# Chapter 3

# Data Visualization

# Introduction

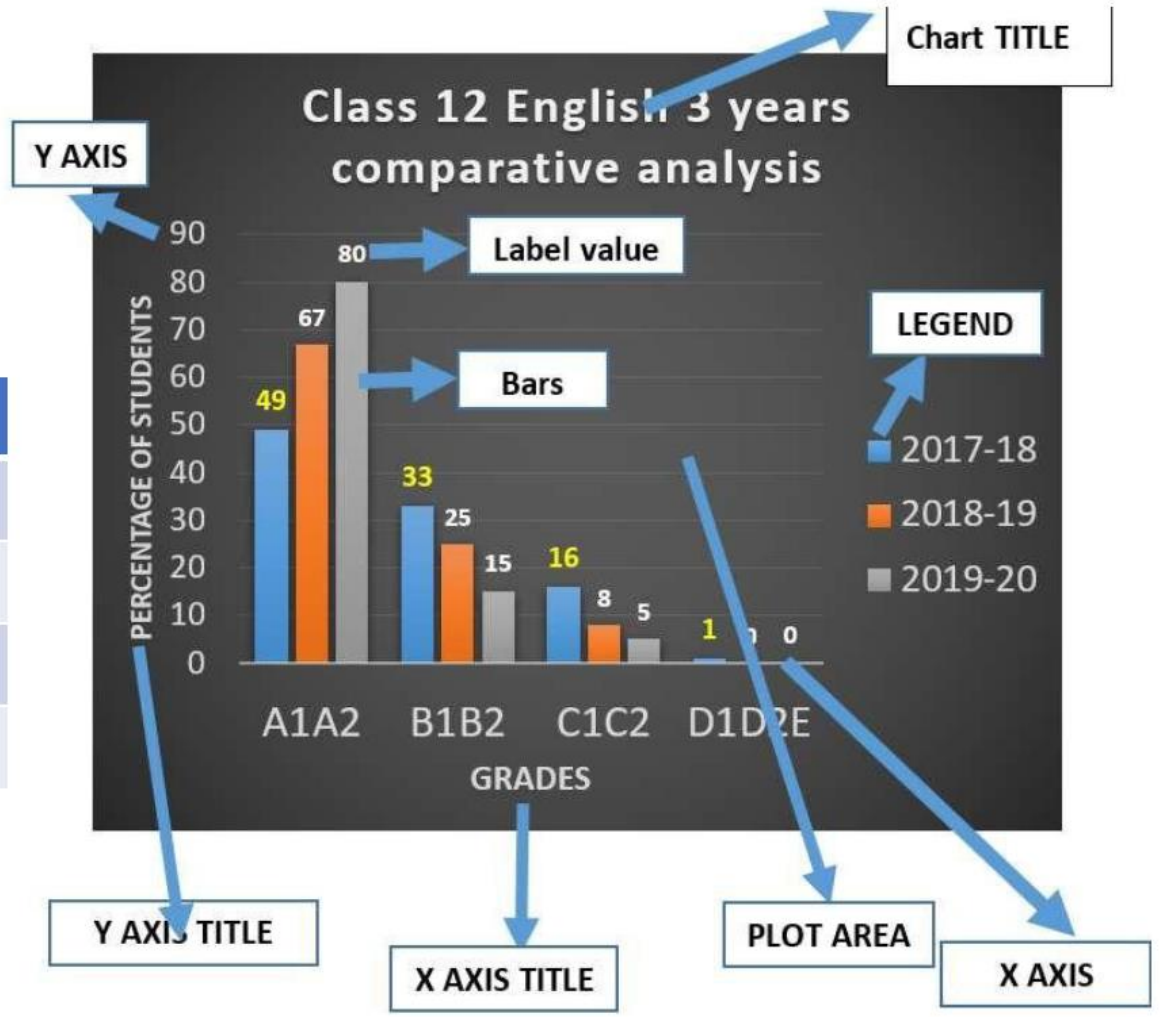
- Visualization of data refers to the Graphical Representation of Data. When the data is represented graphically by means of line chart, bar charts etc. recognition of pattern of changes, correlation, trends, and comparison between different sets of data becomes easy. In this way Visualization helps the management people to take policy decisions. Data visualization is mostly done for statistical data Some examples of such data are:

- Result of surveys on various topics
- Historical sales data of a product
- Literacy rates of different areas
- Growth rates among children of different groups etc.

- There are many ways by which data can be represented graphically these are **Scattered charts, Line charts, Bar Charts, Pie Charts, Histograms, Box Plots** etc.

# VARIOUS COMPONENTS OF CHART

| ENGLISH | | | | |
|---|---|---|---|---|
| | A1A2 | B1B2 | C1C2 | D1D2E |
| 2017-18 | 49 | 33 | 16 | 1 |
| 2018-19 | 67 | 25 | 8 | 0 |
| 2019-20 | 80 | 15 | 5 | 0 |



# How to start making charts in Pandas

- **Matplotlib** is the whole Python package/library used to create 2D graphs and plots by using Python scripts.
- **Pyplot** is a module in matplotlib which supports a very wide variety of graphs and plots. It helps save images in several output formats (PNG, PS and others).
- Import mathplotlib
- Import mathplotlib.pyplot
- **Import mathplotlib.pyplot as plt**

  **OR**
- **Import mathplotlib.pyplot as pt**
- This module contains all the functions required to plot the graphs.

# CUSTOMIZING PLOTS
# List of Pyplot functions to customize plots

- 1. grid - Configure the grid lines.

  **Ex**: **plt.grid(True): shows grid line**
- 2. title - Set a title for the axes.

  **Ex: plt.title("Title name")**
- **3. Adding labels:**

  **a)** xlabel - Set the label for the x-axis.

  **b)** ylabel - Set the label for the y-axis. **plt.xlabel(" x axis label name") plt.ylabel("y axis label")**
- 4. legend - Place a legend on the axes.

  **Ex: plt.legend()**
- 5. savefig - Save the current figure.

  **Ex: plt.savefig(" path&filename")**
- 6. show - Display the chart.

  **Ex: plt.show()**

# Creating Scatter Charts

- Scatter Chart is chart in which only data points are plotted. scatter() function available in pyplot interface is used to draw scatter charts:

**plt.scatter(Xvalues,Yvalues)**

show() function in pyplot interface used to see the plot:

**plt.show()**

- **Marker Styles in Charts :**

To set the marker style in scatter charts use marker argument in scatter() function.

**Plt.scatter(Xvalues,Yvalues, marker = MarkerStyle)**

**Markerstyle Characters**

| | | | |
|---|---|---|---|
| . | Point | s | Square |
| o | Circle | + | Plus |
| D | Diamond | D | Thin Diamond |
| H | Heagon-1 | H | Hexagon-2 |
| * | Star | p | Pentogon |

# Example of Scatter Chart

- import matplotlib.pyplot as plt
- X = [590,540,740,130,810,300,320,230,470,620,770,250]
- Y = [32,36,39,52,61,72,77,75,68,57,48,48]
- Colarr=['r', 'b', 'm', 'k', 'c', 'b', 'w']
- plt.scatter(X,Y, c=Colarr,  marker= "p", s=9)
- plt.show()

# Creating Scatter Charts

- **Applying Colors in Charts :** To set the color of markers in scatter charts use color argument in scatter() function.

**plt.scatter(Xvalues,Yvalues, Color =ColorCode)**

**Color Codes**

| | | | |
|---|---|---|---|
| **b** | **Blue** | **y** | **Yellow** |
| **g** | **Green** | **k** | **Black** |
| **r** | **Red** | **c** | **Cyan** |
| **m** | **Magenta** | **w** | **White** |

- **Showing Legend with Charts :** Legend is the small box that gives the details about colors and patterns used in charts for different values. To show the legend legend() function is used in charts.

**plt.legend (loc= "Location string")**

# Creating Scatter Charts

- **Legend Location Strings**

| | |
|---|---|
| **Best** | **Best suitable place** |
| **Upper right** | **Upper right corner of the chart** |
| **Upper left** | **Upper left corner of the chart** |
| **lower right** | **lower right corner of the chart** |
| **Lower left** | **lower left corner of the chart** |

Apart from the legend() function we also need to specify the label for each of the value that will be used in legend. This is done using label argument while plotting the chart using scatter() or any other function:

**plt.scatter(Xvalues,Yvalues, label = "Any String")**

# Adding Title, X-axis, Y-axis and Legend

- import matplotlib.pyplot as pt
- X=[2018, 2019, 2020, 2021]
- Y=[100, 70, 65, 90]
- pt.scatter(X, Y, c='m', marker='D', s=300, label="Sales")
- pt.xlabel("Years")
- pt.ylabel("Sales of TV")
- pt.title("Yearly sales of television")
- pt.legend(loc="best")
- pt.show()

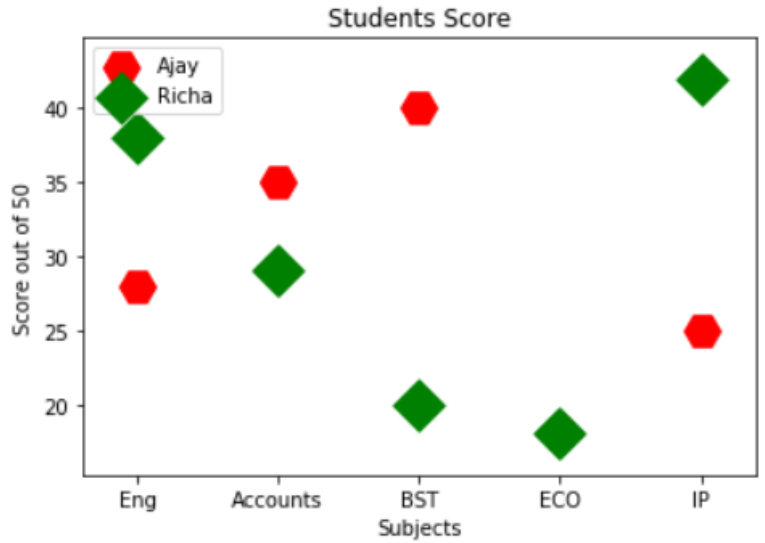
# Creating Scatter Charts

- **Setting Marker Size in Scatter Charts:** Size of marker in scatter Chart can be changed using argument s. The size is given in points. A point is equals to 1/72 inches. Size of marker in scatter chart is given in form of size2 e.g. if you want a ¼ inch (72/4 = 18 points) marker you have to specify the value of s to be 182 = 324 points.

**e.g. plt.scatter(Xvalues,Yvalues, s= 324)**

- **To create scatter chart to display the score of two students in 5 subjects**

```
import matplotlib.pyplot as plt
import numpy as np
xvalues=np.array([1,2,3,4,5])
 score1= np.array([28,35,40,18,25])
score2= np.array([38,29,20,18,42])
plt.scatter(xvalues,score1, s=324,color="r",marker="H", label="Ajay")
plt.scatter(xvalues,score2,s=324, color="g", marker="D", label= "Richa")
 plt.title("Students Score")
 plt.xlabel("Subjects")
 plt.ylabel("Score out of 50")
plt.xticks(ticks=[1,2,3,4,5], labels=['Eng','Accounts','BST','ECO','IP'])
 plt.legend(loc="best")
plt.show()
```



# Practice time:

- Q1)  Create a scatter chart comparing the height and weight for a group of students. Include xlabels, ylabels, legend and give title to the chart.

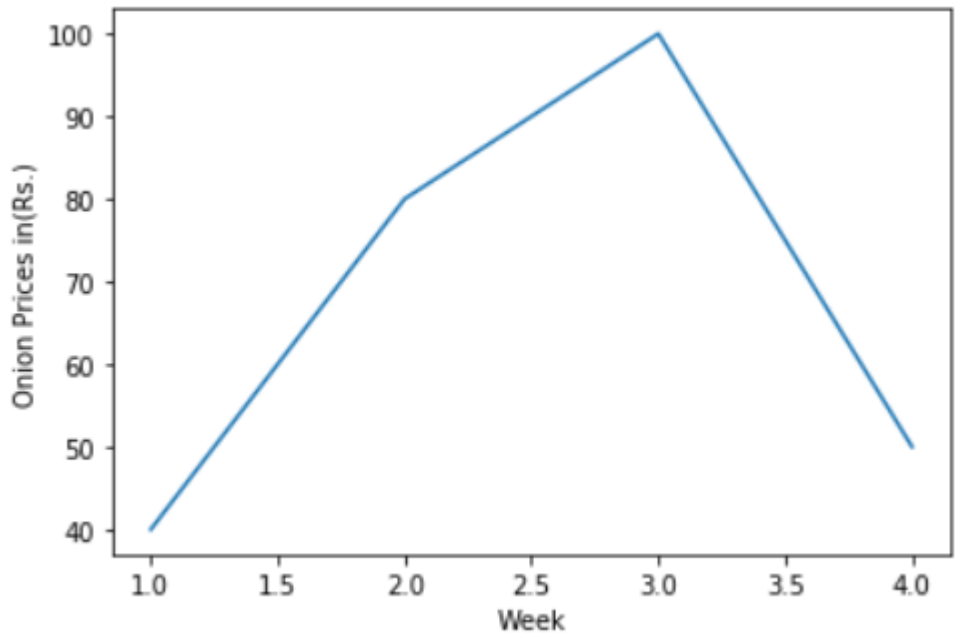
# Line Charts

- Line charts are charts where data points are plotted and they are connected with straight line segments. plot() function in pyplot interface is used to create line charts.

**plt.plot(Xvalues,Yvalues)**

- **A simple example of Line chart plot:**

```
Import matplotlib.pyplot as plt
week=[1, 2, 3, 4]
prices=[40, 80, 100, 50]
plt.plot(week, prices)
plt.xlabel("Week")
plt.ylabel("Onion Prices in(Rs.)")
plt.show()
```



# Applying various settings

- Following arguments are used apart from X and Y axis values.
  - **linecolor/ marker color:** to change the color code of the plotted line we use the following function.

**plt.plot( xvalue, yvalue, colorcode)**

  - **linewidth:** to set the line width we include linewidth=2 or 4 in the plot function.
  - **linestyle or ls :** to set the line style (solid, dashed, dashdot, dotted)
  - **marker :** marker style as used in scatter chart
  - **markersize :** in points (not in points2 as in scatter charts)
  - **markeredgecolor :** color code (same as scatter charts)

# Example:

```
import matplotlib.pyplot as plt

week=[1, 2, 3, 4]

prices=[40, 80, 100, 50]

plt.plot(week, prices, 'c', linewidth=4, ls="dotted", marker="+",
markersize=200, markeregdecolor="b")

plt.xlabel("Week")

plt.ylabel("Onion Prices in(Rs.)")

plt.show()
```

# Practice Time:

- Plot a line chart showing marks secured by students in 10 unit tests. Include, all the line settings learnt in the class today and include title and legend in the chart.

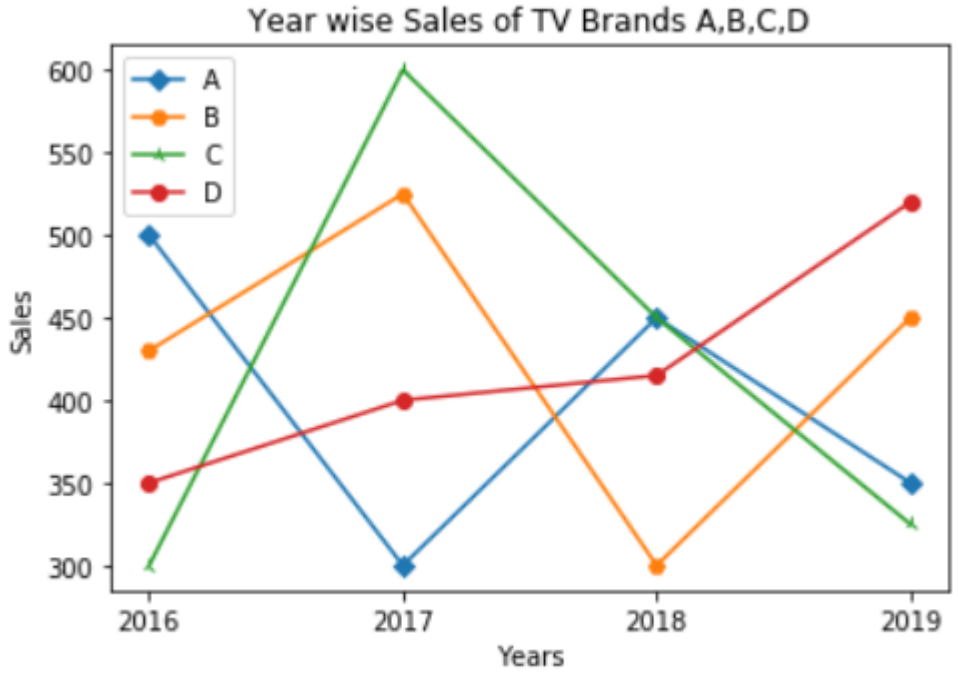
# Specifying plot size and Grid

- Figure size: to change the size of plot we may set up the figure size:
- matplotlib.pyplot.figure(figsize= width, length)
- Or plt.figure(figsize=(15, 7))
- Grid: if you need to see the grid
- plt.grid(True)

# Line Chart

- **To Create Line Charts for plotting Year wise Sales of TV Brands A,B,C,D**

```
import matplotlib.pyplot as plt
import numpy as np
a= np.array([500,300,450,350])
b= np.array([430,525,300,450])
c= np.array([300,600,450,325])
d= np.array([350,400,415,520])
years=[2016,2017,2018,2019]
xvalues=np.array([1,2,3,4])
plt.plot(xvalues,a,marker='D',label="A")
plt.plot(xvalues,b, marker="H", label="B")
plt.plot(xvalues,c, marker="2",label="C")
plt.plot(xvalues,d, marker="o", label="D")
plt.xticks(xvalues,years)
plt.title("Year wise Sales of TV Brands A,B,C,D")
plt.ylabel("Sales")
plt.xlabel("Years")
plt.legend(loc="upper left")
plt.show()
```



# Bar Charts

- **Bar Charts or Column Charts** are charts where numerical data is plotted in form of vertical bars. Sometimes a bar chart may also be plotted horizontally. Bar Chart is normally used to compare the values of different groups of data. Bar Chart in python is plotted using bar() function in pyplot interface.

**plt.bar( Xvalues, Yvalues, width, color,label)**

Width of bar is given in units and this unit is adjusted automatically according to available area for plotting. The width by default is 0.8 units.

- **Multiple Bar Charts :** Single Bar Charts are used to plot one set of data while Multiple Bar Charts are needed when more than one set of data are to be plotted. In case of multiple bar charts single bar chart is plotted for each individual set of data one by one. We need to adjust Xvalue parameter for each set by adding bars width to Xvalue. Here numpy array plays an important role. Since numpy array supports vectorized operation the width of previous bars is be added to the Xvalue parameter for each subsequent bar to be plotted.

**plt.bar( Xvalues, Yvalues, width, color,label)**

# Bar Charts

- Program to plot a bar chart for the sales of TV of brand A in last Four years
- import matplotlib.pyplot as plt
  import numpy as np
  a= np.array([500,300,450,350])
  years=[2016,2017,2018,2019]
  xvalues=np.array([1,2,3,4])
  plt.bar(xvalues, a, width=0.1, label="A", color='r')
  plt.xticks(xvalues, years)
  plt.title("Year wise Sales of TV Brand A")
  plt.ylabel("Sales")
  plt.xlabel("Years")
  plt.legend(loc="upper left")
  plt.show()

# Bar Charts (Contd..)

- To give multiple width and multiple color to bars.
- Width=[0.5, 0.7, 0.8, 0.9]
- color=[‘r’, ‘c’, ‘b’, ‘k’]
- **plt.bar(xvalues, a, width=[0.5, 0.7, 0.8, 0.9], label="A", color=['r‘, ‘b’, ‘c’, ‘m’])**

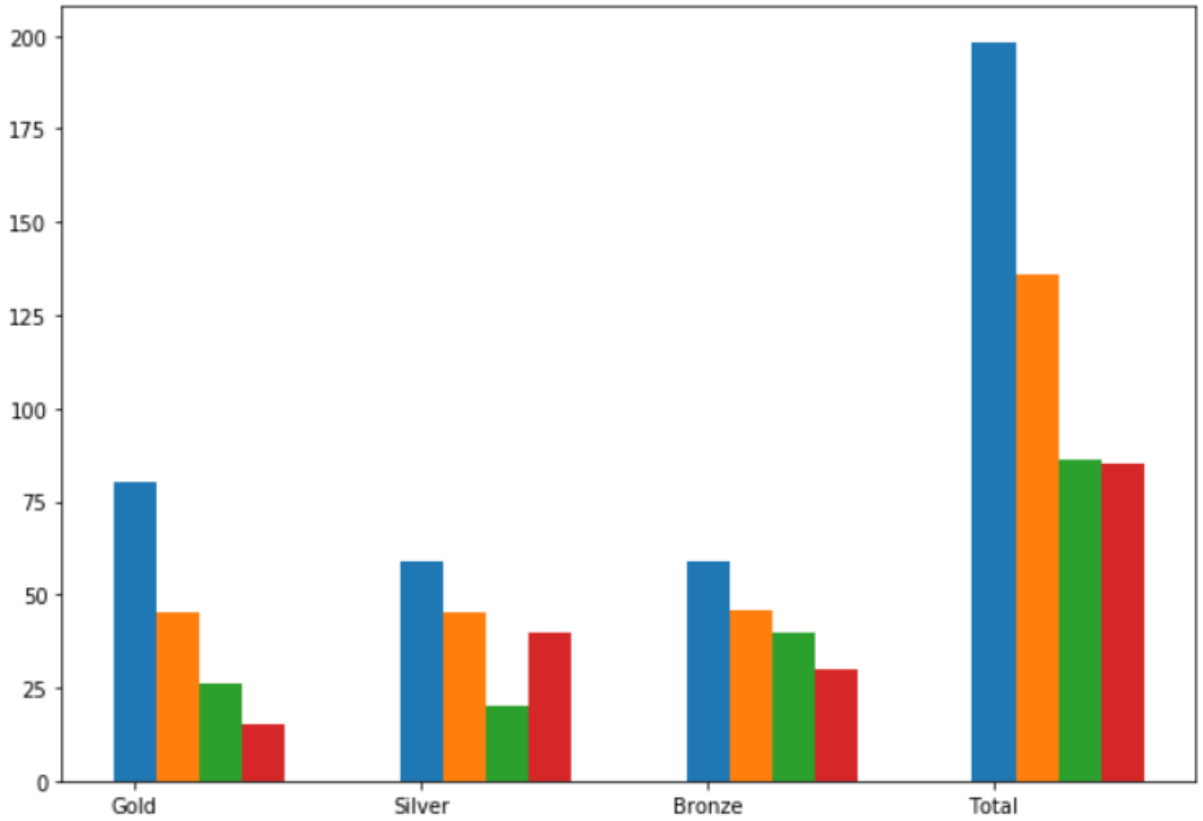
# Practice Time:

- Plot a bar chart showing population of 4 different cities ( Delhi, Chennai, Mumbai and Bangalore) including different width and different color for bars.

# Bar Charts

- Creating multiple Bar charts:

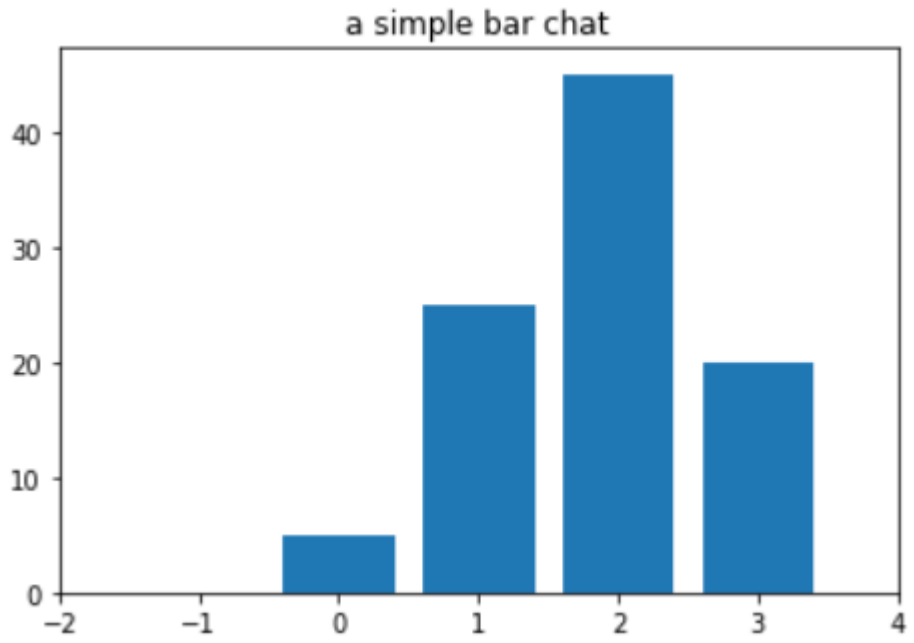
```
import mathplotlib.pyplot as plt
import numpy as np
plt.figure(figsize=(10, 7))
Info=['Gold', 'Silver', 'Bronze', 'Total']
Australia=[80, 59, 59, 198]
England=[45, 45, 46, 136]
India=[26, 20, 40, 86]
Canada=[15, 40, 30, 85]
X=np.arange(len(info))
plt.bar(Info, Australia, width= .15)
plt.bar(X+0.15, England, width=.15)
plt.bar(X+0.30, India, width=.15)
plt.bar(X+0.45, Canada, width=.15)
plt.show()
```



# Settings Xlimits and Ylimits

- Xlim() and ylim() functions are used to set limits for X-axis and Y-axis.
- Syntax:
- <matplotlib.pyplot>.xlim(<xmin>, <xmax>)
- <matplotlib.pyplot>.ylim(<ymin>, <ymax>)
- Example:

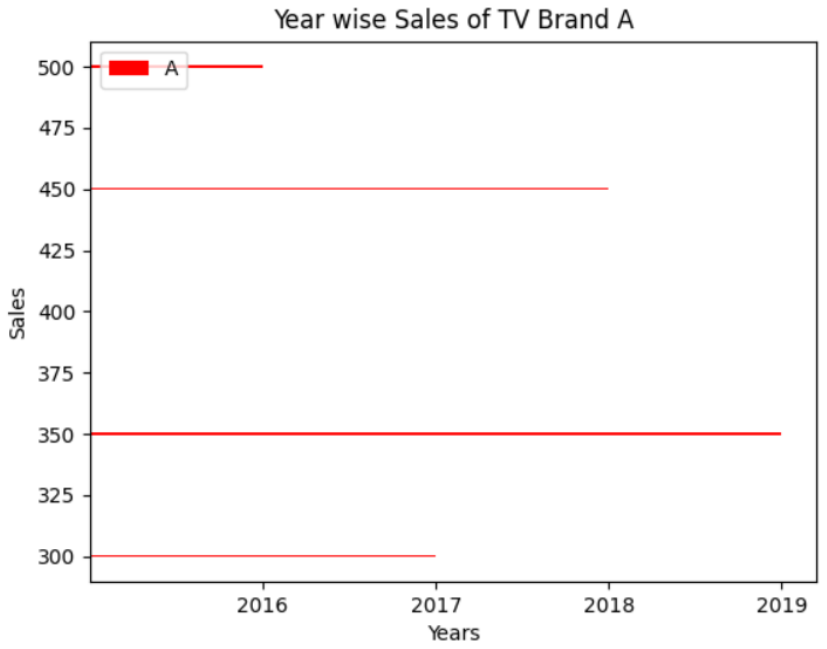
```
X=np.arange(4)
Y=[5.,  25., 45., 20.]
plt.xlim(-2.0, 4.0)
plt.bar(X, Y)
plt.title("a simple bar chat")
plt.show()
```



# Horizontal Bar chart

- To create horizontal bar chart, we will use barh() function, in place of bar().
- The x-axis of bar() will become y-axis for barh() and vice versa.

```
import matplotlib.pyplot as plt
import numpy as np
a= np.array([500,300,450,350])
years=[2016,2017,2018,2019]
xvalues=np.array([1,2,3,4])
plt.barh(a, xvalues,label="A", color='r')
plt.xticks(xvalues, years)
plt.title("Year wise Sales of TV Brand A")
plt.ylabel("Sales")
plt.xlabel("Years")
plt.legend(loc="upper left")
plt.show()
```



# Practice Time:

- Write a program to plot a horizontal bar chart from the height of some (5) students.

# Saving a figure

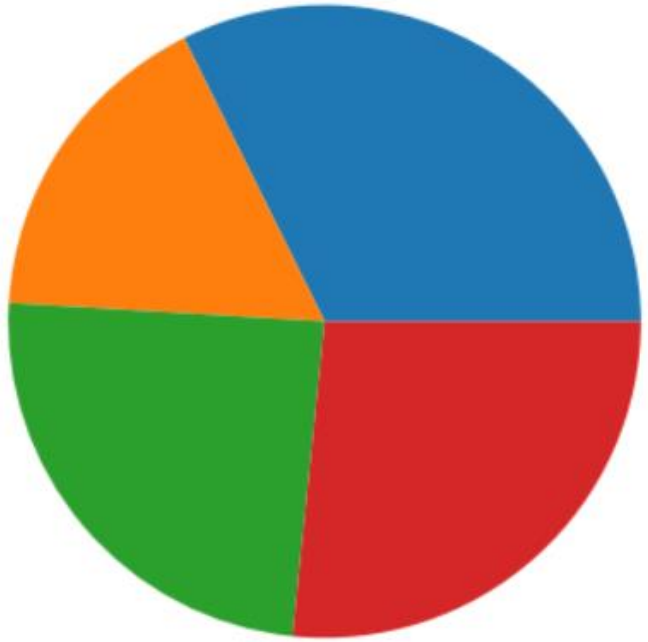
- Savefig() is used to save a plot created using pyplot functions for later use or for keeping records.
- <matplotlib.pyplot>. savefig(<string with file name and path>)
- We can save the figures in popular formats such as, pdf, png, eps, etc.
- Examples:
- Plt.savefig("multibar.pdf")  # it will save in current directory
- Plt.savefig("C:\\data\\multibar.pdf)  # it will save the plot at the given location.

# Pie Charts

- Pie chart is the circular statistical graphic, which is divided into slices. It is typically, used to show parts to the whole and often a % share.
- pie() function of pyplot is used for creating pie charts.
- The pie() function only plots a single data range only.
- The default shape of pie chart is oval, which can be changed to circle by using axis() of pyplot and setting it to equal.
- matplotlib.pyplot.axis("equal")

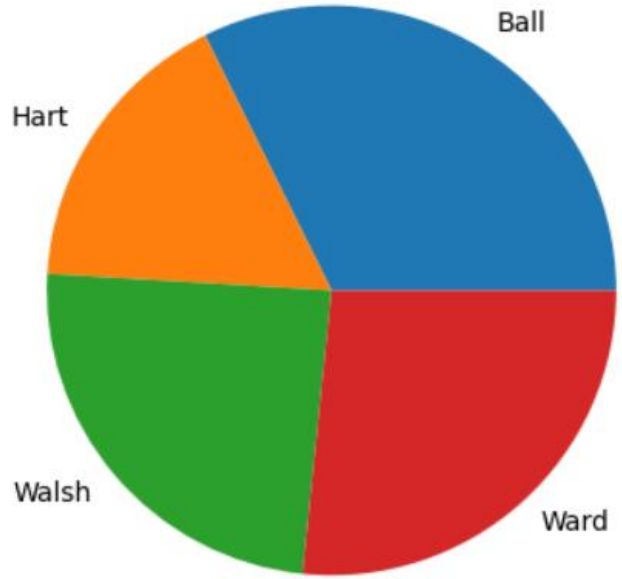
# Pie charts

- Example: Our school houses have collected amount for PM charity fund.
  To plot a pie chart from this data, we using the contribution in thousands:
- contri=[17, 8.8, 12.78, 14]
- plt.pie(contri)
- plt.show()



# Labels of Slices of Pie

- To mark the slices we need to name them within labels.
- contri=[17, 8.8, 12.78, 14]
- Houses=[“Ball”, “Hart”, “Walsh”, “Ward”]
- plt.pie(contri, labels=Houses)
- plt.show()



# Practice Time:

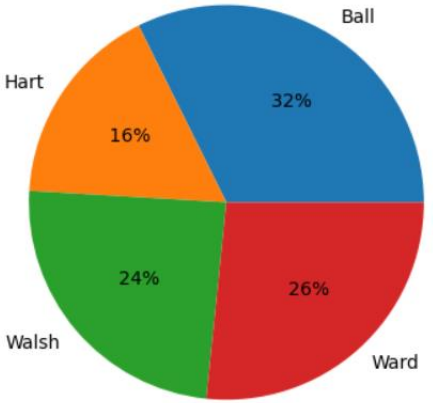
- Our school celebrated volunteering week where each section of class XII dedicated a day for collecting amount for charity being supported by the school. Section A volunteered on Monday, B on Tuesday, C on Wednesday and so on. There are six sections in class XII. Amounts collected by sections A to F are 8000, 12000, 9800, 11200, 15500, 7300.
- Write a program to create a pie chart showing collection amount section wise.

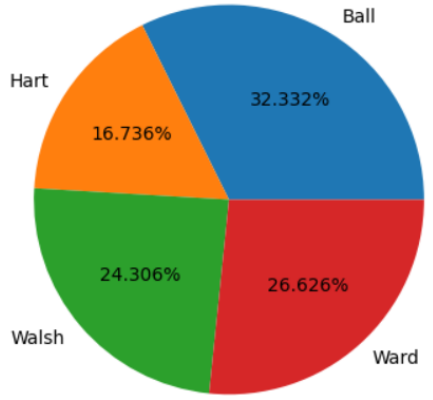
# Adding Formatted Slice Percentage to Pie

- To view percentage of share in pie chart we use the argument, autopct with a format string such as, “%1.1F%%” or “%3d%%”,
- “%[flags][width][.precision] type”
- % : is a special character that signifies that it is a special string.
- Width: total number of characters to be displayed(digits before or after decimal point)
- Flag: is used to pad the value being displayed with zeros when digits of number is less than width.
- Precision: number of digits after decimal number.
- Type: it specifies the type of value, d is for integer and f is for floating values.
- % sign: two percentage signs are needed to print one percentage sign.

# Example:

```
import matplotlib.pyplot as plt
contri=[17, 8.8, 12.78, 14]
Houses=["Ball", "Hart", "Walsh", "Ward"]
plt.axis("equal")
plt.pie(contri, labels=Houses, autopct="%3d%%")
plt.show()
Or
plt.pie(contri, labels=Houses, autopct="%05.3f%%")
```



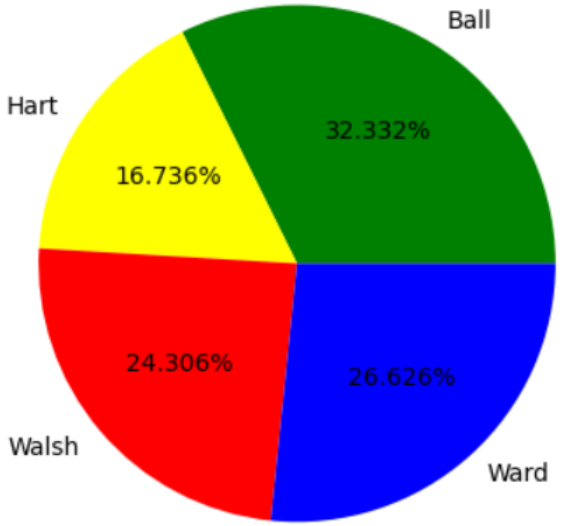
# Changing color of the slides

- plt.pie(contri, labels=Houses,colors=['Green', 'Yellow', 'Red', 'Blue'], autopct="%05.3f%%")
- **Exploding a Slice:**
- Emphasizing on one slice or more slices and pulling them out is known as Exploding a Slice.

For example: if you want to explode first slice, then we will include explode=[0.2, 0, 0, 0] in pie function

If you want to explode first and last slice then,

explode=[0.2, 0, 0, 0.5]



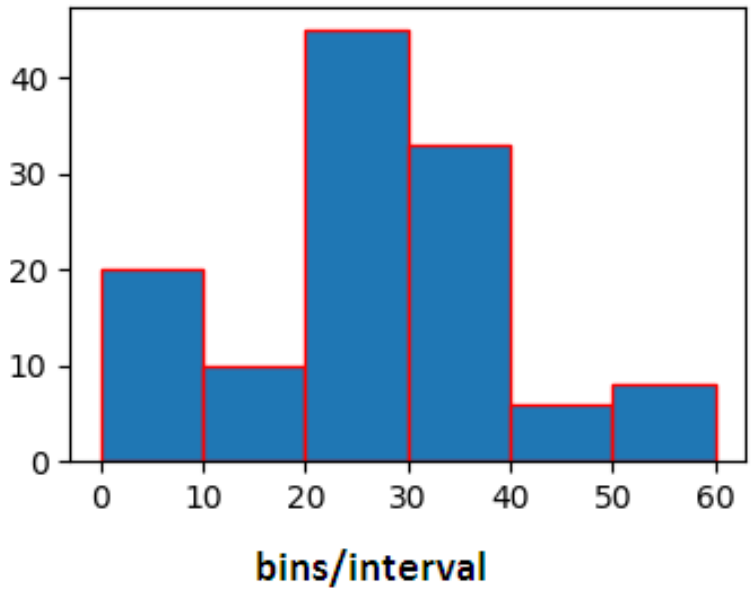
# Histogram

- Histogram is a plot that is used to show frequency distribution of a set of continuous data.

- Histogram provides a visual interpretation of numerical data by showing the number of datapoints that fall within a specified range of values("bins").It is similar to a vertical bar graph but without gaps between the bars.
- hist(columns=<column_name>, bins=10)
- Parameters:
- X=array or sequence of arrays to be plotted  on histogram.
- Bins= (int)3 optional.
- Cumulative= bool, optional
- Histtype={ bar, barstacjked, step, stepfilled}, optional.
- Orientation= horizontal or vertical, optional.

# Simple Histogram

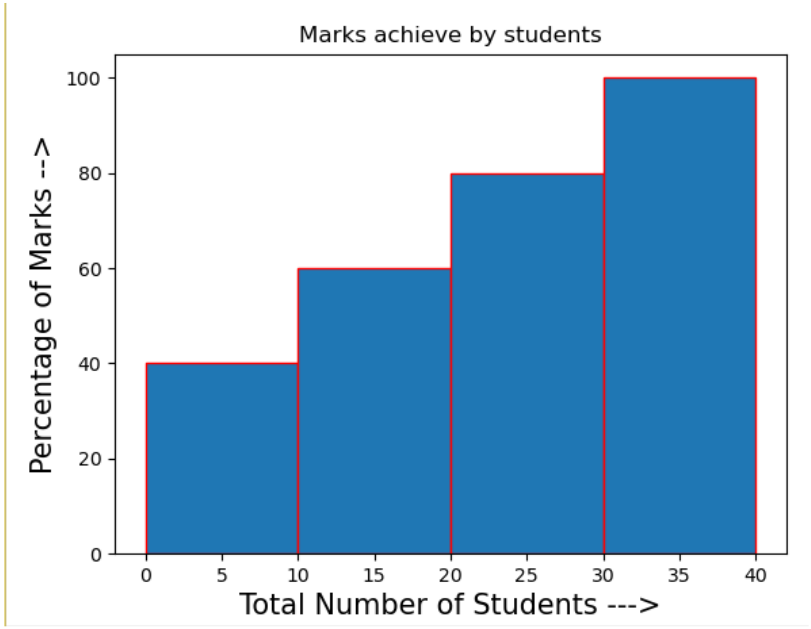
- import numpy as np
- import matplotlib.pyplot as plt
- data=[1,11,21,31,41]
- plt.hist([5,15,25,35,45,55],bins=[0,10,20,30,40,50,60], weights=[20,10,45,33,6,8],edgecolor="red")
- plt.show()



# Histogram

**Q: Example- to plot the histogram for marks achieved by students.**

- import numpy as np
- import matplotlib.pyplot as plt
- plt.hist([5,15,25,35],bins=[0,10,20,30,40], weights=[40,60,80,100],edgecolor="red")
- plt.xlabel('Total Number of Students --->', fontsize=15)
- plt.ylabel('Percentage of Marks -->', fontsize=15)
- plt.title('Marks achieve by students')
- plt.show()

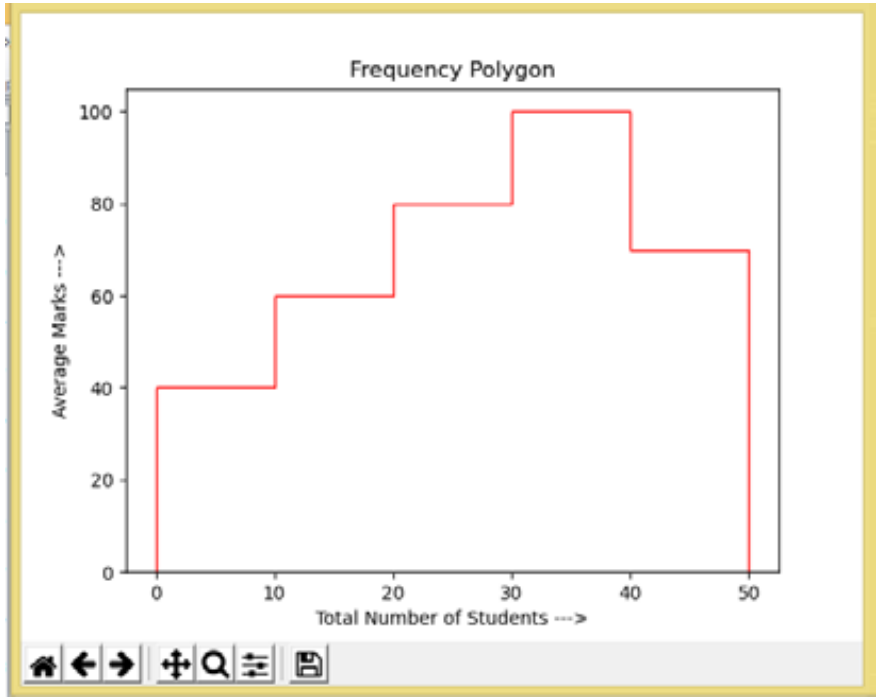


# Frequency Polygon

- It is a graph or chart which is plot using line to join the midpoints of each interval or bin. The heights of the points represent the frequencies. A frequency polygon can be created from the step type histogram or by calculating the midpoints of the bins from the frequency distribution table.
- command – plt.hist(x, bins=20, histtype ='step')
- Then after we join the midpoint of each set of adjacent bins to create a frequency polygon.

# Frequency Polygon

- Example-
- import numpy as np
- import matplotlib.pyplot as plt
- plt.hist([5,15,25,35,45], bins=[0,10,20,30,40,50], weights=[40,60,80,100,70], edgecolor="red",histtype='step')
- plt.xlabel('Total Number of Students --->')
- plt.ylabel('Average Marks --->')
- plt.title('Frequency Polygon')
- plt.show()

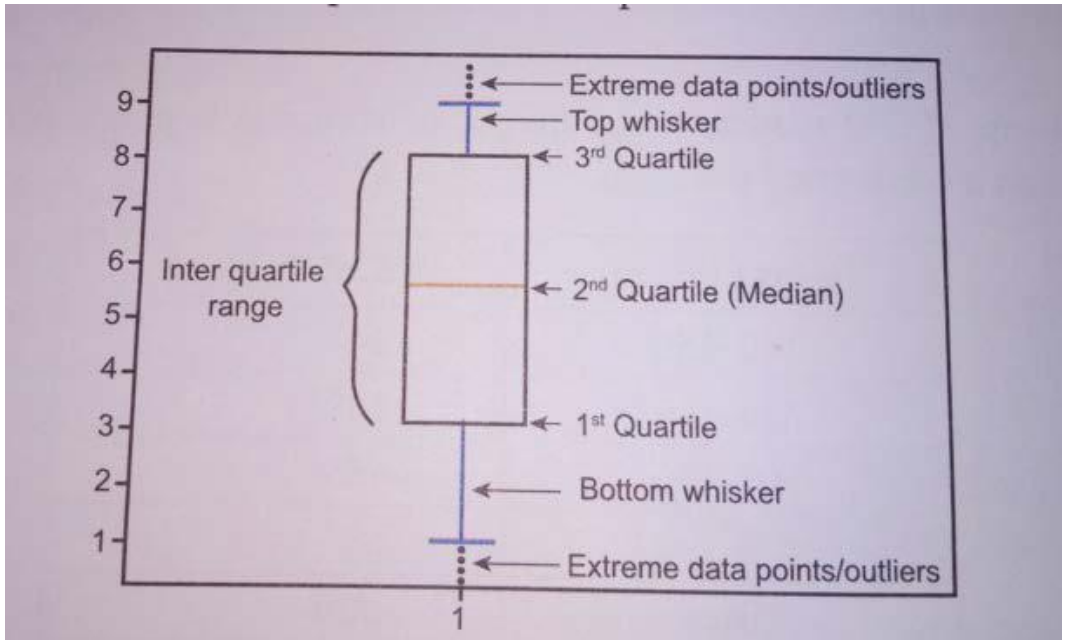


# Attributes and Customization of Frequency Polygon Chart

- plt.xlabel() function used to display label along with x axis.
- plt.ylabel() function used to display label along with y axis.
- plt.title() function used to display the title of the chart.
- plt.hist([5,15,25,35,45], bins=[0,10,20,30,40,50],weights=[40,60,80,100,70], edgecolor="red",histtype='step') – It is used to specify parameters –
- i. data range
- ii. bins
- iii. weights(data values)
- iv. edgecolor – To specify colour of the edges of polygon. &
- v. histtype = step – To create a frequency polygon chart.

# Box Plot

- It is a visual representation of statistical five number summary of a given data set. Where five number summary includes-
- i. Minimum Value
- ii. First Quartile
- iii. Median(Second Quartile)
- iv. Third Quartile
- v. Maximum Value



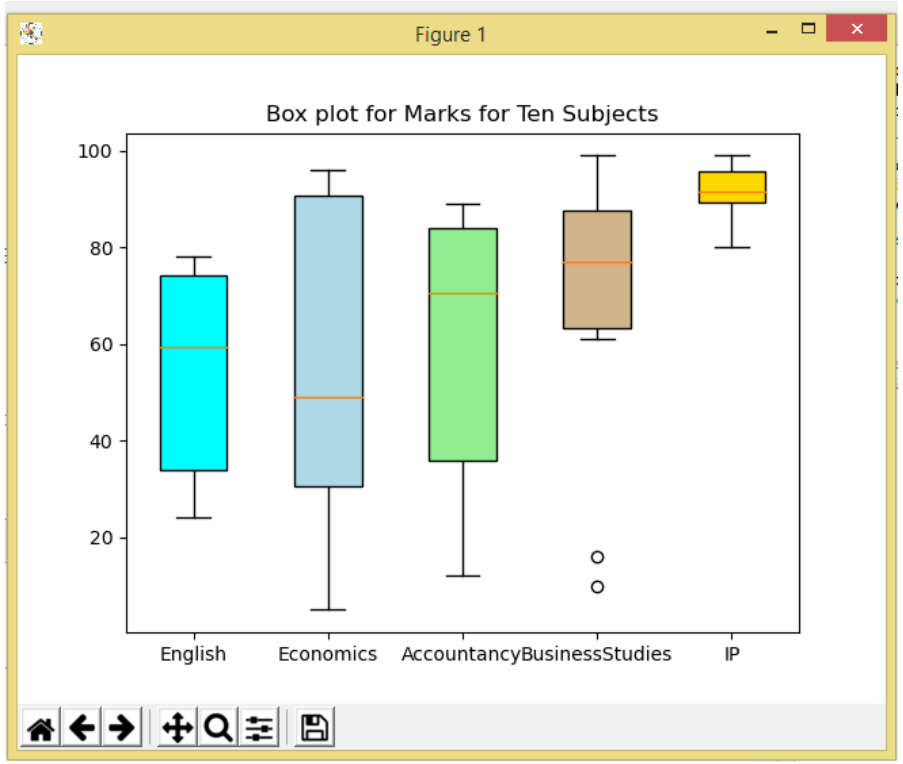

****Box plots are used as they provide a visual summary of the data enabling researchers to quickly identify mean values of the interquartile range, the dispersion of the data set and sign of skewness.**

# Box Plot

- Example- Write a python program to display box plot diagram for different five subjects.

| data | Set of data values | Colour |
|---|---|---|
| marksEng | 72,76,24,40,57,62,75,78,31,32 | cyan |
| marksEco | 62,5,91,25,36,32,96,95,30,90 | lightblue |
| marksAcc | 23,89,12,78,72,89,25,69,68,86 | lightgreen |
| marksBst | 99,73,70,16,81,61,88,98,10,87 | tan |
| marksIP | 90,95,92,96,97,89,80,99,91,85 | gold |

```
import matplotlib.pyplot as plt
marksEng = [72,76,24,40,57,62,75,78,31,32]
marksEco = [62,5,91,25,36,32,96,95,30,90]
marksAcc = [23,89,12,78,72,89,25,69,68,86]
marksBst = [99,73,70,16,81,61,88,98,10,87]
marksIP = [90,95,92,96,97,89,80,99,91,85]
data = [marksEng, marksEco, marksAcc, marksBst, marksIP]
box = plt.boxplot(data, vert=1, patch_artist=True,
labels=['English','Economics','Accountancy','BusinessStudies', 'IP'])
colors = ['cyan', 'lightblue', 'lightgreen', 'tan', 'gold']
for patch, color in zip(box['boxes'], colors):
patch.set_facecolor(color)
plt.title("Box plot for Marks for Ten Subjects")
plt.show()
```

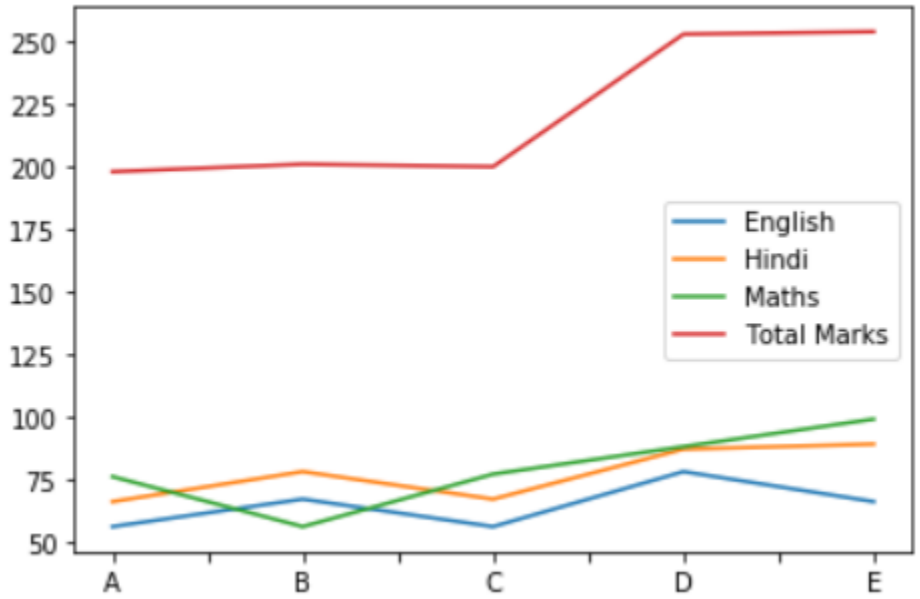


# DataFrame Plotting

- Pandas provides a function plot() which can be used with a DataFrame as:
- <df>.plot()
- It is a versatile function which can plot all types of chart by specifying the kind argument that are:
- Kind: type of the plot, can take values as
- Line: line plot
- Bar: bar plot
- Barh: horizontal bar plot
- Hist: histogram

# DataFrame Plotting

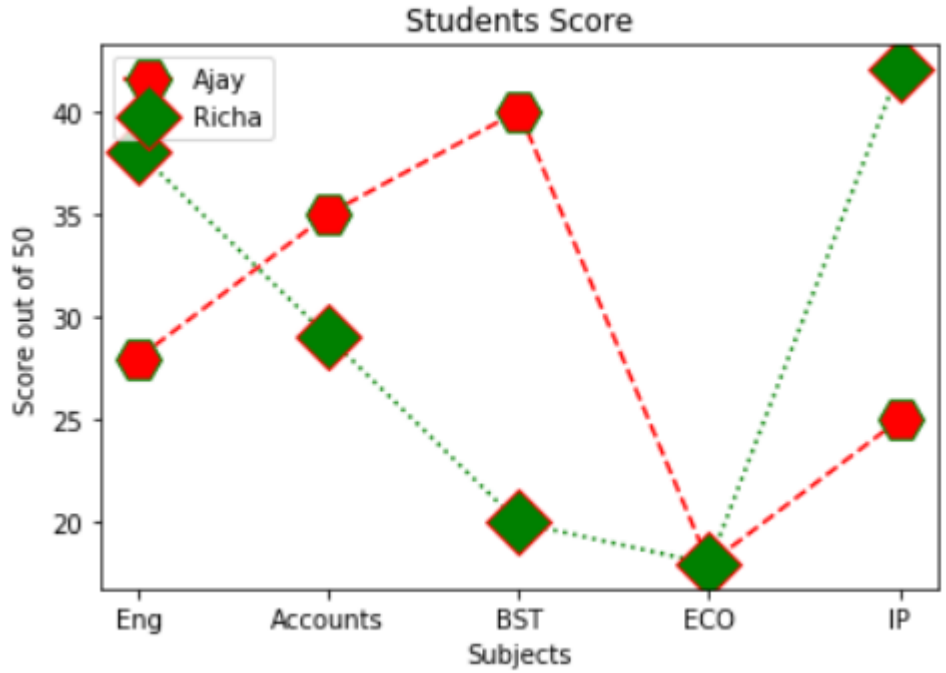
```
import pandas as pd
import matplotlib.pyplot as plt
dict1={'students': ['ram', 'Shyam', 'Ruchika', 'Neha', 'Anya'],
      'English': [56, 67, 56, 78, 66],
      'Hindi': [66, 78, 67, 87, 89],
      'Maths': [76, 56, 77, 88, 99]}
df=pd.DataFrame(dict1, index=['A', 'B', 'C', 'D', 'E'])
print(df)
df["Total Marks"]=df.sum(axis=1)
print(df)
print(df[['students','Total Marks']])
for index, row in df.iterrows():
    print(row['students'])
    print(row['Total Marks'])
    print("--------------")
df.plot(kind='line')
```



# Practical Programs

- **To create Line chart to display the score of two students in 5 subjects**

```
import matplotlib.pyplot as plt
import numpy as np
xvalues=np.array([1,2,3,4,5])
 score1= np.array([28,35,40,18,25])
score2= np.array([38,29,20,18,42])
 plt.plot(xvalues,score1,linestyle='dashed',color='r',marker='H',
markersize=18,markeredgecolor='g', label="Ajay")
plt.plot(xvalues,score2,linestyle='dotted',color='g',marker='D',
markersize=18,markeredgecolor='r', label="Richa")
plt.title("Students Score")
plt.xlabel("Subjects")
 plt.ylabel("Score out of 50")
 plt.xticks(ticks=[1,2,3,4,5], labels=['Eng','Accounts','BST','ECO','IP'])
plt.legend(loc="best")
plt.show()
```



# Practical Programs

- Consider the School is collecting money for charity. Write a program to plot the amount collected vs. days using a bar chart. The ticks on X-axis should have Day names. The graph should have proper titles and axes titles.

# Practical Programs

- Create a DataFrame named prodf which consist of production of Fruits, pulses, rice and wheat by different states. Write a program to plot a bar chart with the column pulses.
- X=[“
- prodf. Plot(kind=“ “,x=range(1, len(pulses)+1)  y=“prodf.pulses)

# Practical Programs

- A Data Frame pdas:
- 1990 2000 2010
- 1. 54 345 895
- 2. 64 485 562
- 3. 79 690 1100
- 4. 96 770 890
- A line chart from the 1990 and 2000 of data framepd.
- A bar chart to plotting the three columns of data framepd.