Web Scraping in Python (and BeautifulSoup)

⭐ Introduction

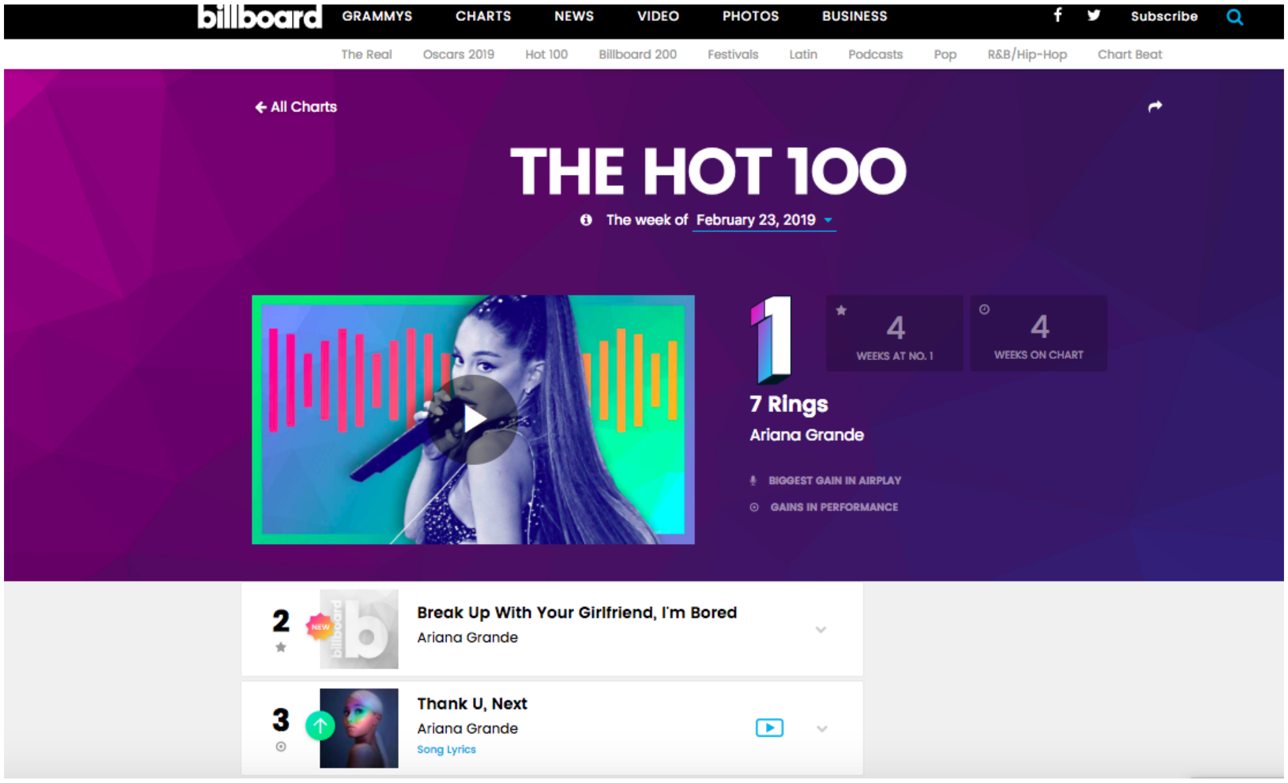
This is a short tutorial to teach you how to scrape websites using the **Python** programming language and the famous library, **BeautifulSoup** using a **Mac** computer.

Reading Time: 15min (if you actually do the coding, then will take longer)

🤔 First, What Is Web Scraping?

Web scraping is the process of extracting information and data from websites. Why do we do web scraping? Unstructured data (HTML format) on the web is transformed into structured data (database or spreadsheet) so that we can analyze and use the data for many purposes.

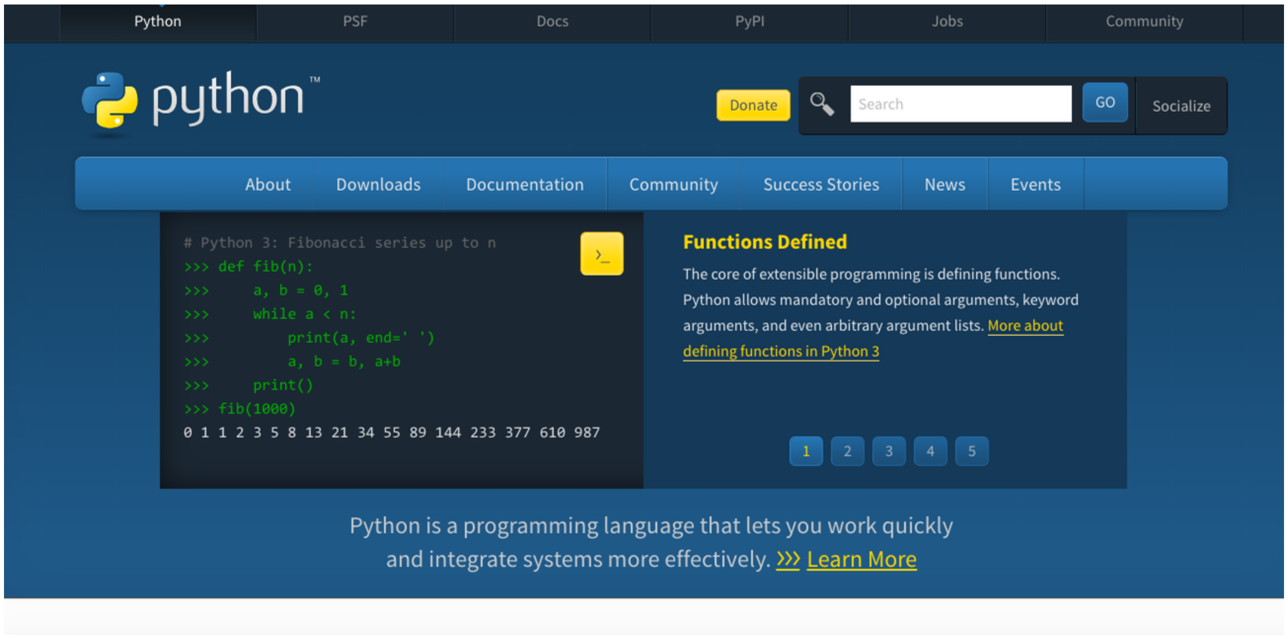
Here's an example. Let's say you want to have a list of the top 100 hit songs from billboard.com:



If you want to store all the songs and their artists name in a spreadsheet, what are you going to do? Copy and paste 100 times? Well, that's one way to do it, but it will sure take time and isn't the most exciting thing to do. If we use web scraping, we can automatically extract the song and artist names and format it nicely all in one easy process.

In these instructions, we will use **Python** as a Scripting language to create a small program that will automatically extract the data we want.

🙄 Why Python?



```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Functions Defined

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. More about defining functions in Python 3

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

- It is a beginner-friendly programming language.
- Python has many awesome libraries and associated softwares that will help you make the task of web scraping easier (such as **BeautifulSoup**, **Request**, **Scrapy**, **Urllib** etc.).

- **If you are new to programming**
  - You at least need to know the basics for using **Python**. I recommend you read this quick Python tutorial: https://www.stavros.io/tutorials/python/
    - this will get you to have some basic understanding of Python
  - If you have the time, there are other useful **Python** tutorials:
    - https://docs.python-guide.org/intro/learning/
    - http://tdc-www.harvard.edu/Python.pdf

⭐ Getting Started ⭐

What you need

- A **Mac** laptop (If you are a Window user, you can still follow my instruction, however, installing process of Python, TextEditor are different so please be aware).

- **Python** version 3
- Terminal window
- Python libraries (I'll teach you how to install these)
  - **BeautifulSoup4** for handling all the HTML processing
  - **Requests** for performing the HTTP requests
- Text editor
  - A text editor is a program that allows you to create and edit a range of programming language files. TLDR: it's where you are going to write the code.
  - You need to download one to follow this tutorial, unless you already have a text editor.
  - My recommended text editors are:
    - **Atom:** https://atom.io/
    - **SublimeText:** https://www.sublimetext.com/
  - Pick one, go to the webpage and follow the download instructions.
  - If you need help with the set up, here are the recommended tutorials to watch
    - **Atom:** https://www.youtube.com/watch?v=DjEuROpsvp4
    - **Sublime Text:** https://www.youtube.com/watch?v=xFciV6Ew5r4

Set Up

- Go to the Python official website https://www.python.org/downloads/ to download Python (must be version 3 or greater)
  - If you are still not sure of how to install Python, watch this video: https://www.youtube.com/watch?v=YYXdXT2l-Gg&vl=en
  - or read this: https://ehmatthes.github.io/pcc/chapter_01/osx_setup.html

- Check that Python is correctly installed by running on your terminal or command prompt
  - In your command line type:

```
python --version
```

    - You should get something like Python 3.7.X (or any other version if you already have installed Python before). If you get an error, that means it is not installed.

- Install Python libraries
  - In your command line type:

```
pip install requests BeautifulSoup4
```

Note: If you fail to execute the above command line, try adding `sudo` at the beginning of the line.

🙈 Let's Write a Program

For this tutorial, we will scrape https://www.billboard.com/charts/hot-100 to get data and save them into a excel spreadsheet. For simplicity, I will only show you how to get the first 100 ranked songs and their artists' names.

---

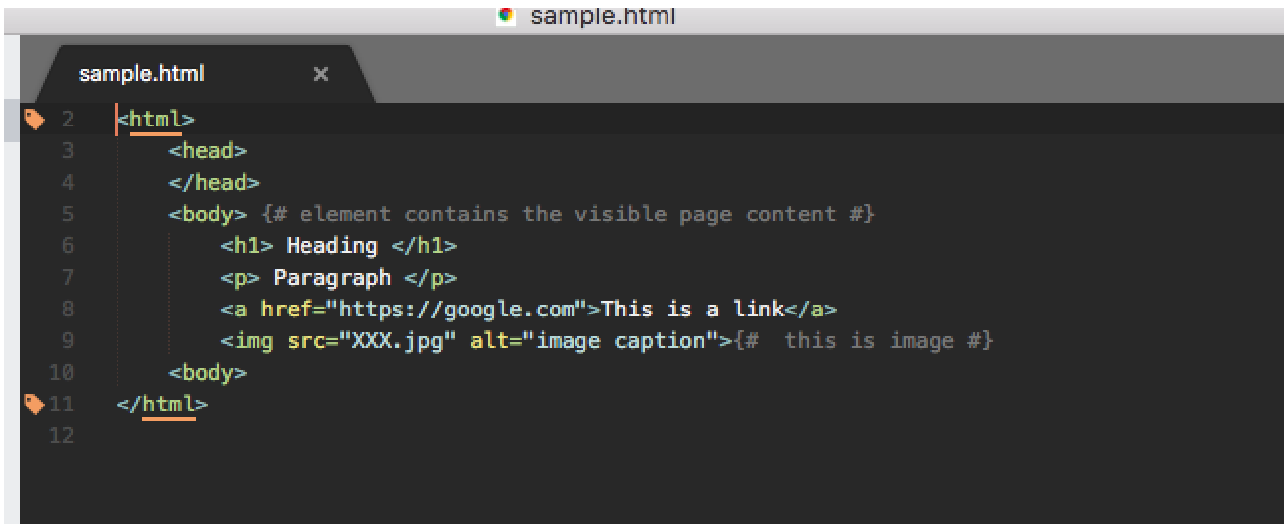
0. Ah, Rules...

1. Read through the website's Terms and Conditions to understand how you can legally use the data. Most sites prohibit you from using the data for commercial purposes.
    - Read Billboard's Terms and Conditions : https://www.billboard.com/p/website-terms-of-use

2. Do not request data from the website too often or aggressively because this may break the website. One per second should be fine.

1. Understand HTML

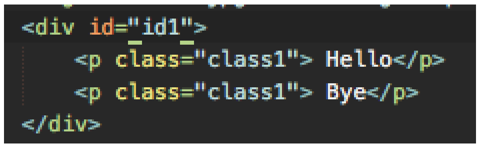
- Before we write a code, we need to understand the basics of HTML.
- **HTML** stands for **H**yper **T**ext **M**arkup **L**anguage.


sample.html

sample.html

```
<html>
    <head>
    </head>
    <body> {# element contains the visible page content #}
        <h1> Heading </h1>
        <p> Paragraph </p>
        <a href="https://google.com">This is a link</a>
        <img src="XXX.jpg" alt="image caption">{#  this is image #}
    <body>
</html>
```

- If you are not familiar with HTML tags, refer to W3Schools Tutorials. It is necessary to understand the basics of HTML in order to successfully web scrape.
- HTML **tags** usually come with **id** or **class** attributes.

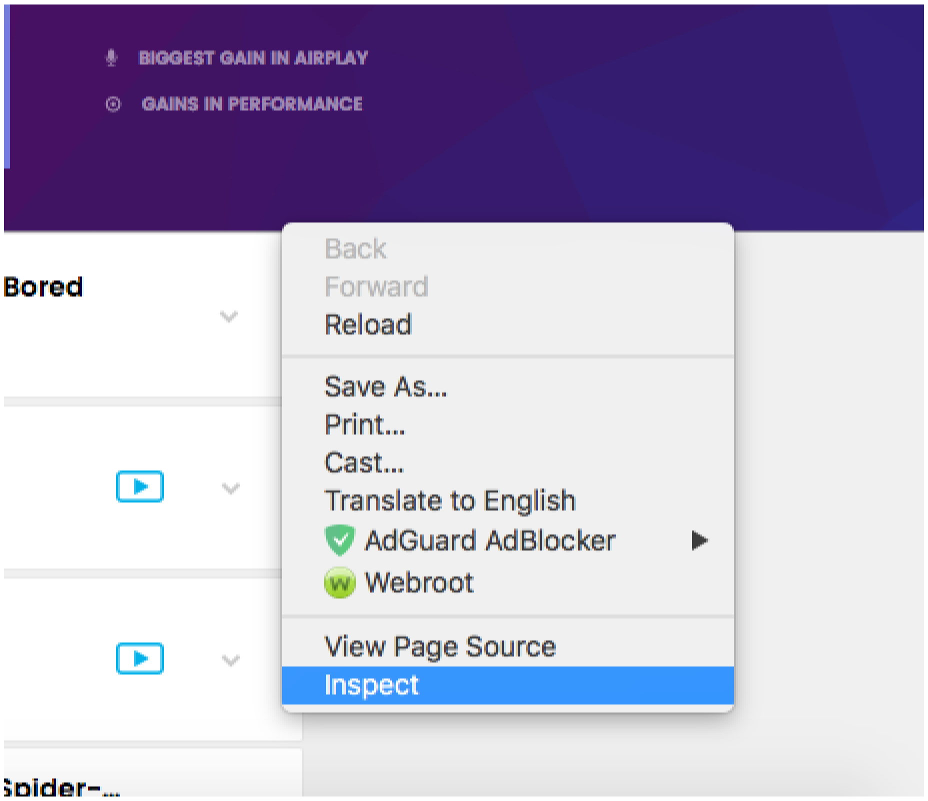


```
<div id="id1">
    <p class="class1"> Hello</p>
    <p class="class1"> Bye</p>
</div>
```

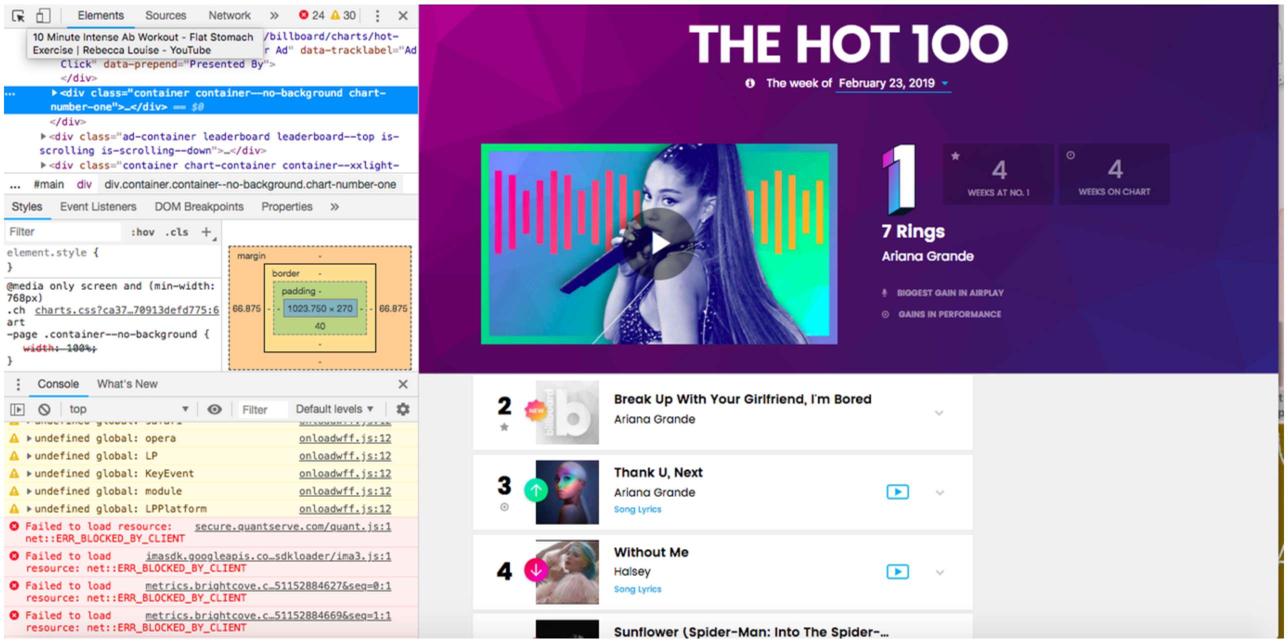
- The **id** attribute specifies a unique **id** for an HTML tag and the value must be UNIQUE (meaning no reuse allowed) within the HTML document.
- The **class** attribute is used to define equal styles for HTML tags within the same class.
- We use tags, ids, and classes to locate the data we want to scrape.

2. Inspect the Website

On the website, right click and click on “**Inspect**”. This allows you to see the raw code behind the site.



Once you've clicked on "Inspect", you should see this console pop up:

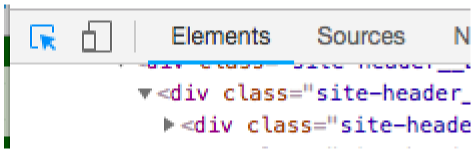


```
10 Minute Intense Ab Workout - Flat Stomach
Exercise | Rebecca Louise - YouTube
        /billboard/charts/hot-
        r Ad" data-tracklabel="Ad
      Click" data-prepend="Presented By">
      </div>
...   ▸<div class="container container--no-background chart-
     number-one">…</div> == $0
     </div>
    ▸<div class="ad-container leaderboard leaderboard--top is-
    scrolling is-scrolling--down">…</div>
    ▸<div class="container chart-container container--xxlight-
... #main div div.container.container--no-background.chart-number-one
```

```
Filter                 :hov .cls +
element.style {
}
@media only screen and (min-width:
768px)
.ch  charts.css?ca37…70913defd775:6
art
-page .container--no-background {
  ~~width: 100%;~~
}
```

```
▸undefined global: opera                         onloadwff.js:12
▸undefined global: LP                            onloadwff.js:12
▸undefined global: KeyEvent                      onloadwff.js:12
▸undefined global: module                        onloadwff.js:12
▸undefined global: LPPlatform                    onloadwff.js:12
Failed to load resource:        secure.quantserve.com/quant.js:1
net::ERR_BLOCKED_BY_CLIENT
Failed to load      imasdk.googleapis.co…sdkloader/ima3.js:1
resource: net::ERR_BLOCKED_BY_CLIENT
Failed to load      metrics.brightcove.c…51152884627&seq=0:1
resource: net::ERR_BLOCKED_BY_CLIENT
Failed to load      metrics.brightcove.c…51152884669&seq=1:1
resource: net::ERR_BLOCKED_BY_CLIENT
```

Notice that on the top left of the console, there is an arrow symbol.

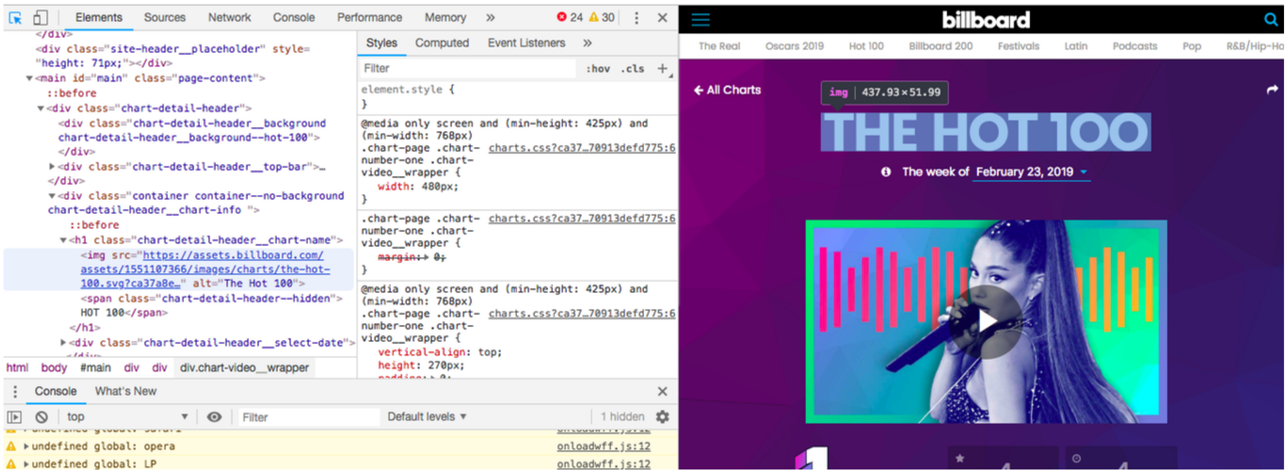


```
<div class="site-header_
  <div class="site-heade
```

Top left

If you click on this arrow and then click on an area of the site itself, the code for that particular item will be highlighted in the console:

Example



```
    </div>
    <div class="site-header__placeholder" style=
    "height: 71px;"></div>
  ▼<main id="main" class="page-content">
      ::before
    ▼<div class="chart-detail-header">
        <div class="chart-detail-header__background
        chart-detail-header__background--hot-100">
        </div>
      ▶<div class="chart-detail-header__top-bar">…
      </div>
      ▼<div class="container container--no-background
      chart-detail-header__chart-info ">
          ::before
        ▼<h1 class="chart-detail-header__chart-name">
            <img src="https://assets.billboard.com/
            assets/1551107366/images/charts/the-hot-
            100.svg?ca37a8e…" alt="The Hot 100">
            <span class="chart-detail-header--hidden">
            HOT 100</span>
          </h1>
        ▶<div class="chart-detail-header__select-date">
```

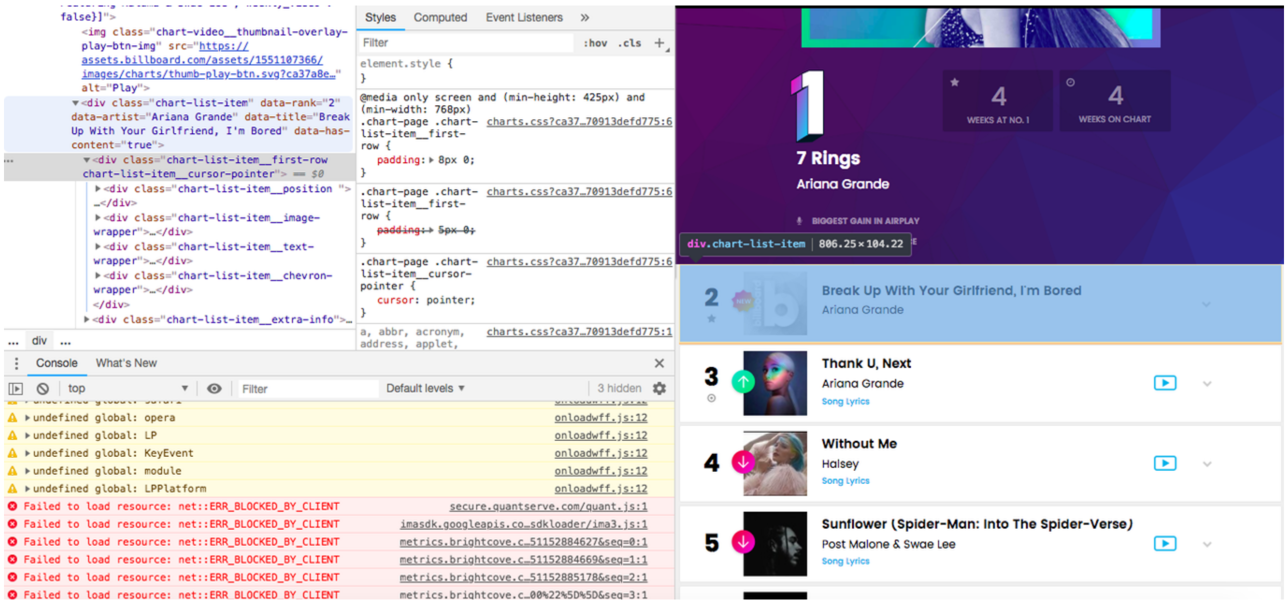
```
element.style {
}
@media only screen and (min-height: 425px) and
(min-width: 768px)
.chart-page .chart-      charts.css?ca37…70913defd775:6
number-one .chart-
video__wrapper {
   width: 480px;
}
.chart-page .chart-      charts.css?ca37…70913defd775:6
number-one .chart-
video__wrapper {
   ~~margin:▸ 0;~~
}
@media only screen and (min-height: 425px) and
(min-width: 768px)
.chart-page .chart-      charts.css?ca37…70913defd775:6
number-one .chart-
video__wrapper {
   vertical-align: top;
   height: 270px;
```

```
▶undefined global: opera      onloadwff.js:12
▶undefined global: LP         onloadwff.js:12
```

In the above image, I am hovering at "**THE HOT 100**" part (highlighted in blue) and on the left console side, you can see **<div>** tag is highlighted.

```
<div class="vjs-poster" tabindex="-1" aria-disabled="false" style="background-image: url(&quot;https://cf-images.us-east-1.prod.boltdns.net/v1/static/1125911414/81778028-1814-46fc-b195-c8643a74c4f4/4eb46bcc-5510-42d2-b6c2-847fa5b18cb3/1280x720/match/image.jpg&quot;);"></div>
```

Now we can try to find the location of song and artist names. Let's try.



```
false}]">
  <img class="chart-video__thumbnail-overlay-play-btn-img" src="https://assets.billboard.com/assets/1551107366/images/charts/thumb-play-btn.svg?ca37a8e…"
  alt="Play">
▼<div class="chart-list-item" data-rank="2" data-artist="Ariana Grande" data-title="Break Up With Your Girlfriend, I'm Bored" data-has-content="true">
  ▼<div class="chart-list-item__first-row chart-list-item__cursor-pointer"> == $0
    ▶<div class="chart-list-item__position ">…</div>
    ▶<div class="chart-list-item__image-wrapper">…</div>
    ▶<div class="chart-list-item__text-wrapper">…</div>
    ▶<div class="chart-list-item__chevron-wrapper">…</div>
   </div>
  ▶<div class="chart-list-item__extra-info">…
```

```
element.style {
}
@media only screen and (min-height: 425px) and
(min-width: 768px)
.chart-page .chart-   charts.css?ca37…70913defd775:6
list-item__first-
row {
   padding: ▶ 8px 0;
}
.chart-page .chart-   charts.css?ca37…70913defd775:6
list-item__first-
row {
   ~~padding: ▶ 5px 0;~~
}
.chart-page .chart-   charts.css?ca37…70913defd775:6
list-item__cursor-
pointer {
   cursor: pointer;
}
a, abbr, acronym,     charts.css?ca37…70913defd775:1
address, applet,
```

```
▶undefined global: opera                                          onloadwff.js:12
▶undefined global: LP                                             onloadwff.js:12
▶undefined global: KeyEvent                                       onloadwff.js:12
▶undefined global: module                                         onloadwff.js:12
▶undefined global: LPPlatform                                     onloadwff.js:12
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT          secure.quantserve.com/quant.js:1
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT   imasdk.googleapis.co…sdkloader/ima3.js:1
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT   metrics.brightcove.c…51152884627&seq=0:1
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT   metrics.brightcove.c…51152884669&seq=1:1
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT   metrics.brightcove.c…51152885178&seq=2:1
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT   metrics.brightcove.c…00%22%5D%5D&seq=3:1
```

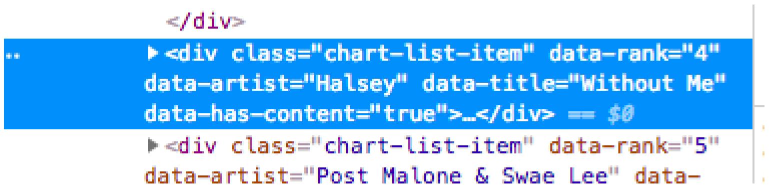
If I hover over the container of rank 2 on the website, it highlights the HTML part of it:

```
<div class="chart-list-item" data-rank="2" data-artist="Ariana Grande" data-title="Break Up With Your Girlfriend, I'm Bored" data-has-content="true">
```

NOTICE! There are attributes called **data-rank**, **data-artist**, **data-title** and the information stated in those attributes are exactly the same as what is shown on the page!

You can try this with some of the other ranked songs, and see that they all have **data-rank, data-artist, data-title** attributes per song.

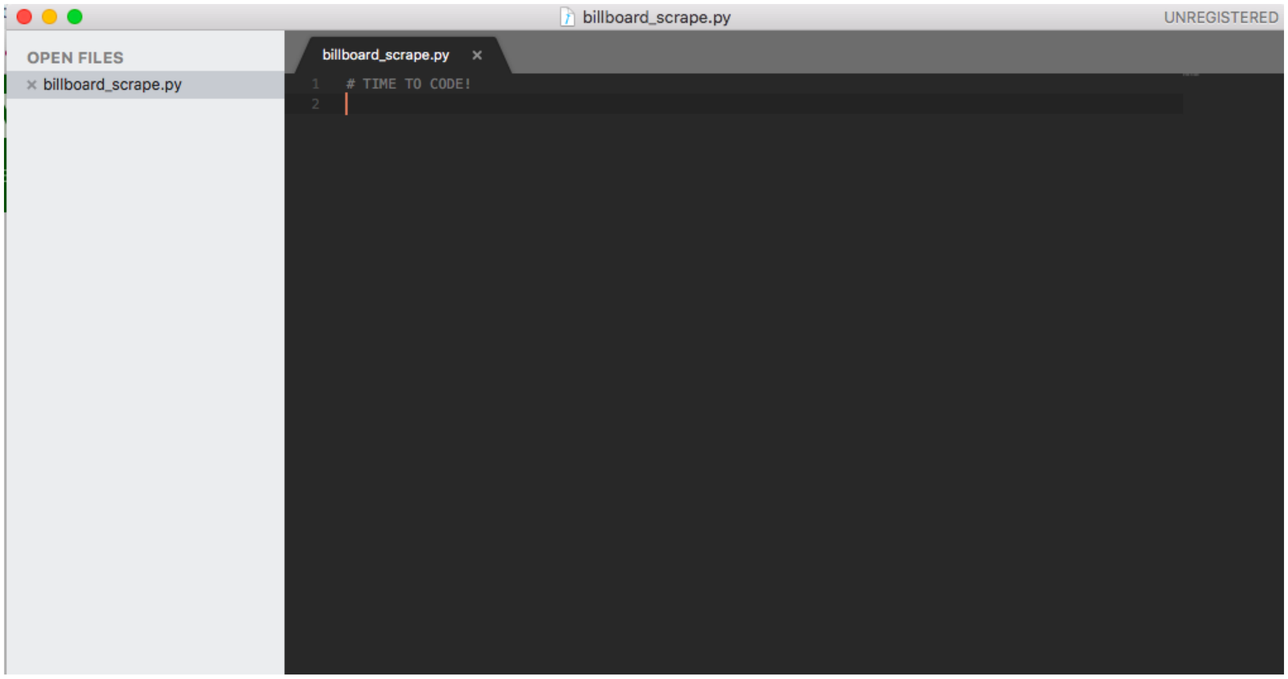
Example:



```
</div>
▶<div class="chart-list-item" data-rank="4"
data-artist="Halsey" data-title="Without Me"
data-has-content="true">…</div> == $0
▶<div class="chart-list-item" data-rank="5"
data-artist="Post Malone & Swae Lee" data-
```

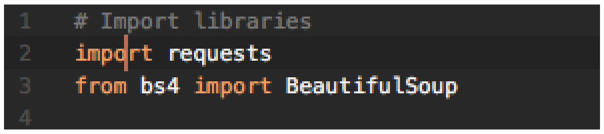
3. Time to Code! Making Web Requests

- Open your text editor and create a file. You can name your file whatever you want but don't forget to put the **.py** extension at the end. For example, I will create a file called "**billboard_scrape.py**". I am using SublimeText3 as text editor .



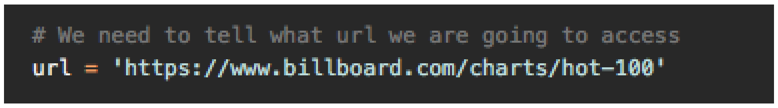
```
# TIME TO CODE!
```

First, we need to import all the libraries that we are going to use. Type the following code into your file (you don't need to put in the comment after the #):



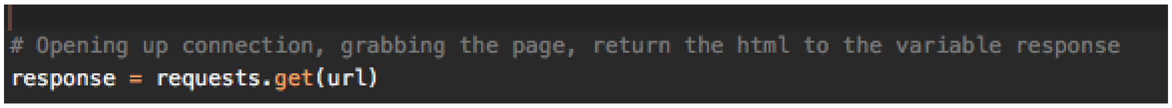
```
# Import libraries
import requests
from bs4 import BeautifulSoup
```

Next, we set the url to the website:



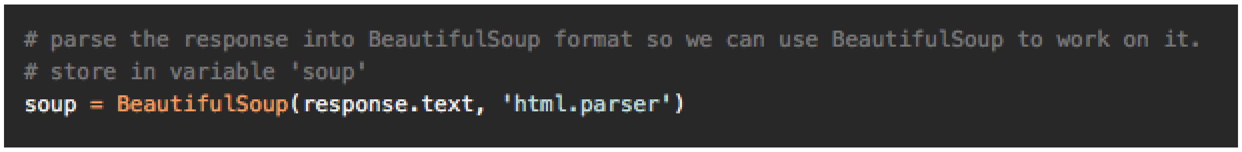
```python
# We need to tell what url we are going to access
url = 'https://www.billboard.com/charts/hot-100'
```

Then, make use of the Python request to get the HTML page of the url declared:



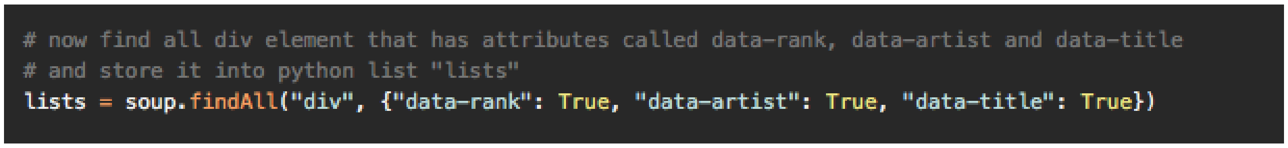
```python
# Opening up connection, grabbing the page, return the html to the variable response
response = requests.get(url)
```

Parse the html with BeautifulSoup so that we can work with a nicer, nested BeautifulSoup data structure:



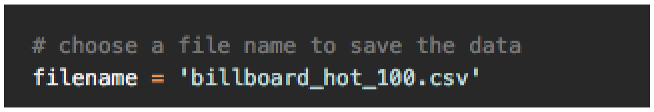
```python
# parse the response into BeautifulSoup format so we can use BeautifulSoup to work on it.
# store in variable 'soup'
soup = BeautifulSoup(response.text, 'html.parser')
```

Now the variable '**soup**' containing the HTML of the page! Let's start extracting the data.



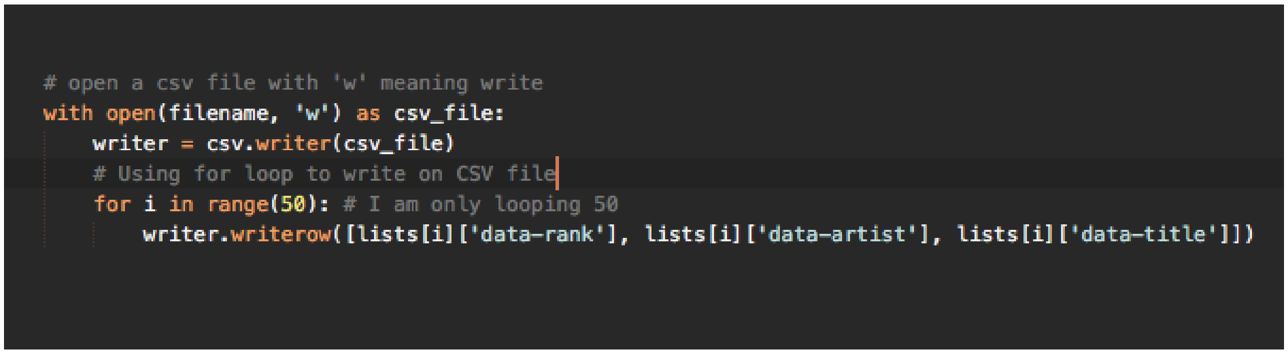
```python
# now find all div element that has attributes called data-rank, data-artist and data-title
# and store it into python list "lists"
lists = soup.findAll("div", {"data-rank": True, "data-artist": True, "data-title": True})
```

Now from the lists, we want to get the values of data-rank, data-artist and data-title. We also need to create a file to store the data. Choose a filename (be careful to not name this file with an existing file because it will overwrite):



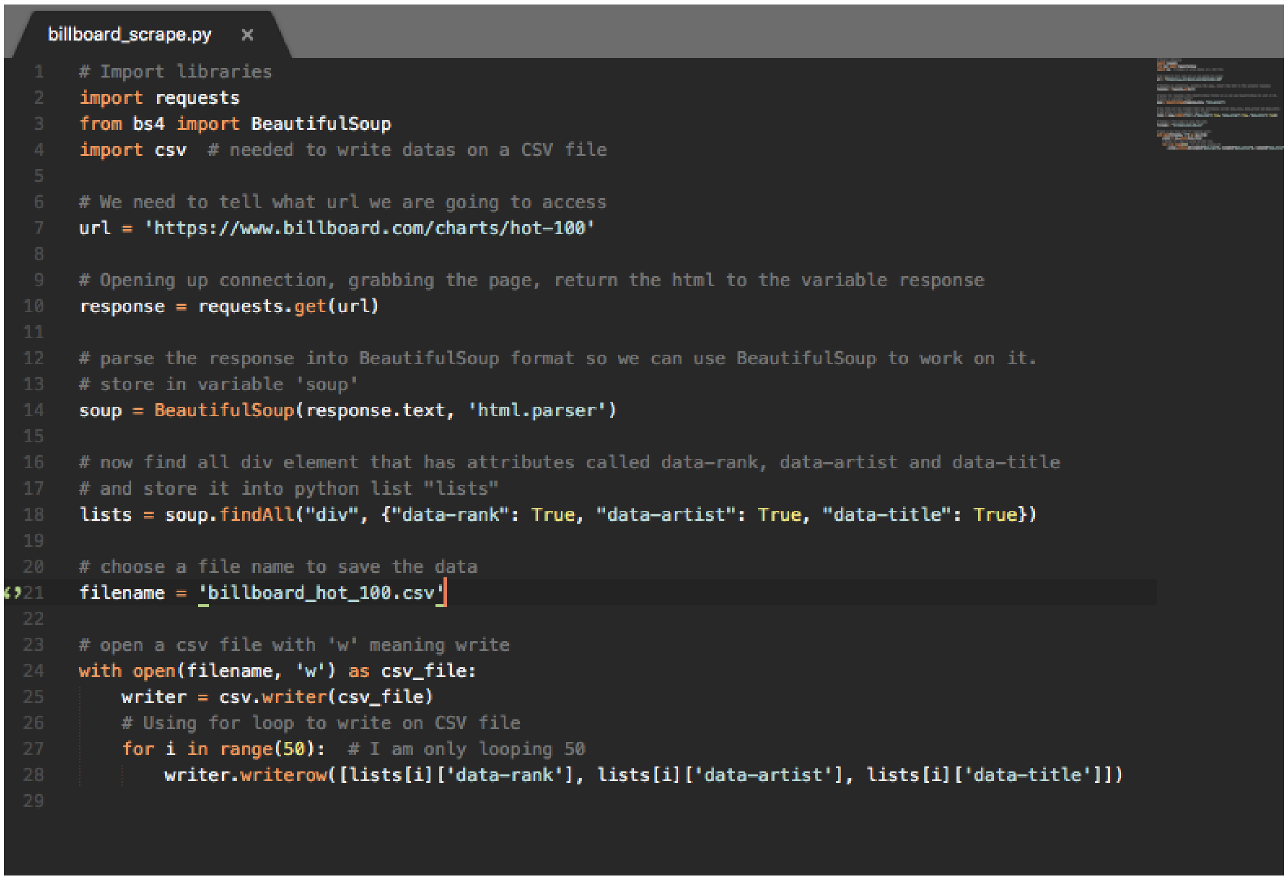
```python
# choose a file name to save the data
filename = 'billboard_hot_100.csv'
```

The following code will write the data on my **billboard_hot_100.csv** file. It is storing the rank, song name, and artist name.
NOTICE: I am looping only 50 times, thus if you want to store all the songs, substitute the **50** to **len(lists).**



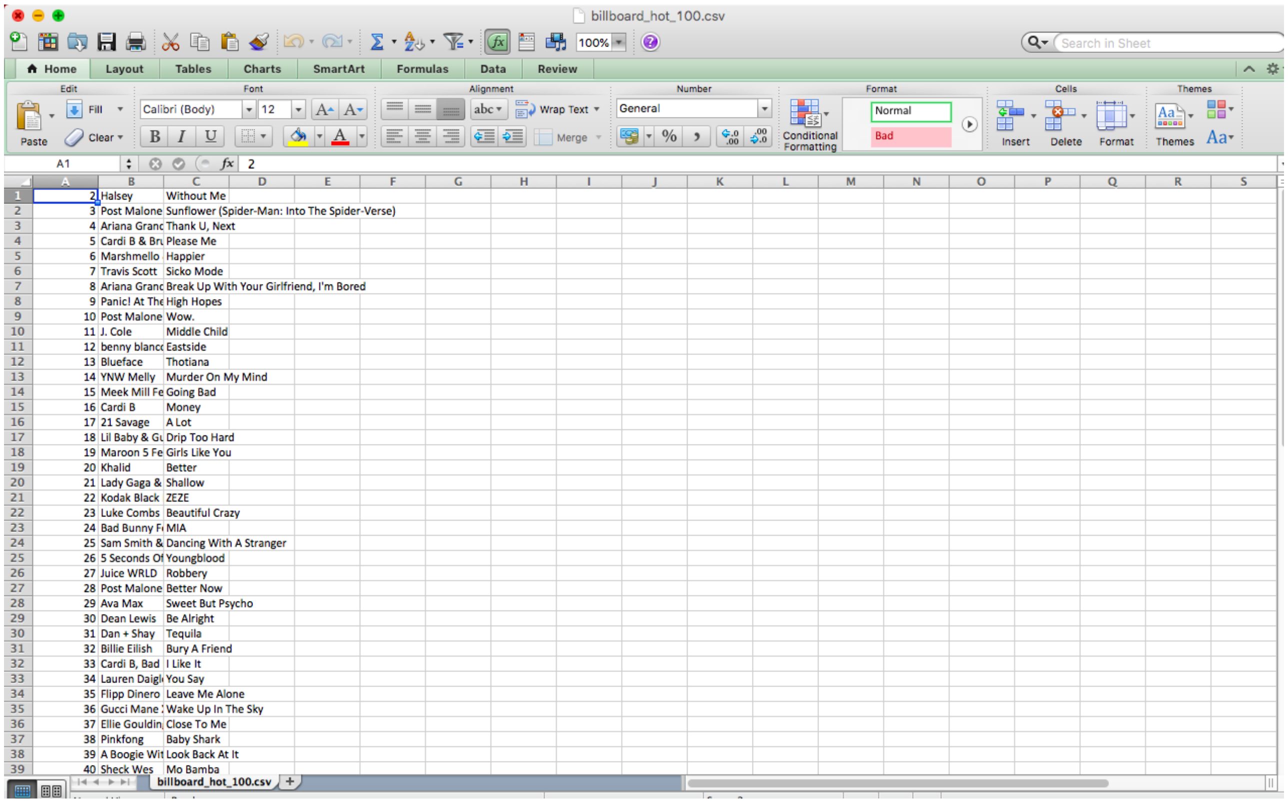
```python
# open a csv file with 'w' meaning write
with open(filename, 'w') as csv_file:
    writer = csv.writer(csv_file)
    # Using for loop to write on CSV file
    for i in range(50): # I am only looping 50
        writer.writerow([lists[i]['data-rank'], lists[i]['data-artist'], lists[i]['data-title']])
```

Here’s a summary of all the code we wrote:



```python
# Import libraries
import requests
from bs4 import BeautifulSoup
import csv  # needed to write datas on a CSV file

# We need to tell what url we are going to access
url = 'https://www.billboard.com/charts/hot-100'

# Opening up connection, grabbing the page, return the html to the variable response
response = requests.get(url)

# parse the response into BeautifulSoup format so we can use BeautifulSoup to work on it.
# store in variable 'soup'
soup = BeautifulSoup(response.text, 'html.parser')

# now find all div element that has attributes called data-rank, data-artist and data-title
# and store it into python list "lists"
lists = soup.findAll("div", {"data-rank": True, "data-artist": True, "data-title": True})

# choose a file name to save the data
filename = 'billboard_hot_100.csv'

# open a csv file with 'w' meaning write
with open(filename, 'w') as csv_file:
    writer = csv.writer(csv_file)
    # Using for loop to write on CSV file
    for i in range(50):  # I am only looping 50
        writer.writerow([lists[i]['data-rank'], lists[i]['data-artist'], lists[i]['data-title']])

```

Run the code and if there are no errors, you’ll see the csv file is created. Mine looks like this:


billboard_hot_100.csv

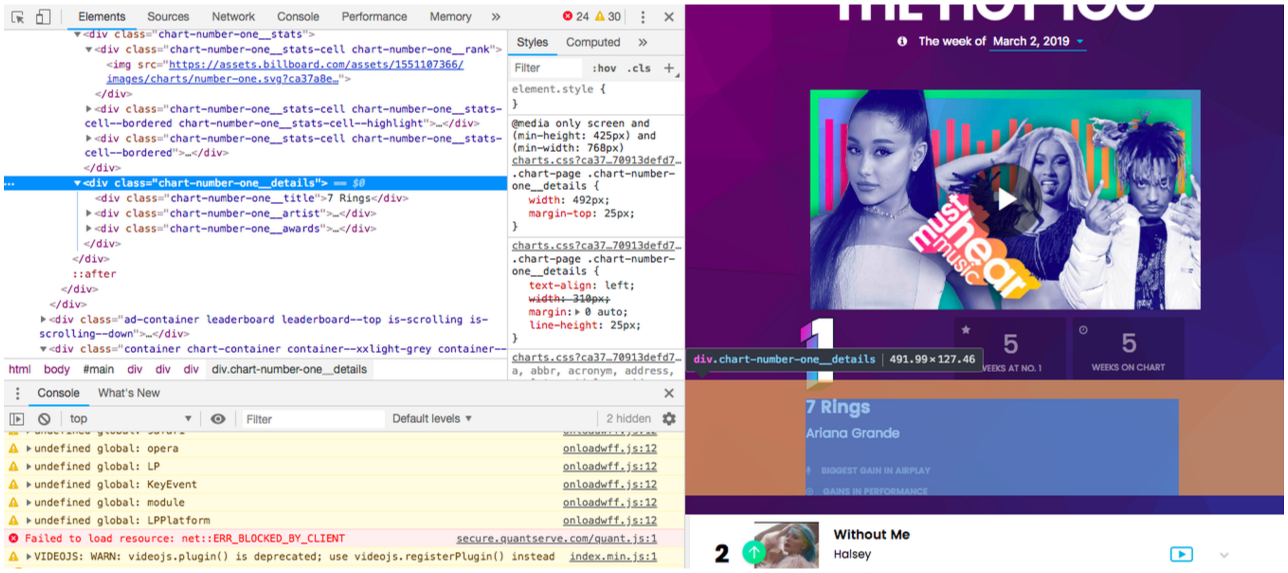
	A	B	C
1	2	Halsey	Without Me
2	3	Post Malone	Sunflower (Spider-Man: Into The Spider-Verse)
3	4	Ariana Granc	Thank U, Next
4	5	Cardi B & Bru	Please Me
5	6	Marshmello	Happier
6	7	Travis Scott	Sicko Mode
7	8	Ariana Granc	Break Up With Your Girlfriend, I'm Bored
8	9	Panic! At The	High Hopes
9	10	Post Malone	Wow.
10	11	J. Cole	Middle Child
11	12	benny blanco	Eastside
12	13	Blueface	Thotiana
13	14	YNW Melly	Murder On My Mind
14	15	Meek Mill Fe	Going Bad
15	16	Cardi B	Money
16	17	21 Savage	A Lot
17	18	Lil Baby & Gu	Drip Too Hard
18	19	Maroon 5 Fe	Girls Like You
19	20	Khalid	Better
20	21	Lady Gaga &	Shallow
21	22	Kodak Black	ZEZE
22	23	Luke Combs	Beautiful Crazy
23	24	Bad Bunny Fe	MIA
24	25	Sam Smith &	Dancing With A Stranger
25	26	5 Seconds Of	Youngblood
26	27	Juice WRLD	Robbery
27	28	Post Malone	Better Now
28	29	Ava Max	Sweet But Psycho
29	30	Dean Lewis	Be Alright
30	31	Dan + Shay	Tequila
31	32	Billie Eilish	Bury A Friend
32	33	Cardi B, Bad	I Like It
33	34	Lauren Daigl	You Say
34	35	Flipp Dinero	Leave Me Alone
35	36	Gucci Mane	Wake Up In The Sky
36	37	Ellie Gouldin	Close To Me
37	38	Pinkfong	Baby Shark
38	39	A Boogie Wit	Look Back At It
39	40	Sheck Wes	Mo Bamba

Phew! We wrote less than 30 lines of code and we got all the rank, artist and song data into a nice neat CSV file!

Extra:

You've probably noticed by now that this code actually didn't get the Rank 1 song (which on Feb 26-2019 was 7-rings by Ariana Grande). Why? Because Rank 1 has different html attributes and doesn't have data-artist attribute.

if you really want to get the Rank 1 song, then here is how you do it:



```
▼<div class="chart-number-one__stats">
  ▼<div class="chart-number-one__stats-cell chart-number-one__rank">
     <img src="https://assets.billboard.com/assets/1551107366/
     images/charts/number-one.svg?ca37a8e…">
   </div>
  ▶<div class="chart-number-one__stats-cell chart-number-one__stats-
  cell--bordered chart-number-one__stats-cell--highlight">…</div>
  ▶<div class="chart-number-one__stats-cell chart-number-one__stats-
  cell--bordered">…</div>
  </div>
▼<div class="chart-number-one__details"> == $0
   <div class="chart-number-one__title">7 Rings</div>
  ▶<div class="chart-number-one__artist">…</div>
  ▶<div class="chart-number-one__awards">…</div>
  </div>
 </div>
 ::after
</div>
</div>
▶<div class="ad-container leaderboard leaderboard--top is-scrolling is-
scrolling--down">…</div>
▼<div class="container chart-container container--xxlight-grey container--
```

html body #main div div div div.chart-number-one__details

```
element.style {
}
@media only screen and
(min-height: 425px) and
(min-width: 768px)
charts.css?ca37…70913defd7…
.chart-page .chart-number-
one__details {
   width: 492px;
   margin-top: 25px;
}
charts.css?ca37…70913defd7…
.chart-page .chart-number-
one__details {
   text-align: left;
   ~~width: 310px;~~
   margin: ▶ 0 auto;
   line-height: 25px;
}
charts.css?ca37…70913defd7…
a, abbr, acronym, address,
```

Console   What's New

top   Filter   Default levels ▾   2 hidden

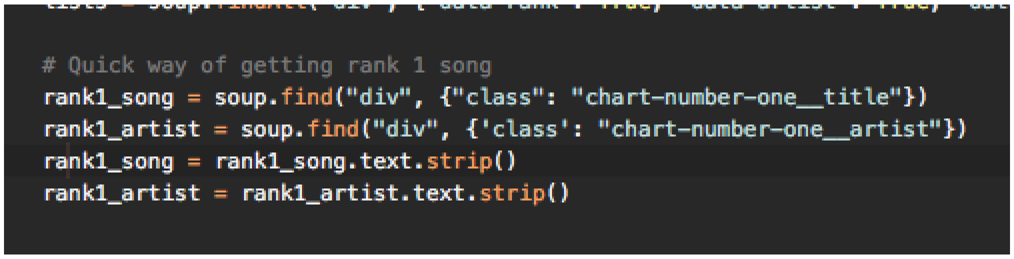
```
▶undefined global: opera                                   onloadwff.js:12
▶undefined global: LP                                      onloadwff.js:12
▶undefined global: KeyEvent                                onloadwff.js:12
▶undefined global: module                                  onloadwff.js:12
▶undefined global: LPPlatform                              onloadwff.js:12
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT   secure.quantserve.com/quant.js:1
▶VIDEOJS: WARN: videojs.plugin() is deprecated; use videojs.registerPlugin() instead  index.min.js:1
```

When I hover the rank 1 song, I notice that they use a class name called "chart-number-one_XXX". Well, isn't it obvious that the class must contains some information about the rank 1 song?

```
<div class="chart-number-one__title">7 Rings</div>
<div class="chart-number-one__artist">
  <a href="/music/ariana-grande">
  Ariana Grande
  </a>
</div>
<div class="chart-number-one__awards">
  <div class="chart-number-one__award-icon">
    <i class="fa fa-microphone"></i> Biggest gain in airplay
  </div>
  <div class="chart-number-one__award-icon">
    <i class="fa fa-dot-circle-o"></i> Gains in performance
  </div>
</div>
```

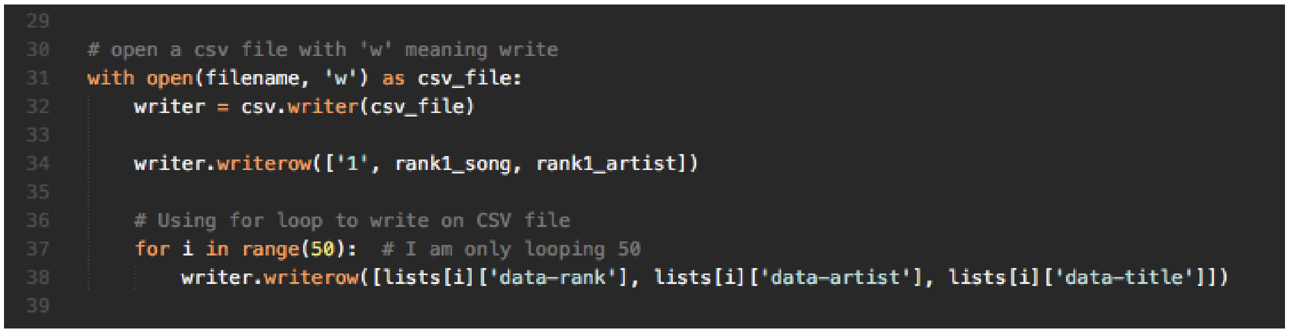
Looks like you can just extract the text of the <div> tag that have the class names of 'chart-number-one_title' and 'chart-number-one_artist' in order to get the rank 1 artist and song name.

One way to do this is by adding the following on line 26:



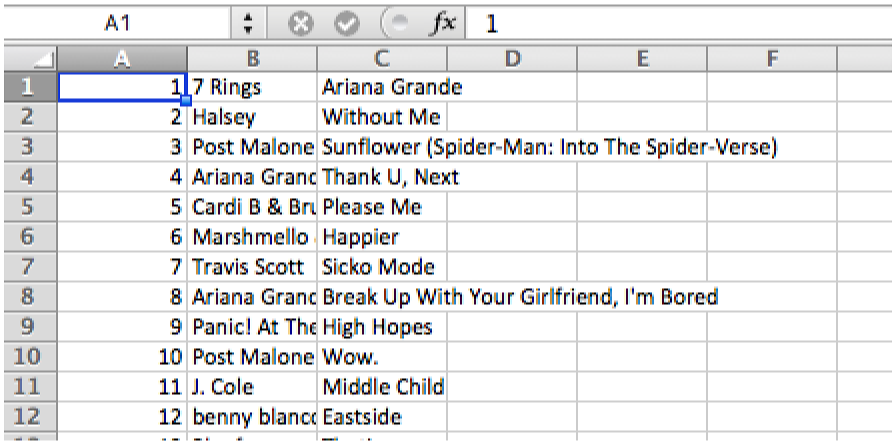
```
lists = soup.findAll('div', {'data-rank': True, 'data-artist': True, 'dat

# Quick way of getting rank 1 song
rank1_song = soup.find("div", {"class": "chart-number-one__title"})
rank1_artist = soup.find("div", {'class': "chart-number-one__artist"})
rank1_song = rank1_song.text.strip()
rank1_artist = rank1_artist.text.strip()
```

And adding a new line at line 34:



```
29
30  # open a csv file with 'w' meaning write
31  with open(filename, 'w') as csv_file:
32      writer = csv.writer(csv_file)
33
34      writer.writerow(['1', rank1_song, rank1_artist])
35
36      # Using for loop to write on CSV file
37      for i in range(50):  # I am only looping 50
38          writer.writerow([lists[i]['data-rank'], lists[i]['data-artist'], lists[i]['data-title']])
39
```

And if you re-run the code, you see rank 1 is added to the CSV file!



A1 fx 1

	A	B	C	D	E	F
1	1	7 Rings	Ariana Grande			
2	2	Halsey	Without Me			
3	3	Post Malone	Sunflower (Spider-Man: Into The Spider-Verse)			
4	4	Ariana Granc	Thank U, Next			
5	5	Cardi B & Bru	Please Me			
6	6	Marshmello	Happier			
7	7	Travis Scott	Sicko Mode			
8	8	Ariana Granc	Break Up With Your Girlfriend, I'm Bored			
9	9	Panic! At The	High Hopes			
10	10	Post Malone	Wow.			
11	11	J. Cole	Middle Child			
12	12	benny blanco	Eastside			

4. Summary

There are other ways of writing code to gathering data, but this code does the job, and now you know what web scraping is as well as the basics of how to do it in Python. Hope you enjoyed!

5. Resources

BeautifulSoup documentation:
https://www.crummy.com/software/BeautifulSoup/bs4/doc/

More on writing / reading CSV file: https://realpython.com/python-csv/#parsing-csv-files-with-pythons-built-in-csv-library

More on Web Scraping: https://hackernoon.com/web-scraping-tutorial-with-python-tips-and-tricks-db070e70e071

User test design

Target audience: Anyone who have interest in what is Web scraping and how to do it. Ideally someone who has some programming experience and fast learner since my tutorial will not teach Python in details. However, even though if user did not know anything about programming should be able to understand and follow easily because I made this instruction not difficult and less technical.

How User test will be done:
My instruction is tested on someone who has MAC and have never done Web Scraping. I will be with her/him when she/he is reading the instruction so that I can answer her/his questions. My success is measured by whether the user is able to understand what is Web Scraping and how to do it (or at least understand what the code is doing). The user doesn't necessary need to actually write code or learn Python unless they want to. If their purpose of reading this is to just understand how to do Web Scraping, this will give them enough ideas on web scraping and apply this into their projects. I don't measure the time it took for user to understand my instructions since learning process is different by individuals.

---

report of the results of the user test

How test was performed:

- I asked my host mother (a person who I live with) to read the instruction. She did not have any programming background or knowledge of Web Scraping. I was in a same room with her when she was reading it.

Improvement of the instruction after the user test

- The participant had asked me few questions during the experiment but overall she told me that she understood my instruction. It was more on my grammar that I needed to fix. Also I need to add some more definitions or find alternative words for few technical words which user may not know. My revision was mainly focused on changing the wording and format to make user easier to follow.
- I decided not to list installation part for Windows user since I don't have experience with Windows.
- Since my writing is not too formal, this instruction is not mean to be a professional guide for companies to their employees or but for anyone.
- From my peer feedback and the user test results, they were able to understand and follow my instruction without any major issue. I got positive feedbacks on my formatting. Since this is a technical instruction involving programming, I decided to format this like a blog post that people would read online. I kept each sentences short and clear thus readers won't get confused or be bored and included lot of pictures to help readers. I also linked my external resources incase if they have difficulty understanding since my tutorial does not cover everything. From the feedback I got, I've designed this instruction to a beginner friendly.
- My tested user took around 15 min to read this instruction without doing the coding part, so this instruction might take time to read depends on the readers.