Experiment II [1]

August 17, 2025

[32]:
```
import numpy as np  # useful for many scientific computing in Python
import pandas as pd # primary data structure library
df_can = pd.read_excel('https://s3-api.us-geo.objectstorage.softlayer.net/
 ↪cf-courses-data/CognitiveClass/DV0101EN/labs/Data_Files/Canada.xlsx',
                       sheet_name='Canada by Citizenship',
                       skiprows=range(20),
                       skipfooter=2)


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

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

[33]:
```
df_can.head()
```

[33]:

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980	…	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
0	Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	16	…	2978	3436	3009	2652	2111	1746	1758	2203	2635	2004
1	Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	1	…	1450	1223	856	702	560	716	561	539	620	603
2	Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	80	…	3616	3626	4807	3623	4005	5393	4752	4325	3774	4331
3	Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	0	…	0	0	1	0	0	0	0	0	0	0
4	Immigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions	0	…	0	0	1	1	0	0	0	0	1	1

[5 rows x 43 columns]

[34]:
```
df_can.columns.values
```

[34]:
```
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)
```

[35]:
```
print(type(df_can.columns))
print(type(df_can.index))
```

```
<class 'pandas.core.indexes.base.Index'>
<class 'pandas.core.indexes.range.RangeIndex'>
```

[36]:
```
df_can.columns.tolist()
df_can.index.tolist()

print (type(df_can.columns.tolist()))
print (type(df_can.index.tolist()))
```

```
<class 'list'>
<class 'list'>
```

[37]:
```
df_can.shape
```

[37]:
```
(195, 43)
```

[38]:
```
df_can.drop(['AREA','REG','DEV','Type','Coverage'], axis=1, inplace=True)
df_can.head(2)
```

[38]:

	OdName	AreaName	RegName	DevName	1980	1981	1982	1983	1984	1985	…	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	…	2978	3436	3009	2652	2111	1746	1758	2203	2635	2004
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	…	1450	1223	856	702	560	716	561	539	620	603

[2 rows x 38 columns]

[39]:
```
df_can.rename(columns={'OdName':'Country', 'AreaName':'Continent', 'RegName':
 ↪'Region'}, inplace=True)
df_can.columns
```

[39]:
```
Index([  'Country', 'Continent',    'Region',   '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')
```

[40]:
```
df_can['Total'] = df_can.apply(pd.to_numeric, errors='coerce').sum(axis=1)
```

[41]:
```
df_can.columns
```

[41]:
```
Index([  'Country', 'Continent',    'Region',   '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,     'Total'],
      dtype='object')
```

[42]:
```
df_can.isnull().sum()
```

[42]:
```
Country      0
Continent    0
Region       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
```

[43]:
```
df_can.describe()
```

[43]:

	1980	1981	1982	1983	1984	\
count	195.000000	195.000000	195.000000	195.000000	195.000000	
mean	508.394872	566.989744	534.723077	387.435897	376.497436	
std	1949.588546	2152.643752	1866.997511	1204.333597	1198.246371	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	13.000000	10.000000	11.000000	12.000000	13.000000	
75%	251.500000	295.500000	275.000000	173.000000	181.000000	
max	22045.000000	24796.000000	20620.000000	10015.000000	10170.000000	

	1985	1986	1987	1988	1989	\
count	195.000000	195.000000	195.000000	195.000000	195.000000	
mean	358.861538	441.271795	691.133333	714.389744	843.241026	
std	1079.309600	1225.576630	2109.205607	2443.606788	2555.048874	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.500000	0.500000	1.000000	1.000000	
50%	17.000000	18.000000	26.000000	34.000000	44.000000	
75%	197.000000	254.000000	434.000000	409.000000	508.500000	
max	9564.000000	9470.000000	21337.000000	27359.000000	23795.000000	

	…	2005	2006	2007	2008	\
count	…	195.000000	195.000000	195.000000	195.000000	
mean	…	1320.292308	1266.958974	1191.820513	1246.394872	
std	…	4425.957828	3926.717747	3443.542409	3694.573544	
min	…	0.000000	0.000000	0.000000	0.000000	

```
25%    …     28.500000     25.000000     31.000000     31.000000
50%    …    210.000000    218.000000    198.000000    205.000000
75%    …    832.000000    842.000000    899.000000    934.500000
max    …  42584.000000  33848.000000  28742.000000  30037.000000

                2009          2010          2011          2012          2013  \
count     195.000000    195.000000    195.000000    195.000000    195.000000
mean     1275.733333   1420.287179   1262.533333   1313.958974   1320.702564
std      3829.630424   4462.946328   4030.084313   4247.555161   4237.951988
min         0.000000      0.000000      0.000000      0.000000      0.000000
25%        36.000000     40.500000     37.500000     42.500000     45.000000
50%       214.000000    211.000000    179.000000    233.000000    213.000000
75%       888.000000    932.000000    772.000000    783.000000    796.000000
max     29622.000000  38617.000000  36765.000000  34315.000000  34129.000000

                Total
count      195.000000
mean     32867.451282
std      91785.498686
min          1.000000
25%        952.000000
50%       5018.000000
75%      22239.500000
max     691904.000000

[8 rows x 35 columns]
```

[44]:
```
df_can.Country
```

[44]:
```
0          Afghanistan
1              Albania
2              Algeria
3       American Samoa
4              Andorra
              …
190          Viet Nam
191     Western Sahara
192              Yemen
193             Zambia
194           Zimbabwe
Name: Country, Length: 195, dtype: object
```

[45]:
```
df_can[['Country', 1980, 1981, 1982, 1983, 1984, 1985]]
```

[45]:
```
              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
..              …   …   …   …   …   …   …
190        Viet Nam  1191  1829  2162  3404  7583  5907
191  Western Sahara     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 rows x 7 columns]
```

[46]:
```
df_can.set_index('Country', inplace=True)
```

[47]:
```
df_can.head(3)
```

[47]:
```
             Continent           Region             DevName  1980  1981  1982  \
Country
Afghanistan      Asia    Southern Asia  Developing regions    16    39    39
Albania        Europe  Southern Europe   Developed regions     1     0     0
Algeria        Africa  Northern Africa  Developing regions    80    67    71

             1983  1984  1985  1986  …  2005  2006  2007  2008  2009  2010  \
Country                              …
Afghanistan    47    71   340   496  …  3436  3009  2652  2111  1746  1758
Albania         0     0     0     1  …  1223   856   702   560   716   561
Algeria        69    63    44    69  …  3626  4807  3623  4005  5393  4752

             2011  2012  2013    Total
Country
Afghanistan  2203  2635  2004  58639.0
Albania       539   620   603  15699.0
Algeria      4325  3774  4331  69439.0

[3 rows x 38 columns]
```

[48]:
```
df_can.index.name = None  # optional: to remove the name of the index
```

[49]:
```
#To avoid this ambuigity, let's convert the column names into strings: '1980'␣
 ↪to '2013'.

df_can.columns = list(map(str, df_can.columns))
[print (type(x)) for x in df_can.columns.values] #<-- uncomment to check type␣
 ↪of column headers
```

```
<class 'str'>
<class 'str'>
<class 'str'>
```

```
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
```

[49]:
```
[None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
```

```
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None]
```

[50]:
```
# useful for plotting later on
years = list(map(str, range(1980, 2014)))
years
```

[50]:
```
['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']
```

[51]:
```
%matplotlib inline

import matplotlib as mpl
import matplotlib.pyplot as plt
```

[52]:
```
print ('Matplotlib version: ', mpl.__version__ )
```

```
Matplotlib version:  3.9.2
```

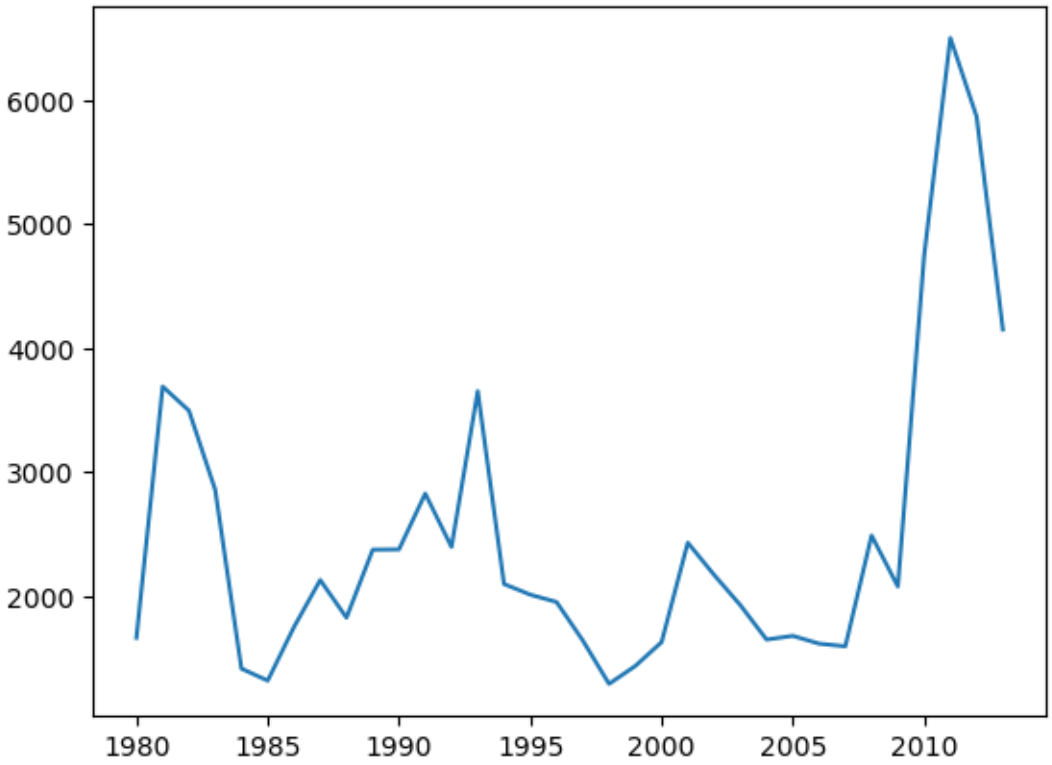
[53]:
```
haiti = df_can.loc['Haiti', years] # Passing in years 1980 - 2013 to exclude␣
 ↪the 'total' column
haiti.head()
```

[53]:
```
1980    1666
1981    3692
1982    3498
1983    2860
1984    1418
Name: Haiti, dtype: object
```

[54]:
```
haiti.plot()
```

[54]:
```
<Axes: >
```

[55]:
```
plt.show()
```

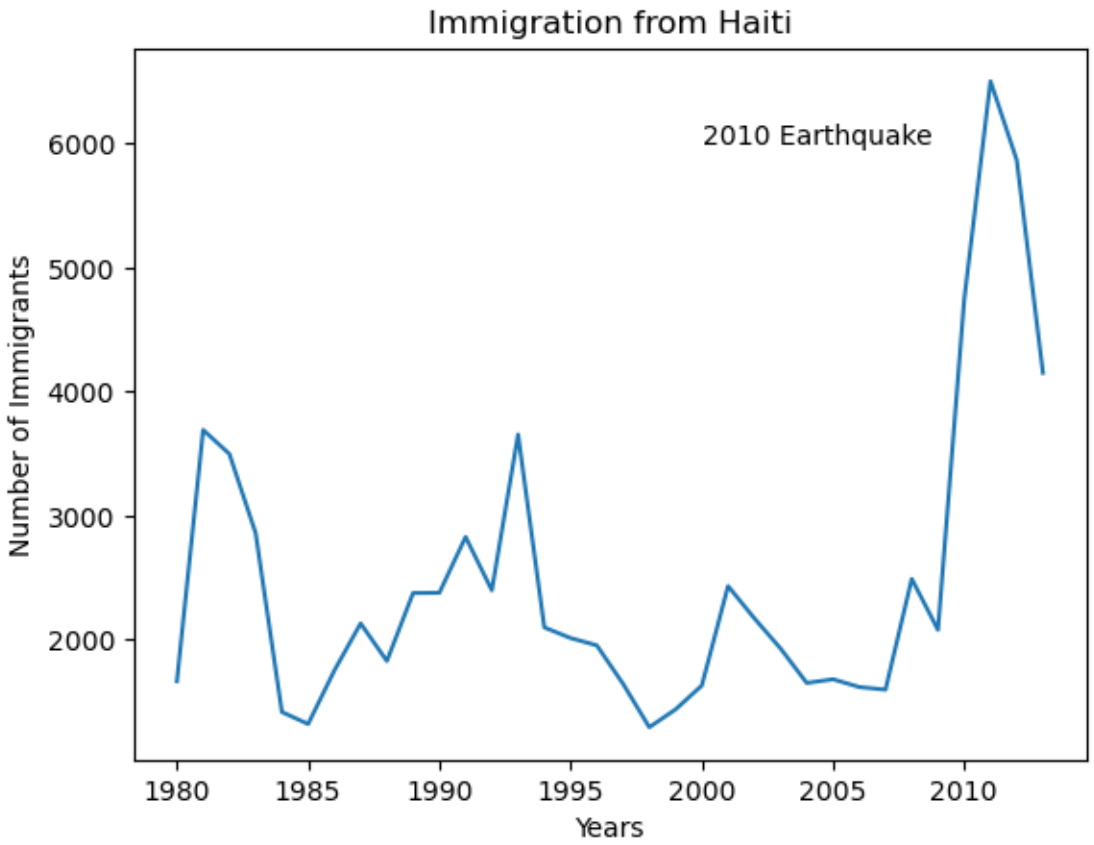


[56]:
```
haiti.plot(kind='line')

plt.title('Immigration from Haiti')
plt.ylabel('Number of Immigrants')
plt.xlabel('Years')

# annotate the 2010 Earthquake.
# syntax: plt.text(x, y, label)
plt.text(20, 6000, '2010 Earthquake') # see note below

plt.show()
```



[57]:
```
#QUESTION :: Let us compare the number of immigrants from India and China from
 ↪1980 to 2013.
#Step 1: Get the data set for China and India.
df_CI = df_can.loc[['India', 'China'], years]
df_CI.head()
```

[57]:

	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	…	\
India	8880	8670	8147	7338	5704	4211	7150	10189	11522	10343	…	
China	5123	6682	3308	1863	1527	1816	1960	2643	2758	4323	…	

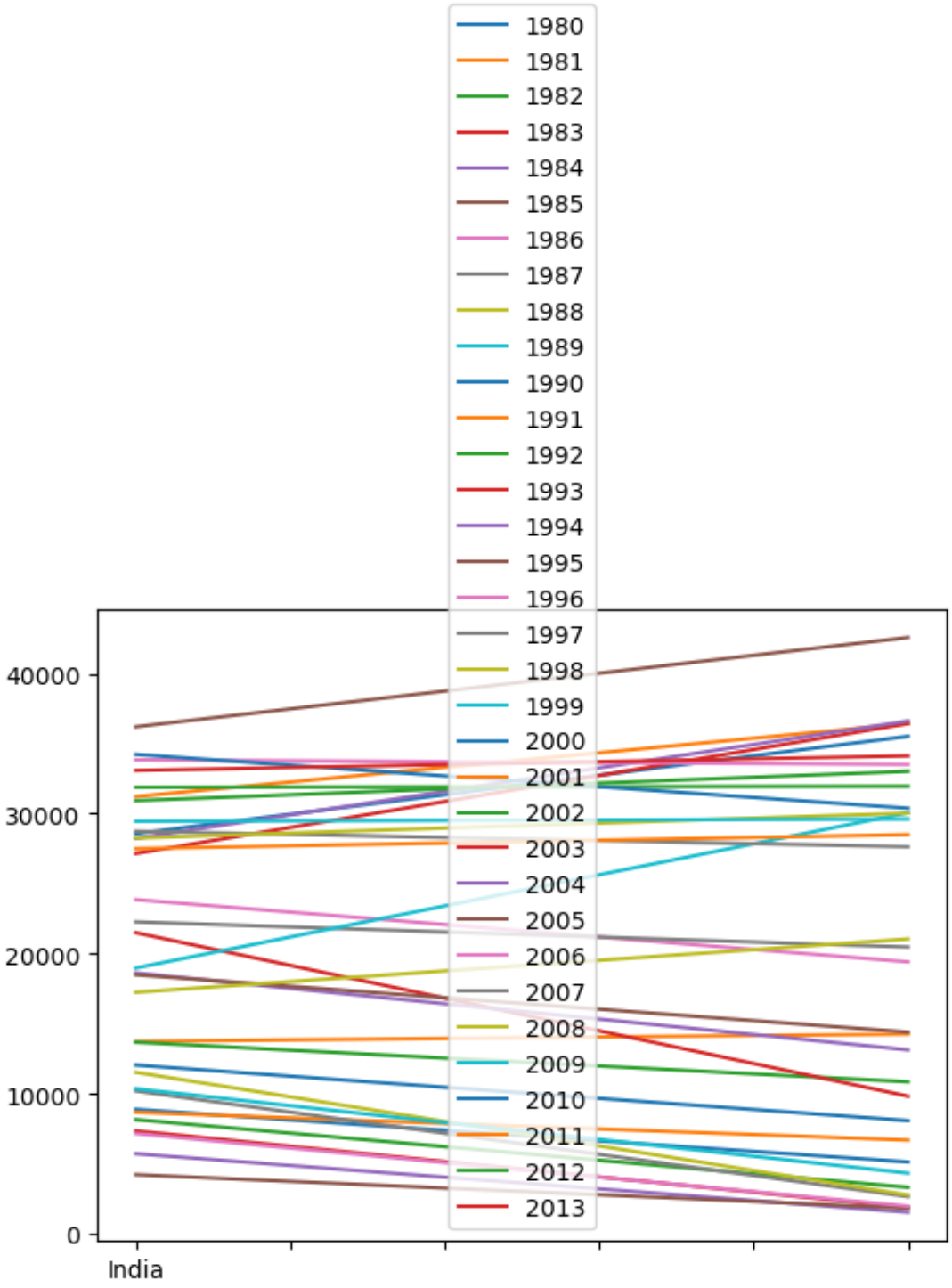
	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
India	28235	36210	33848	28742	28261	29456	34235	27509	30933	33087
China	36619	42584	33518	27642	30037	29622	30391	28502	33024	34129

[2 rows x 34 columns]

[58]:
```
# Step 2: Plot graph. We will explicitly specify line plot by passing in kind
 ↪parameter to plot().

df_CI.plot(kind='line')
```

```
plt.show()
```


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
40000
30000
20000
10000
0
India

[59]:
```
df_CI = df_CI.transpose()
df_CI.head()
```

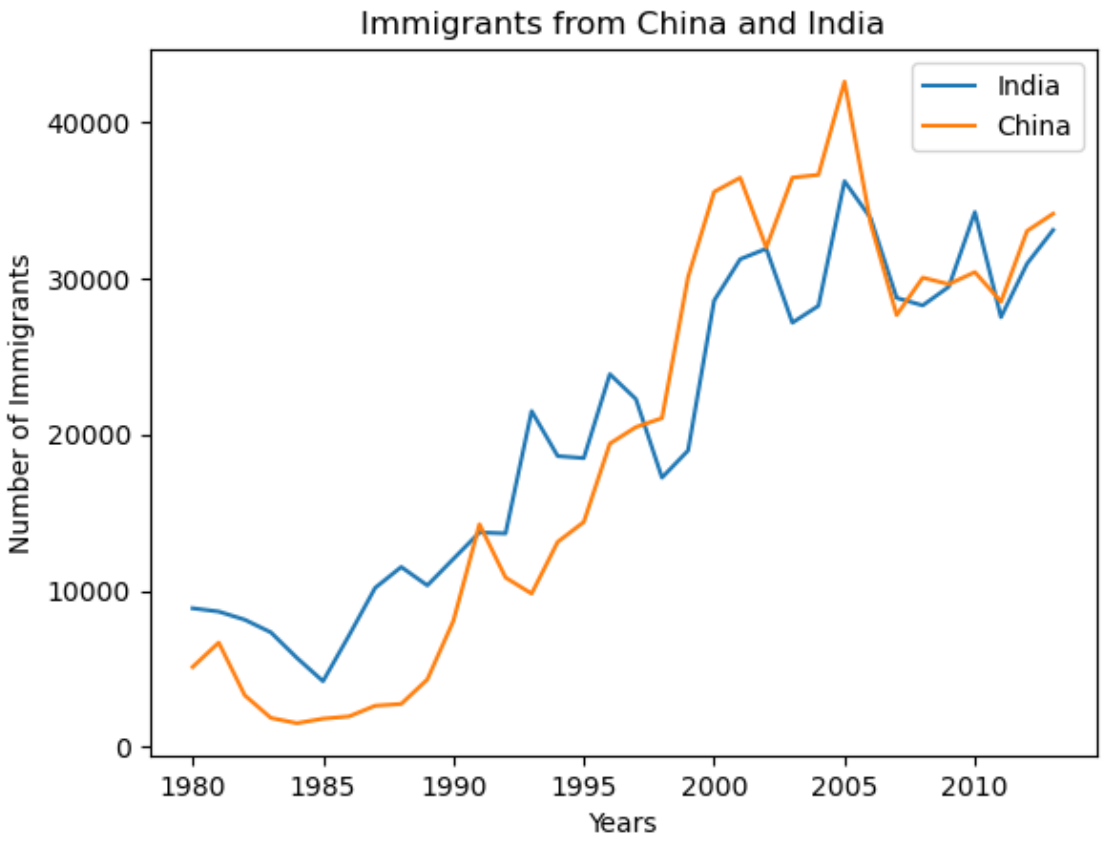
[59]:

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

[60]:
```
df_CI.plot(kind='line')

plt.title('Immigrants from China and India')
plt.ylabel('Number of Immigrants')
plt.xlabel('Years')

plt.show()
```



[61]:
```
# Question: Compare the trend of top 5 countries that contributed the most to␣
↪immigration to Canada.
```

```
# Step 1: Get the dataset. Recall that we created a Total column that␣
 ↪calculates the cumulative immigration
#by country. We will sort on this column to get our top 5 countries using␣
 ↪pandas sort_values() method.

# inplace = True paramemter saves the changes to the original df_can dataframe
df_can.sort_values(by='Total', ascending=False, axis=0, inplace=True)

# get the top 5 entries
df_top5 = df_can.head(5)

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

df_top5
```

[61]:

	India	China	United Kingdom of Great Britain and Northern Ireland	\
1980	8880	5123	22045	
1981	8670	6682	24796	
1982	8147	3308	20620	
1983	7338	1863	10015	
1984	5704	1527	10170	
1985	4211	1816	9564	
1986	7150	1960	9470	
1987	10189	2643	21337	
1988	11522	2758	27359	
1989	10343	4323	23795	
1990	12041	8076	31668	
1991	13734	14255	23380	
1992	13673	10846	34123	
1993	21496	9817	33720	
1994	18620	13128	39231	
1995	18489	14398	30145	
1996	23859	19415	29322	
1997	22268	20475	22965	
1998	17241	21049	10367	
1999	18974	30069	7045	
2000	28572	35529	8840	
2001	31223	36434	11728	
2002	31889	31961	8046	
2003	27155	36439	6797	
2004	28235	36619	7533	
2005	36210	42584	7258	
2006	33848	33518	7140	
2007	28742	27642	8216	
2008	28261	30037	8979	

```
2009  29456  29622                                                    8876
2010  34235  30391                                                    8724
2011  27509  28502                                                    6204
2012  30933  33024                                                    6195
2013  33087  34129                                                    5827

      Philippines  Pakistan
1980         6051       978
1981         5921       972
1982         5249      1201
1983         4562       900
1984         3801       668
1985         3150       514
1986         4166       691
1987         7360      1072
1988         8639      1334
1989        11865      2261
1990        12509      2470
1991        12718      3079
1992        13670      4071
1993        20479      4777
1994        19532      4666
1995        15864      4994
1996        13692      9125
1997        11549     13073
1998         8735      9068
1999         9734      9979
2000        10763     15400
2001        13836     16708
2002        11707     15110
2003        12758     13205
2004        14004     13399
2005        18139     14314
2006        18400     13127
2007        19837     10124
2008        24887      8994
2009        28573      7217
2010        38617      6811
2011        36765      7468
2012        34315     11227
2013        29544     12603
```

[62]:
```
# Step 2: Plot the dataframe. To make the plot more readeable, we will change the size using the figsize parameter.
df_top5.plot(kind='line', figsize=(14, 8)) # pass a tuple (x, y) size

plt.title('Immigration Trend of Top 5 Countries')
```

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

plt.show()
```