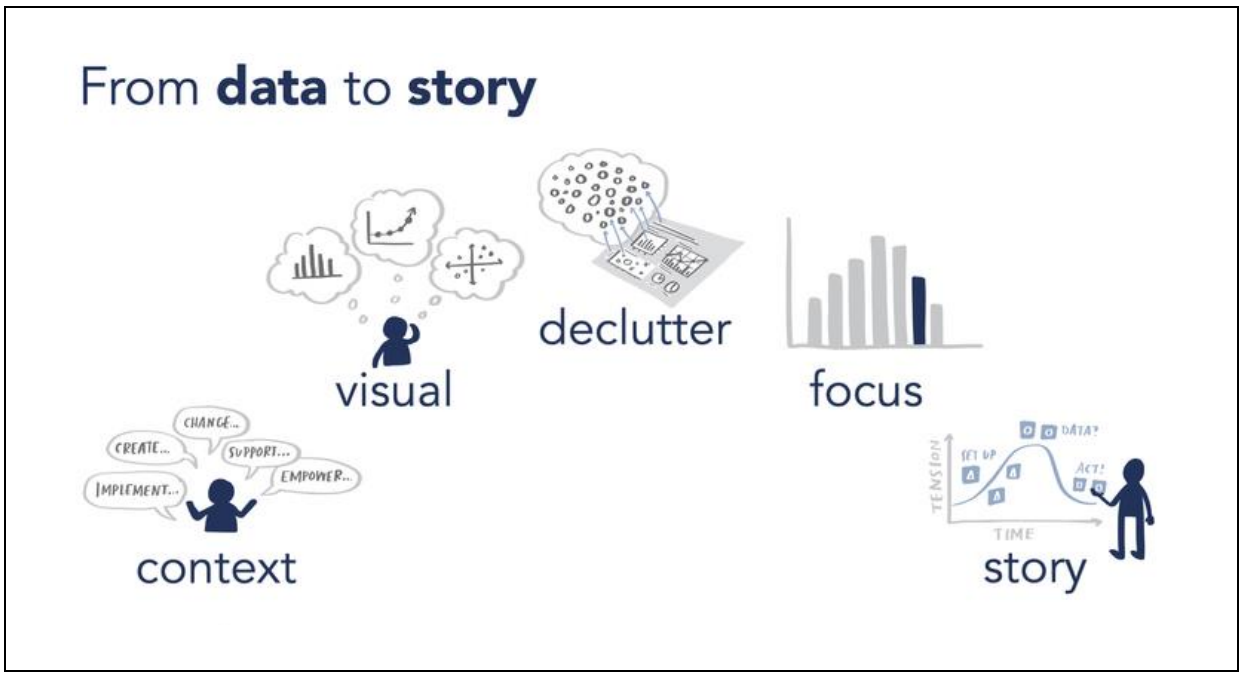
1. DATA VISUALIZATION PART – 1

Contents

1. DATA VISUALIZATION PART – 1



1. Data Visualization

- ✓ **Data Visualization** is the process of converting raw information (text, numbers, or symbols) into a graphical representation.
- ✓ If we visualize the data then it is very easy to understand.

Best quote

- ✓ A picture gives more meaningful information than thousand words

1.1. Example in words

- ✓ Reaching to target

2. Common data visualization techniques

- ✓ Bar charts
- ✓ Pie charts
- ✓ Line graphs
- ✓ Box plot
- ✓ Scatter plot & etc

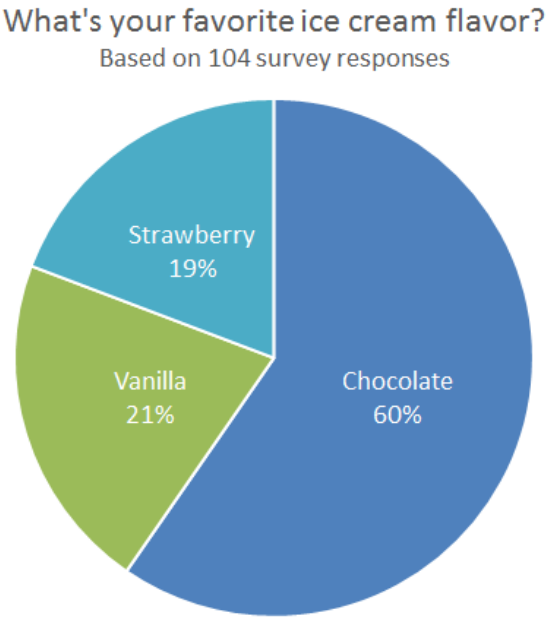
3. Advantages

- ✓ To identify trends, such as whether sales increasing or decreasing.
- ✓ To identify patterns, such as during weekend more sales.
- ✓ To identify relationships, such as if we study more hours then we will get good marks.
- ✓ To identify frequency, such as how often a product is purchased in a specific area & etc

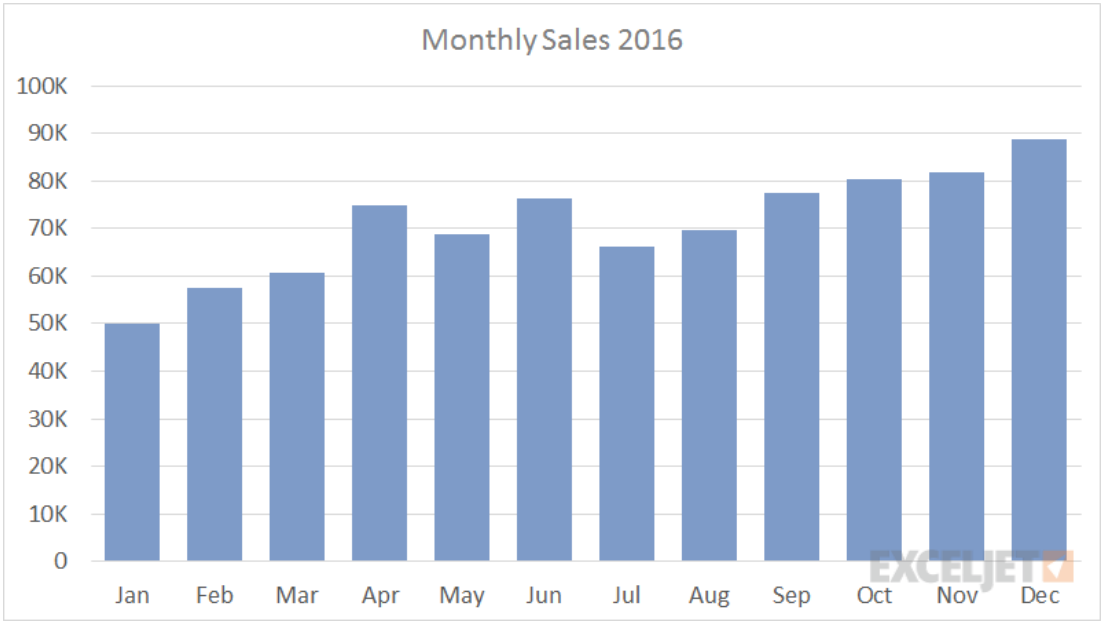
4. Few examples

What's your favorite ice cream flavor?

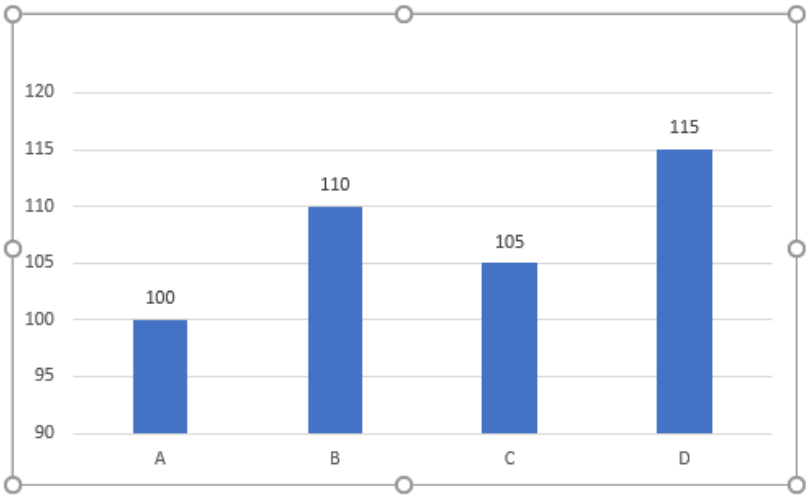
Flavor	Count
Chocolate	62
Vanilla	22
Strawberry	20



Bar chart

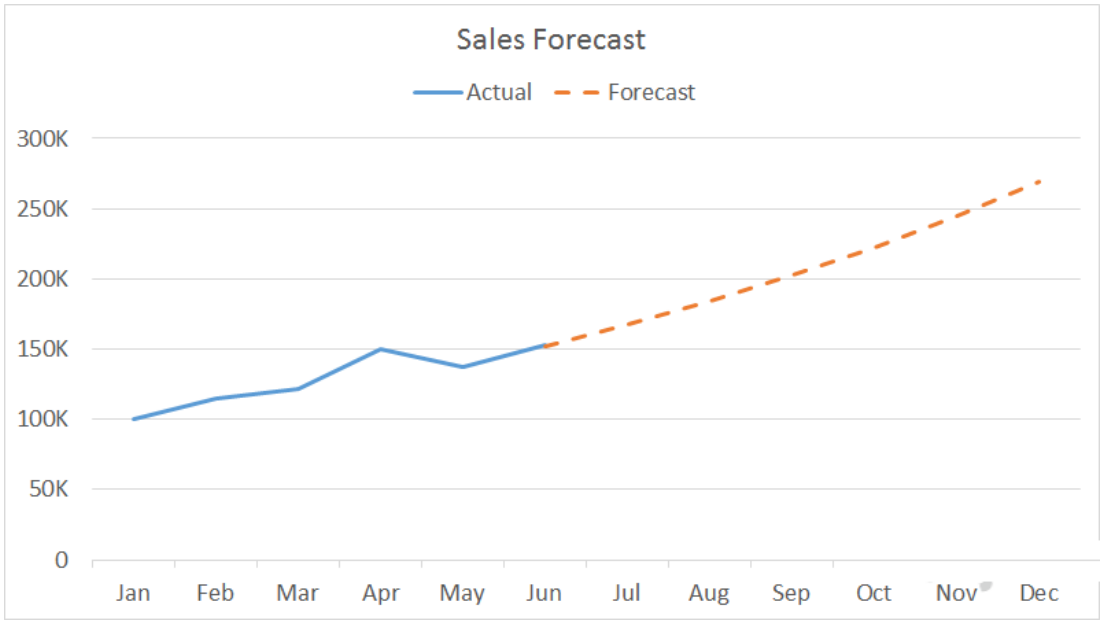


Group	Value
A	100
B	110
C	105
D	115



	Actual	Forecast
Jan	100K	
Feb	115K	
Mar	121K	
Apr	150K	
May	137K	
Jun	152K	152K
Jul		167K
Aug		184K
Sep		202K
Oct		223K
Nov		245K
Dec		269K

Line Chart



5. Matplotlib

- ✓ Matplotlib is the most popular plotting library in python.
- ✓ Using matplotlib we can plot the data.

Environment

- ✓ We can install this library by using pip command.

matplotlib installation

```
pip install matplotlib
```

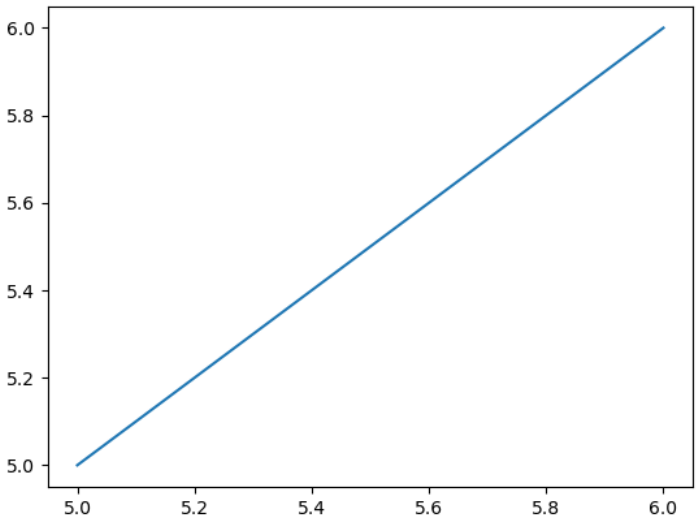
6. Line chart

- ✓ A line chart or line graph is a type of chart which displays information as a series of data points connected by straight line
- ✓ A line chart is often used to visualize a trend in data over intervals of time.

Program Name: Create a simple line chart
demo1.py

```
import matplotlib.pyplot as plt

x = [5, 6]
y = [5, 6]

plt.plot(x, y)

plt.show()
```

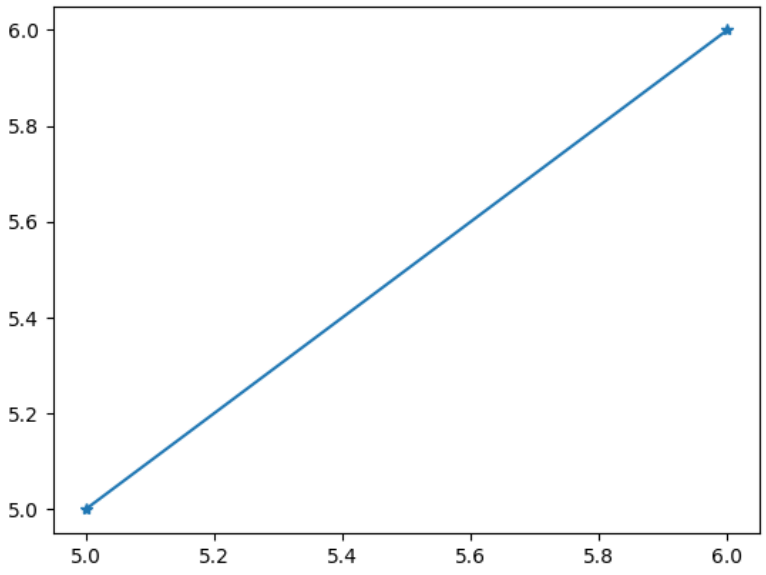
Output



Program Name: Create a simple line chart
demo2.py

```
import matplotlib.pyplot as plt

x = [5, 6]
y = [5, 6]

plt.plot(x, y, marker='*')

plt.show()
```

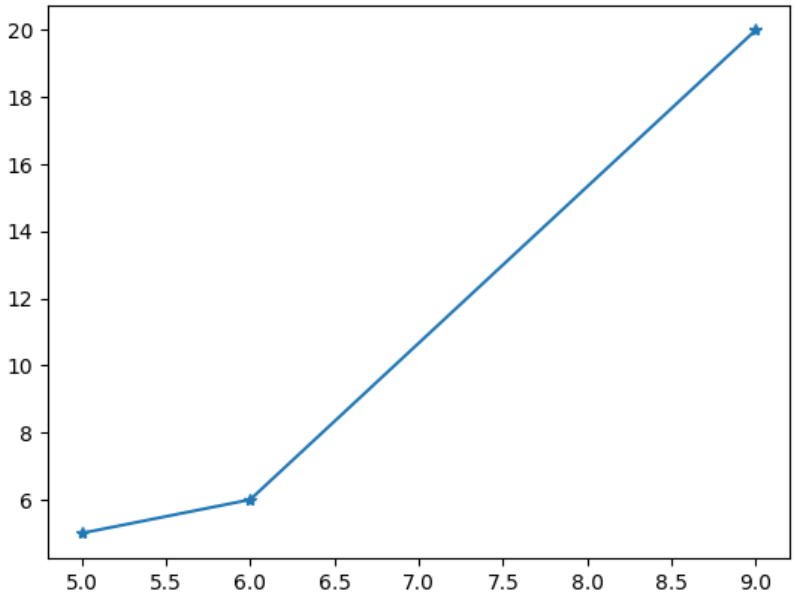
Output



Program Name: Create a simple line chart
demo3.py

```
import matplotlib.pyplot as plt

x = [5, 6, 9]
y = [5, 6, 20]

plt.plot(x, y, marker='*')

plt.show()
```

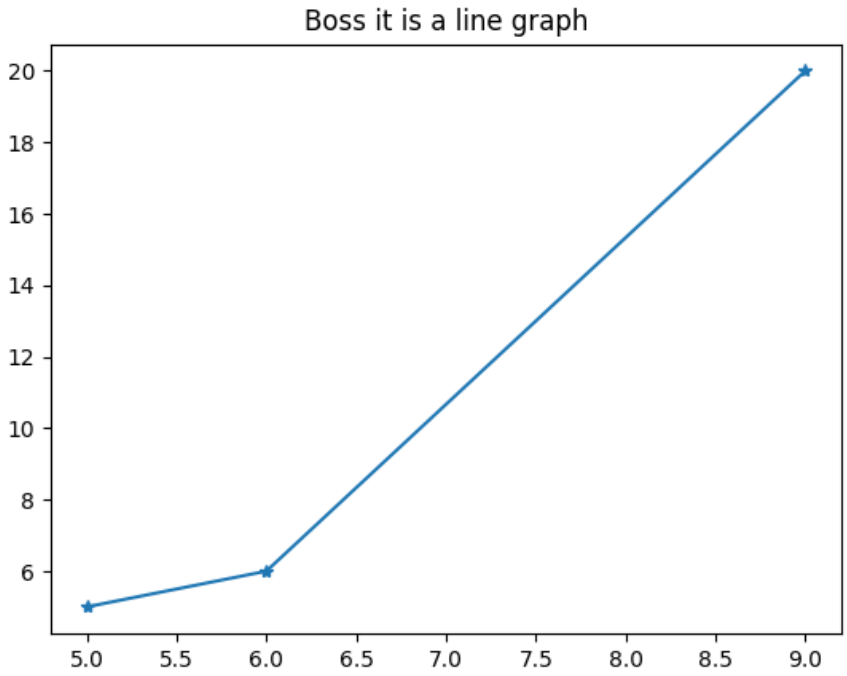
Output



Program Name: Create a simple line chart and title
demo4.py

```
import matplotlib.pyplot as plt

x = [5, 6, 9]
y = [5, 6, 20]

plt.title("Boss it is a line graph")

plt.plot(x, y, marker='*')

plt.show()
```

Output



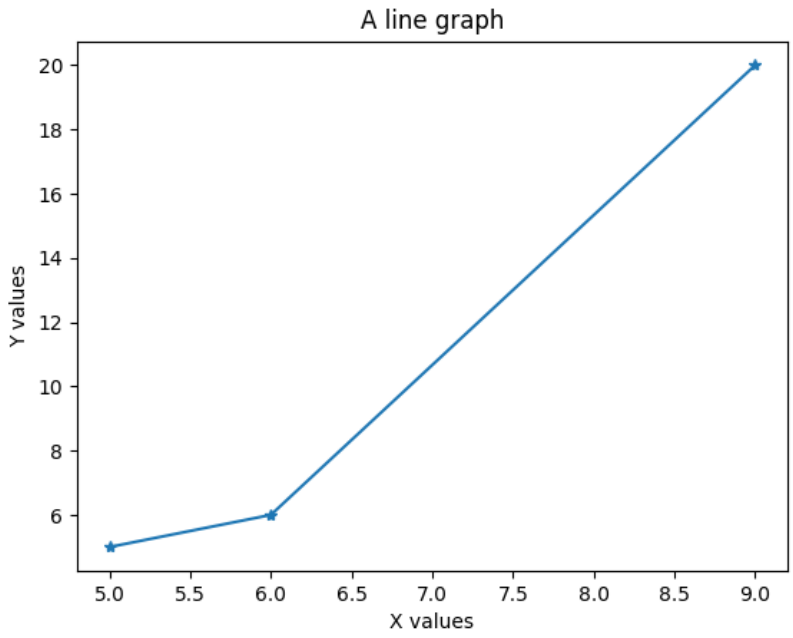
6.1. Labelling the axes

✓ We can label x axis and y axis by using xlabel and ylabel

Program Name: Create a simple line chart and giving title and labelling
demo5.py

```
import matplotlib.pyplot as plt

x = [5, 6, 9]
y = [5, 6, 20]

plt.title("A line graph")

plt.xlabel("X values")
plt.ylabel("Y values")

plt.plot(x, y, marker = '*')

plt.show()
```

Output

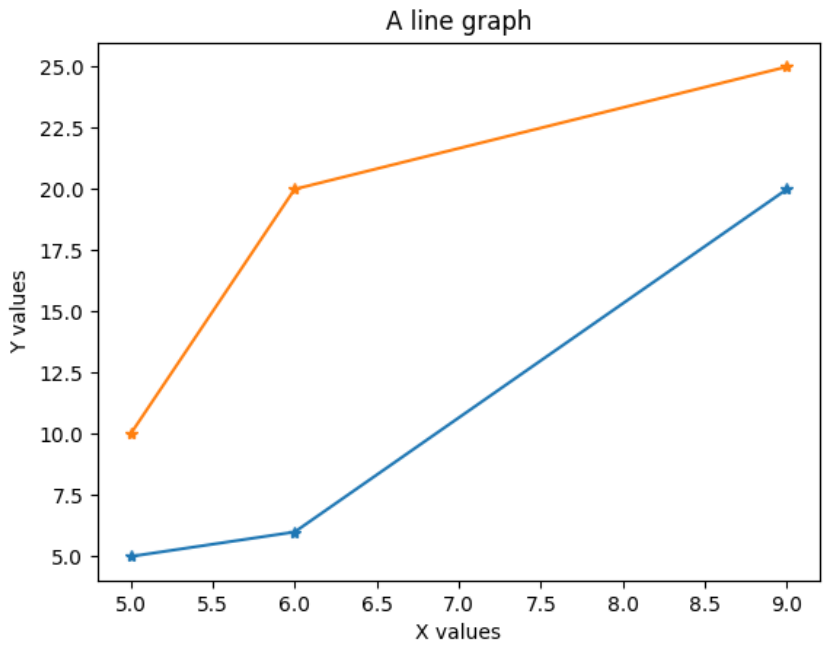


Program Name: Create two lines in single chart

demo6.py

```
import matplotlib.pyplot as plt

x = [5, 6, 9]
y = [5, 6, 20]
p = [10, 20, 25]

plt.title("A line graph")

plt.xlabel("X values")
plt.ylabel("Y values")

plt.plot(x, y, marker = '*')
plt.plot(x, p, marker = '*')

plt.show()
```

Output



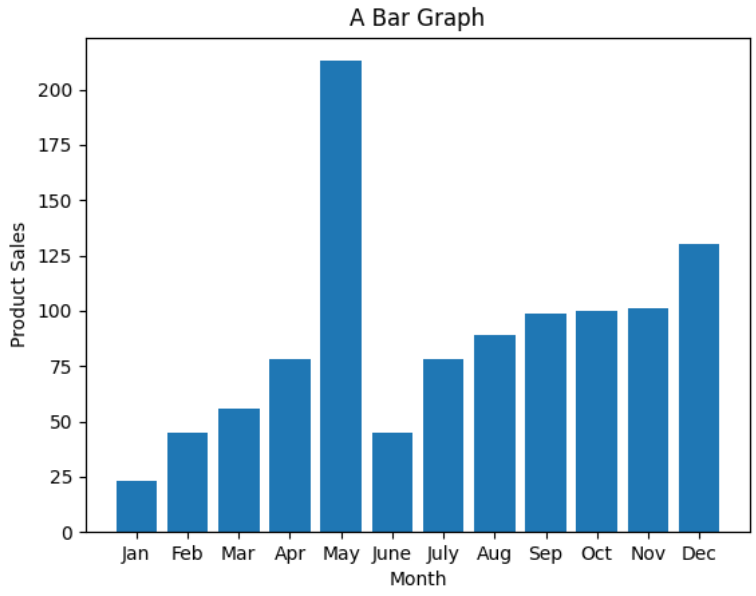
7. Bar Chart

✓ The bar graph is the graphical representation of categorical data.

Program Name: Creating bar chart
demo7.py

```
import matplotlib.pyplot as plt

months = ["Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug",
"Sep", "Oct", "Nov", "Dec"]
sales = [23, 45, 56, 78, 213, 45, 78, 89, 99, 100, 101, 130]

plt.title('A Bar Graph')

plt.xlabel('Month')
plt.ylabel('Product Sales')

plt.bar(months, sales)

plt.show()
```

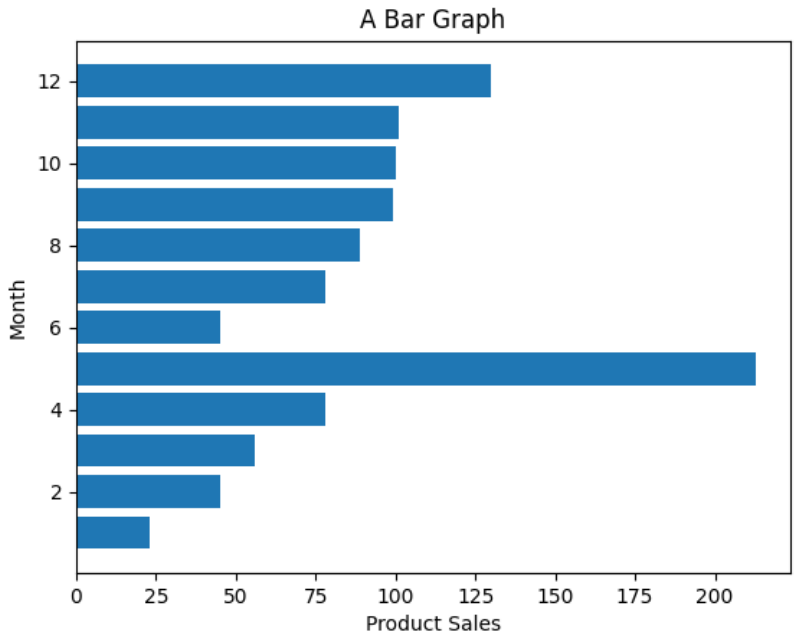
Output



Program Name: Creating horizontal bar chart
demo8.py

```
import matplotlib.pyplot as plt

months = ["Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug",
"Sep", "Oct", "Nov", "Dec"]
sales = [23, 45, 56, 78, 213, 45, 78, 89, 99, 100, 101, 130]

plt.title('A Bar Graph')

plt.xlabel('Product Sales')
plt.ylabel('Month')

plt.barh(months, sales)

plt.show()
```

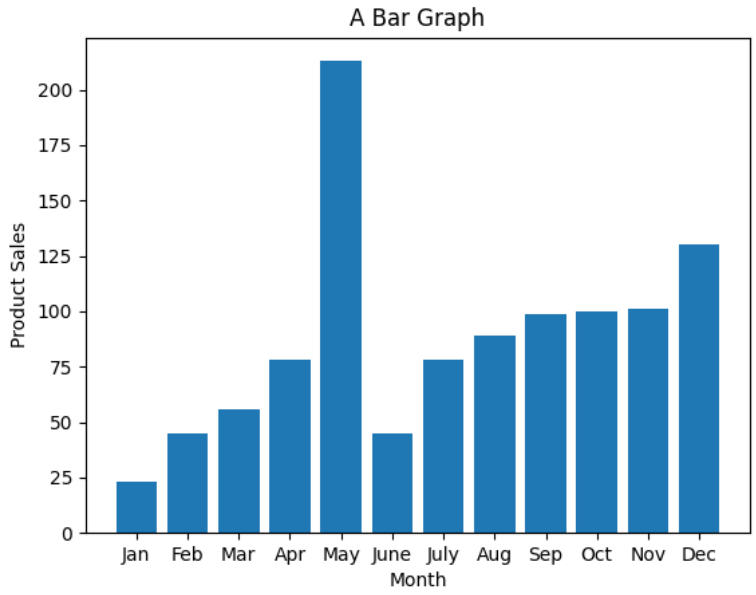
Output



Program Name: Creating horizontal bar chart
File name: demo9.py
File name: sales11.csv

```
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv("sales11.csv")

plt.title('A Bar Graph')

plt.xlabel('Month')
plt.ylabel('Product Sales')

plt.bar(df.month, df.sales)

plt.show()
```

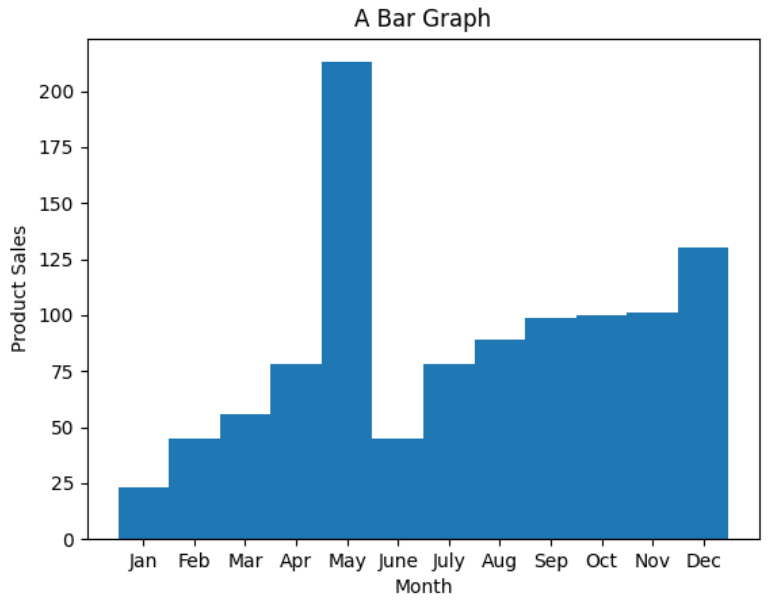
Output



Program Name: Creating bar chart
demo10.py

```
import matplotlib.pyplot as plt

months = ["Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug",
"Sep", "Oct", "Nov", "Dec"]
sales = [23, 45, 56, 78, 213, 45, 78, 89, 99, 100, 101, 130]

plt.title('A Bar Graph')

plt.xlabel('Month')
plt.ylabel('Product Sales')

plt.bar(months, sales, width = 1.0)

plt.show()
```

Output



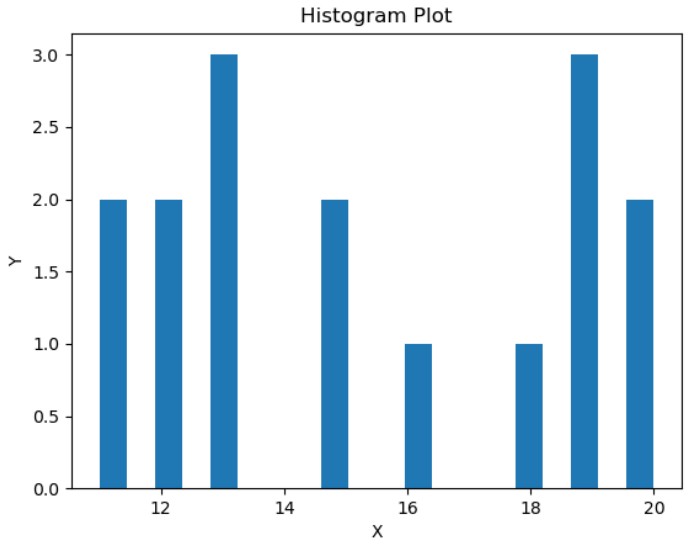
8. Histogram

- ✓ A histogram is the graphical representation of quantitative data.
- ✓ This displays the frequency/count of numerical data in bars.

Program Name: Creating histogram
demo11.py

```
import matplotlib.pyplot as plt

data = [12, 15, 13, 20, 19, 20, 11, 19, 11, 12, 19, 13, 15, 16, 18, 13]


plt.xlabel("X")
plt.ylabel("Y")
plt.title("Histogram Plot")

plt.hist(data, bins = 20)
plt.show()
```

Output



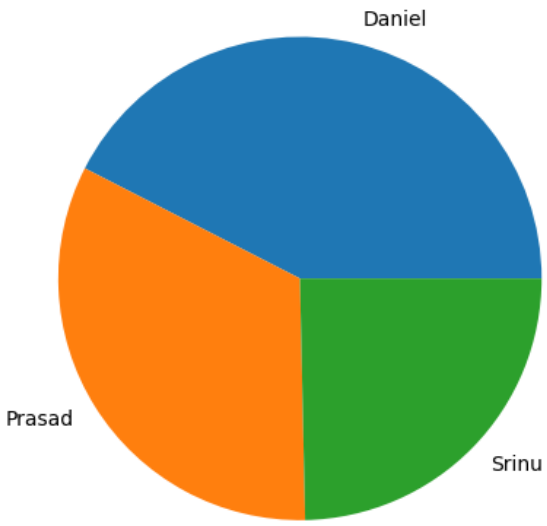
9. Pie Chart

- ✓ This is a circular plot that has been divided into slices displaying numerical proportions.
- ✓ Every slice in the pie chart shows the proportion of the element to the whole.
- ✓ A large category means that it will occupy a larger portion of the pie chart.

Program Name: Creating pie chart
demo12.py

```
import matplotlib.pyplot as plt

students = ['Daniel', 'Prasad', 'Srinu']
points = [62, 48, 36]

plt.pie(points, labels = students)

plt.axis('equal')
plt.show()
```

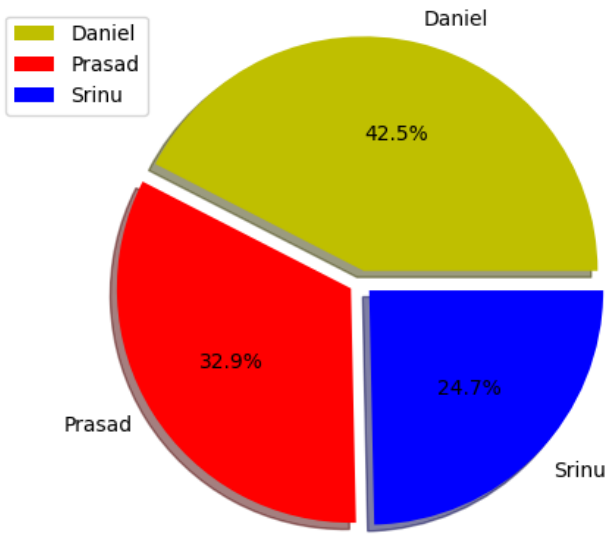
Output



Program Name: Creating pie chart
demo13.py

```
import matplotlib.pyplot as plt

students = ['Daniel', 'Prasad', 'Srinu']
points = [62, 48, 36]

c = ['y', 'r', 'b']

plt.pie(points, labels = students, colors = c , shadow = True,
explode = (0.05, 0.05, 0.05), autopct = '%1.1f%%')

plt.axis('equal')
plt.legend()
plt.show()
```

Output



9.1. Attributes

- ✓ The first parameter to the function is the list of numbers for every category.
  - labels attribute:
    - A list of categories separated by commas is then passed as the argument to labels attribute.
  - colors attribute:
    - To provide the color for every category.
  - To create shadows around the various categories in pie chart.
  - To split each slice of the pie chart into its own.

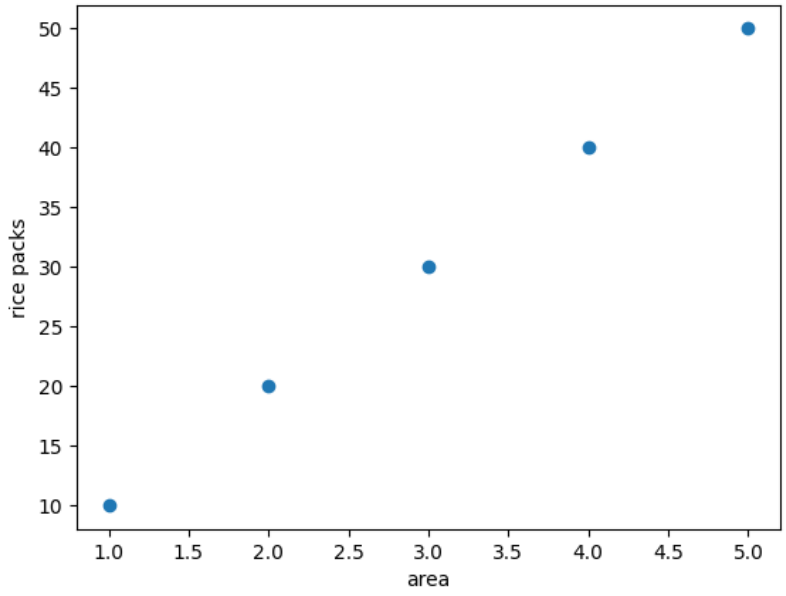
10. Scatter Plot

- ✓ In scatter plot each value in the data set is represented by a dot.
- ✓ By using this plot we can understand the relationship between two variables.

Program Name: Creating Scatter plot
demo14.py

```
import matplotlib.pyplot as plt

area = [1, 2, 3, 4, 5]
rice_packs = [10, 20, 30, 40, 50]

plt.xlabel('area')
plt.ylabel('rice packs')

plt.scatter(area, rice_packs)
plt.show()
```

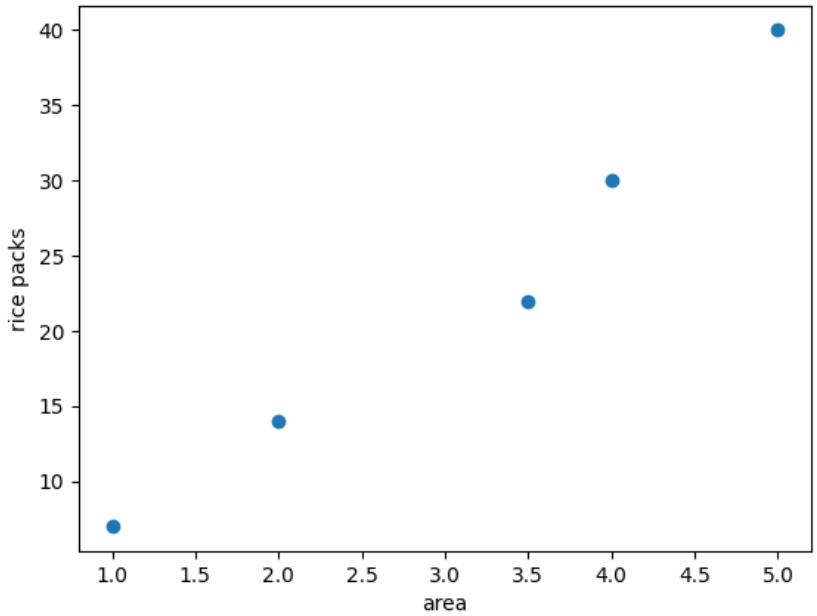
Output



Program Name: Creating Scatter plot
demo15.py

```
import matplotlib.pyplot as plt

area = [1, 2, 3.5, 4, 5]
rice_packs = [7, 14, 22, 30, 40]

plt.xlabel('area')
plt.ylabel('rice packs')

plt.scatter(area, rice_packs)
plt.show()
```

Output



11. Box Plots

- ✓ Box plots help us measure how well data in a dataset is distributed.
- ✓ The graph shows the maximum, minimum, median, first quartile and third quartiles of the dataset.

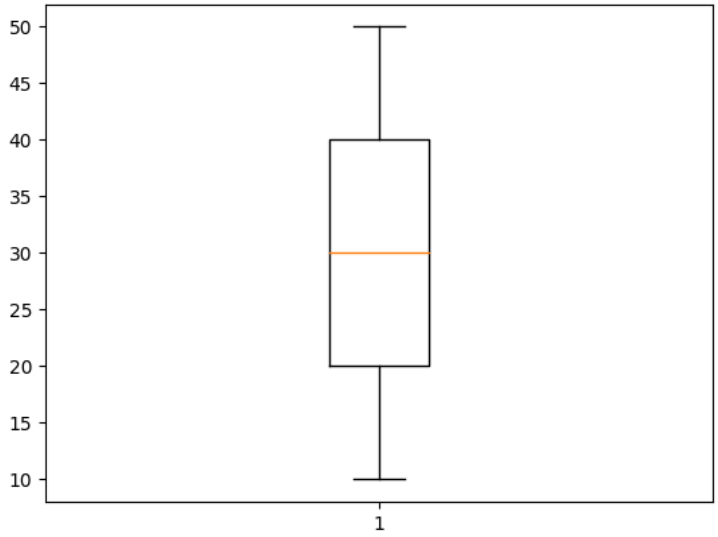
11.1. Use Box plots

- ✓ Use a boxplot when you need to get the overall statistical information about the data distribution.
- ✓ It is a good tool for detecting outliers in a dataset.

Program Name: Creating box plot
demo16.py

```
import matplotlib.pyplot as plt

data = [10, 20, 30, 40, 50]

plt.boxplot(data)
plt.show()
```

Output



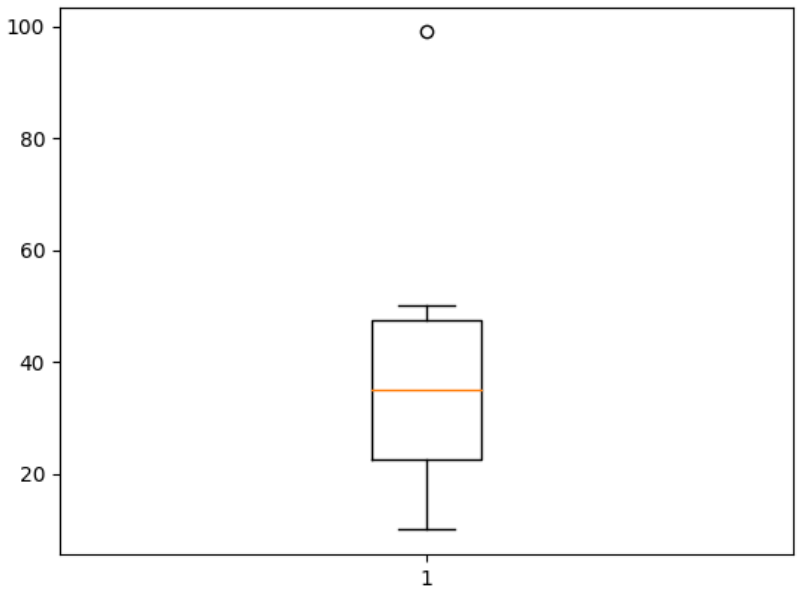
11.2. Box plot explanation

- ✓ The line dividing the box into two shows the median of the data.
- ✓ The end of the box represents the upper quartile (75%) while the start of the box represents the lower quartile (25%).
- ✓ The part between the upper quartile and the lower quartile is known as the Inter Quartile Range (IQR) and helps in approximating 50% of the middle data.

Program Name: Creating box plot with outlier
demo17.py

```
import matplotlib.pyplot as plt

data = [10, 20, 30, 40, 50, 90]

plt.boxplot(data)
plt.show()
```

Output



12. Heatmap

- ✓ A heatmap is a method of data visualization that plots data by replacing numbers with colours.
- ✓ If it is representing with color then it is very easy to understand patterns between different values in the dataset.
- ✓ It is used to visualize data in a two-dimensional format as a coloured map so that different colour variations represent different patterns between features.

12.1. How to understand?

- ✓ A heatmap visualizes the relationship between features as a colour palette.
- ✓ While analysing a heatmap, always remember that **dark shades** represent a **high degree** of linear relationship between features and **light shades** represent a **low degree** of linear relationship between features.

Program Name: Creating box plot
demo18.py

```
import matplotlib.pyplot as plt
import pandas as pd

d = {
    "Apple": [10, 20, 30, 40],
    "Orange": [7, 14, 21, 28],
    "Banana": [55, 15, 8, 12],
    "Pear": [15, 14, 1, 8]
}

i = ['Basket1', 'Basket2', 'Basket3', 'Basket4']

df = pd.DataFrame(d, index = i)

plt.imshow(df, cmap = "YlGnBu")
plt.colorbar()

plt.xticks(range(len(df)), df.columns)
plt.yticks(range(len(df)), df.index)

plt.show()
```

Output