[DataFrame Plot](), [Series Plot]()

Line Plot

Line Plot displays information as a series of data points called 'markers' connected by straight line segments. Used for continuous data sets. Best suited for trend-based visualizations of data over a period of time.
Line plot is a handy tool to display several dependent variables against one independent variable.

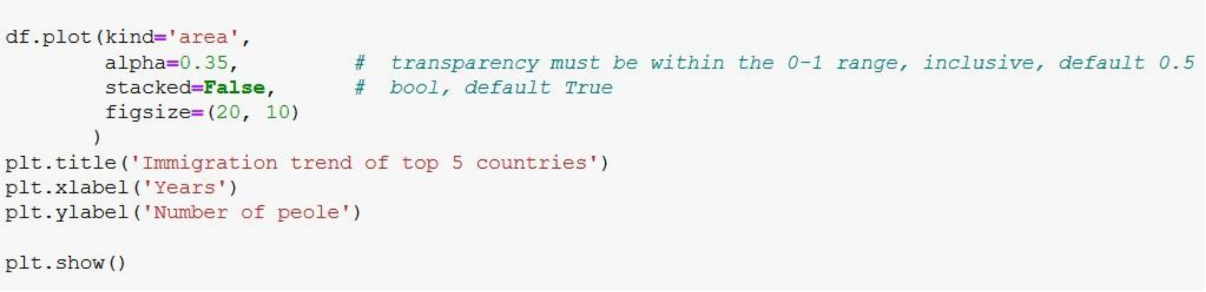
Area Plot

Area plots are stacked by default. And to produce a stacked area plot, each column must be either all positive or all negative values (any **NaN**, i.e. not a number, values will default to 0). Set parameter **stacked** to value **False** to produce an unstacked plot.

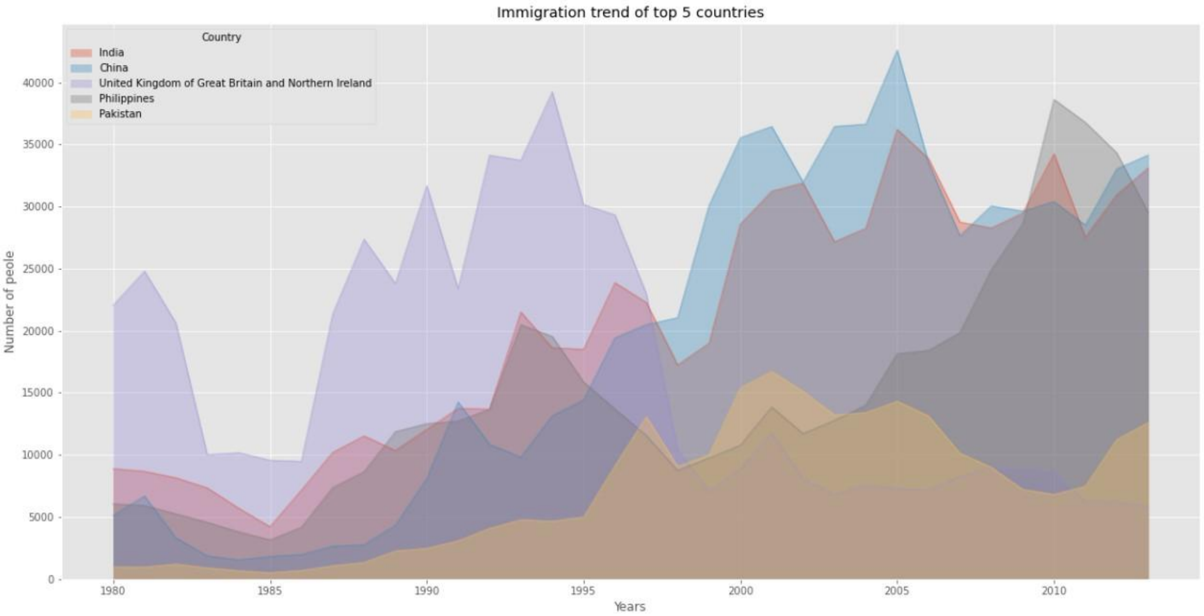
The unstacked plot has a **default** transparency parameter **alpha=0.5** that could be modified.

Two **styles/options of plotting** with matplotlib:

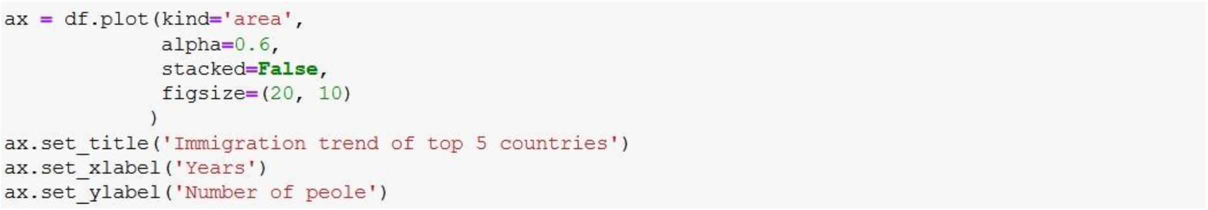
Using **scripting** layer:



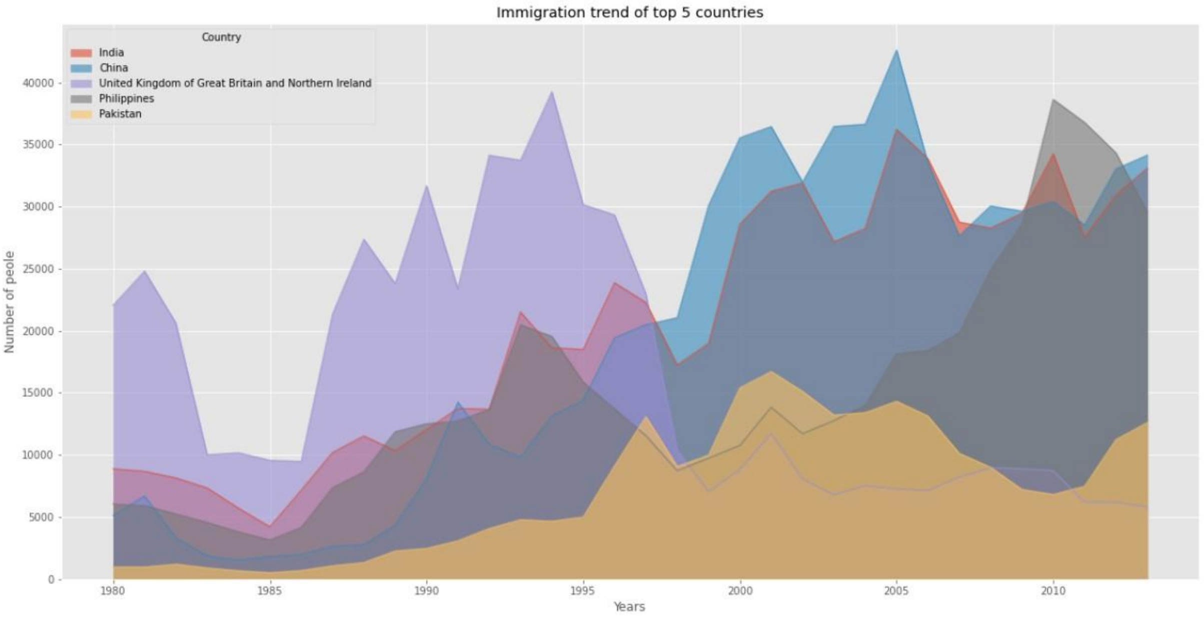
```
df.plot(kind='area',
        alpha=0.35,          #  transparency must be within the 0-1 range, inclusive, default 0.5
        stacked=False,       #  bool, default True
        figsize=(20, 10)
       )
plt.title('Immigration trend of top 5 countries')
plt.xlabel('Years')
plt.ylabel('Number of peole')

plt.show()
```



Using the **Artist** layer:



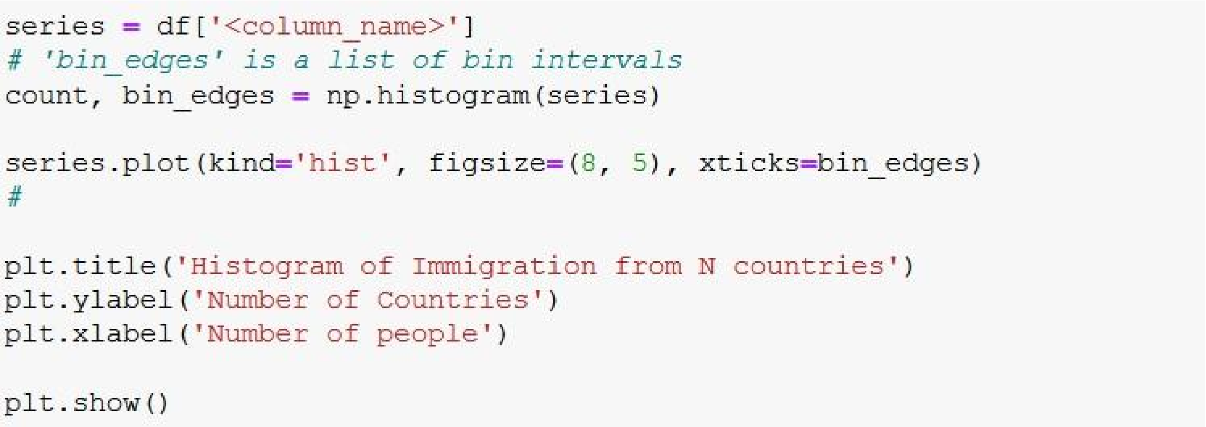
```
ax = df.plot(kind='area',
             alpha=0.6,
             stacked=False,
             figsize=(20, 10)
            )
ax.set_title('Immigration trend of top 5 countries')
ax.set_xlabel('Years')
ax.set_ylabel('Number of peole')
```



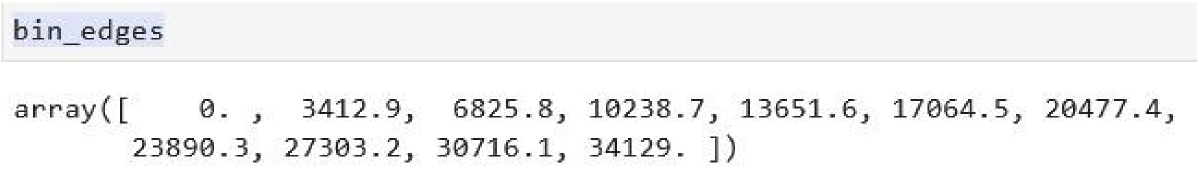
Histograms

Representing the frequency distribution of a numeric dataset by partitioning the x-axis into bins (default 10), assigns each data point in the dataset to a bin, and then counts the number of data points that have been assigned to each bin.
Passing in a **xticks** parameter list of bin sizes, bins edges on the histogram match with the bin size.

Series histogram

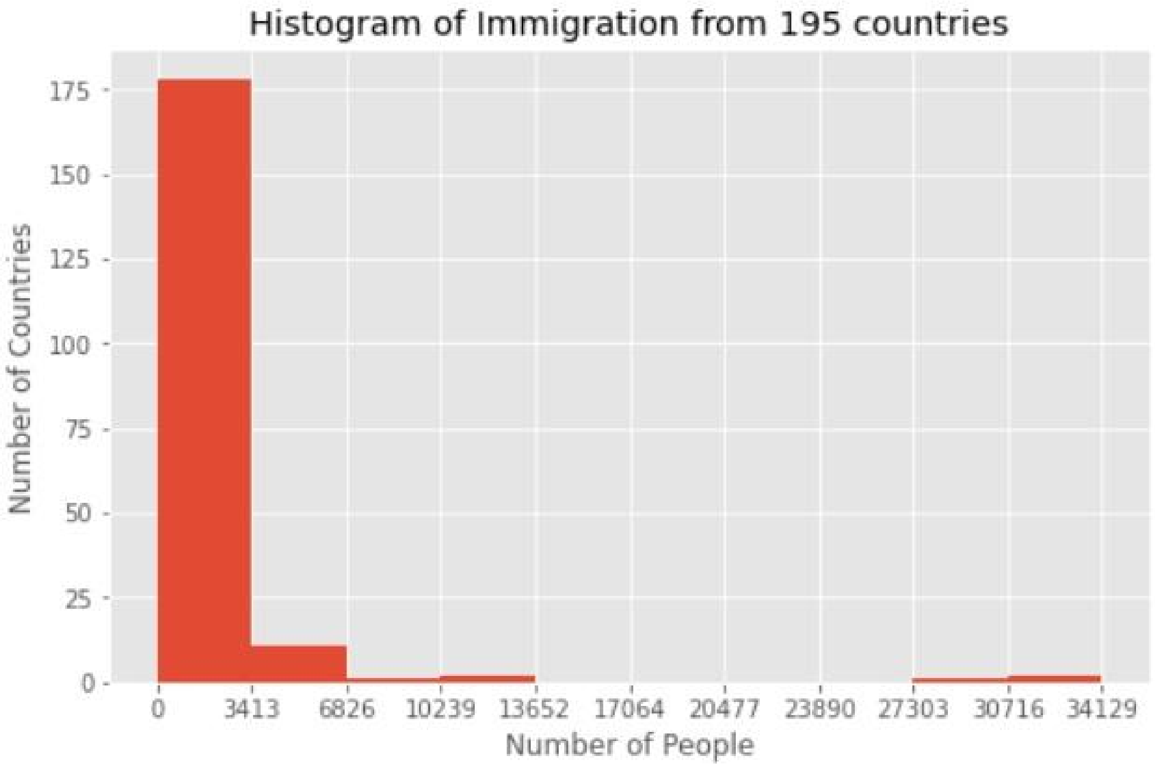


```
series = df['<column_name>']
# 'bin_edges' is a list of bin intervals
count, bin_edges = np.histogram(series)

series.plot(kind='hist', figsize=(8, 5), xticks=bin_edges)
#

plt.title('Histogram of Immigration from N countries')
plt.ylabel('Number of Countries')
plt.xlabel('Number of people')

plt.show()
```

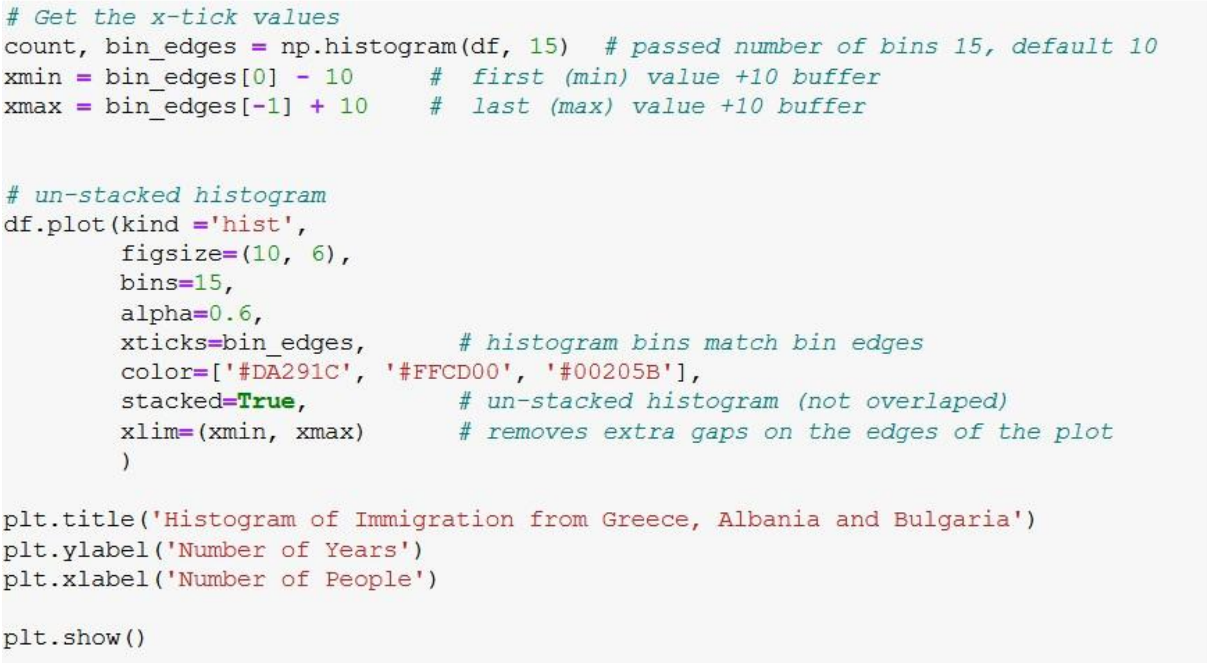


```
bin_edges
```

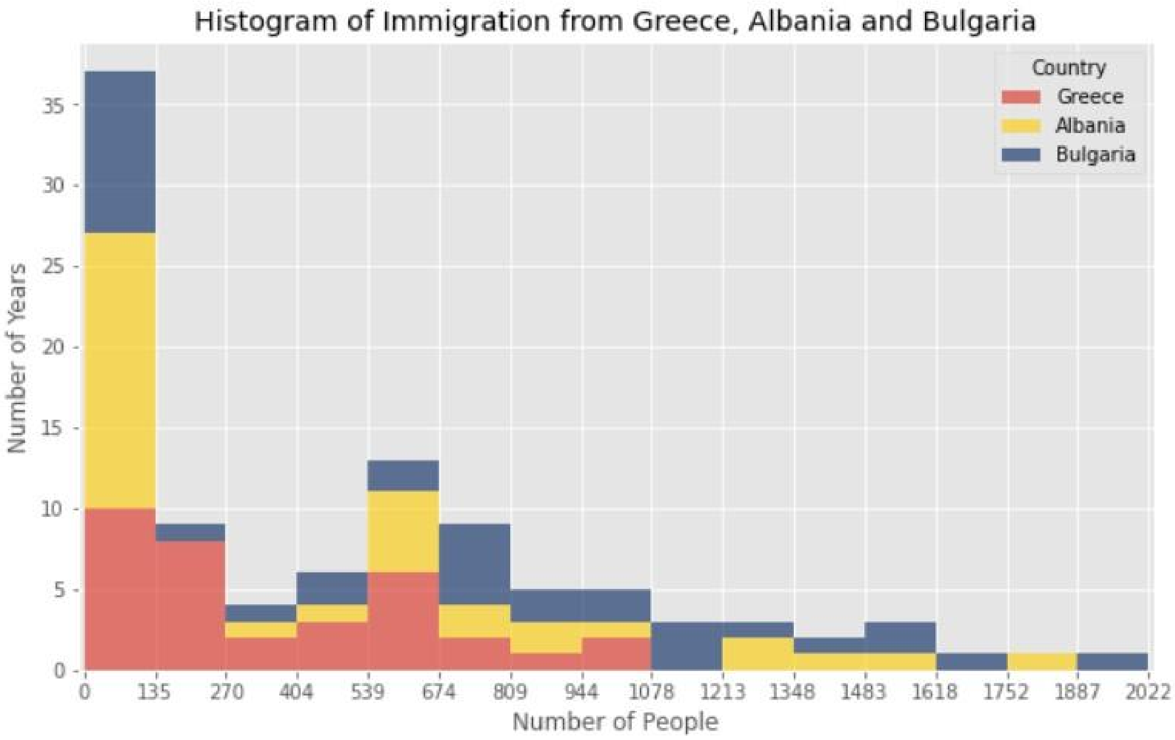
```
array([    0. ,   3412.9,   6825.8, 10238.7, 13651.6, 17064.5, 20477.4,
        23890.3, 27303.2, 30716.1, 34129. ])
```



DataFrame histogram



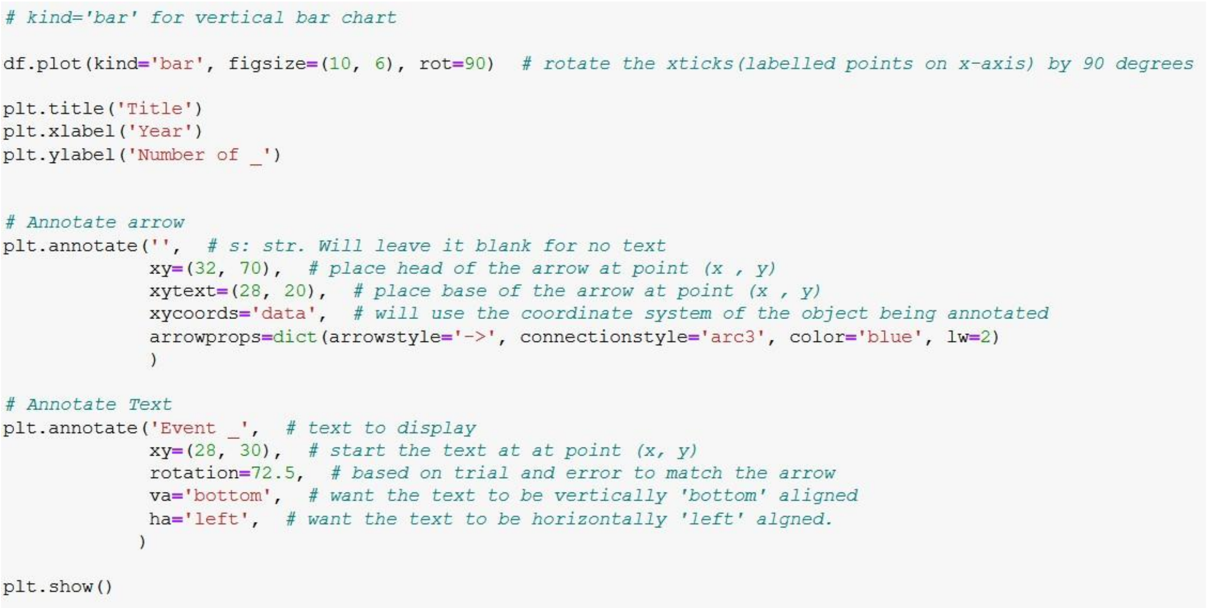
```
# Get the x-tick values
count, bin_edges = np.histogram(df, 15)  # passed number of bins 15, default 10
xmin = bin_edges[0] - 10     #  first (min) value +10 buffer
xmax = bin_edges[-1] + 10    #  last (max) value +10 buffer


# un-stacked histogram
df.plot(kind ='hist',
        figsize=(10, 6),
        bins=15,
        alpha=0.6,
        xticks=bin_edges,       # histogram bins match bin edges
        color=['#DA291C', '#FFCD00', '#00205B'],
        stacked=True,           # un-stacked histogram (not overlaped)
        xlim=(xmin, xmax)       # removes extra gaps on the edges of the plot
        )

plt.title('Histogram of Immigration from Greece, Albania and Bulgaria')
plt.ylabel('Number of Years')
plt.xlabel('Number of People')

plt.show()
```



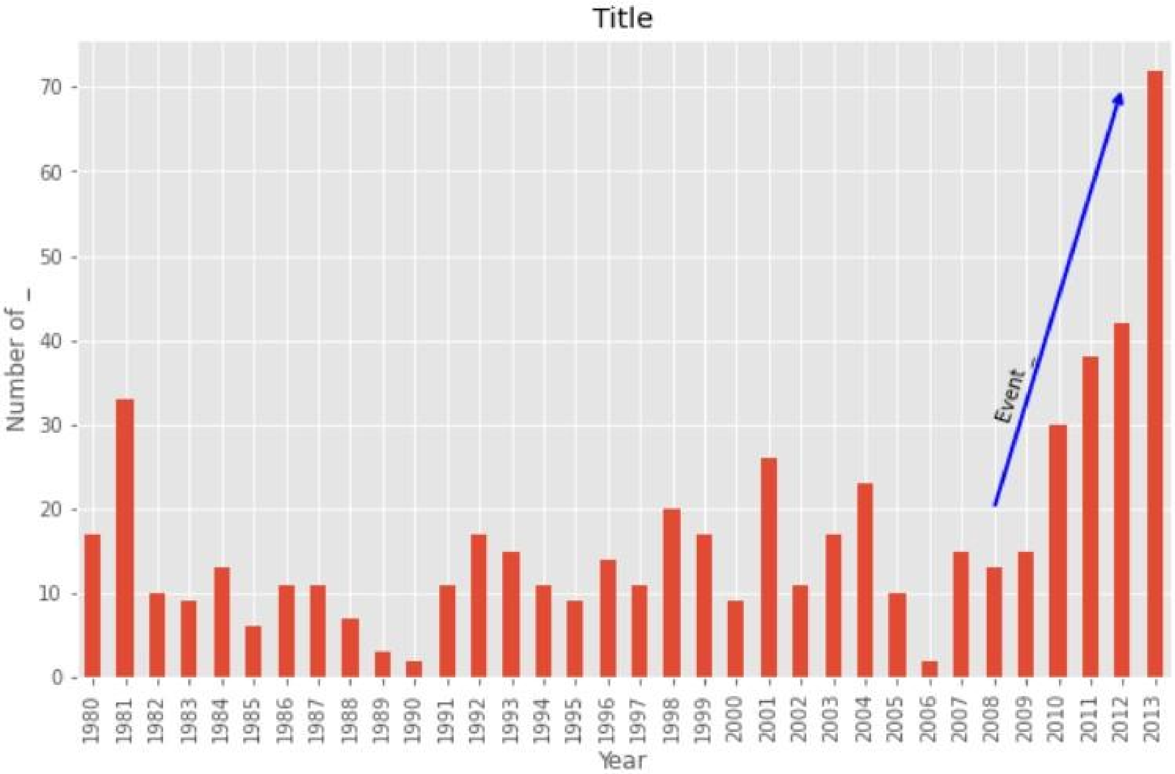
Bar Charts for DataFrame

A bar plot is a way of representing data where the length of the bars represents the magnitude/size of the feature/variable. Usually represent numerical and categorical variables grouped in intervals.

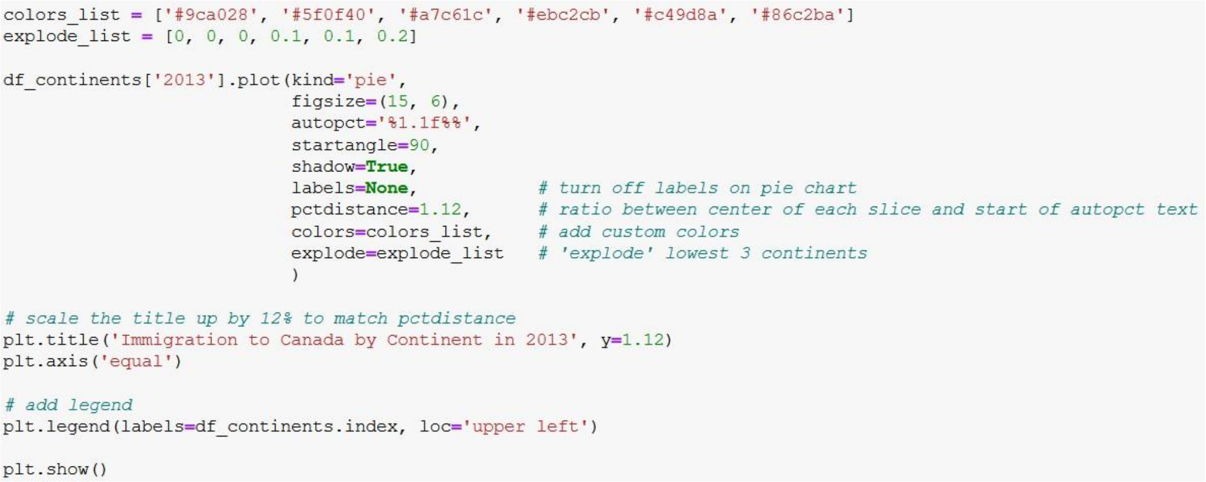
kind=bar for vertical bar chart, **kind=barh** for horizontal



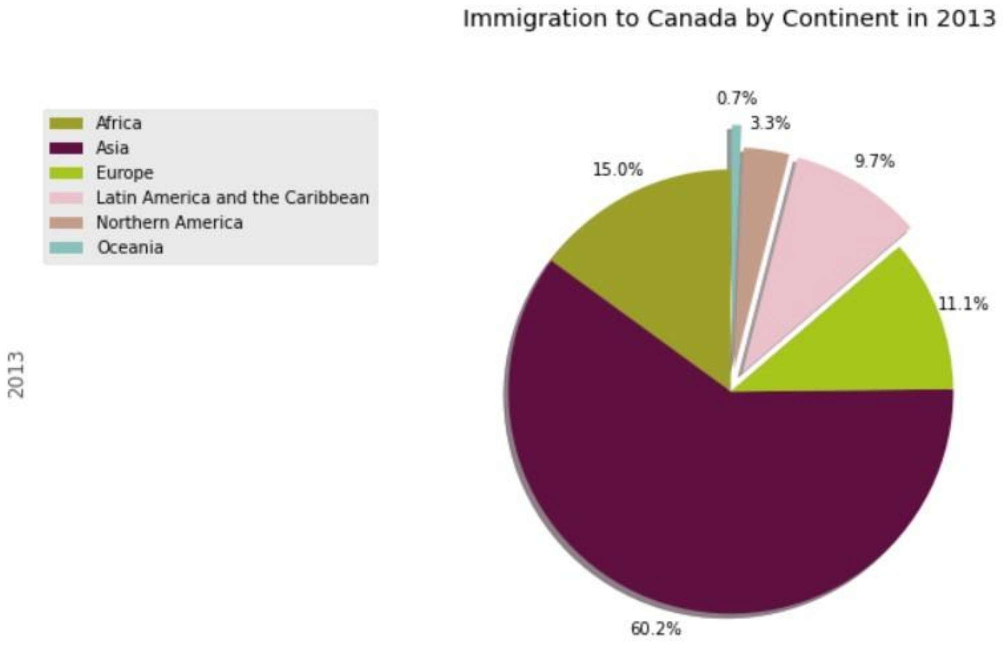
```
# kind='bar' for vertical bar chart

df.plot(kind='bar', figsize=(10, 6), rot=90)  # rotate the xticks(labelled points on x-axis) by 90 degrees

plt.title('Title')
plt.xlabel('Year')
plt.ylabel('Number of _')


# Annotate arrow
plt.annotate('',  # s: str. Will leave it blank for no text
             xy=(32, 70),  # place head of the arrow at point (x , y)
             xytext=(28, 20),  # place base of the arrow at point (x , y)
             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('Event _',  # text to display
             xy=(28, 30),  # start the text at at point (x, y)
             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 horizontally 'left' alged.
            )

plt.show()
```



Pie Chart

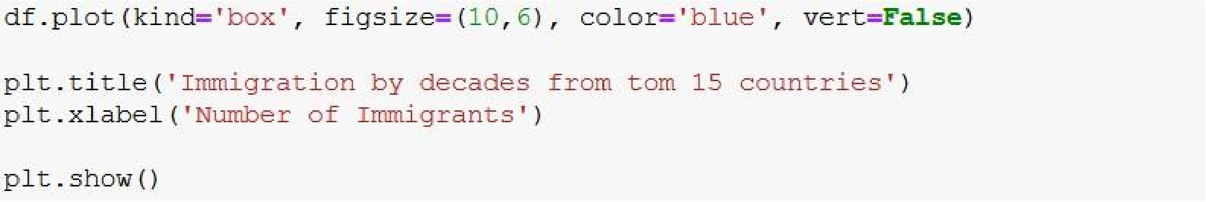


```
colors_list = ['#9ca028', '#5f0f40', '#a7c61c', '#ebc2cb', '#c49d8a', '#86c2ba']
explode_list = [0, 0, 0, 0.1, 0.1, 0.2]

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,      # ratio between center of each slice and start of autopct text
                           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 in 2013', y=1.12)
plt.axis('equal')

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

plt.show()
```

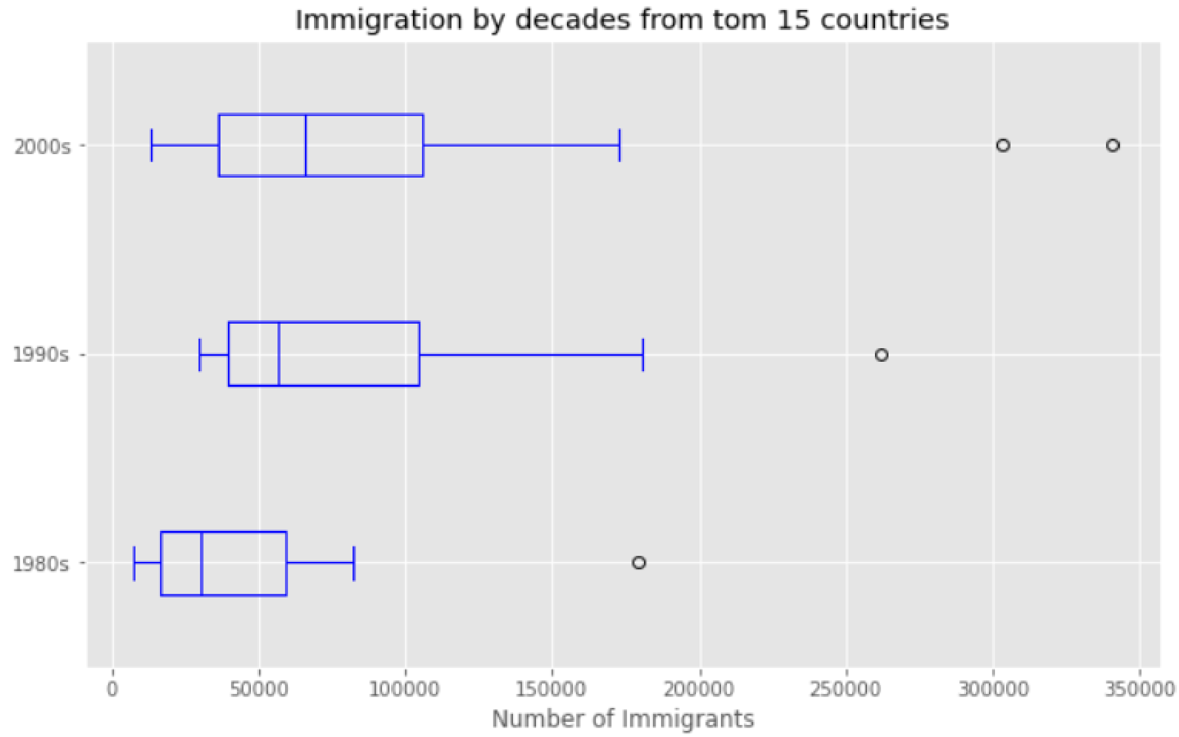


Box Plot (min-Q1-Q2-Q3-max)

Horizontal plot by passing parameter vert=False (default True)



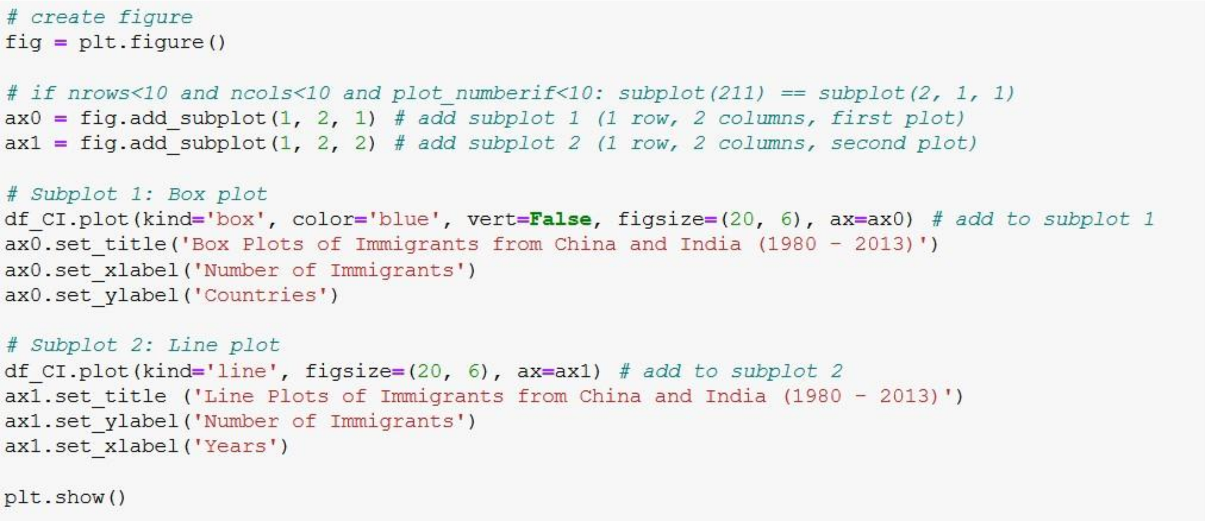
```
df.plot(kind='box', figsize=(10,6), color='blue', vert=False)

plt.title('Immigration by decades from tom 15 countries')
plt.xlabel('Number of Immigrants')

plt.show()
```



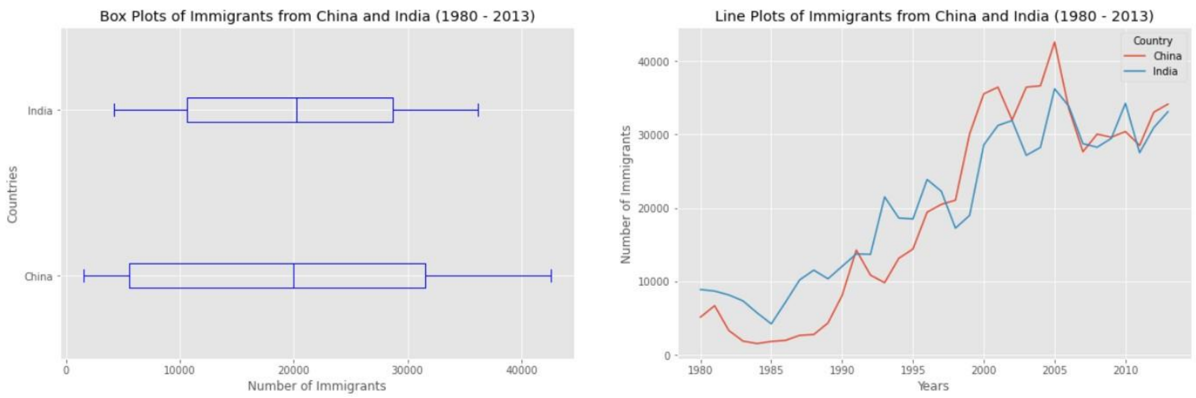
To be outliers values must be < 1.5*IQR or > 1.5*IQR. IQR = Q3(75%)-Q1(25%)
2000s Outlier > 105,505.5 + (1.5 * 69,404)
2000s Outlier > 209,611.5

Subplots

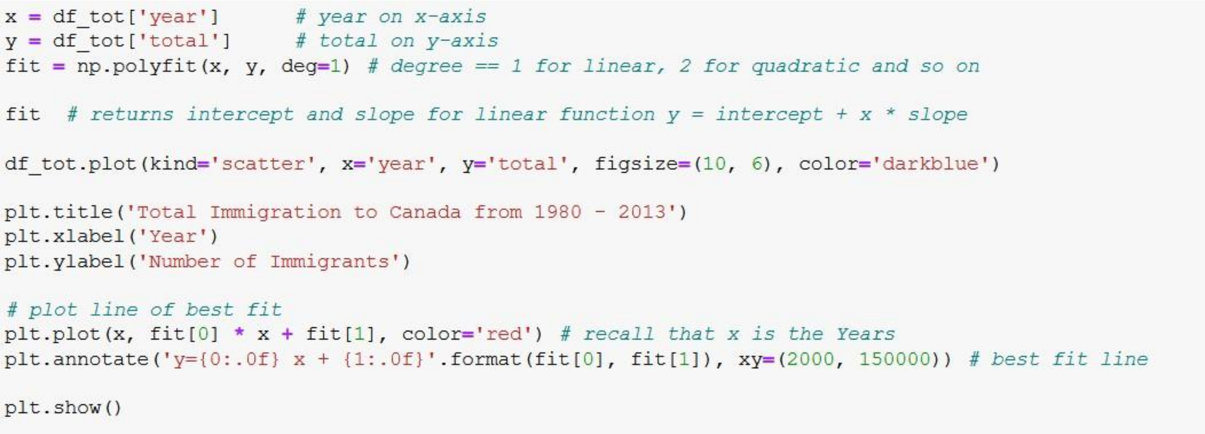
Artist layer prefered. To visualize multiple plots together, create a **figure** (overall canvas) and divide it into **subplots**, each containing a plot.



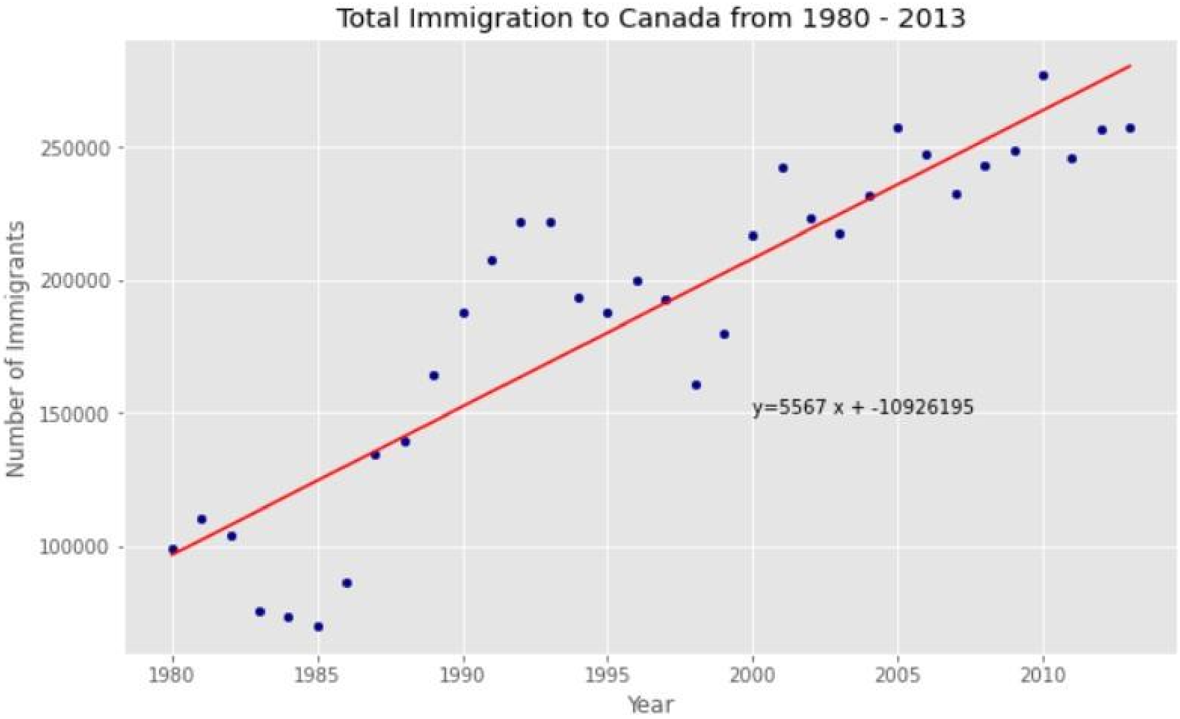
```
# create figure
fig = plt.figure()

# if nrows<10 and ncols<10 and plot_numberif<10: subplot(211) == subplot(2, 1, 1)
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)

# 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()
```



Scatter Plot



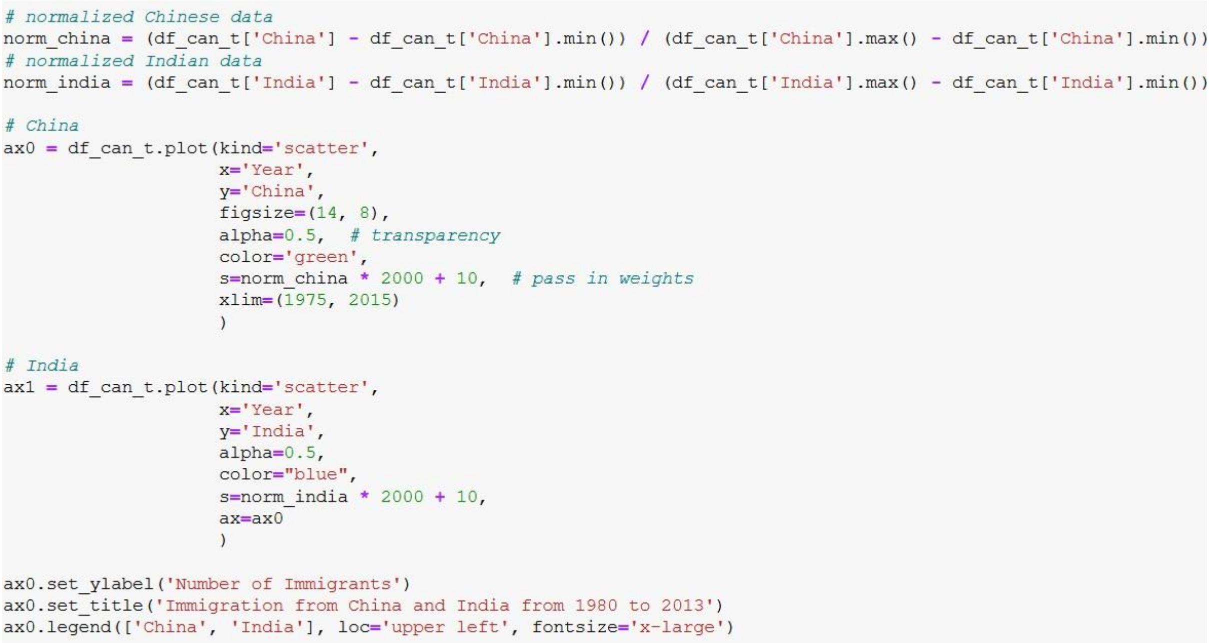
```
x = df_tot['year']       # year on x-axis
y = df_tot['total']      # total on y-axis
fit = np.polyfit(x, y, deg=1) # degree == 1 for linear, 2 for quadratic and so on

fit  # returns intercept and slope for linear function y = intercept + x * slope

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)) # best fit line

plt.show()
```



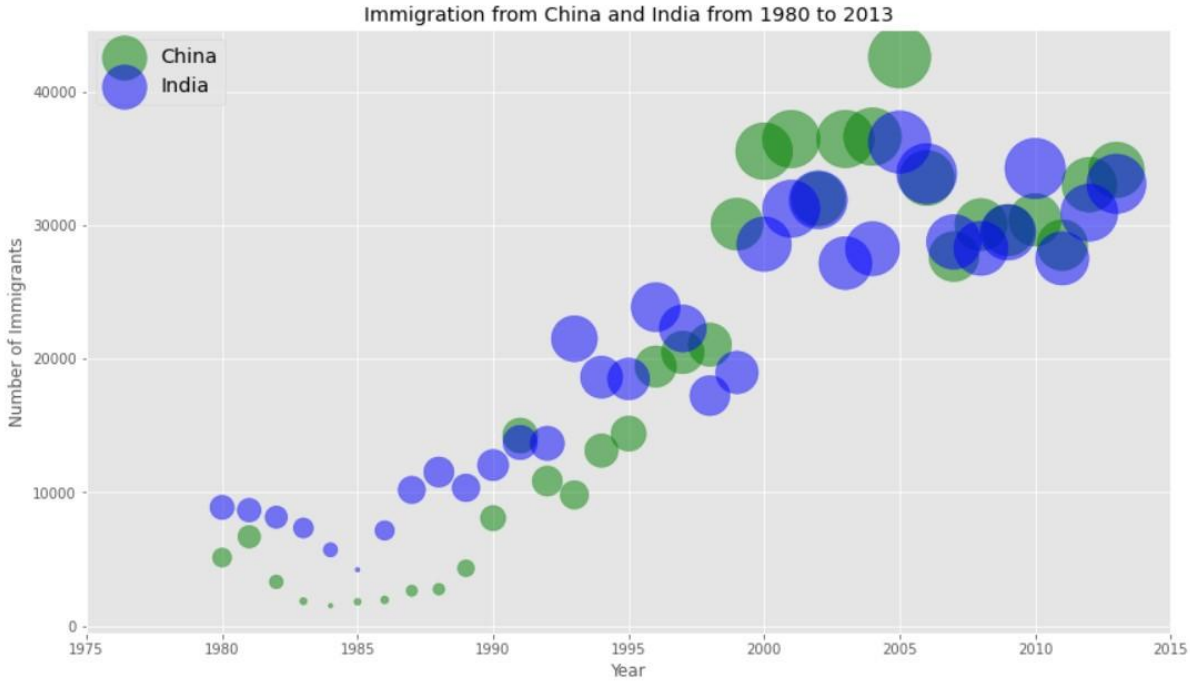
Use intercept and slope to predict future value based on the fitted line.

Bubble Plot

Variation of the scatter plot that displays three dimensions of data (x, y, z). Size of the bubble is determined by the third variable z (weight).



```
# 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())

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

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

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


Immigration from China and India from 1980 to 2013
China
India
Number of Immigrants
Year