```
In [10]: !mamba install openpyxl==3.0.9 -y
```

```
zsh:1: command not found: mamba
```

```
In [2]: import numpy as np  # useful for many scientific computing in Python
        import pandas as pd # primary data structure library
```

```
In [3]: df_can = pd.read_excel(
            'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-Skil
            sheet_name='Canada by Citizenship',
            skiprows=range(20),
            skipfooter=2)
        print('Data downloaded and read into a dataframe!')
```

```
Data downloaded and read into a dataframe!
```

```
In [4]: df_can.head()
```

Out[4]:

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980	...	2004	2005	2006	2007	200
0	Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	16	...	2978	3436	3009	2652	211
1	Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	1	...	1450	1223	856	702	56
2	Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	80	...	3616	3626	4807	3623	400
3	Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	0	...	0	0	1	0	
4	Immigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions	0	...	0	0	1	1	

5 rows × 43 columns

```
In [5]: print(df_can.shape)
```

```
(195, 43)
```

```
In [6]: df_can.drop(['AREA', 'REG', 'DEV', 'Type', 'Coverage'], axis=1, inplace=True)
        df_can.head()
```

Out[6]:

	OdName	AreaName	RegName	DevName	1980	1981	1982	1983	1984	1985	...	2004	2005	2006	2007	2008	2009
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...	2978	3436	3009	2652	2111	1746
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...	1450	1223	856	702	560	716
2	Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	...	3616	3626	4807	3623	4005	5393
3	American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	...	0	0	1	0	0	0
4	Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	...	0	0	1	1	0	0

5 rows × 38 columns

```
In [7]: df_can.rename(columns={'OdName':'Country', 'AreaName':'Continent','RegName':'Region'}, inplace=True)
        df_can.head()
```

Out[7]:

	Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	...	2004	2005	2006	2007	2008	2009
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...	2978	3436	3009	2652	2111	1746
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...	1450	1223	856	702	560	716
2	Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	...	3616	3626	4807	3623	4005	5393
3	American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	...	0	0	1	0	0	0
4	Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	...	0	0	1	1	0	0

5 rows × 38 columns

```
In [8]: #data type of the column labels
        all(isinstance(column, str) for column in df_can.columns)
```

```
Out[8]: False
```

```
In [9]: #Not all column data is string, need to change all to string
        df_can.columns = list(map(str, df_can.columns))

        all(isinstance(column, str) for column in df_can.columns)
```

```
Out[9]: True
```

```
In [11]: #Setting country name as index for easy use of .loc method
         df_can.set_index('Country', inplace=True)

         df_can.head()
```

Out[11]:

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2004	2005	2006	2007	2008	200
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...	2978	3436	3009	2652	2111	174
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...	1450	1223	856	702	560	7
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...	3616	3626	4807	3623	4005	539
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	...	0	0	1	0	0	
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2	...	0	0	1	1	0	

5 rows × 37 columns

```
In [13]: #Adding a "Total" column which adds up all immigration for each country from 1980 to 2013
         df_can['Total'] = df_can.sum(axis=1)

         df_can.head()
```

```
/var/folders/cn/0zbpg_4n7556t3y1p2lb9w1w0000gn/T/ipykernel_15555/1240774001.py:2: FutureWarning: Dropping of nu
isance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will
raise TypeError.  Select only valid columns before calling the reduction.
  df_can['Total'] = df_can.sum(axis=1)
```

Out[13]:

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	201
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...	3436	3009	2652	2111	1746	175
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...	1223	856	702	560	716	56
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...	3626	4807	3623	4005	5393	475
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	0	...	0	1	0	0	0	
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	2	...	0	1	1	0	0	

5 rows × 38 columns

```
In [14]: print(df_can.shape)
```

```
(195, 38)
```

```
In [15]: #List of years from 1980 to 2013
         years = list(map(str, range(1980, 2014)))

         years
```

```
Out[15]: ['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']
```

Visualizing the data

```
In [16]: %matplotlib inline

         import matplotlib as mpl
         import matplotlib.pyplot as plt

         mpl.style.use('ggplot')

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

```
Matplotlib version:  3.4.3
```

Area Plot

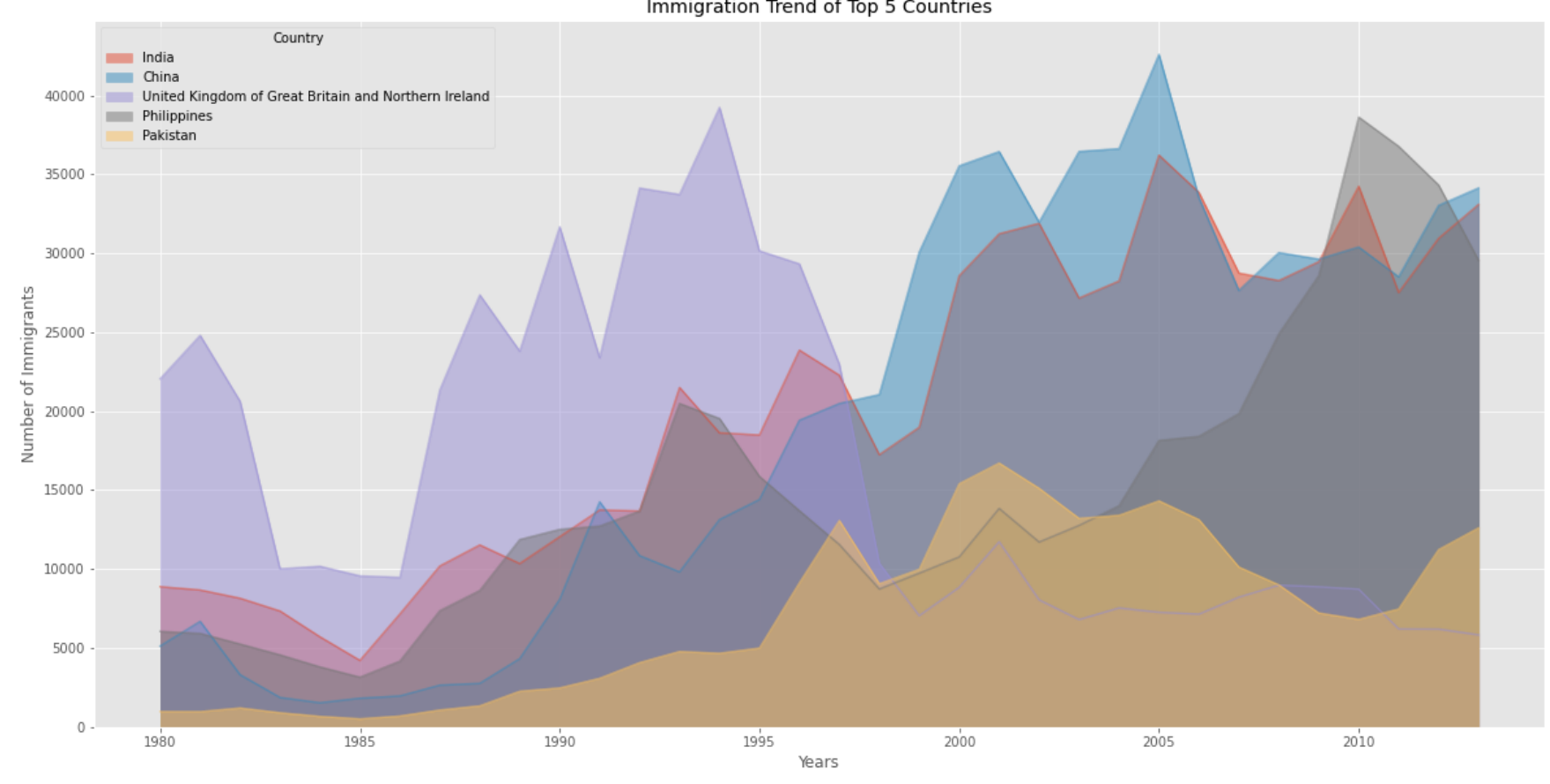
```
In [17]: df_can.sort_values(['Total'], ascending=False, axis=0, inplace=True)

         df_top5 = df_can.head()

         # transpose the dataframe
         df_top5 = df_top5[years].transpose()

         df_top5.head()
```

Out[17]:

Country	India	China	United Kingdom of Great Britain and Northern Ireland	Philippines	Pakistan
1980	8880	5123	22045	6051	978
1981	8670	6682	24796	5921	972
1982	8147	3308	20620	5249	1201
1983	7338	1863	10015	4562	900
1984	5704	1527	10170	3801	668

```
In [18]: # changing the index values of df_top5 to integer type for plotting
         df_top5.index = df_top5.index.map(int)
         df_top5.plot(kind='area',
                      stacked=False,
                      figsize=(20, 10))  # pass a tuple (x, y) size

         plt.title('Immigration Trend of Top 5 Countries')
         plt.ylabel('Number of Immigrants')
         plt.xlabel('Years')

         plt.show()
```


```
In [19]: df_top5.plot(kind='area',
                      alpha=0.25,  # 0 - 1, default value alpha = 0.5
                      stacked=False,
                      figsize=(20, 10))

         plt.title('Immigration Trend of Top 5 Countries')
         plt.ylabel('Number of Immigrants')
         plt.xlabel('Years')

         plt.show()
```

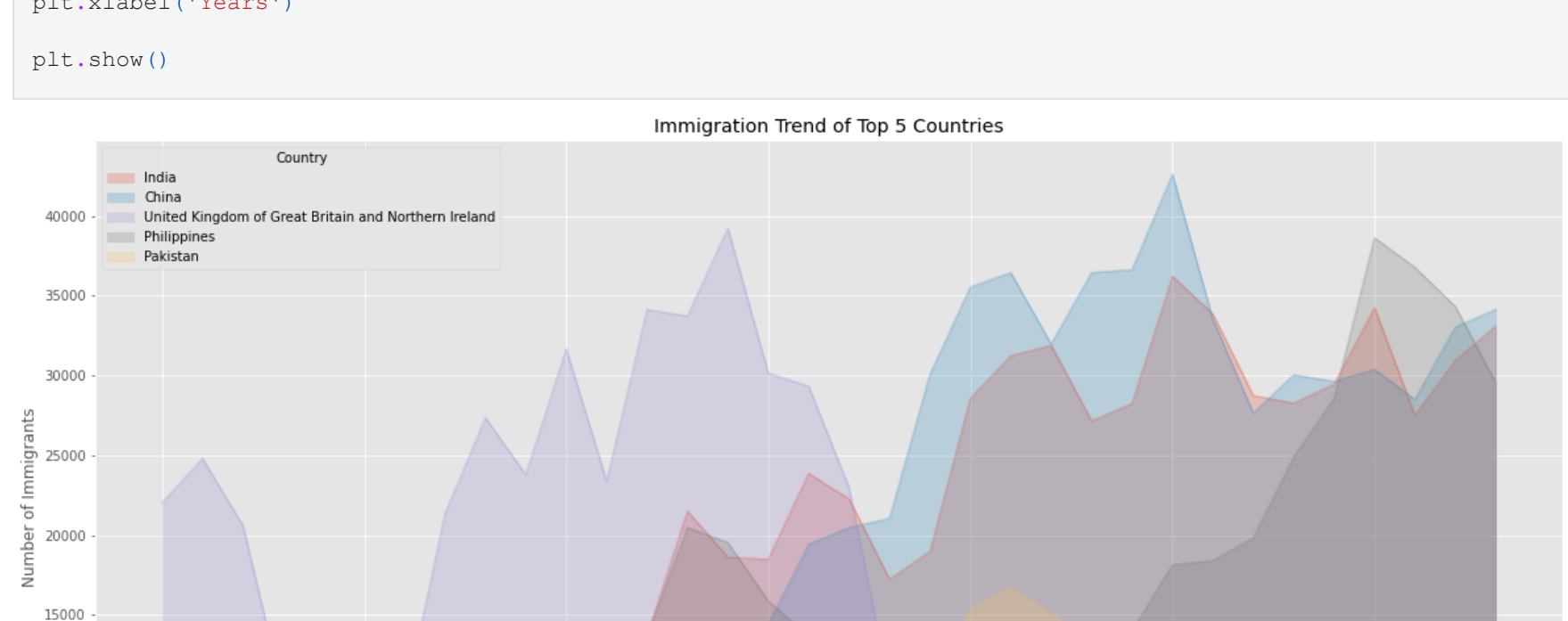



```
In [20]: df_bottom5 = df_can.tail()

         # transpose the dataframe
         df_bottom5 = df_bottom5[years].transpose()

         df_bottom5.tail()
```

Out[20]:

Country	San Marino	New Caledonia	Marshall Islands	Western Sahara	Palau
2009	0	0	0	0	0
2010	1	0	0	0	0
2011	0	0	0	0	0
2012	0	0	0	0	0
2013	0	2	0	0	0

```
In [23]: df_bottom5.index = df_bottom5.index.map(int)
         df_bottom5.plot(kind='area',
                      stacked=False,
                      alpha=0.25,
                      figsize=(20, 10))

         plt.title('Immigration Trend of Bottom 5 Countries')
         plt.ylabel('Number of Immigrants')
         plt.xlabel('Years')

         plt.show()
```



Histograms

```
In [24]: # Creating histogram using immigration to Canada in 2013
         df_can['2013'].head()
```

```
Out[24]: Country
India                                                   33087
China                                                   34129
United Kingdom of Great Britain and Northern Ireland     5827
Philippines                                             29544
Pakistan                                                12603
Name: 2013, dtype: int64
```

```
In [25]: # np.histogram returns 2 values
         count, bin_edges = np.histogram(df_can['2013'])

         print(count) # frequency count
         print(bin_edges) # bin ranges, default = 10 bins
```

```
[178  11   1   2   0   0   0   0   1   2]
[    0.    3412.9   6825.8 10238.7 13651.6 17064.5 20477.4 23890.3 27303.2
 30716.1 34129. ]
```

```
In [26]: df_can['2013'].plot(kind='hist', figsize=(8, 5))

         plt.title('Histogram of Immigration from 195 Countries in 2013')
         plt.ylabel('Number of Countries')
         plt.xlabel('Number of Immigrants')

         plt.show()
```


```
In [27]: # 'using xticks to match the actual bin size for x-label
         count, bin_edges = np.histogram(df_can['2013'])

         df_can['2013'].plot(kind='hist', figsize=(8, 5), xticks=bin_edges)

         #add graph title, x- and y- axes label
         plt.title('Histogram of Immigration from 195 countries in 2013')
         plt.ylabel('Number of Countries')
         plt.xlabel('Number of Immigrants')

         plt.show()
```


```
In [28]: # Multiple histograms on the same plot. Example immigration distribution for Denmark, Norway, and Sweden for ye
         df_can.loc[['Denmark', 'Norway', 'Sweden'], years]
```

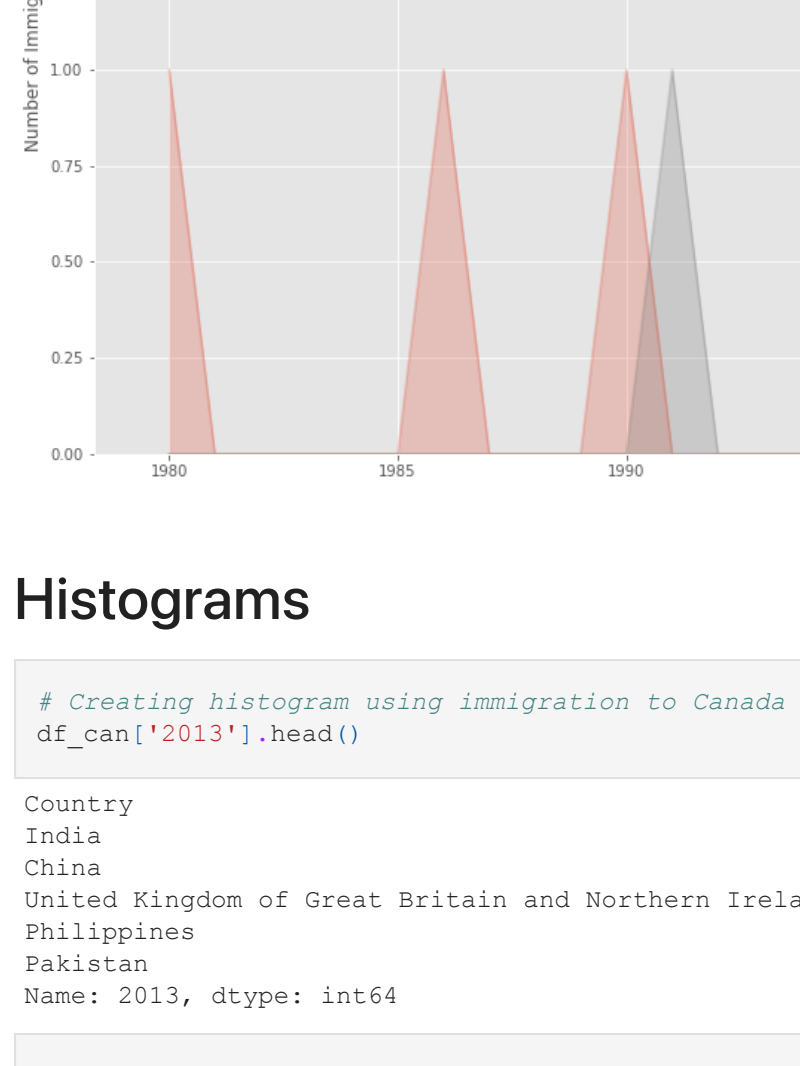
Out[28]:

Country	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2004	2005	2006	2007	2008	2009	2010	2011
Denmark	272	293	299	106	93	73	93	109	129	129	...	89	62	101	97	108	81	92	93
Norway	116	77	106	51	31	54	56	80	73	76	...	73	57	53	73	66	75	46	49
Sweden	281	308	222	176	128	158	187	198	171	182	...	129	205	139	193	165	167	159	134

3 rows × 34 columns

```
In [29]: df_can.loc[['Denmark', 'Norway', 'Sweden'], years].plot.hist()
```

```
Out[29]: <AxesSubplot:ylabel='Frequency'>
```

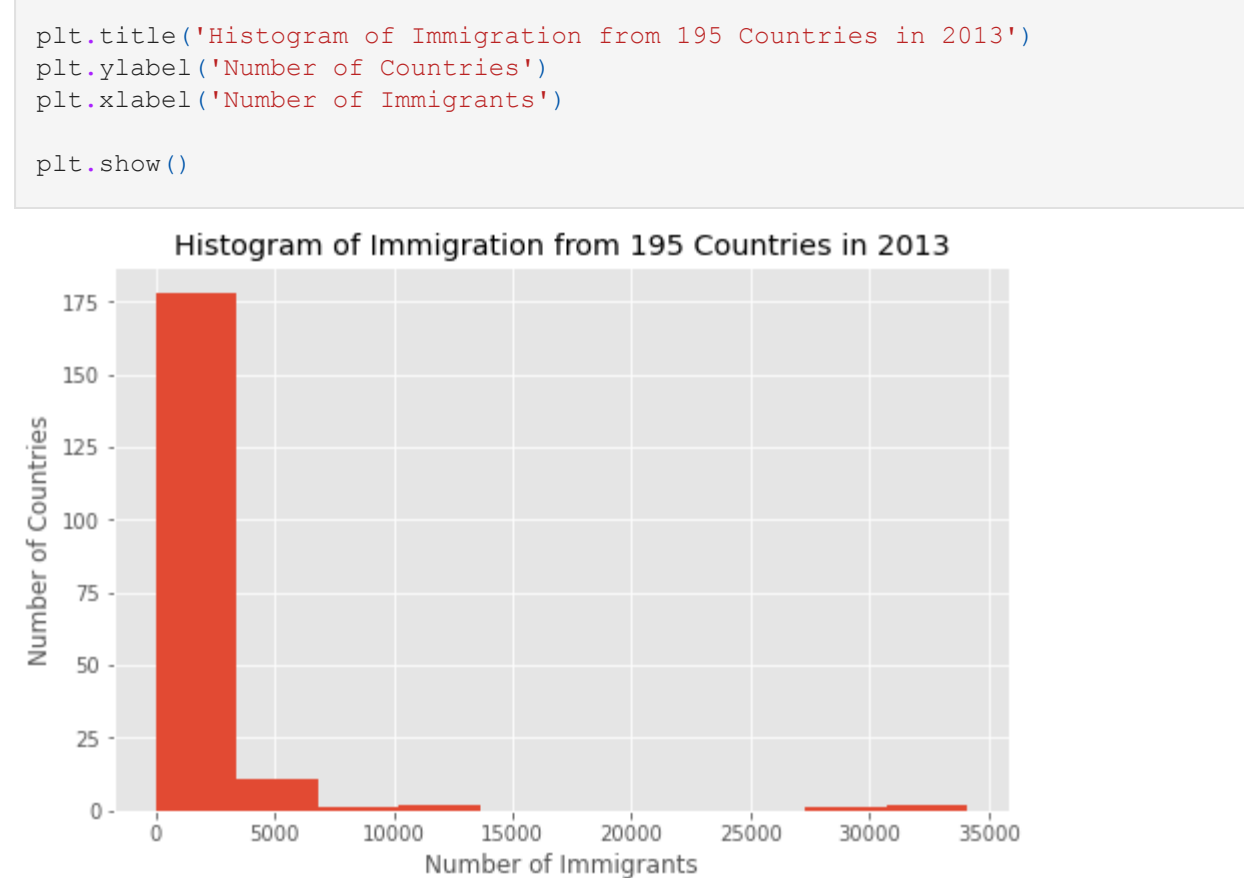



```
In [30]: # need to first transpose the dataframe
         df_t = df_can.loc[['Denmark', 'Norway', 'Sweden'], years].transpose()
         df_t.head()
```

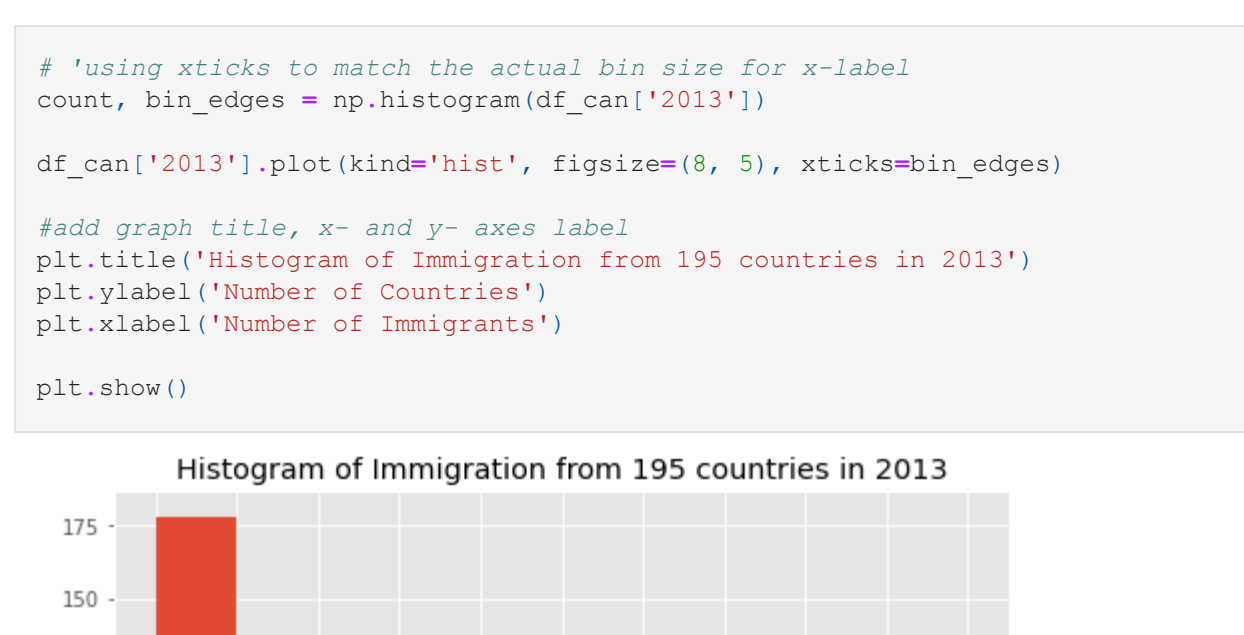
Out[30]:

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

```
In [31]: #then generate the histogram
         df_t.plot(kind='hist', figsize=(10, 6))

         plt.title('Histogram of Immigration from Denmark, Norway, and Sweden from 1980 - 2013')
         plt.ylabel('Number of Years')
         plt.xlabel('Number of Immigrants')

         plt.show()
```



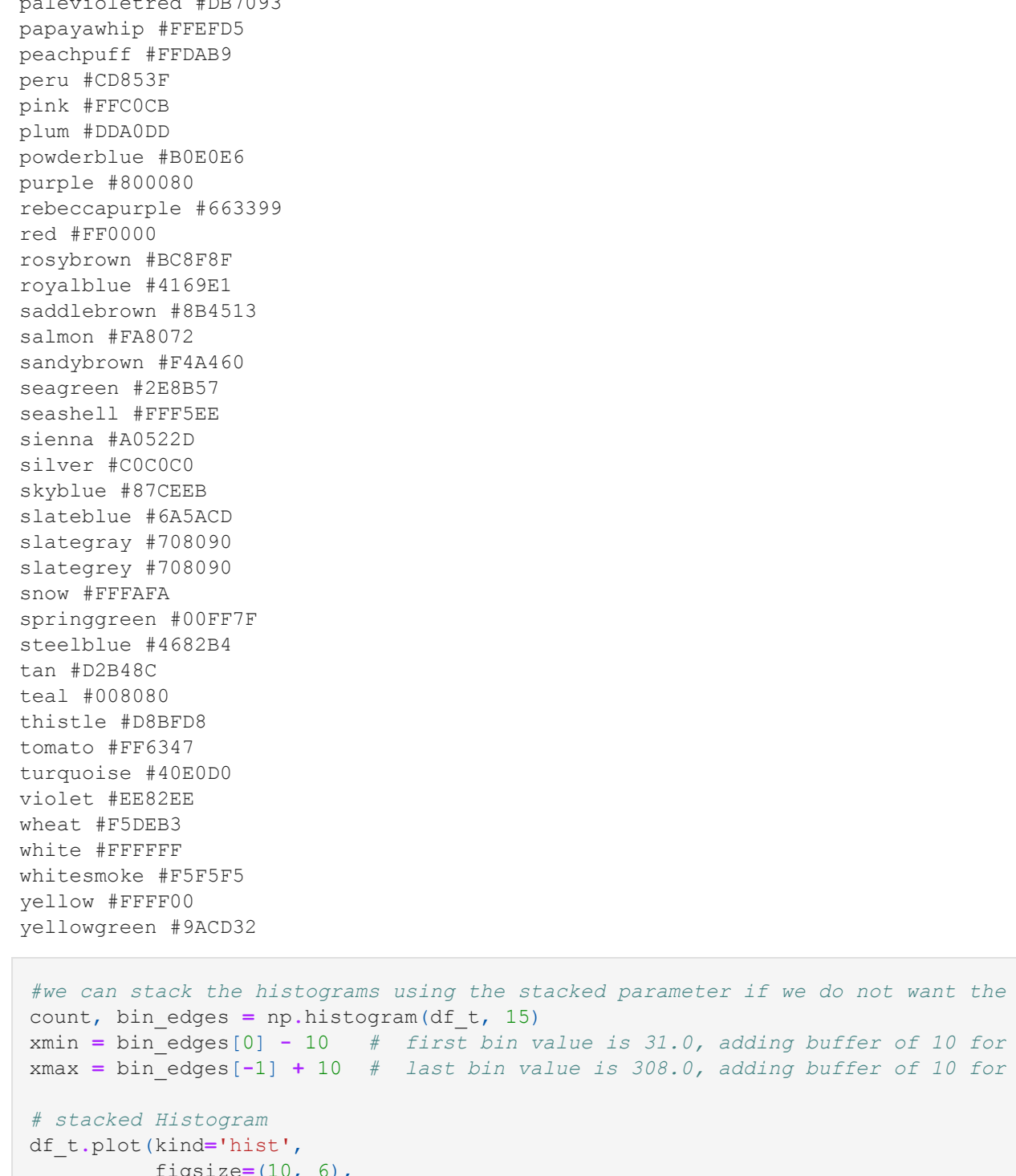

```
In [32]: #increase the bin size to 15 by passing in bins parameter;
         #set transparency to 60% by passing in alpha parameter;
         #label the x-axis by passing in x-label parameter;
         #change the colors of the plots by passing in color parameter.
         count, bin_edges = np.histogram(df_t, 15)

         # un-stacked histogram
         df_t.plot(kind ='hist',
                   figsize=(10, 6),
                   bins=15,
                   alpha=0.6,
                   xticks=bin_edges,
                   color=['coral', 'darkslateblue', 'mediumseagreen']
                  )

         plt.title('Histogram of Immigration from Denmark, Norway, and Sweden from 1980 - 2013')
         plt.ylabel('Number of Years')
         plt.xlabel('Number of Immigrants')

         plt.show()
```




```
In [33]: #Full list of colours in Matplotlib
         import matplotlib
         for name, hex in matplotlib.colors.cnames.items():
             print(name, hex)
```

```
aliceblue #F0F8FF
antiquewhite #FAEBD7
aqua #00FFFF
aquamarine #7FFFD4
azure #F0FFFF
beige #F5F5DC
bisque #FFE4C4
black #000000
blanchedalmond #FFEBCD
blue #0000FF
blueviolet #8A2BE2
brown #A52A2A
burlywood #DEB887
cadetblue #5F9EA0
chartreuse #7FFF00
chocolate #D2691E
coral #FF7F50
cornflowerblue #6495ED
cornsilk #FFF8DC
crimson #DC143C
cyan #00FFFF
darkblue #00008B
darkcyan #008B8B
darkgoldenrod #B8860B
darkgray #A9A9A9
darkgreen #006400
darkgrey #A9A9A9
darkkhaki #BDB76B
darkmagenta #8B008B
darkolivegreen #556B2F
darkorange #FF8C00
darkorchid #9932CC
darkred #8B0000
darksalmon #E9967A
darkseagreen #8FBC8F
darkslateblue #483D8B
darkslategray #2F4F4F
darkslategrey #2F4F4F
darkturquoise #00CED1
darkviolet #9400D3
deeppink #FF1493
deepskyblue #00BFFF
dimgray #696969
dimgrey #696969
dodgerblue #1E90FF
firebrick #B22222
floralwhite #FFFAF0
forestgreen #228B22
fuchsia #FF00FF
gainsboro #DCDCDC
ghostwhite #F8F8FF
gold #FFD700
goldenrod #DAA520
gray #808080
green #008000
greenyellow #ADFF2F
grey #808080
honeydew #F0FFF0
hotpink #FF69B4
indianred #CD5C5C
indigo #4B0082
ivory #FFFFF0
khaki #F0E68C
lavender #E6E6FA
lavenderblush #FFF0F5
lawngreen #7CFC00
lemonchiffon #FFFACD
lightblue #ADD8E6
lightcoral #F08080
lightcyan #E0FFFF
lightgoldenrodyellow #FAFAD2
lightgray #D3D3D3
lightgreen #90EE90
lightgrey #D3D3D3
lightpink #FFB6C1
lightsalmon #FFA07A
lightseagreen #20B2AA
lightskyblue #87CEFA
lightslategray #778899
lightslategrey #778899
lightsteelblue #B0C4DE
lightyellow #FFFFE0
lime #00FF00
limegreen #32CD32
linen #FAF0E6
magenta #FF00FF
maroon #800000
mediumaquamarine #66CDAA
mediumblue #0000CD
mediumorchid #BA55D3
mediumpurple #9370DB
mediumseagreen #3CB371
mediumslateblue #7B68EE
mediumspringgreen #00FA9A
mediumturquoise #48D1CC
mediumvioletred #C71585
midnightblue #191970
mintcream #F5FFFA
mistyrose #FFE4E1
moccasin #FFE4B5
navajowhite #FFDEAD
navy #000080
oldlace #FDF5E6
olive #808000
olivedrab #6B8E23
orange #FFA500
orangered #FF4500
orchid #DA70D6
palegoldenrod #EEE8AA
palegreen #98FB98
paleturquoise #AFEEEE
palevioletred #DB7093
papayawhip #FFEFD5
peachpuff #FFDAB9
peru #CD853F
pink #FFC0CB
plum #DDA0DD
powderblue #B0E0E6
purple #800080
rebeccapurple #663399
red #FF0000
rosybrown #BC8F8F
royalblue #4169E1
saddlebrown #8B4513
salmon #FA8072
sandybrown #F4A460
seagreen #2E8B57
seashell #FFF5EE
sienna #A0522D
silver #C0C0C0
skyblue #87CEEB
slateblue #6A5ACD
slategray #708090
slategrey #708090
snow #FFFAFA
springgreen #00FF7F
steelblue #4682B4
tan #D2B48C
teal #008080
thistle #D8BFD8
tomato #FF6347
turquoise #40E0D0
violet #EE82EE
wheat #F5DEB3
white #FFFFFF
whitesmoke #F5F5F5
yellow #FFFF00
yellowgreen #9ACD32
```

```
In [34]: #we can stack the histograms using the stacked parameter if we do not want the plots to overlap each other,
         count, bin_edges = np.histogram(df_t, 15)
         xmin = bin_edges[0] - 10   #  first bin value is 31.0, adding buffer of 10 for aesthetic purposes
         xmax = bin_edges[-1] + 10  #  last bin value is 308.0, adding buffer of 10 for aesthetic purposes

         # stacked Histogram
         df_t.plot(kind='hist',
                   figsize=(10, 6),
                   bins=15,
                   xticks=bin_edges,
                   color=['coral', 'darkslateblue', 'mediumseagreen'],
                   stacked=True,
                   xlim=(xmin, xmax)
                  )

         plt.title('Histogram of Immigration from Denmark, Norway, and Sweden from 1980 - 2013')
         plt.ylabel('Number of Years')
         plt.xlabel('Number of Immigrants')

         plt.show()
```




```
In [36]: #immigration distribution for Greece, Albania, and Bulgaria
         df_can.loc[['Greece', 'Albania', 'Bulgaria'], years]
```

Out[36]:

Country	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2004	2005	2006	2007	2008	2009	2010	2011
Greece	1065	953	897	633	580	584	547	765	564	773	...	120	100	74	110	107	119	101	102
Albania	1	0	0	0	0	0	1	2	2	3	...	1450	1223	856	702	560	716	561	539
Bulgaria	24	20	12	33	11	24	33	52	43	85	...	2022	1738	1419	1172	994	784	556	365

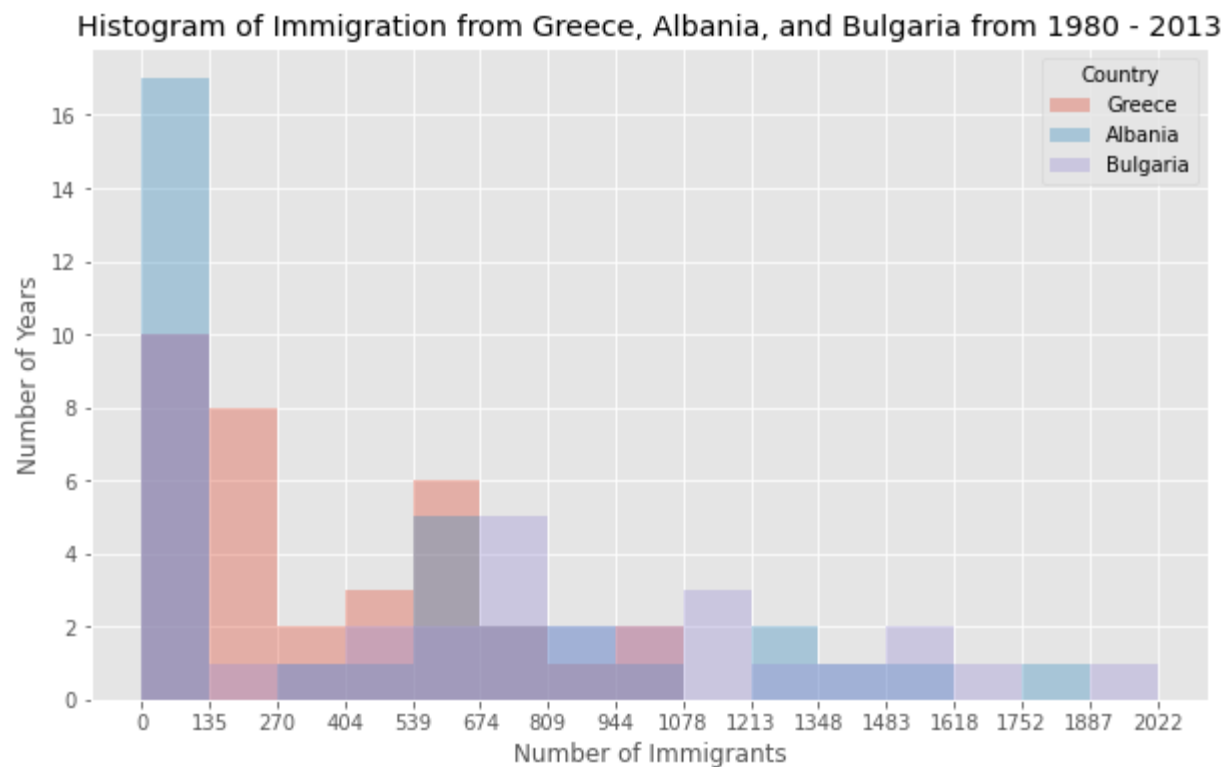
3 rows × 34 columns

```
In [37]: df_t1 = df_can.loc[['Greece', 'Albania', 'Bulgaria'], years].transpose()
         df_t1.head()
```

Out[37]:

Country	Greece	Albania	Bulgaria
1980	1065	1	24
1981	953	0	20
1982	897	0	12
1983	633	0	33
1984	580	0	11

```
In [41]: count, bin_edges = np.histogram(df_t1, 15)
         df_t1.plot(kind='hist', figsize=(10, 6), alpha=.35, bins=15, xticks=bin_edges)

         plt.title('Histogram of Immigration from Greece, Albania, and Bulgaria from 1980 - 2013')
         plt.ylabel('Number of Years')
         plt.xlabel('Number of Immigrants')

         plt.show()
```



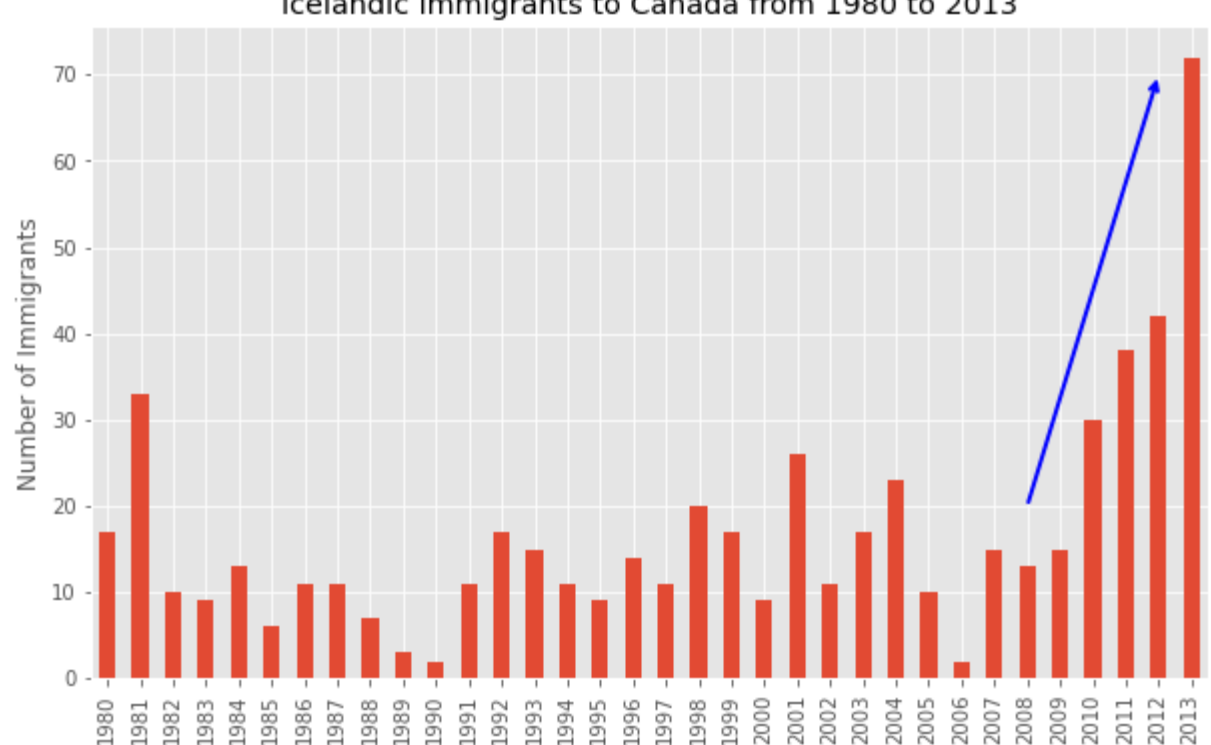
Bar Charts

In [42]:
```
#Plotting Bar Charts
#comparing the number of Icelandic immigrants to Canada from year 1980 to 2013.

df_iceland = df_can.loc['Iceland', years]
df_iceland.head()
```

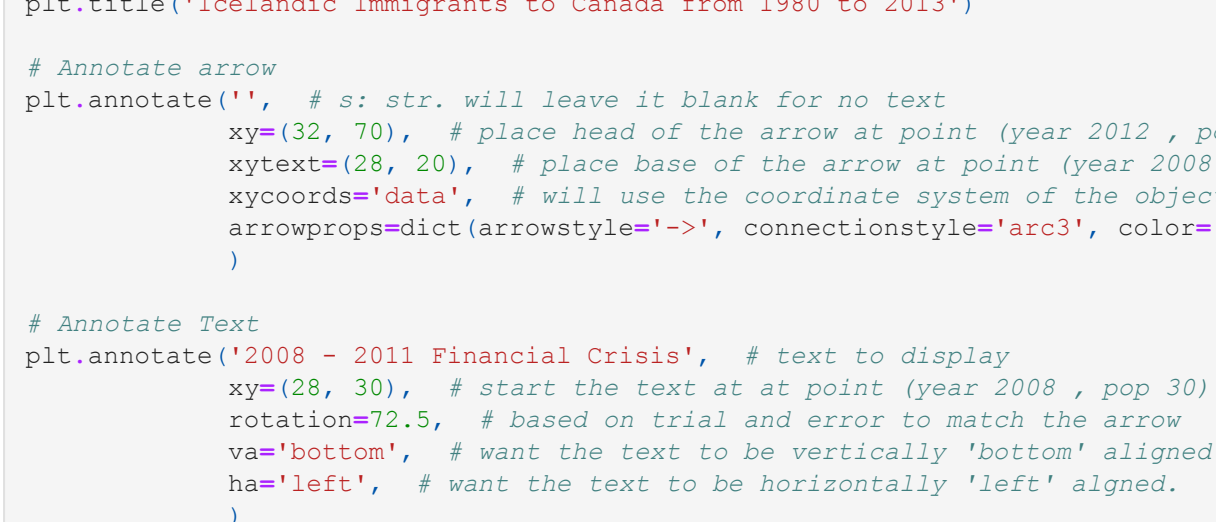
Out[42]:
```
1980    17
1981    33
1982    10
1983     9
1984    13
Name: Iceland, dtype: object
```

In [43]:
```
df_iceland.plot(kind='bar', figsize=(10, 6))

# add x-label, y-label and title to the plot
plt.xlabel('Year')
plt.ylabel('Number of immigrants')
plt.title('Icelandic immigrants to Canada from 1980 to 2013')

plt.show()
```

In [44]:
```
#annotating arrow on the plot
df_iceland.plot(kind='bar', figsize=(10, 6), rot=90)  # rotate the xticks(labelled points on x-axis) by 90 degr

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

# Annotate arrow
plt.annotate('',  # s: str. Will leave it blank for no text
             xy=(32, 70),  # place head of the arrow at point (year 2012 , pop 70)
             xytext=(28, 20),  # place base of the arrow at point (year 2008 , pop 20)
             xycoords='data',  # will use the coordinate system of the object being annotated
             arrowprops=dict(arrowstyle='->', connectionstyle='arc3', color='blue', lw=2)
             )

plt.show()
```



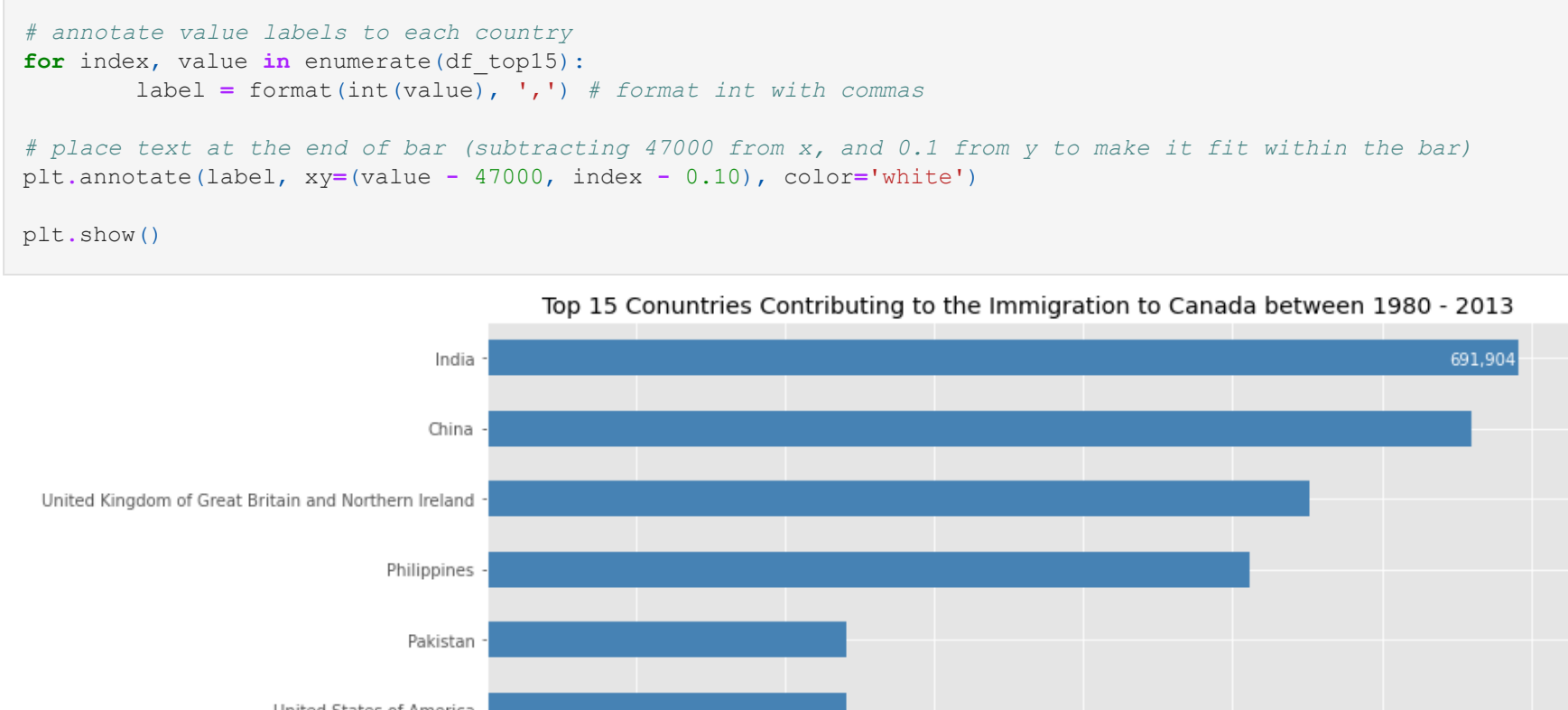
In [45]:
```
#rotation: rotation angle of text in degrees (counter clockwise)
#va: vertical alignment of text ['center' | 'top' | 'bottom' | 'baseline']
#ha: horizontal alignment of text ['center' | 'right' | 'left']

df_iceland.plot(kind='bar', figsize=(10, 6), rot=45)

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

# Annotate arrow
plt.annotate('',  # s: str. will leave it blank for no text
             xy=(32, 70),  # place head of the arrow at point (year 2012 , pop 70)
             xytext=(28, 20),  # place base of the arrow at point (year 2008 , pop 20)
             xycoords='data',  # will use the coordinate system of the object being annotated
             arrowprops=dict(arrowstyle='->', connectionstyle='arc3', color='blue', lw=2)
             )

# Annotate Text
plt.annotate('2008 - 2011 Financial Crisis',  # text to display
             xy=(28, 30),  # start the text at at point (year 2008 , pop 30)
             rotation=72.5,  # based on trial and error to match the arrow
             va='bottom',  # want the text to be vertically 'bottom' aligned
             ha='left',  # want the text to be horizontally 'left' aligned.
             )

plt.show()
```



In [53]:
```
#horizontal bar plot showing the total number of immigrants to Canada from the top 15 countries, from 1980 - 20
#Label each country with the total immigrant count

# sort dataframe on 'Total' column (descending)
df_can.sort_values(by='Total', ascending=True, inplace=True)

# get top 15 countries
df_top15 = df_can['Total'].tail(15)
df_top15
```

Out[53]:
```
Country
Romania                                                  93585
Viet Nam                                                 97146
Jamaica                                                 106431
France                                                  109091
Lebanon                                                 115359
Poland                                                  139241
Republic of Korea                                       142581
Sri Lanka                                               148358
Iran (Islamic Republic of)                              175923
United States of America                                241122
Pakistan                                                241600
Philippines                                             511391
United Kingdom of Great Britain and Northern Ireland    551500
China                                                   659962
India                                                   691904
Name: Total, dtype: int64
```

In [57]:
```
# generate plot
df_top15.plot(kind='barh', figsize=(12, 12), color='steelblue')
plt.xlabel('Number of Immigrants')
plt.title('Top 15 Conuntries Contributing to the Immigration to Canada between 1980 - 2013')

# annotate value labels to each country
for index, value in enumerate(df_top15):
        label = format(int(value), ',') # format int with commas

# place text at the end of bar (subtracting 47000 from x, and 0.1 from y to make it fit within the bar)
plt.annotate(label, xy=(value - 47000, index - 0.10), color='white')

plt.show()
```



Pie Charts

In [59]:
```
#the proportion (percentage) of new immigrants grouped by continents for the entire time period from 1980 to 20
df_continents = df_can.groupby("Continent", axis=0).sum()

print(type(df_can.groupby('Continent', axis=0)))

df_continents.head()
```

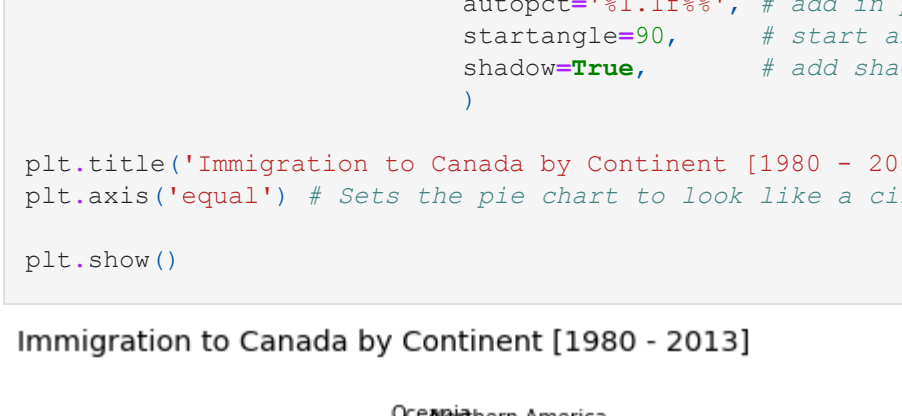
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>

Out[59]:

	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2005	2006	2007	2008	2009
Continent																
Africa	3951	4363	3819	2671	2639	2650	3782	7494	7552	9894	...	27523	29188	28284	29890	34534
Asia	31025	34314	30214	24696	27274	23850	28739	43203	47454	60256	...	159253	149054	133459	139894	141434
Europe	39760	44802	42720	24638	22287	20844	24370	46698	54726	60893	...	35955	33053	33495	34692	35078
Latin America and the Caribbean	13081	15215	16769	15427	13678	15171	21179	28471	21924	25060	...	24747	24676	26011	26547	26867
Northern America	9378	10030	9074	7100	6661	6543	7074	7705	6469	6790	...	8394	9613	9463	10190	8995

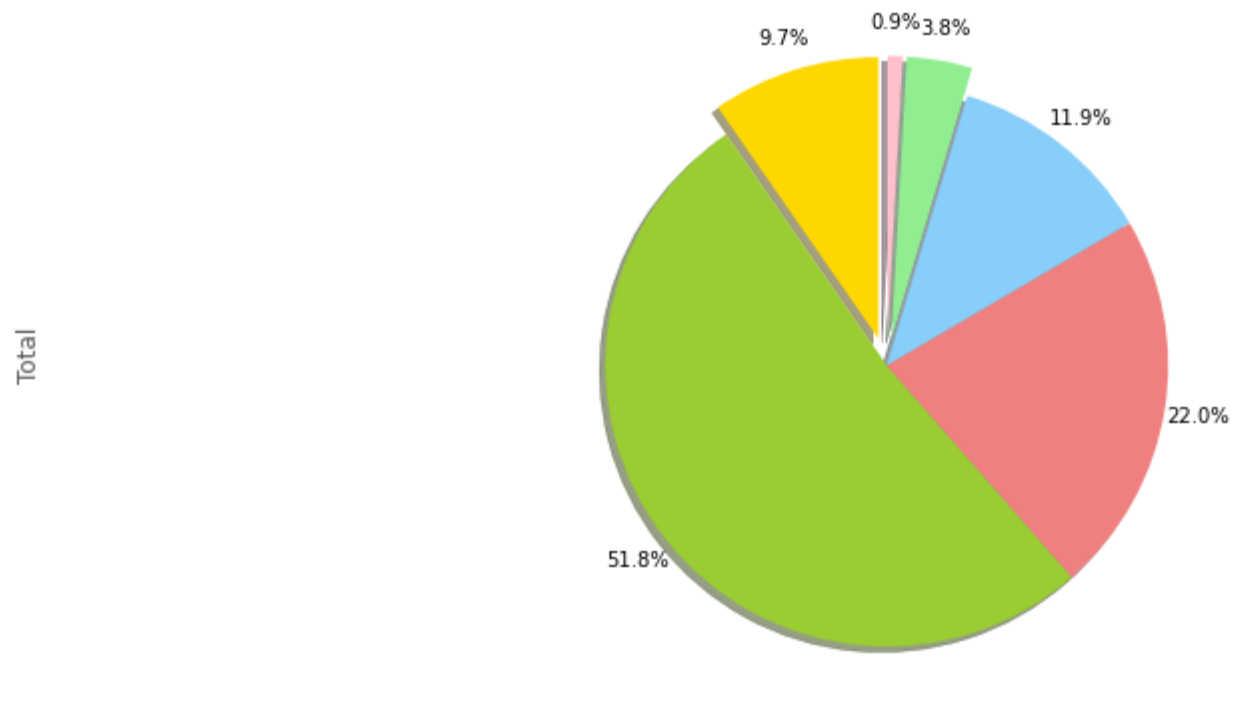
5 rows × 35 columns

In [60]:
```
#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).

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.show()
```



In [61]:
```
#Removing text and figures overlap
#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 C

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=(15, 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
                            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)

plt.axis('equal')
```

Out[61]:
```
(-1.1107545164204682,
 1.1260912315394227,
 -1.1138238637760542,
 1.2053818739377367)
```



In [63]:
```
#New immigrants grouped by continent in the year 2013

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 labe
                            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')

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

plt.show()
```

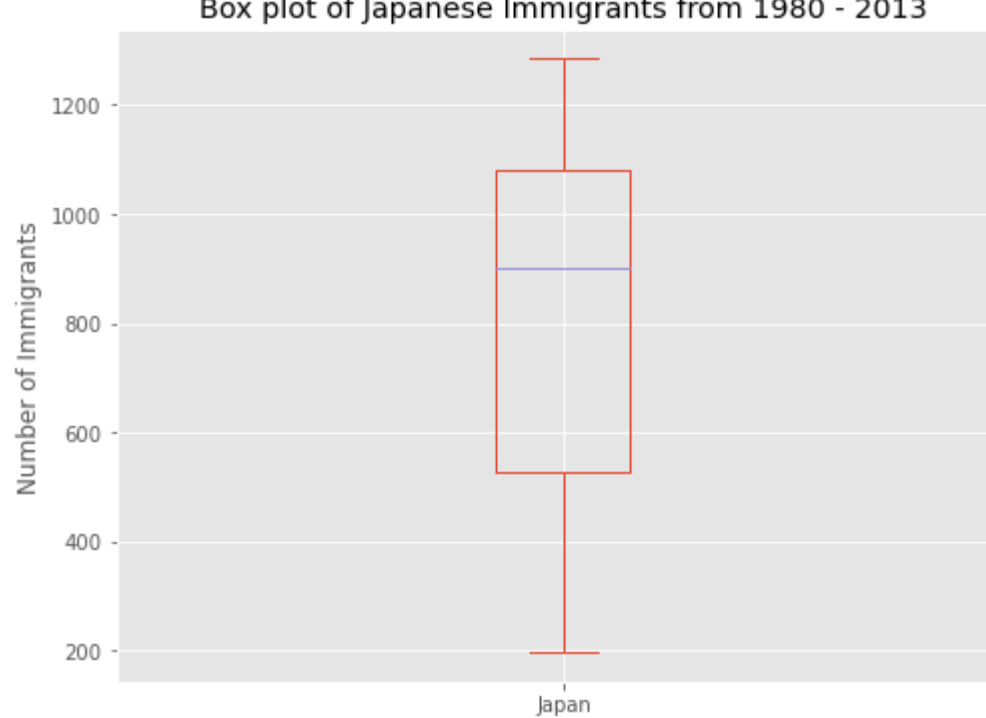
Box Plots

In [64]:
```
# Japanese immigrants between 1980 - 2013

df_japan = df_can.loc[['Japan'], years].transpose()
df_japan.head()
```

Out[64]:

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

In [65]:
```
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()
```



In [66]:
```
df_japan.describe()
```

Out[66]:

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

In [67]:
```
#Analyzing India and China immigration trend

df_CI= df_can.loc[['China', 'India'], years].transpose()
df_CI.head()
```

Out[67]:

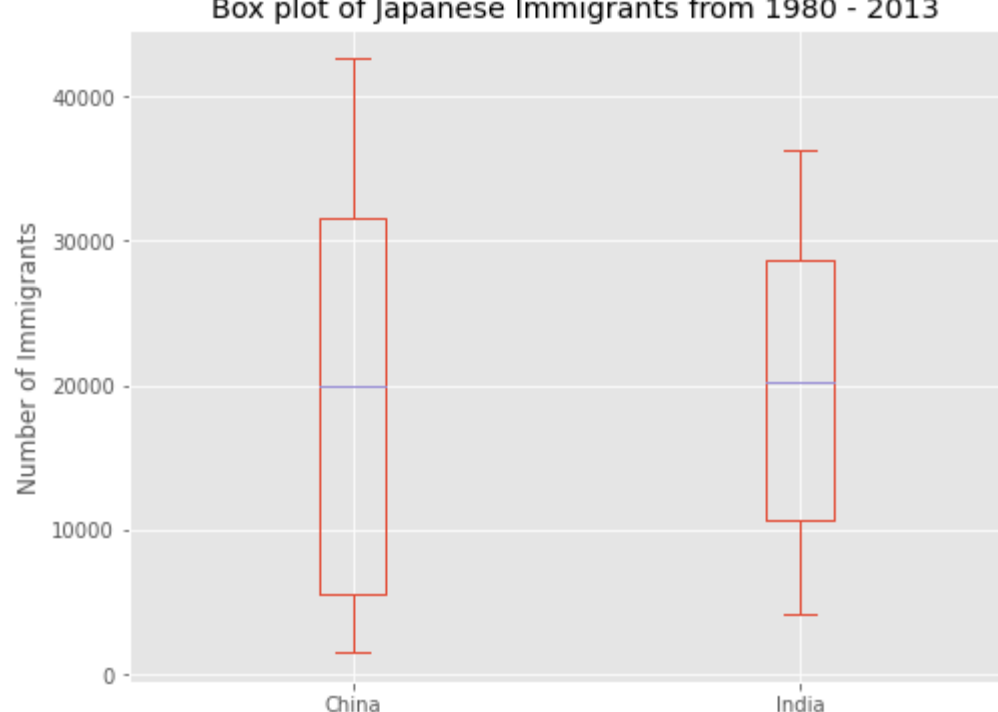
Country	China	India
1980	5123	8880
1981	6682	8670
1982	3308	8147
1983	1863	7338
1984	1527	5704

In [68]:
```
df_CI.describe()
```

Out[68]:

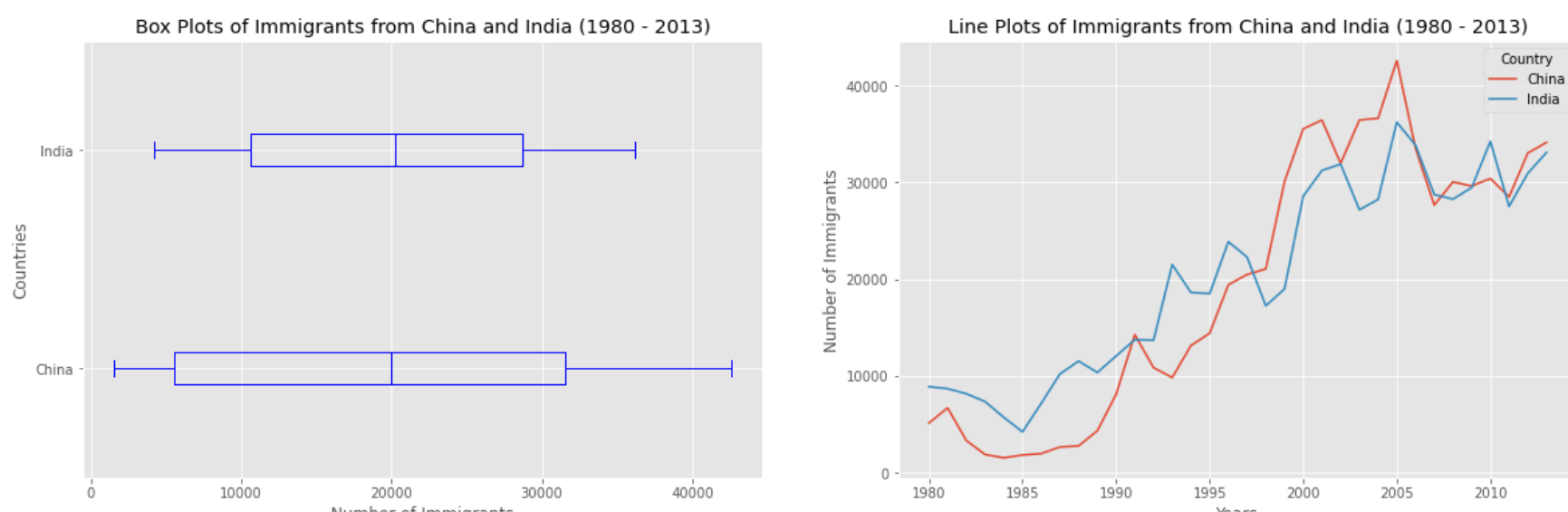
Country	China	India
count	34.000000	34.000000
mean	19410.647059	20350.117647
std	13568.230790	10007.342579
min	1527.000000	4211.000000
25%	5512.750000	10637.750000
50%	19945.000000	20235.000000
75%	31568.500000	28699.500000
max	42584.000000	36210.000000

In [69]:
```
df_CI.plot(kind='box', figsize=(8, 6))

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

plt.show()
```



In [70]:
```
# 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()
```

In [71]:
```
#Subplots: multiple plots within the same figure
#side by side comparison of the box plot with the line plot of China and India's immigration.
#first 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 gr
#plot_number starts at 1, increments across rows first and has a maximum of nrows * ncols as shown below.

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()
```



In [72]:
```
#box plot to visualize the distribution of the top 15 countries (based on total immigration) grouped by the dec
#first sort total immigrant in descending order to get the top 15 countries

df_top15 = df_can.sort_values(['Total'], ascending=False, axis=0).head(15)
df_top15
```

Out[72]:

	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	20
Country																
India	Asia	Southern Asia	Developing regions	8880	8670	8147	7338	5704	4211	7150	...	36210	33848	28742	28261	294
China	Asia	Eastern Asia	Developing regions	5123	6682	3308	1863	1527	1816	1960	...	42584	33518	27642	30037	296
United Kingdom of Great Britain and Northern Ireland	Europe	Northern Europe	Developed regions	22045	24796	20620	10015	10170	9564	9470	...	7258	7140	8216	8979	88
Philippines	Asia	South-Eastern Asia	Developing regions	6051	5921	5249	4562	3801	3150	4166	...	18139	18400	19837	24887	285
Pakistan	Asia	Southern Asia	Developing regions	978	972	1201	900	668	514	691	...	14314	13127	10124	8994	72
United States of America	Northern America	Northern America	Developed regions	9378	10030	9074	7100	6661	6543	7074	...	8394	9613	9463	10190	89
Iran (Islamic Republic of)	Asia	Southern Asia	Developing regions	1172	1429	1822	1592	1977	1648	1794	...	5837	7480	6974	6475	65
Sri Lanka	Asia	Southern Asia	Developing regions	185	371	290	197	1086	845	1838	...	4930	4714	4123	4756	45
Republic of Korea	Asia	Eastern Asia	Developing regions	1011	1456	1572	1081	847	962	1208	...	5832	6215	5920	7294	58
Poland	Europe	Eastern Europe	Developed regions	863	2930	5881	4546	3588	2819	4808	...	1405	1263	1235	1267	10
Lebanon	Asia	Western Asia	Developing regions	1409	1119	1159	789	1253	1683	2576	...	3709	3802	3467	3566	30
France	Europe	Western Europe	Developed regions	1729	2027	2219	1490	1169	1177	1298	...	4429	4002	4290	4532	50
Jamaica	Latin America and the Caribbean	Caribbean	Developing regions	3198	2634	2661	2455	2508	2938	4649	...	1945	1722	2141	2334	24
Viet Nam	Asia	South-Eastern Asia	Developing regions	1191	1829	2162	3404	7583	5907	2741	...	1852	3153	2574	1784	2
Romania	Europe	Eastern Europe	Developed regions	375	438	583	543	524	604	656	...	5048	4468	3834	2837	20

15 rows × 38 columns

In [73]:
```
    # then create a list of all years in decades 80's, 90's, and 00's
    years_80s = list(map(str, range(1980, 1990)))
    years_90s = list(map(str, range(1990, 2000)))
    years_00s = list(map(str, range(2000, 2010)))

    # slice the original dataframe df_can to create a series for each decade
    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)

    # merge the three series into a new data frame
    new_df = pd.DataFrame({'1980s': df_80s, '1990s': df_90s, '2000s':df_00s})

    # display dataframe
    new_df.head()
```

Out[73]:

	1980s	1990s	2000s
Country			
India	82154	180395	303591
China	32003	161528	340385
United Kingdom of Great Britain and Northern Ireland	179171	261966	83413
Philippines	60764	138482	172904
Pakistan	10591	65302	127598

In [74]:
```
new_df.describe()
```

Out[74]:

	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

In [76]:
```
new_df.plot(kind='box', figsize=(10, 6))

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

plt.show()
```

In [77]:
```
#outliers: to be an outlier;
# 1. must be larger than Q3 by at least 1.5 times the interquartile range (IQR), or,
#2. smaller than Q1 by at least 1.5 times the IQR.

# As and example, for the decade 2000s:
#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 outlie

#Outlier > 105,505.5 + (1.5 * 69,404)
#Outlier > 209,611.5
#To check the nymber of entries above the outlier threshold

new_df=new_df.reset_index()
new_df[new_df['2000s']> 209611.5]
```

Out[77]:

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

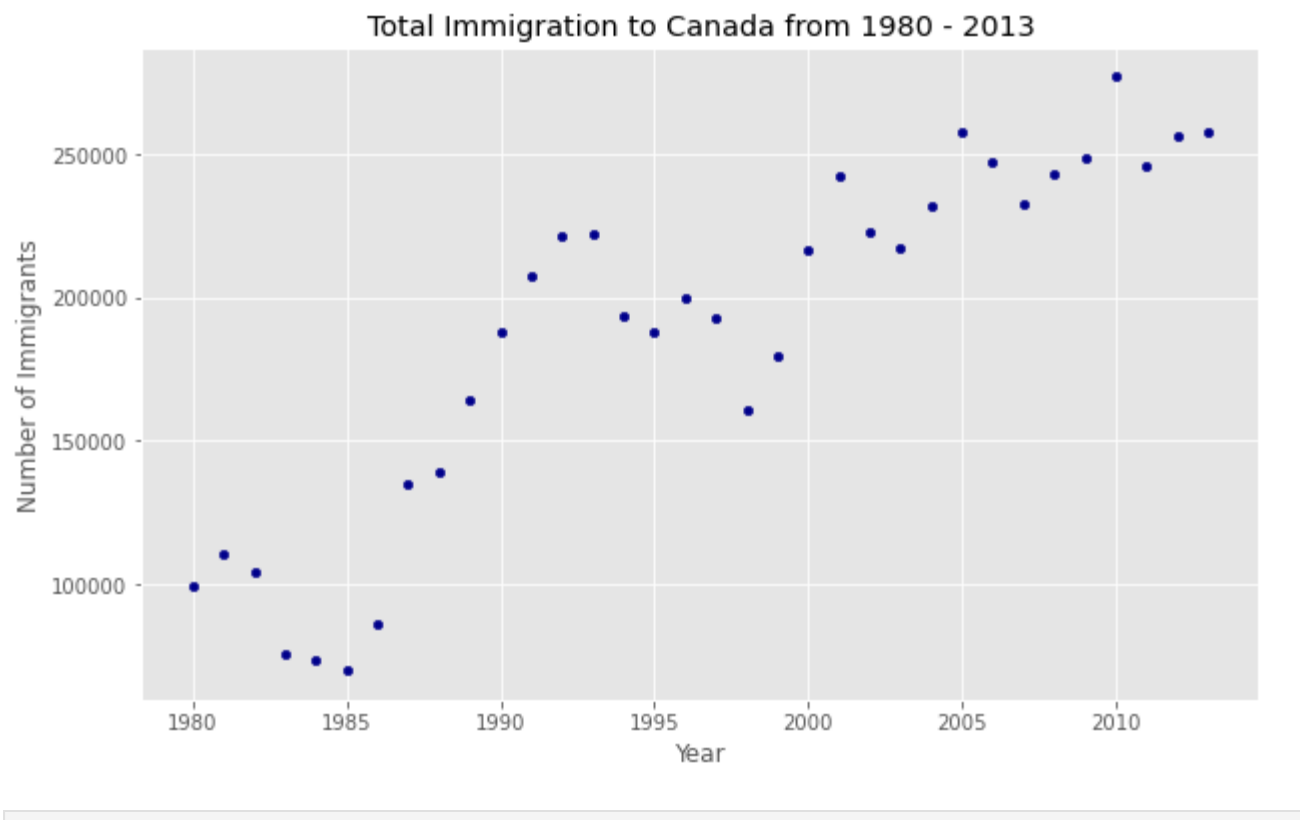
Scatter Plots

In [78]:
```
#visualizing the trend of total immigrantion to Canada (all countries combined) for the years 1980 - 2013.

# we 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 it back in as a column in the df_tot dataframe
df_tot.reset_index(inplace = True)

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

df_tot.head()
```

Out[78]:

	year	total
0	1980	99137
1	1981	110563
2	1982	104271
3	1983	75550
4	1984	73417

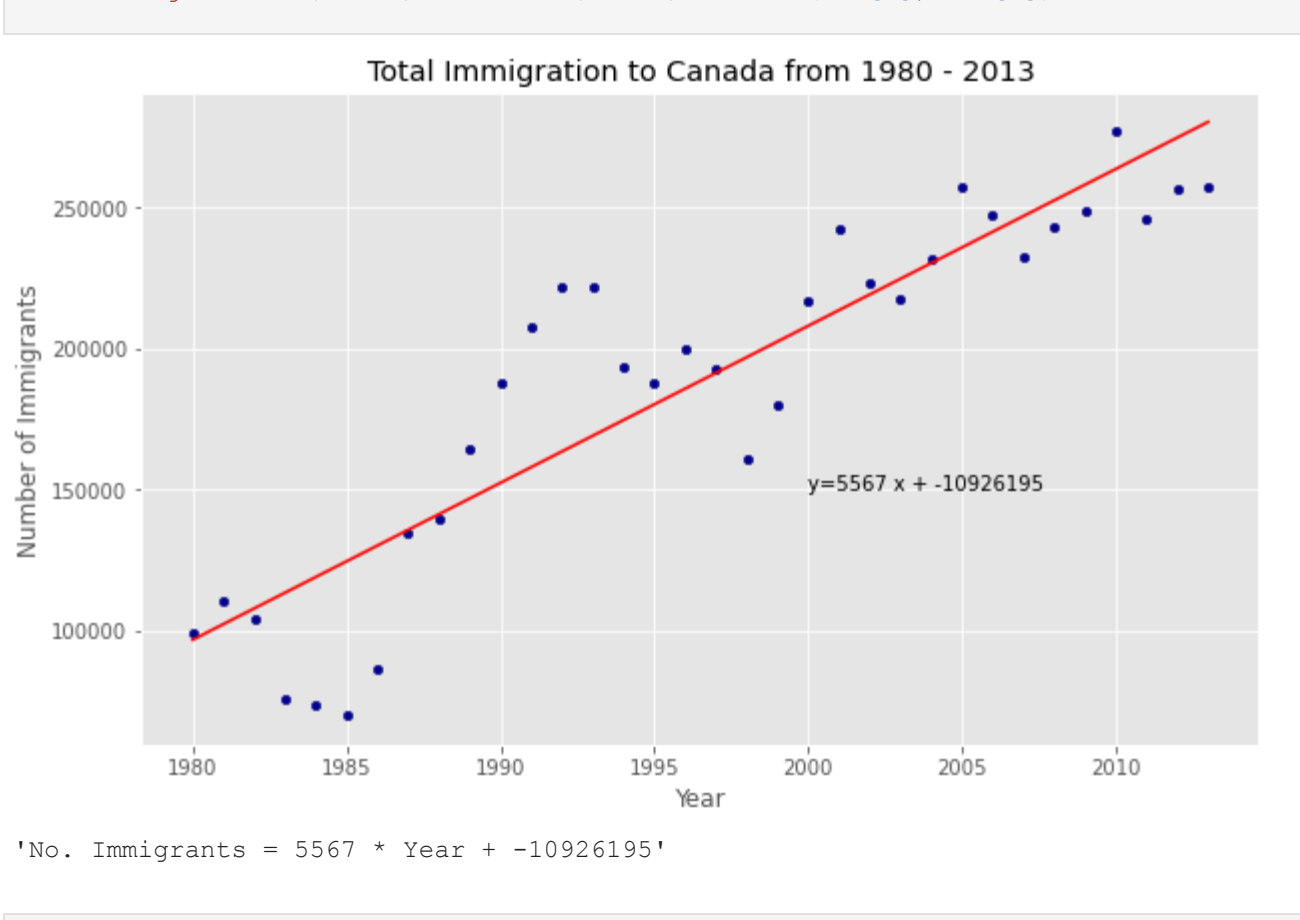
In [79]:
```
df_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6), color='darkblue')

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

plt.show()
```



In [80]:
```python
#best fit line to predict the number of immigrants in 2015.

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

fit
```

Out[80]:
```
array([ 5.56709228e+03, -1.09261952e+07])
```

In [81]:
```python
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])
```



Out[81]:
```
'No. Immigrants = 5567 * Year + -10926195'
```

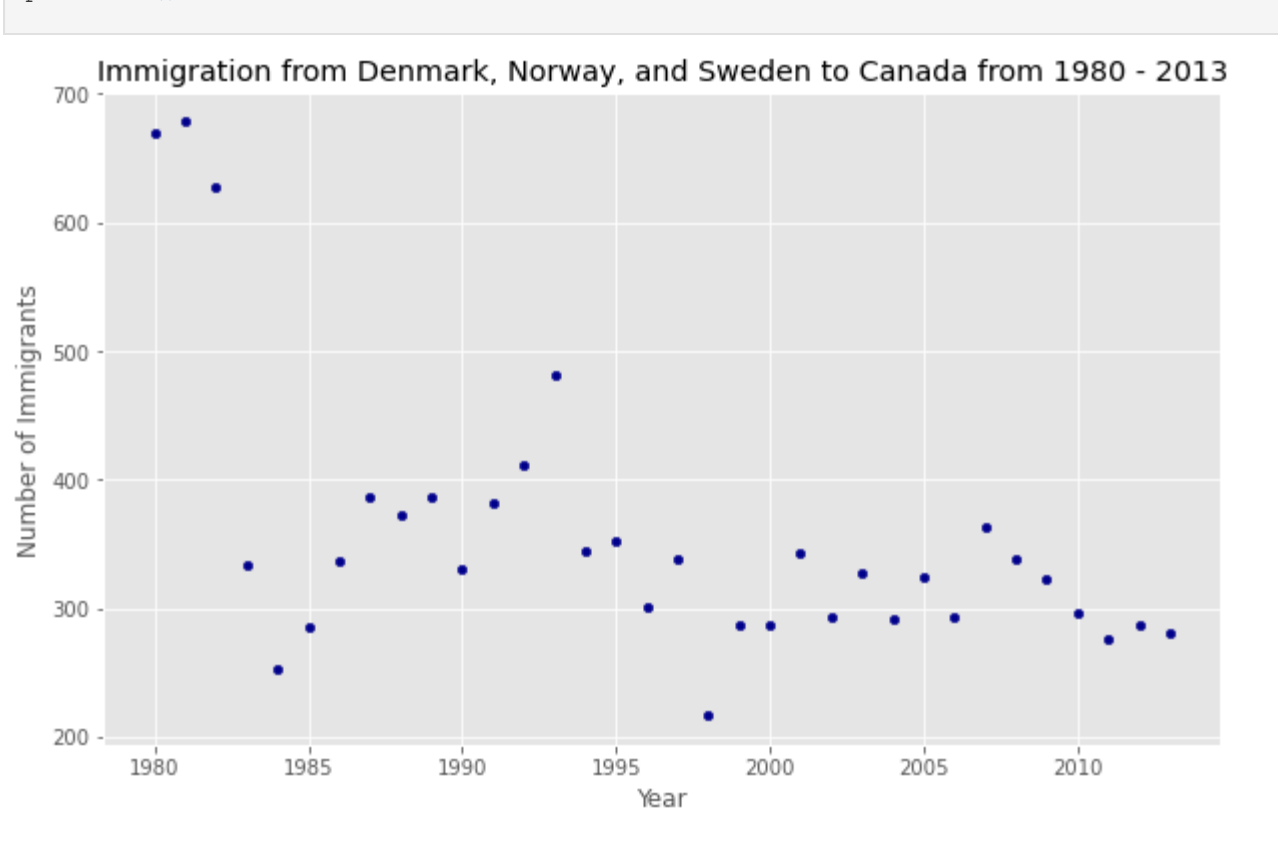
In [ ]:
```python
#Using the equation of line of best fit, we can estimate the number of immigrants in 2015:
#No. Immigrants = 5567 * Year - 10926195
#No. Immigrants = 5567 * 2015 - 10926195
#No. Immigrants = 291,310
```

In [82]:
```python
#scatter plot of the total immigration from Denmark, Norway, and Sweden to Canada from 1980 to 2013?

#Create a dataframe that consists of the numbers associated with Denmark, Norway, and Sweden only. Name it df_c
#Sum the immigration numbers across all three countries for each year and turn the result into a dataframe. Nam
#Reset the index in place.
#Rename the columns to year and total.
#Display the resulting dataframe.


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

# create df_total by summing across three countries for each year
df_total = pd.DataFrame(df_countries.sum(axis=1))

# reset index in place
df_total.reset_index(inplace=True)

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

# change column year from string to int to create scatter plot
df_total['year'] = df_total['year'].astype(int)

df_total.head()
```

Out[82]:

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

In [83]:
```python
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

In [84]:
```python
#Effect of Argentina's great depression
#Comparing Argentina's immigration to Canada with its neighbour Brazil from 1980 to 2013

# 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)

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

df_can_t.reset_index(inplace=True)

df_can_t.head()
```

Out[84]:

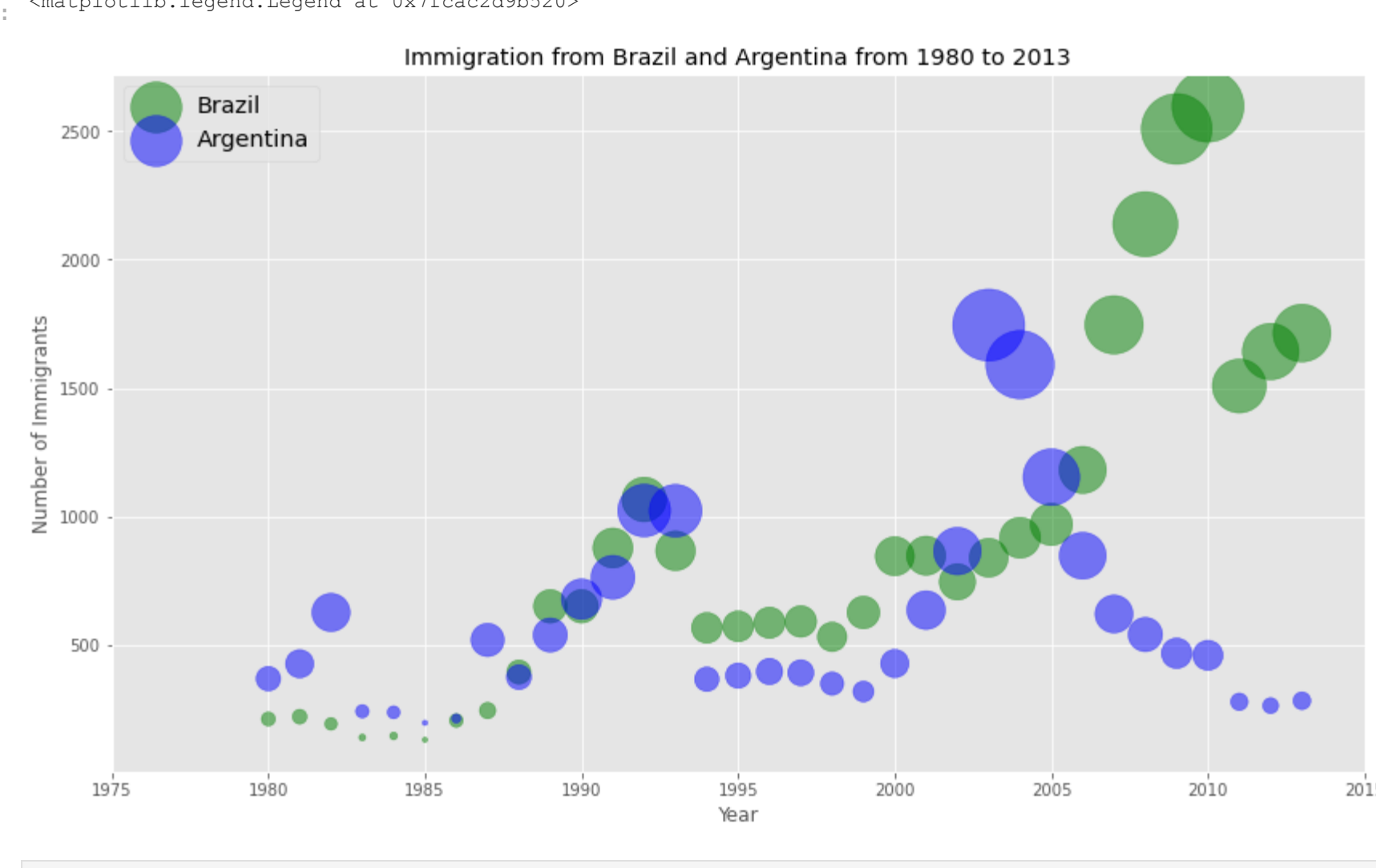
Country	Year	Palau	Western Sahara	Marshall Islands	New Caledonia	San Marino	American Samoa	Tuvalu	Sao Tome and Principe	Vanuatu	...	Poland	Republic of Korea	Sri Lanka	(Isla Repu
0	1980	0	0	0	0	1	0	0	0	0	...	863	1011	185	
1	1981	0	0	0	0	0	1	1	0	0	...	2930	1456	371	[illegible]
2	1982	0	0	0	0	0	0	0	0	0	...	5881	1572	290	[illegible]
3	1983	0	0	0	0	0	0	0	0	0	...	4546	1081	197	[illegible]
4	1984	0	0	0	0	0	0	1	0	0	...	3588	847	1086	[illegible]

5 rows × 196 columns

In [85]:
```python
#Creating normalized weights
#we use feature scaling to bring all values into the range [0, 1]. The general formula is:

#X(prime) = X-X(min)/X(max)-X(min)
#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'].

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

In [86]:
```python
# 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')
```

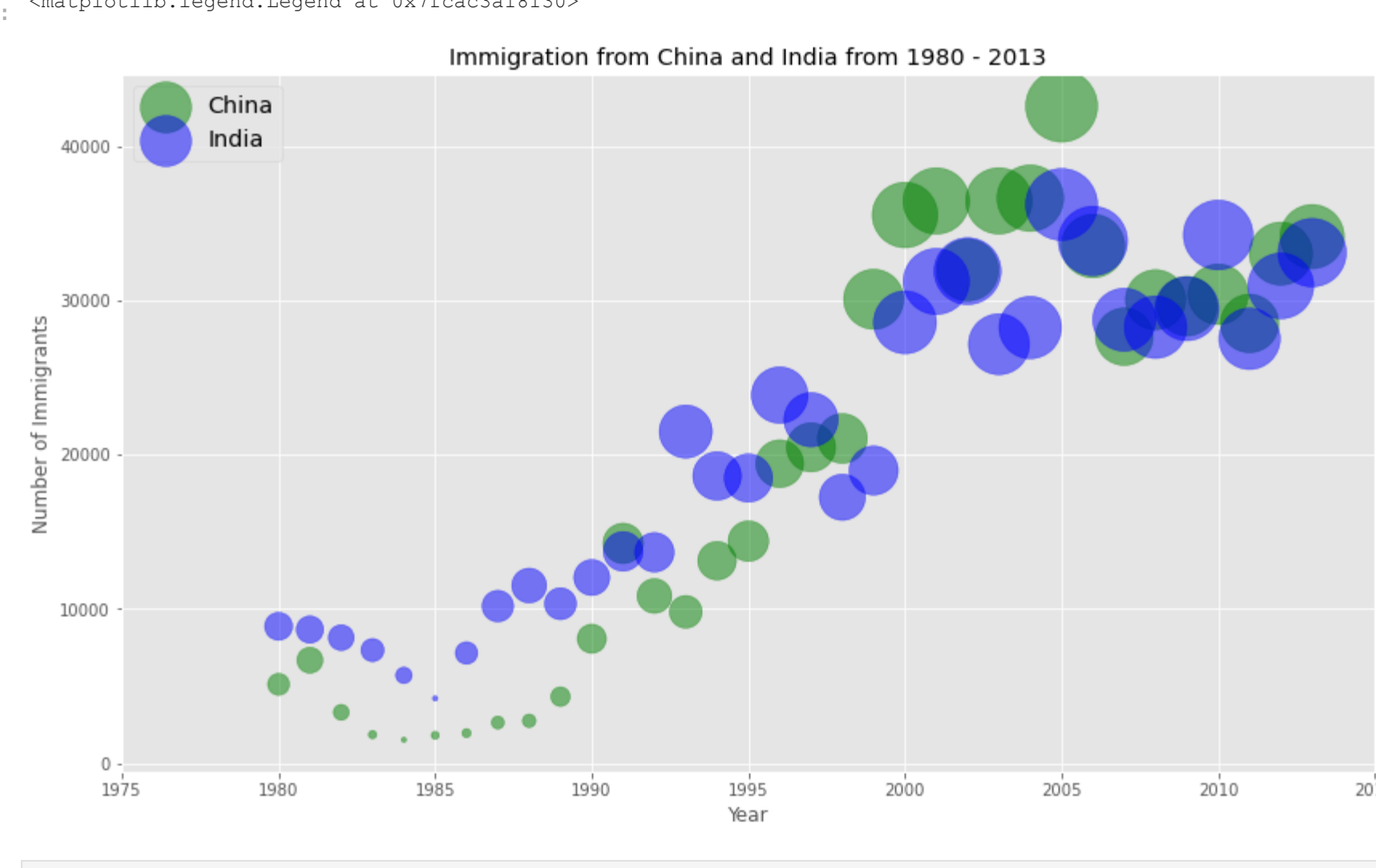
Out[86]:
```
<matplotlib.legend.Legend at 0x7fcac2d9b520>
```



In [87]:
```python
#Bubble plots of immigration from China and India to visualize differences with time from 1980 to 2013.
#using df_can_t that we defined and used in the previous example (box plot).

# normalized Chinese data
norm_china = (df_can_t['China'] - df_can_t['China'].min()) / (df_can_t['China'].max() - df_can_t['China'].min()
# normalized Indian data
norm_india = (df_can_t['India'] - df_can_t['India'].min()) / (df_can_t['India'].max() - df_can_t['India'].min()
```

In [89]:
```python
# 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 - 2013')
ax0.legend(['China', 'India'], loc='upper left', fontsize='x-large')
```

Out[89]:
```
<matplotlib.legend.Legend at 0x7fcac3a18130>
```



In [ ]: