

Using requests, BeautifulSoup, and pandas to
scrape the marathon and half-marathon results

<https://www.runrocknroll.com/Events/Nashville/The-Races>



Saturday, April 27, 2019

To find your results the quickest, please enter your bib number.

First Name	- Gender -
Last Name	Runner Number (bib)
	- Division -

Search

[Submit a Correction](#)

Leaderboards

Top Females		
Gender PL	Name	Time
1	Carson Davis	03:06:25
2	Leah Frazier Allen	03:07:51
3	Emily Soppe	03:11:16
4	Allison Koch	03:15:27
5	Tara Austin	03:15:54
6	Lisa Holding Eagle	03:21:37
7	Rachel Polk	03:22:05
8	Amanda Jackson	03:25:15
9	Marissa Mchugh	03:25:29
10	Jessica Spicola	03:25:43

Top Men		
Gender PL	Name	Time
2	Scott Wietecha	02:34:59
3	Jordan Wilson	02:35:24
4	Steelton Flynn	02:39:59
5	Thomas Ellis	02:42:09
6	Nicholas Tseffos	02:48:42
7	Satoshi Mitsumori	02:50:33
8	Harrison Kieffer	02:51:18
9	Steven Forte	02:54:34
10	Grant Rice	02:55:49
11	Andrew Fisher	02:56:05

Examining the page for 2019 marathon results (<https://www.runrocknroll.com/Events/Nashville/The-Races/Marathon/2019-Results>), we see a search section at the top and two small tables at the bottom. **Using pandas to read this webpage won't get us all the data underlying the search.**

Let's take a closer look at what's happening with the search.

Saturday, April 27, 2019

To find your results the quickest, please enter your bib number.

First Name	- Gender -
Last Name	Runner Number (bib)
	- Division -

Search

[Submit a Correction](#)

Leaderboards

Top Females		
Gender PL	Name	Time
1	Carson Davis	03:06:25
2	Leah Frazier Allen	03:07:51
3	Emily Soppe	03:11:16
4	Allison Koch	03:15:27
5	Tara Austin	03:15:54
6	Lisa Holding Eagle	03:21:37
7	Rachel Polk	03:22:05
8	Amanda Jackson	03:25:15
9	Marissa Mchugh	03:25:29
10	Jessica Spicola	03:25:43

Top Men		
Gender PL	Name	Time
2	Scott Wietecha	02:34:59
3	Jordan Wilson	02:35:24
4	Steelton Flynn	02:39:59
5	Thomas Ellis	02:42:09
6	Nicholas Tseffos	02:48:42
7	Satoshi Mitsumori	02:50:33
8	Harrison Kieffer	02:51:18
9	Steven Forte	02:54:34
10	Grant Rice	02:55:49
11	Andrew Fisher	02:56:05

You can right-click on the webpage and ***Inspect*** the underlying HTML.

Back

Forward

Reload

Save As...

Print...

Cast...

Translate to English

View Page Source

Inspect

Speech





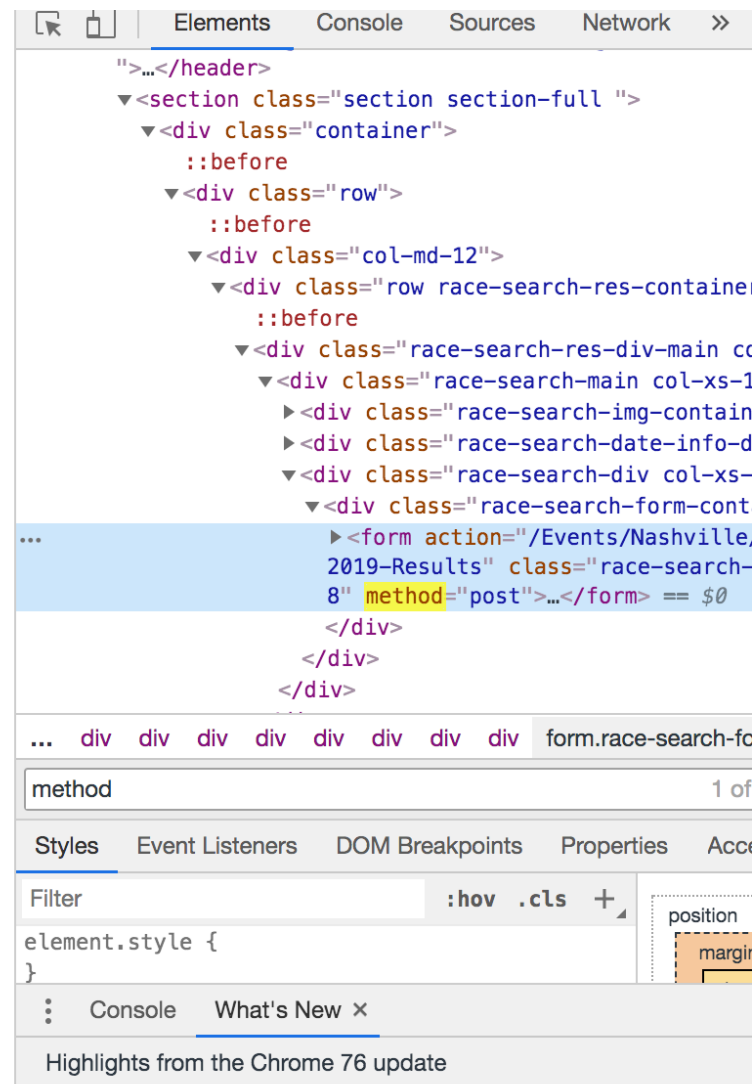
form.race-search-form.col-xs-1 720 × 342.33
2.col-sm-8

First Name	- Gender -
Last Name	Runner Number (bib)
	- Division -
<input type="button" value="Search"/>	
<input type="button" value="Submit a Correction"/>	

Leaderboards

Gender PL	Name	Time
1	Carson Davis	03:06:25
2	Leah Frazier Allen	03:07:51
3	Emily Soppe	03:11:16
4	Allison Koch	03:15:27
5	Tara Austin	03:15:54
6	Lisa Holding Eagle	03:21:37
7	Rachel Polk	03:22:05
8	Amanda Jackson	03:25:15
9	Marissa Mchugh	03:25:29
10	Jessica Spicola	03:25:43

Top Men		
Gender PL	Name	Time
2	Scott Wietecha	02:34:59
3	Jordan Wilson	02:35:24
4	Steelton Flynn	02:39:59
5	Thomas Ellis	02:42:09
6	Nicholas Tseffos	02:48:42
7	Satoshi Mitsumori	02:50:33
8	Harrison Kieffer	02:51:18
9	Steven Forte	02:54:34
10	Grant Rice	02:55:49
11	Andrew Fisher	02:56:05



Searching the
html for ***method***
we see that the
search results are
an http ***post***.

Autocomplete with CSS keyword values

Typing a keyword value like "bold" in the Styles pane now autocompletes to "font-weight: bold".

A new UI for network settings

The "Use large request rows", "Group by frame", "Show overview", and "Capture screenshots" options have moved to the new Network Settings pane.

So we need to use this **requests** method to retrieve our results:

```
result = requests.post(url)
```

for each page

The steps to get a DataFrame from **one page** of results look like this:

- Build a URL by combining the base url with a specific page number
- Use requests.post() to get the results of the post
- Make a [soup](#) from results.text
- Look at the soup to identify the table you want based on one of its attributes (like class)
- Pass the table as a string to pandas read_html()
 - What does that look like? What is the datatype?
- Keep working with the data until you have it a DataFrame

Let's code one page of results to a DataFrame together:

Be sure you have imported BeautifulSoup:

```
from bs4 import BeautifulSoup as BS
```

Paste these two lines of code into a cell (assumes you created the objects for base URLs that were shared in the README)

```
base = urlbase_2019
```

```
page = 99
```

Now you're ready to ***write a function*** to ***iterate*** through all the pages of results for each race and ***create a single DataFrame for each race!***