



Practical Introduction to Web Scraping in Python

by [Colin O'Keefe](#) ⌚ Jan 23, 2018 💬 [21 Comments](#) 🔖 [basics](#) [web-scraping](#)

Table of Contents

- [Web Scraping Basics](#)
- [Setting Up Your Python Web Scraper](#)
- [Making Web Requests](#)
- [Wrangling HTML With BeautifulSoup](#)
- [Using BeautifulSoup to Get Mathematician Names](#)
- [Getting the Popularity Score](#)
- [Putting It All Together](#)
- [Conclusion & Next Steps](#)

A Peer-to-Peer Learning Community for Python Enthusiasts...Just Like You

[pythonistacafe.com](#)

Web Scraping Basics

What is web scraping all about?

Imagine that one day, out of the blue, you find yourself thinking “Gee, I wonder who the five most popular mathematicians are?”

You do a bit of thinking, and you get the idea to use [Wikipedia’s XTools](#) to measure the popularity of a mathematician by equating popularity with pageviews. For example, look at the page on [Henri Poincaré](#). There, you can see that Poincaré’s pageviews for the last 60 day:

Next, you Google “famous mathematicians’ names as well as a web scraper”. You find a list of web scrapers. Now what?

This is where Python and web scraping come in. You select some of that data, and passing

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python

...with a fresh **Python Trick** code snippet every couple of days:

Email Address

[Send Python Tricks »](#)

In this tutorial, you will be writing a Python program that downloads the list of 100 mathematicians and their XTools pages, selects data about their popularity, and finishes by telling us the top 5 most popular mathematicians of all time! Let’s get started.

Setting Up Your Python Web Scraper

You will be using Python 3 and Python [virtual environments](#) throughout the tutorial. Feel free to set things up however you like. Here is how I tend to do it:

Shell

```
$ python3 -m venv venv
$ . ./venv/bin/activate
```

You will need to install only these two packages:

- [requests](#) for performing your HTTP requests
- [BeautifulSoup4](#) for handling all of your HTML processing

Let’s install these dependencies with pip:

Shell

```
$ pip install requests BeautifulSoup4
```

Finally, if you want to follow along, fire up your favorite text editor and create a file called `mathematicians.py`. Get started by including these `import` statements at the top:

Python

```
from requests import get
from requests.exceptions import RequestException
from contextlib import closing
from bs4 import BeautifulSoup
```

Improve Your Python

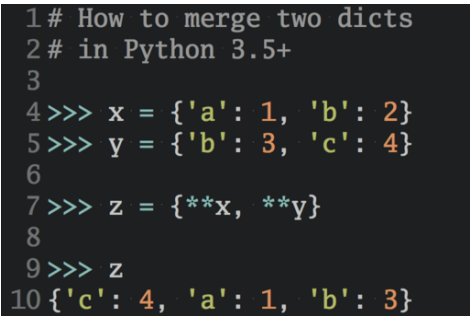
Making Web Requests

Your first task will be to download web pages. The `requests` package comes to the rescue. It aims to be an easy-to-use tool for doing all things HTTP in Python, and it doesn’t dissappoint. In this tutorial, you will need only the `requests.get()` function, but you should definitely checkout the [full documentation](#) when you want to go further.

First, here’s your function:

Python

```
def get_page(url):
    """
    Get the page content from the url.
    """
    try:
        with closing(get(url)) as response:
            return response.text
    except RequestException:
        return None
```



Improve Your Python

...with a fresh  **Python Trick**  code snippet every couple of days:

[Send Python Tricks »](#)

```
def simple_get(url):
    """
    Attempts to get the content at `url` by making an HTTP GET request.
    If the content-type of response is some kind of HTML/XML, return the
    text content, otherwise return None.
    """
    try:
        with closing(get(url, stream=True)) as resp:
            if is_good_response(resp):
                return resp.content
            else:
                return None

    except RequestException as e:
        log_error('Error during requests to {0} : {1}'.format(url, str(e)))
        return None

def is_good_response(resp):
    """
    Returns True if the response seems to be HTML, False otherwise.
    """
    content_type = resp.headers['Content-Type'].lower()
    return (resp.status_code == 200
            and content_type is not None
            and content_type.find('html') > -1)

def log_error(e):
    """
    It is always a good idea to log errors.
    This function just prints them, but you can
    make it do anything.
    """
    print(e)
```

Improve Your Python

The `simple_get()` function accepts a single `url` argument. It then makes a GET request to that URL. If nothing goes wrong, you end up with the raw HTML content for the page you requested. If there were any problems with your request (like the URL is bad, or the remote server is down), then your function returns `None`.

You may have noticed the use of the `closing()` function in your definition of `simple_get()`. The `closing()` function ensures that any network resources are freed when they go out of scope in that `with` block. Using `closing()` like that is good practice and helps to prevent fatal errors and network timeouts.

You can test `simple_get()` like this:

Python>>>

```
>>> from mathematicians import simple_get
>>> raw_html = simple_get('https://realpython.com/blog/')
>>> len(raw_html)
33878

>>> no_html = simple_get('https://realpython.com/blog/nope-not-gonna-find-it')
>>> no_html is None
True
```

Wrangling HTML With

Once you have raw HTML in front of you BeautifulSoup. The BeautifulSoup const document’s structure. The object includ content.

Consider the following quick and contriv

HTML

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python

...with a fresh  **Python Trick**  code snippet every couple of days:

Email Address

Send Python Tricks »

```
<!DOCTYPE html>
<html>
<head>
  <title>Contrived Example</title>
</head>
<body>
<p id="eggman"> I am the egg man </p>
<p id="walrus"> I am the walrus </p>
</body>
</html>
```

If the above HTML is saved in the file `contrived.html`, then you can use `BeautifulSoup` like this:

Python>>>

```
>>> from bs4 import BeautifulSoup
>>> raw_html = open('contrived.html').read()
>>> html = BeautifulSoup(raw_html, 'html.parser')
>>> for p in html.select('p'):
...     if p['id'] == 'walrus':
...         print(p.text)

'I am the walrus'
```

Breaking down the example, you first parse the raw HTML by passing it to the `BeautifulSoup` constructor. `BeautifulSoup` accepts multiple back-end parsers, but the standard back-end is `'html.parser'`, which you supply here as the second argument. (If you neglect to supply that `'html.parser'`, then the code will still work, but you will see a warning print to your screen.)

The `select()` method on your `html` object lets you use [CSS selectors](#) to locate elements in the document. In the above case, `html.select('p')` returns a list of paragraph elements. Each `p` has HTML attributes that you can access like a dict. In the line `if p['id'] == 'walrus'`, for example, you check if the `id` attribute is equal to the string `'walrus'`, which corresponds to `<p id="walrus">` in the HTML.

Improve Your Python

Using BeautifulSoup to Get Mathematician Names

Now that you have given the `select()` method in `BeautifulSoup` a short test drive, how do you find out what to supply to `select()`? The fastest way is to step out of Python and into your web browser’s developer tools. You can use your browser to examine the document in some detail. I usually look for `id` or `class` element attributes or any other information that uniquely identifies the information I want to extract.



To make matters concrete, turn to the [list of mathematicians](#) you saw earlier. If you spend a minute or two looking at this page’s source, you can see that each mathematician’s name appears inside the text content of an `` tag. To make matters even simpler, `` tags on this page seem to contain nothing but names of mathematicians.

Here’s a quick look with Python:

Python>>>

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python

...with a fresh  Python Trick 

code snippet every couple of days:

Email Address

Send Python Tricks »

```
>>> raw_html = simple_get('http://www.fabpedigree.com/james/mathmen.htm')
>>> html = BeautifulSoup(raw_html, 'html.parser')
>>> for i, li in enumerate(html.select('li')):
    print(i, li.text)

0  Isaac Newton
   Archimedes
   Carl F. Gauss
   Leonhard Euler
   Bernhard Riemann

1  Archimedes
   Carl F. Gauss
   Leonhard Euler
   Bernhard Riemann

2  Carl F. Gauss
   Leonhard Euler
   Bernhard Riemann

3  Leonhard Euler
   Bernhard Riemann

4  Bernhard Riemann

# 5 ... and many more...
```

The above experiment shows that some of the `` elements contain multiple names separated by newline characters, while others contain just a single name. With this information in mind, you can write your function to extract a single list of names:

Python

```
def get_names():
    """
    Downloads the page where the list of mathematicians
    and returns a list of strings, one per mathematician
    """
    url = 'http://www.fabpedigree.com/james/mathmen.htm'
    response = simple_get(url)

    if response is not None:
        html = BeautifulSoup(response, 'html.parser')
        names = set()
        for li in html.select('li'):
            for name in li.text.split('\n'):
                if len(name) > 0:
                    names.add(name.strip())
        return list(names)

    # Raise an exception if we failed to get any data from the url
    raise Exception('Error retrieving contents at {}'.format(url))
```

The `get_names()` function downloads the page and iterates over the `` elements, picking out each name that occurs. Next, you add each name to a Python set, which ensures that you don't end up with duplicate names. Finally, you convert the set to a list and return it.

Getting the Popularity

Nice, you're nearly done! Now that you have a function you write is similar to the function that extracts out an integer value from the page.

Again, you should first check out an [example](#) inside an `<a>` element, and the `href` attribute. That's all the information you need to write a function that

Python

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python

...with a fresh  **Python Trick** 
code snippet every couple of days:

Email Address

Send Python Tricks »


```
def get_hits_on_name(name):
    """
    Accepts a `name` of a mathematician and returns the number
    of hits that mathematician's Wikipedia page received in the
    last 60 days, as an `int`
    """
    # url_root is a template string that is used to build a URL.
    url_root = 'https://xtools.wmflabs.org/articleinfo/en.wikipedia.org/{}'.format(name)
    response = simple_get(url_root)

    if response is not None:
        html = BeautifulSoup(response, 'html.parser')

        hit_link = [a for a in html.select('a')
                     if a['href'].find('latest-60') > -1]

        if len(hit_link) > 0:
            # Strip commas
            link_text = hit_link[0].text.replace(',', '')
            try:
                # Convert to integer
                return int(link_text)
            except:
                log_error("couldn't parse {} as an `int`".format(link_text))

    log_error('No pageviews found for {}'.format(name))
    return None
```

Putting It All Together

You have reached a point where you can finally find out which mathematician is most beloved by the public! The plan is simple:

- Get a list of names
- Iterate over the list to get a “popularity score” for each name
- Finish by sorting the names by popularity

Improve Your Python

Simple, right? Well, there’s one thing that hasn’t been mentioned yet: errors.

Working with real-world data is messy, and trying to force messy data into a uniform shape will invariably result in the occasional error jumping in to mess with your nice clean vision of how things ought to be. Ideally, you would like to keep track of errors when they occur in order to get a better sense of the of quality your data.

For your present purposes, you will track instances in which you could not find a popularity score for a given mathematician’s name. At the end of the script, you will print a message showing the number of mathematicians who were left out of the rankings.

Here’s the code:

Python

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python



...with a fresh **Python Trick** code snippet every couple of days:

Email Address

Send Python Tricks »

```
if __name__ == '__main__':
    print('Getting the list of names....')
    names = get_names()
    print('... done.\n')

    results = []

    print('Getting stats for each name....')

    for name in names:
        try:
            hits = get_hits_on_name(name)
            if hits is None:
                hits = -1
            results.append((hits, name))
        except:
            results.append((-1, name))
            log_error('error encountered while processing '
                    '{}', skipping'.format(name))

    print('... done.\n')

    results.sort()
    results.reverse()

    if len(results) > 5:
        top_marks = results[:5]
    else:
        top_marks = results

    print('\nThe most popular mathematicians are:\n')
    for (mark, mathematician) in top_marks:
        print('{} with {} pageviews'.format(mathematician, mark))

    no_results = len([res for res in results if res[0] == -1])
    print('\nBut we did not find results for '
        '{} mathematicians on the list'.format(no_results))
```

Improve Your Python

That’s it!

When you run the script, you should see at the following report:

Shell

The most popular mathematicians are:

Albert Einstein with 1089615 pageviews
Isaac Newton with 581612 pageviews
Srinivasa Ramanujan with 407141 pageviews
Aristotle with 399480 pageviews
Galileo Galilei with 375321 pageviews

But we did not find results for 19 mathematicians on our list

Conclusion & Next Steps

Web scraping is a big field, and you have just finished a brief tour of that field, using Python as your guide. You can get pretty far using just requests and BeautifulSoup, but even if you followed along, you may have come up with four questions:

- What happens if page content load fails? (see [Python API](#).)
- How do I write a web spider or search engine?
- What is this [Scrapy](#) thing I keep hearing about?

These are topics for another post... Keep an eye on our blog for more. We'll also be using a headless browser to deal with dynamic content.

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```



Improve Your Python

...with a fresh  **Python Trick** 
code snippet every couple of days:

Send Python Tricks »

Don't miss the follow up tutorial: [Click here to join the Real Python Newsletter](#) and you'll know when the next installment comes out.

Until then, happy scraping!

 Python Tricks 

Get a short & sweet **Python Trick** delivered to your inbox every couple of days. No spam ever. Unsubscribe any time. Curated by the Real Python team.

Email Address

Send Me Python Tricks »

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

About Colin O'Keefe



Improve Your Python
Colin is a freelance Software Engineer who loves the challenge and good engineering.
[» More about Colin](#)

Each tutorial at Real Python is created by a team of developers so that it meets our high quality standards. The team members who worked on this tutorial are:



[Aldren](#)



[Dan](#)



[Joanna](#)

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python

...with a fresh  **Python Trick** 
code snippet every couple of days:

Email Address

Send Python Tricks »

What Do You Think?

Real Python Comment Policy: The most useful comments are those written with the goal of learning from or helping out other readers—after reading the whole article and all the earlier comments. Complaints and insults generally won’t make the cut here.

21 Comments

Real Python


1 Login

Recommend 7

Tweet

Share

Sort by Best




Join the discussion...

LOG IN WITH


OR SIGN UP WITH DISQUS ?

Name

 **wlbentley** • 9 months ago


in get_hits_on_name, you pass name to the url_root. But the names are in text format with spaces. Does the API accept spaces within the URL? They don't need to be converted to URL encoding?

1 ^ | v • Reply • Share ›

 **soon_to_be_tenant** ➔ wlbentley • 7 months ago

I second this. Why does this code not break given spaces in the names?


^ | v • Reply • Share ›

 **spaceguy01** ➔ soon_to_be_tenant • 7 months ago

Doesn't the spaces get taken out with this line in get_names() function before it's sent to get_hits_on_name?

```
names.add(name.strip())
```

^ | v • Reply • Share ›

 **Putcher** • 10 months ago


Nice introduction to web scraping.

Improve Your Python

One issue I noticed is in the get_hits_on_name function, the word "couldn't". So I think that either needs to be changed to double quotes or the single quote inside the string needs to be escaped with a backslash.


Regarding the messy data, as well as people with no results there are also some which get incorrect page views - for example "Alan M. Turing" gets 271 page views, while the correct page puts him at 4th with 459,127 views. I found cleaning the data further an enjoyable add-on challenge to the article (mainly by automatically following re-direct and disambiguation pages).

1 ^ | v • Reply • Share ›

 **Dan Bader** Mod ➔ Putcher • 10 months ago


Thanks for the heads up, I just fixed the string quotes on that code example.

^ | v • Reply • Share ›

 **Kusa Manohar** • 10 months ago

Nice one

1 ^ | v • Reply • Share ›

 **rnm** • 7 days ago

I keep getting these results:

Getting the list of names....
... done.

Getting stats for each name....
... done.

The most popular mathematician

Felix Hausdorff with -1 pageview
Henri Poincaré with -1 pageview
Blaise Pascal with -1 pageviews
Pierre de Fermat with -1 pagevie
Carl G. J. Jacobi with -1 pagevie

But we did not find results for 100

Why is this happening? I copied it

^ | v • Reply • Share ›

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python

...with a fresh  **Python Trick** 
code snippet every couple of days:

Email Address

Send Python Tricks »



Will • 3 months ago
Could someone explain the log_error function to me? I don't understand how it identifies the error.
^ | v • Reply • Share ›



Brad Solomon ➔ Will • a month ago
You can also use `logging.exception` for cases like this, to log the exception traceback and preface it with a custom message.
^ | v • Reply • Share ›



Kelly ➔ Will • a month ago
The log_error function does not identify any error, it just prints the string generated from this piece of code

except RequestException as e:
log_error('Error during requests to {0} : {1}'.format(url, str(e)))
return None

and

log_error('No pageviews found for {}'.format(name))
^ | v • Reply • Share ›



Aysuh • 3 months ago
I found an error in retrieving content from the fapeins website. anybody else encountered error?
^ | v • Reply • Share ›



Austin Field • 3 months ago
Great artical. Without developing code, I think ScrapeStorm is a tool to help scrape web data.
^ | v • Reply • Share ›



Milos • 3 months ago
Getting the list of names....
... done.

Getting stats for each name....
No pageviews found for Muhammed al-Khowârizmi
No pageviews found for M. E. Camille Jordan
No pageviews found for Bháscara (II) Áchárya
No pageviews found for F.E.J. Émile Borel
No pageviews found for Leonardo `Fibonacci'
No pageviews found for F. L. Gottlob Frege
No pageviews found for Hermann K. H. Weyl
No pageviews found for Alhazen ibn al-Haytham
No pageviews found for Gottfried W. Leibniz
^ | v • Reply • Share ›

Improve Your Python



Arman Asryan • 9 months ago
I keep receiving the following results:

The most popular mathematicians are:

William R. Hamilton with -1 page views
Peter G. L. Dirichlet with -1 page views
Panini of Shalatula with -1 page views
Omar al-Khayyám with -1 page views
Muhammed al-Khowârizmi with -1 page views

But we did not find results for 19 mathematicians on the list

What is the reason behind this strange result ?
^ | v • Reply • Share ›



Colin O ➔ Arman Asryan • 9 months ago
Hi, you can see in the abc
found for a mathematiciar
then reversed. If you forgo
would all be mathematicia

Also, ensure you are usin

try running <https://github.com>
^ | v • Reply • Share ›

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python



...with a fresh **Python Trick**
code snippet every couple of days:

Email Address

Send Python Tricks »



jshell123 • 10 months ago
Nice, thank you. I'll definitely put
^ | v • Reply • Share ›



Sergey Ivanov • 10 months ago

Thanks!

^ | v • Reply • Share ›



NC • 10 months ago

I'm getting a KeyError for 'id' when trying to use BeautifulSoup select on an HTML I got from a URL with:

(urllib.request.urlopen(url)).read()

Any idea how to fix the error?

^ | v • Reply • Share ›



LordOfTheMorning • 10 months ago

Doesn't feel like an honorable job scraping away at the web.

^ | v • Reply • Share ›



xak47d ➔ LordOfTheMorning • 10 months ago

That's basically what every search engine does

2 ^ | v • Reply • Share ›



Dan Bader Mod ➔ LordOfTheMorning • 10 months ago

Well, someone has to do it... ;-)

1 ^ | v • Reply • Share ›

ALSO ON REAL PYTHON

Python "while" Loops (Indefinite Iteration)

5 comments • 11 days ago



Dan Bader — Hey Saurabh, put yourself in the shoes of a total beginner or someone new to Python. This series covers the ...

Python, Boto3, and AWS S3: Demystified

8 comments • a month ago



PeterQuirk — Good article!Note that some of the steps can be simplified by installing the AWS CLI for your platform. Setting ...

Splitting, Concatenating, and Joining Strings in Python

7 comments • 2 months ago



Joe Madaus — Thank you! Makes sense -- I was looking more at the 2nd example you gave but now that I read your first one -- ...

Setting Up Python for Machine Learning on Windows

4 comments • 18 days ago



Andreas — Thanks for this writeup. I found the environment setup very helpful. Windows users are

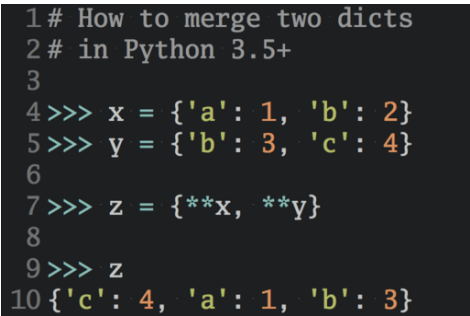
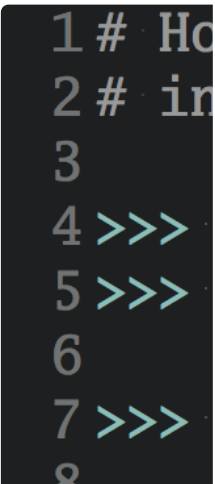
Improve Your Python

[Subscribe](#) [Add Disqus to your site](#)[Add Disqus](#)[Add](#) [Disqus' Privacy Policy](#)[Privacy Policy](#)[Privacy Policy](#)

Keep Reading

[basics](#) [web-scraping](#)

— FREE Email Series —



Improve Your Python

...with a fresh **Python Trick** code snippet every couple of days:

[Send Python Tricks »](#)

```
8
9 >>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

Email...

Get Python Tricks »

No spam. Unsubscribe any time.

All Tutorial Topics

- [advanced](#)[api](#)[basics](#)[best-practices](#)[community](#)[databases](#)[data-science](#)[devops](#)[django](#)[docker](#)[flask](#)[front-end](#)[intermediate](#)[machine-learning](#)[python](#)[testing](#)[tools](#)[web-dev](#)[web-scraping](#)

A Peer-to-Peer
Learning Community
for Python
Enthusiasts...
Just Like You

pythonistacafe.com



Improve Your Python

Table of Contents

- [Web Scraping Basics](#)
- [Setting Up Your Python Web Scraper](#)
- [Making Web Requests](#)
- [Wrangling HTML With BeautifulSoup](#)
- [Using BeautifulSoup to Get Mathematician Names](#)
- [Getting the Popularity Score](#)
- [Putting It All Together](#)
- [Conclusion & Next Steps](#)



```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10 {'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python

...with a fresh **Python Trick** code snippet every couple of days:

Email Address

Send Python Tricks »

Improve Your Python

```
1# How to merge two dicts
2# in Python 3.5+
3
4>>> x = {'a': 1, 'b': 2}
5>>> y = {'b': 3, 'c': 4}
6
7>>> z = {**x, **y}
8
9>>> z
10{'c': 4, 'a': 1, 'b': 3}
```

Improve Your Python



...with a fresh **Python Trick** code snippet every couple of days:

Email Address

Send Python Tricks »