Web scraping and HTML parsing with Beautiful Soup

Let’s suppose you want to get some information from a website? what will you do? The first thing that may come in your mind is to copy and paste the information into your local media. But what if you want a large amount of data on a daily basis and as quickly as possible. In such situations, copy and paste will not work and that’s where you’ll need web scraping.

In this article, we will discuss how to perform web scraping using the requests library and beautifulsoup library in Python.

Requests Module

Requests library is used for making HTTP requests to a specific URL and returns the response. Python requests provide inbuilt functionalities for managing both the request and response.

Syntax

Reading from URL:

```
requests.get(url).text
```

Reading from Local HTML file:

```
open('filename.html').read()
```

BeautifulSoup Library (bs4)

BeautifulSoup is used extract information from the HTML and XML files. It provides a parse tree and the functions to navigate, search or modify this parse tree.

Beautiful Soup is a Python library used to pull the data out of HTML and XML files for web scraping purposes. It produces a parse tree from page source code that can be utilized to drag data hierarchically and more legibly.

BeautifulSoup()

bs4.BeautifulSoup(source, 'html.parser')

In [63]:
```python
import bs4
import requests
import pandas as pd
url='https://www.politifact.com/factchecks'
source = requests.get(url).text
soup = bs4.BeautifulSoup(source,'html.parser')
print(soup)
```

```
<!DOCTYPE html>

<html dir="ltr" lang="en-US">
<head>
<meta charset="utf-8"/>
<meta content="ie=edge" http-equiv="x-ua-compatible"/>
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<title>Fact-checks | PolitiFact </title>
<meta content="PolitiFact is a fact-checking website that rates the accurac
y of claims by elected officials and others on its Truth-O-Meter." name="de
scription">
<meta content="PolitiFact" name="twitter:username">
<meta content="summary" name="twitter:card">
<meta content="PolitiFact" name="twitter:site"/>
<meta content="" name="twitter:url">
<meta content="" name="twitter:title"/>
<meta content="PolitiFact is a fact-checking website that rates the accurac
y of claims by elected officials and others on its Truth-O-Meter." name="tw
```

Functions used with BeautifulSoup

prettify() formatting HTML

Another great utility is the HTML visual formatter which prettifies HTML output.

Frequently, when web-scraping we want to either store or display HTML content somewhere for ingesting it with other tools or debugging.

The .prettify() method restructures HTML output to be more readable by humans: Example:

soup = bs4.BeautifulSoup(source,'html.parser')

data=soup.prettify()

In [61]:
```python
import bs4
import requests
import pandas as pd
url='https://www.politifact.com/factchecks'
source = requests.get(url).text
soup = bs4.BeautifulSoup(source,'html.parser')
data=soup.prettify()
print(data)
```

```
<!DOCTYPE html>
<html dir="ltr" lang="en-US">
 <head>
  <meta charset="utf-8"/>
  <meta content="ie=edge" http-equiv="x-ua-compatible"/>
  <meta content="width=device-width, initial-scale=1" name="viewport"/>
  <title>
   Fact-checks | PolitiFact
  </title>
  <meta content="PolitiFact is a fact-checking website that rates the accur
acy of claims by elected officials and others on its Truth-O-Meter." name
="description">
   <meta content="PolitiFact" name="twitter:username">
    <meta content="summary" name="twitter:card">
     <meta content="PolitiFact" name="twitter:site"/>
     <meta content="" name="twitter:url">
      <meta content="" name="twitter:title"/>
      <meta content="PolitiFact is a fact-checking website that rates the a
ccuracy of claims by elected officials and others on its Truth-O-Meter." na
```

find()

With the find() function, we are able to search for anything in our web page.

In [64]:
```python
soup.find('a',class_='m-statement__name')
```

Out[64]:
```
<a class="m-statement__name" href="/personalities/instagram-posts/" title="In
stagram posts">
Instagram posts
</a>
```

get_text()

As you can see in the previous function we used get_text() to extract the text part of the newly found elements title.

In [65]:
```python
soup.find('a',class_='m-statement__name').get_text()
```

Out[65]:
```
'\nInstagram posts\n'
```

strip()

The strip() method returns a copy of the string with both leading and trailing characters removed (based on the string argument passed).

We use this function in order to remove the empty spaces we have in our title: This function can also be used in any other python usage, not just Beautiful Soup, but in my personal experience, it has come in handy so many times when operating on text elements and that is why I am putting it on this list.

In [66]:
```
soup.find('a',class_='m-statement__name').get_text().strip()
```

Out[66]:
```
'Instagram posts'
```

We will scrape the politifact.com/factchecks.You need to scrap following details from all atricles.

Statement of news, Date of news, Source of news

In [56]:
```
import bs4
import requests
import pandas as pd
url='https://www.politifact.com/factchecks'
source = requests.get(url).text
soup = bs4.BeautifulSoup(source,'html.parser')
data=soup.prettify()
print(data)
```

```
<!DOCTYPE html>
<html dir="ltr" lang="en-US">
 <head>
  <meta charset="utf-8"/>
  <meta content="ie=edge" http-equiv="x-ua-compatible"/>
  <meta content="width=device-width, initial-scale=1" name="viewport"/>
  <title>
   Fact-checks | PolitiFact
  </title>
  <meta content="PolitiFact is a fact-checking website that rates the accur
acy of claims by elected officials and others on its Truth-O-Meter." name
="description">
   <meta content="PolitiFact" name="twitter:username">
    <meta content="summary" name="twitter:card">
     <meta content="PolitiFact" name="twitter:site"/>
     <meta content="" name="twitter:url">
      <meta content="" name="twitter:title"/>
      <meta content="PolitiFact is a fact-checking website that rates the a
ccuracy of claims by elected officials and others on its Truth-O-Meter." na
```

find_all()

In [57]:
```python
x=soup.find_all('li',{'class':"o-listicle__item"})
```

In [53]:
```python
print(x)
```

```
[<li class="o-listicle__item">
<article class="m-statement m-statement--is-medium m-statement--false">
<div class="m-statement__author">
<div class="m-statement__avatar">
<div class="m-statement__image">
<div class="c-image" style="padding-top: 119.27710843373494%;">
<img class="c-image__thumb" height="99" src="https://static.politifact.com/
CACHE/images/politifact/mugs/Screenshot_2023-06-06_at_5.37.31_PM/e604b46bf2
d5b0c84e17acea11173b52.jpg" width="83"/>
<picture>
<img class="c-image__original" height="178" src="https://static.politifact.
com/CACHE/images/politifact/mugs/Screenshot_2023-06-06_at_5.37.31_PM/598e0f
1c330a8fbedf4d925e15b18297.jpg" width="166"/>
</picture>
</div>
</div>
</div>
<div class="m-statement__meta">
<a class="m-statement__name" href="/personalities/benny-johnson/" title="Be
```

In [54]:
```python
print(len(x))
```

```
30
```

In [86]:
```python
x[0].find('a',class_='m-statement__name').get_text().strip()
```

Out[86]:
```
'Benny Johnson'
```

In [56]:
```python
x[0].find('div',class_='m-statement__desc').get_text()[11:23]
```

Out[56]:
```
'June 3, 2023'
```

In [58]:
```python
x[0].find('div',class_="m-statement__quote").get_text().strip()
```

Out[58]:
```
'“New tapes” of Nancy Pelosi show that the Jan. 6, 2021, attack on the U.S. C
apitol “was all planned.”'
```

In [67]:
```python
combined_data=[]
for i in x:
    a=i.find('a',class_='m-statement__name').get_text().strip()
    b=i.find('div',class_='m-statement__desc').get_text()[11:23]
    c=i.find('div',class_="m-statement__quote").get_text().strip()
    combined_data.append((a,b,c))
```

In [60]:
```
print(combined_data)
```

```
[('Benny Johnson', 'June 3, 2023', '“New tapes” of Nancy Pelosi show that the
Jan. 6, 2021, attack on the U.S. Capitol “was all planned.”'), ('Facebook pos
ts', 'May 28, 2023', 'Photos show Target selling children’s clothing with sat
anic imagery.'), ('Facebook posts', 'June 2, 2023', 'Disney "acaba de publica
r" que no puedes entrar a sus parques en Florida y California "sin que tus hi
jos reconozcan que no le pueden llamar a un niño él o ella".'), ('TikTok post
s', 'June 1, 2023', 'Texas Attorney General Ken Paxton was “caught on tape sa
ying he discarded 2.5 million mail-in ballots in Texas.”'), ('Facebook post
s', 'June 3, 2023', 'Boiling tap water causes fluoride in the water to be “mo
re toxic.”'), ('Pramila Jayapal', 'May 28, 2023', 'The average amount of assi
stance for the Supplemental Nutrition Assistance Program is $6 a day.'), ('Ni
kki Haley', 'June 4, 2023', 'Having "biological boys … in their locker rooms”
is a reason why “a third of our teenage girls seriously contemplated suicide
last year."'), ('Instagram posts', 'June 2, 2023', 'In a simulation, an AI-en
abled drone operated by the U.S. Air Force killed its operators and “started
taking out communication towers.”'), ('Ron DeSantis', 'May 25, 2023', 'Donald
Trump "wanted to amnesty 2 million illegal aliens in 2018 when he was preside
nt."'), ('Charlie Kirk', 'June 1, 2023', '“Jamie Foxx left ‘paralyzed and bli
nd’ from blood clot in his brain” after a COVID-19 vaccine injection.'), ('Al
Franken', 'May 17, 2023', '"Americans will get $1.1 B in rebates from health
insurance companies this year cuz of a provision I wrote in the ACA.”'), ('Fa
cebook posts', 'June 29, 202', 'The blue, pink, and white colors in the progr
ess pride flag represent pedophiles'), ('Instagram posts', 'May 31, 2023',
'"The European Union is now warning pregnant women not to get the COVID-19 va
ccine due to the possibility of infertility and miscarriage.”'), ('Instagram
posts', 'May 17, 2023', 'Photos show “Elon Musk and his company are in the fi
nal stages of making a Robot Wife.”'), ('Kevin McCarthy', 'May 28, 2023', '"E
very study has shown” that when work requirements are tied to federal safety-
net programs, “it puts more people to work.”'), ('Facebook posts', 'May 24, 2
023', 'The U.S. "takes 87% of all the prescription medications in the worl
d."'), ('Instagram posts', 'May 22, 2023', 'Video shows boxes of books remove
d from a middle school library that were “banned by the DeSantis regime in Fl
orida.”'), ('Facebook posts', 'May 22, 2023', 'La guanábana “es considerada c
omo la quimioterapia natural”.'), ('Facebook posts', 'May 24, 2023', '30 tons
of lost ammonium nitrate on a California-bound train suggest an orchestrated
conspiracy.'), ('Instagram posts', 'May 17, 2023', 'Pop-Tarts and other foods
“have antifreeze in them.”'), ('Derrick Van Orden', 'April 13, 20', "On excep
tions to Wisconsin's abortion ban"), ('Instagram posts', 'May 28, 2023', 'Vid
eo shows officials in Maricopa County, Arizona, "illegally breaking into" vot
ing machines.'), ('Ron DeSantis', 'May 24, 2023', "“There's not been a single
book banned in the state of Florida.”"), ('Tweets', 'May 29, 2023', 'Chick-fi
l-A “just hired a VP of Diversity, Equity and Inclusion.”'), ('TikTok posts',
'May 30, 2023', "Roseanne Barr made a video of a rainbow in a sprinkler quest
ioning 'what the heck is in our water supply.'"), ('Facebook posts', 'May 23,
2023', '“Arizona BANS Electronic Voting Machines.”'), ('Facebook posts', 'May
22, 2023', '"Si trabajaste en el año 2020 puede ser que te deban muchísimo di
nero” a través de un “crédito que está activo hasta ahora”.'), ('Instagram po
sts', 'May 26, 2023', '“Miami-Dade County has banned” Amanda Gorman’s poem “f
rom elementary schools.”'), ('Alex Epstein', 'May 5, 2023 ', 'A chart on Arct
ic sea ice provides evidence that Al Gore was wrong when he said in 2009 that
the north polar ice cap would lose all of its ice “within the next five to se
ven years.”'), ('Facebook posts', 'May 27, 2023', '"F.ight BREAKS as C.omer S
HUTS UP Biden\'s lawyer with B0MBSHELL."')]
```

```
In [ ]:
df=pd.DataFrame(combined_data,columns=['Source', 'Date', 'Statement'])
```

```
In [ ]:
df
```

Print data from number of web pages--politifacts

```
In [74]:
html_parsing_data=[]
no_of_articles=int(input("No of articles in multiple of 30:   "))
no_of_pages=no_of_articles/30
base_url='https://www.politifact.com/factchecks'
page=""
for k in range(int(no_of_pages)):
    url=base_url+page
    source = requests.get(url).text
    soup = bs4.BeautifulSoup(source,'html.parser')
    data=soup.prettify()
    x=soup.find_all('li',{'class':"o-listicle__item"})
    for i in x:
        a=i.find('a',class_='m-statement__name').get_text().strip()
        b=i.find('div',class_='m-statement__desc').get_text()[11:23]
        c=i.find('div',class_="m-statement__quote").get_text().strip()
        html_parsing_data.append((a,b,c))
    page="/?page="+str(k+1)+"&"
```

```
No of articles in multiple of 30:   90
```

```
In [75]:
df=pd.DataFrame(html_parsing_data,columns=['Source', 'Date', 'Statement'])
```

```
In [76]:
df
```

Out[76]:

	Source	Date	Statement
0	Instagram posts	June 5, 2023	“Hundreds of thousands of innocent people died...
1	Facebook posts	June 5, 2023	“There is no war in Ukraine.”
2	Facebook posts	June 5, 2023	Video shows Elon Musk and Jack Ma presenting “...
3	Bloggers	June 4, 2023	“Publix drops Ben and Jerry’s ‘for the good of...
4	TikTok posts	June 3, 2023	“Cocaine, porn, evidence of child trafficking ...
...	...	...	...
85	Facebook posts	May 10, 2023	Electric vehicles are a "giant computer that c...
86	Facebook posts	May 25, 2023	“Adam Schiff just got served” with a $16 milli...
87	Instagram posts	May 22, 2023	Says Bill Gates visited Jeffrey Epstein’s Isla...
88	Facebook posts	May 25, 2023	“50 U.S. politicians who received satellite ph...
89	MAGA Inc	May 24, 2023	“Ron DeSantis voted against the wall.”

90 rows × 3 columns

We will scrape the International Movies Database (IMDB) at imdb.com for top films released in year 2020 with the highest US box office.

I am organizing the final results as a dataframe with below elements:

name - title of the movie, year - release year of the movie, imdb - IMDB rating of the movie

Web Scraping from IMDB

In [1]:

```python
import bs4
import requests
import pandas as pd
url='https://www.imdb.com/search/title/?release_date=2020-01-01,2020-12-01
sourse=requests.get(url).text
soup=bs4.BeautifulSoup(sourse,'html.parser')
data=soup.prettify()
print(data)
```

```
<!DOCTYPE html>
<html xmlns:fb="http://www.facebook.com/2008/fbml" xmlns:og="http://ogp.me/
ns#">
 <head>
  <meta charset="utf-8"/>
  <script type="text/javascript">
   var IMDbTimer={starttime: new Date().getTime(),pt:'java'};
  </script>
  <script>
   if (typeof uet == 'function') {
      uet("bb", "LoadTitle", {wb: 1});
    }
  </script>
  <script>
   (function(t){ (t.events = t.events || {})["csm_head_pre_title"] = new Da
te().getTime(); })(IMDbTimer);
  </script>
  <title>
   Released between 2020-01-01 and 2020-12-01
```

In [3]:
```
data1=soup.find_all('div',{'class':'lister-item-content'})
print(data1)
```

```
[<div class="lister-item-content">
<h3 class="lister-item-header">
<span class="lister-item-index unbold text-primary">1.</span>
<a href="/title/tt1502397/">Bad Boys for Life</a>
<span class="lister-item-year text-muted unbold">(2020)</span>
</h3>
<p class="text-muted">
<span class="certificate">A</span>
<span class="ghost">|</span>
<span class="runtime">124 min</span>
<span class="ghost">|</span>
<span class="genre">
Action, Comedy, Crime            </span>
</p>
<div class="ratings-bar">
<div class="inline-block ratings-imdb-rating" data-value="6.5" name="ir">
<span class="global-sprite rating-star imdb-rating"></span>
<strong>6.5</strong>
</div>
```

In [4]:
```
print(len(data1))
```

```
50
```

In [5]:
```
data1[0].find('h3',class_='lister-item-header').get_text()
```

Out[5]:
```
'\n1.\nBad Boys for Life\n(2020)\n'
```

In [6]:
```
data1[0].find('a').get_text()
```

Out[6]:
```
'Bad Boys for Life'
```

In [7]:
```
data1[0].find('span',class_='lister-item-year text-muted unbold').get_text
```

Out[7]:
```
'2020'
```

In [8]:
```
data1[0].find('div',class_='inline-block ratings-imdb-rating').get_text().
```

Out[8]:
```
'6.5'
```

In [9]:
```
data1
```

Out[9]:
```
[<div class="lister-item-content">
 <h3 class="lister-item-header">
 <span class="lister-item-index unbold text-primary">1.</span>
 <a href="/title/tt1502397/">Bad Boys for Life</a>
 <span class="lister-item-year text-muted unbold">(2020)</span>
 </h3>
 <p class="text-muted">
 <span class="certificate">A</span>
 <span class="ghost">|</span>
 <span class="runtime">124 min</span>
 <span class="ghost">|</span>
 <span class="genre">
 Action, Comedy, Crime            </span>
 </p>
 <div class="ratings-bar">
 <div class="inline-block ratings-imdb-rating" data-value="6.5" name="ir">
 <span class="global-sprite rating-star imdb-rating"></span>
 <strong>6.5</strong>
 </div>
```

In [10]:
```python
import pandas as pd
movie_data=[]
l=len(data1)
for i in range(l):
    p=data1[i].find('a').get_text()
    try:
        q=data1[i].find('div',class_='inline-block ratings-imdb-rating').g
    except:
        q=None
    r=data1[i].find('span',class_='lister-item-year text-muted unbold').ge
    movie_data.append((p,q,r))
```

In [11]:
```
print(movie_data)
```

```
[('Bad Boys for Life', '6.5', '2020'), ('A Quiet Place Part II', '7.2', '202
0'), ('Sonic the Hedgehog', '6.5', '2020'), ('Birds of Prey and the Fantabulo
us Emancipation of One Harley Quinn', '6.1', '2020'), ('Dolittle', '5.6', '20
20'), ('The Invisible Man', '7.1', 'I) (2020'), ('The Call of the Wild', '6.
7', '2020'), ('Onward', '7.4', 'I) (2020'), ('The Croods: A New Age', '6.9',
'2020'), ('Tenet', '7.3', '2020'), ('Kimetsu no Yaiba: Mugen Ressha-Hen', '8.
2', '2020'), ('Fantasy Island', '4.9', '2020'), ('The New Mutants', '5.3', '2
020'), ('Unhinged', '6.0', 'I) (2020'), ('Underwater', '5.9', '2020'), ('Gret
el & Hansel', '5.5', '2020'), ('Monster Hunter', '5.2', 'I) (2020'), ('Honest
Thief', '6.0', '2020'), ('The Way Back', '6.7', 'I) (2020'), ('Bloodshot',
'5.7', '2020'), ('The Hunt', '6.5', 'II) (2020'), ('The Rhythm Section', '5.
4', '2020'), ('One Planet: Save It', '9.6', '2020'), ('Top Miami Gun Vice', N
one, '2020'), ('After We Collided', '5.0', '2020'), ('Scoob!', '5.6', '202
0'), ('Busanhaeng 2: Bando', '5.5', '2020'), ('The Serpent', '3.0', '2020'),
('Love and Monsters', '6.9', '2020'), ('Break the Silence: The Movie', '8.1',
'2020'), ('Stan the Man', '6.5', '2020'), ('Star Trek: First Frontier', '5.
5', '2020'), ('Timescape', None, 'I) (2020'), ('Ava', '5.4', 'IV) (2020'),
('Ba bai', '6.7', '2020'), ('Jiang Ziya', '6.5', '2020'), ('The Witches', '5.
4', '2020'), ('Come Away', '5.7', '2020'), ('Wendy', '5.7', '2020'), ('Tuls
a', '6.1', '2020'), ('Duo guan', '6.6', '2020'), ('Saving Mbango', '6.1', '20
20'), ('Breaking Bread', '7.8', '2020'), ('Our Story', None, '2020'), ('Dara
iz Jasenovca', '8.1', '2020'), ('The Way I See It', '8.3', '2020'), ('Tayo Ro
yalty Toy Review', None, '2020 TV Movie'), ('Black Wall Street Burning', '6.
8', '2020'), ('Limbo', None, 'VII) (2020'), ('Aloha Surf Hotel', '5.2', '202
0')]
```

In [12]:
```
df=pd.DataFrame(movie_data,columns=['Movie name', 'Rating', 'Year'])
```

```
In [13]:  print(df)
```

```
                                           Movie name Rating          Year
0                                  Bad Boys for Life    6.5          2020
1                                A Quiet Place Part II    7.2          2020
2                                  Sonic the Hedgehog    6.5          2020
3   Birds of Prey and the Fantabulous Emancipation...    6.1          2020
4                                            Dolittle    5.6          2020
5                                  The Invisible Man    7.1      I) (2020
6                                 The Call of the Wild    6.7          2020
7                                              Onward    7.4      I) (2020
8                                 The Croods: A New Age    6.9          2020
9                                               Tenet    7.3          2020
10                       Kimetsu no Yaiba: Mugen Ressha-Hen    8.2          2020
11                                     Fantasy Island    4.9          2020
12                                    The New Mutants    5.3          2020
13                                           Unhinged    6.0      I) (2020
14                                         Underwater    5.9          2020
15                                     Gretel & Hansel    5.5          2020
16                                     Monster Hunter    5.2      I) (2020
17                                       Honest Thief    6.0          2020
18                                       The Way Back    6.7      I) (2020
19                                         Bloodshot    5.7          2020
20                                           The Hunt    6.5     II) (2020
21                                  The Rhythm Section    5.4          2020
22                                 One Planet: Save It    9.6          2020
23                                  Top Miami Gun Vice   None          2020
24                                   After We Collided    5.0          2020
25                                             Scoob!    5.6          2020
26                                 Busanhaeng 2: Bando    5.5          2020
27                                        The Serpent    3.0          2020
28                                  Love and Monsters    6.9          2020
29                         Break the Silence: The Movie    8.1          2020
30                                       Stan the Man    6.5          2020
31                            Star Trek: First Frontier    5.5          2020
32                                         Timescape   None      I) (2020
33                                               Ava    5.4     IV) (2020
34                                            Ba bai    6.7          2020
35                                        Jiang Ziya    6.5          2020
36                                        The Witches    5.4          2020
37                                         Come Away    5.7          2020
38                                             Wendy    5.7          2020
39                                             Tulsa    6.1          2020
40                                          Duo guan    6.6          2020
41                                      Saving Mbango    6.1          2020
42                                     Breaking Bread    7.8          2020
43                                         Our Story   None          2020
44                                  Dara iz Jasenovca    8.1          2020
45                                     The Way I See It    8.3          2020
46                               Tayo Royalty Toy Review   None  2020 TV Movie
47                              Black Wall Street Burning    6.8          2020
48                                             Limbo   None    VII) (2020
49                                   Aloha Surf Hotel    5.2          2020
```

Print data from number of web pages--imdb.com

In [15]:

```
import pandas as pd
movie_data=[]
no_of_articles=int(input("No of movies in multiple of 50:   "))
no_of_pages=no_of_articles/50
base_url='https://www.imdb.com/search/title/?release_date=2020-01-01,2020-
page=""
for k in range(int(no_of_pages)):
    url=base_url+page
    source = requests.get(url).text
    soup = bs4.BeautifulSoup(source,'html.parser')
    data1=soup.find_all('div',{'class':'lister-item-content'})
    l=len(data1)
    for i in range(l):
        p=data1[i].find('a').get_text()
        try:
            q=data1[i].find('div',class_='inline-block ratings-imdb-rating
        except:
            q=None
        r=data1[i].find('span',class_='lister-item-year text-muted unbold'
        movie_data.append((p,q,r))
    page="&start="+str(50+((50*k)+1))+"&ref_=adv_nxt"
df1=pd.DataFrame(movie_data,columns=['Movie name', 'Rating', 'Year'])
```

```
No of movies in multiple of 50:   150
```

In [17]:

```
df1
```

Out[17]:

	Movie name	Rating	Year
0	Bad Boys for Life	6.5	2020
1	A Quiet Place Part II	7.2	2020
2	Sonic the Hedgehog	6.5	2020
3	Birds of Prey and the Fantabulous Emancipation...	6.1	2020
4	Dolittle	5.6	2020
...	...	...	...
145	Defending Jacob	7.8	2020
146	Freaky	6.3	2020
147	Host	6.5	II) (2020
148	Miss Scarlet & the Duke	7.7	2020–
149	Eurovision Song Contest: The Story of Fire Saga	6.5	2020

150 rows × 3 columns