Basic Visualization Tools

Area Plots

What will Learn

- Describe what is an area plot
- Explain how to create an area plot using Matplotlib.

What is an area plot?

An area plot, also known as an area chart or graph, displays the magnitude and proportion of multiple variables over a continuous axis, typically representing time or another ordered dimension.

It's similar to a line plot, but with the area below the line filled with color to emphasize the cumulative magnitude of the variables.

This kind of graph is commonly used when trying to compare two or more quantities.

Generating an area plot

```
df_canada.sort_values(['Total'], ascending = False, axis = 0, inplace = True)
```

	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
Country																					
India	Asia	Southern Asia	Developing regions	8880	8670	8147	7338	5704	4211	7150	...	36210	33848	28742	28261	29456	34235	27509	30933	33087	691904
China	Asia	Eastern Asia	Developing regions	5123	6682	3308	1863	1527	1816	1960	...	42584	33518	27642	30037	29622	30391	28502	33024	34129	659962
United Kingdom of Great Britain and Northern Ireland	Europe	Northern Europe	Developed regions	22045	24796	20620	10015	10170	9564	9470	...	7258	7140	8216	8979	8876	8724	6204	6195	5827	551500
Philippines	Asia	South-Eastern Asia	Developing regions	6051	5921	5249	4562	3801	3150	4166	...	18139	18400	19837	24887	28573	38617	36765	34315	29544	511391
Pakistan	Asia	Southern Asia	Developing regions	978	972	1201	900	668	514	691	...	14314	13127	10124	8994	7217	6811	7468	11227	12603	241600

We use the sort_values function to sort our dataframe in descending order. Here is the result. It turns out that India followed by China, then the United Kingdom, the Philippines, and Pakistan, are the top five countries with the highest immigration to Canada.

Generate the area plots using the first five rows of this dataframe? Not quite yet. First, we need to create a new dataframe of these five countries only and exclude the total column. More importantly, to generate the area plots for these countries, we should plot the years on the horizontal axis and the annual immigration on the vertical axis. Note that Matplotlib plots

the indices of a dataframe on the horizontal axis and with the dataframe as shown, Matplotlib plots the countries on the horizontal axis. To fix this, we need to take the transpose of the dataframe.

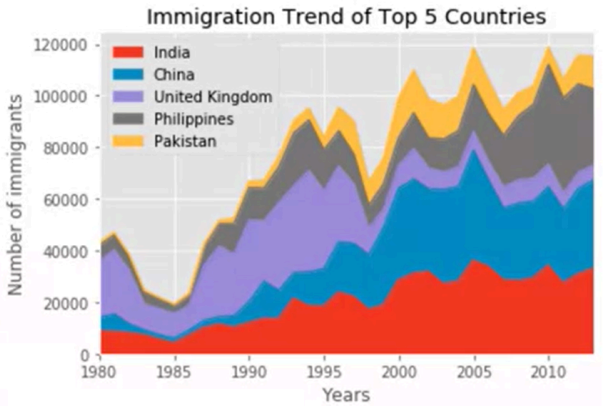
```
years = list(map(str, range(1980, 2014)))

df_canada.sort_values(['Total'], ascending = False, axis = 0, inplace = True)

df_top5 = df_canada.head()
df_top5 = df_top5[years].transpose()
```

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

we create a new dataframe of the top five countries, and we call it ‘df_top5’. We then select only the columns representing the years 1980 - 2013 in order to exclude the total column before applying the transpose method. The resulting dataframe is exactly what we want with five columns, where each column represents one of the top five countries and the years being the indices. Now we can go ahead and use the plot function on dataframe, df_top five to generate the area plots.

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

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

plt.title('Immigration trend of top 5 countries')
plt.ylabel('Number of immigrants')
plt.xlabel('Years')

plt.show()
```



Area plot: Uses

Area plots are particularly effective and depicting data with a cumulative nature, such as

- tracking stock market performance,
- visualizing population demographics,
- displaying the distribution of resources across various sectors.

Area plots provide a visually appealing and intuitive way to showcase the relationship and proportion of multiple variables in a single chart.

Histograms

What will Learn

- Define a histogram with the help of an illustration
- Explore the process of creating a histogram using Matplotlib

What is a histogram?

A histogram is a way of representing the frequency distribution of a numeric data set.

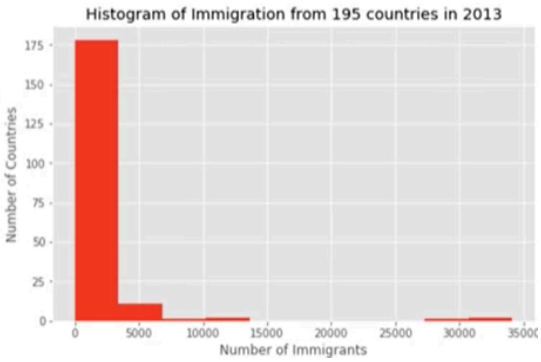
- Partitions the spread of the numeric data into bins,
- Assigns each data point in the data set to a bin,
- Counts the number of data points assigned to each bin.

So, the vertical axis is essentially the frequency or the number of data points in each bin.

Generating a histogram

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

```
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()
```

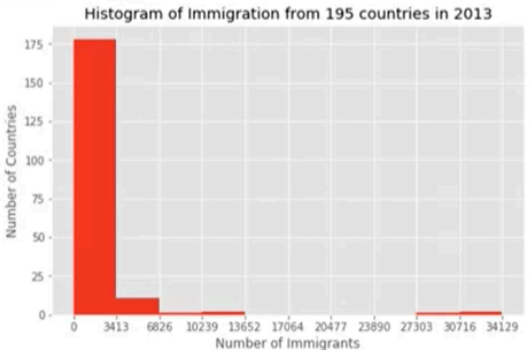


Notice how the bins are not aligned with the tick marks on the horizontal axis, this can make the histogram hard to read.

So, let's try to fix this in order to make our histogram more effective. One way to solve this issue is to borrow the histogram function from the NumPy library, so as usual, we start by importing Matplotlib and its scripting interface, but this time we also import the NumPy library, then we call the NumPy histogram function on the data in column 2013.

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

```
count, bin_edges = np.histogram(df_canada['2013'])

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

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

plt.show()
```



we call the NumPy histogram function on the data in column 2013. This function is going to partition the spread of the data in column 2013 in ten bins of equal width, where ten is the default number of bins. It also computes the number of data points that fall in each bin and then return this frequency of each bin which we are calling ‘count’ here, and the bin edges which we will call ‘bin_ edges’.

Bar Charts

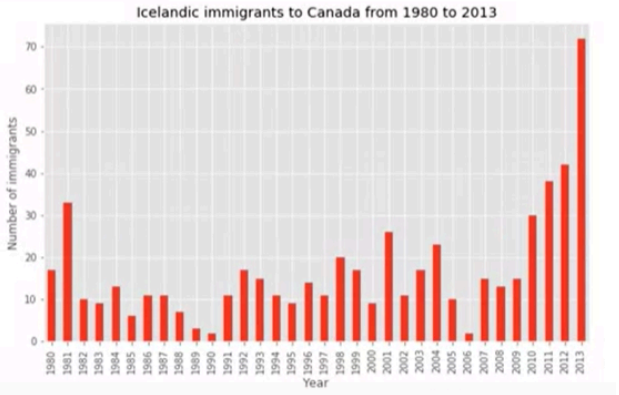
What will Learn

- Describe a bar chart with the help of an illustration
- Explore the process of creating a bar chart using Matplotlib.

What is a bar charts?

A bar chart, also known as a bar graph, is a type of plot where the length of each bar is proportional to the value of the item that it represents. It's commonly used to compare the values of a variable at a given point in time.

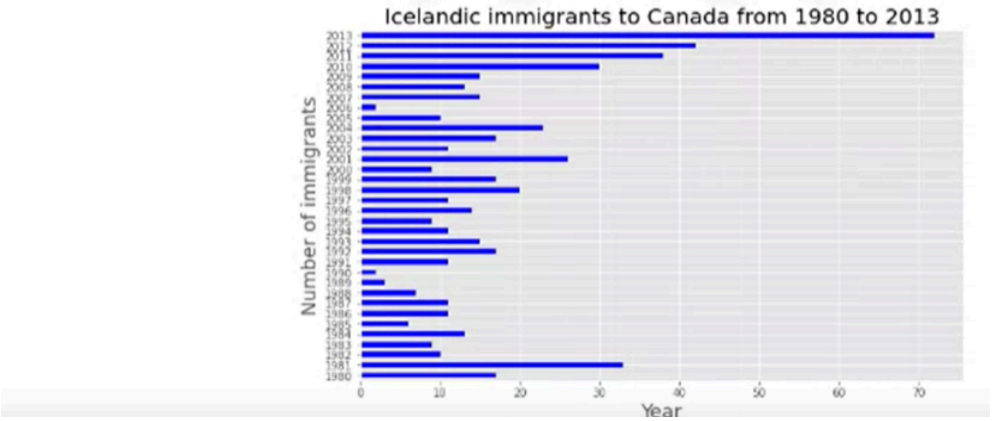
```
import matplotlib as mpl
import matplotlib.pyplot as plt

Years = list (map(str, range (1980, 2014)))
df_iceland = df_canada.loc[ 'Iceland',
years]

df_iceland.plot(kind='bar')

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

plt.show()
```



By examining the bar chart, we noticed that immigration to Canada from Iceland has seen an increasing trend since 2010.

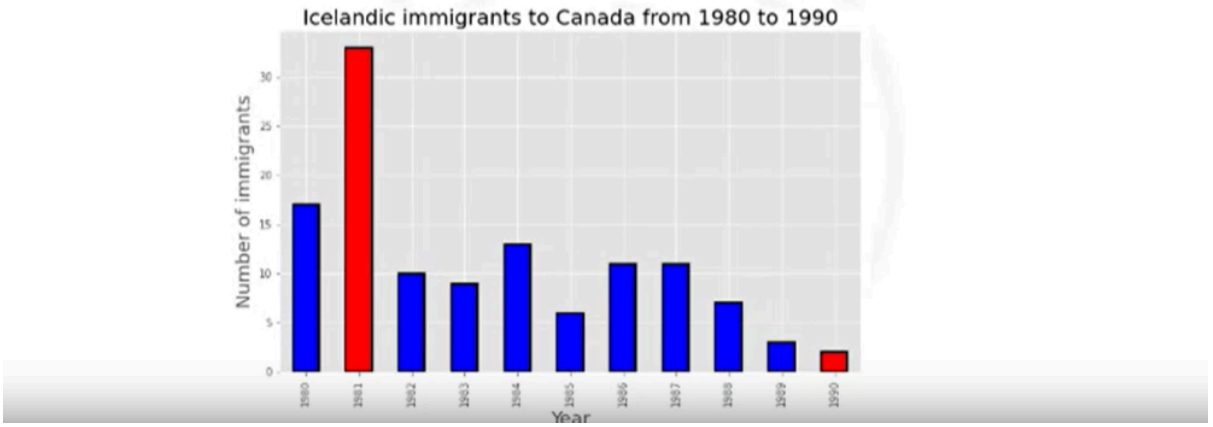
You can also create a bar chart with horizontal bars by assigning bar to the kind parameter of the plot function.Note the use of the color parameter as you can change the color of the bar with this

```
df_iceland.plot(kind='barh', color ='red')
```



Let's suppose you want to highlight the years with highest and lowest number of Icelandic immigrants to Canada between the year 1980 to 1990. You can pass a list of colors to the color parameter accordingly. Here we have highlighted the bars for the years 1981 and 1990 with the color red. By assigning the color of your choice to the edge color parameter, you can change the borderline color of each bar.

```
c=['blue','red',"blue','blue','blue','blue','blue','blue','blue','blue','red']
df_Iceland.plot(kind = 'bar', color = c, edgecolor = 'black')
```



Specialized Visualization Tools

Pie Charts

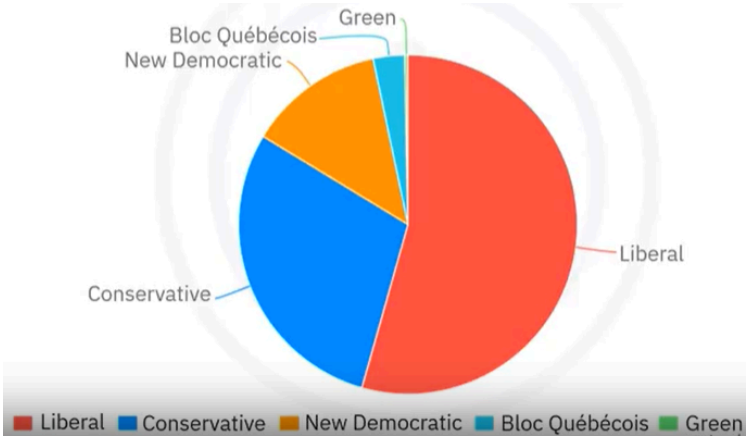
What will Learn

- Describe a pie chart with the help of an example.
- Explore the process of creating a pie chart using Matplotlib.

What is a Pie Charts?

A pie chart is a circular statistical graphic divided into segments to illustrate numerical proportion.

For example, here is a pie chart of the Canadian federal election. It represents the partywise percentage of seats won in the House of Commons.



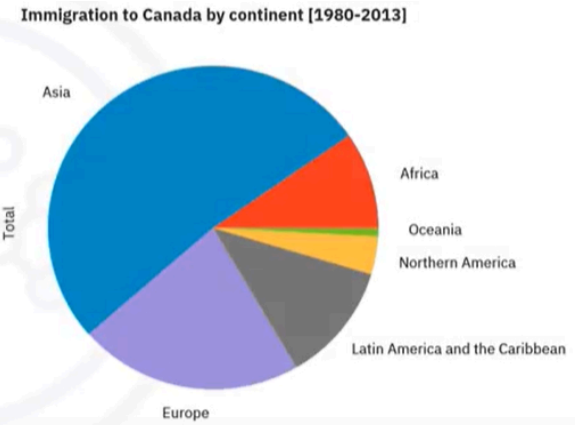
Generating a pie chart

We call the Pandas group by function on df_canada, and we sum the number of immigrants from the countries that belong to the same continent. Here is the resulting data frame, and let's name it df_continents. The resulting data frame has six rows, each representing a continent, and 35 columns representing the years from 1980 to 2013, plus the cumulative sum of immigration for each continent. Now, we're ready to start creating our pie chart.

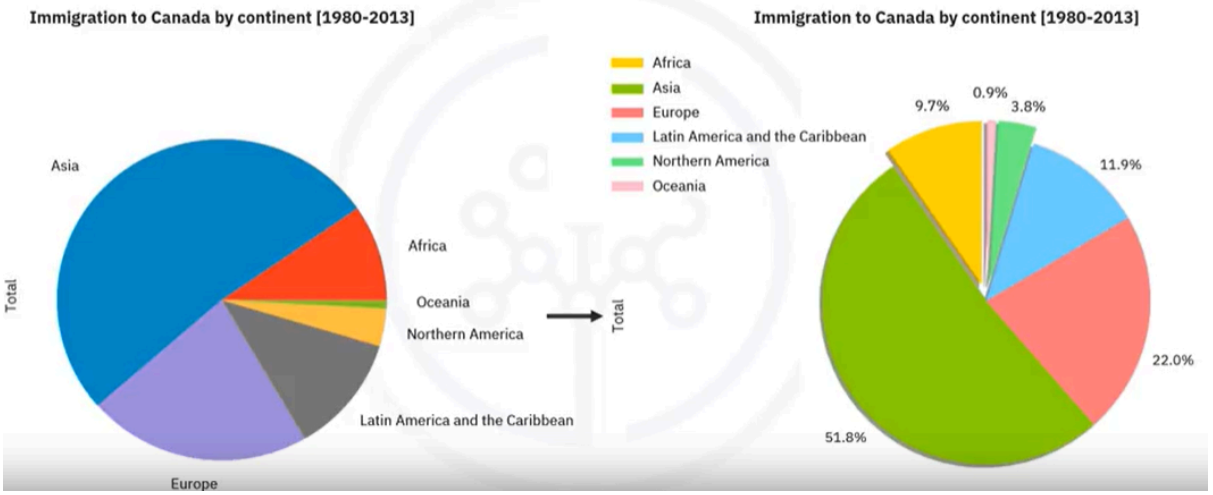
```
df_continents = df_canada.groupby('Continent', axis = 0).sum()
```

	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
Continent																					
Africa	3951	4363	3819	2671	2639	2650	3782	7494	7552	9894	...	27523	29188	28284	29890	34534	40892	35441	38083	38543	618948
Asia	31025	34314	30214	24696	27274	23850	28739	43203	47454	60256	...	159253	149054	133459	139894	141434	163845	146894	152218	155075	3317794
Europe	39760	44802	42720	24638	22287	20844	24370	46698	54726	60893	...	35955	33053	33495	34692	35078	33425	26778	29177	28691	1410947
Latin America and the Caribbean	13081	15215	16769	15427	13678	15171	21179	28471	21924	25060	...	24747	24676	26011	26547	26867	28818	27856	27173	24950	765148
Northern America	9378	10030	9074	7100	6661	6543	7074	7705	6469	6790	...	8394	9613	9463	10190	8995	8142	7677	7892	8503	241142
Oceania	1942	1839	1675	1018	878	920	904	1200	1181	1539	...	1585	1473	1693	1834	1860	1834	1548	1679	1775	55174

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


df_continents['Total'].plot(kind='pie')

plt.title('Immigration to Canada by
Continent [1980-2013]')

plt.show()
```



Pie chart: Explode property



A final point about pie charts, there are some strong critics who oppose using pie charts in any condition. They argue that pie charts do not display accurate data consistently. When it comes to depicting data consistently and communicating the point, bar charts perform significantly better.

Box Plots

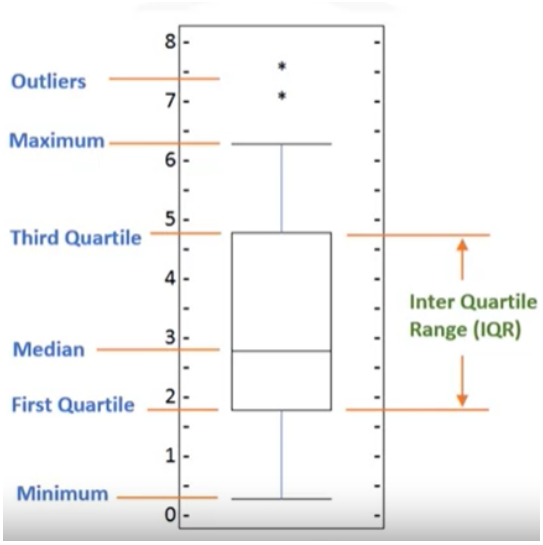
What will Learn

- Describe a box plot with the help of an illustration.
- Explain how to create box plots using Matplotlib

What is a box plot?

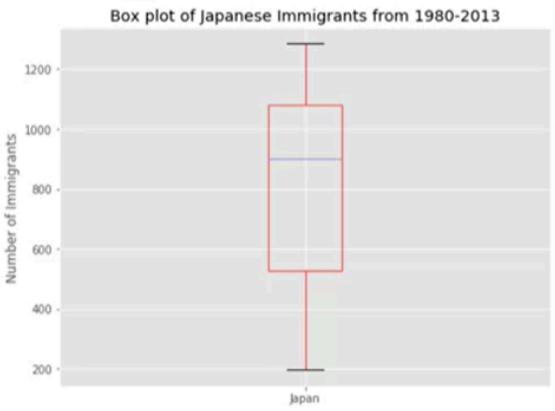
A box plot is a way of statistically representing the distribution of given data through five primary dimensions,

Minimum is the smallest number in the sorted data.
First quartile is the point 25% of the way through the sorted data, in other words, a quarter of the data points are less than this value.
Median is the median of the sorted data,
third quartile is the point 75% of the way through the sorted data. In other words, three quarters of the data points are less than this value,
maximum is the highest number in the sorted data.



Generating a box plot

```
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()
```



we create a new data frame on the data about Japan and we exclude the total column using the years variable. Then we transpose the resulting data frame in the correct format to create the box plot.

From this plot, we can verify that there are no outliers in this data. Also, we can see that the median is closer to the top, indicating more data concentration in the upper half.

Scatter Plots

What will Learn

- Describe a Scatter Plots with the help of an illustration.
- Explain how to create Scatter Plots using Matplotlib

What is a box plot?

A scatter plot is a type of plot that displays values pertaining to typically two variables against each other. Usually, it's a dependent variable that is plotted against an independent variable to determine if any correlation between the two variables exist.

For example, here's a scatter plot of income versus education. And by looking at the plotted data, one can conclude that an individual with more years of education is likely to earn a higher income than an individual with fewer years of education.

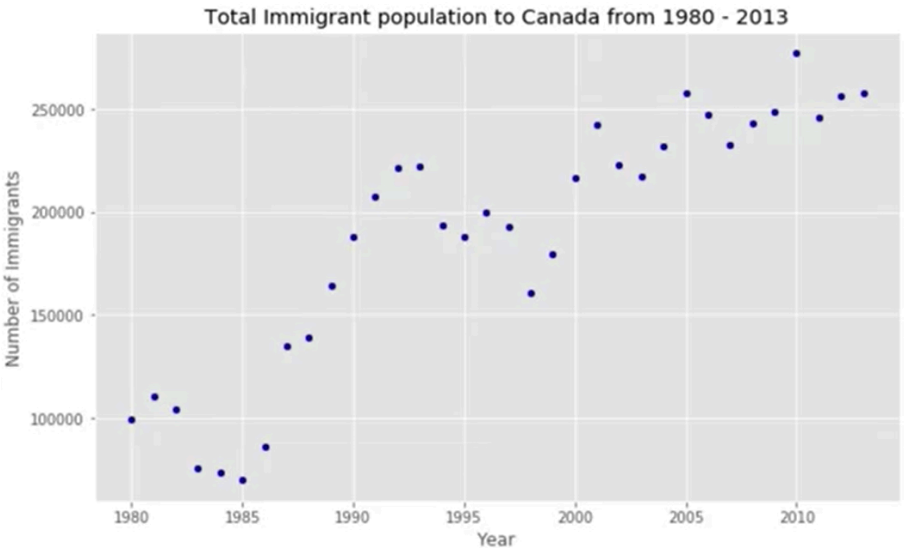
Generating a scatter plot

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

```
df_total.plot(
    kind='scatter',
    x='year',
    y='total',
)

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

plt.show()
```

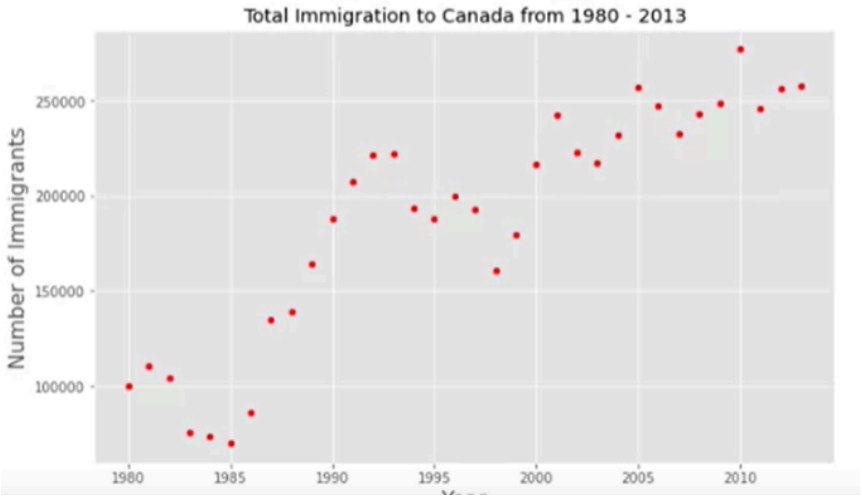
df_total

year	total
1980	99137
1981	110563
1982	104271
1983	75550
1984	73417
.	.
.	.

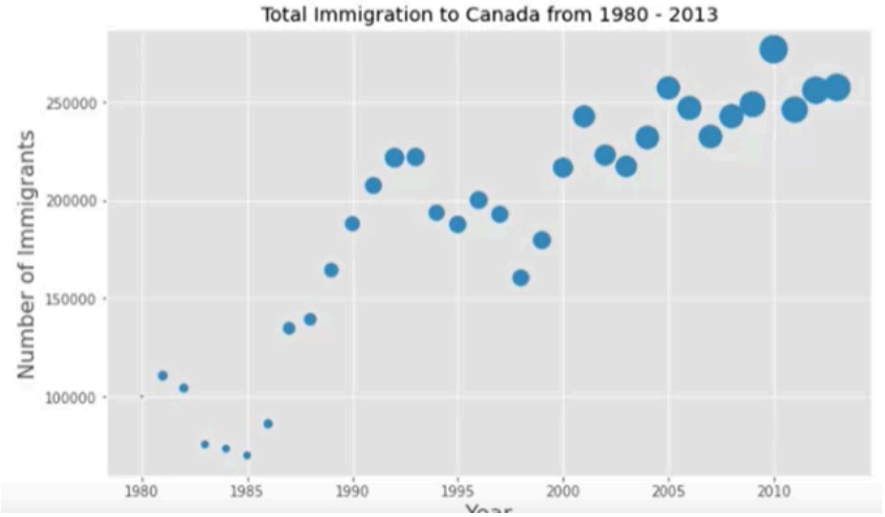


A scatter plot that shows total immigration to Canada from countries all over the world from 1980-2013. The scatter plot clearly depicts an overall rising trend of immigration with time.

Consider the use of the **color parameter**, which we have assigned a value of dark blue. You may like to pick a color of your choice from the available color palettes. For instance, see how the plot is in the color red.



Also use the **s parameter** to represent any third variable if you want to. Here we have included the total from the African continent to represent the size of each marker over the years. And it is evident that the number of immigrants has increased over these years as the size of the markers in the scatter plot is increasing.



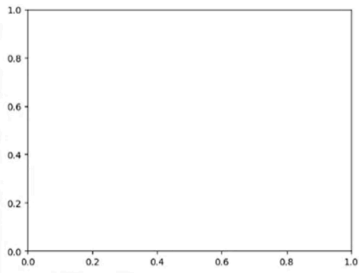
Plotting Directly with Matplotlib

What will Learn

- Explore various functions offered by Matplotlib for data visualization and plotting.
- Differentiate between data storytelling and data visualization.

Importing Libraries, figures and axes

```
import matplotlib.pyplot as plt
import numpy as np

import pandas as pd

# Create figure and axes
fig, ax = plt.subplots()
```



The subplot function and create a figure, the Canvas window and the axes. It's the area where the plot appears. The figure axes pair provides greater control over the figure or Canvas.

Line Plot and Scatter Plot

We'll create some synthetic data to generate the plot using NumPy years is equal to np.arange and passing 1980. 2014 to a range function will generate a 1D array of numbers 1980-2013 excluding 2014 immigrants equals np.random.randint will generate a 1D array of random integers, between 2,000-10,000 range. Size equals 34, specifies the number of elements in the array.

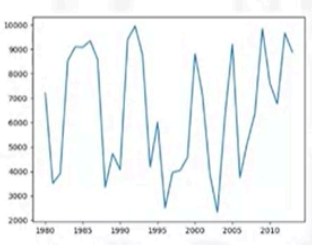
```
# Synthetic data
years = np.arange(1980, 2014)
immigrants = np.random.randint(2000, 10000, size = (34,))
```

X.plot function will create a **line plot**. It takes the x and y values as arguments corresponding to the data to be considered for the x-axis and the y-axis. X.plot years immigrants. Finally display the plot using plt.show function.

If you want to display this data as a **scatter plot** called the scatter function on x instead as shown here. X.scatter and past years and immigrants to it.

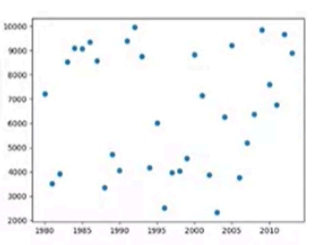
- Line Plot

```
#Create figure and axes
fig, ax = plt.subplots()

# Plot the line
ax.plot(years, immigrants)

# Display the plot
plt.show()
```



- Scatter Plot

```
#Create figure and axes
fig, ax = plt.subplots()

# Plot the line
ax.scatter(years, immigrants)

# Display the plot
plt.show()
```



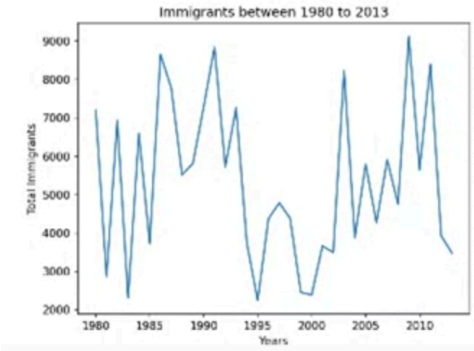
Customizing Attributes

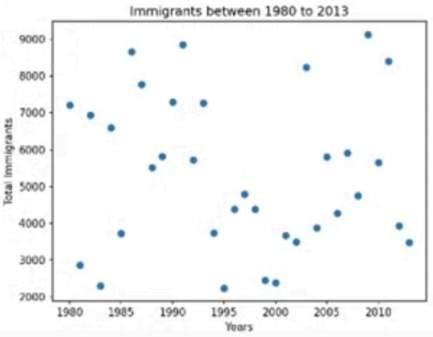
Now let's put a **title** to the plots. To do so, simply pass the title string to the plt.title function. plt.title immigrants between 1980 to 2013.

On both the plots, **label the axis** with plt.xlabel as years and plt.y label as total immigrants on both axes. You can apply the label entitled directly to the axes as shown here. Axes.set_title and pass the title string. Axes.set_ x-label or y-label. See now the plots have the title and the labels.

```
#add title
plt.title('Immigrants between 1980 to 2013')
#add labels
plt.xlabel('Years')
plt.ylabel('Total Immigrants')
```

```
ax.set_title('Immigrants between 1980 to 2013')
ax.set_xlabel('Years')
ax.set_ylabel('Total Immigrants')
```



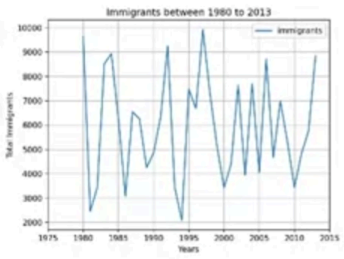


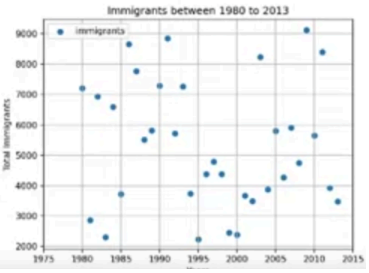
x-lim and **y-lim** functions, you can set the limits on the x and y-axis.

To improve the readability, you can enable **grid lines** with grid function and pass it with true, like x.grid, true.

Legend function will include the legend to your plot.

```
#Limits on x-axis
plt.xlim(1975, 2015)
#or ax.set_xlim()

#Enabling Grid
plt.grid(True)
#or ax.grid()

#Legend
plt.legend(["Immigrants"]) #or ax.legend()
```





Notice that the x-axis now starts with 1975 and ends with 2015, and now it shows the grid at the background of the plot.

Marker styles, colors and sizes

You can customize the plot with line styles, marker styles, and color customization options to achieve the desired visual effects.

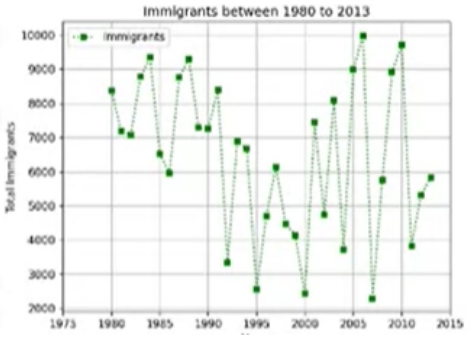
With **marker**, select a style to represent data points in your plot, like **S** for **square** and **0** for **dots**.

For line plot, the marker is marker size, while in scatter it's just S.

Similarly, you select a **color** and a **size** for it.

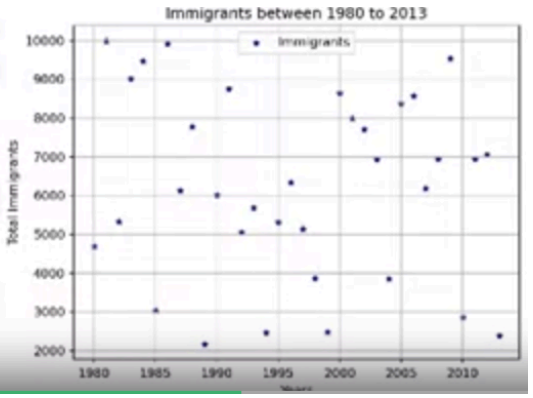
With the **line style** parameter, you can select different line styles, such as **solid, dashed, or dotted lines**.

```
# Customizing the appearance of Plot
ax.plot(years, immigrants,
        marker='s',
        markersize=5,
        color='green',
        linestyle="dotted")
```



Notice the use of **LOC** in the legend parameter to specify the location where the legend is placed on the figure.

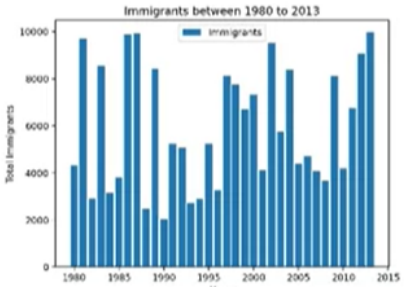
```
# Customizing Scatter Plot
ax.scatter(years, immigrants,
           marker='*',
           s = 20,
           color='darkblue')

#Legend at upper center of the figure
ax.legend(["Immigrants"], loc='upper center')
```



Bar plot, histograms and Pie

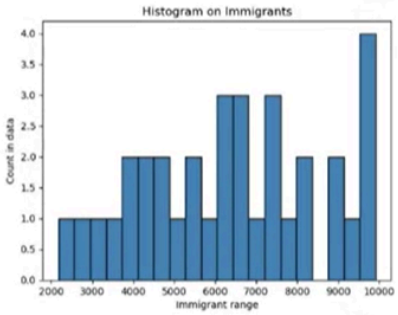
Bar plot to represent the number of immigrants for each year. Ax.bar function creates a bar plot, passing years corresponding to the x-axis and the immigrants to the y-axis. The height of each bar represents the number of immigrants for that year.

```
# Plot the bar
ax.bar(years, immigrants)
```



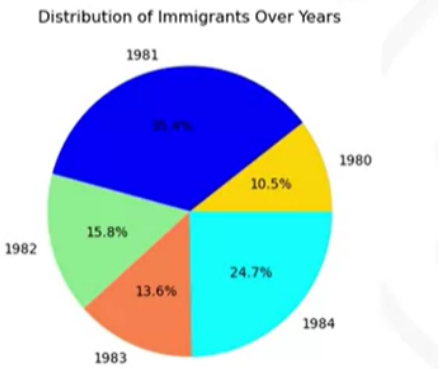
Ax.hist generates a **histogram** with bins set to 20. Notice the use of edge color and color parameters to specify the border and bar column of the histogram.

```
#histogram on immigrants
ax.hist(immigrants, bins=20, edgecolor='black',color='steelblue')
```



Ax.pie will generate a **pie** on the axis. Here we're plotting a pie for only five years, 1980 to 1984. Colors you are assigned to each pie slice has a list of colors and labels are set is years. Auto PCT displays the percentage of immigrants with one decimal point.

```
#Pie on immigrants
ax.pie(immigrants[0:5], labels=years[0:5],
       colors = ['gold','blue','lightgreen','coral','cyan'],
       autopct='%1.1f%%')
```



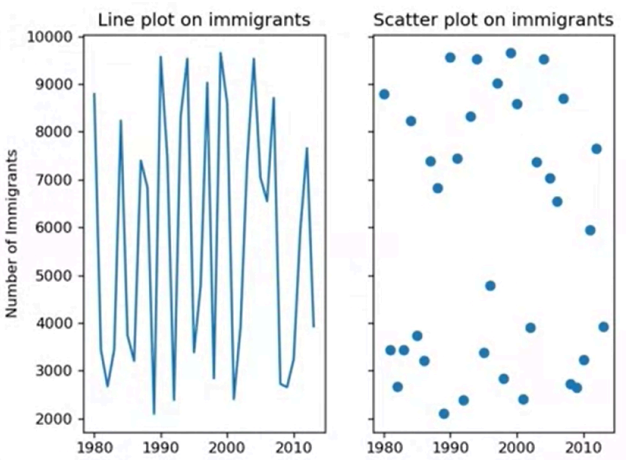
Multiple Plots and sub-plotting

how to display more than one plot on the same figure and specify the number of rows and columns to be created to the subplots function.

For instance, **let's create a line and scatter plot in one row**, plt.subplots and pass one to it. Both the subplot will be sharing the same y-axis as the data in the y-axis is the same, so assign the Sharey parameter as true. Now you have two axes, axs with index zero and axs with index one, and you can plot your plots as you did earlier.

Axs0.plot function for the line **axs1.scatter** for a scatter plot. And see you have created two plots together.

```
# Create a figure with two axes in a row
fig, axs = plt.subplots(1, 2, sharey=True)

#Plotting in first axes - the left one
axs[0].plot(years, immigrants)
axs[0].set_title("Line plot on immigrants")

#Plotting in second axes - the right one
axs[1].scatter(years, immigrants)
axs[1].set_title("Scatter plot on immigrants")
```



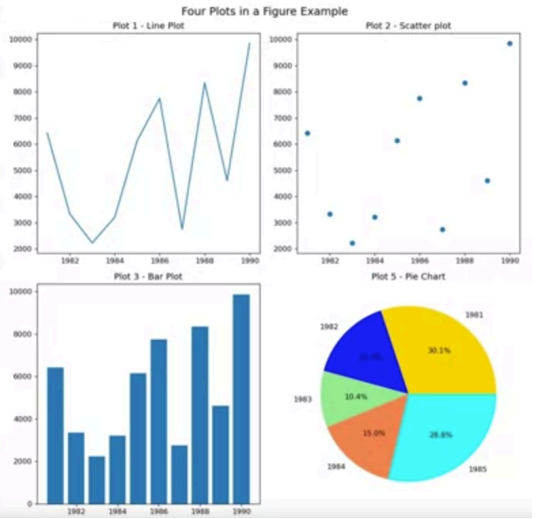
Alternatively, we can use the **add underscore subplot** function. It takes three arguments, first, the number of rows, second, columns and the last is an index of the subplot. You can create different axes on the figure and add subplots as shown here. Ax 1 equals figure.add_subplot 2, 2, 1. This one means the first axes on the two-by-two divided figure. Then on these axes, create the plot like an ax1.plot.

- Three Arguments
  - Number of rows
  - Columns
  - Index of the subplot

```
# Add the first subplot (top-left)
axs1 = fig.add_subplot(2, 2, 1)
#Plotting in first axes - the left one
axs1.plot(total_immigrants)
axs1.set_title("Plot 1 - Line plot")
```

Similarly, you can plot all the plots you want on the four axes. Like this. You have all your plots on this figure.

```
# Add the second subplot (top-right)
ax2 = fig.add_subplot(2, 2, 2)
ax2.scatter(years, immigrants)
ax2.set_title('Plot 2 - Scatter plot')

# Add the third subplot (bottom-left)
ax3 = fig.add_subplot(2, 2, 3)
ax3.bar(years, immigrants)
ax3.set_title('Plot 3 - Bar Plot')

# Add the fourth subplot (bottom-right)
ax4 = fig.add_subplot(2, 2, 4)
ax4.pie(immigrants[0:5], labels=years[0:5],
        colors = ['gold','blue','lightgreen','coral','cyan'],
        autopct='%1.1f%%')
ax4.set_aspect('equal')
ax4.set_title('Plot 5 - Pie Chart')
```



Data storytelling and Data visualisations

These are two different terms and serve different purposes.

Data storytelling:

- the art of storytelling that involves creating a narrative around the data.
- It presents a compelling and engaging story.

Data visualisation:

- an important aspect of data storytelling and evolves
- creating informative charts to understand and explore patterns, trends, and relationships within the data.It brings data to life

https://www.forbes.com/sites/brentdykes/2016/03/31/data-storytelling-the-essential-data-science-skill-everyone-needs/

Summary: Basic and Specialized Visualization Tools

Congratulations! You have completed this module. At this point in the course, you know:

- A pie chart is a circular statistical graphic, divided into segments, to illustrate numerical proportion.
- The process of creating a pie chart involves importing Matplotlib to represent a large set of data over a period of time.
- A box plot is a way of statistically representing given data distribution through five main dimensions.
- The five main dimensions are minimum, first quartile, median, third quartile, and maximum.
- You can create a box plot using Matplotlib.
- A scatter plot displays values pertaining to typically two variables against each other.
- The process of creating a scatter plot involves importing Matplotlib to visualize a large set of data.
- Matplotlib is a versatile plotting library that offers a flexible interface for creating various types of plots.
- Matplotlib's Pyplot module offers a convenient way to create and customize plots quickly.
- Data Storytelling is the 'art of storytelling' that involves creating a narrative around the data.
- Data visualization is an important aspect of data storytelling and involves creating engaging visuals.