bachelor of Science in Mathematik des Massachusetts Institute of Technology und einen Master of Science in Informatik von der Brown University. Teds Weblog finden Sie unter <http://www.sauria.com/blog>.

**Bill Mietelski** ist seit 20 Jahren IT-Geek. Aktuell ist er Software-Entwickler bei einer führenden amerikanischen medizinischen Forschungseinrichtung in der Gegend von Chicago und arbeitet an statistischen Studien. Wenn er nicht in seinem Büro sitzt oder anderswo an einen Computer gefesselt ist, finden Sie ihn wahrscheinlich auf dem Golfplatz, wo er kleine weiße Bälle jagt.