

O'REILLY®

Python

von Kopf bis Fuß

Grundlagen und Praxis der
Python-Programmierung

Paul Barry

Übersetzung von Jørgen W. Lang



3. Auflage
Verwendet Jupyter Notebook



Ein gehirnfrendliches Buch

Paul Barry

Lektorat: Ariane Hesse

Übersetzung: Jörgen W. Lang

Copy-Editing: Sibylle Feldmann, www.richtiger-text.de

Satz: Ulrich Borstelmann, www.borstelmann.de

Herstellung: Stefanie Weidner

Umschlaggestaltung: Ellie Volckhausen, Susan Thompson, Michael Oréal, www.oreal.de

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 978-3-96009-239-1

PDF 978-3-96010-844-3

ePub 978-3-96010-845-0

3., erweiterte und aktualisierte Auflage 2024

Translation Copyright für die deutschsprachige Ausgabe © 2024 by dpunkt.verlag GmbH

Wieblinger Weg 17

69123 Heidelberg

Authorized German translation of the English edition of *Head First Python 3rd Edition*, ISBN 9781492051299 © 2023 Paul Barry.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

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.

Schreiben Sie uns:

Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: komentar@oreilly.de.

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardwarebezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag und Übersetzer können jedoch für Schäden haftbar gemacht werden, die im Zusammenhang mit der Verwendung dieses Buchs stehen.

Copyright und Urheberrechte:

Die durch die dpunkt.verlag GmbH vertriebenen digitalen Inhalte sind urheberrechtlich geschützt. Der Nutzer verpflichtet sich, die Urheberrechte anzuerkennen und einzuhalten. Es werden keine Urheber-, Nutzungs- und sonstigen Schutzrechte an den Inhalten auf den Nutzer übertragen. Der Nutzer ist nur berechtigt, den abgerufenen Inhalt zu eigenen Zwecken zu nutzen. Er ist nicht berechtigt, den Inhalt im Internet, in Intranets, in Extranets oder sonst wie Dritten zur Verwertung zur Verfügung zu stellen. Eine öffentliche Wiedergabe oder sonstige Weiterveröffentlichung und eine gewerbliche Vervielfältigung der Inhalte wird ausdrücklich ausgeschlossen. Der Nutzer darf Urheberrechtsvermerke, Markenzeichen und andere Rechtsvorbehalte im abgerufenen Inhalt nicht entfernen.

Wieder einmal der Python-Community gewidmet.

Für all die großzügigen Menschen, die beständig daran arbeiten,
Python zu der wunderbaren Programmiersprache zu machen, die
sie heute ist. Ich danke euch.



← Paul Barry

Paul Barry lebt mit seiner Frau Deirdre in der irischen Kleinstadt Carlow etwa 80 Kilometer von der Hauptstadt Dublin entfernt. Ihre drei Kinder (Joseph, Aaron und Aideen) sind mittlerweile erwachsen und »flügge geworden«.

Paul arbeitet an der South East Technological University (SETU). Sein Hauptarbeitsplatz ist der Kilkenny Road Campus Carlow, wo er als Mitarbeiter der Informatikabteilung der Universität seine Vorlesungen hält. Paul unterrichtet schon seit *langer Zeit*. Dabei setzt er seit fast 15 Jahren Python in allen Klassengruppen ein.

Paul besitzt einen Master- und einen Bachelor-Abschluss in Informatik sowie Zusatzqualifikationen im Bereich Learning & Teaching. Er ist nie dazu gekommen, seinen Doktor zu machen, daher sollte man ihn auch nicht als »Professor« bezeichnen (obwohl er sich darüber freut, wenn das trotzdem geschieht).

Den größten Teil der 1980er- und 1990er-Jahre arbeitete Paul im IT-Sektor, und zwar hauptsächlich im kanadischen Gesundheitswesen. Er hat darüber hinaus weitere Bücher und – vor langer Zeit – Artikel für das *Linux Journal* geschrieben.

All das bedeutet, dass Paul (leider) *etwas in die Jahre gekommen ist*. Bitte sagen Sie's nicht weiter.

Der Inhalt (im Überblick)

	Intro	xxi
0	Warum Python? Ähnlich und doch anders	1
1	Eintauchen: Sprung ins kalte Wasser	43
2	Listen aus Zahlen: Listendaten verarbeiten	81
3	Listen von Dateien: Funktionen, Module und Dateien	127
4	Formatierte String-Literale: Tabellen aus Daten	177
5	Daten organisieren: Die richtige Datenstruktur	225
6	Eine Web-App erstellen: Webentwicklung	259
7	Bereitstellung: Code überall ausführen	317
8	Mit HTML arbeiten: Web-Scraping	349
9	Mit Daten arbeiten: Datenmanipulation	389
9 ^{1/2}	Mit Dataframes arbeiten: Tabellarische Daten	427
10	Datenbanken: Dinge ordnen	451
11	Listenabstraktionen: Datenbankintegrationen	507
12	Bereitstellung in neuem Licht: Der letzte Schliff	571
	Anhang: Die zehn wichtigsten Themen, die wir nicht behandelt haben	601
	Index	615

Der Inhalt (jetzt ausführlich)

Intro

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?

Wir wissen, was Sie denken	xxiii
READ ME	xxviii
Die neueste Python-Version installieren	xxx
Python allein ist nicht genug	xxxi
Konfigurieren Sie VS Code ganz nach Ihrem Geschmack	xxxii
Fügen Sie zwei notwendige Erweiterungen zu VS Code hinzu	xxxiii
Die Python-Unterstützung von VS Code ist auf dem neuesten Stand	xxxiv
Das Team der technischen Sachverständigen	xxxvi
Danksagungen	xxxvii

Warum Python?

O

Ähnlich und doch anders

Wie Sie vermutlich schon wissen, beginnt Python mit dem Zählen bei null.

Python hat eine Menge mit anderen Programmiersprachen **gemeinsam**. Es gibt **Variablen**, **Schleifen**, **Bedingungen**, **Funktionen** und so weiter. In diesem Eröffnungskapitel nehmen wir Sie mit auf eine **kleine Besichtigungstour**, die Ihnen einen **Überblick** über die Grundlagen von Python vermitteln soll. Das heißt, wir zeigen Ihnen die Sprache, aber ohne zu sehr ins Detail zu gehen. Sie werden lernen, mit Jupyter Notebook (innerhalb von VS Code) eigenen Code zu **schreiben** und **auszuführen**. Dabei werden Sie staunen, wie viel Programmierfunktionalität direkt in Python **eingebaut** ist. Das werden Sie **nutzen**, um verschiedene Aufgaben zu erledigen. Außerdem erfahren Sie, dass Python viele Konzepte mit anderen Programmiersprachen gemeinsam hat, diese aber **etwas anders** umsetzt. Verstehen Sie uns hier nicht falsch: Wir meinen hier die **guten** Unterschiede, nicht die *schlechten*. Lesen Sie weiter, um mehr zu erfahren.



Vorbereitungen, Code auszuführen	7
Vorbereitung für Ihre erste Begegnung mit Jupyter	8
Füllen wir den Notebook-Editor mit etwas Code	9
Drücken Sie Shift+Enter, um Ihren Code auszuführen	10
Was ist, wenn Sie mehr als eine Karte ziehen wollen?	15
Ein genauerer Blick auf den Code zum Ziehen einer Karte	17
Die »Großen Vier«: Liste, Tupel, Dictionary und Set	18
Den Kartenstapel mit einem Set modellieren	19
Der »print dir«-Combo-Mambo	20
Hilfe für die Ausgaben von dir	21
Das Set mit Karten füllen	22
Das fühlt sich wie ein Stapel Karten an	24
Was genau ist »card« eigentlich?	25
Suchen Sie etwas?	28
Kurze Pause für eine Bestandsaufnahme	29
Python besitzt eine umfangreiche Standardbibliothek	30
Mit Python schreiben Sie nur den Code, den Sie brauchen	34
Gerade als Sie dachten, Sie seien endlich fertig ...	41

1 Eintauchen

Sprung ins kalte Wasser

Eine neue Sprache lernt man am besten, indem man Code schreibt.

Und wenn Sie Code schreiben wollen, brauchen Sie ein **echtes** Problem, das es zu lösen gilt. Wie der Zufall es will, gibt es in diesem Kapitel eins. Hier beginnen Sie Ihre Karriere in der Applikationsentwicklung mit Python, indem Sie zusammen mit unserem freundlichen **Schwimmcoach** einen Sprung ins kalte Wasser wagen. Sie beginnen mit Pythons **Strings** und wie Sie sie nach Herzenslust **manipulieren** können. Unterwegs erstellen Sie eine Python-basierte Lösung für das Problem des Coachs. Außerdem erfahren Sie mehr über Pythons eingebaute **Listen**-Datenstruktur. Sie lernen, wie **Variablen** funktionieren und was Pythons **Fehlermeldungen** bedeuten, ohne dass Sie hierfür gleich einen Tauchschein brauchen. Und das alles, während wir ein *echtes* Problem mit *echtem* Python-Code lösen. Also, nichts wie rein – und zwar kopfüber!



Wie arbeitet der Coach im Moment?	45
Der Coach braucht eine bessere Stoppuhr	46
Bürogespräch	48
Die Datei und die Tabelle sind »verwandt«	51
Aufgabe 1: Daten aus dem Dateinamen extrahieren	52
Ein String ist ein Objekt mit Attributen	53
Daten des Schwimmers aus dem Dateinamen extrahieren	58
Versuchen Sie nicht, zu raten, was eine Methode tut ...	59
Einen String auftrennen (»splitten«)	60
Es gibt noch was zu tun	62
Lesen Sie Fehlermeldungen von unten nach oben	66
Vorsicht beim Kombinieren von Methodenaufrufen	67
Probieren wir es mit einer anderen String-Methode	69
Wir brauchen nur noch ein paar Variablen	72
Aufgabe Nummer 1 ist erledigt!	77
Aufgabe 2: Die Daten in der Datei verarbeiten	78

2

Listen aus Zahlen **Listendaten verarbeiten**

Je mehr Code Sie schreiben, desto besser werden Sie. Ganz einfach.

Auch in diesem Kapitel schreiben Sie Python-Code, um dem Coach zu helfen. Sie lernen, wie Sie Daten aus der **Datei** des Coachs **lesen** und die enthaltenen Zeilen in einer **Liste**, einer von Pythons eingebauten **Datenstrukturen**, speichern können. Neben der Erstellung von Listen aus Daten in einer Datei lernen Sie auch, Listen von Grund auf neu zu erstellen und bei Bedarf **dynamisch wachsen** zu lassen. Außerdem werden Sie Listen mit einer von Pythons beliebtesten Schleifenkonstrukten, der **for**-Schleife, verarbeiten. Sie werden Daten aus einem Datenformat in ein anderes **konvertieren**, und Sie werden einen neuen besten Freund (Ihren eigenen Python-**BFF**) kennenlernen. Nach Kaffee und Kuchen ist es jetzt Zeit, die Ärmel hochzukrempeln und sich wieder an die Arbeit zu machen.

Aufgabe 2: Die Daten in der Datei verarbeiten	82
Holen Sie sich eine Kopie der Daten des Coachs	83
Die open-BIF funktioniert mit Dateien	84
Datei mit with öffnen (und schließen)	85
Variablen werden bei Bedarf dynamisch erstellt	88
Eigentlich brauchen Sie die Daten in der Datei	89
Wir haben die Schwimmer-Daten aus der Datei	91
Der nächste Schritt kommt uns bekannt vor	94
Das vorherige Kapitel zahlt sich aus	97
Einen Zeitstring in einen Zeitwert umwandeln	98
Mit Python zu Hundertstelsekunden	100
Ein kurzer Rückblick auf Pythons for-Schleife	102
Jetzt geht's rund – for-Schleifen gegen while-Schleifen	105
Jetzt läuft es fast von selbst, und Sie machen große Fortschritte!	107
Wir behalten Kopien der konvertierten Werte	108
Eine Liste der Listenmethoden ausgeben	109
Es ist Zeit, den Durchschnitt zu berechnen	114
Den Durchschnittswert in einen Schwimmzeitstring umwandeln	115
Es ist Zeit, die Einzelteile zusammenzufügen	119
Aufgabe 2 hat (endlich) die Ziellinie überquert!	122

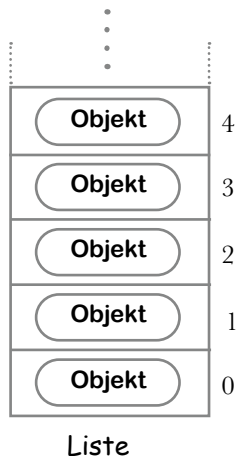


3 Listen von Dateien

Funktionen, Module und Dateien

Ihr Code kann nicht ewig in einem Notebook leben. Er will frei sein.

Und wenn es darum geht, Ihren Code zu befreien und mit anderen zu **teilen**, dann ist eine selbst erstellte **Funktion** der erste Schritt, auf den kurz darauf ein **Modul** folgt, mit dem Sie Ihren Code organisieren und weitergeben können. In diesem Kapitel werden Sie aus dem bisher geschriebenen Code direkt eine Funktion und auf dem Weg auch gleich ein **gemeinsam nutzbares** Modul erstellen. Ihr Modul wird sich sofort an die Arbeit machen, während Sie **for**-Schleifen, **if**-Anweisungen, Tests auf bestimmte **Bedingungen** sowie die Python-Standardbibliothek, **PSL** (*Python Standard Library*), verwenden, um die Schwimmdaten des Coachs zu verarbeiten. Außerdem werden Sie lernen, Ihre Funktionen zu **komentieren** (was *immer* eine gute Idee ist). Es gibt viel zu tun, also an die Arbeit!



Sie haben den nötigen Code schon fast beisammen	129
Eine Funktion in Python erstellen	130
Speichern Sie Ihren Code, so oft Sie wollen	131
Einfach den Code kopieren reicht nicht	132
Sämtlicher nötiger Code muss kopiert werden	133
Module verwenden, um Code weiterzugeben	140
Erfreuen Sie sich am Glanz der zurückgegebenen Daten	141
Funktionen geben bei Bedarf ein Tupel zurück	143
Holen wir uns eine Liste der Dateinamen des Coachs	149
Zeit für etwas Detektivarbeit ...	150
Was können Sie mit Listen anstellen?	151
Liegt das Problem bei Ihren Daten oder Ihrem Code?	159
Entscheidungen über Entscheidungen	163
Suchen wir den Doppelpunkt »in« dem String	164
Sind Sie auf 60 verarbeitete Dateien gekommen?	171
Der Code für den Coach nimmt langsam Form an ...	172

4

Formatierte String-Literale Tabellen aus Daten

Manchmal ist die einfachste Lösung die beste.

In diesem Kapitel kommen wir endlich dazu, die Balkendiagramme für den Coach zu erstellen. Hierfür werden Sie nichts als **Strings** verwenden. Wie Sie schon wissen, besitzt Python bereits eine Menge **eingebauter Funktionalität**. Python's **formatierte String-Literale**, auch *f-Strings* genannt, erweitern diese Möglichkeiten noch einmal auf ziemlich clevere Weise. Es klingt vielleicht seltsam, dass wir vorschlagen, Balkendiagramme mit Text zu erstellen, Sie werden aber sehr bald merken, dass es längst nicht so *absurd* ist, wie es scheint. Unterwegs werden Sie Python einsetzen, um **Dateien** zu erstellen und einen Webbrowser zu starten – und das alles mit nur wenigen Zeilen Code. Am Ende bekommt der Coach endlich, was er sich so lange gewünscht hat: die **automatische Erzeugung** von Diagrammen aus seinen Schwimmdaten. Also, nichts wie los!

Lizzie (Under 14) 100m Back



Average time: 1:29.20

Einfache Balkendiagramme mit HTML und SVG	182
Von einem einfachen Diagramm zum Balkendiagramm für den Coach	185
Die im HTML benötigten Strings mit Code erstellen	186
Die String-Verkettung skaliert nicht	189
f-Strings sind ein sehr beliebtes Python-Feature	194
Mit f-Strings ist die Erzeugung von SVG ein Kinderspiel!	195
Die Daten sind vollständig, oder nicht?	196
Sicherstellen, dass alle benötigten Daten zurückgegeben werden	197
Die Zahlen sind da, aber sind sie auch benutzbar?	198
Es fehlt nur noch das Ende der Webseite	207
Wie das Lesen aus Dateien klappt auch das Schreiben in Dateien völlig schmerzfrei	208
Es ist Zeit, Ihr Kunstwerk zu präsentieren	211
Jetzt sind nur noch zwei ästhetische Anpassungen nötig ...	212
Eine weitere selbst geschriebene Funktion	214
Erweitern wir das Modul um eine neue Funktion	215
Was ist mit dem Hundertstelwert los?	218
Runden ist nicht das Richtige (jedenfalls nicht in diesem Fall)	219
Es geht gut voran ...	221

5

Daten organisieren

Die richtige Datenstruktur

Ihr Code muss seine Daten im Arbeitsspeicher ablegen können.

Und bei der Anordnung von Daten **im Arbeitsspeicher** entscheidet oft die Wahl der Datenstruktur darüber, ob es eine unordentliche Lösung gibt, die *irgendwie* funktioniert, oder eine **elegante** Lösung, die *gut* funktioniert. In diesem Kapitel lernen Sie eine weitere Python-Datenstruktur kennen, das **Dictionary**. Es wird oft mit der allgegenwärtigen Liste kombiniert, um **komplexe** Datenstrukturen zu schaffen. Der Coach muss auf einfache Weise die Daten eines beliebigen Schwimmers auswählen können. Mit einem Dictionary wird das zum Kinderspiel!

⋮	⋮
Schlüssel#4	Objekt
Schlüssel#1	Objekt
Schlüssel#3	Objekt
Schlüssel#2	Objekt

Dictionary

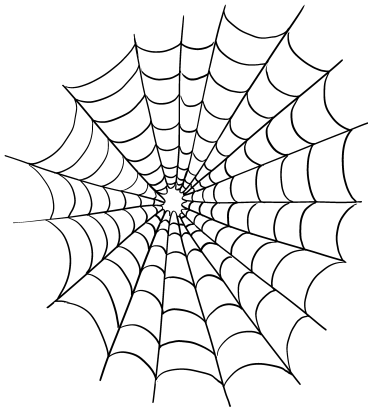
Eine Liste mit den Namen der Schwimmer erstellen	227
Der Liste-Set-Liste-Trick	229
Jetzt hat der Coach eine Liste mit Namen	231
Eine kleine Änderung macht einen »großen« Unterschied	232
Jedes Tupel ist einmalig	233
Superschnelle Lookups mit Dictionaries	236
Dictionaries verwenden Schlüssel/Wert-Paare für das Lookup	237
Anatomie eines Dictionary	240
Dictionaries sind für schnelle Lookups optimiert	248
Das gesamte Dictionary ausgeben	249
Das pprint-Modul erstellt einen »Pretty Print« Ihrer Daten	250
Das Dictionary mit den Listen ist leicht zu verarbeiten	251
Langsam nimmt die Sache Gestalt an	252

6 Eine Web-App erstellen

Webentwicklung

Fragen Sie zehn Programmierer, welches Web-Framework Sie nehmen sollen ...

... und Sie bekommen wahrscheinlich elf Antworten! 😊 Bei der Webentwicklung mangelt es Python wirklich nicht an technischen *Wahlmöglichkeiten*. Jede von ihnen hat eine eigene loyale und unterstützende Entwicklergemeinschaft. In diesem Kapitel tauchen Sie ein in die Welt der **Webentwicklung**. Sie werden schnell eine Web-App für den Coach bauen, in der man die Balkendiagramme für beliebige Schwimmer betrachten kann. Unterwegs lernen Sie die Verwendung von **HTML-Templates** (Schablonen), Funktions**dekoratoren**, **GET-** und **POST-HTTP-Methoden** und vieles mehr. Es gilt keine Zeit zu verschwenden: Der Coach will endlich sein neues System vorführen. Also an die Arbeit!



Flask aus dem PyPI installieren	261
Den Ordner für die Web-App vorbereiten	262
Bei der Arbeit mit Code haben Sie verschiedene Optionen	265
Anatomie einer Flask-Web-App	267
Schrittweiser Aufbau der Web-App ...	274
Was hat es mit diesem NameError auf sich?	279
Flask unterstützt Session-Verwaltung	281
Flasks Session-Verwaltung benutzt ein Dictionary	282
Den Code mit der »besseren Lösung« reparieren	285
Der Einsatz von Jinja2-Templates spart Zeit	291
base.html erweitern, um weitere Seiten zu erstellen	293
Drop-down-Menüs dynamisch erzeugen	296
Irgendwie müssen die Formulardaten verarbeitet werden	301
Die Formulardaten liegen in einem Dictionary vor	302
Für Funktionsparameter können Standardwerte angegeben werden	307
Standardparameterwerte sind optional	308
Die finale Version Ihres Code, Teil 1 von 2	309
Die finale Version Ihres Code, Teil 2 von 2	310
Für eine erste Web-App sieht das schon ganz gut aus	312
Das System des Coachs ist einsatzbereit	313

7 Bereitstellung

Code überall ausführen

Code auf dem eigenen Rechner auszuführen, ist eine Sache ...

Aber eigentlich wollen Sie Ihren Code so **bereitstellen**, dass auch andere Nutzer ihn einfach weiterverwenden können. Je *geringer* der Aufwand, desto besser. In diesem Kapitel werden Sie die Web-App des Coachs fertigstellen, sie mit etwas **Stil** versehen und sie dann in der **Cloud bereitstellen**. Trotzdem wird die Komplexität nicht ins Unermessliche steigen. In einer früheren Auflage dieses Buchs haben wir damit angegeben, dass die Bereitstellung »etwa zehn Minuten« dauert, und das ist bereits ziemlich schnell ... In diesem Kapitel werden Sie für die Bereitstellung nur noch zehn **Schritte** benötigen. Die Cloud wartet schon darauf, die Web-App des Coachs zu hosten. Zeit für die Bereitstellung!



Etwas stimmt immer noch nicht ganz	325
Jinja2 führt den Code zwischen {{ und }} aus	330
Zehn Schritte zur Cloud-Bereitstellung	332
Ein Beginner-Account reicht völlig aus	333
Niemand hält Sie davon ab, einfach loszulegen ...	334
Im Zweifel nutzen Sie die Standardeinstellungen	335
Die Platzhalter-Web-App macht noch nicht viel	336
Eigenen Code auf PythonAnywhere bereitstellen	337
Packen Sie Ihren Code in der Konsole aus	338
Konfigurieren Sie den Web-Tab, damit er auf Ihren Code verweist	339
Die WSGI-Dateien der Web-App anpassen	340
Ihre in der Cloud gehostete Web-App ist bereit!	344

8

Mit HTML arbeiten Web-Scraping

In einer perfekten Welt wären alle benötigten Daten leicht zugänglich.

Das ist aber nur selten der Fall. So werden Daten beispielsweise im Web veröffentlicht. In HTML eingebettete Daten müssen von Webbrowsern **gerendert** und von Menschen **gelesen** werden können. Was aber, wenn Sie diese Daten mit Code **verarbeiten** müssen? Geht das überhaupt? Glücklicherweise ist Python so etwas wie ein Champion, wenn es um das maschinelle Auslesen – das sogenannte **Scraping** – von Daten aus Webseiten geht, und in diesem Kapitel werden wir Ihnen zeigen, wie das funktioniert. Sie werden außerdem lernen, wie die ausgelesenen HTML-Seiten **geparst** werden, um nutzbare Daten zu **extrahieren**. Unterwegs werden Ihnen **Slices** (»Scheiben«) und **Soup** (»Suppe«) begegnen, aber keine Sorge, das hier ist immer noch *Python von Kopf bis Fuß* und nicht *Kochen von Kopf bis Fuß*.



Der Coach braucht mehr Daten	350
Machen Sie sich vor dem Scrapen mit den Daten vertraut	352
Wir brauchen einen Aktionsplan ...	353
Schrittweise Anleitung zum Web-Scraping	354
Zeit für etwas Web-Scraping-Technologie	356
Das rohe HTML-Markup von Wikipedia auslesen	359
Die ausgelesenen Daten untersuchen	360
Slices können aus beliebigen Folgen herausgeschnitten werden	362
Anatomie der Slices, Teil 1 von 3	363
Anatomie der Slices, Teil 2 von 3	364
Anatomie der Slices, Teil 3 von 3	365
Zeit für etwas HTML-Parsing-Power	370
Die »Suppe« nach interessanten Tags durchsuchen	371
Die zurückgegebene »Suppe« ist ebenfalls durchsuchbar	372
Welche Tabelle enthält die gesuchten Daten?	375
Vier große Tabellen und vier Gruppen mit Weltrekorden	377
Jetzt können wir die Daten auslesen	378
Daten aus allen Tabellen extrahieren, Teil 1 von 2	382
Daten aus allen Tabellen extrahieren, Teil 2 von 2	383
Die verschachtelte Schleife war die Lösung!	386

9 Mit Daten arbeiten

Datenmanipulation

Manchmal sind die Daten nicht so angeordnet, wie sie gebraucht werden.

Vielleicht haben Sie eine *verschachtelte Liste* (*List of Lists*), brauchen aber eigentlich ein *verschachteltes Dictionary*. Oder vielleicht müssen Sie einen Wert in einer Datenstruktur mit einem Wert in einer anderen Datenstruktur verbinden, ohne dass sie richtig zusammenpassen. Das kann schnell ziemlich frustrierend werden. Aber keine Sorge: Die Macht von Python steht Ihnen auch hierbei zur Seite. In diesem Kapitel benutzen Sie Python, um Ihre Daten **durch die Mangel zu drehen**. Das Ziel ist es, die von der *Wikipedia*-Website ausgelesenen Daten vom Ende des vorherigen Kapitels in etwas wirklich *Nützliches* umzuwandeln. Sie werden lernen, wie Sie Pythons Dictionary als **Lookup-Tabelle nutzen** können. Hier geht es um Umwandlungen, Integrationen, Aktualisierungen, Bereitstellungen und mehr. Und ganz am Ende wird die Spezifikation, die der Coach auf der Papierserviette notiert hat, *Wahrheit*. Wetten, Sie können es kaum abwarten? Wir auch nicht ... also los!



Daten Ihrem Willen unterwerfen ...	390
Jetzt haben Sie die nötigen Daten ...	394
Wenden Sie Ihr Wissen an!	396
Haben wir zu viele Daten?	399
Die Staffeldaten ausfiltern	400
Nun können wir unsere Balkendiagramme aktualisieren	401
Python besitzt eine eingebaute JSON-Bibliothek	403
JSON ist textbasiert, aber nicht schön	404
Weiter mit der Web-App-Integration	408
Eine Anpassung und ein Copy-and-paste-Vorgang reichen	409
Die Weltrekorde zum Balkendiagramm hinzufügen	410
Ist Ihre neueste Version der Web-App bereit?	414
PythonAnywhere ist für Sie da ...	418
Auch das Hilfsprogramm muss hochgeladen werden	419
Die neueste Version der Web-App bei PythonAnywhere bereitstellen	420
Den neuesten Code auf PythonAnywhere ausführen	421
Hilfsprogramme vor der Bereitstellung testen	422
Die Aufgabe täglich um 1:00 Uhr morgens ausführen	423

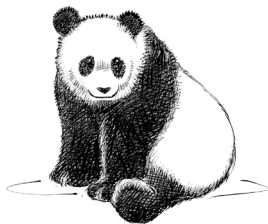
Mit ~~Elefanten~~ Dataframes arbeiten

9^{1/2} **Tabellarische Daten**

Manchmal wollen anscheinend alle Daten der Welt tabellarisch sein.

Tabellarische Daten gibt es *überall*. Die Schwimmweltrekorde aus dem vorherigen Kapitel liegen als **tabellarische** Daten vor. Wenn Sie alt genug sind, um sich an Telefonbücher zu erinnern, wissen Sie, dass auch diese Daten tabellarisch sind. Kontoauszüge, Rechnungen, sogar Tabellenkalkulationen sind – Sie haben es natürlich geahnt – tabellarische Daten. In diesem *kurzen* Kapitel lernen Sie einige Dinge über **pandas**, die beliebteste Python-Bibliothek zur Datenanalyse. Leider werden wir hier nur die Spitze des Eisbergs betrachten können. Trotzdem werden Sie genug lernen, um das nächste Mal, wenn Sie mit tabellarischen Daten arbeiten müssen, die am häufigsten verwendete pandas-Datenstruktur nutzen zu können, den **Dataframe**.

Der Elefant im Raum ... oder ist es ein Panda?	428
Ein verschachteltes Dictionary mit pandas?	429
Halten Sie sich zunächst an die Konvention	430
Eine Liste mit pandas-Dataframes	431
Spalten aus einem Dataframe auswählen	432
Dataframe zu Dictionary, erster Versuch	433
Unnötige Daten aus einem Dataframe entfernen	434
Den pandas-Bedingungsausdruck verneinen	435
Dataframe zu Dictionary, zweiter Versuch	436
Dataframe zu Dictionary, dritter Versuch	437
Noch ein verschachteltes Dictionary	438
Vergleich zwischen gazpacho und pandas	442
Dies war nur ein winziger Einblick ...	448



10 Datenbanken

Dinge ordnen

Irgendwann müssen Sie die Daten Ihrer Applikationen verwalten.

Und wenn Sie Ihre Daten besser **verwalten** müssen, reicht Python (für sich genommen) möglicherweise nicht aus. In solchen Fällen hilft der Griff zur **Datenbank**-Engine Ihrer Wahl. Um nicht den Überblick zu verlieren, beschränken wir uns an dieser Stelle auf Datenbank-Engines, die das gute alte **SQL** unterstützen. Sie werden hier nicht nur eine Datenbank **erstellen** und darin ein paar **Tabellen** anlegen, Sie werden auch Daten zur Datenbank **hinzufügen**, **daraus auswählen** und **löschen**. Für alle diese Aufgaben werden Sie **SQL**-Abfragen verwenden, die von Ihrem Python-Code koordiniert werden.



Der Coach hat sich gemeldet ...	452
Planung zahlt sich aus ...	455
Schritt 1: Eine Datenbankstruktur festlegen	457
Serviettenstruktur und -daten	459
Das DBcm-Modul von PyPI installieren	460
Einstieg in DBcm und SQLite	461
DBcm und »with« als Team	462
Benutzen Sie dreifach doppelte Anführungszeichen für Ihren SQL-Code	464
Nicht jede SQL-Anweisung gibt etwas zurück	466
Ihre Tabellen sind bereit (und Schritt 1 ist erledigt)	471
Welche Schwimmer-Dateien brauchen wir?	472
Schritt 2: Die Datenbanktabelle mit Inhalt füllen	473
Sicherheit durch Pythons SQL-Platzhalter	475
Wiederholen wir dieses Vorgehen für die Ereignisse	490
Jetzt fehlt nur noch die times-Tabelle ...	494
Die Zeiten stehen in den Schwimmer-Dateien ...	495
Ein Hilfsprogramm zur Datenbankaktualisierung, 1 von 2	501
Ein Hilfsprogramm zur Datenbankaktualisierung, 2 von 2	502
Schritt 2 ist (endlich) abgeschlossen	503

11 Listenabstraktionen Datenbankintegrationen

Zeit für die Integration Ihrer Datenbanktabellen.

Mithilfe der Tabellen in der Datenbank kann Ihre Web-App die vom Coach benötigte **Flexibilität** erreichen. In diesem Kapitel erstellen wir ein Modul mit verschiedenen **Hilfsfunktionen** (Utilities), über die Ihre Web-App die Datenbank-Engine **nutzen** kann. Außerdem werden Sie eine echte Python-Superkraft kennenlernen, die Ihnen hilft, mit weniger Code mehr zu erreichen: **Listenabstraktionen**. Daneben werden Sie eine Menge Ihres schon vorhandenen Codes auf neue und interessante Weise **wiederverwenden**. Wir haben jede Menge **Integration** vor uns. Also los!

Testen wir die Abfragen in einem neuen Notebook	510
Fünf Codezeilen werden zu einer	513
Combo-Mambo ohne Dunder	515
Eine Abfrage erledigt, drei fehlen noch ...	520
Zwei Abfragen erledigt, zwei fehlen noch ...	522
Zu guter Letzt, die letzte Abfrage ...	523
Der Code für die Datenbankhilfsfunktionen, Teil 1 von 2	528
Der Code für die Datenbankhilfsfunktionen, Teil 2 von 2	529
Wir sind fast bereit für die Datenbankintegration	532
Es ist Zeit, den Datenbankcode zu integrieren!	540
Was ist mit dem Template los?	548
Eine Liste der Ereignisse anzeigen ...	552
Jetzt brauchen wir nur noch ein Balkendiagramm ...	556
Überprüfung des aktuellen Codes in swimclub.py	558
Begegnung mit dem SVG-erzeugenden Jinja2-Template	560
Das Modul convert_utils	562
list zip ... wie bitte?!?	565
Ihre Datenbankintegrationen sind fertig!	567

```
[sql for sql in dir(queries) if not sql.startswith("_")]
```

12

Bereitstellung in neuem Licht

Der letzte Schliff

Das Ende vom Anfang Ihrer Python-Reise ist fast erreicht.

In diesem letzten Kapitel passen Sie Ihre Web-App so an, dass sie nicht mehr SQLite, sondern **MariaDB** als Datenbank-Backend verwendet. Danach gibt es noch ein paar kleinere Änderungen, um die neueste Version Ihrer Web-App auf PythonAnywhere bereitzustellen. Dadurch unterstützt das System des Coachs eine **beliebige** Zahl von Schwimmern, die an einer **beliebigen** Zahl von Trainingseinheiten teilnehmen. In diesem Kapitel gibt es zu Python selbst nicht viel Neues. Die meiste Zeit werden Sie damit verbringen, den schon vorhandenen Code für die Zusammenarbeit mit MariaDB und PythonAnywhere anzupassen. Ihr Python-Code existiert nie **isoliert**, sondern **inter-agiert** mit seiner Umgebung und dem System, auf dem er läuft.

```
mysql> select count(*) from events;
+-----+
| count(*) |
+-----+
|      14 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from swimmers;
+-----+
| count(*) |
+-----+
|      23 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from times;
+-----+
| count(*) |
+-----+
|     467 |
+-----+
1 row in set (0.00 sec)
```

Migration zu MariaDB	575
Die Daten des Coachs zu MariaDB umziehen	576
Drei Anpassungen für schema.sql	577
Die Tabellen wiederverwenden, Teil 2 von 2	578
Überprüfen, ob die Tabellen korrekt eingerichtet wurden	579
Die vorhandenen Daten zu MariaDB übertragen	580
Die Abfragen mit MariaDB kompatibel machen	582
Die Datenbankhilfsfunktionen müssen ebenfalls angepasst werden	583
Eine neue Datenbank auf PythonAnywhere anlegen	586
Das Dictionary mit Datenbankinformationen anpassen	587
Alles in die Cloud kopieren	588
Die Web-App mit dem neuesten Code aktualisieren	589
Nur noch wenige Schritte ...	590
Die Cloud-Datenbank mit Daten füllen	591
Zeit für eine PythonAnywhere-Probefahrt	592
Stimmt mit PythonAnywhere etwas nicht?	594
Der Coach ist überglücklich!	596

Anhang

Die zehn wichtigsten Dinge, die wir nicht behandelt haben

Wir sind fest davon überzeugt, dass man wissen muss, wann es reicht.

Besonders wenn Ihr Autor aus Irland stammt, einem Land, das dafür berühmt ist, Menschen mit der Gabe hervorzubringen, zu ignorieren, wann sie besser mit dem Reden aufhören sollten. 😊 Dabei genießen wir es, über unsere Lieblingsprogrammiersprache Python zu sprechen. In diesem Anhang zeigen wir Ihnen die zehn wichtigsten Themen, die wir Ihnen gerne »von Kopf bis Fuß« erzählt hätten. Leider hatten wir aber keine weiteren 400 Seiten zur Verfügung. Hier finden Sie Informationen zu Klassen, Exception-Handling, zum Testen eines Walrosses (Ernsthaft? Ein Walross? Ja, ein Walross), zu Switches, Dekoratoren, Kontextmanagern und Nebenläufigkeit sowie Tipps zu Typen, virtuellen Umgebungen und Programmierwerkzeugen. Wie gesagt: Es gibt immer etwas, über das man noch reden kann. Also, blättern Sie um – und haben Sie Freude an den nächsten zwölf Seiten!

1. Klassen	602
2. Ausnahmen (Exceptions)	605
3. Tests	606
4. Der Walross-Operator	607
5. Wo ist switch? Welcher switch?	608
6. Fortgeschrittene Sprachmerkmale	609
7. Nebenläufigkeit	610
8. Typhinweise	611
9. Virtuelle Umgebungen	612
10. Werkzeuge	613



Wie man dieses Buch benutzt

Das Intro



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

Für wen ist dieses Buch gedacht?

Wenn Sie *alle* folgenden Fragen mit »Ja« beantworten können ...

- 1 Wissen Sie bereits, wie man in anderen Sprachen programmiert?
- 2 Soll Python Teil Ihres Werkzeugkastens werden, damit Sie genauso viel Spaß haben können wie all die anderen Pythonistas?
- 3 Lernen Sie besser, indem Sie sich die Hände schmutzig machen, anstatt nur zuzuhören?

»Pythonista«:
Eine Person, die es
liebt, in Python zu
programmieren.

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

Wer sollte besser die Finger von diesem Buch lassen?

Wenn Sie mindestens eine dieser Fragen mit »Ja« beantworten können ...

- 1 Haben Sie noch nie programmiert und können *if*-Anweisungen nicht von *Schleifen* unterscheiden?
- 2 Sind Sie Python-Expertin oder -Experte und suchen nach einer kurzen Sprachreferenz?
- 3 Hassen Sie es, Neues zu lernen? Sind Sie davon überzeugt, dass kein noch so gutes Python-Buch Sie dazu bringen könnte, laut loszulachen oder aufzustöhnen, weil alles so albern ist? Lassen Sie sich lieber zu Tode langweilen?

... dann ist dieses Buch nichts für Sie.

[Hinweis vom Marketingteam:
Dieses Buch ist für alle Menschen
mit einer Kreditkarte geeignet.]



Wir wissen, was Sie denken

»Was hat *das* in einem ernsthaften Python-Buch zu suchen?«

»Was sollen die ganzen Abbildungen?«

»Kann ich auf diese Weise wirklich etwas *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 Ungewöhnliches. So ist es nun einmal gebaut, und es hilft Ihnen zu überleben.

Heutzutage ist es eher unwahrscheinlich, dass Sie als Tigerfutter enden. Aber man weiß ja nie.

Also, was macht Ihr Gehirn mit den normalen, langweiligen Routinesachen, die Ihnen so begegnen? Es tut *alles*, damit es dadurch nicht bei seiner *eigentlichen* Arbeit gestört wird: die *wirklich wichtigen* Dinge zu erfassen. Es gibt sich nicht damit ab, all das langweilige Zeug zu speichern, der »Das-hier-ist-offensichtlich-nicht-wichtig«-Filter lässt sie gar nicht erst durch.

Aber woher *weiß* Ihr Gehirn, was wichtig ist? Angenommen, Sie machen eine längere Wildniswanderung und ein Tiger springt aus dem Gebüsch. Was passiert in Ihrem Kopf?

Neuronen feuern. Gefühle kochen hoch. *Chemische Substanzen durchfluten Sie.*

Und so weiß Ihr Gehirn:

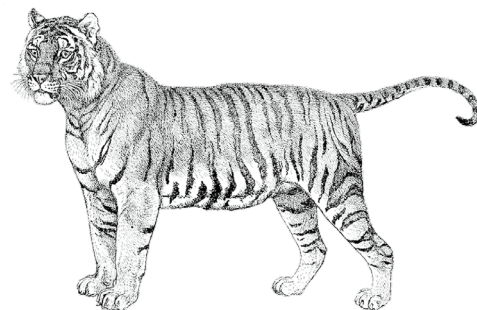
Das muss wichtig sein! Vergiss es nicht!

Und jetzt stellen Sie sich vor, Sie sind zu Hause oder in der Bibliothek. Sie befinden sich in einem sicheren, warmen, tigerfreien Bereich. Sie studieren, bereiten sich auf eine Prüfung vor. Oder Sie versuchen, irgendein schwieriges technisches 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 will sicherstellen, dass diese *offensichtlich unwichtigen* Inhalte keine knappen Ressourcen blockieren. Ressourcen, die besser dafür verwendet würden, die *wirklich wichtigen* Dinge zu speichern. Wie Tiger. Wie die Gefahren des Feuers. Oder dass Sie nie wieder in Shorts snowboarden sollten.

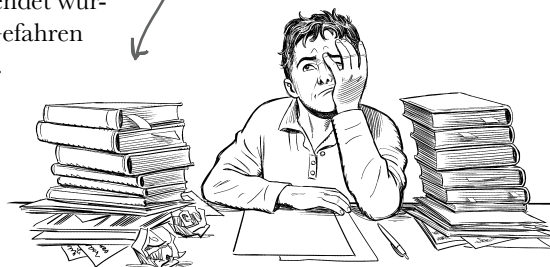
Leider gibt es kein Patentrezept, um Ihrem Gehirn zu sagen: »Hey, Gehirn, vielen Dank, aber egal, wie langweilig dieses Buch auch ist und wie klein der Ausschlag auf meiner emotionalen Richterskala gerade ist, ich will wirklich, dass du diesen Kram behältst.«

Ihr Gehirn denkt,
DAS HIER sei wichtig.



Ihr Gehirn
denkt, DAS
HIER brauche
es sich nicht
zu merken.

Klasse. Nur noch
638 öde, trockene
und langweilige
Seiten.



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 beziehen, an- statt 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, 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 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 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, 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.

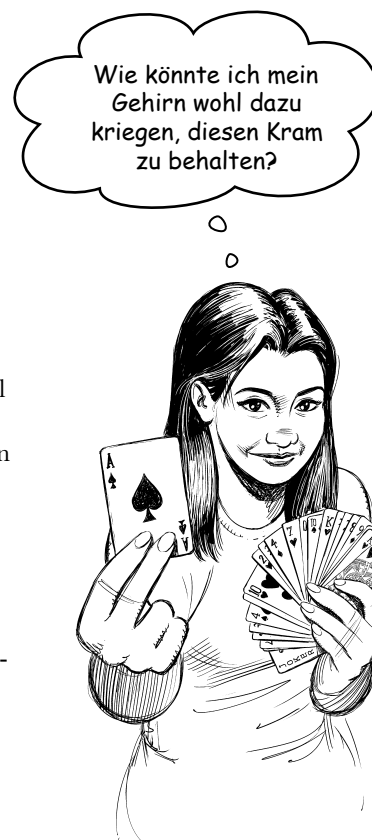
Wie bringen Sie also Ihr Gehirn dazu, Python für so wichtig zu halten wie einen Tiger?

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.

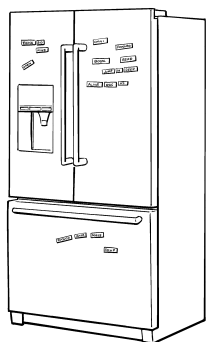
Wir haben **mehrere unterschiedliche Lernstile** eingesetzt, denn vielleicht bevorzugen *Sie* ein Schritt-für-Schritt-Vorgehen, während ein anderer erst einmal den groben Zusammenhang verstehen und ein Dritter einfach nur ein Codebeispiel sehen möchte. Aber ganz abgesehen von den jeweiligen Lernvorlieben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



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 sich selbst Notizen.

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

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

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

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

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

⑤ Trinken Sie Wasser. Viel.

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

⑥ Reden Sie drüber. Laut.

Sprechen aktiviert einen anderen Teil des Gehirns. Wenn Sie etwas verstehen 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 Zeichnungen. 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 findet zu großen Teilen beim Lösen der Übungen statt. 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.

READ ME

Dieses Buch ist eine Lernerfahrung, kein Referenzwerk. Wir haben absichtlich alles weggelassen, was dem Lernerfolg im Wege stehen könnte, egal woran Sie gerade im Buch arbeiten. Und nur beim ersten Durchlesen müssen Sie tatsächlich am Anfang beginnen, da das Buch später davon ausgeht, dass Sie bestimmte Dinge bereits gesehen und gelernt haben.

Dieses Buch soll Sie möglichst schnell auf den Weg bringen.

Wenn Sie etwas wissen müssen, bringen wir es Ihnen bei. Sie werden hier also keine langen Listen mit technischen Details finden, keine Tabellen mit Python-Operatoren oder deren Präzedenzregeln. Wir behandeln hier nicht *alles*. Trotzdem haben wir sehr hart daran gearbeitet, die wirklich wichtigen Dinge so gut wir können abzudecken, damit Sie Python schnell ins Hirn bekommen und es dort auch bleibt. Die einzige Voraussetzung ist, dass Sie bereits wissen, wie man mit einer beliebigen anderen Programmiersprache programmiert.

Sie können ohne Probleme eine beliebige Version von Python 3 verwenden.

Also gut, das ist vielleicht etwas zu stark vereinfacht. Sie brauchen *mindestens* Python 3.6. Aber diese Version ist bereits Ende 2016 erschienen. Es ist eher unwahrscheinlich, dass viele Menschen sich im Alltag noch darauf verlassen. Zu Ihrer Information: Als dieses Buch in Druck ging, stand Python 3.12 gerade vor der Tür.

Die Aktivitäten sind NICHT optional – Sie müssen die Arbeit selbst machen.

Die Übungen und Aktivitäten sind kein Beiwerk. Sie sind ein wesentlicher Bestandteil des Buchs. Einige sollen Ihnen beim Merken helfen, andere beim Verstehen und noch andere beim Anwenden. ***Überspringen Sie nicht die Übungen.***

Die Redundanz ist wichtig und beabsichtigt.

Ein wesentliches Anliegen eines *Von-Kopf-bis-Fuß*-Buchs ist, dass wir wollen, dass Sie es *wirklich* kapieren. Und wir wollen, dass Sie sich am Ende des Buchs an das Gelernte erinnern. Beständigkeit und Erinnern ist bei den meisten Referenzbüchern nicht das Ziel, aber in diesem Buch geht es ums *Lernen*. Deshalb werden Sie einige der hier gezeigten Konzepte mehr als einmal sehen.

Die Beispiele sind so kurz wie möglich.

Unsere Leser sagen uns, dass sie es frustrierend finden, sich durch 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 spezifisch für Lehrzwecke geschrieben und sind nicht immer rundum funktionsfähig (obwohl wir versucht haben, das so weit wie möglich sicherzustellen).

Wir haben den gesamten Code und die Ressourcen dieses Buchs ins Internet gestellt, damit Sie sie bei Bedarf nutzen können. Dennoch glauben wir, dass man viel lernen kann, wenn man den Code eintippt, vor allem, wenn man eine (für einen selbst) neue Programmiersprache lernt. Aber für die »Gib-mir-einfach-den-Code«-Typen da draußen, hier ist die GitHub-Seite zu diesem Buch:

<https://github.com/headfirstpython/third>

Anmerkung: Anders als in anderen Büchern der *Von-Kopf-bis-Fuß*-Reihe haben wir in diesem Buch darauf verzichtet, die Codebeispiele einzudeutschen. Das Fehlerrisiko ist einfach zu hoch. Außerdem werden Variablen, Funktionen usw. in der Praxis ohnehin so gut wie immer englisch benannt.

Bei den Kopf-Nuss-Übungen gibt es keine Antworten.

Für einige von ihnen gibt es einfach keine richtige Antwort. Und bei anderen besteht ein Teil der Lernerfahrung darin, dass Sie entscheiden, ob und wann Ihre Antworten richtig sind.

Nicht alle Probefahrt-Übungen haben Antworten.

Bei einigen Übungen bitten wir Sie lediglich, eine Reihe von Anweisungen zu befolgen. Sie haben die Möglichkeit zu überprüfen, ob das, was Sie getan haben, tatsächlich funktioniert hat. Im Gegensatz zu anderen Übungen gibt es hier aber keine richtigen Antworten.

Die neueste Python-Version installieren

Was hier zu tun ist, hängt davon ab, welches Betriebssystem Sie verwenden. Wir gehen davon aus, dass dies entweder *Windows*, *macOS* oder *Linux* ist. Die gute Nachricht ist, dass auf allen drei Plattformen das aktuelle Python lauffähig ist. Schlechte Nachrichten gibt es keine.

Wenn Python bei Ihnen bereits in Version 3 läuft, können Sie direkt umblättern – Sie sind bereit. Falls Sie Python noch nicht installiert haben oder eine ältere Version verwenden, wählen Sie den Absatz, der auf Sie zutrifft, und lesen Sie weiter.



Installation unter Windows

Die wunderbaren Python-Experten von Microsoft arbeiten hart, um sicherzustellen, dass die neueste Python-Version immer über den *Windows Store* erhältlich ist. Öffnen Sie den Store und suchen Sie nach »Python«, wählen Sie die aktuellste Version und klicken Sie dann auf den **Herunterladen**-Button. Warten Sie geduldig, während sich die Fortschrittsanzeige von 0 nach 100 % bewegt. Sobald die Installation beendet ist, können Sie umblättern – Sie sind bereit für den nächsten Schritt.

← Unter Windows 11 oder neuer wollen Sie vielleicht das neue Windows-Terminal herunterladen, das (bei Bedarf) eine bessere/schönere Kommandozeilenumgebung bietet.

Installation unter macOS

Aktuellen Macs liegt oft ein älteres Python-Release bei. *Verwenden Sie dies nicht*. Besuchen Sie stattdessen die Python-Homepage unter <https://www.python.org> und bewegen Sie den Mauszeiger über den »Downloads«-Link. Es erscheint ein Untermenü. Klicken Sie auf den Button unterhalb des Texts »Download for macOS«. Das aktuelle Release sollte nun heruntergeladen werden. Die Python-Website erkennt von selbst, dass Sie die Seite von einem Mac aus aufrufen. Sobald der Download beendet ist, führen Sie das Installationsprogramm aus, das bereits in Ihrem Downloads-Ordner auf Sie wartet. Klicken Sie auf den **Fortfahren**-Button, bis keine weiteren **Fortfahren**-Buttons mehr angezeigt werden. Sobald die Installation beendet ist, können Sie umblättern – Sie sind bereit für den nächsten Schritt.

← Es ist nicht nötig, ältere auf Ihrem Mac vorinstallierte Python-Versionen zu entfernen. Sie werden durch diese Version ersetzt.

Wenn Sie den *Homebrew Package Manager* verwenden, haben Sie Glück, und es gibt eine weitere Möglichkeit. Homebrew unterstützt den Download und die Installation des aktuellen Python-Release.

Installation unter Linux

Die *Von-Kopf-bis-Fuß-Coder* sind ein zusammengewürfeltes Team von Technikfans, deren Aufgabe es ist, die *Von-Kopf-bis-Fuß*-Autoren auf dem richtigen und schmalen Weg zu halten (was nicht immer einfach ist). Die Coder lieben *Linux*, speziell die *Ubuntu*-Distribution. Daher werden auch wir sie hier verwenden.

Es ist nicht weiter überraschend, dass eine aktuelle Version von Python 3 in der neuesten Ubuntu-Distribution bereits vorinstalliert ist. Ist das bei Ihnen auch so, war's das schon. Möglicherweise wollen Sie noch **apt** benutzen, um das `python3-pip`-Package zu installieren.

Wenn Sie eine andere Linux-Distribution als Ubuntu nutzen, verwenden Sie den Paketmanager Ihres Systems, um Python 3 auf Ihrem Linux-System zu installieren. Sobald das erledigt ist, können Sie umblättern – Sie sind bereit für den nächsten Schritt.