Learning Scrapy

Scrapy is a Python, open source scraping framework built to extract web data with ease. With a wealth of features accessible through simple coding or plain configuration, Scrapy saves you weeks of development time and allows you to accurately and efficiently extract needed data. It has highly active, growing community and has become the framework of choice for hackers, startup entrepreneurs, and web scraping specialists.

This book covers Scrapy version 1.0. It explains the fundamentals of Scrapy, and how to extract data from any source, clean it up, and mold it to meet your requirements using Python and third-party APIs. It covers how to efficiently feed scraped data into databases, search engines, and stream-data processing systems like Apache Spark. By the end of this book, you'll be able to confidently scrape data and utilize it to your applications.

Who this book is written for
If you are a software developer, data scientist, NLP fan, machine learning enthusiast, or you just need to migrate your company's wiki from a legacy platform, then this book is for you. It is perfect for those who need effortless and instant access to large amounts of semi-structured data.

What you will learn from this book
- How to use HTML and write XPath to extract the data you need
- How to write Scrapy spiders with Python, and perform web crawls
- Ways to push your data into any database, search engine, or analytics system
- How to configure your spider to download files and images, plus use proxies
- Ways to create efficient pipelines that shape data
- How to use Twisted event-driven APIs to process hundreds of items concurrently
- Tricks to make your crawler super-fast and memory efficient, and fine-tune Scrapy's performance
- The best way to perform large-scale distributed crawls with scrapyd and scrapinghub


Dimitrios Kouzis-Loukas
Learn the art of efficient web scraping and crawling with Python

$34.99 US
£22.99 UK

Prices do not include local sales tax or VAT where applicable
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Introducing Scrapy'
- A synopsis of the book’s content
- More information on Learning Scrapy
About the Author

Dimitrios Kouzis-Loukas has over fifteen years experience as a topnotch software developer. He uses his acquired knowledge and expertise to teach a wide range of audiences how to write great software, as well.

He studied and mastered several disciplines, including mathematics, physics, and microelectronics. His thorough understanding of these subjects helped him raise his standards beyond the scope of "pragmatic solutions." He knows that true solutions should be as certain as the laws of physics, as robust as ECC memories, and as universal as mathematics.

Dimitrios now develops distributed, low-latency, highly-availability systems using the latest datacenter technologies. He is language agnostic, yet has a slight preference for Python, C++, and Java. A firm believer in open source software and hardware, he hopes that his contributions will benefit individual communities as well as all of humanity.
Let me take a wild guess. One of these two stories is curiously similar to yours:

Your first encounter with Scrapy was while searching the net for something along the lines of “web scraping Python”. You had a quick look at it and thought, “This is too complex...I just need something simple.” You went on and developed a Python script using requests, struggled a bit with beautiful soup, but finally made something cool. It was kind of slow, so you let it run overnight. You restarted it a few times, ignored some semi-broken links and non-English characters, and in the morning, most of the website was proudly on your hard disk. Sadly, for some unknown reason, you didn't want to see your code again. The next time you had to scrape something, you went directly to scrapy.org and this time the documentation made perfect sense. Scrapy now felt like it was elegantly and effortlessly solving all of the problems that you faced, and it even took care of problems you hadn't thought of yet. You never looked back.

Alternatively, your first encounter with Scrapy was while doing research for a web-scraping project. You needed something robust, fast, and enterprise-grade, so most of the fancy one-click web-scraping tools were out of question. You needed it to be simple but at the same time flexible enough to allow you to customize its behavior for different sources, provide different types of output feeds, and reliably run 24/7 in an automated manner. Companies that provided scraping as a service seemed too expensive and you were more comfortable using open source solutions than feeling locked on vendors. From the very beginning, Scrapy looked like a clear winner.
No matter how you got here, I'm glad to meet you on a book that is entirely devoted to Scrapy. Scrapy is the secret of web-scraping experts throughout the world. They know how to maneuver it to save them hours of work, deliver stellar performance, and keep their hosting bills to an absolute minimum. If you are less experienced and you want to achieve their results, unfortunately, Google will do you a disservice. The majority of Scrapy information on the Web is either simplistic and inefficient or complex. This book is an absolute necessity for everyone who wants accurate, accessible, and well-organized information on how to make the most out of Scrapy. It is my hope that it will help the Scrapy community grow even further and give it the wide adoption that it rightfully deserves.

What this book covers

*Chapter 1, Introducing Scrapy,* will introduce you to this book and Scrapy, and will allow you to set clear expectations for the framework and the rest of the book.

*Chapter 2, Understanding HTML and XPath,* aims to bring web-crawling beginners up to speed with the essential web-related technologies and techniques that we will use thereafter.

*Chapter 3, Basic Crawling,* is where we learn how to install Scrapy and crawl a website. We develop this example step by step by showing you the methodology and the way of thinking behind every action. After this chapter, you will be able to crawl the majority of simple websites.

*Chapter 4, From Scrapy to a Mobile App,* shows us how we can use our scraper to populate a database and feed a mobile application. After this chapter, you will have a clear appreciation of the benefits that web crawling brings in time to market terms.

*Chapter 5, Quick Spider Recipes,* demonstrates more powerful spider features, allowing us to log in, scrape faster, consume APIs, and crawl lists of URLs.

*Chapter 6, Deploying to Scrapinghub,* shows us how to deploy spiders to Scrapinghub’s cloud servers and enjoy availability, easy deployment, and control.

*Chapter 7, Configuration and Management,* is a well-organized presentation of the impressive number of features that one can enable and fine-tune using Scrapy’s configuration.
Chapter 8, *Programming Scrapy*, takes our knowledge to a whole new level by showing us how to use the underlying Twisted engine and Scrapy's architecture to extend every aspect of its functionality.

Chapter 9, *Pipeline Recipes*, presents numerous examples where we alter Scrapy's functionality to insert into databases such as MySQL, Elasticsearch, and Redis, interface APIs, and legacy applications with virtually no degradation of performance.

Chapter 10, *Understanding Scrapy's Performance*, will help us understand how Scrapy spends its time, and what exactly we need to do to increase its performance.

Chapter 11, *Distributed Crawling with Scrapyd and Real-Time Analytics*, is our final chapter showing how to use scrapyd in multiple servers to achieve horizontal scalability, and how to feed crawled data to an Apache Spark server that performs stream analytics on it.
Introducing Scrapy

Welcome to your Scrapy journey. With this book, we aim to take you from a Scrapy beginner—someone who has little or no experience with Scrapy—to a level where you will be able to confidently use this powerful framework to scrape large datasets from the web or other sources. In this chapter, we will introduce you to Scrapy and talk to you about some of the great things you can achieve with it.

Hello Scrapy

Scrapy is a robust web framework for scraping data from various sources. As a casual web user, you will often find yourself wishing to be able to get data from a website that you’re browsing on a spreadsheet program like Excel (see Chapter 3, Basic Crawling) in order to access it while you’re offline or to perform calculations. As a developer, you’ll often wish to be able to combine data from various data sources, but you are well aware of the complexities of retrieving or extracting them. Scrapy can help you complete both easy and complex data extraction initiatives.

Scrapy is built upon years of experience in extracting massive amounts of data in a robust and efficient manner. With Scrapy, you are able to do with a single setting what would take various classes, plug-ins, and configuration in most other scraping frameworks. A quick look at Chapter 7, Configuration and Management will make you appreciate how much you can achieve in Scrapy with a few lines of configuration.
From a developer's perspective, you will also appreciate Scrapy's event-based architecture (we will explore it in depth in Chapter 8, Programming Scrapy and Chapter 9, Pipeline Recipes). It allows us to cascade operations that clean, form, and enrich data, store them in databases, and so on, while enjoying very low degradation in performance—if we do it in the right way, of course. In this book, you will learn exactly how to do so. Technically speaking, being event-based, Scrapy allows us to disconnect latency from throughput by operating smoothly while having thousands of connections open. As an extreme example, imagine that you aim to extract listings from a website that has summary pages with a hundred listings per page. Scrapy will effortlessly perform 16 requests on that site in parallel, and assuming that, on an average, a request takes a second to complete, you will be crawling at 16 pages per second. If you multiply that with the number of listings per page, you will be generating 1600 listings per second. Imagine now that for each of those listings you have to do a write to a massively concurrent cloud storage, which takes 3 seconds (very bad idea) on an average. In order to support the throughput of 16 requests per second, it turns out that we need to be running $1600 \cdot 3 = 4800$ write requests in parallel (you will see many such interesting calculations in Chapter 9, Pipeline Recipes). For a traditional multithreaded application, this would translate to 4800 threads, which would be a very unpleasant experience for both you and the operating system. In Scrapy's world, 4800 concurrent requests is business as usual as long as the operating system is okay with it. Furthermore, memory requirements of Scrapy closely follow the amount of data that you need for your listings in contrast to a multithreaded application, where each thread adds a significant overhead as compared to a listing's size.

In a nutshell, slow or unpredictable websites, databases, or remote APIs won't have devastating consequences on your scraper's performance, since you can run many requests concurrently, and manage everything from a single thread. This translates to lower hosting bills, opportunity for co-hosting scrapers with other applications, and simpler code (no synchronization necessary) as compared to typical multithreaded applications.

More reasons to love Scrapy

Scrapy has been around for more than half a decade, and is mature and stable. Beyond the performance benefits that we mentioned in the previous section, there are several other reasons to love Scrapy:

- **Scrapy understands broken HTML**

  You can use Beautiful Soup or lxml directly from Scrapy, but Scrapy provides **selectors**—a higher level XPath (mainly) interface on top of lxml. It is able to efficiently handle broken HTML code and confusing encodings.
Chapter 1

- Community
  Scrapy has a vibrant community. Just have a look at the mailing list at https://groups.google.com/forum/#!forum/scrapy-users and the thousands of questions in Stack Overflow at http://stackoverflow.com/questions/tagged/scrapy. Most questions get answered within minutes. More community resources are available at http://scrapy.org/community/.

- Well-organized code that is maintained by the community
  Scrapy requires a standard way of organizing your code. You write little Python modules called spiders and pipelines, and you automatically gain from any future improvements to the engine itself. If you search online, you will find quite a few professionals who have Scrapy experience. This means that it's quite easy to find a contractor who will help you maintain or extend your code. Whoever joins your team won't have to go through the learning curve of understanding the peculiarities of your own custom crawler.

- Growing feature set but also quality focused
  If you have a quick look at the Release Notes (http://doc.scrapy.org/en/latest/news.html), you will notice that there is a growth, both in features and in stability/bug fixes.

About this book: aim and usage
With this book, we aim to teach you Scrapy by using focused examples and realistic datasets. Most chapters focus on crawling an example property rental website. We chose this, because it's representative of most of the web crawling projects, allows us to present interesting variations, and is at the same time simple. Having this example as the main theme helps us focus on Scrapy without distraction.

We start by running small crawls of a few hundred pages, and we scale it out to performing distributed crawling of fifty thousand pages within minutes in Chapter 11, Distributed Crawling with Scrapyd and Real-Time Analytics. In the process, we will show you how to connect Scrapy with services like MySQL, Redis, and Elasticsearch, use the Google geocoding API to find coordinates for the location of our example properties, and feed Apache Spark to predict the keywords which affect property prices the most.
Be prepared to read this book several times. Maybe you can start by skimming through it to understand its structure. Then read a chapter or two, learn, experiment for a while, and then move further. Don't be afraid to skip a chapter if you feel familiar with it. In particular, if you know HTML and XPath, there's no point spending much time on Chapter 2, Understanding HTML and XPath. Don't worry; this book still has plenty for you. Some chapters like Chapter 8, Programming Scrapy combine the elements of a reference and a tutorial, and go in depth into programming concepts. That's an example of a chapter one might like to read a few times, while allowing a couple of weeks of Scrapy practice in between. You don't need to perfectly master Chapter 8, Programming Scrapy before moving, for example, to Chapter 9, Pipeline Recipes, which is full of applications. Reading the latter will help you understand how to use the programming concepts, and if you wish, you can reiterate as many times as you like.

We have tried to balance the pace to keep the book both interesting and beginner-friendly. One thing we can't do though, is teach Python in this book. There are several excellent books on the subject, but what I would recommend is trying a bit more relaxed attitude while learning. One of the reasons Python is so popular is that it's relatively simple, clean, and it reads well as English. Scrapy is a high-level framework that requires learning from Python beginners and experts alike. You could call it "the Scrapy language". As a result, I would recommend going through the material, and if you feel that you find the Python syntax confusing, supplement your learning with some of the excellent online Python tutorials or free Python online courses for beginners at Coursera or elsewhere. Rest assured, you can be quite a good Scrapy developer without being a Python expert.

The importance of mastering automated data scraping
For many of us, the curiosity and the mental satisfaction in mastering a cool technology like Scrapy is sufficient to motivate us. As a pleasant surprise, while learning this great framework, we enjoy a few benefits that derive from starting the development process from data and the community instead of the code.
Chapter 1

Developing robust, quality applications, and providing realistic schedules

In order to develop modern high-quality applications, we need realistic, large datasets, if possible, before even writing a single line of code. Modern software development is all about processing large amounts of less-than-perfect data in real time to extract knowledge and actionable insights. When we develop software and apply it to large datasets, small errors and oversights are difficult to detect and might lead us to costly erroneous decisions. It’s easy, for example, to overlook entire states while trying to study demographics, just because of a bug that silently drops data when the state name is too long. By carefully scraping, and having production-quality, large, real-world datasets during development (or even earlier) during design exploration, one can find and fix bugs, and make informed engineering decisions.

As another example, imagine that you want to design an Amazon-style "if you like this, you might also like that"-style recommendation system. If you are able to crawl and collect a real-world dataset before you even start, you will quickly become aware of the issues related to invalid entries, discontinued products, duplicates, invalid characters, and performance issues due to skewed distributions. Data will force you to design algorithms robust enough to handle the products bought by thousands of people as well as new entries with zero sales. Compare that to software developed in isolation that will later, potentially after weeks of development, face the ugliness of real-world data. The two approaches might eventually converge, but the ability to provide schedule estimates you can commit to, and the quality of software as the project’s time progresses will be significantly different. Starting from data, leads to a much more pleasant and predictable software development experience.

Developing quality minimum viable products quickly

Large realistic datasets are even more essential for start-ups. You might have heard of the "Lean Startup", a term coined by Eric Ries to describe the business development process under conditions of extreme uncertainty like tech-start-ups. One of the key concepts of that framework is that of the minimum viable product (MVP)—a product with limited functionality that one can quickly develop and release to a limited audience in order to measure reactions and validate business hypotheses. Based on the reactions, a start-up might choose to continue with further investments, or "pivot" to something more promising.
Some aspects of this process that are easy to overlook are very closely connected with the data problems that Scrapy solves for us. When we ask potential customers to try our mobile app, for example, we as developers or entrepreneurs ask them to judge the functionality imagining how this app will look when completed. This might be a bit too much imagining for a non-expert. The distance between an app which shows "product 1", "product 2", and "user 433", and an application that provides information on "Samsung UN55J6200 55-Inch TV", which has a five star rating from user "Richard S." and working links that take you directly to a product detail page (despite the fact we didn't write it), is significant. It's very difficult for people to judge the functionality of an MVP objectively, unless the data that we use is realistic and somewhat exciting.

One of the reasons that some start-ups have data as an afterthought is the perception that collecting them is expensive. Indeed, we would typically need to develop forms, administration screens, and spend time entering data—or we could just use Scrapy and crawl a few websites before writing even a single line of code. You will see in Chapter 4, From Scrapy to a Mobile App, how easy it is to develop a simple mobile app as soon as you have data.

Scraping gives you scale; Google couldn't use forms

While on the subject of forms, let's consider how they affect the growth of a product. Imagine for a second Google founders creating the first version of their engine incorporating a form that every webmaster has to fill, and copy-paste the text for every page on their website. They should then accept the license agreement to allow Google to process, store, and present their content while pocketing most of the advertising profits. Can you imagine the incredible amount of time and effort required to explain the vision and convince people to get involved in this process? Even if the market was starving for an excellent search engine (as it proved to be the case), this engine wouldn't be Google because its growth would be extremely slow. Even the most sophisticated algorithms wouldn't be able to offset the lack of data. Google uses web crawlers that move through links from page to page, filling their massive databases. Webmasters don't have to do anything at all. Actually, it requires a bit of effort to prevent Google from indexing your pages.
The idea of Google using forms might sound a bit ridiculous, but how many forms does a typical website require a user to fill? A login form, a new listing form, a checkout form, and so on. How much do those forms really cost by hindering application's growth? If you know your audience/customers enough, it is highly likely that you have a clue on the other websites they are typically using, and might already have an account with. For example, a developer will likely have a Stack Overflow and a GitHub account. Could you—with their permission—scrape those sites as soon as they give you their username, and auto-fill their photos, their bio, and a few recent posts? Can you perform some quick text analytics on the posts they are mostly interested in, and use it to adapt your site's navigation structure and suggested products or services? I hope you can see how replacing forms with automated data scraping can allow you to better serve your audience, and grow at web-scale.

**Discovering and integrating into your ecosystem**

Scraping data naturally leads you to discover and consider your relationship with the communities related to your endeavors. When you scrape a data source, naturally some questions arise: Do I trust their data? Do I trust the companies who I get data from? Should I talk to them to have a more formal cooperation? Am I competing or cooperating with them? How much would it cost me to get these data from another source? Those business risks are there anyway, but the scraping process helps us become aware of them earlier, and develop mitigation strategies.

You will also find yourself wondering what do you give back to those websites or communities? If you give them free traffic, they will likely be happy. On the other hand, if your application doesn't provide some value to your source, maybe your relationship is a bit ephemeral unless you talk to them and find a way to cooperate. By getting data from various sources, you are primed to develop products friendlier to the existing ecosystem that respect established market players, disrupting only when it's worth the effort. Established players might also help you grow faster—for example, if you have an application that uses data feeds from two or three distinct ecosystems of a hundred thousand users each, your service might end up connecting three hundred thousand users in a creative way which benefits everybody. For example, if you create a start-up that combines a rock music and a t-shirt printing community, you end up with a mixture of two ecosystems, and both you and the communities will likely benefit and grow.
Being a good citizen in a world full of spiders

There are a few things one needs to be aware of while developing scrapers. Irresponsible web scraping can be annoying and even illegal in some cases. The two most important things to avoid are denial-of-service (DoS) attack like behavior and violating copyrights.

In the first one, a typical visitor might be visiting a new page every few seconds. A typical web crawler might be downloading tens of pages per second. That is more than ten times the traffic that a typical user generates. This might reasonably make the website owners upset. Use throttling to reduce the traffic you generate to an acceptable user-like level. Monitor the response times, and if you see them increasing, reduce the intensity of your crawl. The good news is that Scrapy provides out-of-the-box implementation of both these functionalities (see Chapter 7, Configuration and Management).

On copyrights, obviously, take a look at the copyright notice of every website you scrape, and make sure you understand what is allowed and what is not. Most sites allow you to process information from their site as long as you don't reproduce them claiming that it's yours. What is nice to have is a User-Agent field on your requests that allows webmasters to know who you are and what you do with their data. Scrapy does this by default by using your BOT_NAME as a User-Agent when making requests. If this is a URL or a name that clearly points to your application, then the webmaster can visit your site, and learn more about how you use their data. Another important aspect is allowing any webmaster to prevent you from accessing certain areas of their website. Scrapy provides functionality (RobotsTxtMiddleware) that respects their preferences as expressed on the web-standard robots.txt file (see an example of that file at http://www.google.com/robots.txt). Finally, it's good to provide the means for webmasters to express their desire to be excluded from your crawls. At the very least, it must be easy for them to find a way to communicate with you and express any concerns.

Laws differ from country to country, and I'm by no means in a position to give legal advice. Please seek professional legal advice if you feel the need before relying too heavily on scraping for your projects. This applies to the entire content of this book.

What Scrapy is not

Finally, it's easy to misunderstand what Scrapy can do for you mainly because the terms Data Scraping and all the related terminology is somewhat fuzzy, and many terms are used interchangeably. I will try to clarify some of these areas to prevent confusion and save you some time.
Scrapy is not Apache Nutch, that is, it's not a generic web crawler. If Scrapy visits a website it knows nothing about, it won't be able to make anything meaningful out of it. Scrapy is about extracting structured information, and requires manual effort to set up the appropriate XPath or CSS expressions. Apache Nutch will take a generic page and extract information, such as keywords, from it. It might be more suitable for some applications and less for others.

Scrapy is not Apache Solr, Elasticsearch, or Lucene; in other words, it has nothing to do with a search engine. Scrapy is not intended to give you references to the documents that contain the word "Einstein" or anything else. You can use the data extracted by Scrapy, and insert them into Solr or Elasticsearch as we do at the beginning of Chapter 9, Pipeline Recipes, but that's just a way of using Scrapy, and not something embedded into Scrapy.

Finally, Scrapy is not a database like MySQL, MongoDB, or Redis. It neither stores nor indexes data. It only extracts data. That said, you will likely insert the data that Scrapy extracts to a database, and there is support for many of them, which will make your life easier. Scrapy isn't a database though, and its outputs could easily be just files on a disk or even no output at all—although I'm not sure how this could be useful.

Summary
In this chapter, we introduced you to Scrapy, gave you an overview of what it can help you with, and described what we believe is the best way to use this book. We also presented several ways with which automated data scraping can benefit you by helping you quickly develop high-quality applications that integrate nicely with existing ecosystems. In the following chapter, we will introduce you to HTML and XPath, two very important web languages that we will use in every Scrapy project.
Where to buy this book
You can buy Learning Scrapy from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.