



This branch is [5 commits ahead](#), [5 commits behind](#) master.

 **hoffm386** fix objectives spacing ... on Aug 1  6

[View code](#)

 README.md

Scraping Concerts - Lab

Introduction

Now that you've seen how to scrape a simple website, it's time to again practice those skills on a full-fledged site!

In this lab, you'll practice your scraping skills on an online music magazine and events website called Resident Advisor.

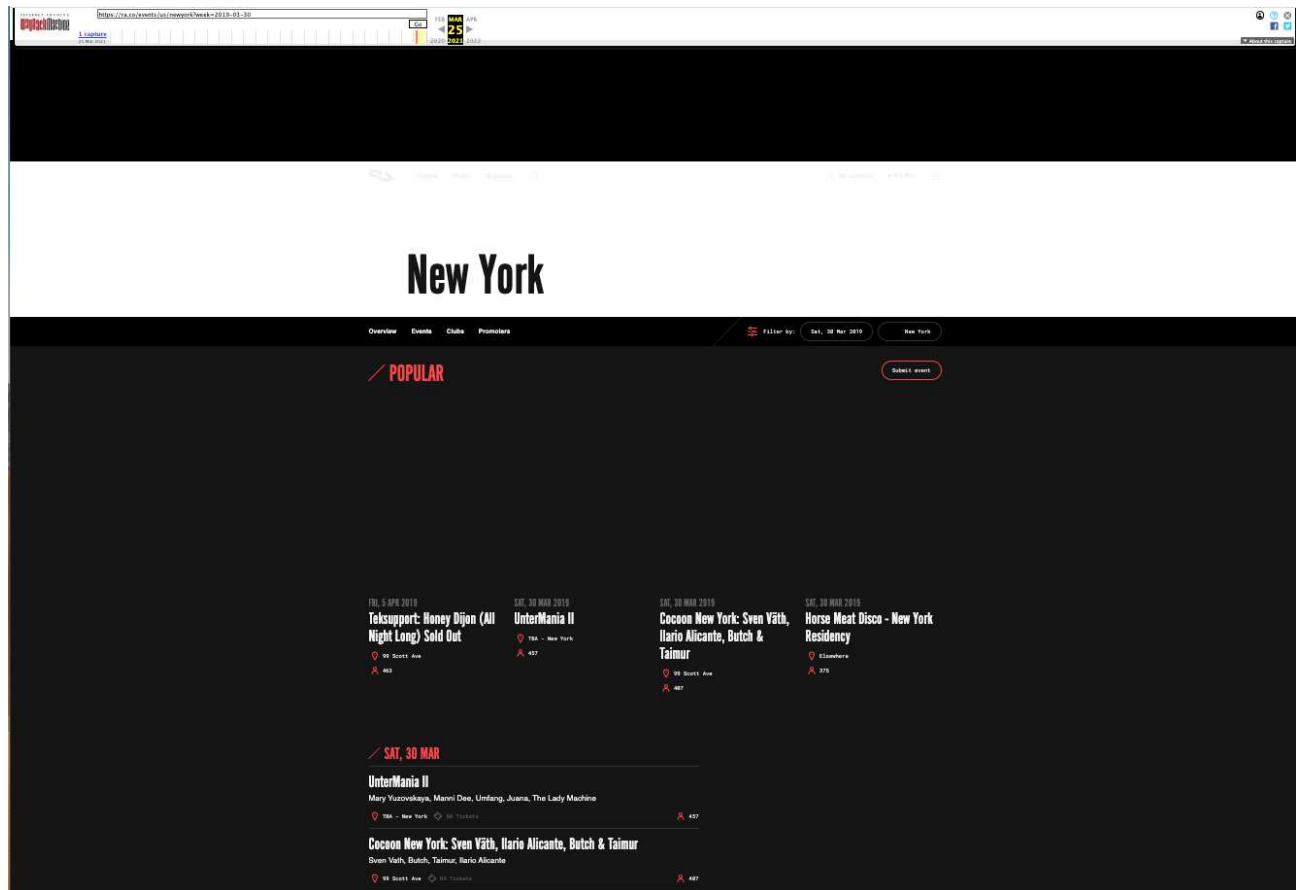
Objectives

- You will be able to:
- Create a full scraping pipeline that involves traversing over many pages of a website, dealing with errors and storing data

View the Website

For this lab, you'll be scraping the <https://ra.co> website. For reproducibility we will use the [Internet Archive](#) Wayback Machine to retrieve a version of this page from March 2019.

Start by navigating to the events page [here](#) in your browser. It should look something like this:



Open the Inspect Element Feature

Next, open the inspect element feature from your web browser in order to preview the underlying HTML associated with the page.

Write a Function to Scrape all of the Events on the Given Page

The function should return a Pandas DataFrame with columns for the `Event_Name` , `Venue` , and `Number_of_Attendees` .

Start by importing the relevant libraries, making a request to the relevant URL, and exploring the contents of the response with `BeautifulSoup`. Then fill in the `scrape_events` function with the relevant code.

```
# Relevant imports
import requests
from bs4 import BeautifulSoup
import numpy as np
import pandas as pd
import time
```

```
EVENTS_PAGE_URL = "https://web.archive.org/web/20210326225933/https://ra.co/events/us"
```

```
# Exploration: making the request and parsing the response
response = requests.get(EVENTS_PAGE_URL)
soup = BeautifulSoup(response.content, "html.parser")
```

```
# Find the container with event listings in it

# This page is organized somewhat unusually, and many of
# the CSS attributes seem auto-generated. We notice that
# there is a div with "events-all" in its attributes that
# looks promising
events_all_div = soup.find('div', attrs={"data-tracking-id": "events-all"})

# The actual content is nested in a ul containing a single
# li within that div. Unclear why they are using a "list"
# concept for one element, but let's go ahead and select it
event_listings = events_all_div.find("ul").find("li")

# Print out some chunks of the text inside to make sure we
# have everything we need in here

# Beginning has events for March 30th
print(event_listings.text[:200])
print()

# Later we have events for March 31st
march_31st_start = event_listings.text.find("Sun, 31 Mar")
print(event_listings.text[march_31st_start:march_31st_start + 200])

# It looks like everything we need will be inside this event_listings tag
```

Sat, 30 MarUnterMania IIMary Yuzovskaya, Manni Dee, Umfang, Juana, The Lady
 MachineTBA - New YorkRARA Tickets457Cocoon New York: Sven V  th, Ilario Alicante,
 Butch & TaimurSven Vath, Butch, Taimur, Il

Sun, 31 MarSunday: Soul SummitNowadaysRARA Tickets132New Dad & Aaron Clark
 (Honcho)Aaron Clark, New DadAce Hotel3ParadiscoOccupy The DiscoLe Bain3Sunday
 Soiree: Unknown Showcase (Detroit)Ryan Dahl, Ha

```

# Find a list of events by date within that container

# Now we look at what is inside of that event_listings li tag.
# Based on looking at the HTML with developer tools, we see
# that there are 13 children of that tag, all divs. Each div
# is either a container of events on a given date, or empty

# Let's create a collection of those divs. recursive=False
# means we stop at 1 level below the event_listings li
dates = event_listings.findChildren(recursive=False)

# Now let's print out the start of the March 30th and March
# 31st sections again. This time each is in its own "date"
# container

# March 30th is at the 0 index
print("0 index:", dates[0].text[:200])
print()
# The 1 index is empty. We'll need to skip this later
print("1 index: ", dates[1].text)
print()
# March 31st is at the 2 index
print("2 index:", dates[2].text[:200])

# Now we know we can loop over all of the items in the dates
# list of divs to find the dates, although some will be blank
# so we'll need to skip them

```

0 index: Sat, 30 MarUnterMania IIMary Yuzovskaya, Manni Dee, Umfang, Juana, The
 Lady MachineTBA - New YorkRARA Tickets457Cocoon New York: Sven V  th, Ilario
 Alicante, Butch & TaimurSven Vath, Butch, Taimur, Il

1 index:

2 index: Sun, 31 MarSunday: Soul SummitNowadaysRARA Tickets132New Dad & Aaron
 Clark (Honcho)Aaron Clark, New DadAce Hotel3ParadiscoOccupy The DiscoLe
 Bain3Sunday Soiree: Unknown Showcase (Detroit)Ryan Dahl, H

```
# Extract the date (e.g. Sat, 30 Mar) from one of those containers

# Grabbing just one to practice on
first_date = dates[0]

# This div contains a div with the date, followed by several uls
# containing actual event information

# The div with the date happens to have another human-readable
# CSS class, so let's use that to select it then grab its text
date = first_date.find("div", class_="sticky-header").text

# There is a / thing used for aesthetic reasons; let's remove it
date = date.strip("/")
date

'Sat, 30 Mar'

# Extract the name, venue, and number of attendees from one of the
# events within that container

# As noted previously, the div with information about events on
# this date contains several ul tags, each with information about
# a specific event. Get a list of them.
# (Again this is an odd use of HTML, to have an unordered list
# containing a single list item. But we scrape what we find!)
first_date_events = first_date.findChildren("ul")

# Grabbing the first event ul to practice on
first_event = first_date_events[0]

# Each event ul contains a single h3 with the event name, easy enough
name = first_event.find("h3").text

# Venue and attendees is more complicated. Across the bottom are 1-3
# divs with height 30. The 0th contains a location pin SVG and then
# the location text. The -1th (last), when present, contains a person
# icon SVG and then the number of attendees. Sometimes there is a
# middle div with a ticket icon SVG and the words "RA Tickets", which
# we will plan to ignore

# First, get all 1-3 divs that match this description
venue_and_attendees = first_event.findAll("div", attrs={"height": 30})
# The venue is the 0th (left-most) div, get its text
venue = venue_and_attendees[0].text
# The number of attendees is the last div (although it's sometimes
```

```
# missing), get its text
num_attendees = int(venue_and_attendees[-1].text)

# Print out everything for one event
print("Name:", name)
print("Venue:", venue)
print("Date:", date)
print("Number of attendees:", num_attendees)
```

```
Name: UnterMania II
Venue: TBA - New York
Date: Sat, 30 Mar
Number of attendees: 457
```

```
# Testing that code out on an event with a missing attendee count
last_event = first_date_events[-1]
```

```
name = last_event.find("h3").text
```

```
venue_and_attendees = last_event.findAll("div", attrs={"height": 30})
venue = venue_and_attendees[0].text
num_attendees = int(venue_and_attendees[-1].text)
```

```
-----
ValueError                                Traceback (most recent call last)
```

```
<ipython-input-7-aa83fdadf99e> in <module>
      8 venue_and_attendees = last_event.findAll("div", attrs={"height": 30})
      9 venue = venue_and_attendees[0].text
--> 10 num_attendees = int(venue_and_attendees[-1].text)
```

```
ValueError: invalid literal for int() with base 10: 'H0L0'
```

```
# Ok, that crashes because there is no attendee count. Let's
# put a try/except and set the attendee count to NaN, since
# that represents "missing data" reasonably
```

```
try:
    num_attendees = int(venue_and_attendees[-1].text)
except ValueError:
    num_attendees = np.nan
```

```
print("Name:", name)
print("Venue:", venue)
print("Date:", date)
print("Number of attendees:", num_attendees)

# Now we have code that should work for events with and
# without attendee counts

Name: Petra, Matthusen & Lang, White & Pitsiokos, and Zorn
Venue: HØLØ
Date: Sat, 30 Mar
Number of attendees: nan

# Loop over all of the event entries, extract this information
# from each, and assemble a dataframe

# Create an empty list to hold results
rows = []

# Loop over all date containers on the page
for date_container in dates:

    # First check if this is one of the empty divs. If it is,
    # skip ahead to the next one
    if not date_container.text:
        continue

    # Same logic as above to extract the date
    date = date_container.find("div", class_="sticky-header").text
    date = date.strip("'")

    # This time, loop over all of the events
    events = date_container.findChildren("ul")
    for event in events:

        # Same logic as above to extract the name, venue, attendees
        name = event.find("h3").text
        venue_and_attendees = event.findAll("div", attrs={"height": 30})
        venue = venue_and_attendees[0].text
        try:
            num_attendees = int(venue_and_attendees[-1].text)
        except ValueError:
            num_attendees = np.nan

        # New piece here: appending the new information to rows list
        rows.append([name, venue, date, num_attendees])
```

```
# Make the list of lists into a dataframe and display
df = pd.DataFrame(rows)
df
```

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {
    vertical-align: top;
}
```

```
.dataframe thead th {
    text-align: right;
}
```

```
</style>
```

	0	1	2	3
0	UnterMania II	TBA - New York	Sat, 30 Mar	457.0
1	Cocoon New York: Sven V��th, Ilario Alicante, B...	99 Scott Ave	Sat, 30 Mar	407.0
2	Horse Meat Disco - New York Residency	Elsewhere	Sat, 30 Mar	375.0
3	Rave: Underground Resistance All Night	Nowadays	Sat, 30 Mar	232.0
4	Believe You Me // Beta Librae, Stephan Kimbel,...	TBA - New York	Sat, 30 Mar	89.0
...
114	A Night at the Baths	C'mon Everybody	Fri, 5 Apr	1.0
115	Blaqk Audio	Music Hall of Williamsburg	Fri, 5 Apr	1.0
116	Erik the Lover	Erv's	Fri, 5 Apr	1.0
117	Wax On Visions	Starliner	Fri, 5 Apr	1.0

	0	1	2	3
118	Schimanski & Good Looks present: Mercer	Schimanski	Fri, 5 Apr	1.0

119 rows × 4 columns

```
# Bring it all together in a function that makes the request, gets the
# list of entries from the response, loops over that list to extract the
# name, venue, date, and number of attendees for each event, and returns
# that list of events as a dataframe

def scrape_events(events_page_url):
    # Make the request and parse the response as HTML
    response = requests.get(events_page_url)
    soup = BeautifulSoup(response.content, "html.parser")

    # Find the container with the relevant content
    events_all_div = soup.find('div', attrs={"data-tracking-id": "events-all"})
    event_listings = events_all_div.find("ul").find("li")
    dates = event_listings.findChildren(recursive=False)

    # Loop over all dates, an all events on each date, and
    # add them to the list
    rows = []
    for date_container in dates:

        if not date_container.text:
            continue

        date = date_container.find("div", class_="sticky-header").text
        date = date.strip("'")

        events = date_container.findChildren("ul")
        for event in events:

            name = event.find("h3").text
            venue_and_attendees = event.findAll("div", attrs={"height": 30})
            venue = venue_and_attendees[0].text
            try:
                num_attendees = int(venue_and_attendees[-1].text)
            except ValueError:
                num_attendees = np.nan

            rows.append([name, venue, date, num_attendees])

    df = pd.DataFrame(rows)
    # This time also specify the column names
```

```
df.columns = ["Event_Name", "Venue", "Event_Date", "Number_of_Attendees"]
return df
```

```
scrape_events(EVENTS_PAGE_URL)
```

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {
    vertical-align: top;
}
```

```
.dataframe thead th {
    text-align: right;
}
```

```
</style>
```

	Event_Name	Venue	Event_Date	Number_of_Attendees
0	UnterMania II	TBA - New York	Sat, 30 Mar	457.0
1	Cocoon New York: Sven V��th, Ilario Alicante, B...	99 Scott Ave	Sat, 30 Mar	407.0
2	Horse Meat Disco - New York Residency	Elsewhere	Sat, 30 Mar	375.0
3	Rave: Underground Resistance All Night	Nowadays	Sat, 30 Mar	232.0
4	Believe You Me // Beta Librae, Stephan Kimbel,...	TBA - New York	Sat, 30 Mar	89.0
...
114	A Night at the Baths	C'mon Everybody	Fri, 5 Apr	1.0
115	Blaqk Audio	Music Hall of Williamsburg	Fri, 5 Apr	1.0
116	Erik the Lover	Erv's	Fri, 5 Apr	1.0

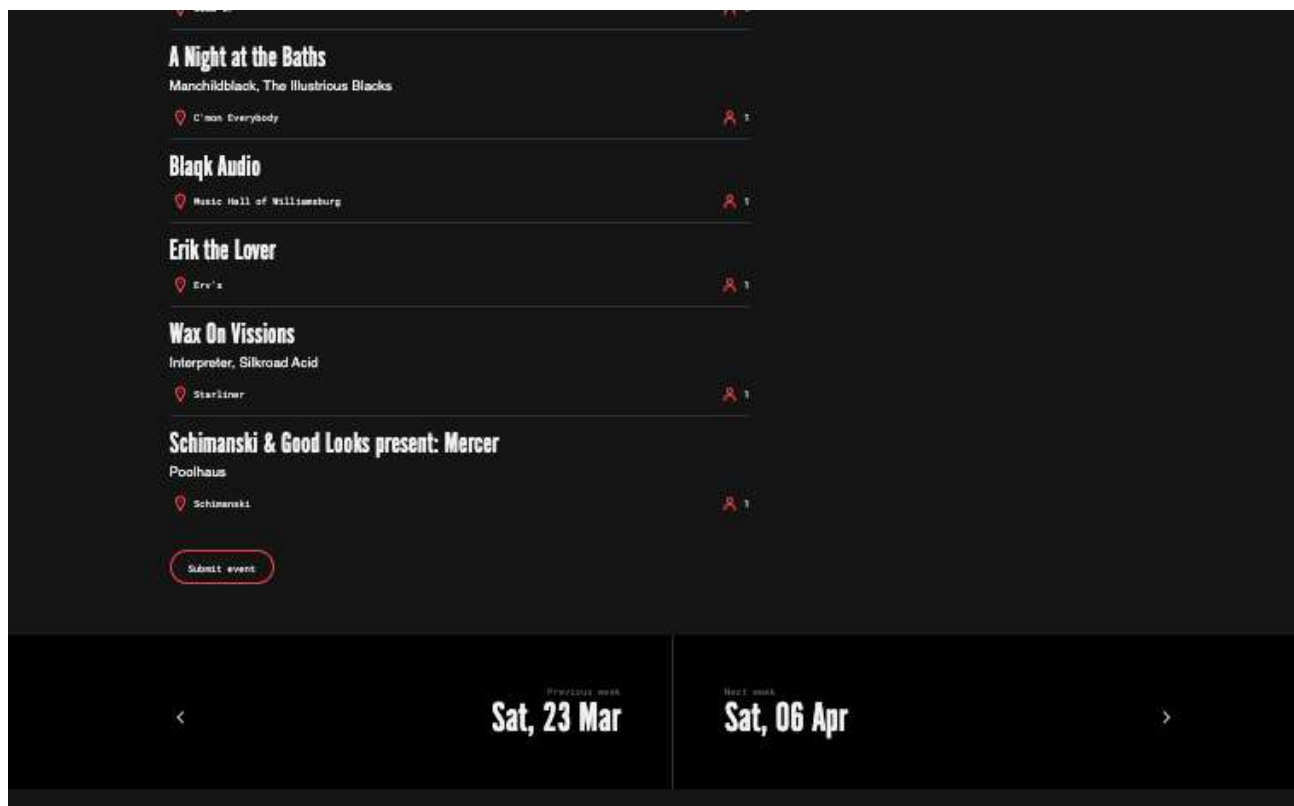
	Event_Name	Venue	Event_Date	Number_of_Attendees
117	Wax On Visions	Starliner	Fri, 5 Apr	1.0
118	Schimanski & Good Looks present: Mercer	Schimanski	Fri, 5 Apr	1.0

119 rows × 4 columns

Write a Function to Retrieve the URL for the Next Page

As you scroll down, there should be a button labeled "Next Week" that will take you to the next page of events. Write code to find that button and extract the URL from it.

This is a relative path, so make sure you add `https://web.archive.org` to the front to get the URL.



```
# Find the link, find the relative path, create the URL for the current `soup`
```

```
# This is tricky again, since there are not a lot of
# human-readable CSS classes
```

```
# One unique thing we notice is a > icon on the part where
# you click to go to the next page. It's an SVG with an
```

```
# aria-label of "Right arrow"
svg = soup.find("svg", attrs={"aria-label": "Right arrow"})

# That SVG is inside of a div
svg_parent = svg.parent

# And the tag right before that div (its "previous sibling")
# is an anchor (link) tag with the path we need
link = svg.parent.previousSibling

# Then we can extract the path from that link to build the full URL
relative_path = link.get("href")
next_page_url = "https://web.archive.org" + relative_path
next_page_url

'https://web.archive.org/web/20210326225933/https://ra.co/events/us/newyork?
week=2019-04-06'

# Fill in this function, to take in the current page's URL and return the
# next page's URL
def next_page(url):
    # Get the content
    response = requests.get(url)
    soup = BeautifulSoup(response.content, "html.parser")

    # Extract the relative path to build the full URL
    svg = soup.find("svg", attrs={"aria-label": "Right arrow"})
    svg_parent = svg.parent
    link = svg.parent.previousSibling
    relative_path = link.get("href")
    next_page_url = "https://web.archive.org" + relative_path
    return next_page_url

next_page(EVENTS_PAGE_URL)

'https://web.archive.org/web/20210326225933/https://ra.co/events/us/newyork?
week=2019-04-06'
```

Scrape the Next 500 Events

In other words, repeatedly call `scrape_events` and `next_page` until you have assembled a dataframe with at least 500 rows.

Display the data sorted by the number of attendees, greatest to least.

We recommend adding a brief `time.sleep` call between `requests.get` calls to avoid rate limiting.

```
# Make a dataframe to store results. We will concatenate
# additional dfs as they are returned
overall_df = pd.DataFrame()

current_url = EVENTS_PAGE_URL
while overall_df.shape[0] <= 500:
    # Get all events from the current URL
    df = scrape_events(current_url)
    time.sleep(.2)
    # Add the data to the overall df
    overall_df = pd.concat([overall_df, df])
    # Get the next URL and set it as the current URL
    current_url = next_page(current_url)
    time.sleep(.2)
```

```
overall_df
```

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {
    vertical-align: top;
}
```

```
.dataframe thead th {
    text-align: right;
}
```

```
</style>
```

	Event_Name	Venue	Event_Date	Number_of_Attendees
0	UnterMania II	TBA - New York	Sat, 30 Mar	457.0
1	Cocoon New York: Sven V��th, Ilario	99 Scott Ave	Sat, 30 Mar	407.0

	Event_Name	Venue	Event_Date	Number_of_Attendees
	Alicante, B...			
2	Horse Meat Disco - New York Residency	Elsewhere	Sat, 30 Mar	375.0
3	Rave: Underground Resistance All Night	Nowadays	Sat, 30 Mar	232.0
4	Believe You Me // Beta Librae, Stephan Kimbel,...	TBA - New York	Sat, 30 Mar	89.0
...
119	Sleepy & Boo, Unseen., Dysco, Joeoh	Rose Gold	Fri, 3 May	2.0
120	Diving for Disco with Jake From Extra Water	Our Wicked Lady	Fri, 3 May	2.0
121	The Happy Hour After Work Party at Doha Nightclub	Doha Club	Fri, 3 May	1.0
122	Best of the Boogie	Erv's	Fri, 3 May	1.0
123	[CANCELLED] Sound New York Pres Schuld, Sven M...	30 Wall St	Fri, 3 May	145.0

606 rows × 4 columns

```
# Display in the specified sorted order
overall_df.sort_values("Number_of_Attendees", ascending=False)
```

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {
    vertical-align: top;
}
```

```
.dataframe thead th {
    text-align: right;
}
```

</style>

	Event_Name	Venue	Event_Date	Number_of_Attendees
0	Zero presents... The Masquerade	The 1896	Sat, 6 Apr	919.0
65	Secret Solstice Pre-Party (Free Entry): Metro ...	Kings Hall - Avant Gardner	Thu, 18 Apr	670.0
0	Nina Kraviz / James Murphy / Justin Cudmore	Knockdown Center	Sat, 20 Apr	501.0
89	Stavroz live! presented by Zero	The Williamsburg Hotel	Fri, 12 Apr	481.0
91	Teksupport: Honey Dijon (All Night Long) Sold Out	99 Scott Ave	Fri, 5 Apr	463.0
...
56	420: A Musical Experience	The Kraine Theater	Mon, 22 Apr	NaN
61	420: A Musical Experience	The Kraine Theater	Tue, 23 Apr	NaN
75	420: A Musical Experience	The Kraine Theater	Wed, 24 Apr	NaN
34	Klandestino Brunch with Electronic Music	Avena Downtown	Sat, 27 Apr	NaN
48	Digital Clippings	Magick City	Sun, 28 Apr	NaN

606 rows × 4 columns

Summary

Congratulations! In this lab, you successfully developed a pipeline to scrape a website for concert event information!

Releases

No releases published

Packages

No packages published

Contributors 3



mathymitchell



hoffm386 Erin R Hoffman



mas16 matt

Languages

Jupyter Notebook 100.0%