Explore Weather Trends (Term 1)

JANUARY 28

Udacity Nano Degree (Data Analysis)
Student: Suman Debnath
Country: India

UDACITY

Explore Weather Trends

NanoDegree Term1 - Project

Project:

In this project, we had to analyze local and global temperature data and compare the temperature trends where you live to overall global temperature trends.

For this project I used Python to extract data from CSV, though it was not yet covered in the class so far, but I just thought to get started to get a feel, and I am quite happy to learn few interesting stuffs with Pandas, and looking forward to learn from scratch in the upcoming class duing the Nano Degree

Github:

All the code and data is checked in my Github reposotory:

https://github.com/suman-d/UdacityNanoDegreeDataAnalysis

Extract the data

At the very beginning, we extracted the data from the Database using SQL and download the data in CSV.

```
SELECT *
FROM global_data; # Saved in global_data.csv

SELECT *
FROM city_data;   # Saved in city_data.csv

SELECT *
FROM city_list;   # Saved in city_list.csv
```

Open up the CSV

Next, I tried to load the CSV data using Python(Pandas) and created 3 DataFrame

```
In [73]: import pandas as pd
         import matplotlib
         import matplotlib.pyplot as plt
         import matplotlib.ticker as ticker

         import warnings
         warnings.filterwarnings('ignore')

         %matplotlib inline
```

```
In [74]: df_city_data = pd.read_csv('city_data.csv')
         df_global_data = pd.read_csv('global_data.csv')
         df_city_list = pd.read_csv('city_list.csv')
```

Create a line chart

Next I tried to create a function which takes a city_data, global_data and my_city name and returns a DataFram which contains the temperature data for my_city and the world.

As in the data set there are lots of years the data was not present, so I dropped those year's data.

The fnction also returns the cofficient of correlation of the city data w.r.t global data.

I used the “rolling” method for DataFrame to calculate the moving average and appended a new column with the moving average.

The iPython Notebok with the code can be found @ [my github account suman-d]

In [75]:
```
def mycity_to_global(city_df, global_df, city='Pune'):
    '''
    This function takes the following:
    city_df -> [DataFrame] All city temperature,
    global_df -> [DataFrame] The global temperature and
    city -> [String] one city name(mycity) and

    Returns a tuple of :
    1. DataFram which contains which contains both the global temperature and the city temperature
    2. DataFram with the coff. of correlation between the global temperature and city temperature

    '''
    df_my_city_data = city_df[city_df.city == city]
    list_my_city_year = list(df_my_city_data['year'])

    df_my_global_data = global_df.loc[global_df['year'].isin(list_my_city_year)]
    df_my_global_data.index = range(len(df_my_global_data))
    df_my_global_data.rename(columns={'avg_temp':'global_avg_temp'}, inplace=True)

    df = pd.merge(df_my_city_data, df_my_global_data, on='year')
    df.rename(columns={'avg_temp': "{}_avg_temp".format(city.lower())}, inplace=True)

    city_col = 'moving_avg_{}'.format(city.lower())

    df['moving_avg_global'] = df['global_avg_temp'].rolling(window=2).mean()
    df[city_col] = df['{}_avg_temp'.format(city.lower())].rolling(window=2).mean()

    df.dropna(axis=0, inplace=True)

    # Calculating the Coff. of Correlation
    coff = df[[city_col, 'moving_avg_global']].corr()

    return df, coff
```

Now, I tried to extract my city data(Pune) and the Cofficient of Corelation

```
In [76]: df, c_pune = mycity_to_global(df_city_data, df_global_data, city='Pune')

In [77]: len(df)
Out[77]: 208

In [78]: df.head()
```

Out[78]:

	year	city	country	pune_avg_temp	global_avg_temp	moving_avg_global	moving_avg_pune
1	1797	Pune	India	25.17	8.51	8.390	24.780
2	1798	Pune	India	24.05	8.67	8.590	24.610
3	1799	Pune	India	24.68	8.51	8.590	24.365
4	1800	Pune	India	24.67	8.48	8.495	24.675
5	1801	Pune	India	23.94	8.59	8.535	24.305

And Cofficient of Corelation

```
In [80]: c_pune.head()
```

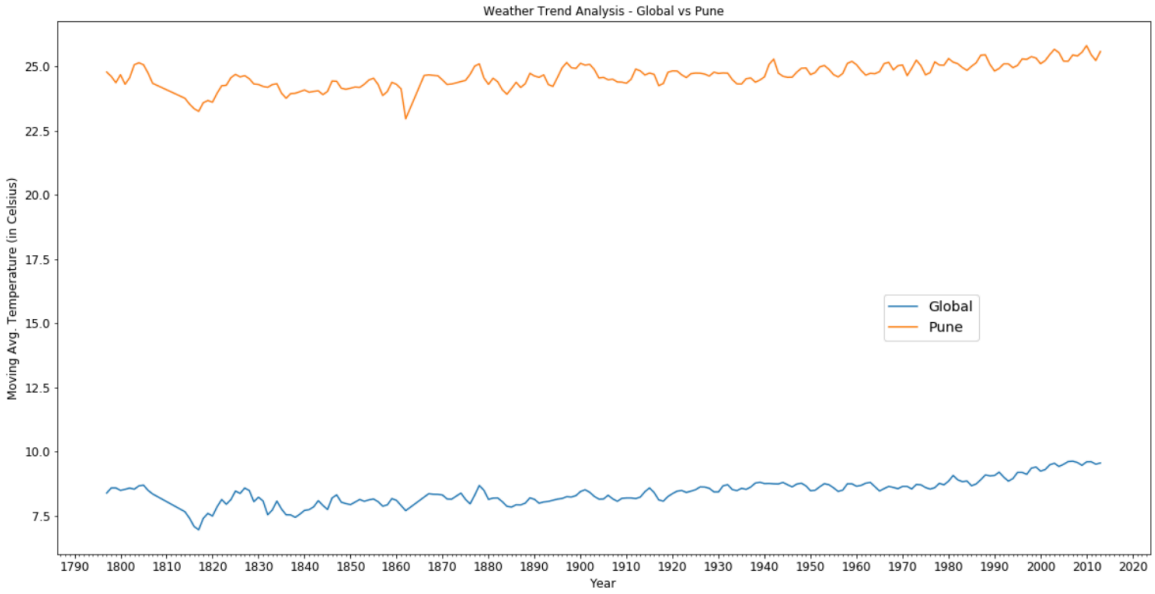
Out[80]:

	moving_avg_pune	moving_avg_global
moving_avg_pune	1.000000	0.870916
moving_avg_global	0.870916	1.000000

Next I tried to write a function which can help to generate the line graph and also help to save the graph in a PNG file, which we can re-use later.

In [81]:
```
def gen_charts(df, cities, saveas):
    '''
    This function takes "df", "cities", "saveas", where:

    df -> [DataFrame] The DataFrame which contains all the temperature data (for all the cities in
cluding global temperature)
    cities -> [List] A list of all the cities
    saveas -> [String] Some name of the image file, which this function will generate and save

    '''
    ax = plt.axes()
    ax.xaxis.set_major_locator(ticker.MultipleLocator(10))
    ax.xaxis.set_minor_locator(ticker.MultipleLocator(1))

    plt.rc('font', size=12)
    plt.rc('axes', titlesize=12)
    plt.rc('axes', labelsize=12)
    plt.rc('xtick', labelsize=12)
    plt.rc('ytick', labelsize=12)
    plt.rc('legend', fontsize=12)
    plt.rc('figure', titlesize=12)

    plt.plot(df.year, df.moving_avg_global)
    for c in cities:
        plt.plot(df.year, df['moving_avg_{}'.format(c.lower())])

    cities.insert(0, 'Global')
    plt.legend(cities, bbox_to_anchor=(.85,0.5), fontsize='large')


    plt.xlabel("Year")
    plt.ylabel("Moving Avg. Temperature (in Celsius)")
    title = " vs ".join(cities)
    plt.title("Weather Trend Analysis - " + title)
    #plt.legend(["Global", "India(Pune)"], bbox_to_anchor=(.85,0.5), fontsize='large')
    plt.rcParams["figure.figsize"] = [20,10]

    plt.savefig(saveas)
```

In [83]:
```
df, c_pune = mycity_to_global(df_city_data, df_global_data, city='Pune')
p = gen_charts(df, cities=['Pune'], saveas="Trend_Analysys_Pune.png")
```



Observation:

The chart shows that my city (Pune) is quict hot w.r.t average global temperature,
The chart also suggests that from 1960 onwards, the planet is gradually getting hotter, as we can observe a slight consistent inclination in temperature.

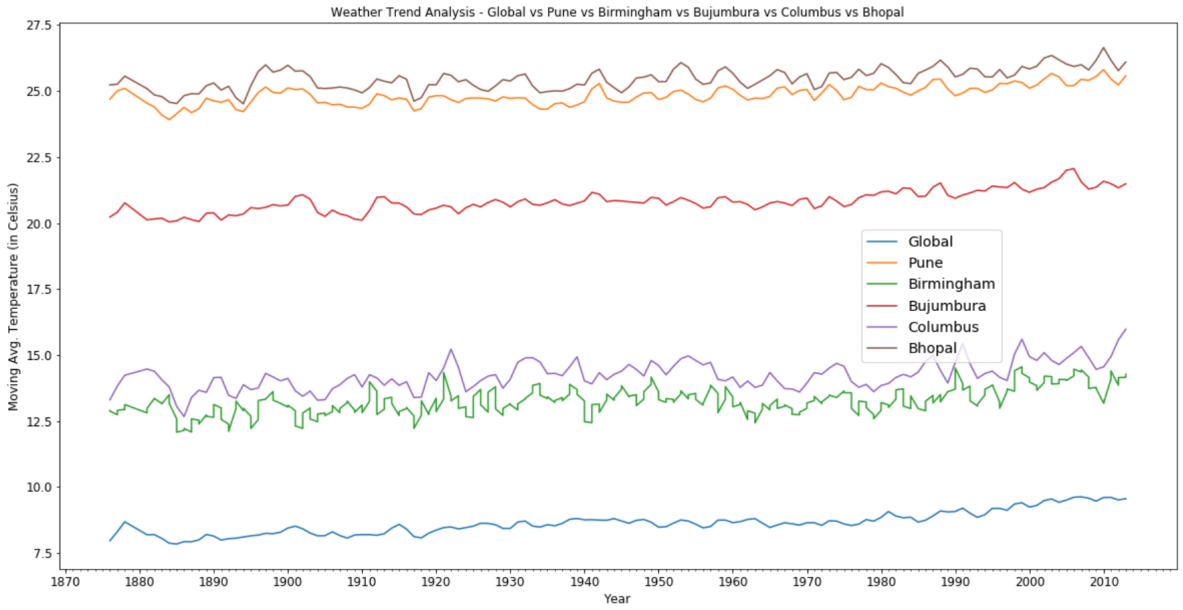
Now, I wanted to check where my city temperature looks like w.r.t other city.
So, I tried to create a new function which can take a list of cities which we want to explore and returns a datafram with all the cities temperature.

In [88]:
```
def manycity_to_globle(city_df, global_df, cities=['Pune']):
    '''
    This function takes the following:
    city_df -> [DataFrame] All city temperature,
    global_df -> [DataFrame] The global temperature and
    city -> [List] one or more city

    Returns:
    DataFram which contains which contains all the city temperature and the global temperature

    '''
    city_dfs = []
    city_crr = []
    for city in cities:
        _df, _crr = mycity_to_global(city_df, global_df, city=city)
        city_dfs.append(_df)
        city_crr.append(_crr)

    for d in city_dfs:
        global_df = global_df.merge(d, on='year')

    dup_column = [i for i in global_df.columns if i.startswith('moving_avg_global_') \
                                                  or i.startswith('global_avg_temp_')]
    global_df.drop(dup_column, axis=1, inplace=True)

    return global_df
```

Now, I tried to check multiply cities data

cities = ['Pune', 'Birmingham', 'Bujumbura', 'Columbus', 'Bhopal']

```
In [90]: cities = ['Pune', 'Birmingham', 'Bujumbura', 'Columbus', 'Bhopal']
df = manycity_to_globle(df_city_data, df_global_data, cities=cities)
gen_charts(df, cities=cities, saveas="Trend_Analysys_Multiple_City.png")
```



So, now we can check any city temperature data, and compre the same with other city tempature or/and global temperature. Also we can now know the coefficient of correletation, so we can predict one citiy's temperature if we know the others and the confficient of correlation ☺