Visualization with Python by Kate (Introduction to Matplotlib and Line Plots)

Introduction

The aim of this Project is to introduce you to Matplotlib and creating Line Plots.

The Dataset: Immigration to Canada from 1980 to 2013

Dataset Source: International migration flows to and from selected countries - The 2015 revision. In this lab, we will focus on the Canadian immigration data.

We have already **pre-processed** the data, we will use the **clean data** saved in the csv format for this lab. The Canada Immigration dataset can be fetched from here.

Next, we'll do is import two key data analysis modules: *pandas* and *numpy*

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

Let's download and import our primary Canadian Immigration dataset using *pandas*'s read_csv () method.

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

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

Output:
Data read into a pandas dataframe!

Let's view the top 5 rows of the dataset using the head() function.

```
df_can.head()
# tip: You can specify the number of rows you'd like to see as follows:
df_can.head(10)
```

[3]:
```
df_can.head()
# tip: You can specify the number of rows you'd like to see as follows: df_can.head(
```

[3]:

	Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	...	200
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...	343
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...	122
2	Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	...	362
3	American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	...	
4	Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	...	

5 rows × 39 columns

Let's set Country as the index, it will help you to plot the charts easily, by refering to the country names as index value

df_can.set_index('Country', inplace=True)
# tip: The opposite of set is reset. So to reset the index, we can use df_can.reset_index()

#let's check
df_can.head(3)

[5]:
```
#let's check
df_can.head(3)
```

[5]:

Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...
Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	496	...
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	1	...
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	69	...

3 rows × 38 columns

# optional: to remove the name of the index
df_can.index.name = None

Since we converted the years to string, let's declare a variable that will allow us to easily call upon the full range of years:

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

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

Visualizing Data using Matplotlib

Matplotlib: Standard Python Visualization Library

The primary plotting library we will explore in the course is Matplotlib. As mentioned on their website:
Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, the jupyter notebook, web application servers, and four graphical user interface toolkits.
If you are aspiring to create impactful visualization with python, Matplotlib is an essential tool to have at your disposal.

Matplotlib.Pyplot

One of the core aspects of Matplotlib is matplotlib.pyplot. It is Matplotlib's scripting layer which we studied in details in the videos about Matplotlib. Recall that it is a collection of command style functions that make Matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In this lab, we will work with the scripting layer to learn how to generate line plots. In future labs, we will get to work with the Artist layer as well to experiment first hand how it differs from the scripting layer.

Let's start by importing matplotlib and matplotlib.pyplot as follows:

```
# we are using the inline backend
%matplotlib inline

import matplotlib as mpl
import matplotlib.pyplot as plt

#optional: check if Matplotlib is loaded.
print('Matplotlib version: ', mpl.__version__)  # >= 2.0.0

#optional: apply a style to Matplotlib
print(plt.style.available)
mpl.style.use(['ggplot']) # optional: for ggplot-like style
```

Output:
[‘Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark', 'seaborn-dark-palette', 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']

Plotting in *pandas*

Fortunately, pandas has a built-in implementation of Matplotlib that we can use. Plotting in *pandas* is as simple as appending a .plot() method to a series or dataframe.

Line Pots (Series/Dataframe)

What is a line plot and why use it?
A line chart or line plot is a type of plot which displays information as a series of data points called 'markers' connected by straight line segments. It is a basic type of chart common in many fields. Use line plot when you have a continuous data set. These are best suited for trend-based visualizations of data over a period of time.

Let's start with a case study:
In 2010, Haiti suffered a catastrophic magnitude 7.0 earthquake. The quake caused widespread devastation and loss of life and about three million people were affected by this natural disaster. As part of Canada's humanitarian effort, the Government of Canada stepped up its effort in accepting refugees from Haiti. We can quickly visualize this effort using a Line plot:
Question: Plot a line graph of immigration from Haiti using df.plot().

First, we will extract the data series for Haiti.

#Since we converted the years to string,
#let's declare a variable that will allow us to easily call upon the full range of years:
years = list(map(str, range(1980, 2014)))
#creating data series

haiti = df_can.loc['Haiti', years] # passing in years 1980 - 2013 to exclude the 'total' column
haiti.head()

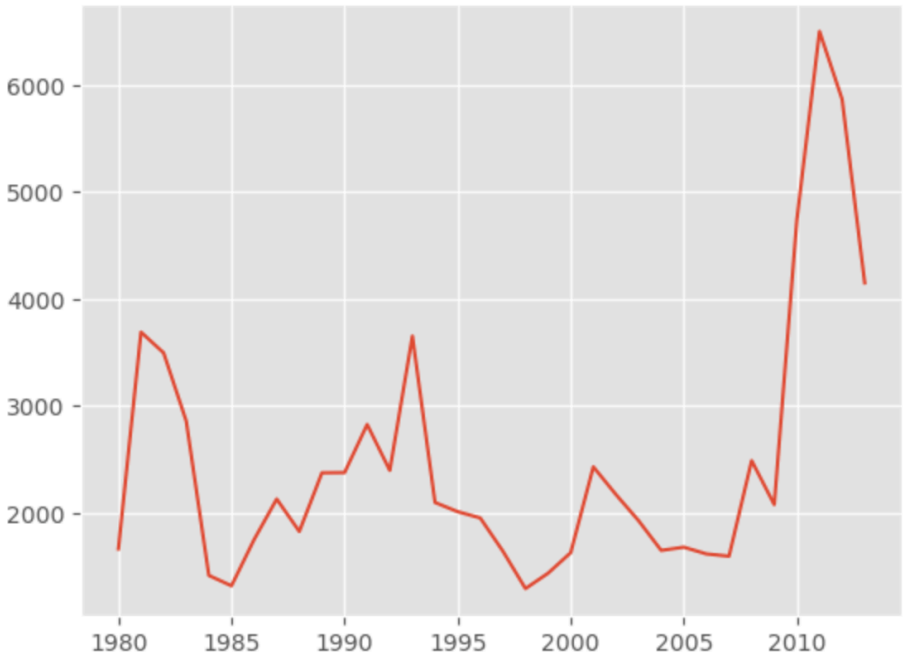
1980 1666
1981 3692
1982 3498
1983 2860
1984 1418
Name: Haiti, dtype: object

Next, we will plot a line plot by appending .plot() to the haiti dataframe.

haiti.plot()

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

[12]: <AxesSubplot:>
```



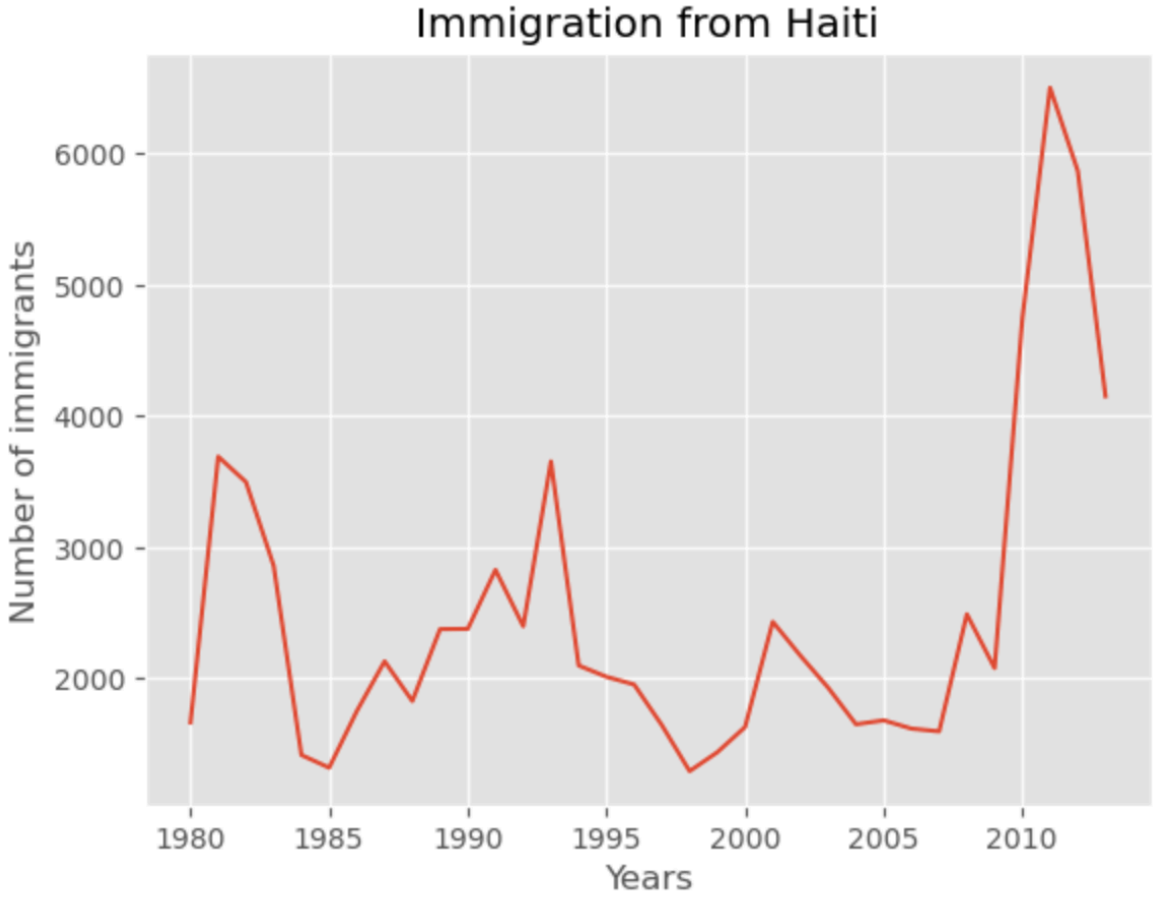
pandas automatically populated the x-axis with the index values (years), and the y-axis with the column values (population).

Also, let's label the x and y axis using plt.title(), plt.ylabel(), and plt.xlabel() as follows:

```
haiti.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
```



We can clearly notice how number of immigrants from Haiti spiked up from 2010 as Canada stepped up its efforts to accept refugees from Haiti. Let's annotate this spike in the plot by using the plt.text() method.

However, notice that years are of type *string*. Let's change the type of the index values to *integer* first.

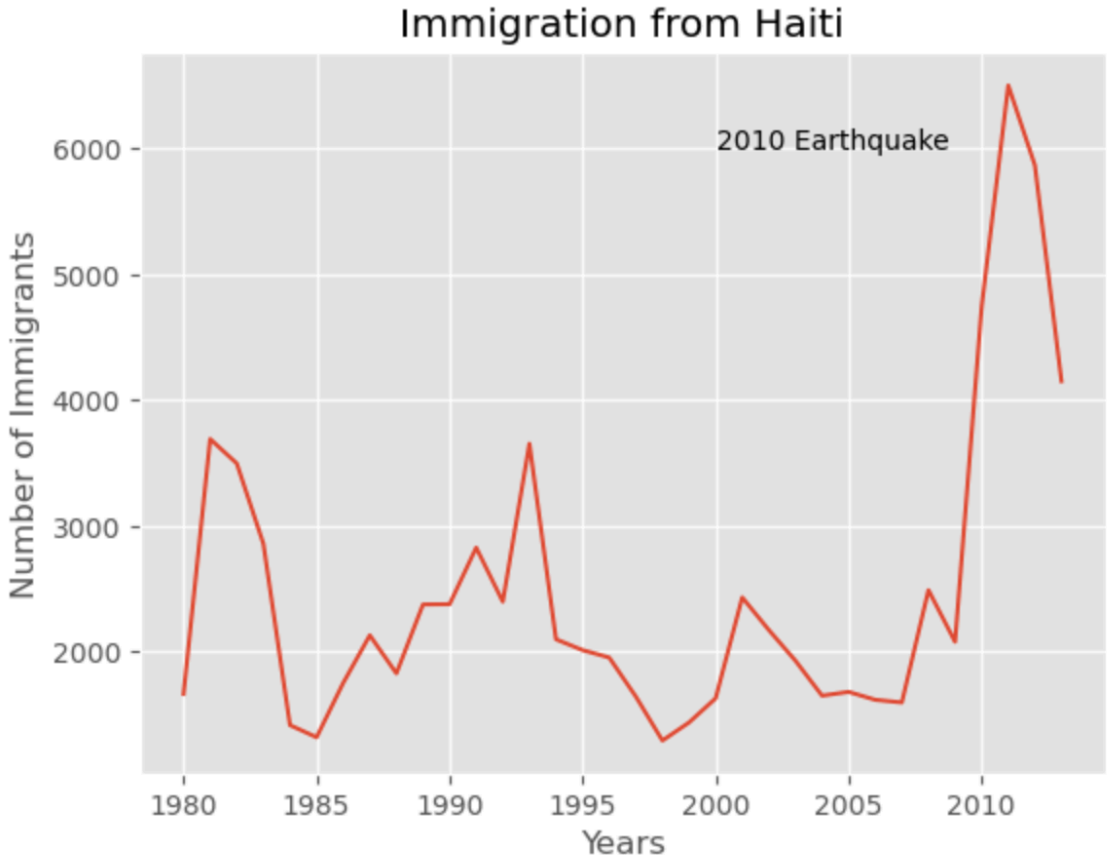
```
haiti.index = haiti.index.map(int)
haiti.plot(kind='line')

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(2000, 6000, '2010 Earthquake') # see note below

plt.show()
```



We can easily add more countries to line plot to make meaningful comparisons immigration from different countries.

Let's compare the number of immigrants from India and China from 1980 to 2013.

Step 1: Get the data set for China and India, and display the dataframe.

```
df_CI = df_can.loc[['India', 'China'], years]
df_CI
```

[18]:
```
df_CI = df_can.loc[['India', 'China'], years]
df_CI
```

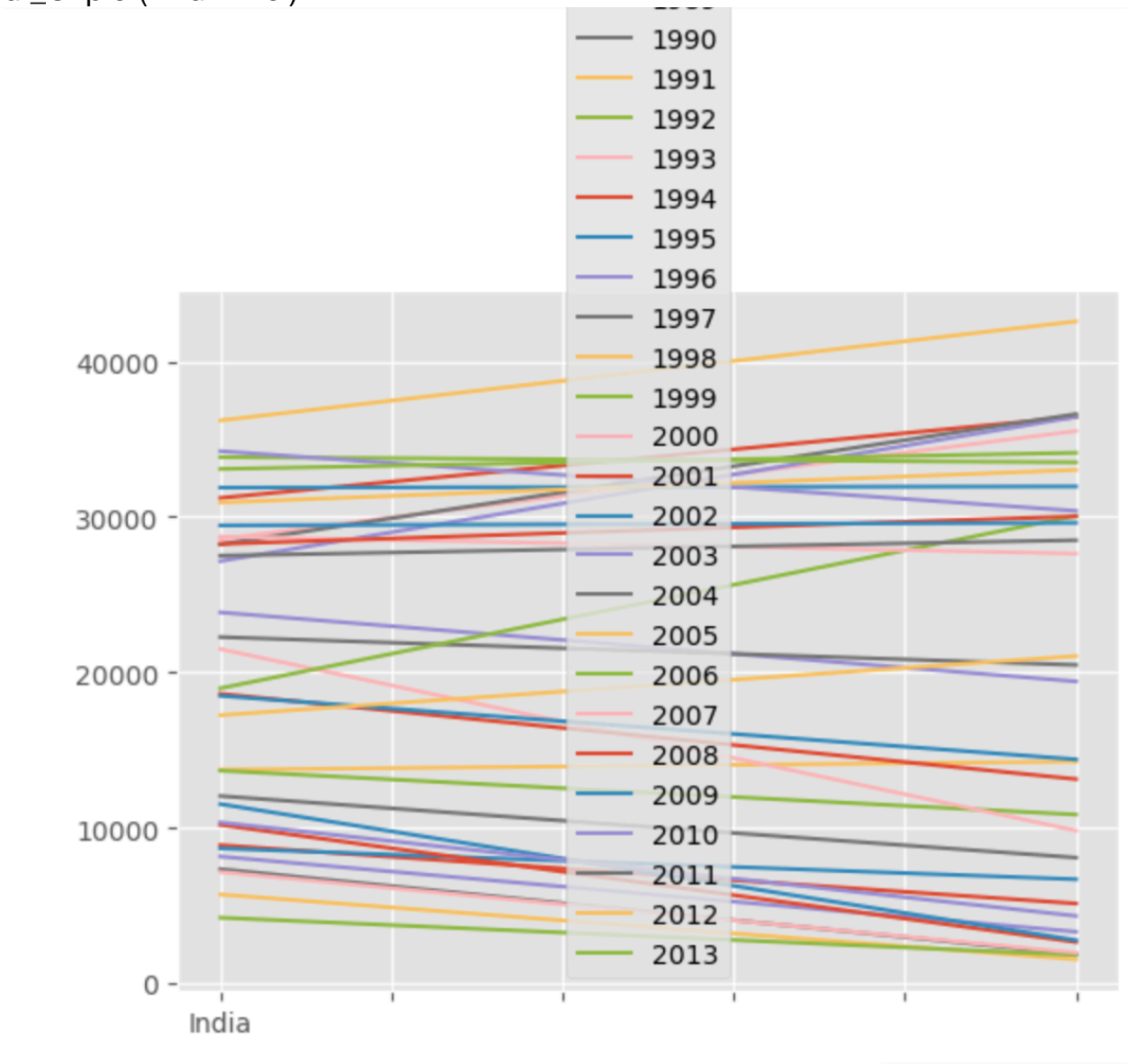
[18]:

	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2004	2005	2006
India	8880	8670	8147	7338	5704	4211	7150	10189	11522	10343	...	28235	36210	33848
China	5123	6682	3308	1863	1527	1816	1960	2643	2758	4323	...	36619	42584	33518

2 rows × 34 columns

Step 2: Plot graph. We will explicitly specify line plot by passing in kind parameter to plot().

df_CI.plot(kind='line')



That doesn't look right...

Recall that *pandas* plots the indices on the x-axis and the columns as individual lines on the y-axis. Since df_CI is a dataframe with the country as the index and years as the columns, we must first transpose the dataframe using transpose() method to swap the row and columns.

df_CI = df_CI.transpose()
df_CI.head()

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

[25]:

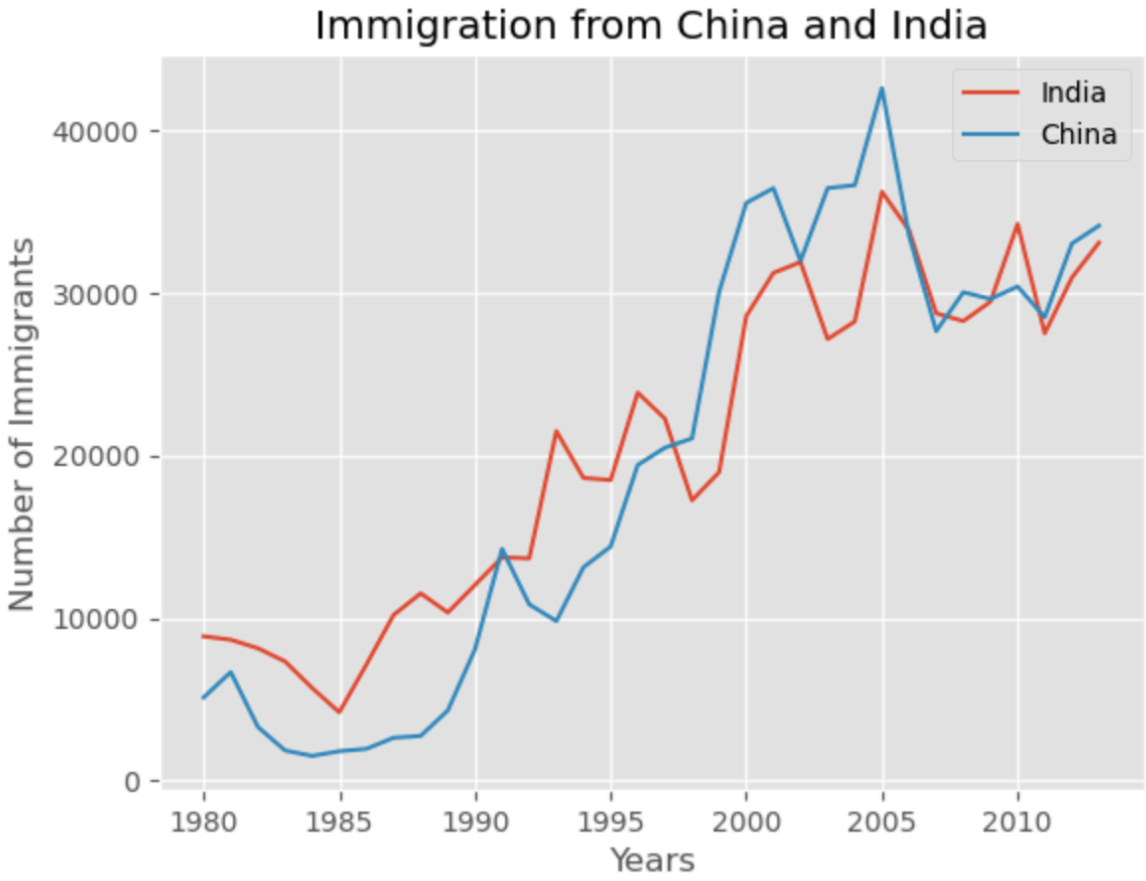
	India	China
1980	8880	5123
1981	8670	6682
1982	8147	3308
1983	7338	1863
1984	5704	1527

pandas will auomatically graph the two countries on the same graph. Now we go ahead and plot the new transposed dataframe.

df_CI.plot(kind='line')

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

plt.show()



From the above plot, we can observe that the China and India have very similar immigration trends through the years.

Line plot is a handy tool to display several dependent variables against one independent variable. However, it is recommended that no more than 5-10 lines on a single graph; any more than that and it becomes difficult to interpret.

Now let's compare the trend of top 5 countries that contributed the most to immigration to Canada.

```
df_can.sort_values(by='Total', ascending=False, axis=0, inplace=True)
top_5 = df_can.head(5)
top_5

df_top5 = df_can.loc[['India', 'China', 'United Kingdom of Great Britain and Northern Ireland', 'Philippines', 'Pakistan'], years]
df_top5
df_top5 = df_top5.transpose()
df_top5.head()
```

```
df_top5.plot(kind='line')

plt.title('The trend of top 5 countries that contributed the most to immigration to Canada')
plt.ylabel('Number of Immigrants')
plt.xlabel('Years')

plt.show()
```