In [2]:
```
from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas #
 import figurecanvas, agg stands for anti-grain-geometry
from matplotlib.figure import Figure # import Figure artisit
fig = Figure()
canvas = FigureCanvas(fig)
```

In [5]:
```
import os
```

In [8]:
```
os.getcwd()
```

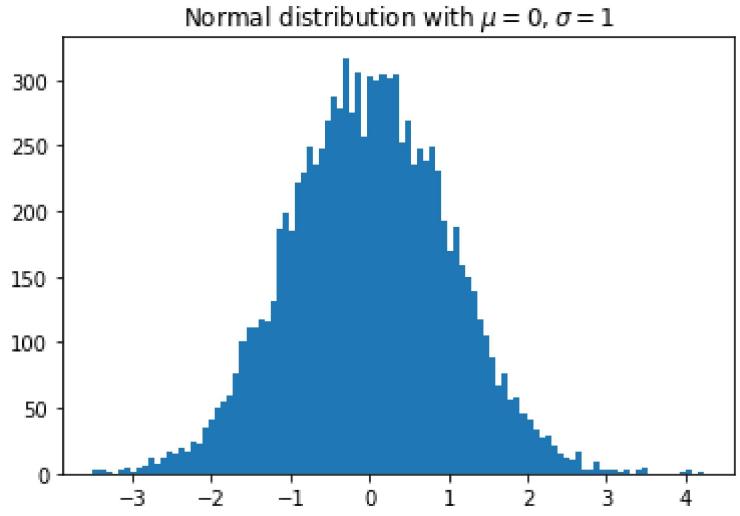
Out[8]:
```
'C:\\Users\\hssai'
```

In [10]:
```
os.chdir('C:\\Users\\hssai\\Desktop\\data visualization')
```

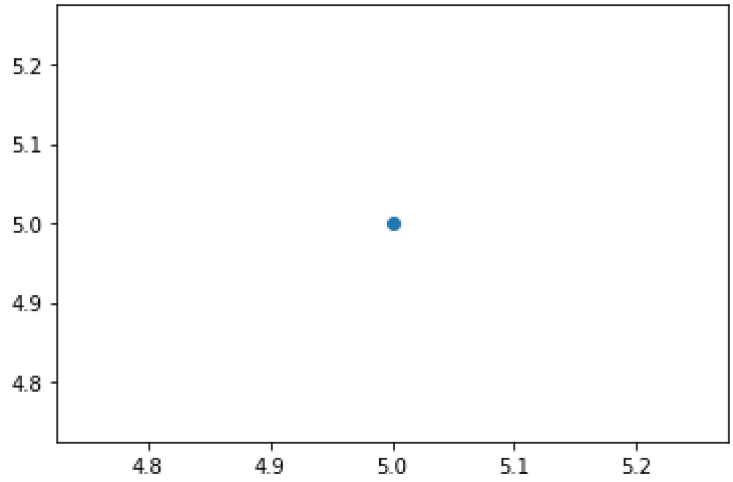
In [11]:
```
# create 10000 random numbers using numpy

import numpy as np
x = np.random.randn(10000)

ax = fig.add_subplot(111) #create an axes artist

ax.hist(x, 100) #generate a histogram of the 10000 numbers

# add a title to the figure and save it
ax.set_title('Normal distribution with $\mu=0, \sigma=1$')
fig.savefig('matplotlib_histogram.png')
```

```
C:\Users\hssai\anaconda3\lib\site-packages\ipykernel_launcher.py:6: MatplotlibDeprecationWarning: Adding an axes using the same arguments as a previous axes currently reuses the earlier instance.  In a future version, a new instance will always be created and returned.  Meanwhile, this warning can be suppressed, and the future behavior ensured, by passing a unique label to each axes instance.
```

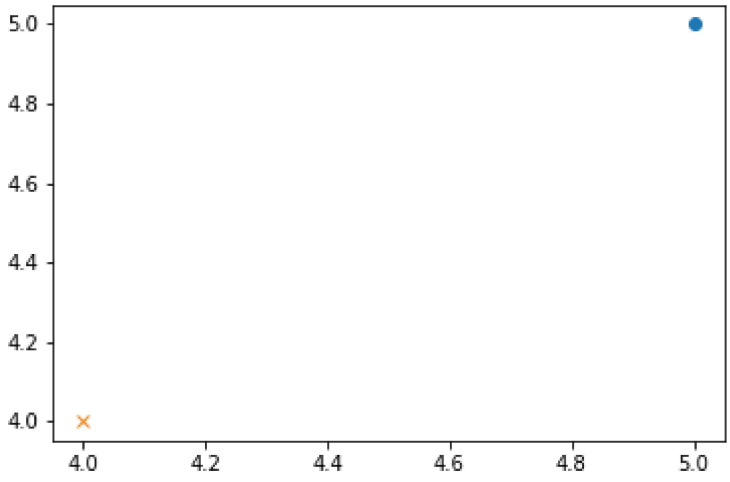
In [12]:
```
import matplotlib.pyplot as plt
import numpy as np
# create 10000 random numbers using numpy
x = np.random.randn(10000)

plt.hist(x, 100) #generate a histogram of the 10000 numbers

plt.title(r'Normal distribution with $\mu=0, \sigma=1$')
plt.savefig('matplotlib_histogram.png')
plt.show()
```



In [14]:
```
#to show the plots in the same place
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(5, 5, 'o')
plt.show()
```



In [173]:
```
# notebook backend, check if an active figure exists.
%matplotlib notebook
import matplotlib.pyplot as plt

plt.plot(5, 5, 'o')
plt.plot(4,4,'x')
```



Out[173]:
```
[<matplotlib.lines.Line2D at 0x1aa9318d3c8>]
```

I. Pandas Basic

Using Pandas to read daa into Pandas Dataframe

In [27]:
```
import numpy as np # useful for many scientific computing in python
import pandas as pd  # primary data structure library
#from __future__ import print_function # adds compatibility to python2
```

In [28]:
```
!pip install xlrd
```

```
Requirement already satisfied: xlrd in c:\users\hssai\anaconda3\lib\site-pack
ages (1.2.0)
```

In [617]:
```
# read the data into a pandas dataframe
df_canada = pd.read_excel(
                      'https://s3-api.us-geo.objectstorage.softlayer.net/cf-cou
rses-data/CognitiveClass/DV0101EN/labs/Data_Files/Canada.xlsx',
                       sheet_name='Canada by Citizenship',
                       skiprows = range(20),
                       skip_footer = 2)
```

In [618]:
```
df_canada.head()
```

Out[618]:

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980
0	Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	16
1	Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	1
2	Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	80
3	Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	0
4	Immigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions	0

5 rows × 43 columns

In [353]:
```
df_canada.tail()
```

Out[353]:

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1
192	Immigrants	Foreigners	Yemen	935	Asia	922	Western Asia	902	Developing regions	
193	Immigrants	Foreigners	Zambia	903	Africa	910	Eastern Africa	902	Developing regions	
194	Immigrants	Foreigners	Zimbabwe	903	Africa	910	Eastern Africa	902	Developing regions	
195	Immigrants	Foreigners	Unknown	999	World	999	World	999	World	44
196	Immigrants	Both	Total	999	World	999	World	999	World	143

5 rows × 43 columns

In [354]:
```
# to get the basic information
df_canada.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 197 entries, 0 to 196
Data columns (total 43 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Type      197 non-null    object
 1   Coverage  197 non-null    object
 2   OdName    197 non-null    object
 3   AREA      197 non-null    int64 
 4   AreaName  197 non-null    object
 5   REG       197 non-null    int64 
 6   RegName   197 non-null    object
 7   DEV       197 non-null    int64 
 8   DevName   197 non-null    object
 9   1980      197 non-null    int64 
 10  1981      197 non-null    int64 
 11  1982      197 non-null    int64 
 12  1983      197 non-null    int64 
 13  1984      197 non-null    int64 
 14  1985      197 non-null    int64 
 15  1986      197 non-null    int64 
 16  1987      197 non-null    int64 
 17  1988      197 non-null    int64 
 18  1989      197 non-null    int64 
 19  1990      197 non-null    int64 
 20  1991      197 non-null    int64 
 21  1992      197 non-null    int64 
 22  1993      197 non-null    int64 
 23  1994      197 non-null    int64 
 24  1995      197 non-null    int64 
 25  1996      197 non-null    int64 
 26  1997      197 non-null    int64 
 27  1998      197 non-null    int64 
 28  1999      197 non-null    int64 
 29  2000      197 non-null    int64 
 30  2001      197 non-null    int64 
 31  2002      197 non-null    int64 
 32  2003      197 non-null    int64 
 33  2004      197 non-null    int64 
 34  2005      197 non-null    int64 
 35  2006      197 non-null    int64 
 36  2007      197 non-null    int64 
 37  2008      197 non-null    int64 
 38  2009      197 non-null    int64 
 39  2010      197 non-null    int64 
 40  2011      197 non-null    int64 
 41  2012      197 non-null    int64 
 42  2013      197 non-null    int64 
dtypes: int64(37), object(6)
memory usage: 66.3+ KB
```

In [355]:
```
# to get the list of column headers:
df_canada.columns.values
```

Out[355]:
```
array(['Type', 'Coverage', 'OdName', 'AREA', 'AreaName', 'REG', 'RegName',
       'DEV', 'DevName', 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], dtype=object)
```

In [356]:
```
# get the list of indices
df_canada.index.values
```

Out[356]:
```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196], dtype=int64)
```

In [357]:
```
# to get the size of Dataframe (rows, columns)
df_canada.shape
```

Out[357]:
```
(197, 43)
```

In [358]:
```
# drop the area and coverage columns
df_canada.drop(['AREA', 'Coverage'], axis=1, inplace = True)
```

In [359]:
```
# Rename oldname and regname columns
df_canada.rename(columns={'OdName':'Country', 'RegName':'Region'}, inplace = T
rue)
```

In [360]:
```
# sums up the total immigrants by country
df_canada['Total'] = df_canada.sum(axis=1)
```

In [361]:
```
# check nulls objects
df_canada.isnull().sum()
```

Out[361]:
```
Type        0
Country     0
AreaName    0
REG         0
Region      0
DEV         0
DevName     0
1980        0
1981        0
1982        0
1983        0
1984        0
1985        0
1986        0
1987        0
1988        0
1989        0
1990        0
1991        0
1992        0
1993        0
1994        0
1995        0
1996        0
1997        0
1998        0
1999        0
2000        0
2001        0
2002        0
2003        0
2004        0
2005        0
2006        0
2007        0
2008        0
2009        0
2010        0
2011        0
2012        0
2013        0
Total       0
dtype: int64
```

In [362]:
```
# view a quick summary of each column
df_canada.describe()
```

Out[362]:

	REG	DEV	1980	1981	1982	1983	
count	197.000000	197.000000	197.000000	197.000000	197.000000	197.000000	1
mean	1246.477157	902.741117	1453.167513	1306.000000	1230.203046	905.431472	8
std	1179.730385	9.782793	10784.524807	9449.373841	8864.905615	6503.149859	64
min	905.000000	901.000000	0.000000	0.000000	0.000000	0.000000	
25%	914.000000	902.000000	0.000000	0.000000	0.000000	0.000000	
50%	922.000000	902.000000	14.000000	10.000000	12.000000	12.000000	
75%	926.000000	902.000000	266.000000	299.000000	299.000000	197.000000	2
max	5501.000000	999.000000	143137.000000	128641.000000	121175.000000	89185.000000	882

8 rows × 37 columns

In [363]:
```
## Indexing and Selection (slicing)

# return a serie of countries
df_canada.Country
```

Out[363]:
```
0          Afghanistan
1              Albania
2              Algeria
3       American Samoa
4              Andorra
             ...
192              Yemen
193             Zambia
194           Zimbabwe
195            Unknown
196              Total
Name: Country, Length: 197, dtype: object
```

In [364]:
```
# return a dataframe
df_canada[['Country', 1980, 1981, 1982, 1983, 1984, 1985]]
```

Out[364]:

	Country	1980	1981	1982	1983	1984	1985
0	Afghanistan	16	39	39	47	71	340
1	Albania	1	0	0	0	0	0
2	Algeria	80	67	71	69	63	44
3	American Samoa	0	1	0	0	0	0
4	Andorra	0	0	0	0	0	0
...	...	...	...	...	...	...	...
192	Yemen	1	2	1	6	0	18
193	Zambia	11	17	11	7	16	9
194	Zimbabwe	72	114	102	44	32	29
195	Unknown	44000	18078	16904	13635	14855	14368
196	Total	143137	128641	121175	89185	88272	84346

197 rows × 7 columns

In [365]:
```
df_canada.set_index('Country', inplace= True)
```

In [366]:
```
df_canada.head()
```

Out[366]:

Country	Type	AreaName	REG	Region	DEV	DevName	1980	1981	1982	1983	.
Afghanistan	Immigrants	Asia	5501	Southern Asia	902	Developing regions	16	39	39	47	.
Albania	Immigrants	Europe	925	Southern Europe	901	Developed regions	1	0	0	0	.
Algeria	Immigrants	Africa	912	Northern Africa	902	Developing regions	80	67	71	69	.
American Samoa	Immigrants	Oceania	957	Polynesia	902	Developing regions	0	1	0	0	.
Andorra	Immigrants	Europe	925	Southern Europe	901	Developed regions	0	0	0	0	.

5 rows × 41 columns

In [367]:
```
## get the full row data
print(df_canada.loc['Japan'])
```

```
Type                Immigrants
AreaName                  Asia
REG                        906
Region           Eastern Asia
DEV                        901
DevName     Developed regions
1980                       701
1981                       756
1982                       598
1983                       309
1984                       246
1985                       198
1986                       248
1987                       422
1988                       324
1989                       494
1990                       379
1991                       506
1992                       605
1993                       907
1994                       956
1995                       826
1996                       994
1997                       924
1998                       897
1999                      1083
2000                      1010
2001                      1092
2002                       806
2003                       817
2004                       973
2005                      1067
2006                      1212
2007                      1250
2008                      1284
2009                      1194
2010                      1168
2011                      1265
2012                      1214
2013                       982
Total                    29514
Name: Japan, dtype: object
```

In [368]:
```
# for year 2013
print(df_canada.loc['Japan', 2013])
```

```
982
```

In [369]:
```
# for years 1980 to 1985
print(df_canada.loc['Japan',[1980, 1981, 1982, 1983, 1984, 1985]])
print(df_canada.iloc[87,[3,4,5,6,7,8]])
```

```
1980    701
1981    756
1982    598
1983    309
1984    246
1985    198
Name: Japan, dtype: object
Region          Eastern Asia
DEV                      901
DevName    Developed regions
1980                     701
1981                     756
1982                     598
Name: Japan, dtype: object
```

In [370]:
```
# convert the column name into strings
df_canada.columns = list(map(str, df_canada.columns))
df_canada.columns
```

Out[370]:
```
Index(['Type', 'AreaName', 'REG', 'Region', 'DEV', 'DevName', '1980', '1981',
       '1982', '1983', '1984', '1985', '1986', '1987', '1988', '1989', '199
0',
       '1991', '1992', '1993', '1994', '1995', '1996', '1997', '1998', '199
9',
       '2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007', '200
8',
       '2009', '2010', '2011', '2012', '2013', 'Total'],
      dtype='object')
```

II. Matplotlib

Creating Line Plots

In [371]:
```
# we are using the inline backend
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
```

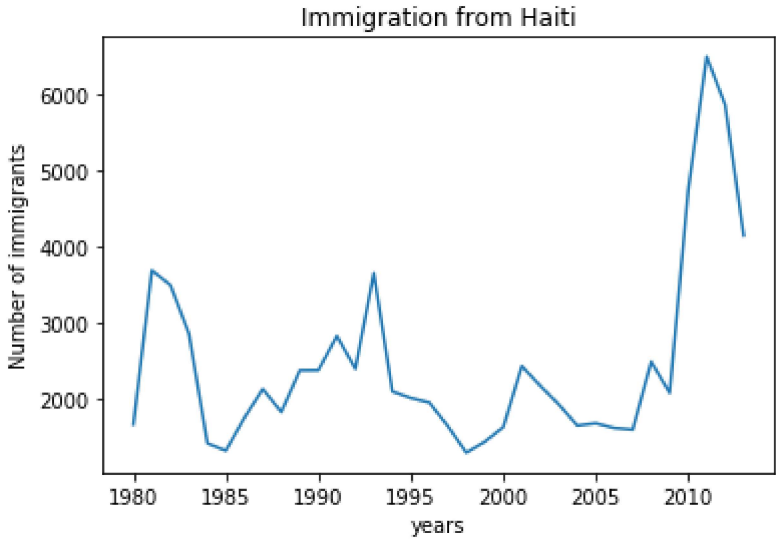
In [372]:
```
df_canada.head()
```

Out[372]:

Country	Type	AreaName	REG	Region	DEV	DevName	1980	1981	1982	1983	.
Afghanistan	Immigrants	Asia	5501	Southern Asia	902	Developing regions	16	39	39	47	.
Albania	Immigrants	Europe	925	Southern Europe	901	Developed regions	1	0	0	0	.
Algeria	Immigrants	Africa	912	Northern Africa	902	Developing regions	80	67	71	69	.
American Samoa	Immigrants	Oceania	957	Polynesia	902	Developing regions	0	1	0	0	.
Andorra	Immigrants	Europe	925	Southern Europe	901	Developed regions	0	0	0	0	.

5 rows × 41 columns

In [373]:
```
years = list(map(str,range(1980, 2014)))
df_canada.loc['Haiti', years].plot(kind  ='line')


plt.title('Immigration from Haiti')
plt.ylabel('Number of immigrants')
plt.xlabel('years')

plt.show()  #need this line to show the updates made to the figure
```

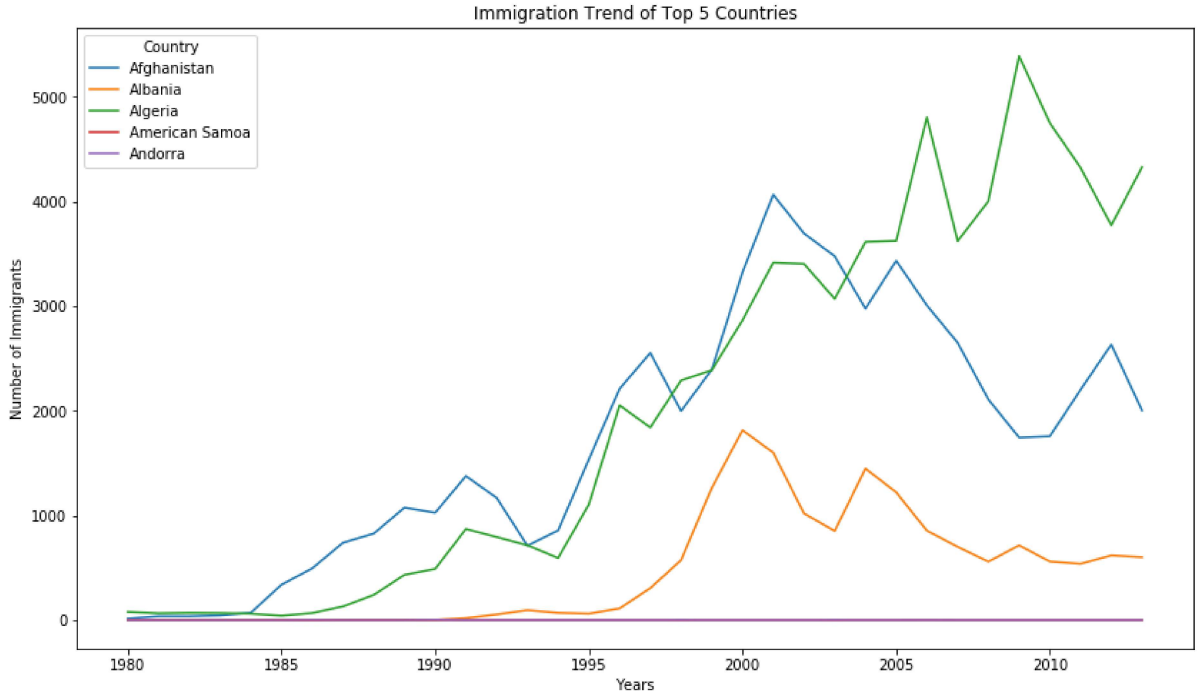


- compare the trend of top 5 countries that contributed the most to immigration to Canada

In [386]:

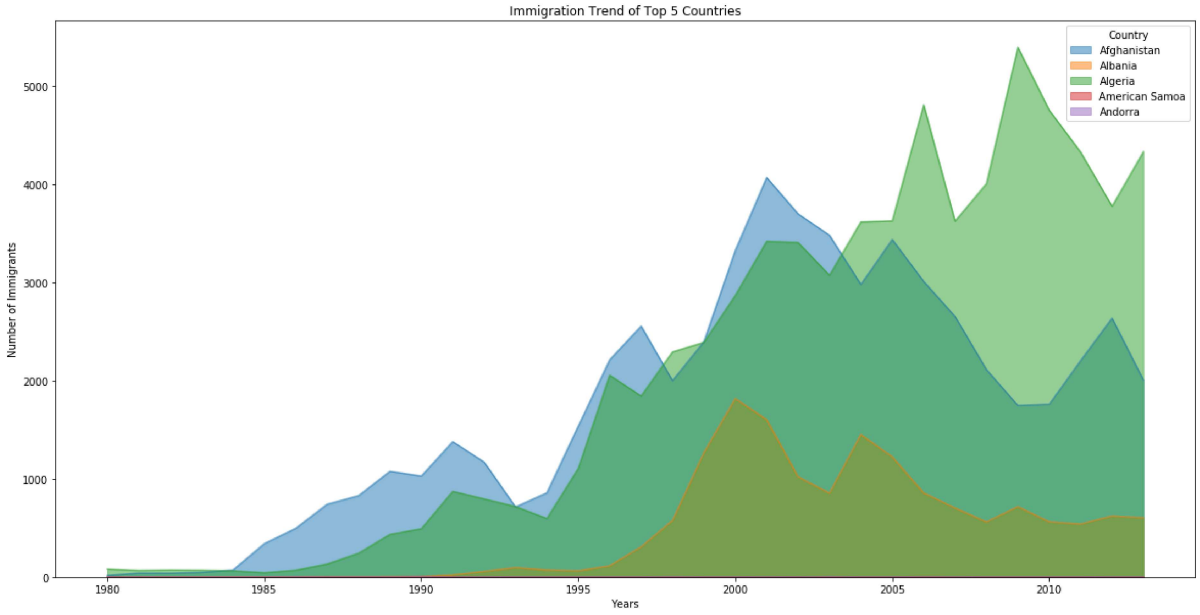
```
# get the top 5 entries

df_top5 = df_canada.head(5)
df_top5

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


df_top5.plot(kind = 'line', figsize = (14,8))

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

plt.show()
```

Country	Afghanistan	Albania	Algeria	American Samoa	Andorra
1980	16	1	80	0	0
1981	39	0	67	1	0
1982	39	0	71	0	0
1983	47	0	69	0	0
1984	71	0	63	0	0
1985	340	0	44	0	0
1986	496	1	69	0	2
1987	741	2	132	1	0
1988	828	2	242	0	0
1989	1076	3	434	1	0
1990	1028	3	491	2	3
1991	1378	21	872	0	0
1992	1170	56	795	0	1
1993	713	96	717	0	0
1994	858	71	595	0	0
1995	1537	63	1106	0	0
1996	2212	113	2054	0	0
1997	2555	307	1842	0	0
1998	1999	574	2292	0	2
1999	2395	1264	2389	0	0
2000	3326	1816	2867	0	0
2001	4067	1602	3418	0	1
2002	3697	1021	3406	0	0
2003	3479	853	3072	0	2
2004	2978	1450	3616	0	0
2005	3436	1223	3626	0	0
2006	3009	856	4807	1	1
2007	2652	702	3623	0	1
2008	2111	560	4005	0	0
2009	1746	716	5393	0	0
2010	1758	561	4752	0	0
2011	2203	539	4325	0	0
2012	2635	620	3774	0	1
2013	2004	603	4331	0	1



Creating an Area plots

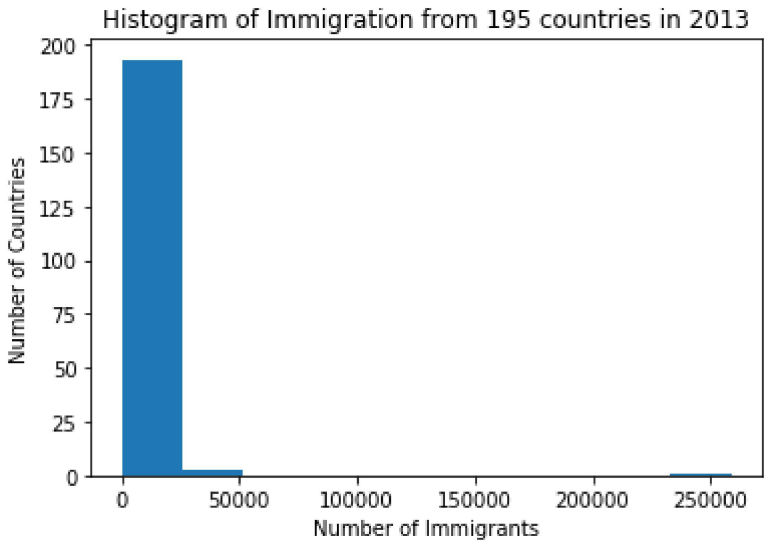
In [391]:
```
df_top5.index = df_top5.index.map(int)
df_top5.plot(kind = 'area', stacked = False, figsize=(20,10),)

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

plt.show()
```



Histograms

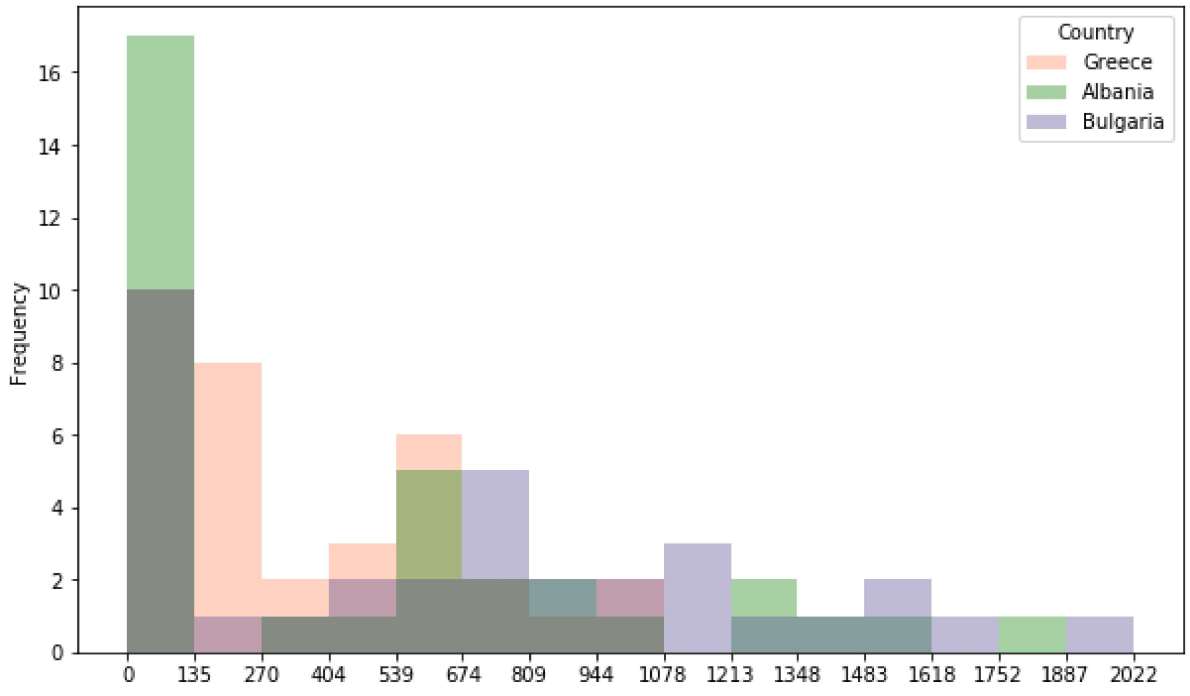
In [393]:
```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

In [394]:
```
# Histogram depict the distribution of immigration to Canada in 2013:

df_canada['2013']. plot(kind='hist')

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

plt.show()
```



- Display the immigration distribution for Greece, Albania, and Bulgaria for years 1980-2013 using an overlapping plot with 15 bins and a transparency value of 0.35

In [401]:
```
# create a dataframce of the countries of interest

df_cof = df_canada.loc[['Greece', 'Albania', 'Bulgaria'], years]

#transpose the dataframe

df_cof = df_cof.transpose()

# let's get the x=tick values

count, bin_edges = np.histogram(df_cof, 15)

# Un-stacked Histogram

df_cof.plot(kind = 'hist',
            figsize =(10,6),
            bins =15,
            alpha =0.35,
            xticks =bin_edges,
            color = ['coral', 'green', 'darkslateblue'])

plt.show()
```

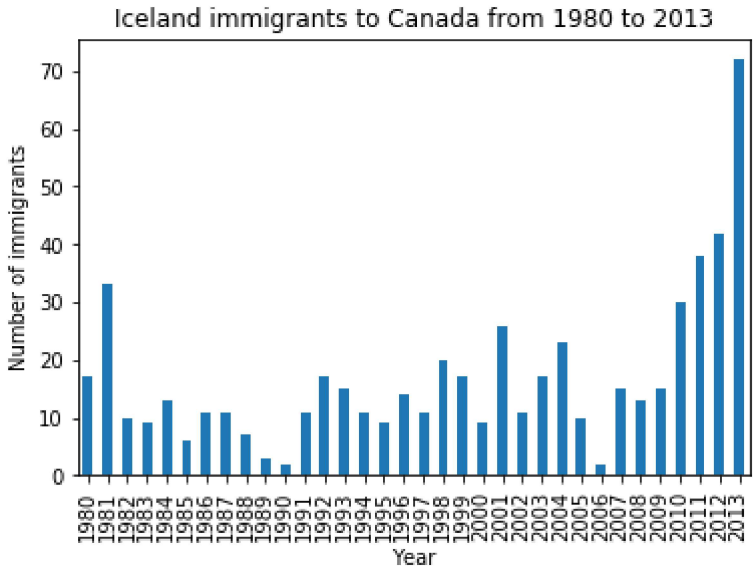


Bar Chart

- Unlik the histogram, bar chart is commonly used to compare the values of a variable at a given point in time

- visualizing in discrete fashion how immigration from Iceland to Canada looked like from 1980 to 2013.

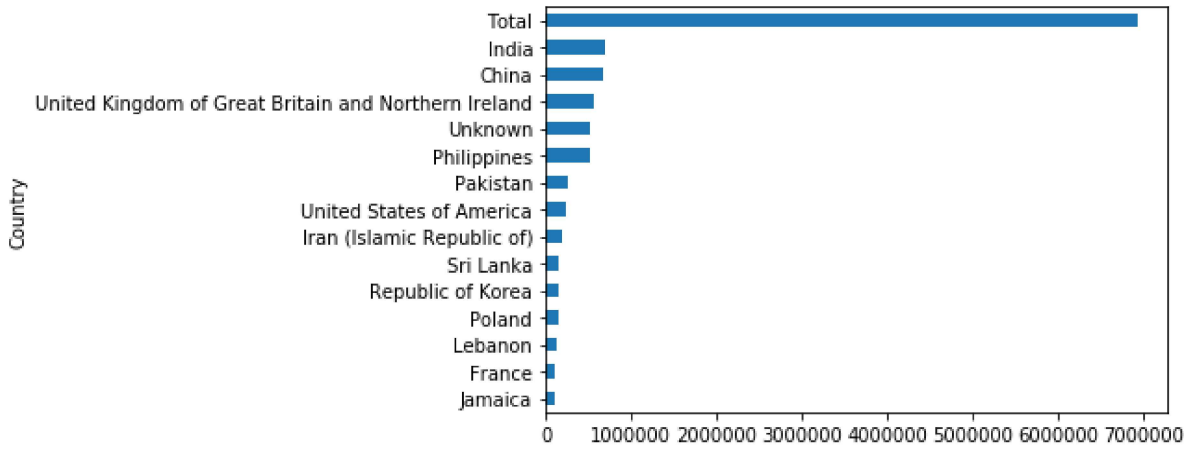
In [403]:
```
years = list(map(str, range(1980,2014)))

df_iceland = df_canada.loc['Iceland', years]

df_iceland.plot(kind = 'bar')
plt.title('Iceland immigrants to Canada from 1980 to 2013')
plt.xlabel('Year')
plt.ylabel('Number of immigrants')
plt.show()
```



- creating a horizontal bar plot showing the total number of immigrants to Canada from the top 15 countries, for the period 1980 -2013, label each country with the total immigrant count

In [409]:
```
# sort dataframe on 'Total' column in descending order
df_canada.sort_values(by = 'Total', ascending = True, inplace = True )

# get the top 15 countries
df_top15 = df_canada['Total'].tail(15)
df_top15

# generate plot
df_top15.plot(kind = 'barh')
```

Out[409]: <matplotlib.axes._subplots.AxesSubplot at 0x1aa93e46308>



Pie chart

- it's a statistical graphic divided into slices to illustrate numerical proportion.

In [411]:
```
df_canada.head()
```

Out[411]:

Country	Type	AreaName	REG	Region	DEV	DevName	1980	1981	1982	1983	...
Western Sahara	Immigrants	Africa	912	Northern Africa	902	Developing regions	0	0	0	0	...
Sao Tome and Principe	Immigrants	Africa	911	Middle Africa	902	Developing regions	0	0	0	0	...
Canada	Immigrants	Northern America	905	Northern America	901	Developed regions	0	0	0	0	...
San Marino	Immigrants	Europe	925	Southern Europe	901	Developed regions	1	0	0	0	...
New Caledonia	Immigrants	Oceania	928	Melanesia	902	Developing regions	0	0	0	0	...

5 rows × 41 columns

In [ ]:
```
df_canada.drop(['AREA', 'Coverage'], axis=1, inplace = True)
```

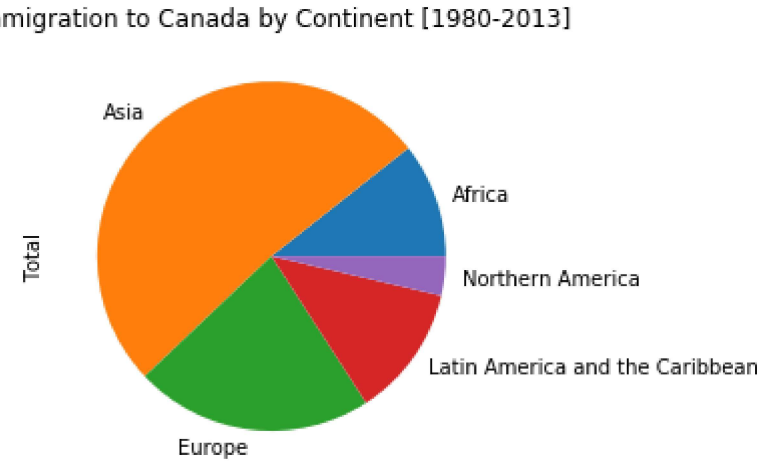
In [423]:
```
df_continents = df_canada.groupby('AreaName', axis=0).sum()
df_continents.drop(['World', 'Oceania'] , axis = 0 , inplace = True)
df_continents.drop(['REG', 'DEV'] , axis = 1 , inplace = True)

df_continents
```

Out[423]:

AreaName	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2005
Africa	3951	4363	3819	2671	2639	2650	3782	7494	7552	9894	...	27523
Asia	31025	34314	30214	24696	27274	23850	28739	43203	47454	60256	...	159253
Europe	39760	44802	42720	24638	22287	20844	24370	46698	54726	60893	...	35955
Latin America and the Caribbean	13081	15215	16769	15427	13678	15171	21179	28471	21924	25060	...	24747
Northern America	9378	10030	9074	7100	6661	6543	7074	7705	6469	6790	...	8394

5 rows × 35 columns

In [424]:
```
df_continents['Total'].plot(kind='pie')
plt.title('Immigration to Canada by Continent [1980-2013]')

plt.show()
```

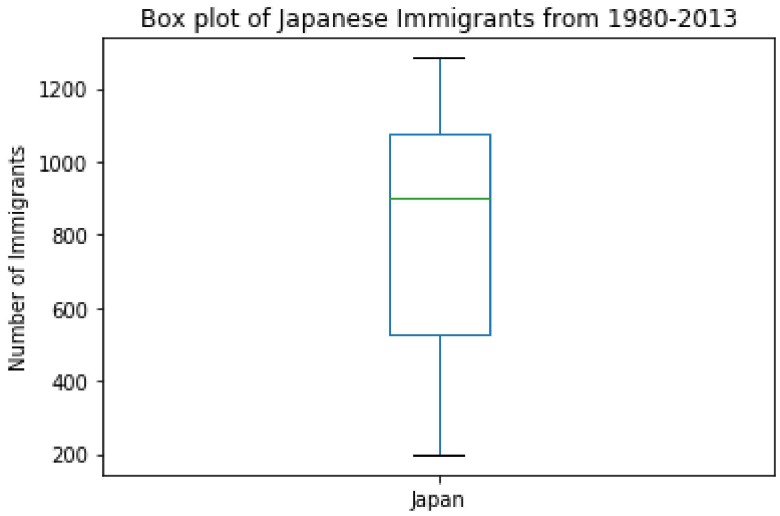


Box Plots

- boxplot is way of statistically representing the distribution of given data through 5 main dimensions

- creating a boxplot to visualize immigration from Japan to Canada

In [425]:
```
df_japan = df_canada.loc[['Japan'], years].transpose()

df_japan.plot(kind ='box')

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



Scatter Plots

- scatter plots is type of plot that displays values pertaining typically two variables against each other. Usually is a dependent variable to be plotted against an independent variable in order to determine if there's any correlation exist.

- plotting a scatter plot of the total annual immigration to Canada from 1980 to 2013.

In [466]:
```
df_total = df_canada.loc[['Total'], years].transpose()

# reset the indexes
df_total.reset_index(inplace=True)

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

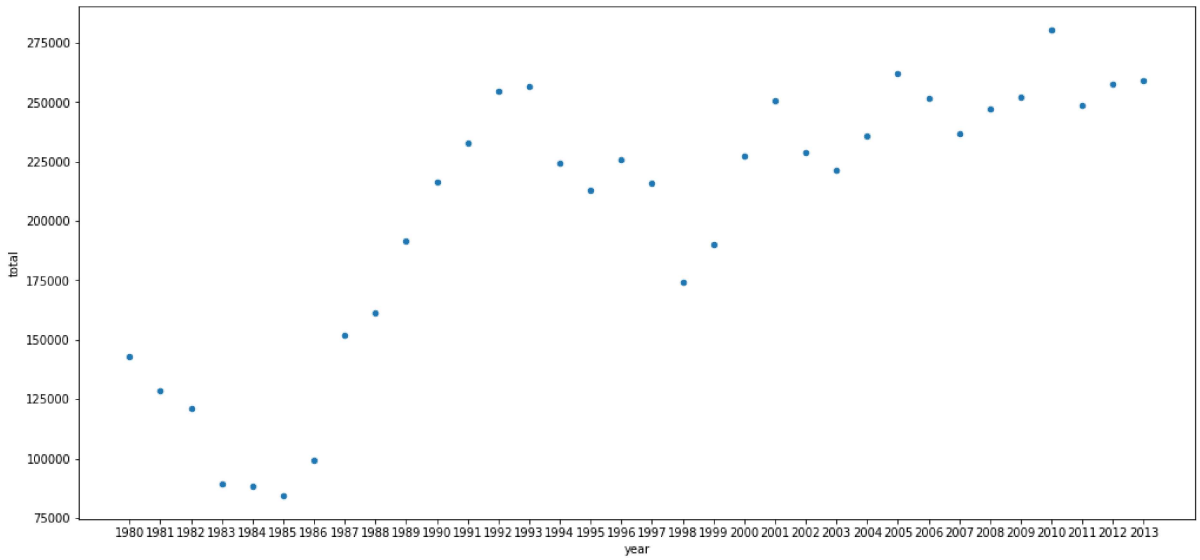
In [467]:
```
df_total
```

Out[467]:

	year	total
0	1980	143137
1	1981	128641
2	1982	121175
3	1983	89185
4	1984	88272
5	1985	84346
6	1986	99351
7	1987	152075
8	1988	161585
9	1989	191550
10	1990	216451
11	1991	232802
12	1992	254787
13	1993	256638
14	1994	224382
15	1995	212864
16	1996	226071
17	1997	216036
18	1998	174195
19	1999	189950
20	2000	227455
21	2001	250636
22	2002	229049
23	2003	221349
24	2004	235822
25	2005	262242
26	2006	251640
27	2007	236753
28	2008	247244
29	2009	252170
30	2010	280687
31	2011	248748
32	2012	257903
33	2013	259021

In [478]:
```
df_total.plot(kind ='scatter',
              x = 'year',
              y = 'total',figsize=(17, 8) )
```

Out[478]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa9445efc8>
```



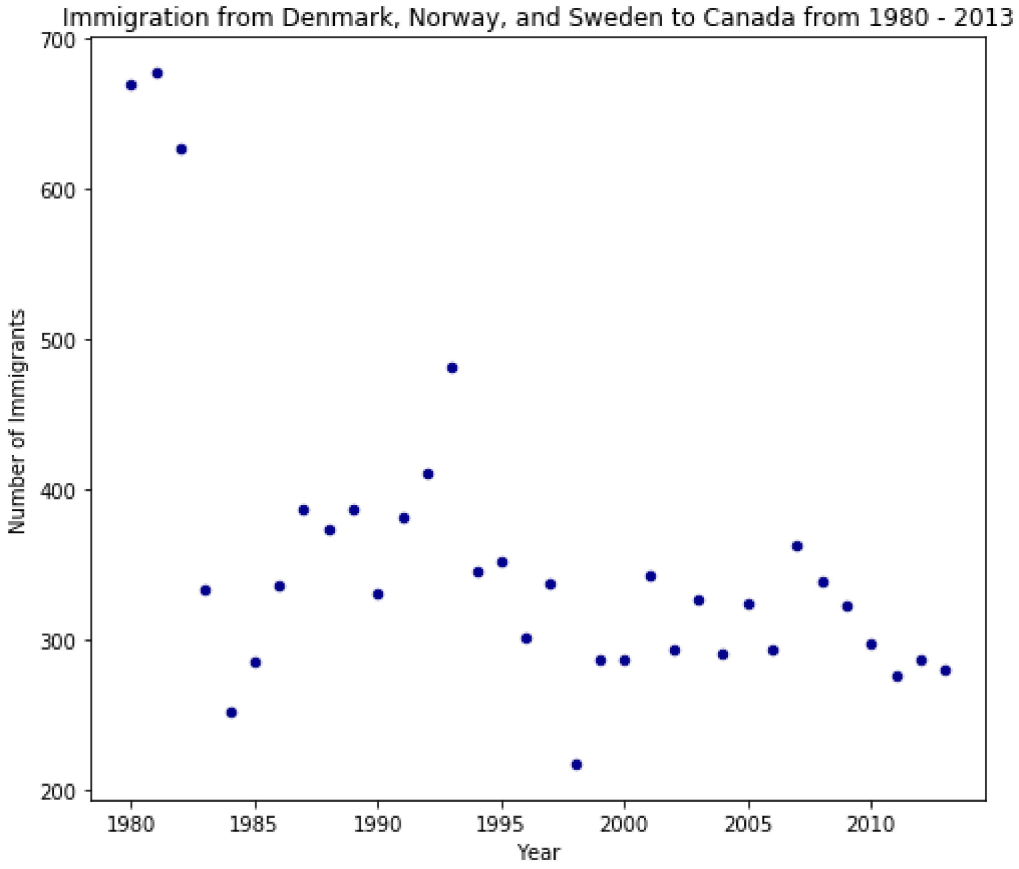
- Scatter plot showing the total immigration from Denmark, Norwar, and Sweden to Canada from 1980 to 2013

In [473]:
```
# 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_total1 = pd.DataFrame(df_countries.sum(axis=1))
# reset index in place
df_total1.reset_index(inplace=True)
# rename columns
df_total1.columns = ['year', 'total']
# change column year from string to int to create scatter plot
df_total1['year'] = df_total1['year'].astype(int)
# show resulting dataframe
df_total1.head()
```

Out[473]:

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

In [479]:
```
# generate scatter plot
df_total1.plot(kind='scatter', x='year', y='total', figsize=(8, 7), color='dar
kblue')
# add title and label to axes
plt.title('Immigration from Denmark, Norway, and Sweden to Canada from 1980 -
 2013')
plt.xlabel('Year')
plt.ylabel('Number of Immigrants')
# show plot
plt.show()
```

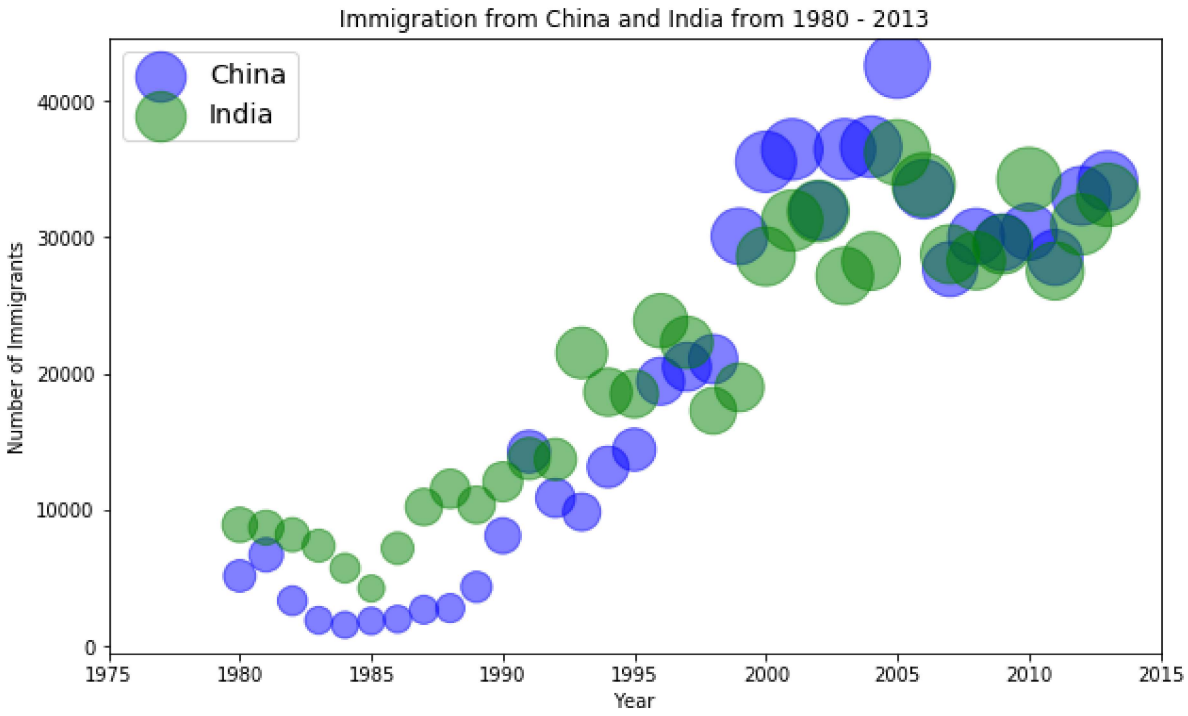


Bubble Plots

In [495]:

```
# transpose the dataframe to years
df_canada_t = df_canada[years].transpose()

# convert the years to type int
df_canada_t.index = map(int, df_canada_t.index)
df_canada_t.index

# label the index to years
df_canada_t.index.name = 'Year'
df_canada_t.index

# reset the index to bring the year as a column
df_canada_t.reset_index(inplace=True)

# view the changes
df_canada_t.head()


# normalize China data
normalize_china = (df_canada_t['China'] - df_canada_t['China'].min())/ (df_can
ada_t['China'].max() - df_canada_t['China'].min())

# normalize India data
#normalize_India = (df_canada_t['India'] - df_canada_t['India'].min())/ (df_ca
nada_t['India'].max() - df_canada_t['India'].min())


# china plot

ax0 = df_canada_t.plot(kind = 'scatter',
                       x = 'Year',
                       y = 'China',
                       figsize = (10,6),
                       alpha = 0.5,
                       color = 'Blue',
                      s = normalize_china * 1000 + 200,
                       xlim = (1975, 2015)
                       )

# India plot

ax1 = df_canada_t.plot(kind = 'scatter',
                       x = 'Year',
                       y = 'India',
                     alpha = 0.5,
                       color = 'Green',
                      s = normalize_India * 1000 + 200,
                       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[495]: <matplotlib.legend.Legend at 0x1aa97e90b48>



Word Clouds

- word cloud is a depiction of the frequency of different words in some textual data.

In [ ]:
```
# # install wordcloud
# !conda install -c conda-forge wordcloud==1.4.1 --yes

# import package and its set of stopwords
from wordcloud import WordCloud, STOPWORDS

print ('Wordcloud is installed and imported!')
```

In [630]:
```
df_canada.head()
```

Out[630]:

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980
0	Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	16
1	Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	1
2	Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	80
3	Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	0
4	Immigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions	0

5 rows × 43 columns

In [631]:
```
# create a word cloud object

text = " ".join(str(each) for each in df_canada.RegName)
alice_wc = WordCloud(
             background_color = 'white',
             max_words =2000,
            # stopwords=stopwords
)

# generate the word cloud
alice_wc.generate(text)
```

Out[631]:
```
<wordcloud.wordcloud.WordCloud at 0x1aa98e001c8>
```

In [635]:
```
# visualize the wordcloud created

fig = plt.figure()
fig.set_figwidth(14)  #set the width
fig.set_figheight(18) # set the height

plt.imshow(alice_wc, interpolation ='bilinear')

plt.axis('off')

plt.show()
```

Seaborn and Regression Plots

- seaborn is a python visualization library based on Matplotlib

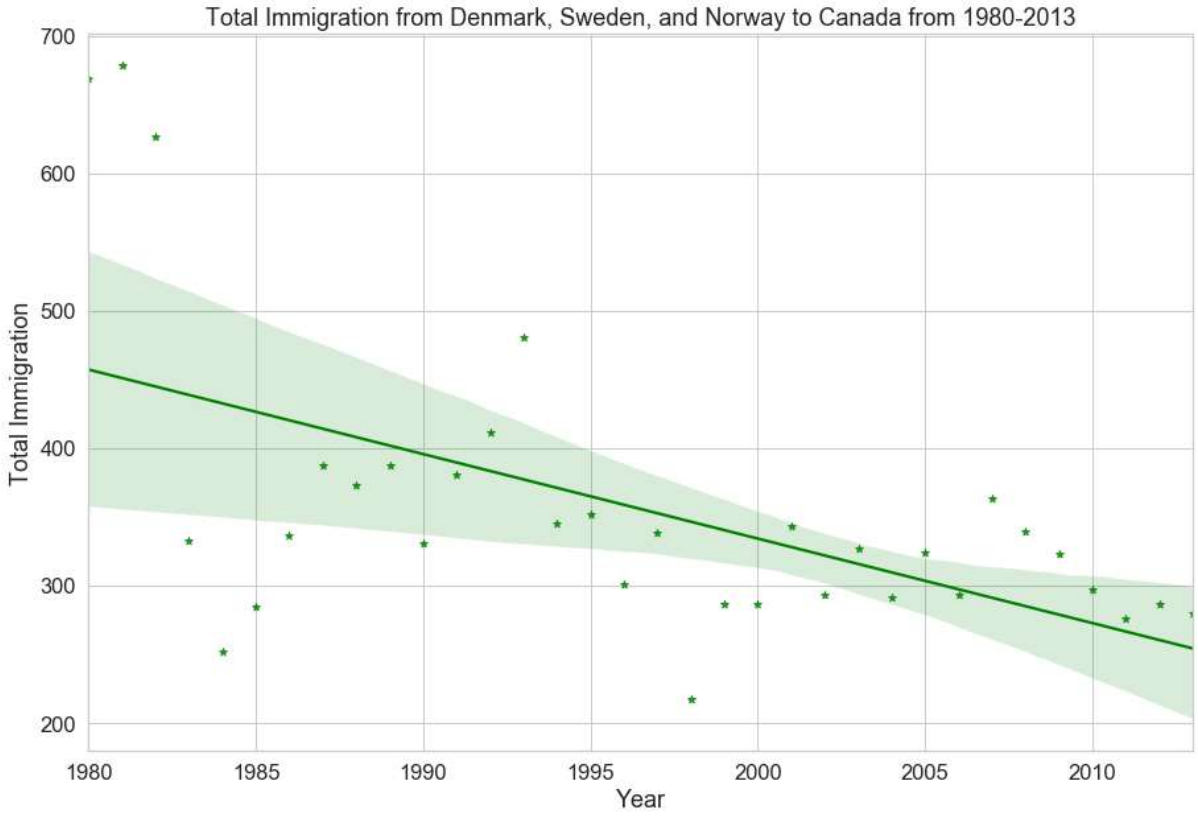
- use seaborn to create a scatter plot with a regression line to visualize the total immigration from Denmark, Sweden, and Norway to Canada from 1980 to 2013.

In [501]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [512]:

```
# create a df_countries dataframe

df_countries = df_canada.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 the column year from string to int to create scatter plot

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

# define figure size

plt.figure(figsize=(15,10))

# define background style  and front size

sns.set(font_scale =1.5)
sns.set_style('whitegrid')

# generate plot and add title and axes labels

ax = sns.regplot(x='year', y ='total', data = df_total, color ='green', marker
= '*')
ax.set(xlabel = 'Year', ylabel = 'Total Immigration')
ax.set_title('Total Immigration from Denmark, Sweden, and Norway to Canada fro
m 1980-2013')
```

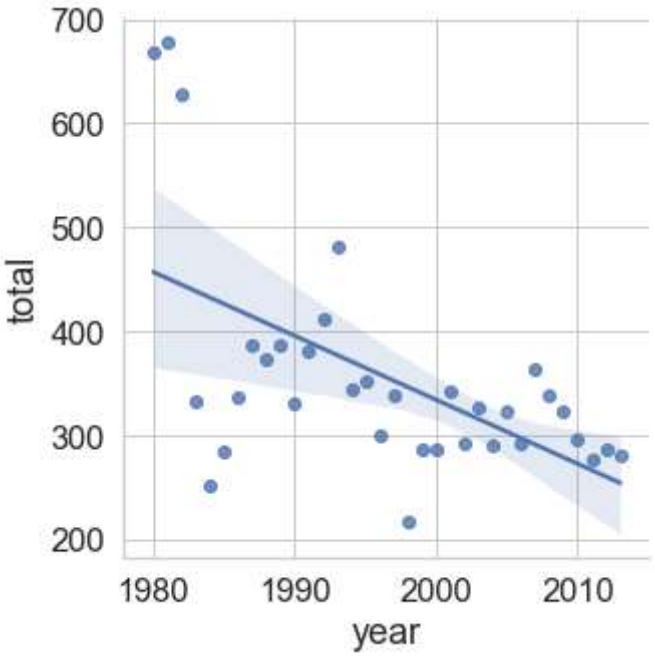
Out[512]: Text(0.5, 1.0, 'Total Immigration from Denmark, Sweden, and Norway to Canada from 1980-2013')



- create a scatter plot along with a regression line to highlight any trends in the data.

In [513]:
```
df_total['year'] = df_total['year'].astype(int)

sns.lmplot(x = 'year', y = 'total', data = df_total)
```

Out[513]: <seaborn.axisgrid.FacetGrid at 0x1aa97e99ac8>



III. Generating Maps

- Folium is a powerful python library that helps create several types of Leaflet maps.
- it enables the binding of data to a map for choropleth visualization as well as passing visualization as markers on the map.
- with Folium, you can create maps of different styles such as street level map, stamen map, Mapbox, and supports custom tilesets with Mapbox API keys.

- A Choropleth map is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable being displayed on the map, such as population density or per-capita income. The choropleth map provides an easy way to visualize how a measurement varies across a geographic area or it shows the level of variability within a region.

- Generate a choropleth map of the world showing immigration to Canada.

In [518]:
```
!pip install folium
```

```
Collecting folium
  Downloading folium-0.12.1-py2.py3-none-any.whl (94 kB)
Requirement already satisfied: numpy in c:\users\hssai\anaconda3\lib\site-pac
kages (from folium) (1.16.6)
Collecting branca>=0.3.0
  Downloading branca-0.4.2-py3-none-any.whl (24 kB)
Requirement already satisfied: requests in c:\users\hssai\anaconda3\lib\site-
packages (from folium) (2.18.4)
Requirement already satisfied: jinja2>=2.9 in c:\users\hssai\anaconda3\lib\si
te-packages (from folium) (2.11.1)
Requirement already satisfied: idna<2.7,>=2.5 in c:\users\hssai\anaconda3\lib
\site-packages (from requests->folium) (2.6)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\hssai\anacon
da3\lib\site-packages (from requests->folium) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hssai\anaconda3
\lib\site-packages (from requests->folium) (2019.11.28)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in c:\users\hssai\anacon
da3\lib\site-packages (from requests->folium) (1.22)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\hssai\anaconda3\l
ib\site-packages (from jinja2>=2.9->folium) (1.1.1)
Installing collected packages: branca, folium
Successfully installed branca-0.4.2 folium-0.12.1
```

In [539]:
```
!wget --quiet https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-dat
a/CognitiveClass/DV0101EN/labs/Data_Files/world_countries.json -O world_countr
ies.json
print('GeoJSON file downloaded!')
```

```
GeoJSON file downloaded!

SYSTEM_WGETRC = c:/progra~1/wget/etc/wgetrc
syswgetrc = C:\Program Files (x86)\Gow/etc/wgetrc
```

In [638]:
```
# clean up the dataset to remove unnecessary columns (eg. REG)
df_canada.drop(['AREA','RegName','DevName','Type','Coverage'], axis=1, inplace
=True)
# let's rename the columns so that they make sense
df_canada.rename(columns={'OdName':'Country', 'AreaName':'Continent','RegName'
:'Region'}, inplace=True)

# for the consistency, let's cast all column labels of type string

df_canada.columns = list(map(str, df_canada.columns))

# add total column

df_canada['Total'] = df_canada.sum(axis=1)




df_canada.head()
```

Out[638]:

	Country	Continent	REG	DEV	1980	1981	1982	1983	1984	1985	...	2005	2006	2007
0	Afghanistan	Asia	5501	902	16	39	39	47	71	340	...	3436	3009	2652
1	Albania	Europe	925	901	1	0	0	0	0	0	...	1223	856	702
2	Algeria	Africa	912	902	80	67	71	69	63	44	...	3626	4807	3623
3	American Samoa	Oceania	957	902	0	1	0	0	0	0	...	0	1	0
4	Andorra	Europe	925	901	0	0	0	0	0	0	...	0	1	1

5 rows × 39 columns

In [639]:
```
import folium
```

In [640]:
```
# GeoJson is used to define the boundaries of all world countries.
# download countries geojson file
!wget --quiet https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-dat
a/CognitiveClass/DV0101EN/labs/Data_Files/world_countries.json -O world_countr
ies.json

print('GeoJSON file downloaded!')
```

```
GeoJSON file downloaded!

SYSTEM_WGETRC = c:/progra~1/wget/etc/wgetrc
syswgetrc = C:\Program Files (x86)\Gow/etc/wgetrc
```

In [641]:
```
world_geo = r'world_countries.json' # geojson file

    # create a plain world map
world_map = folium.Map(location=[0, 0], zoom_start=2, tiles='cartodbpositron')
world_map
```

Out[641]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [643]:
```
# generate choropleth map using the total immigration of each country to Canad
a from 1980 to 2013
#world_map.choropleth(
#    geo_data=world_geo,
#    data=df_canada,
#    columns=['Country', 'Total'],
#    key_on='feature.properties.name',
#    fill_color= 'YlOrRd',
#    fill_opacity=0.7,
#    line_opacity=0.2,
#    legend_name= 'Immigration to Canada')

# display map

#world_map
```

In [ ]:
```

```