Pie Charts, Box Plots, Scatter Plots, and Bubble Plots

Estimated time needed: **30** minutes

Objectives

After completing this lab you will be able to:

- Explore Matplotlib library further
- Create pie charts, box plots, scatter plots and bubble charts

Table of Contents

Importing Libraries

```
#Import primary modules.
import numpy as np  # useful for many scientific computing in Python
import pandas as pd # primary data structure library

#Importing Matplotlib
#%matplotlib inline

import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.style.use('ggplot') # optional: for ggplot-like style

# check for latest version of Matplotlib
print('Matplotlib version: ', mpl.__version__) # >= 2.0.0
```

```
Matplotlib version:  3.5.3
```

Importing Data

Dataset: Immigration to Canada from 1980 to 2013 - International migration flows to and from selected countries - The 2015 revision from United Nation's website. In this lab, we will focus on the Canadian Immigration data and use the *already cleaned dataset* and can be fetched from here. You can refer to the lab on data pre-processing wherein this dataset is cleaned for a quick refresh your Panads skill Data pre-processing with Pandas

```
df_can = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/Canada.csv')
```

```
print('Data read into a pandas dataframe!')
```

```
Data read into a pandas dataframe!
```

```
df_can.head()
```

```
          Country Continent           Region             DevName  1980
1981  \
0     Afghanistan      Asia    Southern Asia  Developing regions    16
39
1         Albania    Europe  Southern Europe   Developed regions     1
0
2         Algeria    Africa  Northern Africa  Developing regions    80
67
3  American Samoa   Oceania        Polynesia  Developing regions     0
1
4         Andorra    Europe  Southern Europe   Developed regions     0
0

   1982  1983  1984  1985  ...  2005  2006  2007  2008  2009  2010
2011  \
0    39    47    71   340  ...  3436  3009  2652  2111  1746  1758
2203
1     0     0     0     0  ...  1223   856   702   560   716   561
539
2    71    69    63    44  ...  3626  4807  3623  4005  5393  4752
4325
3     0     0     0     0  ...     0     1     0     0     0     0
0
4     0     0     0     0  ...     0     1     1     0     0     0
0

   2012  2013  Total
0  2635  2004  58639
1   620   603  15699
2  3774  4331  69439
3     0     0      6
4     1     1     15

[5 rows x 39 columns]
```

Let's find out how many entries there are in our dataset.

```
# print the dimensions of the dataframe
print(df_can.shape)
```

```
(195, 39)
```

Visualizing Data using Matplotlib

For plotting the data easilty, let's first set the country name as index - useful for quickly looking up countries using .loc method.

```
df_can.set_index('Country', inplace=True)

# Let's view the first five elements and see how the dataframe was changed
df_can.head()
```

```
                Continent           Region              DevName  1980
1981  \
Country

Afghanistan         Asia    Southern Asia  Developing regions    16
39
Albania           Europe  Southern Europe   Developed regions     1
0
Algeria           Africa  Northern Africa  Developing regions    80
67
American Samoa   Oceania        Polynesia  Developing regions     0
1
Andorra           Europe  Southern Europe   Developed regions     0
0

                1982  1983  1984  1985  1986  ...  2005  2006  2007
2008  \
Country                                       ...

Afghanistan       39    47    71   340   496  ...  3436  3009  2652
2111
Albania            0     0     0     0     1  ...  1223   856   702
560
Algeria           71    69    63    44    69  ...  3626  4807  3623
4005
American Samoa     0     0     0     0     0  ...     0     1     0
0
Andorra            0     0     0     0     2  ...     0     1     1
0

                2009  2010  2011  2012  2013  Total
Country
Afghanistan     1746  1758  2203  2635  2004  58639
Albania          716   561   539   620   603  15699
Algeria         5393  4752  4325  3774  4331  69439
American Samoa     0     0     0     0     0      6
Andorra            0     0     0     1     1     15

[5 rows x 38 columns]
```

Notice now the country names now serve as indices.

```
print('data dimensions:', df_can.shape)
```

```
data dimensions: (195, 38)
```

Finally, let's create a list of years from 1980 - 2013, this will come in handy when we start plotting the data

```
years = list(map(str, range(1980, 2014)))
print(years)
```

```
['1980', '1981', '1982', '1983', '1984', '1985', '1986', '1987',
'1988', '1989', '1990', '1991', '1992', '1993', '1994', '1995',
'1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011',
'2012', '2013']
```

Pie Charts

A `pie chart` is a circular graphic that displays numeric proportions by dividing a circle (or pie) into proportional slices. You are most likely already familiar with pie charts as it is widely used in business and media. We can create pie charts in Matplotlib by passing in the `kind=pie` keyword.

Let's use a pie chart to explore the proportion (percentage) of new immigrants grouped by continents for the entire time period from 1980 to 2013.

Step 1: Gather data.

We will use *pandas* `groupby` method to summarize the immigration data by `Continent`. The general process of `groupby` involves the following steps:

1. **Split:** Splitting the data into groups based on some criteria.
2. **Apply:** Applying a function to each group independently: .sum() .count() .mean() .std() .aggregate() .apply() .etc..
3. **Combine:** Combining the results into a data structure.

```
# group countries by continents and apply sum() function
df_continents = df_can.groupby('Continent', axis=0).sum()

# note: the output of the groupby method is a `groupby' object.
# we can not use it further until we apply a function (eg .sum())
print(type(df_can.groupby('Continent', axis=0)))

df_continents.head()
```

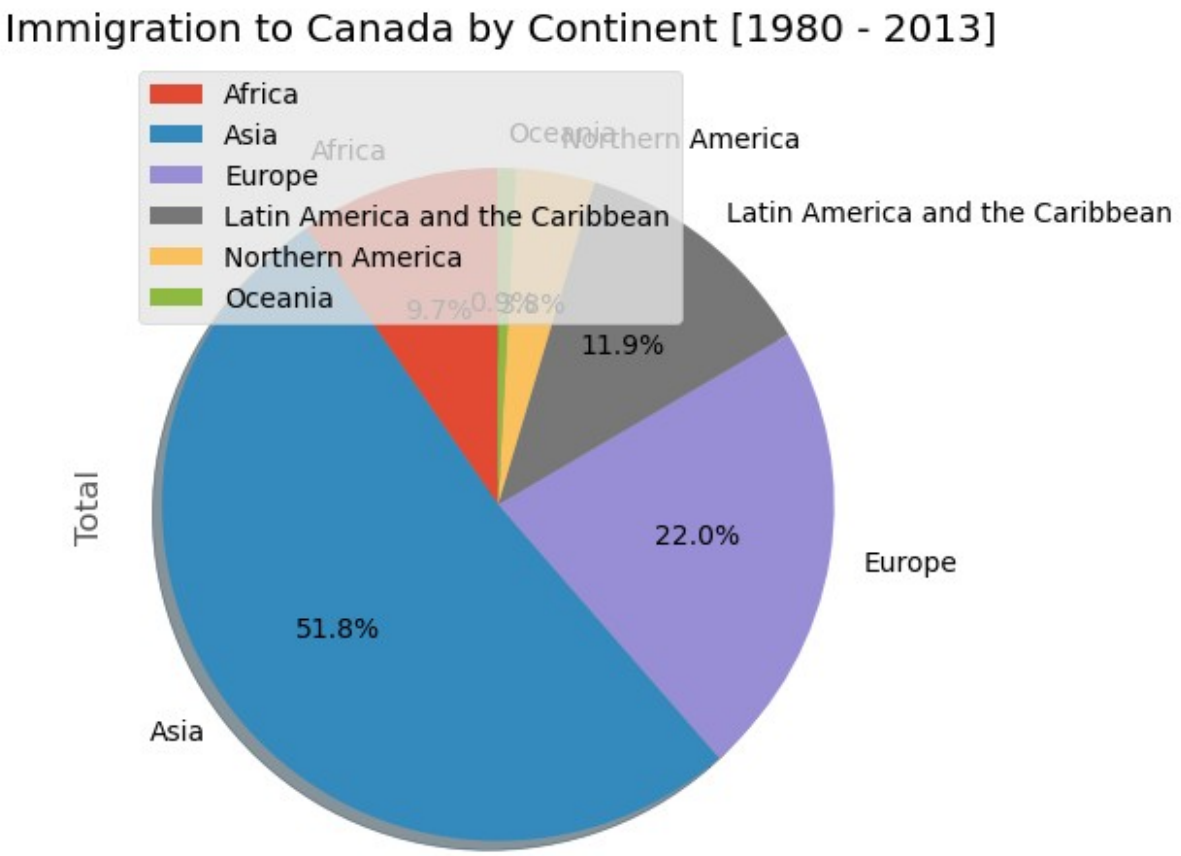
```
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>

                                  1980   1981   1982   1983   1984
1985  \
Continent

Africa                             3951   4363   3819   2671   2639
2650
Asia                              31025  34314  30214  24696  27274
23850
Europe                            39760  44802  42720  24638  22287
20844
Latin America and the Caribbean   13081  15215  16769  15427  13678
15171
Northern America                   9378  10030   9074   7100   6661
6543

                                  1986   1987   1988   1989  ...
2005  \
Continent                                                   ...

Africa                             3782   7494   7552   9894  ...
27523
Asia                              28739  43203  47454  60256  ...
159253
Europe                            24370  46698  54726  60893  ...
35955
Latin America and the Caribbean   21179  28471  21924  25060  ...
24747
Northern America                   7074   7705   6469   6790  ...
8394

                                   2006    2007    2008    2009
2010  \
Continent

Africa                             29188   28284   29890   34534
40892
Asia                              149054  133459  139894  141434
163845
Europe                             33053   33495   34692   35078
33425
Latin America and the Caribbean    24676   26011   26547   26867
28818
Northern America                    9613    9463   10190    8995
8142

                                   2011    2012    2013    Total
Continent
Africa                             35441   38083   38543   618948
```

```
Asia                                146894  152218  155075  3317794
Europe                               26778   29177   28691  1410947
Latin America and the Caribbean      27856   27173   24950   765148
Northern America                      7677    7892    8503   241142

[5 rows x 35 columns]
```

Step 2: Plot the data. We will pass in `kind = 'pie'` keyword, along with the following additional parameters:

- `autopct` - is a string or function used to label the wedges with their numeric value. The label will be placed inside the wedge. If it is a format string, the label will be `fmt%pct`.
- `startangle` - rotates the start of the pie chart by angle degrees counterclockwise from the x-axis.
- `shadow` - Draws a shadow beneath the pie (to give a 3D feel).

```
# autopct create %, start angle represent starting point
df_continents['Total'].plot(kind='pie',
                            figsize=(5, 6),
                            autopct='%1.1f%%', # add in percentages
                            startangle=90,     # start angle 90°
(Africa)
                            shadow=True,       # add shadow
                            )

plt.title('Immigration to Canada by Continent [1980 - 2013]')
plt.axis('equal') # Sets the pie chart to look like a circle.
plt.legend(labels=df_continents.index, loc='upper left')


plt.show()
```

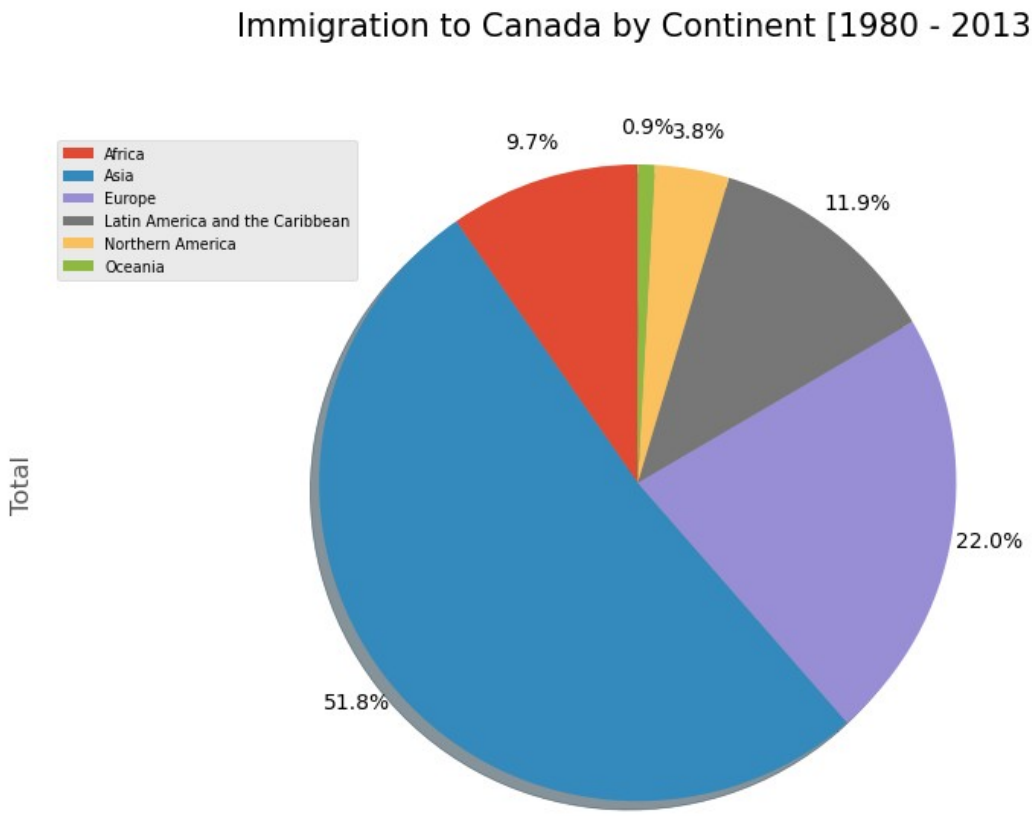


The above visual is not very clear, the numbers and text overlap in some instances. Let's make a few modifications to improve the visuals:

- Remove the text labels on the pie chart by passing in `legend` and add it as a seperate legend using `plt.legend()`.
- Push out the percentages to sit just outside the pie chart by passing in `pctdistance` parameter.
- Pass in a custom set of colors for continents by passing in `colors` parameter.
- **Explode** the pie chart to emphasize the lowest three continents (Africa, North America, and Latin America and Caribbean) by passing in `explode` parameter.

```
colors_list = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue',
'lightgreen', 'pink']
explode_list = [0.1, 0, 0, 0, 0.1, 0.1] # ratio for each continent
with which to offset each wedge.

df_continents['Total'].plot(kind='pie',
                            figsize=(10, 6),
                            autopct='%1.1f%%',
                            startangle=90,
```

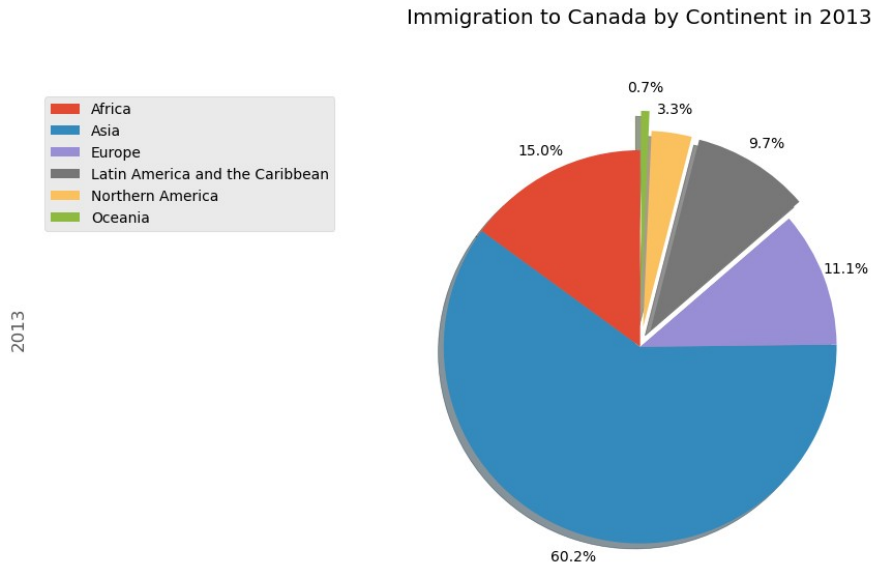
```
                            shadow=True,
                            labels=None,         # turn off labels on
pie chart
                            pctdistance=1.12,    # the ratio between
the center of each pie slice and the start of the text generated by
autopct
                            #colors=colors_list,  # add custom colors
                            #explode=explode_list # 'explode' lowest 3
continents
                            )

# scale the title up by 12% to match pctdistance
plt.title('Immigration to Canada by Continent [1980 - 2013]', y=1.12,
fontsize = 15)

plt.axis('equal')

# add legend
plt.legend(labels=df_continents.index, loc='upper left', fontsize=7)

plt.show()
```



Question: Using a pie chart, explore the proportion (percentage) of new immigrants grouped by continents in the year 2013.

Note: You might need to play with the explore values in order to fix any overlapping slice values.

```
    #The correct answer is:
    explode_list = [0.0, 0, 0, 0.1, 0.1, 0.2] # ratio for each
continent with which to offset each wedge.

    df_continents['2013'].plot(kind='pie',
                               figsize=(15, 6),
                               autopct='%1.1f%%',
                               startangle=90,
                               shadow=True,
                               labels=None,                 # turn
off labels on pie chart
                               pctdistance=1.12,            # the
ratio between the pie center and start of text label
                               explode=explode_list         #
'explode' lowest 3 continents
                               )

    # scale the title up by 12% to match pctdistance
    plt.title('Immigration to Canada by Continent in 2013', y=1.12)
    plt.axis('equal')

    # add legend
    plt.legend(labels=df_continents.index, loc='upper left')

    # show plot
    plt.show()
```



Box Plots

A `box plot` is a way of statistically representing the *distribution* of the data through five main dimensions:

- **Minimum:** The smallest number in the dataset excluding the outliers.
- **First quartile:** Middle number between the `minimum` and the `median`.
- **Second quartile (Median):** Middle number of the (sorted) dataset.
- **Third quartile:** Middle number between `median` and `maximum`.
- **Maximum:** The largest number in the dataset excluding the outliers.

To make a `boxplot`, we can use `kind=box` in `plot` method invoked on a *pandas* series or dataframe.

Let's plot the box plot for the Japanese immigrants between 1980 - 2013.

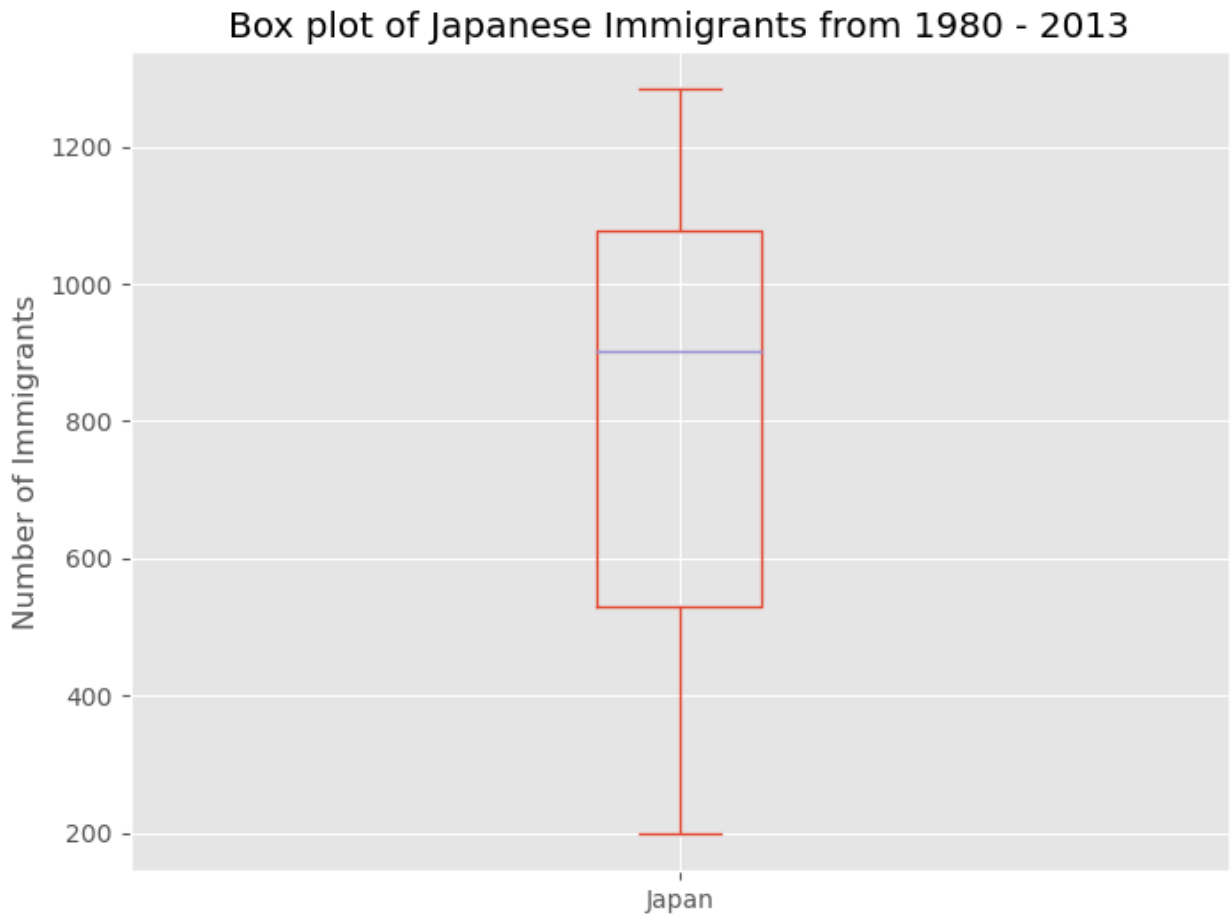
Step 1: Get the subset of the dataset. Even though we are extracting the data for just one country, we will obtain it as a dataframe. This will help us with calling the `dataframe.describe()` method to view the percentiles.

```
# to get a dataframe, place extra square brackets around 'Japan'.
df_japan = df_can.loc[['Japan'], years].transpose()
df_japan.head()
```

```
Country  Japan
1980       701
1981       756
1982       598
1983       309
1984       246
```

Step 2: Plot by passing in `kind='box'`.

```
df_japan.plot(kind='box', figsize=(8, 6))

plt.title('Box plot of Japanese Immigrants from 1980 - 2013')
plt.ylabel('Number of Immigrants')

plt.show()
```



We can immediately make a few key observations from the plot above:

1. The minimum number of immigrants is around 200 (min), maximum number is around 1300 (max), and median number of immigrants is around 900 (median).
2. 25% of the years for period 1980 - 2013 had an annual immigrant count of ~500 or fewer (First quartile).
3. 75% of the years for period 1980 - 2013 had an annual immigrant count of ~1100 or fewer (Third quartile).

We can view the actual numbers by calling the `describe()` method on the dataframe.

```
df_japan.describe()
```

```
Country        Japan
count      34.000000
mean      814.911765
std       337.219771
min       198.000000
25%       529.000000
50%       902.000000
75%      1079.000000
max      1284.000000
```

One of the key benefits of box plots is comparing the distribution of multiple datasets. In one of the previous labs, we observed that China and India had very similar immigration trends. Let's analyze these two countries further using box plots.

Question: Compare the distribution of the number of new immigrants from India and China for the period 1980 - 2013.

Step 1: Get the dataset for China and India and call the dataframe **df_CI**.

```
### type your answer here
df_CI = df_can.loc[['India', 'China'], years].transpose()
df_CI.head()
```

```
Country  India  China
1980      8880   5123
1981      8670   6682
1982      8147   3308
1983      7338   1863
1984      5704   1527
```

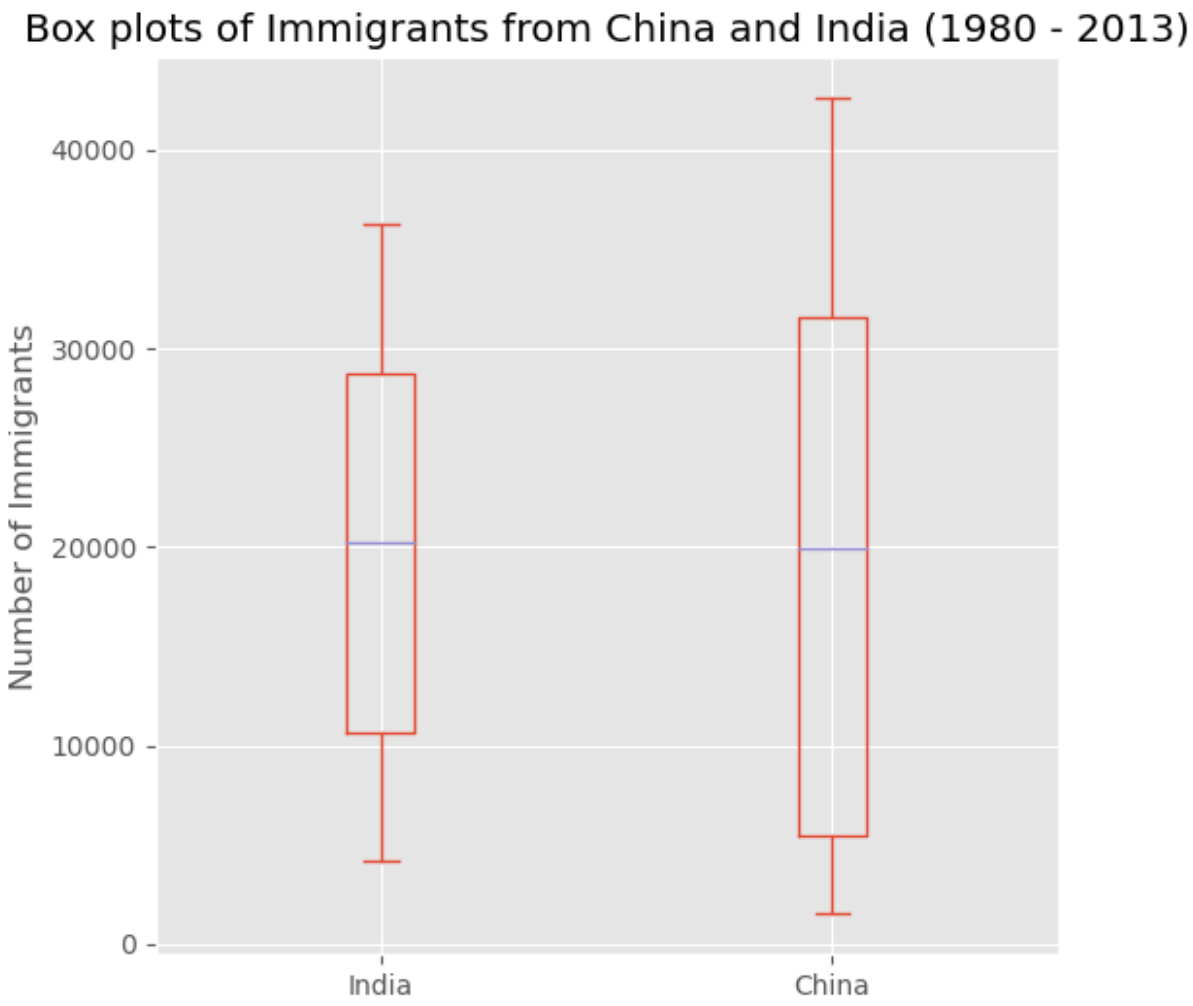
Let's view the percentiles associated with both countries using the `describe()` method.

```
### type your answer here
df_CI.describe()
```

```
Country         India         China
count       34.000000     34.000000
mean     20350.117647  19410.647059
std      10007.342579  13568.230790
min       4211.000000   1527.000000
25%      10637.750000   5512.750000
50%      20235.000000  19945.000000
75%      28699.500000  31568.500000
max      36210.000000  42584.000000
```

Step 2: Plot data.

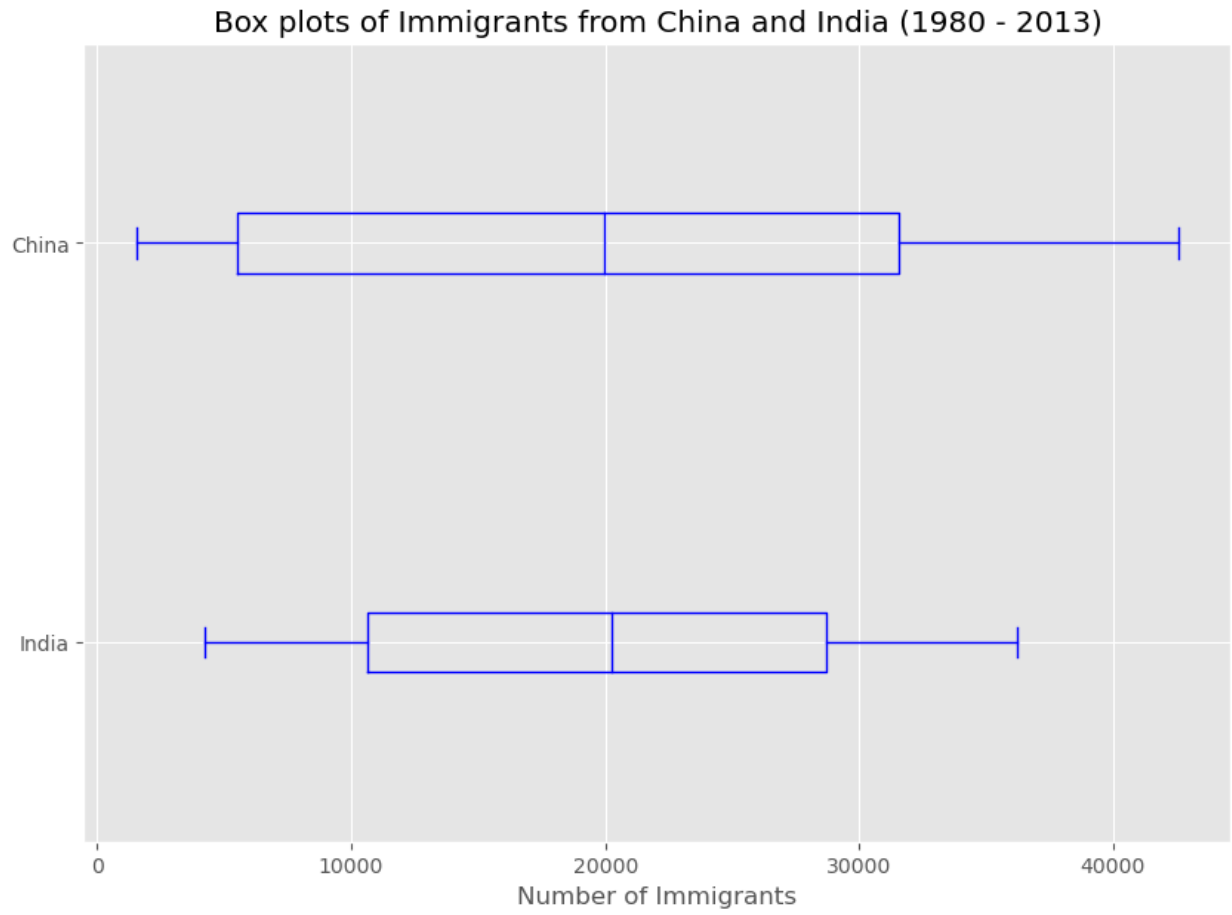
```
### type your answer here
df_CI.plot(kind='box', figsize=(6, 6))

plt.title('Box plots of Immigrants from China and India (1980 - 2013)')
plt.ylabel('Number of Immigrants')

plt.show()
```



We can observe that, while both countries have around the same median immigrant population (~20,000), China's immigrant population range is more spread out than India's. The maximum population from India for any year (36,210) is around 15% lower than the maximum population from China (42,584).

If you prefer to create horizontal box plots, you can pass the `vert` parameter in the **plot** function and assign it to *False*. You can also specify a different color in case you are not a big fan of the default red color.

```
# horizontal box plots
df_CI.plot(kind='box', figsize=(10, 7), color='blue', vert=False)

plt.title('Box plots of Immigrants from China and India (1980 - 2013)')
plt.xlabel('Number of Immigrants')

plt.show()
```



Subplots

Often times we might want to plot multiple plots within the same figure. For example, we might want to perform a side by side comparison of the box plot with the line plot of China and India's immigration.

To visualize multiple plots together, we can create a **`figure`** (overall canvas) and divide it into **`subplots`**, each containing a plot. With **subplots**, we usually work with the **artist layer** instead of the **scripting layer**.

Typical syntax is :

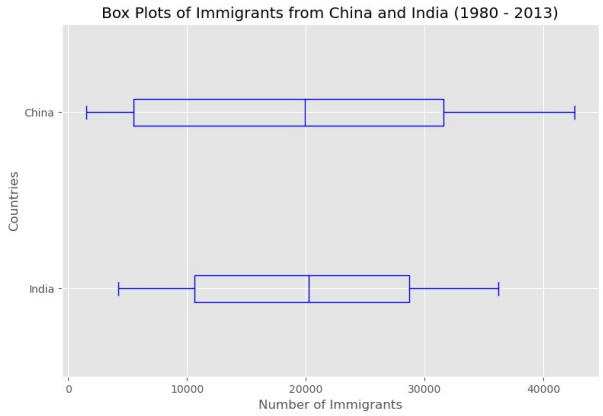
```
fig = plt.figure() # create figure
ax = fig.add_subplot(nrows, ncols, plot_number) # create subplots
```

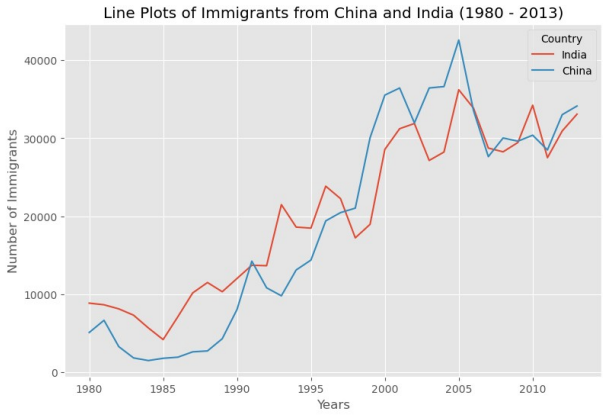
Where

- `nrows` and `ncols` are used to notionally split the figure into (`nrows` * `ncols`) sub-axes,
- `plot_number` is used to identify the particular subplot that this function is to create within the notional grid. `plot_number` starts at 1, increments across rows first and has a maximum of `nrows` * `ncols` as shown below.

We can then specify which subplot to place each plot by passing in the `ax` paramemter in `plot()` method as follows:

```
fig = plt.figure() # create figure

ax0 = fig.add_subplot(1, 2, 1) # add subplot 1 (1 row, 2 columns,
first plot)
ax1 = fig.add_subplot(1, 2, 2) # add subplot 2 (1 row, 2 columns,
second plot). See tip below**

# Subplot 1: Box plot
df_CI.plot(kind='box', color='blue', vert=False, figsize=(20, 6),
ax=ax0) # add to subplot 1
ax0.set_title('Box Plots of Immigrants from China and India (1980 -
2013)')
ax0.set_xlabel('Number of Immigrants')
ax0.set_ylabel('Countries')

# Subplot 2: Line plot
df_CI.plot(kind='line', figsize=(20, 6), ax=ax1) # add to subplot 2
ax1.set_title ('Line Plots of Immigrants from China and India (1980 -
2013)')
ax1.set_ylabel('Number of Immigrants')
ax1.set_xlabel('Years')

plt.show()
```





Tip regarding subplot convention

In the case when `nrows`, `ncols`, and `plot_number` are all less than 10, a convenience exists such that a 3-digit number can be given instead, where the hundreds represent `nrows`, the tens represent `ncols` and the units represent `plot_number`. For instance,

```
   subplot(211) == subplot(2, 1, 1)
```

produces a subaxes in a figure which represents the top plot (i.e. the first) in a 2 rows by 1 column notional grid (no grid actually exists, but conceptually this is how the returned subplot has been positioned).

Let's try something a little more advanced.

Previously we identified the top 15 countries based on total immigration from 1980 - 2013.

Question: Create a box plot to visualize the distribution of the top 15 countries (based on total immigration) grouped by the *decades* `1980s`, `1990s`, and `2000s`.

Step 1: Get the dataset. Get the top 15 countries based on Total immigrant population. Name the dataframe **df_top15**.

```
### type your answer here
df_top15 = df_can.sort_values(['Total'], ascending=False,
axis=0).head(15)
df_top15
```

```
Continent  \
Country

India
Asia
China
Asia
United Kingdom of Great Britain and Northern Ir...
Europe
Philippines
Asia
Pakistan
Asia
United States of America
Northern America
Iran (Islamic Republic of)
Asia
Sri Lanka
Asia
Republic of Korea
Asia
Poland
Europe
Lebanon
Asia
France
Europe
Jamaica                                                 Latin America and
the Caribbean
Viet Nam
Asia
```

```
Romania
Europe

                                                                  Region
\
Country

India                                                    Southern Asia

China                                                     Eastern Asia

United Kingdom of Great Britain and Northern Ir...     Northern Europe

Philippines                                           South-Eastern Asia

Pakistan                                                 Southern Asia

United States of America                               Northern America

Iran (Islamic Republic of)                               Southern Asia

Sri Lanka                                                Southern Asia

Republic of Korea                                         Eastern Asia

Poland                                                   Eastern Europe

Lebanon                                                   Western Asia

France                                                   Western Europe

Jamaica                                                       Caribbean

Viet Nam                                              South-Eastern Asia

Romania                                                  Eastern Europe


                                                                DevName
1980  \
Country

India                                                 Developing regions
8880
China                                                 Developing regions
5123
United Kingdom of Great Britain and Northern Ir...   Developed regions
22045
Philippines                                           Developing regions
6051
Pakistan                                              Developing regions
```

978
United States of America Developed regions
9378
Iran (Islamic Republic of) Developing regions
1172
Sri Lanka Developing regions
185
Republic of Korea Developing regions
1011
Poland Developed regions
863
Lebanon Developing regions
1409
France Developed regions
1729
Jamaica Developing regions
3198
Viet Nam Developing regions
1191
Romania Developed regions
375

1981 1982
1983 \
Country

India 8670 8147
7338
China 6682 3308
1863
United Kingdom of Great Britain and Northern Ir... 24796 20620
10015
Philippines 5921 5249
4562
Pakistan 972 1201
900
United States of America 10030 9074
7100
Iran (Islamic Republic of) 1429 1822
1592
Sri Lanka 371 290
197
Republic of Korea 1456 1572
1081
Poland 2930 5881
4546
Lebanon 1119 1159
789
France 2027 2219

```
1490
Jamaica                                                2634   2661
2455
Viet Nam                                               1829   2162
3404
Romania                                                 438    583
543

                                                       1984  1985  1986
...  \
Country
...
India                                                  5704  4211  7150
...
China                                                  1527  1816  1960
...
United Kingdom of Great Britain and Northern Ir...    10170  9564  9470
...
Philippines                                            3801  3150  4166
...
Pakistan                                                668   514   691
...
United States of America                               6661  6543  7074
...
Iran (Islamic Republic of)                             1977  1648  1794
...
Sri Lanka                                              1086   845  1838
...
Republic of Korea                                       847   962  1208
...
Poland                                                 3588  2819  4808
...
Lebanon                                                1253  1683  2576
...
France                                                 1169  1177  1298
...
Jamaica                                                2508  2938  4649
...
Viet Nam                                               7583  5907  2741
...
Romania                                                 524   604   656
...

                                                       2005   2006
2007  \
Country

India                                                 36210  33848
28742
China                                                 42584  33518
```

```
27642
United Kingdom of Great Britain and Northern Ir...   7258   7140
8216
Philippines                                         18139  18400
19837
Pakistan                                            14314  13127
10124
United States of America                             8394   9613
9463
Iran (Islamic Republic of)                           5837   7480
6974
Sri Lanka                                            4930   4714
4123
Republic of Korea                                    5832   6215
5920
Poland                                               1405   1263
1235
Lebanon                                              3709   3802
3467
France                                               4429   4002
4290
Jamaica                                              1945   1722
2141
Viet Nam                                             1852   3153
2574
Romania                                              5048   4468
3834

                                                     2008   2009
2010  \
Country

India                                               28261  29456
34235
China                                               30037  29622
30391
United Kingdom of Great Britain and Northern Ir...   8979   8876
8724
Philippines                                         24887  28573
38617
Pakistan                                             8994   7217
6811
United States of America                            10190   8995
8142
Iran (Islamic Republic of)                           6475   6580
7477
Sri Lanka                                            4756   4547
4422
Republic of Korea                                    7294   5874
```

```
5537
Poland                                                  1267   1013
795
Lebanon                                                 3566   3077
3432
France                                                  4532   5051
4646
Jamaica                                                 2334   2456
2321
Viet Nam                                                1784   2171
1942
Romania                                                 2837   2076
1922

                                                        2011   2012
2013  \
Country

India                                                  27509  30933
33087
China                                                  28502  33024
34129
United Kingdom of Great Britain and Northern Ir...      6204   6195
5827
Philippines                                            36765  34315
29544
Pakistan                                                7468  11227
12603
United States of America                                7676   7891
8501
Iran (Islamic Republic of)                              7479   7534
11291
Sri Lanka                                               3309   3338
2394
Republic of Korea                                       4588   5316
4509
Poland                                                   720    779
852
Lebanon                                                 3072   1614
2172
France                                                  4080   6280
5623
Jamaica                                                 2059   2182
2479
Viet Nam                                                1723   1731
2112
Romania                                                 1776   1588
1512

                                                        Total
```

```
Country
India                                                691904
China                                                659962
United Kingdom of Great Britain and Northern Ir...   551500
Philippines                                          511391
Pakistan                                             241600
United States of America                             241122
Iran (Islamic Republic of)                           175923
Sri Lanka                                            148358
Republic of Korea                                    142581
Poland                                               139241
Lebanon                                              115359
France                                               109091
Jamaica                                              106431
Viet Nam                                              97146
Romania                                               93585

[15 rows x 38 columns]
```

Step 2: Create a new dataframe which contains the aggregate for each decade. One way to do that:

1. Create a list of all years in decades 80's, 90's, and 00's.
2. Slice the original dataframe df_can to create a series for each decade and sum across all years for each country.
3. Merge the three series into a new data frame. Call your dataframe **new_df**.

```
### type your answer here
#1:
years_80s = list(map(str, range(1980, 1990)))
years_90s = list(map(str, range(1990, 2000)))
years_00s = list(map(str, range(2000, 2010)))
print(years_80s)
print(years_90s)
print(years_00s)

#2:
df_80s = df_top15.loc[:, years_80s].sum(axis=1)
df_90s = df_top15.loc[:, years_90s].sum(axis=1)
df_00s = df_top15.loc[:, years_00s].sum(axis=1)

new_df = pd.DataFrame({'1980s': df_80s, '1990s': df_90s,
'2000s':df_00s})

new_df.head()
```

```
['1980', '1981', '1982', '1983', '1984', '1985', '1986', '1987',
'1988', '1989']
['1990', '1991', '1992', '1993', '1994', '1995', '1996', '1997',
'1998', '1999']
```

```
['2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007',
'2008', '2009']
```

```
                                                    1980s   1990s
2000s
Country

India                                                82154  180395
303591
China                                                32003  161528
340385
United Kingdom of Great Britain and Northern Ir...  179171  261966
83413
Philippines                                          60764  138482
172904
Pakistan                                             10591   65302
127598
```

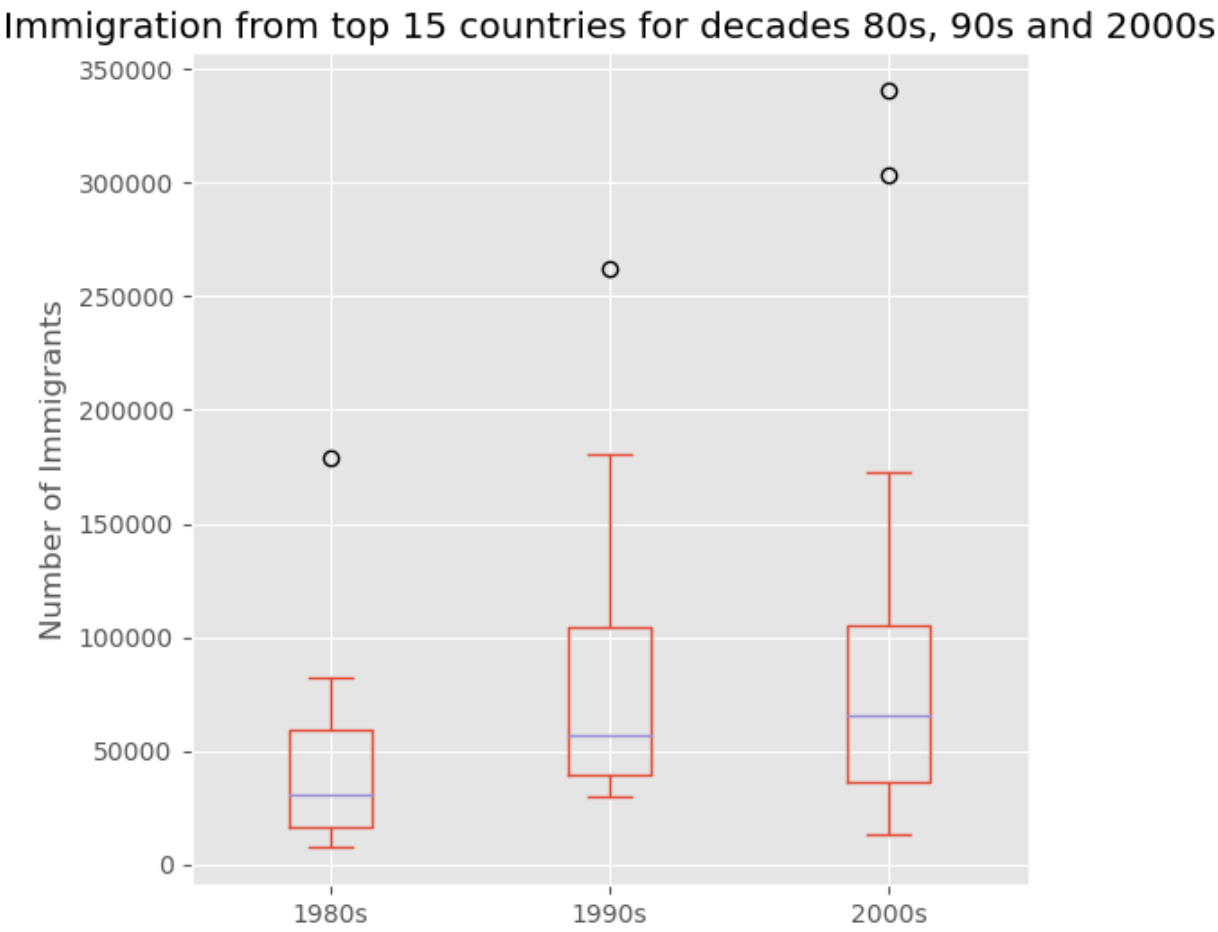
Let's learn more about the statistics associated with the dataframe using the `describe()` method.

```
### type your answer here
new_df.describe()
```

```
                1980s          1990s          2000s
count       15.000000      15.000000      15.000000
mean     44418.333333   85594.666667   97471.533333
std      44190.676455   68237.560246  100583.204205
min       7613.000000   30028.000000   13629.000000
25%      16698.000000   39259.000000   36101.500000
50%      30638.000000   56915.000000   65794.000000
75%      59183.000000  104451.500000  105505.500000
max     179171.000000  261966.000000  340385.000000
```

Step 3: Plot the box plots.

```
### type your answer here
new_df.plot(kind='box', figsize=(6, 6))

plt.title('Immigration from top 15 countries for decades 80s, 90s and
2000s')
plt.ylabel('Number of Immigrants')

plt.show()
```



Note how the box plot differs from the summary table created. The box plot scans the data and identifies the outliers. In order to be an outlier, the data value must be:

- larger than Q3 by at least 1.5 times the interquartile range (IQR), or,
- smaller than Q1 by at least 1.5 times the IQR.

Let's look at decade 2000s as an example:

- Q1 (25%) = 36,101.5
- Q3 (75%) = 105,505.5
- IQR = Q3 - Q1 = 69,404

Using the definition of outlier, any value that is greater than Q3 by 1.5 times IQR will be flagged as outlier.

Outlier > 105,505.5 + (1.5 * 69,404) Outlier > 209,611.5

```
# let's check how many entries fall above the outlier threshold
new_df = new_df.reset_index()
new_df[new_df['2000s'] > 209611.5]
```

```
  Country  1980s   1990s   2000s
0   India  82154  180395  303591
1   China  32003  161528  340385
```

China and India are both considered as outliers since their population for the decade exceeds 209,611.5.

The box plot is an advanced visualizaiton tool, and there are many options and customizations that exceed the scope of this lab. Please refer to Matplotlib documentation on box plots for more information.

Scatter Plots

A `scatter plot` (2D) is a useful method of comparing variables against each other. `Scatter` plots look similar to `line plots` in that they both map independent and dependent variables on a 2D graph. While the data points are connected together by a line in a line plot, they are not connected in a scatter plot. The data in a scatter plot is considered to express a trend. With further analysis using tools like regression, we can mathematically calculate this relationship and use it to predict trends outside the dataset.

Let's start by exploring the following:

Using a `scatter plot`, let's visualize the trend of total immigrantion to Canada (all countries combined) for the years 1980 - 2013.

Step 1: Get the dataset. Since we are expecting to use the relationship betewen `years` and `total population`, we will convert `years` to `int` type.

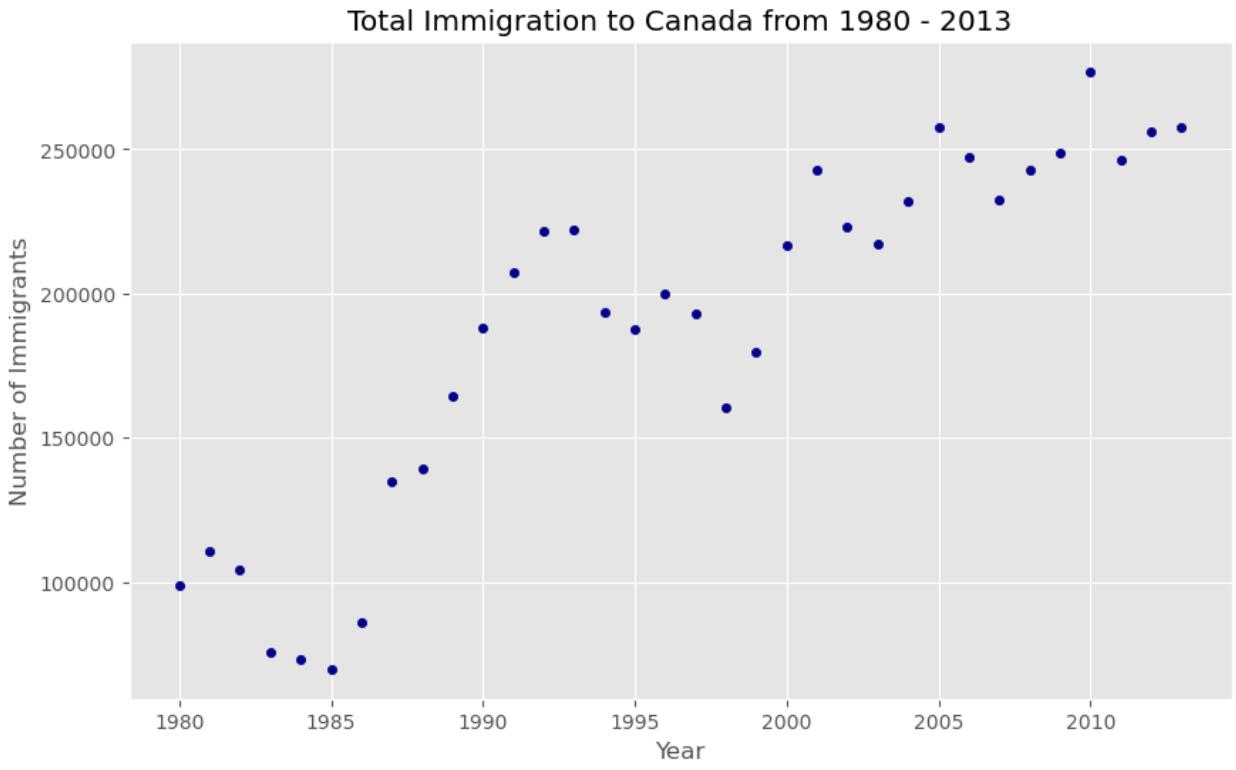
```
# we can use the sum() method to get the total population per year
df_tot = pd.DataFrame(df_can[years].sum(axis=0))

# change the years to type int (useful for regression later on)
df_tot.index = map(int, df_tot.index)

# reset the index to put in back in as a column in the df_tot
dataframe
df_tot.reset_index(inplace = True)

# rename columns
df_tot.columns = ['year', 'total']

# view the final dataframe
df_tot.head()
```

```
   year   total
0  1980   99137
1  1981  110563
2  1982  104271
```

```
3  1983   75550
4  1984   73417
```

Step 2: Plot the data. In `Matplotlib`, we can create a `scatter` plot set by passing in `kind='scatter'` as plot argument. We will also need to pass in `x` and `y` keywords to specify the columns that go on the x- and the y-axis.

```
df_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6),
color='darkblue')

plt.title('Total Immigration to Canada from 1980 - 2013')
plt.xlabel('Year')
plt.ylabel('Number of Immigrants')

plt.show()
```



Notice how the scatter plot does not connect the data points together. We can clearly observe an upward trend in the data: as the years go by, the total number of immigrants increases. We can mathematically analyze this upward trend using a regression line (line of best fit).

So let's try to plot a linear line of best fit, and use it to predict the number of immigrants in 2015.

Step 1: Get the equation of line of best fit. We will use **Numpy**'s `polyfit()` method by passing in the following:

- `x`: x-coordinates of the data.

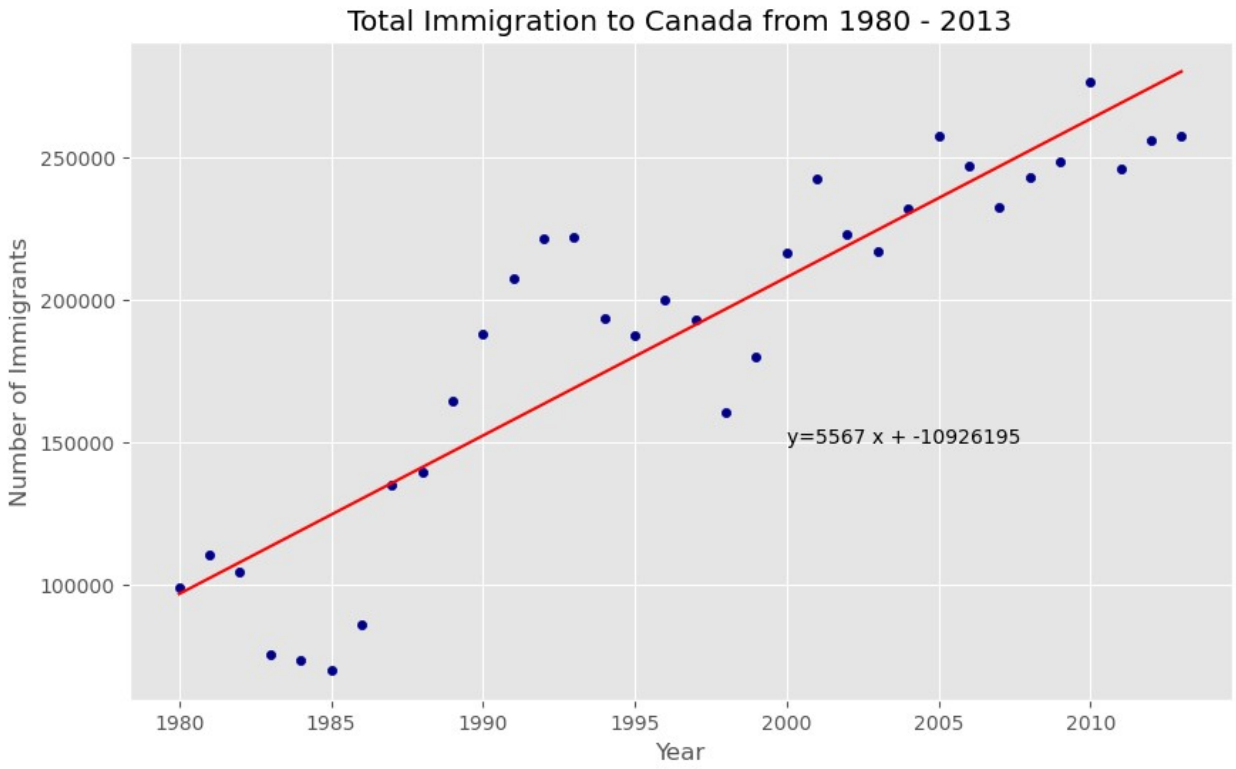
- `y`: y-coordinates of the data.
- `deg`: Degree of fitting polynomial. 1 = linear, 2 = quadratic, and so on.

```
x = df_tot['year']      # year on x-axis
y = df_tot['total']     # total on y-axis
fit = np.polyfit(x, y, deg=1)

fit

array([ 5.56709228e+03, -1.09261952e+07])
```

The output is an array with the polynomial coefficients, highest powers first. Since we are plotting a linear regression `y= a * x + b`, our output has 2 elements `[5.56709228e+03, -1.09261952e+07]` with the the slope in position 0 and intercept in position 1.

Step 2: Plot the regression line on the `scatter plot`.

```
df_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6),
color='darkblue')

plt.title('Total Immigration to Canada from 1980 - 2013')
plt.xlabel('Year')
plt.ylabel('Number of Immigrants')

# plot line of best fit
plt.plot(x, fit[0] * x + fit[1], color='red') # recall that x is the
Years
plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit[0], fit[1]), xy=(2000,
150000))

plt.show()

# print out the line of best fit
'No. Immigrants = {0:.0f} * Year + {1:.0f}'.format(fit[0], fit[1])
```




```
'No. Immigrants = 5567 * Year + -10926195'
```

Using the equation of line of best fit, we can estimate the number of immigrants in 2015:

```
No. Immigrants = 5567 * Year - 10926195
No. Immigrants = 5567 * 2015 - 10926195
No. Immigrants = 291,310
```

When compared to the actual from Citizenship and Immigration Canada's (CIC) 2016 Annual Report, we see that Canada accepted 271,845 immigrants in 2015. Our estimated value of 291,310 is within 7% of the actual number, which is pretty good considering our original data came from United Nations (and might differ slightly from CIC data).

As a side note, we can observe that immigration took a dip around 1993 - 1997. Further analysis into the topic revealed that in 1993 Canada introcuded Bill C-86 which introduced revisions to the refugee determination system, mostly restrictive. Further amendments to the Immigration Regulations cancelled the sponsorship required for "assisted relatives" and reduced the points awarded to them, making it more difficult for family members (other than nuclear family) to immigrate to Canada. These restrictive measures had a direct impact on the immigration numbers for the next several years.

Question: Create a scatter plot of the total immigration from Denmark, Norway, and Sweden to Canada from 1980 to 2013?

Step 1: Get the data:

1. Create a dataframe the consists of the numbers associated with Denmark, Norway, and Sweden only. Name it **df_countries**.
2. Sum the immigration numbers across all three countries for each year and turn the result into a dataframe. Name this new dataframe **df_total**.
3. Reset the index in place.
4. Rename the columns to **year** and **total**.
5. Display the resulting dataframe.

```
# 1
df_can.loc[['Denmark', 'Norway', 'Sweden'], years]
df_countries = df_can.loc[['Denmark', 'Norway', 'Sweden'],
years].transpose()
df_countries.head(5)
```

```
Country  Denmark  Norway  Sweden
1980         272     116     281
1981         293      77     308
1982         299     106     222
1983         106      51     176
1984          93      31     128
```

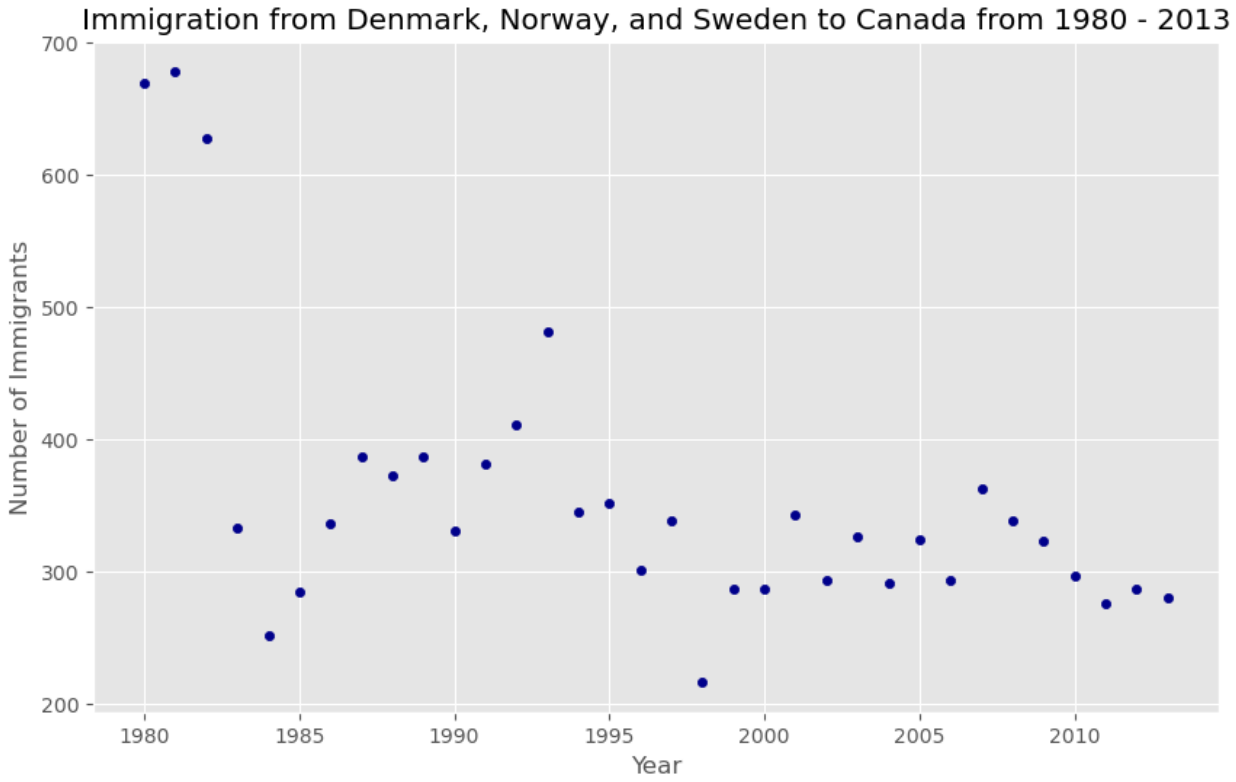
```
df_countries = df_can.loc[['Denmark', 'Norway', 'Sweden'],
years].transpose()

df_total = pd.DataFrame(df_countries.sum(axis=1))

df_total.reset_index(inplace=True)

df_total.columns = ['year', 'total']

df_total['year'] = df_total['year'].astype(int)

df_total.head()
```

```
   year  total
0  1980    669
1  1981    678
2  1982    627
3  1983    333
4  1984    252
```

Step 2: Generate the scatter plot by plotting the total versus year in **df_total**.

```
### type your answer here

df_total.plot(kind='scatter', x='year', y='total', figsize=(10, 6),
color='darkblue')

plt.title('Immigration from Denmark, Norway, and Sweden to Canada from
1980 - 2013')
```

```
plt.xlabel('Year')
plt.ylabel('Number of Immigrants')

plt.show()
```



Bubble Plots

A `bubble plot` is a variation of the `scatter plot` that displays three dimensions of data (x, y, z). The data points are replaced with bubbles, and the size of the bubble is determined by the third variable `z`, also known as the weight. In `maplotlib`, we can pass in an array or scalar to the parameter `s` to `plot()`, that contains the weight of each point.

Let's start by analyzing the effect of Argentina's great depression.

Argentina suffered a great depression from 1998 to 2002, which caused widespread unemployment, riots, the fall of the government, and a default on the country's foreign debt. In terms of income, over 50% of Argentines were poor, and seven out of ten Argentine children were poor at the depth of the crisis in 2002.

Let's analyze the effect of this crisis, and compare Argentina's immigration to that of it's neighbour Brazil. Let's do that using a `bubble plot` of immigration from Brazil and Argentina for the years 1980 - 2013. We will set the weights for the bubble as the *normalized* value of the population for each year.

Step 1: Get the data for Brazil and Argentina. Like in the previous example, we will convert the `Years` to type int and include it in the dataframe.

```
# transposed dataframe
df_can_t = df_can[years].transpose()

# cast the Years (the index) to type int
df_can_t.index = map(int, df_can_t.index)

# let's label the index. This will automatically be the column name
when we reset the index
df_can_t.index.name = 'Year'

# reset index to bring the Year in as a column
df_can_t.reset_index(inplace=True)

# view the changes
df_can_t.head()
```

```
Country  Year  Afghanistan  Albania  Algeria  American Samoa  Andorra
Angola  \
0        1980           16        1       80               0        0
1
1        1981           39        0       67               1        0
3
2        1982           39        0       71               0        0
6
3        1983           47        0       69               0        0
6
4        1984           71        0       63               0        0
4

Country  Antigua and Barbuda  Argentina  Armenia  ...  \
0                          0        368        0  ...
1                          0        426        0  ...
2                          0        626        0  ...
3                          0        241        0  ...
4                         42        237        0  ...

Country  United States of America  Uruguay  Uzbekistan  Vanuatu  \
0                            9378      128           0        0
1                           10030      132           0        0
2                            9074      146           0        0
3                            7100      105           0        0
4                            6661       90           0        0

Country  Venezuela (Bolivarian Republic of)  Viet Nam  Western Sahara
Yemen  \
0                                      103      1191               0
1
```

```
1                                       117      1829                0
2
2                                       174      2162                0
1
3                                       124      3404                0
6
4                                       142      7583                0
0

Country  Zambia  Zimbabwe
0            11        72
1            17       114
2            11       102
3             7        44
4            16        32

[5 rows x 196 columns]
```

Step 2: Create the normalized weights.

There are several methods of normalizations in statistics, each with its own use. In this case, we will use feature scaling to bring all values into the range [0, 1]. The general formula is:

where X is the original value, X' is the corresponding normalized value. The formula sets the max value in the dataset to 1, and sets the min value to 0. The rest of the data points are scaled to a value between 0-1 accordingly.

```
# normalize Brazil data
norm_brazil = (df_can_t['Brazil'] - df_can_t['Brazil'].min()) /
(df_can_t['Brazil'].max() - df_can_t['Brazil'].min())

# normalize Argentina data
norm_argentina = (df_can_t['Argentina'] - df_can_t['Argentina'].min())
/ (df_can_t['Argentina'].max() - df_can_t['Argentina'].min())
```

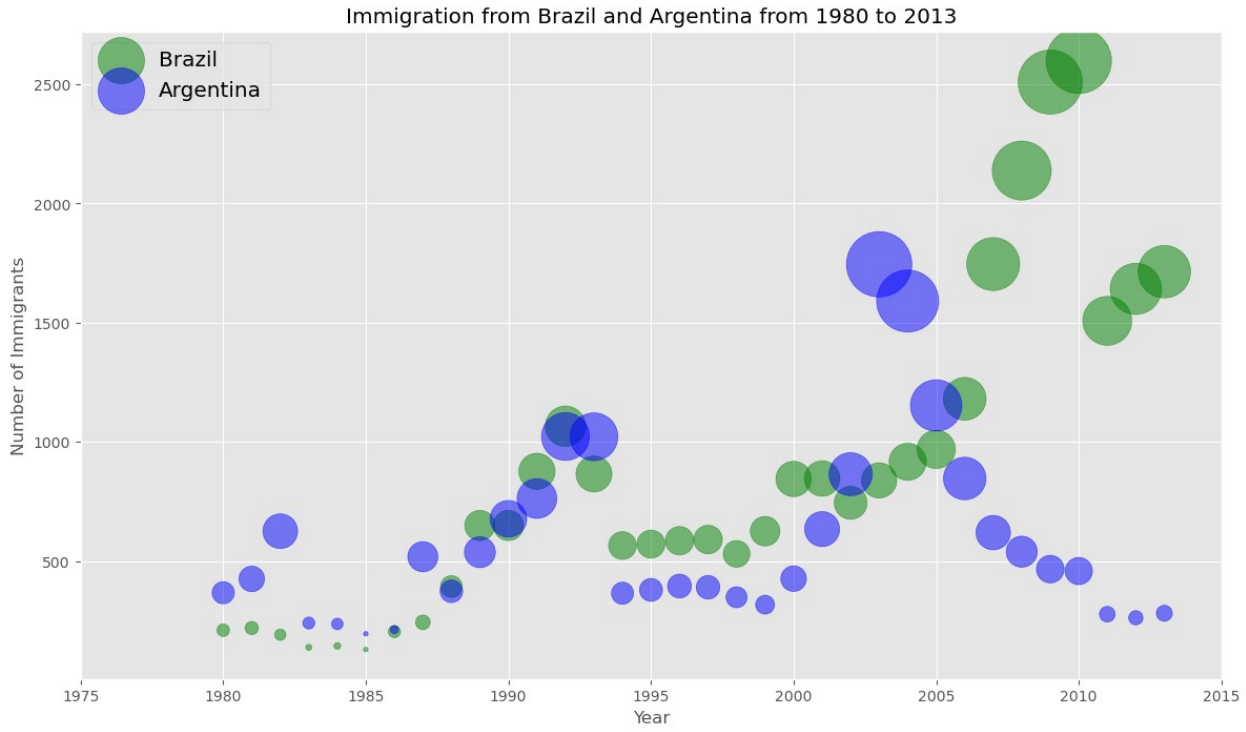
Step 3: Plot the data.

- To plot two different scatter plots in one plot, we can include the axes one plot into the other by passing it via the `ax` parameter.
- We will also pass in the weights using the `s` parameter. Given that the normalized weights are between 0-1, they won't be visible on the plot. Therefore, we will:
  - multiply weights by 2000 to scale it up on the graph, and,
  - add 10 to compensate for the min value (which has a 0 weight and therefore scale with $\times 2000$).

```
# Brazil
ax0 = df_can_t.plot(kind='scatter',
                    x='Year',
```

```
                      y='Brazil',
                      figsize=(14, 8),
                      alpha=0.5,  # transparency
                      color='green',
                      s=norm_brazil * 2000 + 10,  # pass in weights
                      xlim=(1975, 2015)
                      )

# Argentina
ax1 = df_can_t.plot(kind='scatter',
                      x='Year',
                      y='Argentina',
                      alpha=0.5,
                      color="blue",
                      s=norm_argentina * 2000 + 10,
                      ax=ax0
                      )

ax0.set_ylabel('Number of Immigrants')
ax0.set_title('Immigration from Brazil and Argentina from 1980 to
2013')
ax0.legend(['Brazil', 'Argentina'], loc='upper left', fontsize='x-
large')
```

```
<matplotlib.legend.Legend at 0x7f5fa8dac750>
```



The size of the bubble corresponds to the magnitude of immigrating population for that year, compared to the 1980 - 2013 data. The larger the bubble is, the more immigrants are in that year.

From the plot above, we can see a corresponding increase in immigration from Argentina during the 1998 - 2002 great depression. We can also observe a similar spike around 1985 to 1993. In fact, Argentina had suffered a great depression from 1974 to 1990, just before the onset of 1998 - 2002 great depression.

On a similar note, Brazil suffered the *Samba Effect* where the Brazilian real (currency) dropped nearly 35% in 1999. There was a fear of a South American financial crisis as many South American countries were heavily dependent on industrial exports from Brazil. The Brazilian government subsequently adopted an austerity program, and the economy slowly recovered over the years, culminating in a surge in 2010. The immigration data reflect these events.

Question: Previously in this lab, we created box plots to compare immigration from China and India to Canada. Create bubble plots of immigration from China and India to visualize any differences with time from 1980 to 2013. You can use **df_can_t** that we defined and used in the previous example.

Step 1: Normalize the data pertaining to China and India.

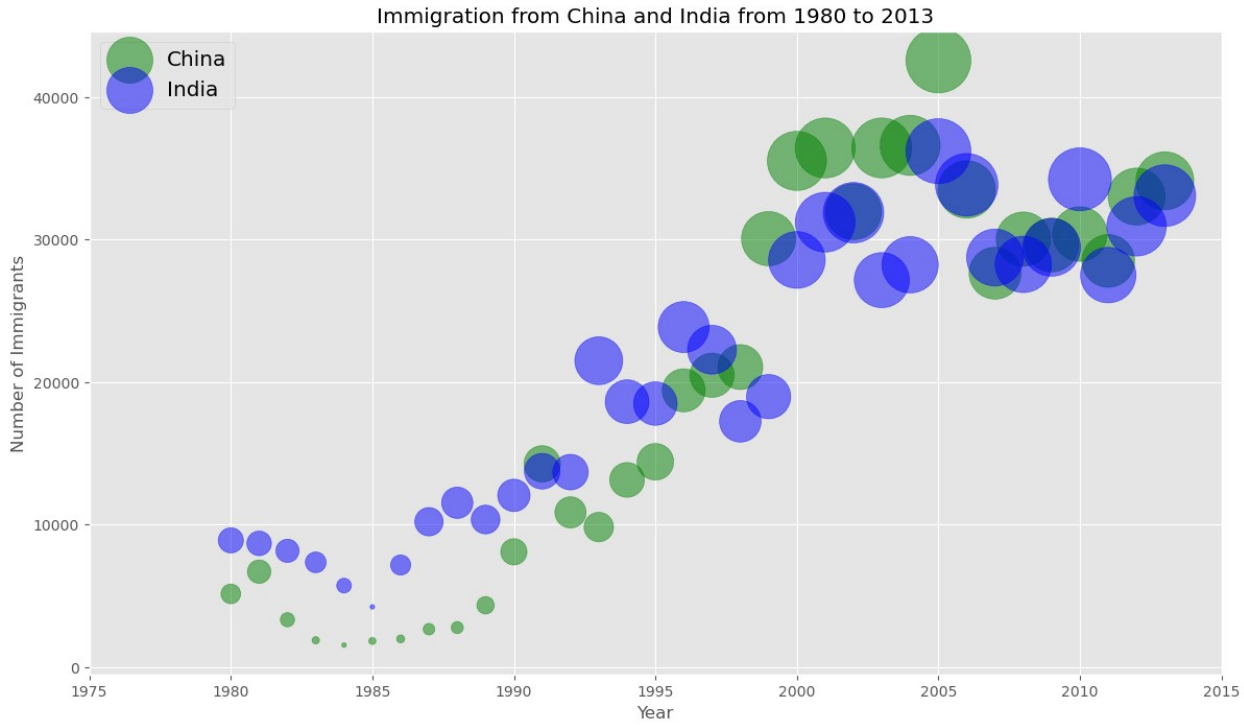
```
### type your answer here
# normalize China data
norm_China = (df_can_t['China'] - df_can_t['China'].min()) /
(df_can_t['China'].max() - df_can_t['China'].min())

# normalize Indian data
norm_India = (df_can_t['India'] - df_can_t['India'].min()) /
(df_can_t['India'].max() - df_can_t['India'].min())
```

Step 2: Generate the bubble plots.

```
### type your answer here

# China
ax0 = df_can_t.plot(kind='scatter',
                    x='Year',
                    y='China',
                    figsize=(14, 8),
                    alpha=0.5,  # transparency
                    color='green',
                    s=norm_China * 2000 + 10,  # pass in weights
                    xlim=(1975, 2015)
                    )

# India
ax1 = df_can_t.plot(kind='scatter',
                    x='Year',
                    y='India',
```

```
                    alpha=0.5,
                    color="blue",
                    s=norm_India * 2000 + 10,
                    ax=ax0
                    )

ax0.set_ylabel('Number of Immigrants')
ax0.set_title('Immigration from China and India from 1980 to 2013')
ax0.legend(['China', 'India'], loc='upper left', fontsize='x-large')
```

```
<matplotlib.legend.Legend at 0x7f5fa8c24bd0>
```



Thank you for completing this lab!

Author

Alex Aklson

Other Contributors

Jay Rajasekharan, Ehsan M. Kermani, Slobodan Markovic, Weiqing Wang, Pooja.

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-06-11	2.7	Pooja	Updated the file to work with clean data

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-05-29	2.6	Weiqing Wang	Fixed typos and code spells.
2021-01-20	2.5	LakshmiHolla	Changed TOC markdown section
2021-01-05	2.4	LakshmiHolla	Changed markdown for outliers
2020-11-12	2.3	LakshmiHolla	Added example code for outliers
2020-11-03	2.2	LakshmiHolla	Changed URL of excel file
2020-09-29	2.1	LakshmiHolla	Made fix to a boxplot label
2020-08-27	2.0	Lavanya	Moved lab to course repo in GitLab