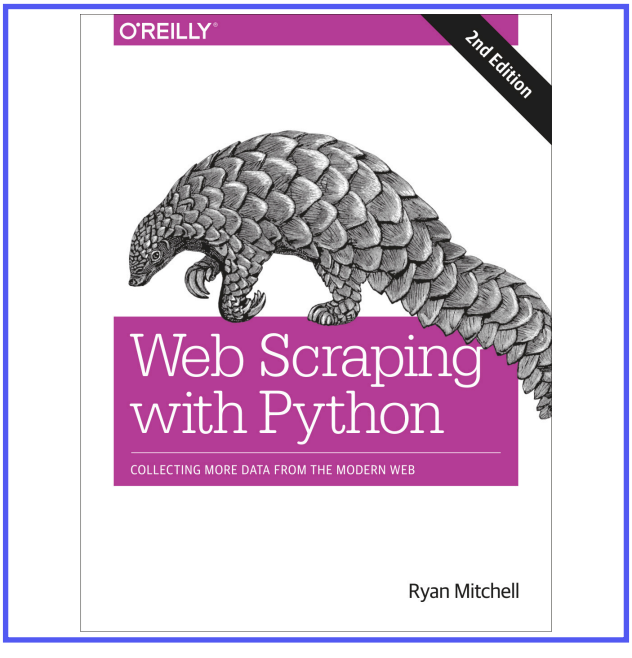


ebookbig.com

or scan the QR code to access the digital study portal.

Structuring Crawlers 58
Crawling Sites Through Search 58
Crawling Sites Through Links 61
Crawling Multiple Page Types 64
Thinking About Web Crawler Models 65

5. Scrapy. 67
Installing Scrapy 67
Initializing a New Spider 68
Writing a Simple Scraper 69
Spidering with Rules 70
Creating Items 74
Outputting Items 76
The Item Pipeline 77
Logging with Scrapy 80
More Resources 80

6. Storing Data. 83
Media Files 83
Storing Data to CSV 86
MySQL 88
Installing MySQL 89
Some Basic Commands 91
Integrating with Python 94
Database Techniques and Good Practice 97
"Six Degrees" in MySQL 100
Email 103

Part II. Advanced Scraping

7. Reading Documents. 107
Document Encoding 107
Text 108
Text Encoding and the Global Internet 109
CSV 113
Reading CSV Files 113
PDF 115
Microsoft Word and .docx 117

8. Cleaning Your Dirty Data. 121
Cleaning in Code 121

Data Normalization 124
Cleaning After the Fact 126
OpenRefine 126

9. Reading and Writing Natural Languages. 131
Summarizing Data 132
Markov Models 135
Six Degrees of Wikipedia: Conclusion 139
Natural Language Toolkit 142
Installation and Setup 142
Statistical Analysis with NLTK 143
Lexicographical Analysis with NLTK 145
Additional Resources 149

10. Crawling Through Forms and Logins. 151
Python Requests Library 151
Submitting a Basic Form 152
Radio Buttons, Checkboxes, and Other Inputs 154
Submitting Files and Images 155
Handling Logins and Cookies 156
HTTP Basic Access Authentication 157
Other Form Problems 158

11. Scraping JavaScript. 161
A Brief Introduction to JavaScript 162
Common JavaScript Libraries 163
Ajax and Dynamic HTML 165
Executing JavaScript in Python with Selenium 166
Additional Selenium Webdrivers 171
Handling Redirects 171
A Final Note on JavaScript 173

12. Crawling Through APIs. 175
A Brief Introduction to APIs 175
HTTP Methods and APIs 177
More About API Responses 178
Parsing JSON 179
Undocumented APIs 181
Finding Undocumented APIs 182
Documenting Undocumented APIs 184
Finding and Documenting APIs Automatically 184
Combining APIs with Other Data Sources 187

More About APIs 190

13. Image Processing and Text Recognition. 193
Overview of Libraries 194
Pillow 194
Tesseract 195
NumPy 197
Processing Well-Formatted Text 197
Adjusting Images Automatically 200
Scraping Text from Images on Websites 203
Reading CAPTCHAs and Training Tesseract 206
Training Tesseract 207
Retrieving CAPTCHAs and Submitting Solutions 211

14. Avoiding Scraping Traps. 215
A Note on Ethics 215
Looking Like a Human 216
Adjust Your Headers 217
Handling Cookies with JavaScript 218
Timing Is Everything 220
Common Form Security Features 221
Hidden Input Field Values 221
Avoiding Honeypots 223
The Human Checklist 224

15. Testing Your Website with Scrapers. 227
An Introduction to Testing 227
What Are Unit Tests? 228
Python unittest 228
Testing Wikipedia 230
Testing with Selenium 233
Interacting with the Site 233
unittest or Selenium? 236

16. Web Crawling in Parallel. 239
Processes versus Threads 239
Multithreaded Crawling 240
Race Conditions and Queues 242
The threading Module 245
Multiprocess Crawling 247
Multiprocess Crawling 249
Communicating Between Processes 251

Multiprocess Crawling—Another Approach 253

17. Scraping Remotely. 255
Why Use Remote Servers? 255
Avoiding IP Address Blocking 256
Portability and Extensibility 257
Tor 257
PySocks 259
Remote Hosting 259
Running from a Website-Hosting Account 260
Running from the Cloud 261
Additional Resources 262

18. The Legalities and Ethics of Web Scraping. 263
Trademarks, Copyrights, Patents, Oh My! 263
Copyright Law 264
Trespass to Chattels 266
The Computer Fraud and Abuse Act 268
robots.txt and Terms of Service 269
Three Web Scrapers 272
eBay versus Bidder's Edge and Trespass to Chattels 272
United States v. Auernheimer and The Computer Fraud and Abuse Act 274
Field v. Google: Copyright and robots.txt 275
Moving Forward 276

Index. 279

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 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" 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.