O'REILLY®

第2版

Web Scraping with Python

Python网络数据采集（影印版）

東南大學出版社

Ryan Mitchell 著

第2版

Python网络数据采集（影印版）

Web Scraping with Python

Ryan Mitchell 著

l • Tokyo O'REILLY®

, Inc.授权东南大学出版社出版

南京 东南大学出版社

图书在版编目(CIP)数据

Python网络数据采集:第2版:英文/(美)瑞安·米切尔(Ryan Mitchell)著. —影印本. —南京:东南大学出版社,2018.11

书名原文:Web Scraping with Python, 2E

ISBN 978-7-5641-7977-9

Ⅰ.①P… Ⅱ.①瑞… Ⅲ.①软件工具—程序设计—英文 Ⅳ.①TP311.561

中国版本图书馆CIP数据核字(2018)第209232号

图字:10-2018-193号

Python网络数据采集 第2版(影印版)

出版发行:东南大学出版社
地　　址:南京四牌楼2号　　邮编:210096
出 版 人:江建中
网　　址:http://www.seupress.com
电子邮件:press@seupress.com
印　　刷:常州市武进第三印刷有限公司
开　　本:787毫米×980毫米　16开本
印　　张:19.25
字　　数:377千字
版　　次:2018年11月第1版
印　　次:2018年11月第1次印刷
书　　号:ISBN 978-7-5641-7977-9
定　　价:89.00元

本社图书若有印装质量问题,请直接与营销部联系。电话(传真):025-83791830

Preface

To those who have not developed the skill, computer programming can seem like a kind of magic. If programming is magic, *web scraping* is wizardry: the application of magic for particularly impressive and useful—yet surprisingly effortless—feats.

In my years as a software engineer, I've found that few programming practices capture the excitement of both programmers and laymen alike quite like web scraping. The ability to write a simple bot that collects data and streams it down a terminal or stores it in a database, while not difficult, never fails to provide a certain thrill and sense of possibility, no matter how many times you might have done it before.

Unfortunately, when I speak to other programmers about web scraping, there's a lot of misunderstanding and confusion about the practice. Some people aren't sure it's legal (it is), or how to handle problems like JavaScript-heavy pages or required logins. Many are confused about how to start a large web scraping project, or even where to find the data they're looking for. This book seeks to put an end to many of these common questions and misconceptions about web scraping, while providing a comprehensive guide to most common web scraping tasks.

Web scraping is a diverse and fast-changing field, and I've tried to provide both high-level concepts and concrete examples to cover just about any data collection project you're likely to encounter. Throughout the book, code samples are provided to demonstrate these concepts and allow you to try them out. The code samples themselves can be used and modified with or without attribution (although acknowledgment is always appreciated). All code samples are available on GitHub (*http://www.pythonscraping.com/code/*) for viewing and downloading.

What Is Web Scraping?

The automated gathering of data from the internet is nearly as old as the internet itself. Although *web scraping* is not a new term, in years past the practice has been more commonly known as *screen scraping*, *data mining*, *web harvesting*, or similar

variations. General consensus today seems to favor *web scraping*, so that is the term I use throughout the book, although I also refer to programs that specifically traverse multiple pages as *web crawlers* or refer to the web scraping programs themselves as *bots*.

In theory, web scraping is the practice of gathering data through any means other than a program interacting with an API (or, obviously, through a human using a web browser). This is most commonly accomplished by writing an automated program that queries a web server, requests data (usually in the form of HTML and other files that compose web pages), and then parses that data to extract needed information.

In practice, web scraping encompasses a wide variety of programming techniques and technologies, such as data analysis, natural language parsing, and information security. Because the scope of the field is so broad, this book covers the fundamental basics of web scraping and crawling in Part I and delves into advanced topics in Part II. I suggest that all readers carefully study the first part and delve into the more specific in the second part as needed.

Why Web Scraping?

If the only way you access the internet is through a browser, you're missing out on a huge range of possibilities. Although browsers are handy for executing JavaScript, displaying images, and arranging objects in a more human-readable format (among other things), web scrapers are excellent at gathering and processing large amounts of data quickly. Rather than viewing one page at a time through the narrow window of a monitor, you can view databases spanning thousands or even millions of pages at once.

In addition, web scrapers can go places that traditional search engines cannot. A Google search for "cheapest flights to Boston" will result in a slew of advertisements and popular flight search sites. Google knows only what these websites say on their content pages, not the exact results of various queries entered into a flight search application. However, a well-developed web scraper can chart the cost of a flight to Boston over time, across a variety of websites, and tell you the best time to buy your ticket.

You might be asking: "Isn't data gathering what APIs are for?" (If you're unfamiliar with APIs, see Chapter 12.) Well, APIs can be fantastic, if you find one that suits your purposes. They are designed to provide a convenient stream of well-formatted data from one computer program to another. You can find an API for many types of data you might want to use, such as Twitter posts or Wikipedia pages. In general, it is preferable to use an API (if one exists), rather than build a bot to get the same data. However, an API might not exist or be useful for your purposes, for several reasons:

- You are gathering relatively small, finite sets of data across a large collection of websites without a cohesive API.
- The data you want is fairly small or uncommon, and the creator did not think it warranted an API.
- The source does not have the infrastructure or technical ability to create an API.
- The data is valuable and/or protected and not intended to be spread widely.

Even when an API *does* exist, the request volume and rate limits, the types of data, or the format of data that it provides might be insufficient for your purposes.

This is where web scraping steps in. With few exceptions, if you can view data in your browser, you can access it via a Python script. If you can access it in a script, you can store it in a database. And if you can store it in a database, you can do virtually anything with that data.

There are obviously many extremely practical applications of having access to nearly unlimited data: market forecasting, machine-language translation, and even medical diagnostics have benefited tremendously from the ability to retrieve and analyze data from news sites, translated texts, and health forums, respectively.

Even in the art world, web scraping has opened up new frontiers for creation. The 2006 project "We Feel Fine" (*http://wefeelfine.org/*) by Jonathan Harris and Sep Kamvar scraped a variety of English-language blog sites for phrases starting with "I feel" or "I am feeling." This led to a popular data visualization, describing how the world was feeling day by day and minute by minute.

Regardless of your field, web scraping almost always provides a way to guide business practices more effectively, improve productivity, or even branch off into a brand-new field entirely.

About This Book

This book is designed to serve not only as an introduction to web scraping, but as a comprehensive guide to collecting, transforming, and using data from uncooperative sources. Although it uses the Python programming language and covers many Python basics, it should not be used as an introduction to the language.

If you don't know any Python at all, this book might be a bit of a challenge. Please do not use it as an introductory Python text. With that said, I've tried to keep all concepts and code samples at a beginning-to-intermediate Python programming level in order to make the content accessible to a wide range of readers. To this end, there are occasional explanations of more advanced Python programming and general computer science topics where appropriate. If you are a more advanced reader, feel free to skim these parts!

If you're looking for a more comprehensive Python resource, *Introducing Python* by Bill Lubanovic (O'Reilly) is a good, if lengthy, guide. For those with shorter attention spans, the video series *Introduction to Python* by Jessica McKellar (O'Reilly) is an excellent resource. I've also enjoyed *Think Python* by a former professor of mine, Allen Downey (O'Reilly). This last book in particular is ideal for those new to programming, and teaches computer science and software engineering concepts along with the Python language.

Technical books are often able to focus on a single language or technology, but web scraping is a relatively disparate subject, with practices that require the use of databases, web servers, HTTP, HTML, internet security, image processing, data science, and other tools. This book attempts to cover all of these, and other topics, from the perspective of "data gathering." It should not be used as a complete treatment of any of these subjects, but I believe they are covered in enough detail to get you started writing web scrapers!

Part I covers the subject of web scraping and web crawling in depth, with a strong focus on a small handful of libraries used throughout the book. Part I can easily be used as a comprehensive reference for these libraries and techniques (with certain exceptions, where additional references will be provided). The skills taught in the first part will likely be useful for everyone writing a web scraper, regardless of their particular target or application.

Part II covers additional subjects that the reader might find useful when writing web scrapers, but that might not be useful for all scrapers all the time. These subjects are, unfortunately, too broad to be neatly wrapped up in a single chapter. Because of this, frequent references are made to other resources for additional information.

The structure of this book enables you to easily jump around among chapters to find only the web scraping technique or information that you are looking for. When a concept or piece of code builds on another mentioned in a previous chapter, I explicitly reference the section that it was addressed in.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://github.com/REMitchell/python-scraping*.

This book is here to help you get your job done. If the example code in this book is useful to you, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Web Scraping with Python*, Second Edition by Ryan Mitchell (O'Reilly). Copyright 2018 Ryan Mitchell, 978-1-491-998557-1."

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at *permissions@oreilly.com*.

Unfortunately, printed books are difficult to keep up-to-date. With web scraping, this provides an additional challenge, as the many libraries and websites that the book references and that the code often depends on may occasionally be modified, and code samples may fail or produce unexpected results. If you choose to run the code samples, please run them from the GitHub repository rather than copying from the book directly. I, and readers of this book who choose to contribute (including, perhaps, you!), will strive to keep the repository up-to-date with required modifications and notes.

In addition to code samples, terminal commands are often provided to illustrate how to install and run software. In general, these commands are geared toward Linux-based operating systems, but will usually be applicable for Windows users with a properly configured Python environment and pip installation. When this is not the case, I have provided instructions for all major operating systems, or external references for Windows users to accomplish the task.

O'Reilly Safari

Safari (formerly Safari Books Online) is a membership-based training and reference platform for enterprise, government, educators, and individuals.

Members have access to thousands of books, training videos, Learning Paths, interactive tutorials, and curated playlists from over 250 publishers, including O'Reilly Media, Harvard Business Review, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Adobe, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, and Course Technology, among others.

For more information, please visit *http://oreilly.com/safari*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)

707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://oreil.ly/1ePG2Uj*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

Just as some of the best products arise out of a sea of user feedback, this book never could have existed in any useful form without the help of many collaborators, cheerleaders, and editors. Thank you to the O'Reilly staff and their amazing support for this somewhat unconventional subject; to my friends and family who have offered advice and put up with impromptu readings; and to my coworkers at HedgeServ, whom I now likely owe many hours of work.

Thank you, in particular, to Allyson MacDonald, Brian Anderson, Miguel Grinberg, and Eric VanWyk for their feedback, guidance, and occasional tough love. Quite a few sections and code samples were written as a direct result of their inspirational suggestions.

Thank you to Yale Specht for his limitless patience for the past four years and two editions, providing the initial encouragement to pursue this project, and stylistic feedback during the writing process. Without him, this book would have been written in half the time but would not be nearly as useful.

Finally, thanks to Jim Waldo, who really started this whole thing many years ago when he mailed a Linux box and *The Art and Science of C* to a young and impressionable teenager.

Table of Contents

PART I

Building Scrapers

This first part of this book focuses on the basic mechanics of web scraping: how to use Python to request information from a web server, how to perform basic handling of the server's response, and how to begin interacting with a website in an automated fashion. By the end, you'll be cruising around the internet with ease, building scrapers that can hop from one domain to another, gather information, and store that information for later use.

To be honest, web scraping is a fantastic field to get into if you want a huge payout for relatively little upfront investment. In all likelihood, 90% of web scraping projects you'll encounter will draw on techniques used in just the next six chapters. This section covers what the general (albeit technically savvy) public tends to think of when they think of "web scrapers":

- Retrieving HTML data from a domain name
- Parsing that data for target information
- Storing the target information
- Optionally, moving to another page to repeat the process

This will give you a solid foundation before moving on to more complex projects in Part II. Don't be fooled into thinking that this first section isn't as important as some of the more advanced projects in the second half. You will use nearly all the information in the first half of this book on a daily basis while writing web scrapers!