How To Perform Python Web Scraping in 12 Minutes

A short tutorial on using the AutoScraper Python library with no prior knowledge of HTML or XPath

Byron Dolon Follow

Mar 16 · 10 min read ★

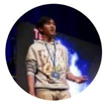


Photo by Miguel A. Amutio on Unsplash.

Web scraping can be an incredibly useful tool for data engineers, data scientists, and more. While there are many publicly available datasets and APIs that you can use as inputs for your projects or businesses, there is also a wealth of freely available information on the web that you can collect using web scraping.

Now, when it comes to the practice of web scraping, there are many different tools and methods across different languages that you can use. A lot of web scraping libraries will require you to have some knowledge of how websites work. You may also have to spend some time learning to use a particular library. For example, the Beautiful Soup library in Python is very capable of extracting data out of HTML files and is therefore widely used in web scraping contexts. However, to use this library, you'll need to have some familiarity with HTML as well as some time to familiarize yourself with the module-specific methods.

I'm not saying that learning HTML and how websites work in general is bad. However, if it's something you're not really interested in and you'd just like to use websites as a source of data, it may be better to use a library like AutoScraper! There are paid solutions for automatic web scraping, but if you have Python knowledge, you can take advantage of this lightweight web scraping library to grab content from a web page without having to know any HTML terms. Instead, all you need is the ability to copy and paste text (more or less).

In this article, I'd like to show you what AutoScraper can do by going through a short practical demonstration of how you can use it to get started with web scraping with no prior knowledge. We'll go through a few examples of getting famous quote-related data only using AutoScraper (and just ten minutes of your time).

If you want to follow along, make sure you have AutoScraper installed (available with pip). Also, for a nice way to check on progress as we go through examples, I use the `pprint` module (data pretty printer) and work through the examples on a Jupyter Notebook. Import these two libraries and then let's get started!

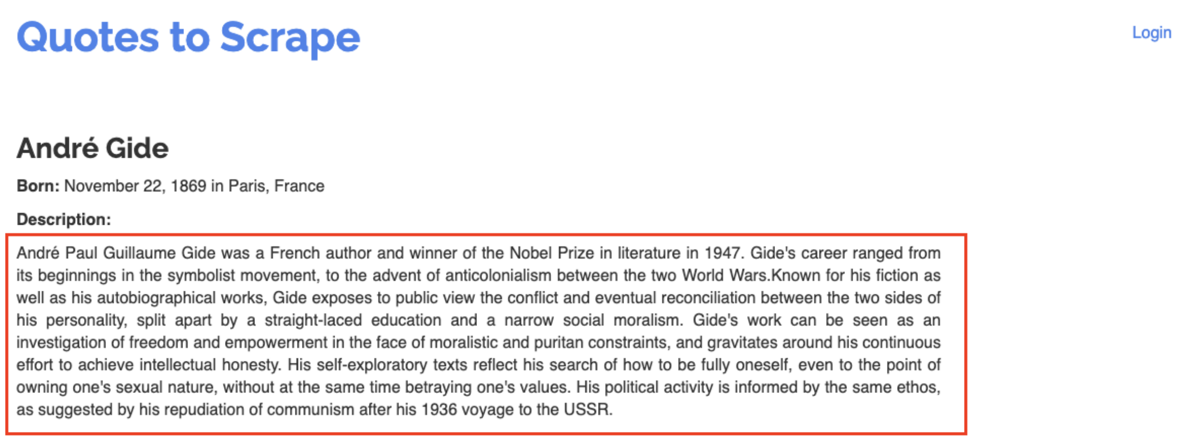
```
from autoscraper import AutoScraper
from pprint import pprint
```

Reminder: You should make sure to check what kind of data (if any) can be scraped from a particular source. One good practice is to check the `robots.txt` *file to see if the site actively allows scraping. If you see that* `robots.txt` *forbids certain user agents (like the ones you would use for web scraping) from* *accessing the site, you would be better off looking for another source of data. You should also check the Terms and Conditions page of the site you're looking to scrape (typically linked on the footer of the site) to see if it allows automated data collection/scraping. Scrape responsibly!*

. . .

Getting One Element From a Web Page

First, let's try to get one thing from the web page. Something useful for someone looking to get information about an author they would like to know more about is the short biography/description.



André Gide

Born: November 22, 1869 in Paris, France

Description:

André Paul Guillaume Gide was a French author and winner of the Nobel Prize in literature in 1947. Gide's career ranged from its beginnings in the symbolist movement, to the advent of anticolonialism between the two World Wars.Known for his fiction as well as his autobiographical works, Gide exposes to public view the conflict and eventual reconciliation between the two sides of his personality, split apart by a straight-laced education and a narrow social moralism. Gide's work can be seen as an investigation of freedom and empowerment in the face of moralistic and puritan constraints, and gravitates around his continuous effort to achieve intellectual honesty. His self-exploratory texts reflect his search of how to be fully oneself, even to the point of owning one's sexual nature, without at the same time betraying one's values. His political activity is informed by the same ethos, as suggested by his repudiation of communism after his 1936 voyage to the USSR.

Source: Quotes to Scrape

Here’s where the copy-and-paste part comes in. The Profile page for each author looks similar for each author, so to allow our AutoScraper to grab the description for every profile URL we pass to it, we need to train it first on one site. Doing that looks like this:

First, we set up a URL for the page we want to scrape (in this case, it's the Profile page). The part of the URL that differs across authors is the author name, so to access the profile page of J.K. Rowling, you'd need to set `desired_author` to `J-K-Rowling`. It’s good to do this at the very beginning instead of having to rewrite the URL for every new author you want to look at.

To make the AutoScraper work on many different sites, we first have to “train” it on a sample site with the data we want to gather. In this case, we’re going for the “Description” data, which is all the text inside the box in the image above. The training data must come in the form of a list, even if you’re only taking one element, so what we need to do is copy and paste the description and assign it to `profile_training_data`.

Then, we create a new instance of the `AutoScraper` class and run the `build` method, providing the `profile_url` (the page we want to scrape) and passing the `profile_training_data` to `wanted_list` (the data we want to gather). This newly defined `AutoScraper` will then learn how to grab the training data you wanted. It will also remember how to do so for similar sites you pass to it.

To test the scraper on other links, let’s run the following code:

Notebook result — authors' profiles

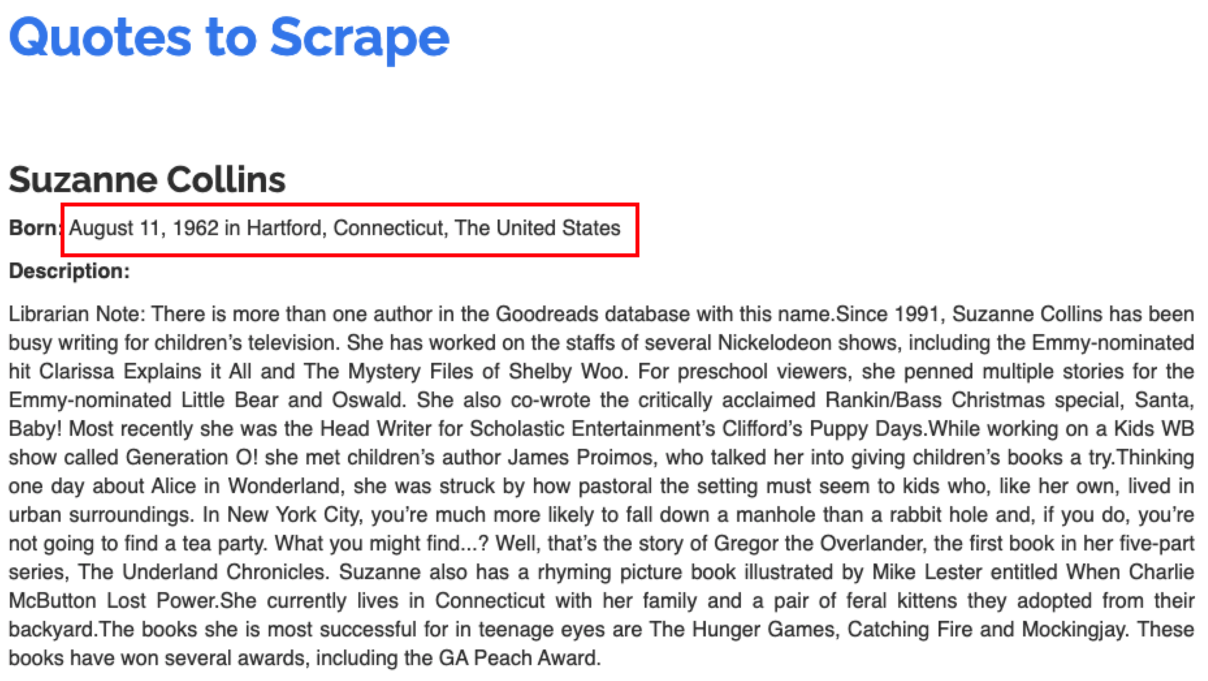
What we've done above is grab the profiles of three new authors: Albert Einstein, J.K. Rowling, and Suzanne Collins. All we did is assign the three authors' names to a list, then loop through that list and use the trained `profile_scraper` to get the profiles of these three authors. For storage, we defined a dictionary to store all the results in, where the keys are the author names and the values are the author descriptions. Of course, you're free to store them in a different manner or set up your own pipeline to move these results into a database (sqlite3, MongoDB, etc.) depending on your use case.

To get the data, we've used the `get_result_exact` method from the `AutoScraper` class that returns a list of the results from the sites that match the rule that `profile_scraper` learned from the training data. Later, we'll see another method that allows for fuzzy matching when scraping data.

. . .

Getting Multiple Elements From a Page

Now that we've covered the basics, let's try another simple example where we just grab multiple things from a web page with AutoScraper. Let's say we want to grab the date of birth and place of birth for each author from their profile page.


Quotes to Scrape

Suzanne Collins

Born: August 11, 1962 in Hartford, Connecticut, The United States

Description:

Librarian Note: There is more than one author in the Goodreads database with this name.Since 1991, Suzanne Collins has been busy writing for children's television. She has worked on the staffs of several Nickelodeon shows, including the Emmy-nominated hit Clarissa Explains it All and The Mystery Files of Shelby Woo. For preschool viewers, she penned multiple stories for the Emmy-nominated Little Bear and Oswald. She also co-wrote the critically acclaimed Rankin/Bass Christmas special, Santa, Baby! Most recently she was the Head Writer for Scholastic Entertainment's Clifford's Puppy Days.While working on a Kids WB show called Generation O! she met children's author James Proimos, who talked her into giving children's books a try.Thinking one day about Alice in Wonderland, she was struck by how pastoral the setting must seem to kids who, like her own, lived in urban surroundings. In New York City, you're much more likely to fall down a manhole than a rabbit hole and, if you do, you're not going to find a tea party. What you might find...? Well, that's the story of Gregor the Overlander, the first book in her five-part series, The Underland Chronicles. Suzanne also has a rhyming picture book illustrated by Mike Lester entitled When Charlie McButton Lost Power.She currently lives in Connecticut with her family and a pair of feral kittens they adopted from their backyard.The books she is most successful for in teenage eyes are The Hunger Games, Catching Fire and Mockingjay. These books have won several awards, including the GA Peach Award.

Source: Quotes to Scrape

The code will look more or less the same as before — only this time, we pass a list of multiple values into the training data. Doing so should look like this:

Then we can test the new scraper on three new authors. Again, the code will look more or less the same as last time, but we’re going to pass an additional argument to the `get_result_exact` method. Doing so will look like this:

Notebook result — using “grouped” for aliases

By setting `grouped=True`, we can see the `rule_id` of each item. This `rule_id` can be thought of as the category label of the returned data. Since we passed multiple values to `wanted_list` earlier, we need to have `grouped=True` to make sure that we can identify the returned results. The returned result from `get_result_exact` will be a dictionary of items with the `rule_id` as the key and a list of the found data points as the values.

It’s great that we can identify the metadata items by distinct groups, but you’ll see that we get arbitrary names for each of the rules. We can fix this by adding an alias for the rules when passing in the training data. Instead of passing a list to the `build` method, we'll be passing a dictionary of values into the `wanted_dict` parameter. This dictionary should have the keys as the alias (category label or name of the data to gather) and then the values as the data to gather (what went into `wanted_list` earlier). Re-training the `metadata_scraper` will look like this:

Then, we can run the AutoScraper on multiple authors using the same format as before, but with one modification. Instead of passing `grouped=True` to the `get_result_exact` method, we pass `group_by_alias=True`. This means that the method will return a dictionary with the aliases defined earlier as keys and a list of matching data as the values. The code to do this is as follows:

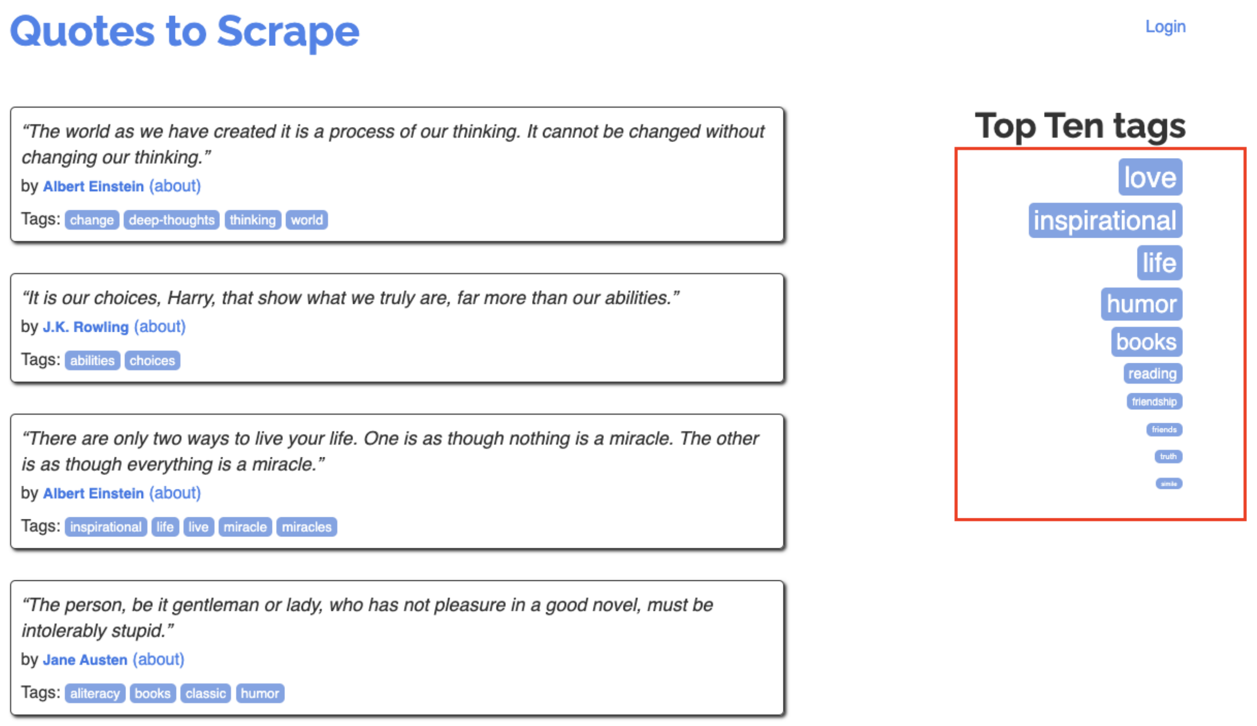
Notebook result — grouped with named aliases

. . .

Scraping Exact Results vs. Scraping Similar Results From a Web Page

So far, we've only been using the `get_result_exact` method to scrape data. The AutoScraper library also features the `get_result_similar` method, which can help if you want to get additional results from a web page that may not be in the exact same format as your training data.

To test this out, let's try to grab the top tags from the main Quotes to Scrape page:


Quotes to Scrape

Login

"The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking."
by Albert Einstein (about)
Tags: change deep-thoughts thinking world

"It is our choices, Harry, that show what we truly are, far more than our abilities."
by J.K. Rowling (about)
Tags: abilities choices

"There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle."
by Albert Einstein (about)
Tags: inspirational life live miracle miracles

"The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid."
by Jane Austen (about)
Tags: aliteracy books classic humor

Top Ten tags

love
inspirational
life
humor
books
reading
friendship
friends
truth
simile

Source: Quotes to Scrape

Here, we'll need to copy and paste the top ten tags found in the list on the right side of the page and assign them to a list for our training data. As before, we'll be using `wanted_dict` so that we can assign an alias to the `rule_id` that AutoScraper will learn. Training this top tag scraper will look like this:

Now that we have trained the `tags_scraper`, let's see what happens when we call `get_result_exact` on the same site we used to train the AutoScraper:

```
tags_scraper.get_result_exact(tag_url)
```

Notebook result — get_result_exact

As you can see, the code correctly returns the top tags that you can see on the main page. However, there are also tags for each visible post on the page, which may be another interesting data set to grab. To also get this data, we can do the following:

```
tags_scraper.get_result_similar(tag_url, grouped=True)
```

Notebook result — get_result_similar

Above, we used the `get_result_similar` method to get a list of all the tags on the page, even though we didn't include the post tags on the page in our training data. Depending on your use case, this method may be useful to grab more data outside the exact scope of the data you used to train the AutoScraper.

. . .

Customizing the AutoScraper

As a note, you can also take a look at all the rules your scraper learned when you train it without having to get results from it. To do so, we'll take a look at the `stack_list` of the `tags_scraper` that we already trained like so:

```
existing_rules = [rule["stack_id"] for rule in
tags_scraper.stack_list]
pprint(existing_rules)
```

Notebook result — Stack list names (AutoScraper learned rules)

Above, we're simply finding the names of the rules the `tags_scraper` has learned by using some list comprehension to put them all in a list. The output shows that the scraper has learned many rules, but if you look at the output from `tags_scraper.get_result_similar`, you'll see that each rule contains the same data. You can edit the scraper so it only contains the rules you need by either using the `remove_rules` or the `keep_rules` method. All you need to do is pass a list of rules you want to remove or rules you want to keep for the respective method.

In our case, since we would only like to take the top tags from the page, let's edit out the rules that return results with more than one value. If you scroll through the earlier output from `tags_scraper`, you'll find that the top tags are present in just the rules with a single-value list. We can neatly accomplish this with another list comprehension snippet:

Then, to check that we've removed all but one rule, let's run the same list comprehension snippet we used before to look at all the rules that currently exist in `tags_scraper`:

```
edited_existing_rules = [rule["stack_id"] for rule in
tags_scraper.stack_list]
pprint(edited_existing_rules)
```

Notebook result — Stack list names (AutoScraper learned rules after edits)

Voila! We've edited the `tags_scraper` to now only have our desired list of rules.

. . .

Making Trained Scrapers Reusable

After going through the work of training an AutoScraper, it makes sense to have a way to store the trained scraper for future use. Doing so will be a lot more efficient than having to retrain a scraper every time you need to run it.

Using the `save` method will serialize the rules (which we know are saved in the `stack_list` assigned to the AutoScraper that we initialized) as JSON and save them to a file path you specify. Then, when you want to use the saved scraper again, you can use the `load` method to replace the existing `stack_list` of the instance of the `AutoScraper` class you call it on. This can be done simply like this:

```
# saving the tags_scraper we trained before
tags_scraper.save("tags_scraper_trained")


# fresh session might look like
tags_scraper_trained = AutoScraper()
tags_scraper_trained.load("tags_scraper_trained")
```

Now, we can use the `tags_scraper_trained` instance as we did earlier with either the `get_result_exact` or `get_result_similar` method. Saving the `AutoScraper` instance may be a good first step once you’ve gone through the work of training and customizing your scraper in an exploratory phase. Then, when you’re ready to scrape data at regular intervals, you can call the `load` method at the very beginning of your code to use the saved AutoScraper.

. . .

Conclusion

And that’s all!

Web scraping can be a useful practice to add to your toolkit, whether you’re a data engineer looking for a way to ingest data without an API or a data scientist who wants to collect their own data for a machine learning project instead of using an existing dataset.

If you're fairly new to web scraping or aren't too interested in learning things like HTML or XPath, AutoScraper might be the tool that you can rely on. Automated tools do have their limitations, so depending on how much you need to customize your web scraping operations, you may find that you need to learn to use other libraries that are more hands-on.

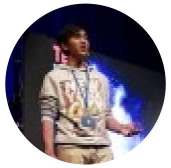
Regardless, I hope you've found this short introduction to the AutoScraper library useful. Best of luck on your (responsible) data collection adventures!

Thanks to Anupam Chugh.

Programming Data Science Python Web Scraping Software Development

70

WRITTEN BY



Byron Dolon

Follow

Medium has become a place to store my “how to do tech stuff” type guides. Come check out my notes on data-related shenanigans!

Better Programming

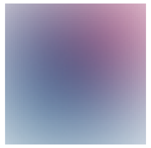
Follow

Advice for programmers. Here’s why you should subscribe: https://bit.ly/bp-subscribe

More From Medium

The 7 Best Thematic Map Types for Geospatial Data

Abdishakur in Towards Data Science

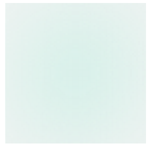


The Full Picture of America’s Obsession with Avocados.

Gwynn Group

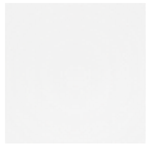
The Case for Design Technology at Noodle.ai

Tony Chu



A Retrospective with the Archives Unleashed Project

Samantha Fritz in Archives Unleashed

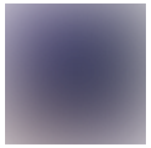


[Spotlight] CARE Principles

Open Data Charter in opendatacharter

Building a recommendation engine for music and podcast Part 1: Plan

HARDI RATHOD in Chatbots Life



Send Google Analytics Hit Level Data to BigQuery

muffaddal qutbuddin in Towards Data Science

Hidden tricks for running AutoML experiment from Azure Machine Learning SDK

Lida Ghahremanlou in Towards Data Science