Weather Data Collection: Web Scraping using Python

Somesh Routray May 8, 2020 · 5 min read

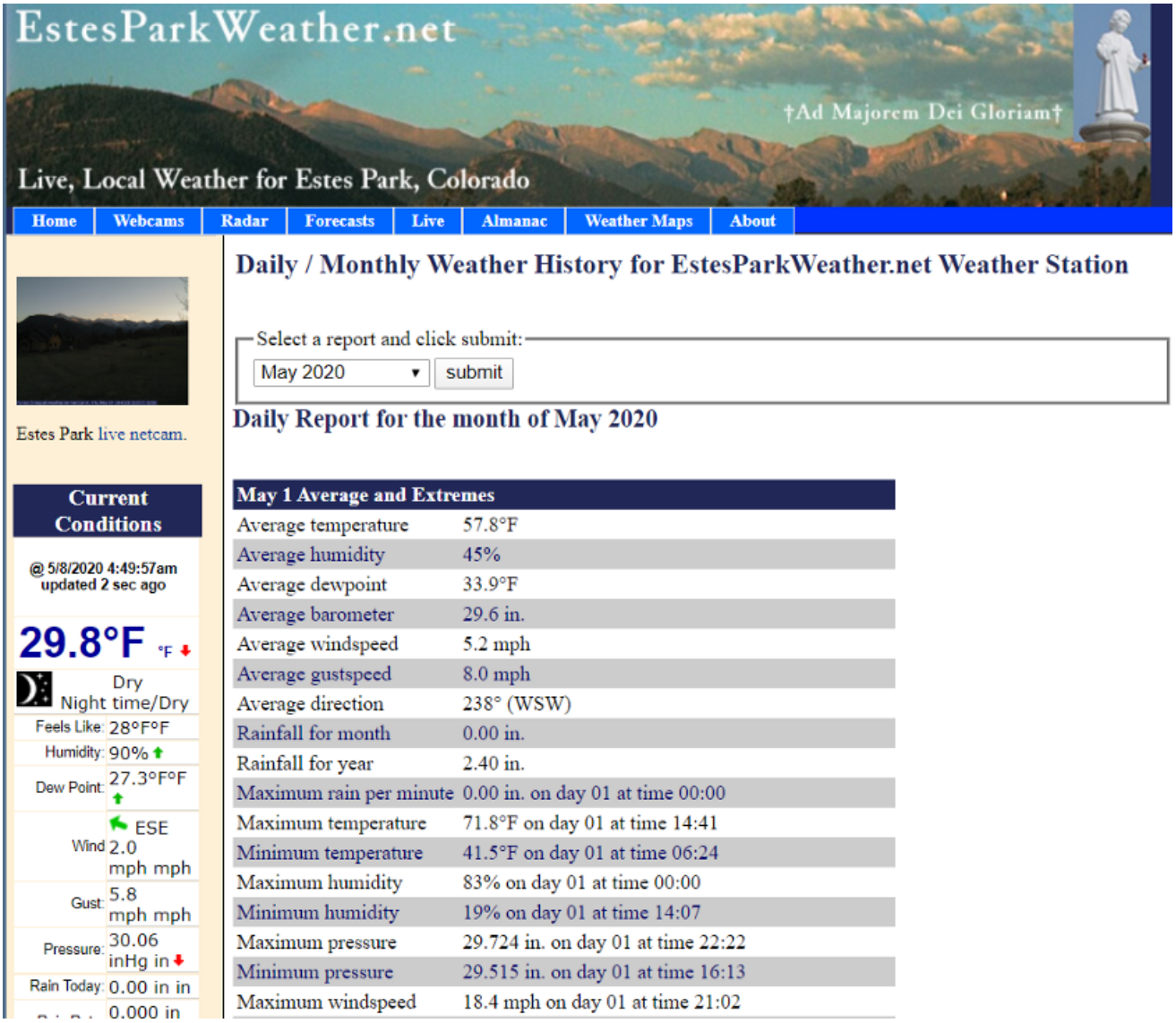
Image source

In this blog, I will be discussing how to scrape data from a website. Hopefully, you have

already visited my data collection technique using an API. In the real-world scenario, we may come across different data sources like databases, log files, Structured files, Services or API, etc. These sources may contain data in the form of records, graphs, or texts. Hence the data is available everywhere and to perform different innovative data science techniques we need to parse those data. We know Wikipedia is one of the biggest sources of data in the form of texts.

How to parse these kinds of textual data?

Here web scraping comes into the picture! It is a technique to extract the data using HTML tags. Here I will discuss this technique to scrape the weather data from the EstesPark Weather website. This website was primarily created as a public service for residents of Estes Park, Colorado, and Vicinity. Below is the screenshot of the website. You can follow the link to get the website.



Source: https://www.estesparkweather.net/archive_reports.php?date=202005

Some useful information about the website

1. The website contains date wise weather data like average temperature, average humidity, average dewpoint, etc. These data are store in the HTML web table.
2. There is a drop-down where you will get the liberty to choose the month and year to see the weather data.
3. Each time you change the drop-down selection, the date value will change according to the selected month and year but in ***yyyymm*** format. Refer to the image above as I choose **May 2020 ,** in the link, date value is changed to 202005.

> *https://www.estesparkweather.net/archive_reports.php?date=202005*

First import all required libraries for the case study

```
import bs4
from bs4 import BeautifulSoup
import requests
import pandas as pd
from datetime import datetime
```

Let's understand the piece of code.

```
url='https://www.estesparkweather.net/archive_reports.phpdate=202005'
page = requests.get(url)
print(page)
soup =  BeautifulSoup(page.content,'html.parser')
print(soup)
```

The **requests** library allows you to send HTTP requests easily and there's no need to manually add query strings to your url, or to form-encode your POST data. The urllib3 module inside the requests module makes the url in keep-alive state and you can pool the data continuously.

A working url will give you a status code 200. Which means the url is working fine.

BeautifulSoup library helps you to parse the HTML content on a webpage and XML content in XML file. The docstring of the BeautifulSoup is below.

> *Most of the methods you'll call on a BeautifulSoup object are inherited from PageElement or Tag.*
>
> *Internally, this class defines the basic interface called by the tree builders when converting an HTML/XML document into a data structure. The interface abstracts away the differences between parsers. To write a new tree builder, you'll need to understand these methods as a whole.*
>
> *These methods will be called by the BeautifulSoup constructor:*
> ** reset()*
> ** feed(markup)*
>
> *The tree builder may call these methods from its feed() implementation:*
> ** handle_starttag(name, attrs) # See note about return value*
> ** handle_endtag(name)*
> ** handle_data(data) # Appends to the current data node*
> ** endData(containerClass) # Ends the current data node*

End of the show, you should be able to build a tree using 'start tag' events, 'end tag' events, 'data' events, and "done with data" events.

Inside BeautifulSoup constructor I entered the HTML content of the url and given command as 'html.parser'. This will give me the HTML content of the webpage. For XML , you can use 'lxml'.

```
1 print(soup)
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<!-- DW6 -->
<head>
<!-- Copyright 2005 Macromedia, Inc. All rights reserved. -->
<meta content="300" http-equiv="Refresh"/>
<meta content="no-cache" http-equiv="Pragma"/>
<meta content="no-cache" http-equiv="Cache-Control"/>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
<meta content="37.27465, -122.02295" name="ICBM"/>
<meta content="Estes Park, Colorado Weather Station" name="DC.title"/>
<meta content="Gregory Truta" name="author"/>
<meta content="© 2007 EstesParkWeather.net" name="copyright"/>
<meta content="weather, Weather, temperature, dew point, humidity, forecast, Davis Vantage Pro, Estes Park Colorado Weather,
Colorado Weather, CO Weather, weather conditions, live weather, live weather conditions, weather data, weather history, Weath
erLink, Weather-Display" name="Keywords"/>
<meta content="Weather conditions for Estes Park, Colorado" name="Description"/>
<link href="favicon.ico" rel="SHORTCUT ICON" type="image/x-icon"/>
<title>Estes Park Weather - Home/Forecasts</title>
```

Copyright Somesh

What is the soup then?

It is a BeautifulSoup object that has methods designed specifically to work with HTML content.

```
1 type(soup)
```

```
bs4.BeautifulSoup
```

Copyright Somesh

As I mentioned earlier the weather data on the website is in the HTML web table form. So we look for the table in the HTML content and analyze the rows and columns.

```
table = soup.find_all('table')

raw_data = [row.text.splitlines() for row in table]
raw_data = raw_data[:-9]

for i in range(len(raw_data)):
 raw_data[i] = raw_data[i][2:len(raw_data[i]):3]
print(raw_data)
```

In the drop-down when we select May 2020, it will give you separate tables with all the weather attribute values for each day. So we will split the rows in the table and append in a list. And finally, you will get a list of lists. Noteworthy, each sub-lists contain weather attributes and their values for a particular month.

How to get weather data of other months ?

As mentioned earlier, the link contains the page value equals to drop-down value but in yyyymm format. So you make a date range and strip the dates into the required format and concatenate with the string urls.

```
Dates_r = pd.date_range(start = '1/1/2009',end = '08/05/2020',freq =
'M')
dates = [str(i)[:4] + str(i)[5:7] for i in Dates_r]
dates[0:5]
```

```
for k in range(len(dates)):
    url = "http://www.estesparkweather.net/archive_reports.php?date="
    url += dates[k]
```

Now, you can perform string stripping techniques to create a data set. An example is below.

```
for u in url:
 for i in range(len(raw_data)):
 c = [‘.’.join(re.findall(“\d+”,str(raw_data[i][j].split()[:5])))for
j in range(len(raw_data[i]))]
 df_list.append(c)
 index.append(dates[k] + c[0])
 f_index = [index[i] for i in range(len(index)) if len(index[i]) > 6]
 data = [df_list[i][1:] for i in range(len(df_list)) if
len(df_list[i][1:]) == 19]
```

To make the dates as an index to the dataset you can use the below code.

```
final_index = [datetime.strptime(str(f_index[i]),
‘%Y%m%d’).strftime(‘%Y-%m-%d’) for i in range(len(f_index))]
```

You can make a list of the weather attributes such as humidity, temperature, rainfall for column names or you can choose custom column names for your data.

Congratulations !!! You are at a PANDAS touch far from creating a DataFrame.

In DataFrame() you will give ***data*** value as the data you have made for all the weather attribute values, ***columns*** equals to the list variable containing custom column names and finally the ***index*** value will be the list variable of dates.

```
22 df.head()
```

Out[25]:

	Average temperature (°F)	Average humidity (%)	Average dewpoint (°F)	Average barometer (in)	Average windspeed (mph)	Average gustspeed (mph)	Average direction (°deg)	Rainfall for month (in)	Rainfall for year (in)	Maximum rain per minute	Maximum temperature (°F)	Minimum temperature (°F)	Maximum humidity (%)	Minimu humid (
2020-07-01	57.8	45	33.9	29.6	5.2	8.0	238	0.00	2.40	0.00	71.8	41.5	83	
2020-07-02	47.1	83	42.0	29.7	3.1	5.0	16	0.30	2.70	0.02	57.3	39.9	95	
2020-07-03	51.6	59	36.4	29.7	4.9	7.5	244	0.33	2.73	0.01	65.4	36.1	93	
2020-07-04	45.1	58	28.9	29.8	7.6	11.8	248	0.37	2.77	0.01	52.2	38.5	94	
2020-07-05	48.7	24	12.2	30.0	6.3	9.7	241	0.37	2.77	0.00	62.3	37.9	50	

I'd love to hear any comments about the above analysis — feel free to leave a message below,

Python Data Science AI Machine Learning Python Programming