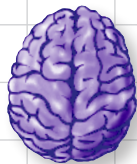


Head First Python

A Brain-Friendly Guide



Load important Python concepts directly into your brain



Model data as lists, tuples, sets, and dictionaries

Don't get in a pickle: use DB-API instead



Objects?
Decorators?
Generators?
They're all here.



Create a modern webapp with Flask



Share your code with modules

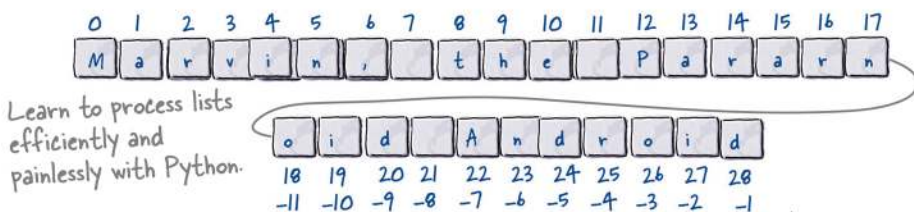


Paul Barry

Python

What will you learn from this book?

Want to learn the Python language without slogging your way through how-to manuals? With *Head First Python*, you'll quickly grasp Python's fundamentals, working with the built-in data structures and functions. Then you'll move on to building your very own webapp, exploring database management, exception handling, and data wrangling. If you're intrigued by what you can do with context managers, decorators, comprehensions, and generators, it's all here. This second edition is a complete learning experience that will help you become a Python programmer in no time.



Wow your friends by producing data-driven web pages (just like this one) using a few lines of Python code.

Gain the confidence to read, understand, and explain complex lines of code (while maintaining your sanity).

```
when2 = {dest: [k for k, v in fts.items() if v == dest] for dest in set(fts.values())}
```

What's so special about this book?

Based on the latest research in cognitive science and learning theory, *Head First Python* uses a visually rich format to engage your mind, rather than a text-heavy approach that puts you to sleep. Why waste your time struggling with new concepts? This multi-sensory learning experience is designed for the way your brain really works.

“A Python book should be as much fun as the language is. With *Head First Python*, master teacher Paul Barry delivers a quick-paced, entertaining and engaging guide to the language that will leave you well prepared to write real-world Python code.”

— Dr. Eric Freeman, computer scientist, technology educator, former CTO of Disney Online

“*Head First Python* is a great introduction to both the language and how to use Python in the real world.... If you're looking for a great introduction to Python, then this is the place to start.”

— David Griffiths, author and Agile coach

Python

US \$49.99

CAN \$57.99

ISBN: 978-1-491-91953-8



9



twitter.com/headfirstlabs
facebook.com/HeadFirst

oreilly.com
headfirstlabs.com

Advance Praise for *Head First Python, Second Edition*

“A Python book should be as much fun as the language is. With *Head First Python*, master teacher Paul Barry delivers a quick-paced, entertaining and engaging guide to the language that will leave you well prepared to write real-world Python code.”

— **Dr. Eric Freeman, computer scientist, technology educator, and former CTO of Disney Online**

“*Head First Python* is a great introduction to both the language and how to use Python in the real world. It’s full of practical advice on coding for the web and databases, and it doesn’t shy away from difficult subjects like collections and immutability. If you’re looking for a great introduction to Python, then this is the place to start.”

— **David Griffiths, author and Agile coach**

“With major changes and updates from the first edition, this edition of *Head First Python* is sure to become a favourite in the rapidly growing collection of great Python guides. The content is structured to deliver high impact to the reader, and is heavily focused on being productive as soon as possible. All the necessary topics are covered with great clarity, and the entertaining delivery makes this book a delight to read.”

— **Caleb Hattingh, author of *20 Python Libraries You Aren’t Using (But Should)* and *Learning Cython***

“Here’s a clear and clean entry into the Python pool. No bellyflops, and you’ll go deeper than you expected to.”

— **Bill Lubanovic, author of *Introducing Python***

Praise for the first edition

“*Head First Python* is a great introduction to not just the Python language, but Python as it’s used in the real world. The book goes beyond the syntax to teach you how to create applications for Android phones, Google’s App Engine, and more.”

— **David Griffiths, author and Agile coach**

“Where other books start with theory and progress to examples, *Head First Python* jumps right in with code and explains the theory as you read along. This is a much more effective learning environment, because it engages the reader to *do* from the very beginning. It was also just a joy to read. It was fun without being flippant and informative without being condescending. The breadth of examples and explanation covered the majority of what you’ll use in your job every day. I’ll recommend this book to anyone starting out on Python.”

— **Jeremy Jones, coauthor of *Python for Unix and Linux System Administration***

Praise for other Head First books

“Kathy and Bert’s *Head First Java* transforms the printed page into the closest thing to a GUI you’ve ever seen. In a wry, hip manner, the authors make learning Java an engaging ‘what’re they gonna do next?’ experience.”

— **Warren Keuffel, *Software Development Magazine***

“Beyond the engaging style that drags you forward from know-nothing into exalted Java warrior status, *Head First Java* covers a huge amount of practical matters that other texts leave as the dreaded ‘exercise for the reader...’ It’s clever, wry, hip and practical—there aren’t a lot of textbooks that can make that claim and live up to it while also teaching you about object serialization and network launch protocols.”

— **Dr. Dan Russell, Director of User Sciences and Experience Research
IBM Almaden Research Center (and teaches Artificial Intelligence at
Stanford University)**

“It’s fast, irreverent, fun, and engaging. Be careful—you might actually learn something!”

— **Ken Arnold, former Senior Engineer at Sun Microsystems
Coauthor (with James Gosling, creator of Java), *The Java Programming
Language***

“I feel like a thousand pounds of books have just been lifted off of my head.”

— **Ward Cunningham, inventor of the Wiki and founder of the Hillside Group**

“Just the right tone for the geeked-out, casual-cool guru coder in all of us. The right reference for practical development strategies—gets my brain going without having to slog through a bunch of tired, stale professor-speak.”

— **Travis Kalanick, cofounder and CEO of Uber**

“There are books you buy, books you keep, books you keep on your desk, and thanks to O’Reilly and the Head First crew, there is the penultimate category, Head First books. They’re the ones that are dog-eared, mangled, and carried everywhere. *Head First SQL* is at the top of my stack. Heck, even the PDF I have for review is tattered and torn.”

— **Bill Sawyer, ATG Curriculum Manager, Oracle**

“This book’s admirable clarity, humor and substantial doses of clever make it the sort of book that helps even nonprogrammers think well about problem-solving.”

— **Cory Doctorow, co-editor of *Boing Boing*
Author, *Down and Out in the Magic Kingdom*
and *Someone Comes to Town, Someone Leaves Town***

Praise for other Head First books

“I received the book yesterday and started to read it...and I couldn’t stop. This is definitely très ‘cool.’ It is fun, but they cover a lot of ground and they are right to the point. I’m really impressed.”

— **Erich Gamma, IBM Distinguished Engineer, and coauthor of *Design Patterns***

“One of the funniest and smartest books on software design I’ve ever read.”

— **Aaron LaBerge, VP Technology, ESPN.com**

“What used to be a long trial and error learning process has now been reduced neatly into an engaging paperback.”

— **Mike Davidson, CEO, Newsvine, Inc.**

“Elegant design is at the core of every chapter here, each concept conveyed with equal doses of pragmatism and wit.”

— **Ken Goldstein, Executive Vice President, Disney Online**

“I ♥ *Head First HTML with CSS & XHTML*—it teaches you everything you need to learn in a ‘fun-coated’ format.”

— **Sally Applin, UI Designer and Artist**

“Usually when reading through a book or article on design patterns, I’d have to occasionally stick myself in the eye with something just to make sure I was paying attention. Not with this book. Odd as it may sound, this book makes learning about design patterns fun.

“While other books on design patterns are saying ‘Bueller...Bueller...Bueller...’ this book is on the float belting out ‘Shake it up, baby!’”

— **Eric Wuehler**

“I literally love this book. In fact, I kissed this book in front of my wife.”

— **Satish Kumar**

Other related books from O'Reilly

Learning Python

Programming Python

Python in a Nutshell

Python Cookbook

Fluent Python

Other books in O'Reilly's Head First series

Head First Ajax

Head First Android Development

Head First C

Head First C#, Third Edition

Head First Data Analysis

Head First HTML and CSS, Second Edition

Head First HTML5 Programming

Head First iPhone and iPad Development, Third Edition

Head First JavaScript Programming

Head First jQuery

Head First Networking

Head First PHP & MySQL

Head First PMP, Third Edition

Head First Programming

Head First Python, Second Edition

Head First Ruby

Head First Servlets and JSP, Second Edition

Head First Software Development

Head First SQL

Head First Statistics

Head First Web Design

Head First WordPress

For a full list of titles, go to headfirstlabs.com/books.php.

Head First Python

Second Edition

Wouldn't it be dreamy if there were a Python book that didn't make you wish you were anywhere other than stuck in front of your computer writing code? I guess it's just a fantasy...



Paul Barry

O'REILLY®

Beijing • Boston • Farnham • Sebastopol • Tokyo

Head First Python, Second Edition

by Paul Barry

Copyright © 2017 Paul Barry. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://safaribooksonline.com>). For more information, contact our corporate/institutional sales department: (800) 998-9938 or corporate@oreilly.com.

Series Creators:	Kathy Sierra, Bert Bates
Editor:	Dawn Schanafelt
Cover Designer:	Randy Comer
Production Editor:	Melanie Yarbrough
Proofreader:	Rachel Monaghan
Indexer:	Lucie Haskins
Page Viewers:	Deirdre, Joseph, Aaron, and Aideen

Printing History:

November 2010: First edition.

November 2016: Second edition.



The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. The *Head First* series designations, *Head First Python*, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc., was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

No weblogs were inappropriately searched in the making of this book, and the photos on this page (as well as the one on the author page) were supplied by *Aideen Barry*.



This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN: 978-1-491-91953-8

[M]

I continue to dedicate this book to all those generous people in the Python community who continue to help make Python what it is today.

And to all those that made learning Python and its technologies just complex enough that people need a book like *this* to learn it.

Author of Head First Python, 2nd Edition

While out walking, Paul pauses to discuss the correct pronunciation of the word “tuple” with his long-suffering wife.



This is Deirdre's usual reaction. 😊

Paul Barry lives and works in *Carlow, Ireland*, which is a small town of 35,000 people or so, located just over 80km southwest of the nation's capital: *Dublin*.

Paul has a *B.Sc. in Information Systems*, as well as an *M.Sc. in Computing*. He also has a postgraduate qualification in *Learning and Teaching*.

Paul has worked at *The Institute of Technology, Carlow* since 1995, and lectured there since 1997. Prior to becoming involved in teaching, Paul spent a decade in the IT industry working in Ireland and Canada, with the majority of his work within a healthcare setting. Paul is married to Deirdre, and they have three children (two of whom are now in college).

The Python programming language (and its related technologies) has formed an integral part of Paul's undergraduate courses since the 2007 academic year.

Paul is the author (or coauthor) of four other technical books: two on Python and two on *Perl*. In the past, he's written a heap of material for *Linux Journal Magazine*, where he was a contributing editor.

Paul was raised in *Belfast, Northern Ireland*, which may go some of the way toward explaining his take on things as well as his funny accent (unless, of course, you're also from “The North,” in which case Paul's outlook and accent are *perfectly normal*).

Find Paul on *Twitter* ([@barrypj](https://twitter.com/barrypj)), as well as at his home on the Web: <http://paulbarry.itcarlow.ie>.

Table of Contents (Summary)

1	The Basics: <i>Getting Started Quickly</i>	1
2	List Data: <i>Working with Ordered Data</i>	47
3	Structured Data: <i>Working with Structured Data</i>	95
4	Code Reuse: <i>Functions and Modules</i>	145
5	Building a Webapp: <i>Getting Real</i>	195
6	Storing and Manipulating Data: <i>Where to Put Your Data</i>	243
7	Using a Database: <i>Putting Python's DB-API to Use</i>	281
8	A Little Bit of Class: <i>Abstracting Behavior and State</i>	309
9	The Context Management Protocol: <i>Hooking into Python's with Statement</i>	335
10	Function Decorators: <i>Wrapping Functions</i>	363
11	Exception Handling: <i>What to Do When Things Go Wrong</i>	413
11 ^{3/4}	A Little Bit of Threading: <i>Dealing with Waiting</i>	461
12	Advanced Iteration: <i>Looping like Crazy</i>	477
A	Installing: <i>Installing Python</i>	521
B	Pythonanywhere: <i>Deploying Your Webapp</i>	529
C	Top Ten Things We Didn't Cover: <i>There's Always More to Learn</i>	539
D	Top Ten Projects Not Covered: <i>Even More Tools, Libraries, and Modules</i>	551
E	Getting Involved: <i>The Python Community</i>	563

Table of Contents (the real thing)

Intro

Your brain on Python. Here you are trying to *learn* something, while here your *brain* is, doing you a favor by making sure the learning doesn't *stick*. Your brain's thinking, "Better leave room for more important things, like which wild animals to avoid and whether naked snowboarding is a bad idea." So how *do* you trick your brain into thinking that your life depends on knowing how to program in Python?

Who is this book for?	xxviii
We know what you're thinking	xxix
We know what your <i>brain</i> is thinking	xxix
Metacognition: thinking about thinking	xxxi
Here's what WE did	xxxii
Read me	xxxiv
Acknowledgments	xxxvii

the basics

1

Getting Started Quickly**Get going with Python programming as quickly as possible.**

In this chapter, we introduce the basics of programming in Python, and we do this in typical *Head First* style: by jumping right in. After just a few pages, you'll have run your first sample program. By the end of the chapter, you'll not only be able to run the sample program, but you'll understand its code too (and more besides). Along the way, you'll learn about a few of the things that make **Python** the programming language it is.

Understanding IDLE's Windows	4
Executing Code, One Statement at a Time	8
Functions + Modules = The Standard Library	9
Data Structures Come Built-in	13
Invoking Methods Obtains Results	14
Deciding When to Run Blocks of Code	15
What "else" Can You Have with "if"?	17
Suites Can Contain Embedded Suites	18
Returning to the Python Shell	22
Experimenting at the Shell	23
Iterating Over a Sequence of Objects	24
Iterating a Specific Number of Times	25
Applying the Outcome of Task #1 to Our Code	26
Arranging to Pause Execution	28
Generating Random Integers with Python	30
Coding a Serious Business Application	38
Is Indentation Driving You Crazy?	40
Asking the Interpreter for Help on a Function	41
Experimenting with Ranges	42
Chapter 1's Code	46



list data

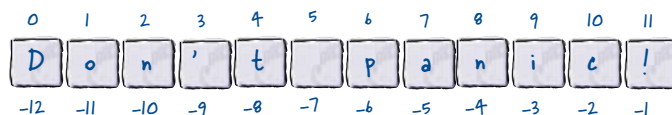
2

Working with Data

All programs process data, and Python programs are no exception.

In fact, take a look around: *data is everywhere*. A lot of, if not most, programming is all about data: *acquiring* data, *processing* data, *understanding* data. To work with data effectively, you need somewhere to *put* your data when processing it. Python shines in this regard, thanks (in no small part) to its inclusion of a handful of *widely applicable* data structures: **lists**, **dictionaries**, **tuples**, and **sets**. In this chapter, we'll preview all four, before spending the majority of this chapter digging deeper into **lists** (and we'll deep-dive into the other three in the next chapter). We're covering these data structures early, as most of what you'll likely do with Python will revolve around working with data.

Numbers, Strings...and Objects	48
Meet the Four Built-in Data Structures	50
An Unordered Data Structure: Dictionary	52
A Data Structure That Avoids Duplicates: Set	53
Creating Lists Literally	55
Use Your Editor When Working on More Than a Few Lines of Code	57
“Growing” a List at Runtime	58
Checking for Membership with “in”	59
Removing Objects from a List	62
Extending a List with Objects	64
Inserting an Object into a List	65
How to Copy a Data Structure	73
Lists Extend the Square Bracket Notation	75
Lists Understand Start, Stop, and Step	76
Starting and Stopping with Lists	78
Putting Slices to Work on Lists	80
Python’s “for” Loop Understands Lists	86
Marvin’s Slices in Detail	88
When Not to Use Lists	91
Chapter 2’s Code, 1 of 2	92



structured data

3

Working with Structured Data**Python's list data structure is great, but it isn't a data**

panacea. When you have *truly* structured data (and using a list to store it may not be the best choice), Python comes to your rescue with its built-in **dictionary**. Out of the box, the dictionary lets you store and manipulate any collection of *key/value pairs*. We look long and hard at Python's dictionary in this chapter, and—along the way—meet **set** and **tuple**, too. Together with the **list** (which we met in the previous chapter), the dictionary, set, and tuple data structures provide a set of built-in data tools that help to make Python and data a powerful combination.

A Dictionary Stores Key/Value Pairs	96
How to Spot a Dictionary in Code	98
Insertion Order Is NOT Maintained	99
Value Lookup with Square Brackets	100
Working with Dictionaries at Runtime	101
Updating a Frequency Counter	105
Iterating Over a Dictionary	107
Iterating Over Keys and Values	108
Iterating Over a Dictionary with “items”	110
Just How Dynamic Are Dictionaries?	114
Avoiding KeyErrors at Runtime	116
Checking for Membership with “in”	117
Ensuring Initialization Before Use	118
Substituting “not in” for “in”	119
Putting the “setdefault” Method to Work	120
Creating Sets Efficiently	124
Taking Advantage of Set Methods	125
Making the Case for Tuples	132
Combining the Built-in Data Structures	135
Accessing a Complex Data Structure's Data	141
Chapter 3's Code, 1 of 2	143



Name: Ford Prefect
 Gender: Male
 Occupation: Researcher
 Home Planet: Betelgeuse Seven

code reuse

4

Functions and Modules**Reusing code is key to building a maintainable system.**

And when it comes to reusing code in Python, it all starts and ends with the humble **function**. Take some lines of code, give them a name, and you've got a function (which can be reused). Take a collection of functions and package them as a file, and you've got a **module** (which can also be reused). It's true what they say: *it's good to share*, and by the end of this chapter, you'll be well on your way to **sharing** and **reusing** your code, thanks to an understanding of how Python's functions and modules work.

Reusing Code with Functions	146
Introducing Functions	147
Invoking Your Function	150
Functions Can Accept Arguments	154
Returning One Value	158
Returning More Than One Value	159
Recalling the Built-in Data Structures	161
Making a Generically Useful Function	165
Creating Another Function, 1 of 3	166
Specifying Default Values for Arguments	170
Positional Versus Keyword Assignment	171
Updating What We Know About Functions	172
Running Python from the Command Line	175
Creating the Required Setup Files	179
Creating the Distribution File	180
Installing Packages with “pip”	182
Demonstrating Call-by-Value Semantics	185
Demonstrating Call-by-Reference Semantics	186
Install the Testing Developer Tools	190
How PEP 8-Compliant Is Our Code?	191
Understanding the Failure Messages	192
Chapter 4's Programs	194



module

building a webapp

5

Getting Real**At this stage, you know enough Python to be dangerous.**

With this book's first four chapters behind you, you're now in a position to productively use Python within any number of application areas (even though there's still lots of Python to learn). Rather than explore the long list of what these application areas are, in this and subsequent chapters, we're going to structure our learning around the development of a web-hosted application, which is an area where Python is especially strong. Along the way, you'll learn a bit more about Python.

Python: What You Already Know	196
What Do We Want Our Webapp to Do?	200
Let's Install Flask	202
How Does Flask Work?	203
Running Your Flask Webapp for the First Time	204
Creating a Flask Webapp Object	206
Decorating a Function with a URL	207
Running Your Webapp's Behavior(s)	208
Exposing Functionality to the Web	209
Building the HTML Form	213
Templates Relate to Web Pages	216
Rendering Templates from Flask	217
Displaying the Webapp's HTML Form	218
Preparing to Run the Template Code	219
Understanding HTTP Status Codes	222
Handling Posted Data	223
Refining the Edit/Stop/Start/Test Cycle	224
Accessing HTML Form Data with Flask	226
Using Request Data in Your Webapp	227
Producing the Results As HTML	229
Preparing Your Webapp for the Cloud	238
Chapter 5's Code	241



storing and manipulating data

6

Where to Put Your Data

Sooner or later, you'll need to safely store your data somewhere.

And when it comes to **storing data**, Python has you covered. In this chapter, you'll learn about storing and retrieving data from *text files*, which—as storage mechanisms go—may feel a bit simplistic, but is nevertheless used in many problem areas. As well as storing and retrieving your data from files, you'll also learn some tricks of the trade when it comes to manipulating data. We're saving the “serious stuff” (storing data in a database) until the next chapter, but there's plenty to keep us busy for now when working with files.

Doing Something with Your Webapp's Data	244
Python Supports Open, Process, Close	245
Reading Data from an Existing File	246
A Better Open, Process, Close: “with”	248
View the Log Through Your Webapp	254
Examine the Raw Data with View Source	256
It's Time to Escape (Your Data)	257
Viewing the Entire Log in Your Webapp	258
Logging Specific Web Request Attributes	261
Log a Single Line of Delimited Data	262
From Raw Data to Readable Output	265
Generate Readable Output With HTML	274
Embed Display Logic in Your Template	275
Producing Readable Output with Jinja2	276
The Current State of Our Webapp Code	278
Asking Questions of Your Data	279
Chapter 6's Code	280

Form Data	Remote_addr	User_agent	Results
<code>ImmutableMultiDict([('phrase', 'hitch-hiker'), ('letters', 'aeiou')])</code>	<code>127.0.0.1</code>	<code>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</code>	<code>{'e', 'i'}</code>

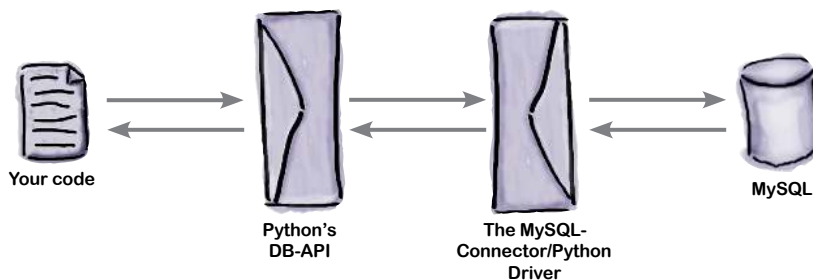
using a database

7

Putting Python's DB-API to Use

Storing data in a relational database system is handy. In this chapter, you'll learn how to write code that interacts with the popular **MySQL** database technology, using a generic database API called **DB-API**. The DB-API (which comes standard with every Python install) allows you to write code that is easily transferred from one database product to the next... assuming your database talks SQL. Although we'll be using MySQL, there's nothing stopping you from using your DB-API code with your favorite relational database, whatever it may be. Let's see what's involved in using a relational database with Python. There's not a lot of new Python in this chapter, but using Python to talk to databases is a **big deal**, so it's well worth learning.

Database-Enabling Your Webapp	282
Task 1: Install the MySQL Server	283
Introducing Python's DB-API	284
Task 2: Install a MySQL Database Driver for Python	285
Install MySQL-Connector/Python	286
Task 3: Create Our Webapp's Database and Tables	287
Decide on a Structure for Your Log Data	288
Confirm Your Table Is Ready for Data	289
Task 4: Create Code to Work with Our Webapp's Database and Tables	296
Storing Data Is Only Half the Battle	300
How Best to Reuse Your Database Code?	301
Consider What You're Trying to Reuse	302
What About That Import?	303
You've Seen This Pattern Before	305
The Bad News Isn't Really All That Bad	306
Chapter 7's Code	307



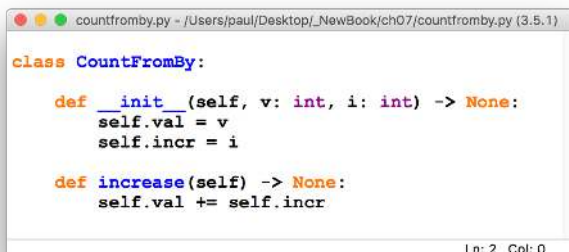
8

a little bit of class

Abstracting Behavior and State**Classes let you bundle code behavior and state together.**

In this chapter, you're setting your webapp aside while you learn about creating Python **classes**. You're doing this in order to get to the point where you can create a context manager with the help of a Python class. As creating and using classes is such a useful thing to know about anyway, we're dedicating this chapter to them. We won't cover everything about classes, but we'll touch on all the bits you'll need to understand in order to confidently create the context manager your webapp is waiting for.

Hooking into the “with” Statement	310
An Object-Oriented Primer	311
Creating Objects from Classes	312
Objects Share Behavior but Not State	313
Doing More with CountFromBy	314
Invoking a Method: Understand the Details	316
Adding a Method to a Class	318
The Importance of “self”	320
Coping with Scoping	321
Prefix Your Attribute Names with “self”	322
Initialize (Attribute) Values Before Use	323
Dunder “init” Initializes Attributes	324
Initializing Attributes with Dunder “init”	325
Understanding CountFromBy's Representation	328
Defining CountFromBy's Representation	329
Providing Sensible Defaults for CountFromBy	330
Classes: What We Know	332
Chapter 8's Code	333



```

class CountFromBy:
    def __init__(self, v: int, i: int) -> None:
        self.val = v
        self.incr = i

    def increase(self) -> None:
        self.val += self.incr
  
```

Ln: 2 Col: 0

the context management protocol



Hooking into Python's with Statements

It's time to take what you've just learned and put it to work.

Chapter 7 discussed using a **relational database** with Python, while Chapter 8 provided an introduction to using **classes** in your Python code. In this chapter, both of these techniques are combined to produce a **context manager** that lets us extend the `with` statement to work with relational database systems. In this chapter, you'll hook into the `with` statement by creating a new class, which conforms to Python's **context management protocol**.

What's the Best Way to Share Our Webapp's Database Code?	336
Managing Context with Methods	338
You've Already Seen a Context Manager in Action	339
Create a New Context Manager Class	340
Initialize the Class with the Database Config	341
Perform Setup with Dunder "enter"	343
Perform Teardown with Dunder "exit"	345
Reconsidering Your Webapp Code, 1 of 2	348
Recalling the "log_request" Function	350
Amending the "log_request" Function	351
Recalling the "view_the_log" Function	352
It's Not Just the Code That Changes	353
Amending the "view_the_log" Function	354
Answering the Data Questions	359
Chapter 9's Code, 1 of 2	360

```
File Edit Window Help Checking our log DB
$ mysql -u vsearch -p vsearchlogDB
Enter password:
Welcome to MySQL monitor...

mysql> select * from log;
+----+-----+-----+-----+-----+-----+-----+
| id | ts       | phrase                | letters | ip       | browser_string | results          |
+----+-----+-----+-----+-----+-----+-----+
| 1  | 2016-03-09 13:40:46 | life, the uni ... ything | aeiou   | 127.0.0.1 | firefox        | {'u', 'e', 'i', 'a'} |
| 2  | 2016-03-09 13:42:07 | hitch-hiker           | aeiou   | 127.0.0.1 | safari         | {'i', 'e'}         |
| 3  | 2016-03-09 13:42:15 | galaxy                 | xyz     | 127.0.0.1 | chrome        | {'y', 'x'}         |
| 4  | 2016-03-09 13:43:07 | hitch-hiker           | xyz     | 127.0.0.1 | firefox        | set()              |
+----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.0 sec)

mysql> quit
Bye
```