DATA VISUALIZING FROM CSV FORMAT TO CHART USING PYTHON

JULY 4, 2019

In this article, we will download a data set from an online resource and create a working visualization of that data. As we have discussed in the **previous article**, the internet is being bombarded by lots of data each second. An incredible amount and variety of data can be found online. The ability to analyze data allows you to discover the patterns and connections.

We will access and visualize the data store in **CSV** format. We will use **Python's CSV** module to process weather data. We will analyze the high and low temperatures over the period in two different locations. Then we will use **matplotlib** to generate a chart.

By the end of this article, you'll be able to work with different datasets and build complex **visualizations**. It is essential to be able to access and

visualize online data which contains a wide variety of real-world datasets.

THE CSV FILE FORMAT

CSV stands for **C**omma **S**eparated **V**alues. As the name suggests, it is a kind of file in which values are separated by commas.

It is one of the simpler ways to store the data in a textual format as a series of comma separated values. The resulting file is called a **CSV** file.

For example, below given is a line of weather data on which we are going to work in this article.

2014-1-6,62,43,52,18,7,-1,56,33,9,30.3,30.2,. . ., 195

This is the weather data. We will start with a small dataset. It is **CSV** formatted data **Stika, Alaska**. You can download datasets from https://www.wunderground.com. This is how your **CSV** file of data will look like:

	A	B	C	D	E	F	G
1	STATION	NAME	DATE	PRCP	TAVG	TMAX	TMIN
2	USW0002	SITKA AIRF	7/1/2018	0.25		62	50
3	USW0002	SITKA AIRF	7/2/2018	0.01		58	53
4	USW0002	SITKA AIRF	7/3/2018	0		70	54
5	USW0002	SITKA AIRF	7/4/2018	0		70	55
6	USW0002	SITKA AIRF	7/5/2018	0		67	55
7	USW0002	SITKA AIRF	7/6/2018	0		59	55
8	USW0002	SITKA AIRF	7/7/2018	0		58	55
9	USW0002	SITKA AIRF	7/8/2018	0		62	54
10	USW0002	SITKA AIRF	7/9/2018	0		66	55
11	USW0002	SITKA AIRF	########	0.44		59	53
12	USW0002	SITKA AIRF	########	0.29		56	50
13	USW0002	SITKA AIRF	########	0.02		63	49
14	USW0002	SITKA AIRF	########	0		65	48

The first line is called the header.

PARSING THE CSV FILE HEADERS

We can easily parse the values and extract the required information using the **Python's csv** module. Let's start by analyzing the first line of the file which contains the headers used for data.

1. Create a **python** file name **weather_data.py**

2. Write the following statement to import the **CSV** module:

```
import csv
```

3. Download the data file from here.

DOWNLOAD DATA FILE

4. Open the file using Python's **open** function and print the headers:

```
1.  filename = 'sitka_weather_07-2018_simple.csv'
2.  with open(filename) as f:
3.    reader = csv.reader(f)  #line 1
4.    header_row = next(reader)  #line 2
5.    print(header_row)  #line 3
```

After importing the **CSV** module, we store the name of the file in the variable **filename**. We then open the file using the **open** function and store the result file object in **f**.

Next, on line 1, we have called the **reader** function of the **CSV** module and passed the file object **f** to it as an argument. This function creates a reader object associated with that file.

The **CSV** module contains a **next()** function which returns the next line in the file. **next()** function accepts a reader object as an argument. So, it returns the next line of the file with which reader object is associated. We only need to call the **next()** function once to get the first line of the file which contains header normally. So, the header is stored in the variable **header_row** . The next line, just prints the header row.

Here is the output of the above code snippet:

['STATION', 'NAME', 'DATE', 'AWND', 'PGTM', 'PRCP', 'SNWD', 'TAVG', 'TMAX', 'TMIN', 'WDF2', 'WDF5', 'WSF2', 'WSF5', 'WT01', 'WT02', 'WT04', 'WT05', 'WT08']

reader processes the first line of comma-separated values and stores each as an item in the list. For example, **TMAX** denotes maximum temperature for that day.

PRINTING THE HEADERS WITH THEIR POSITIONS

We usually print header with their position in the list, to make it easier to understand the file header data.

```
1.  import csv
2.
3.  filename = 'sitka_weather_2018_full.csv'
4.
5.  with open(filename) as f:
6.      reader = csv.reader(f)
7.      header_row = next(reader)
8.      for index,column_header in enumerate(header_row):
9.          print(index, column_header)
```

OUTPUT:

0 STATION

1 NAME

2 DATE

3 AWND

4 PGTM

5 PRCP

6 SNWD

7 TAVG

8 TMAX

9 TMIN

10 WDF2

11 WDF5

12 WSF2

13 WSF5

14 WT01

15 WT02

16 WT04

17 WT05

18 WT08

We have used the **enumerate()** function on the list to get the index of each item in the list and as well the value. **NOTE:** We have removed the **print(header_row)** line to get a more detailed version of data.

Here we can see that the **date** and respective **max temperature** are stored in column **2** and **8** respectively. Let's extract these.

EXTRACTING AND READING DATA

Now that we know which columns of data we need, let's read in some of that data. First, we'll read in the high temperature for each day:

```
highs = []
    for row in reader:
        highs.append(row[8])  #appending high temperatures
    print(highs)
```

We make an empty list named highs. Then, we iterate through each row in the reader and keep appending the high temperature which is available on index 8 to the list. The reader object continues from where it left in the **CSV** file and automatically returns a new line on its current position. Then we print the list. It looks like below:

['48', '48', '46', '42', '46', '44', '39', '36', '34', '28', '34', '41', '53', '63', '60', '54', '47', '46', '42', '45', '43', '41', '41', '40', ... ,]

As we can see that list returned is in the form of strings. Now, we convert these strings to number using **int()** so that they can be read by **matplotlib**.

```
for row in reader:
  if row[8] == ‘’:
    continue   # There are some empty strings which can’t be converted to int
high = int(row[8]) #Convert to int
highs.append(high)  #appending high temperatures
```

Now our data is ready to for plotting.

[48, 48, 46, 42, 46, 44, 39, 36, 34, 28, 34, 41, 53, 63, 60, 54, 47, 46, 42, 45, 43, 41, 41, 40, 40, 41, 39, 40, 40, 39, 36, 35, 35, 34, 42, 41, 39, 42, 39, 37, 37, 40, 43, 41, 40, 38, 36, 37, 39, 39, 38, 41, 42, 41, 39, 38, 42, 39, 40,. . .,]

PLOTTING DATA IN TEMPERATURE CHART USING MATPLOTLIB

To visualize the temperature data, we will first create a plot of daily high temperatures using **matplotlib**.

So, here’s the Code.

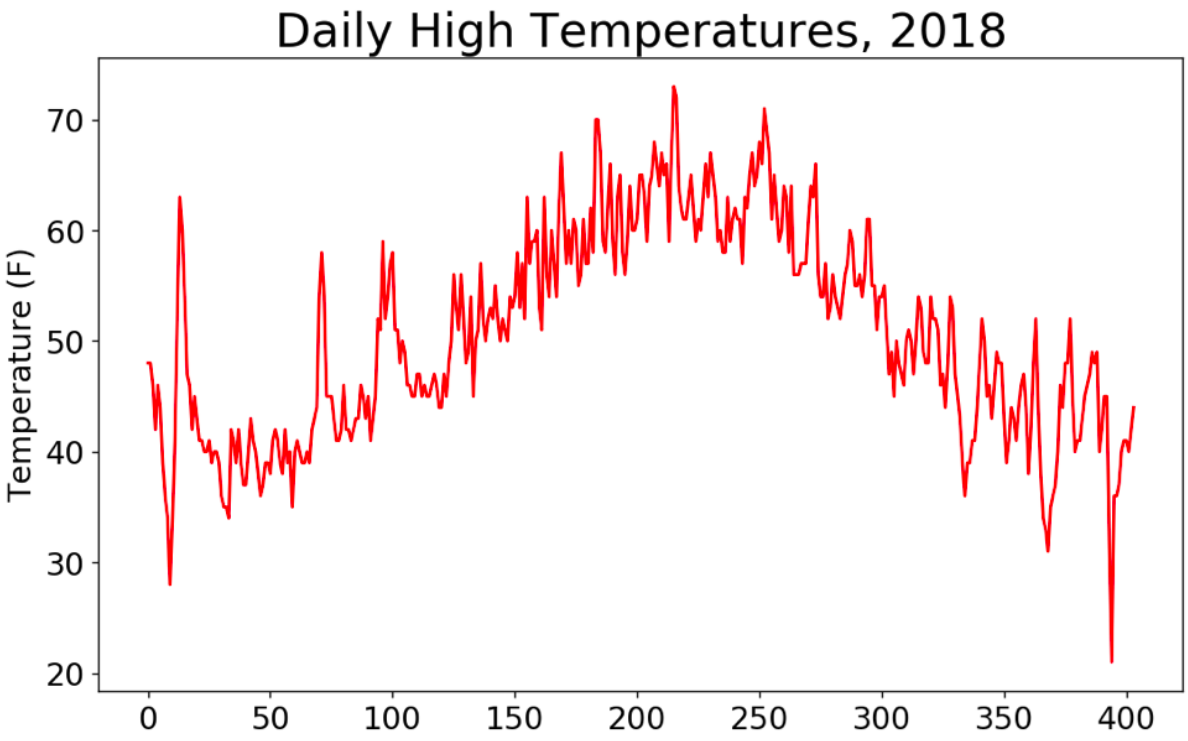
```
import csv
from matplotlib import pyplot as plt
filename = 'sitka_weather_2018_full.csv'

with open(filename) as f:
    reader = csv.reader(f)
    header_row = next(reader)
    highs = []
```

```
    for row in reader:
        if row[8]=='':
            continue
        high = int(row[8],10)
        highs.append(high)  #appending high temperatures

    #Plot Data
    fig = plt.figure(dpi = 128, figsize = (10,6))
    plt.plot(highs, c = 'red') #Line 1
    #Format Plot
    plt.title("Daily High Temperatures, 2018", fontsize = 24)
    plt.xlabel('',fontsize = 16)
    plt.ylabel("Temperature (F)", fontsize = 16)
    plt.tick_params(axis = 'both', which = 'major' , labelsize = 16)
    plt.show()
```

Running the Above Code, you'll get this.



Thank you for reading. I hope It will be helpful. Comment If you find any difficulty. I'll love to solve your problem.

Here’re some more Articles, you might be interested:

— Data Visualization in Python Using Simple Line Chart

— Developing Chat Application in Python with Source Code

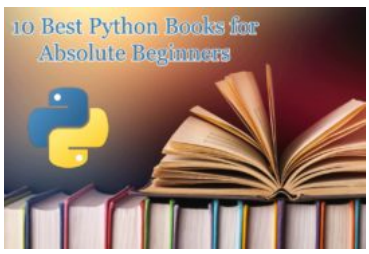
— Top 5 Python Web Frameworks to Learn

AUTHOR

AHMED BASHIR

RELATED POSTS

5 SITE IDEAS FOR YOUR



10 BEST PYTHON

SENDING SMS USING

FIRST PRACTICE RUN

NOVEMBER 6, 2019

BOOKS FOR ABSOLUTE BEGINNERS

OCTOBER 24, 2019

PYTHON AND TWILIO API – TUTORIAL

SEPTEMBER 22, 2019

WRITE A COMMENT

Name

Email

Website

Enter your comment here..

Save my name, email, and website in this browser for the next time I comment.

Notify me when new comments are added.

POST COMMENT

Type and hit enter...

Email Address *

Choose Technology *

SUBSCRIBE

TOP ARTICLES

SIGNUP LOGIN PAGE IN PHP WITH DATABASE MYSQL SOURCE CODE

MAY 27, 2018

HERE ARE THE TEN BEST PROGRAMMING LANGUAGES TO LEARN IN 2019

DECEMBER 22, 2018

CRUD OPERATIONS IN ASP.NET CORE MVC

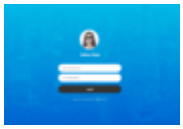
NOVEMBER 10, 2018

CAR LOCATION TRACKING ANDROID APP WITH FIREBASE TUTORIAL

AUGUST 28, 2018

OPEN SOURCE BULK SMS SENDER ANDROID APP

APRIL 16, 2018



LOGIN PAGE IN ASP.NET CORE MVC WITH DATABASE

OCTOBER 31, 2018

CRUD OPERATIONS WEB APP USING PHP & MYSQL | PART 2

MARCH 12, 2019

TOP