O'REILLY®

Aktuell zu Python 3
2. Auflage

Python

von Kopf bis Fuß

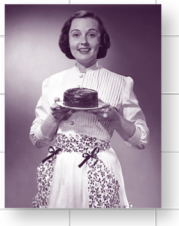
Laden Sie sich die wesentlichen Python-Kenntnisse direkt ins Hirn

Modellieren Sie Daten als Listen, Tupel, Sets und Dictionaries

Vermeiden Sie saure Überraschungen: Benutzen Sie die DB-API

Objekte? Dekoratoren? Generatoren? Bitteschön!

Erstellen Sie eine moderne Web-App mit Flask



Verwenden Sie Module, um Ihren Code mit anderen zu teilen

Paul Barry

Deutsche Übersetzung von Jørgen W. Lang

Python von Kopf bis Fuß

Zweite Auflage

Paul Barry

Deutsche Übersetzung von Jørgen W. Lang

O'REILLY®

Beijing • Boston • Farnham • Sebastopol • Tokyo

Paul Barry

Lektorat: Alexandra Follenius
Übersetzung: Jørgen W. Lang
Korrektorat: Sibylle Feldmann, www.richtiger-text.de
Satz: Ulrich Borstelmann, www.borstelmann.de
Herstellung: Susanne Bröckelmann
Umschlaggestaltung: Michael Oréal, www.oreal.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 978-3-96009-035-9
PDF 978-3-96010-135-2
ePub 978-3-96010-136-9
mobi 978-3-96010-137-6

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.

2. Auflage 2017

5 4 3 2 1 0

Ich widme dieses Buch auch weiterhin den großzügigen Menschen in der Python-Gemeinschaft, die daran mitgewirkt haben, Python zu dem zu machen, was es heute ist.

Und denen, die das Lernen von Python und seinen Technologien immerhin so komplex gemacht haben, dass ein Buch wie *dieses* beim Erlernen eine Hilfe sein kann.

Über den Autor von Python von Kopf bis Fuß, 2. Auflage

Bei einem Spaziergang macht Paul eine Pause, um mit seiner langmütigen Frau zu diskutieren, wie man >>Tupel<< ausspricht.

Und so reagiert Deirdre normalerweise darauf. ☺

Paul Barry lebt und arbeitet in *Carlow*, *Irland*, einer kleinen Stadt mit etwa 35.000 Einwohnern ca. 80 km südwestlich der Hauptstadt *Dublin*.

Paul hat einen Titel als *B.Sc. in Information Systems* sowie als *M.Sc. in Computing*. Außerdem besitzt er eine Lehrerlaubnis im Bereich *Learning and Teaching*.

Paul arbeitet seit 1995 am *Institute of Technology, Carlow* und hält dort seit 1997 Vorlesungen. Vor seiner Lehrtätigkeit verbrachte Paul ein Jahrzehnt in der IT-Industrie und arbeitete in Irland und Kanada, meistens im Gesundheitswesen. Paul ist mit Deirdre verheiratet, und gemeinsam haben sie drei Kinder (Joseph, Aaron und Aideen, von denen zwei inzwischen selbst studieren).

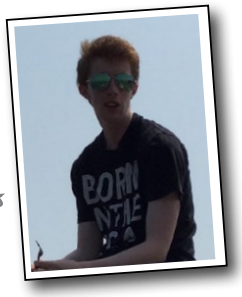
Die Programmiersprache Python und verwandte Technologien sind seit 2007 ein integraler Bestandteil von Pauls Vordiplom-Kursen.

Paul ist Autor (oder Koautor) von vier weiteren technischen Büchern: zwei über Python und zwei über Perl. In der Vergangenheit hat er als Redakteur für das *Linux Journal Magazine* eine Menge Artikel geschrieben.

Aufgewachsen ist Paul in *Belfast*, *Nordirland*, was seine Persönlichkeit und seinen lustigen Akzent erklärt (es sei denn, Sie stammen selbst »aus dem Norden«, und empfinden Pauls Wesen und Akzent als *vollkommen normal*).

Paul finden Sie auf *Twitter* (*@barrypj*) und auf seiner Website unter: *http://paulbarry.itcarlow.ie*.

Joseph



Aaron

Aideen

Der Inhalt (im Überblick)

Der Inhalt (jetzt ausführlich)

Einführung

Ihr Gehirn und Python. *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, wie man in Python programmiert?

Die Grundlagen

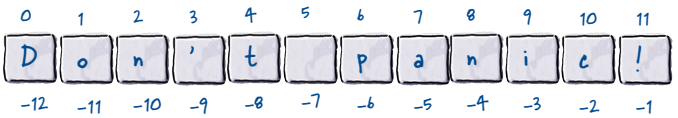
1 Volle Kraft voraus!

Finden Sie einen möglichst schnellen Einstieg in die Programmiersprache Python. In diesem Kapitel geben wir Ihnen einen Einblick in die Grundlagen der Python-Programmierung, und zwar wie es bei *Von Kopf bis Fuß*-Büchern üblich ist: indem wir gleich loslegen. Nach ein paar Seiten ist Ihr erstes Programm bereits lauffähig. Am Ende des Kapitels können Sie nicht nur Ihr Beispielprogramm ausführen, sondern auch den Code verstehen (und noch einiges mehr). Unterwegs lernen Sie ein paar der Dinge, die Python als Programmiersprache ausmachen. Lassen Sie uns also keine Zeit verschwenden. Blättern Sie um, und los geht's!

Listendaten

2 Mit geordneten Daten arbeiten

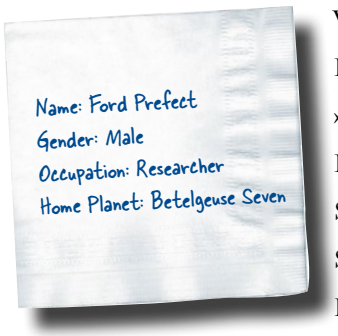
Alle Programme verarbeiten Daten, und Python-Programme sind da keine Ausnahme. Sehen Sie sich einmal um: *Daten sind überall*. Programmierung hat fast immer mit Daten zu tun: Daten *sammeln*, Daten *verarbeiten*, Daten *verstehen*. Um effektiv damit arbeiten zu können, brauchen Sie einen Ort, um Ihre Daten *abzulegen*. Dank einiger *äußerst vielseitiger* Datenstrukturen kann Python hier besonders punkten: **Listen**, **Dictionaries**, **Tupel** und **Sets**. In diesem Kapitel werden wir alle vier vorstellen, den Großteil des Kapitels werden wir uns jedoch eingehend mit **Listen** befassen. (Die anderen drei Strukturen werden wir in Kapitel 3 genauer betrachten.) Wir gehen absichtlich so früh auf diese Datenstrukturen ein, da die meisten Arbeiten in Python mit Daten zu tun haben.



Strukturierte Daten

3 Mit strukturierten Daten arbeiten

Pythons Listen sind eine großartige Datenstruktur, aber kein Allheilmittel. Wenn Sie mit *wirklich* strukturierten Daten arbeiten müssen (und eine Liste zum Speichern nicht geeignet ist), kann Pythons eingebaute **Dictionary**-(Wörterbuch-)Struktur möglicherweise helfen. Sammlungen von *Schlüssel/Wert-Paaren* lassen sich problemlos speichern und bearbeiten. Wir werden Pythons Dictionary-Struktur in diesem Kapitel eingehend untersuchen und Ihnen dabei auch gleich **Sets** und **Tupel** vorstellen. Gemeinsam mit Listen (bekannt aus dem vorigen Kapitel) bieten Dictionary, Set und Tupel eine Reihe vorgefertigter Datenstrukturen, die die Arbeit mit Daten in Python deutlich erleichtern.



Code wiederverwenden

4 Funktionen und Module

Die Wiederverwendbarkeit von Code ist Voraussetzung für ein wartbares System. Und bei Python ist Anfang und Ende aller Wiederverwendbarkeit die **Funktion**. Nehmen Sie ein paar Codezeilen, geben Sie ihnen einen Namen, und schon haben Sie eine (wiederverwendbare) Funktion. Nehmen Sie eine Sammlung von Funktionen und packen Sie sie in eine eigene Datei, und schon haben Sie ein **Modul** (das ebenfalls wiederverwendet werden kann). Es stimmt schon: *Teilen hilft*. Am Ende dieses Kapitels werden Sie wissen, wie Code mithilfe von Pythons Funktionen und Modulen **wiederverwendet** und **mit anderen geteilt** werden kann.

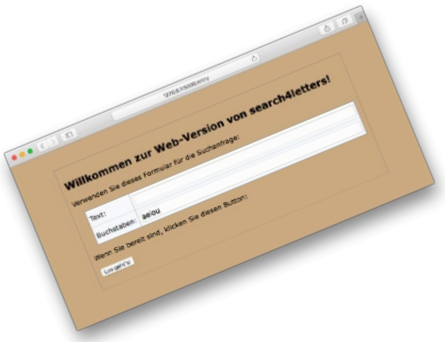
module

Eine Webapplikation erstellen

5 Auf ins wahre Leben!

Jetzt kennen Sie genug Python, um richtig loszulegen.

Nachdem Sie die ersten vier Kapitel dieses Buchs gemeistert haben, sind Sie in der Lage, Python in jeder Programmiersituation einzusetzen (obwohl es noch eine Menge zu lernen gibt). Anstatt die lange Liste verschiedener Programme abzuarbeiten, wollen wir uns in den folgenden Kapiteln mit der Erstellung einer webbasierten Applikation beschäftigen – einem Bereich, in dem Python besonders stark ist. Unterwegs werden Sie etwas mehr über Python lernen. Bevor wir loslegen, wollen wir Ihre bisherigen Python-Kenntnisse aber noch ein wenig auffrischen.



Daten speichern und bearbeiten

6 Wo kommen die Daten hin?

Früher oder später müssen Sie Ihre Daten irgendwo sicher speichern.

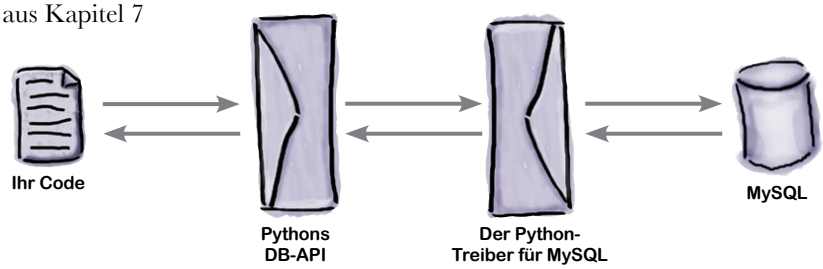
Und wenn es um **Datenspeicherung** geht, ist Python ganz für Sie da. In diesem Kapitel lernen Sie, wie man Daten in *Textdateien* speichert und sie wieder ausliest. Das erscheint einem als Speichermechanismus vielleicht ein wenig simpel, wird aber in vielen Problembereichen eingesetzt. Neben dem Speichern und Auslesen von Daten aus Dateien werden Sie außerdem noch ein paar Tricks zum Bearbeiten von Daten erfahren. Das »ernste Zeug« (Daten in einer Datenbank speichern) heben wir uns für das folgende Kapitel auf, trotzdem wird uns die Arbeit mit Dateien jetzt schon ganz ordentlich auf Trab halten.

Formulardaten	IP-Adresse	Browser	Ergebnisse
ImmutableMultiDict([('phrase', 'hitch-hiker'), ('letters', 'aeiou')])	127.0.0.1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36	{'e', 'i'}

Datenbanken benutzen

7 Die DB-API von Python verwenden

Datenspeicherung in einer relationalen Datenbank ist praktisch. In diesem Kapitel lernen Sie, Code für die Zusammenarbeit mit der beliebten **MySQL**-Datenbank zu schreiben. Hierfür verwenden wir eine allgemeine Datenbankschnittstelle namens **DB-API**. Mit dieser API (Teil der Python-Standardinstallation) können Sie Code schreiben, der sich leicht zwischen verschiedenen Datenbankprodukten austauschen lässt – sofern die verwendete Datenbank SQL versteht. Wir werden MySQL als Datenbank benutzen, der DB-API-Code kann aber auch mit einer beliebigen anderen relationalen Datenbank verwendet werden. Zunächst wollen wir sehen, was für den Einsatz relationaler Datenbanken mit Python gebraucht wird. In diesem Kapitel gibt es eher wenig neuen Python-Code. Allerdings ist der Einsatz von Python für die Kommunikation mit Datenbanken ein **sehr wichtiges Thema**, über das man auf jeden Fall Bescheid wissen sollte.

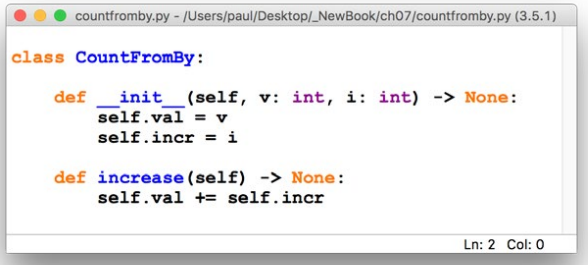


Ein bisschen Klasse

8 Verhalten und Zustand abstrahieren

Klassen speichern Verhalten und Zustand gemeinsam.

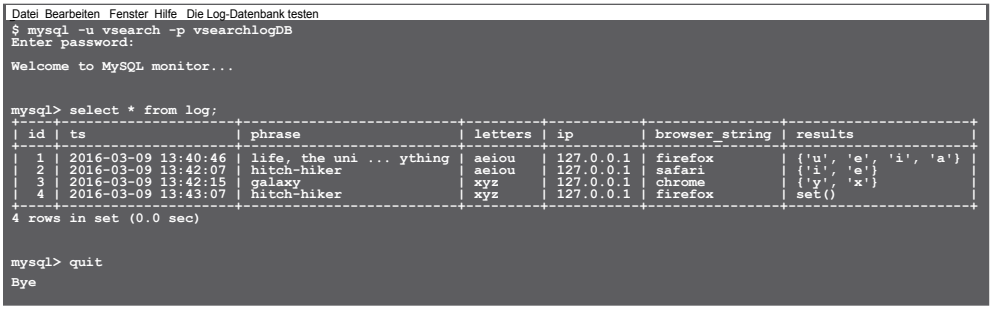
In diesem Kapitel legen wir die Webapplikation zur Seite und beschäftigen uns stattdessen mit Python-**Klassen**. Der Anlass ist, dass wir mithilfe einer Python-Klasse einen Kontextmanager erstellen wollen. Und weil die Erstellung und Programmierung von Klassen sowieso wissenswert ist, haben wir ihnen dieses Kapitel gewidmet. Sie werden hier nicht alles über Klassen lernen, aber wir wollen zumindest die Dinge ansprechen, die Sie brauchen, um für die Erstellung des Kontextmanagers Ihrer Webapplikation fit zu werden. Dann wollen wir mal schauen, worum es hier eigentlich geht.



Das Kontextmanagement-Protokoll

9 Sich in Pythons with-Anweisung einklinken

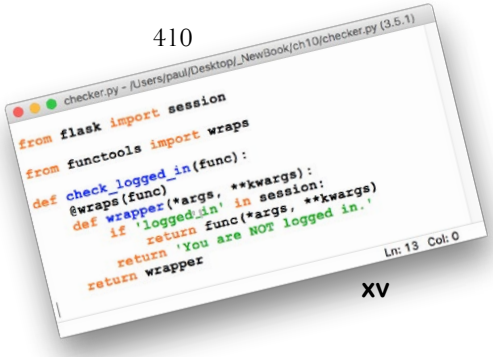
Es ist Zeit, das Gelernte anzuwenden. In Kapitel 7 haben wir über die Verwendung einer **relationalen Datenbank** mit Python gesprochen, während Kapitel 8 eine Einführung in Python-Klassen enthielt. Nun werden wir beide Techniken kombinieren, um einen **Kontextmanager** zu erstellen. Dieser soll die `with`-Anweisung so erweitern, dass sie auch mit relationalen Datenbanken funktioniert. In diesem Kapitel klinken Sie sich in die `with`-Anweisung ein, indem Sie eine neue Klasse erstellen, die mit Pythons **Kontextmanagement-Protokoll** konform ist.



Funktionsdekoratoren

10 Funktionen verpacken

Für die Erweiterung von Code gibt es neben dem Kontextmanagement-Protokoll auch noch andere Optionen. In Python können Sie auch Funktions**dekoratoren** benutzen. Damit können Sie Funktionscode erweitern, *ohne* diesen zu verändern. Auf den ersten Blick erscheint das wie schwarze Magie, aber keine Sorge: So schlimm ist es nicht. Trotzdem sehen viele Python-Programmierer das Schreiben von Funktionsdekoratoren als eher schwierig an. Dadurch werden Dekoratoren weniger eingesetzt als eigentlich sinnvoll. In diesem Kapitel wollen wir Ihnen zeigen, dass die Erstellung und Verwendung eigener Dekoratoren nicht so schwer ist, wie oft behauptet wird.



Mit Ausnahmen umgehen

11 Was zu tun ist, wenn mal etwas schiefgeht

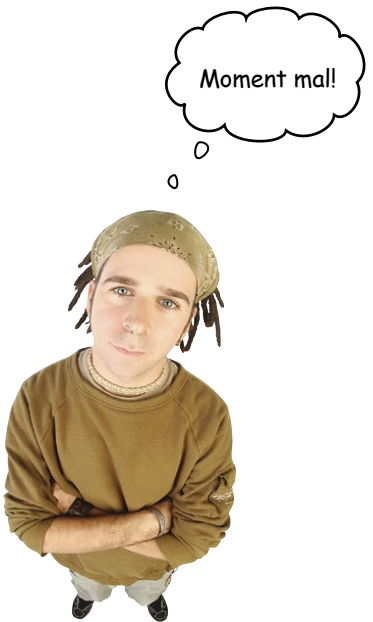
Egal wie gut Ihr Code ist, irgendetwas geht immer schief. Sie haben alle Beispiele in diesem Buch erfolgreich durchgearbeitet und sind sich ziemlich sicher, dass der bisherige Code auch funktioniert. Aber ist der deshalb wirklich robust? Vermutlich nicht. Es ist (bestenfalls) naiv, zu glauben, man könne Code schreiben und es würde schon alles gut gehen. Vorsicht ist hier deutlich besser als Vertrauensseligkeit. Wenn Ihr Code auch dann noch funktionieren soll, wenn die Dinge aus dem Ruder laufen, müssen Sie sorgfältig vorgehen. In diesem Kapitel zeigen wir nicht nur, was alles schiefgehen kann, sondern auch, was in solchen Fällen (oder oft sogar davor) zu tun ist.


```
   ...
Exception
      +-- StopIteration
      +-- StopAsyncIteration
      +-- ArithmeticError
      |     +-- FloatingPointError
      |     +-- OverflowError
      |     +-- ZeroDivisionError
      +-- AssertionError
      +-- AttributeError
      +-- BufferError
      +-- EOFError
            ...
```

Ein bisschen Threading

11 3/4 Taten statt Warten

Manchmal braucht Code wirklich ziemlich lang für die Ausführung. Abhängig davon, wem das auffällt, kann das ein Problem sein oder nicht. Wenn Ihr Code »hinter den Kulissen« 30 Sekunden für die Ausführung braucht, ist die Wartezeit vermutlich kein Thema. Wenn aber ein Benutzer 30 Sekunden auf eine Antwort Ihrer Applikation wartet, merkt das jeder. Die Lösung hängt davon ab, welche Aufgabe Ihre Applikation hat (und wer warten muss). In diesem kurzen Kapitel zeigen wir ein paar mögliche Optionen und sehen uns dann eine Lösung für das tatsächliche Problem an: *Was passiert, wenn etwas sehr lange dauert?*



Fortgeschrittene Iteration

12 Schleifen wie verrückt

Es ist erstaunlich, wie viel Zeit unsere Programme in Schleifen verbringen.

Das überrascht nicht, weil die meisten Programme die gleiche Aufgabe viele Male durchführen müssen. Für die Optimierung von Schleifen gibt es zwei Ansätze: 1. die Verbesserung der Schleifensyntax (um die Definition einer Schleife zu erleichtern) und 2. die Verbesserung der Schleifenausführung (um sie schneller zu machen). Als Python 2 noch jung war (also vor *sehr, sehr* langer Zeit), haben die Designer der Sprache ein Merkmal entwickelt, das beide Ansätze implementiert. Es nennt sich **Comprehension** (eine Form der Abstraktion). Lassen Sie sich durch den seltsamen Namen nicht täuschen. Am Ende dieses Kapitels werden Sie sich fragen, wie Sie bisher ohne Comprehensions leben konnten.

Installation

A

Python installieren

Das Wichtigste zuerst: Wir wollen Python auf Ihrem Computer installieren. Egal ob Sie *Windows*, *macOS* oder *Linux* im Einsatz haben, Python ist für Sie da. Wie das im Einzelnen geht, hängt vom verwendeten Betriebssystem ab (ja, wissen wir – eine Riesenüberraschung, oder?). Die Python-Community arbeitet hart daran, für alle beliebten Plattformen einen Installer bereitzustellen. In diesem kurzen Anhang zeigen wir Ihnen, wie Sie Python auf Ihrem Computer installieren können.

PythonAnywhere

B

Ihre Webapplikation bereitstellen

Am Ende von Kapitel 5 haben wir behauptet, dass die Bereitstellung Ihrer Webapplikation in der Cloud nur zehn Minuten dauert.

Das Versprechen wollen wir jetzt einlösen. In diesem Anhang zeigen wir Ihnen, wie Sie Ihre Webapplikation über *PythonAnywhere* bereitstellen können – und zwar in nur zehn Minuten. *PythonAnywhere* ist bei Python-Programmierern sehr beliebt, und der Grund liegt auf der Hand: Die Plattform funktioniert exakt wie erwartet, bietet großartige Unterstützung für Python (und Flask) – und das Beste daran: Sie können Ihre Webapplikation dort kostenlos unterstellen. Sehen wir uns *PythonAnywhere* einmal genauer an.

Die 10 wichtigsten Dinge, die wir nicht behandelt haben

Es gibt immer noch etwas zu lernen

Wir hatten nie die Absicht, alles zu behandeln.

Das Ziel dieses Buchs war, Ihnen genug Python zu zeigen, um Sie möglichst schnell auf den Weg zu bringen. Wir hätten deutlich mehr behandeln können, haben wir aber nicht. Hier besprechen wir die zehn wichtigsten Dinge, die wir – auf vielleicht weiteren 600 Seiten – auch noch hätten besprechen können. Vermutlich werden Sie nicht alle davon interessieren. Trotzdem sollten Sie zumindest kurz duchblättern, um zu sehen, ob nicht doch etwas Ihren Geschmack trifft oder eine drängende Frage beantwortet. Alle in diesem Anhang besprochenen Programmiertechniken sind direkter Bestandteil von Python und dem Interpreter.

Die 10 wichtigsten Projekte, die wir nicht behandelt haben

Noch mehr Werkzeuge, Bibliotheken und Module

Wir wissen, was Sie beim Lesen der Überschrift gedacht haben.

Warum um alles in der Welt haben die den letzten Anhang nicht einfach *Die 20 wichtigsten Dinge, die wir nicht behandelt haben* genannt? Warum noch mal zehn? Im vorigen Anhang haben wir uns auf Dinge beschränkt, die direkter Bestandteil von Python sind (Teile der »beiliegenden Batterien« sozusagen). In diesem Anhang werfen wir das Netz weiter aus und sprechen über eine Reihe von Technologien, die es gibt, *weil* es Python gibt. Hier ist viel Gutes zu entdecken, und wie beim vorigen Kapitel lohnt sich selbst das Überfliegen *auf jeden Fall*.

E

Mitmachen

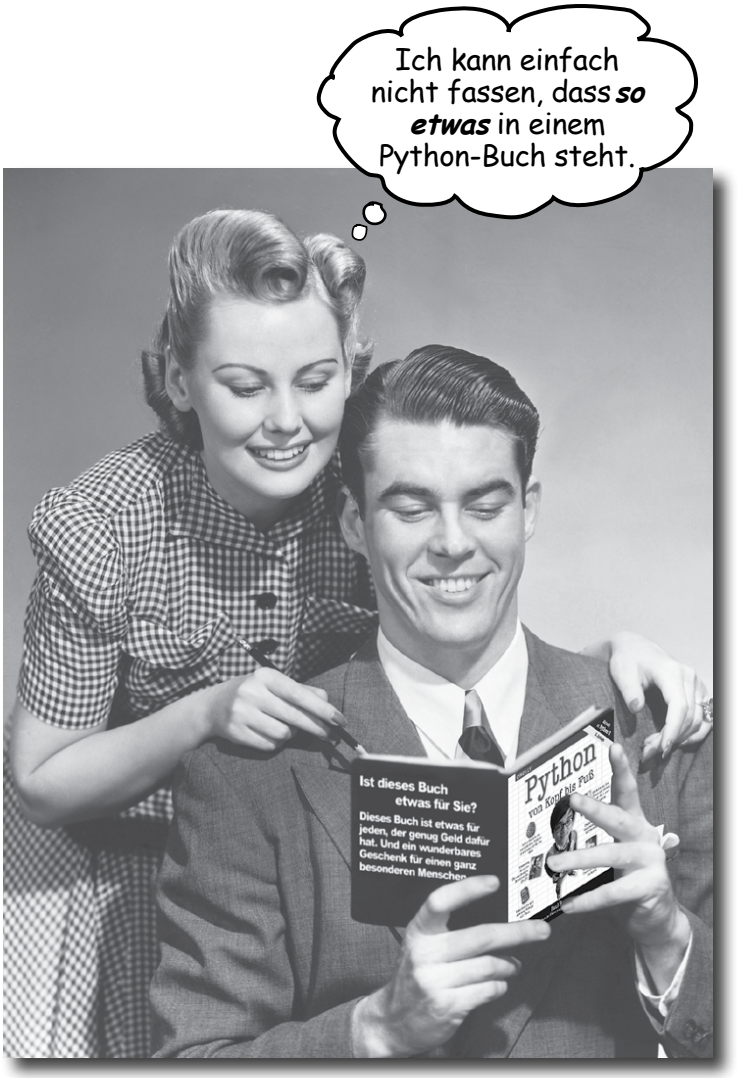
Die Python-Gemeinschaft

Python ist viel mehr als nur eine großartige Programmiersprache.

Es ist auch eine großartige Gemeinschaft. Die Python-Gemeinschaft ist einladend, verschiedenartig, offen, freundlich, selbstlos und freigiebig. Wir wundern uns die ganze Zeit, warum das bis heute noch niemand auf eine Grußkarte gedruckt hat! Im Ernst, Python ist mehr als nur die Sprache. Um Python ist ein komplettes Ökosystem entstanden – in Form von Büchern, Blogs, Websites, Konferenzen, Treffen, Benutzergruppen und Persönlichkeiten. In diesem Anhang untersuchen wir die Python-Gemeinschaft und sehen, was sie zu bieten hat. Sitzen Sie nicht nur rum und programmieren im stillen Kämmerlein: **Machen Sie mit!**

Wie man dieses Buch benutzt

Einführung



In diesem Abschnitt beantworten wir die brennende Frage:
»Und? Warum steht SO WAS in einem Python-Buch?«

Für wen ist dieses Buch?

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

1. Wissen Sie schon, wie man in einer anderen Programmiersprache programmiert?
2. Wünschen Sie sich, Python wäre Teil der Werkzeuge, mit denen Sie Neues schaffen können?
3. Ist es Ihnen lieber, etwas in Angriff zu nehmen und das Gelernte dann anzuwenden, als sich stundenlang in einer Vorlesung zu langweilen?

… ist dieses Buch etwas für Sie.

Wer sollte eher Abstand von diesem Buch nehmen?

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

1. Wissen Sie schon die meisten wichtigen Dinge zur Programmierung mit Python?
2. Suchen Sie nach einem Referenzbuch zu Python, das alle Details minutiös aufdröselt?
3. Sollen lieber 15 schreiende Affen Ihre Zehennägel herausreißen, als dass Sie etwas Neues lernen? Glauben Sie, ein Python-Buch sollte *alles* abdecken und seine Leser dabei zu Tränen langweilen, weil mehr ja besser ist?

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

Dies ist kein Referenzwerk, und wir gehen davon aus, dass Sie bereits programmiert haben.

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

Wir wissen, was Sie gerade denken

»Kann *das* wirklich ein seriöses Python-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:

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 beziehen, 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. 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?

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, 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.

Wir ziehen die Aufmerksamkeit des Lesers auf den Lernstoff – nachhaltig. 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 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 Python 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 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, Python 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.***

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.

Wir haben ***mehrere unterschiedliche Lernstile*** eingesetzt, denn vielleicht bevorzugen *Sie* ein Schritt-für-Schritt-Vorgehen, während ein anderer 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.*

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 sich selbst Notizen.**
   Wir haben sie entworfen, aber wenn wir sie auch für Sie lösen würden, wäre dass, 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 wollen 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, 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.