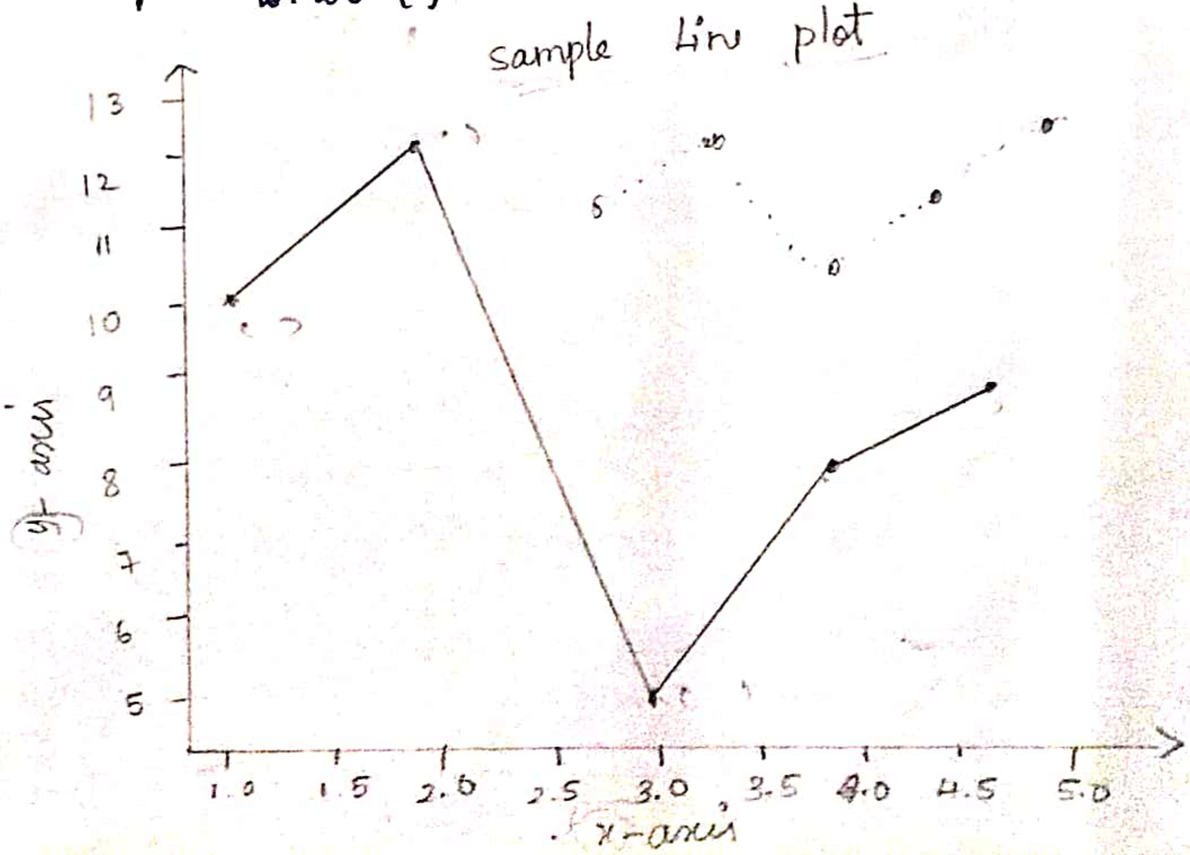
Simple line plot is used to illustrate the relationship ~~between dependent~~ ~~the~~ Variables

It shows graphical representation of data

It shows a ~~simple~~ long

simple Line plot :

```
# import matplotlib
import matplotlib.pyplot as plt

# sample data
x = [1, 2, 3, 4, 5]
y = [10, 12, 5, 8, 9]

# create a line plot
plt.plot(x, y)            Linestyle = ':'
# customize the plot      color = "blue",
                                   black
plt.title('sample Line plot')      pink
plt.xlabel('x-axis')               red
plt.ylabel('Y-axis')

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



Scatter plot }

Scatter plot [illegible] the relationship between two or more variable.

It represents data on x, y Cartesian plane.

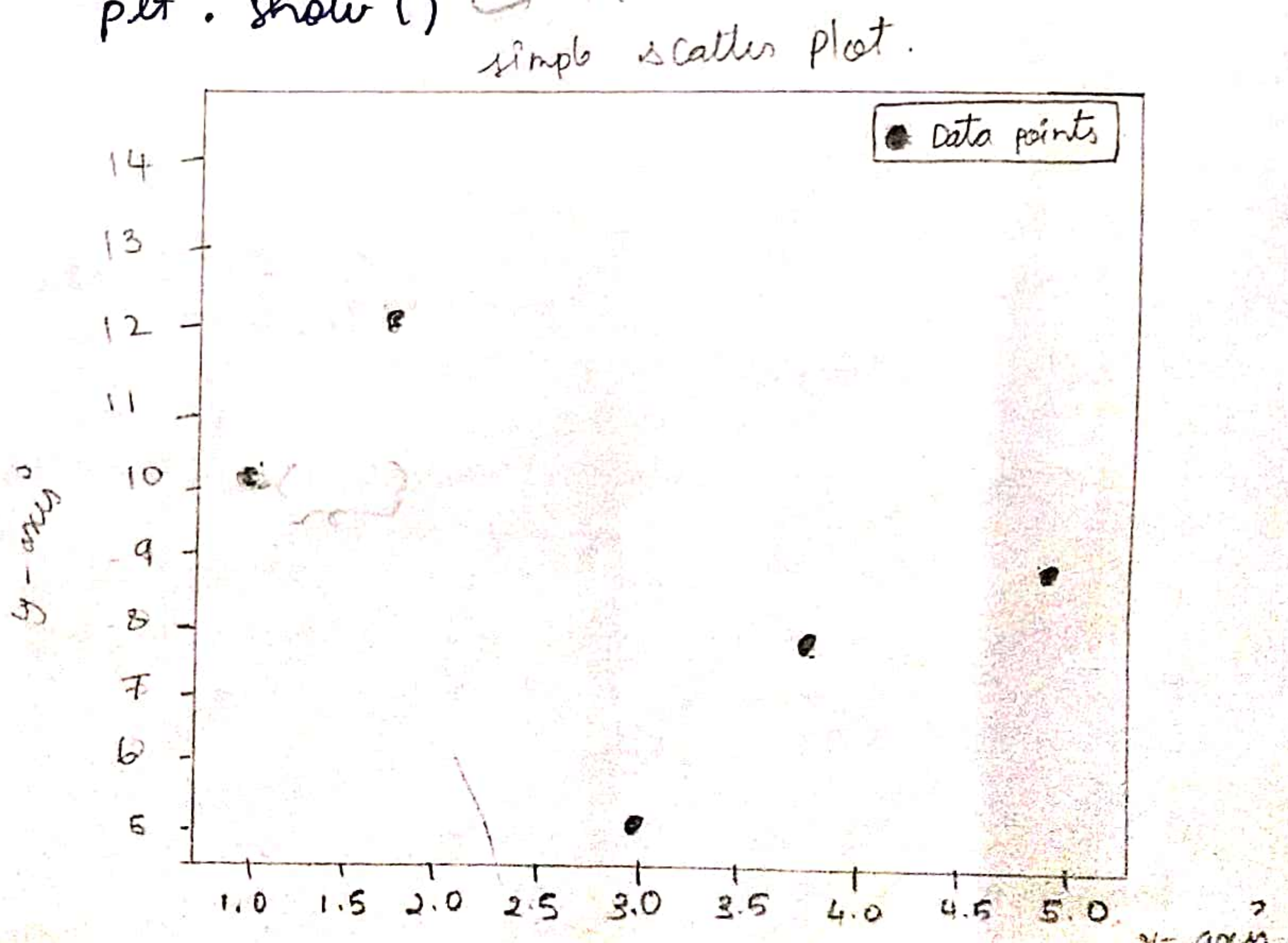
It works based on two condition.

=> x depends on y

=> x independent y

simple scatter plots:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 12, 5, 8, 9].

# create a scatter plot

plt.scatter(x, y, label = 'Data points', colour='blue',
                marker = 'o')

# customize the plot.
plt.title('simple scatter plot')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend()

plt.show()
```



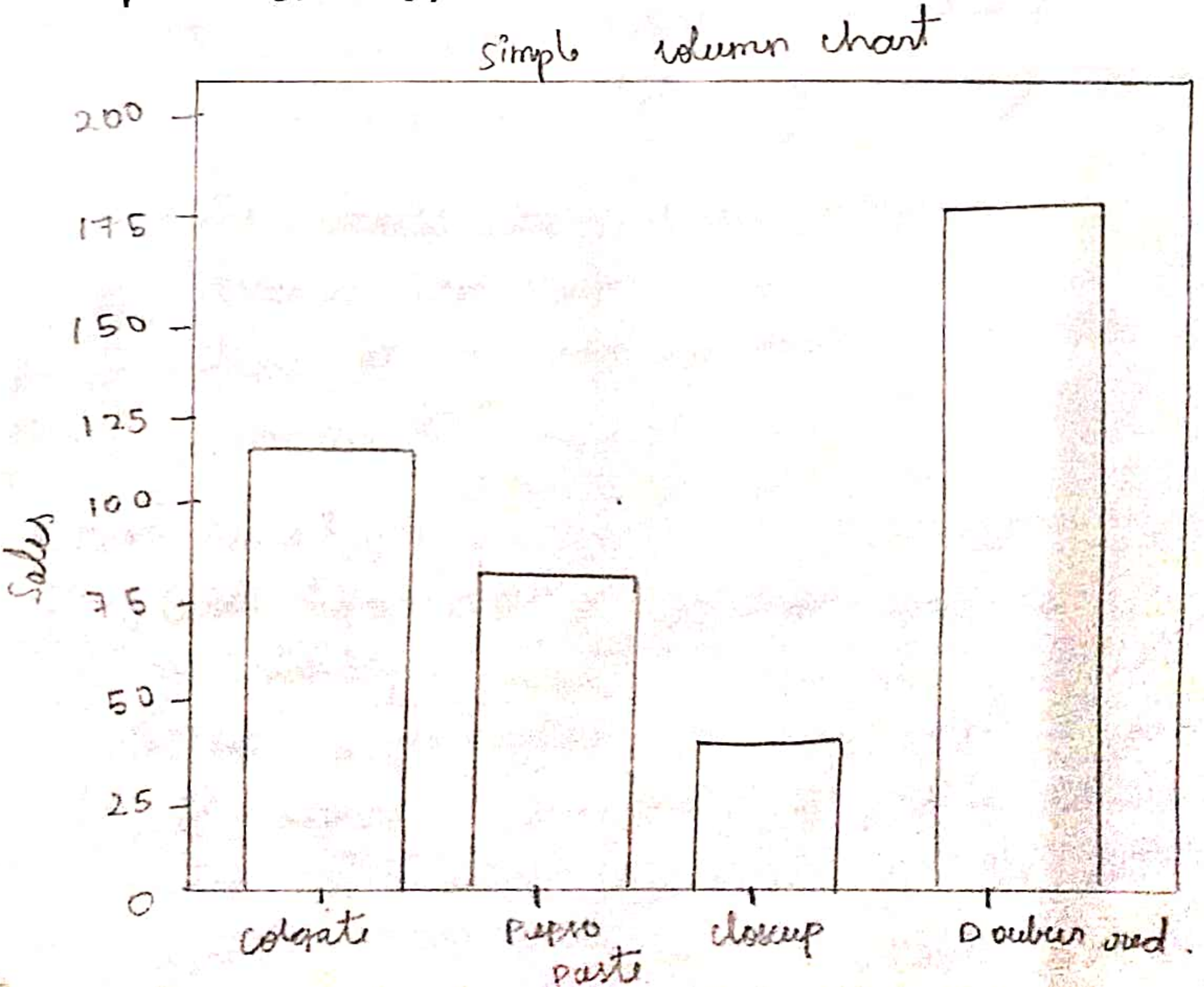
Bar chart :-

* Bars can be categorized horizontally or vertically.

* It shows representation of data

simple Bar chart

```
import matplotlib.pyplot as plt
# sample data
paste = ['colgate', 'Pepsco', 'closeup', 'Doubur red']
sales = [120, 80, 40, 180]
# create a column chart
plt.bar(paste, sales, color='green')
plt.title('simple column chart')
plt.xlabel('paste')
plt.ylabel('sales')
plt.show()
```



Visualizing errors:

> Visualizing errors in your data is essential for understanding the uncertaintly or variability associated with your measurements.

> Matplotlib provides various ways to visualize errors, and one common approach is to use error bars.

> uncertaintly in the context of data and measurements refers to the lack of exact knowledge about a quantity or value.

> It represents the range of possible values that a measurement or data point might have due to factors such as measurement errors, variations, or limitations in data collection processes.

> The choice of visualization method depends on your data, goals, and the type of uncertainty you want to communicate.

Error Bars:

> Error bars are a common way to visualize uncertainty → [illegible]

> They can be added to data points in plots to represent the range of possible values or the standard deviation of the data.
(Line, bar, scatter plots)

sample code for Errorbar:

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

# sample data
x = np.arange(1, 6)
y = np.array([10, 12, 5, 8, 9])

y_err = np.array([1, 2, 1, 2, 1])  # Eg errors values.

# Create a simple line plot with error bars.
plt.errorbar(x, y, yerr = y_err, fmt = 'O',
             colour = 'b', capsize = 4)

# customize the plot
plt.title('Data with Error Bars')
plt.xlabel('x-axis')
plt.ylabel('y-axis')

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

range → 1, 2, 3, 4, 5

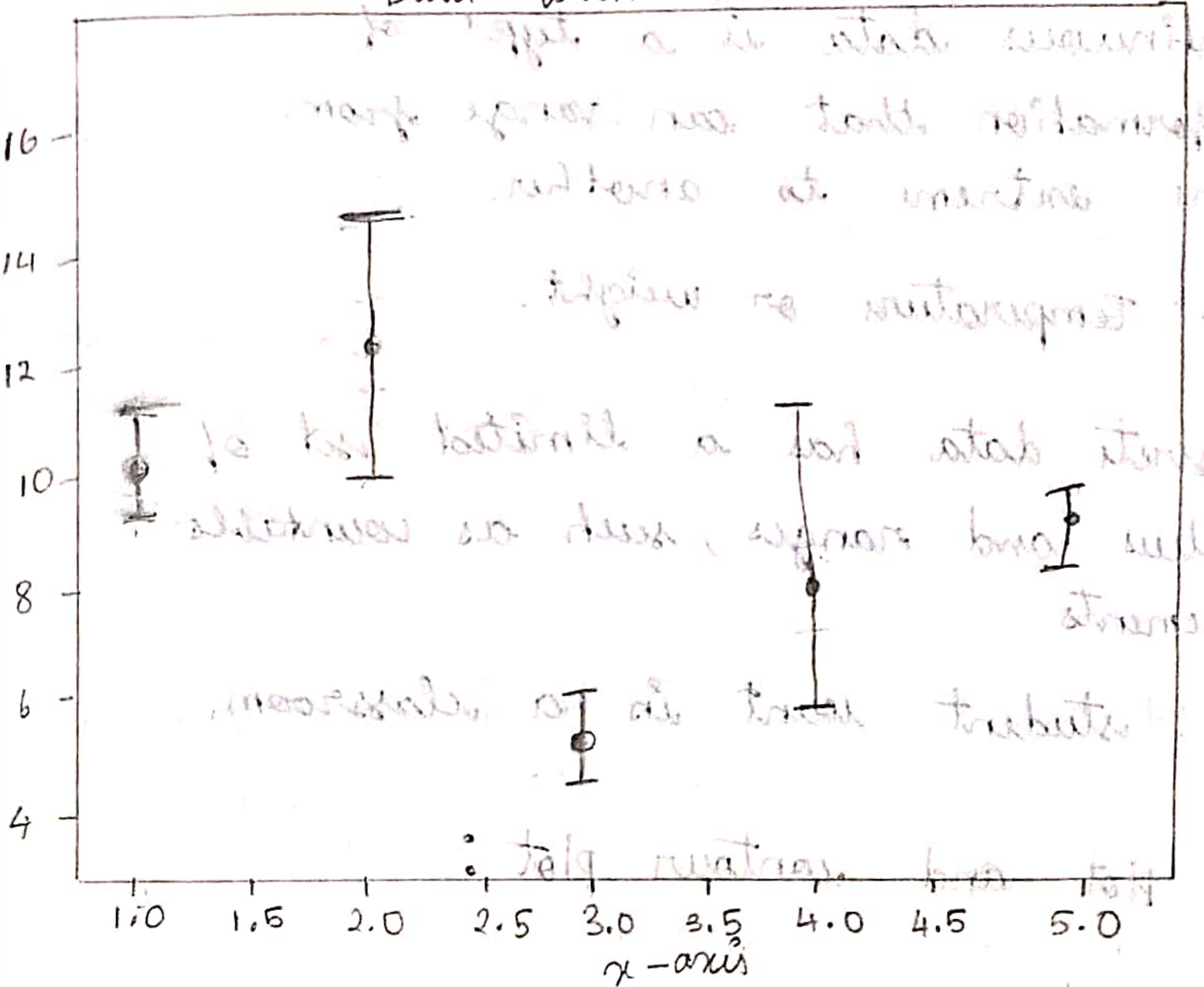
Format of Data points → marker O → To show in circle format

'b' → blue

capsize → size of the end of Error Bar

y-axis

Data with Error Bars.



> This is usually used in scientific & Statistical Analysis to show the erros (or) uncertainty for visulation.

Distribution:

The Distribution refers to how the data is clustered around certain values or ranges.

Use

gain insights into the characteristics and pattern of the data --> making informed decisions and predictions.

1. Continuous data is a type of information that can range from one extreme to another.

Eg: Temperature or weight.

2. Discrete data has a limited set of Values and ranges, such as countable elements

Eg: student count in a classroom.

Density plot and contour plot:

density plot:

two - dimensional colormap.

A density plot is a graphical representation of data density to visualize the distribution of data points.

Eg: heat maps & 2D histograms.

contour plot:

A contour plot represents a three - dimensional surface by showing lines (contours) at constant Values of the third dimension visualizing functions of two variables.

Eg: elevation maps, Temperature distributions.

To display three-dimensional data in two dimensions. using contours or color-coded regions.

Density and contour plots are commonly used in data visualization to represent the distribution of data points in 2-D space.

plt.contourf
plt.imshow
np.meshgrid function, which builds two-dimensional grids from one-dimensional array.

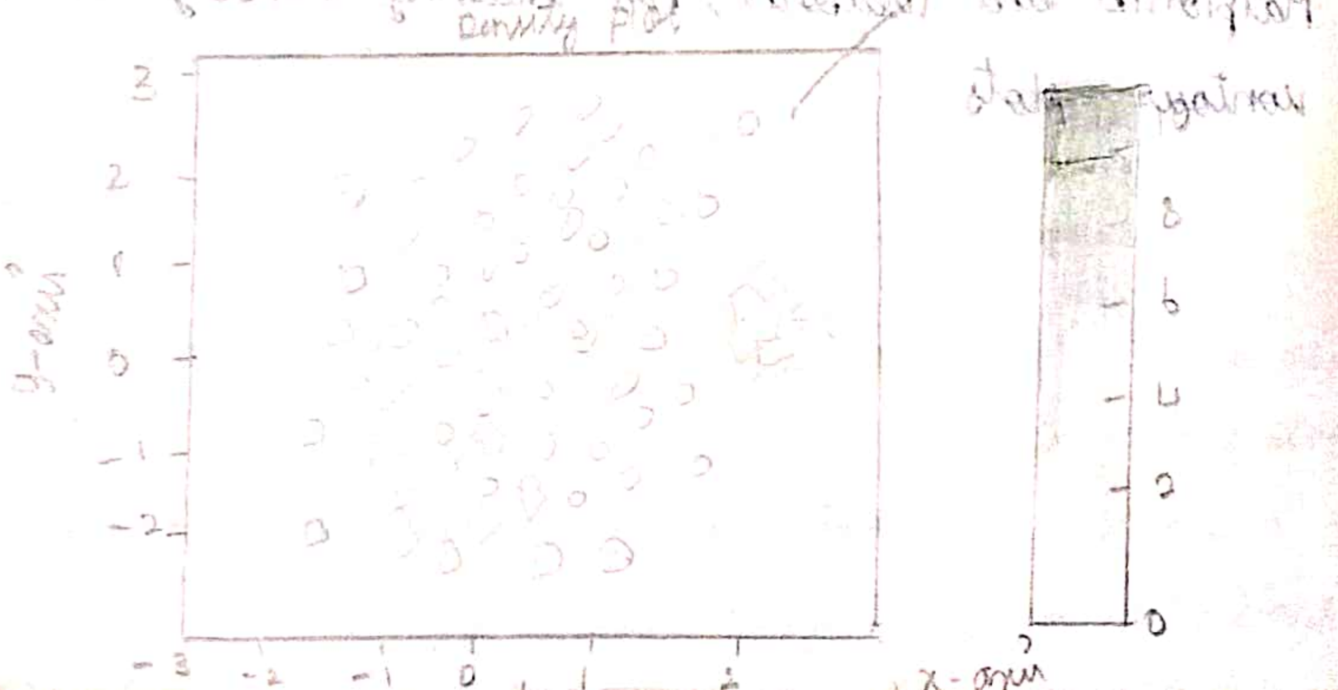
use :

- > visualizing data density ✓
- > Identifying trends ✓
- > spotting patterns ✓

Python provides several libraries, including Matplotlib and seaborn, For creating density and contour plots

Density plot program:

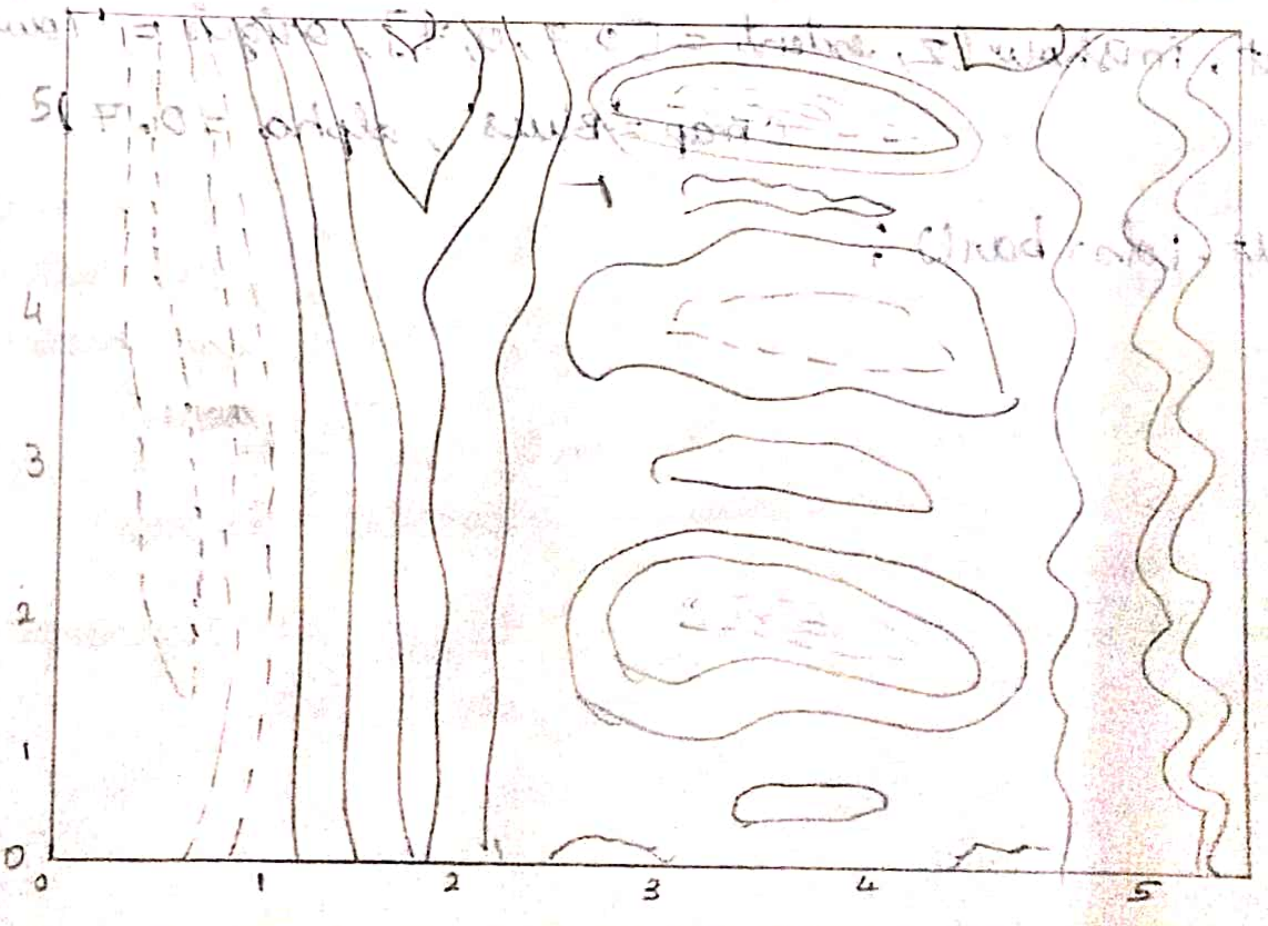
```
import numpy as np
import matplotlib.pyplot as plt

# Generate random data for demonstration
  np. random. seed (0)
  x = np. random. randn (1000)
  y = np. random. randn (1000)

# create a density plot (2D histogram)
                                          color
plt. hist2d (x, y, bins = (40,40), cmap = 'Blues')
plt. colorbar ( )
plt. title ('Density plot')
plt. xlabel ('x-axis')
plt. ylabel ('Y-axis')
plt. show ()
```



meshgrid :

```
import matplotlib.pyplot as plt
plt.style.use('seaborn-white')
import numpy as np

def f(x,y):
    return np.sin(x)**12 + np.cos(15+y*x) *
                                      np.cos(x)

x = np.linespace(0,5,30)
y = np.linespace(0,5,60)

X,Y = np.meshgrid(x,y)
Z = f(X,Y)
plt.contour(X,Y,Z, colours='blue');
plt.contour(X,Y,Z,20, cmap="Blues"); → To change full colour into blue
```

$\sin^{12}$



As the same coding, add

levels. plot.

```
plt.contourf(X,Y,Z,20,cmap='Blues')
plt.colorbar();
```

blue regions are "peaks", white regions are "valleys".
(In simple words, Dark lines are peaks, Dotted lines are valleys.)

show the color variation in the plot diagram.

plt.imshow:

```
contours = plt.contour(X,Y,Z,3, colors='blue')
plt.clabel(contours, inline=True, fontsize=10)
plt.imshow(Z, extent=[0,7,0,7], origin='lower',
           cmap='Blues', alpha=0.7)
plt.colorbar();
```

It is used to show the image in semi-transparent.

Distribution :

distribution refers to the way data is spread or organised within a dataset.

Histograms, box plots, density plots, and quantitle-quantitle (Q-Q) plots are commonly used visualizations to understand and represent data distributions.

Distributions are widely used in various fields, including finance, engineering, social sciences, biology, and more, to model and analyze real-world data.

Histogram :

> A histogram is a graphical representation of the distribution of a dataset.

> Data is divided into intervals or "bins", and the number of data points that fall into each bin is represented as bars or rectangles.

Bins : The range of data values is divided into several intervals or bins.

Frequency : The height of each bar or rectangle in the histogram.

Continuous Data : Histograms are commonly used to represent continuous data, such as measurements, but they can also be used for discrete data.

Uses :

> gaining insights into the distribution of data.

> identifying patterns (or) outliers.

> understanding the central tendency and spread of the dataset.

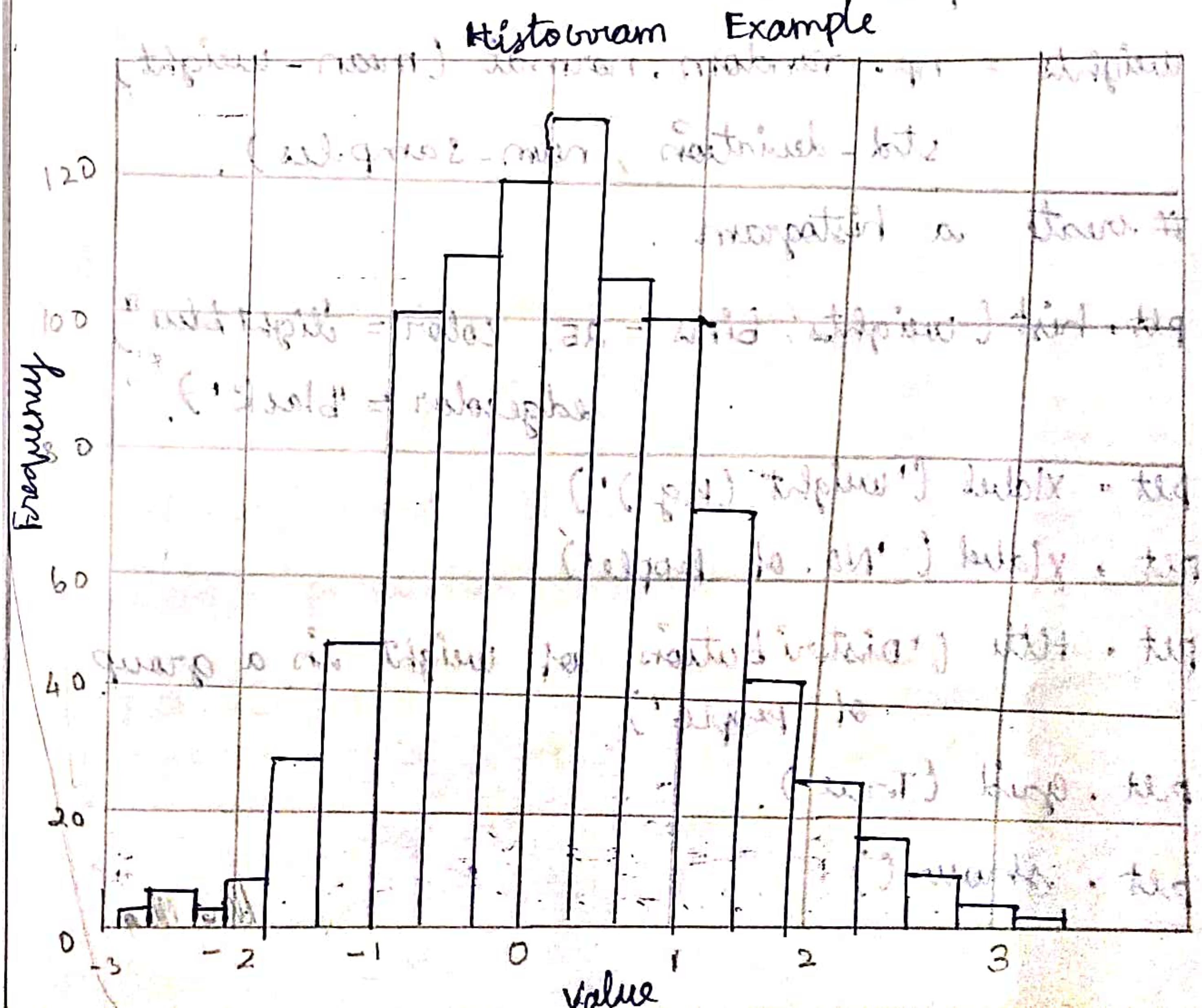
Various fields :

Statistics
data analysis
data visualization.

Sample code for histogram :

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

# Generate 1000 random data
```

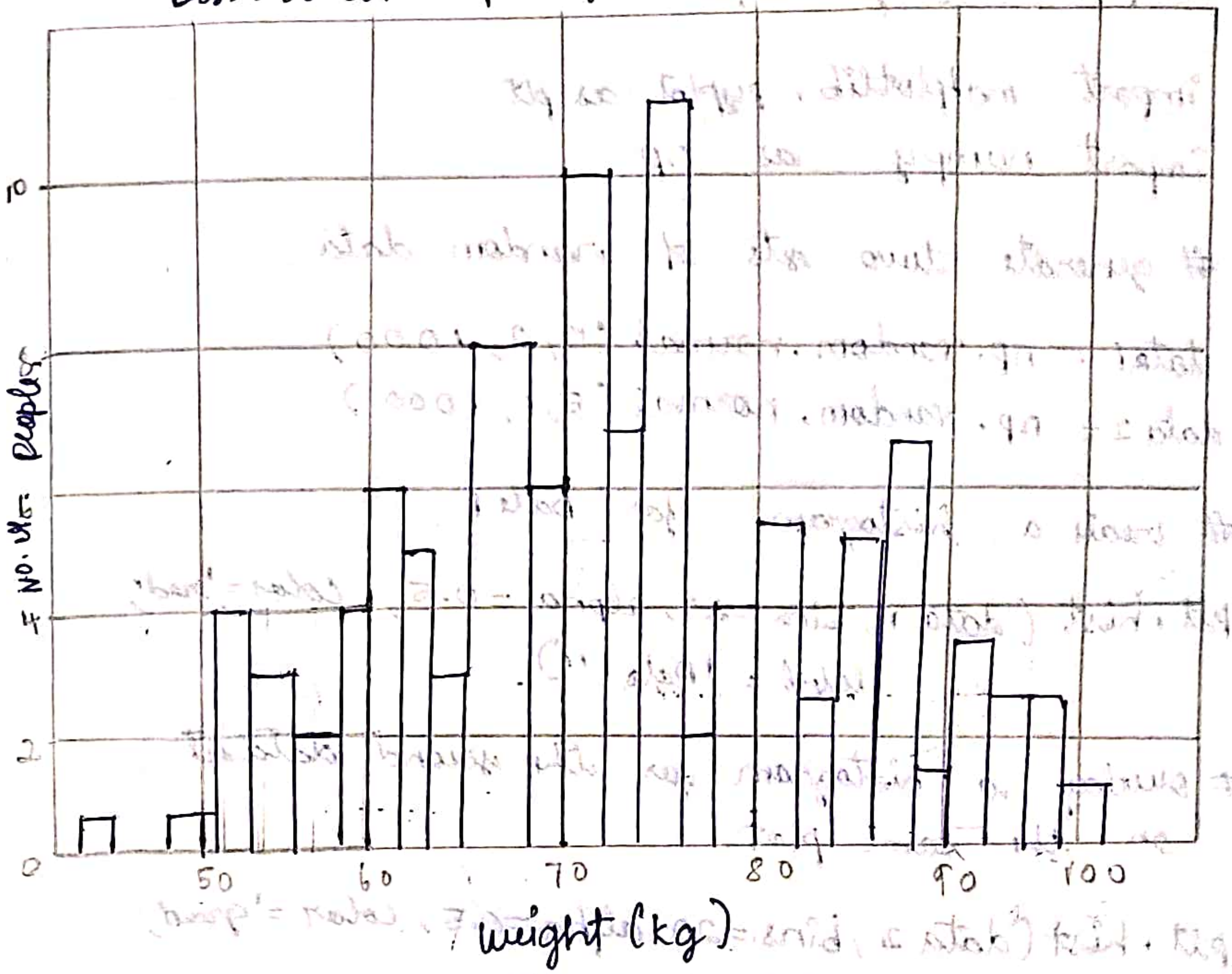
```
data = np.random.randn(1000)

# create a histogram.

plt.hist(data, bins=20, color='skyblue',
         edgecolor='black')

plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram Example')
plt.grid(True)
plt.show()
```



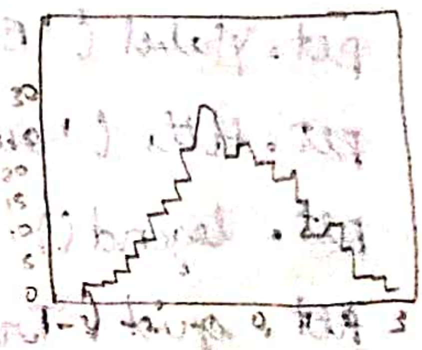
code to show students weight.

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

# generate a random weight data for a
  group of people.

np.random.seed(0)  # For reproducibility.
mean_weight = 70  # Mean weight in kg
std_deviation = 10  # sd in kg.
num_samples = 100
weights = np.random.normal(mean_weight,
            std_deviation, num_samples).
# create a histogram.
plt.hist(weights, bins = 25, color = 'light blue',
                  edgecolor = 'black').
plt.xlabel('weight (kg)')
plt.ylabel('No. of people')
plt.title('Distribution of weight in a group
           of people')
plt.grid(True).
plt.show()
```

Distribution of weight in a group of people



Eg -2:



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

plt.style.use('seaborn-white')
data = np.random.randn(1000)

plt.hist(data, bins 20, density = True, alpha=0.7,
         histtype ='stepfilled', color = 'blue',
         edge color ='green')

plt.show().
```

Multiple Histogram :

import matplotlib. pyplot as plt.
import numpy as np

generate two sets of random data.

data1 = np. random .normal (0, 2, 1000)
data 2= np. random. normal (5, 1, 1000)

create a histogram. for Data 1.

plt. hist (data 1, bins =20, alpha =0.5, color = 'red',
label = 'Data1'').

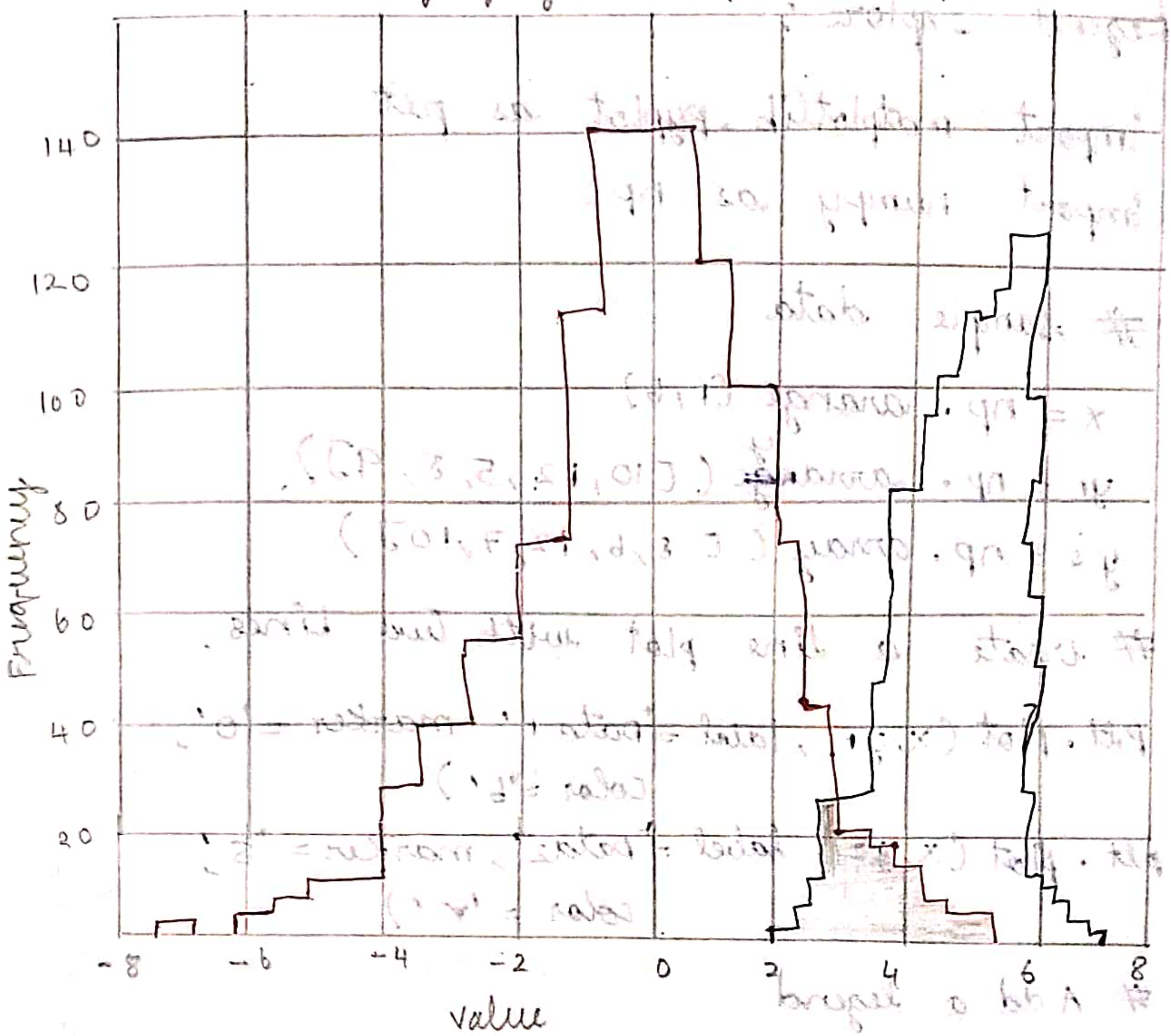
overlay a histogram for the second dataset on the same plot.

plt. hist (data 2, bins =20, alpha =0.5, color ='green',
label ='Data2'').

plt. xlabel ('Value')
plt. ylabel ('Frequency')
plt. title ('overlaying Multiple Histograms')
plt. legend ()
plt. grid (True)

plt. show()

Overlaying Multiple Histograms



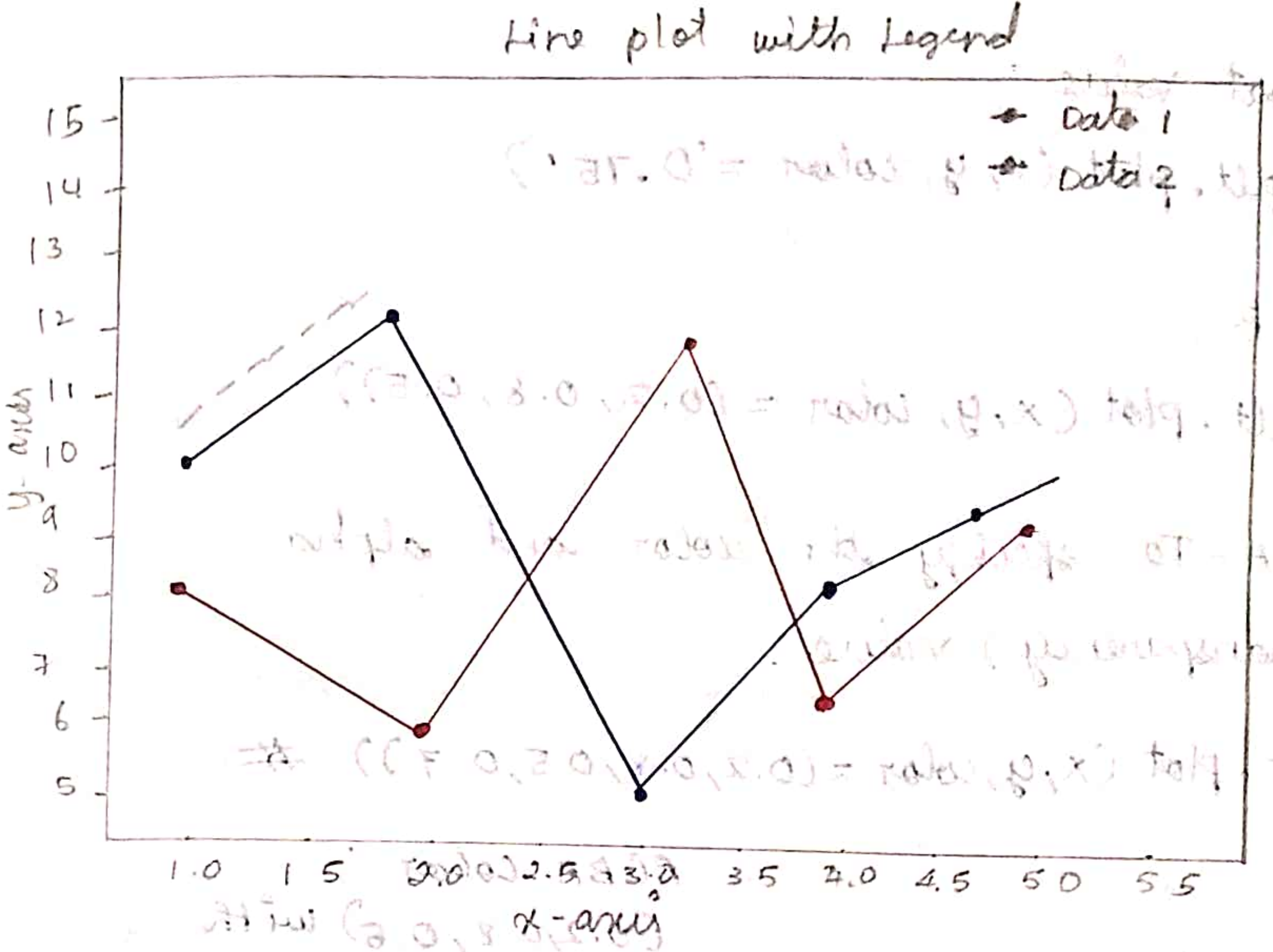
Legends in a Matplotlib:

Legends in a Matplotlib plot provide additional information about the elements on the plot, such as the meaning of different colors, line styles, or markers to improve the clarity of the plot.

It easier for viewers to understand the data.

Legend explore :

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

# sample data
 x = np. arange (1,6)
 y1 = np. array ( [10, 12, 5, 8, 9])
 y2 = np. array ( [ 8, 6, 12, 7, 10] )
# create a line plot with two lines.
plt. plot (x, y1, label = 'Data 1', marker = 'o',
                          color = 'b')
plt. plot (x, y2, label = 'Data2', marker = 's',
                          color = 'r')

# Add a legend
 plt. legend ()

plt. title ('Line Plot with Legend')
plt . xlabel ('X-axis')
plt . ylabel ( 'y-axis')
plt. show ().
```



Colors :

In Matplotlib, colors can be specified in several ways,

> by name

> hexadecimal RGB Value

> float value between 0 and 1

> RGB or RGBA color.

By Name :

plt. plot (x, y , color ='red')

Hexadecimal RGB Value

plt. plot (x, y, color = '#FF5733')

Bayesian Data Analysis:

Problem definition → Data collection → Model development → prior distribution → Data Analysis → Results communication.

Prior distribution:

Before observing any data, we need to specify prior distributions for the model parameters, lets assume a weakly informative prior for θ, such as a normal distribution centered at 0 with moderate standard deviation.

$$\frac{50 \times 70}{100} = 35 \qquad \frac{50 \times 30}{100} = 15$$

35 15

Contigency Table :

gender (Male / Female) → Voting preference (A/B).

	Voting A	Voting B	Total
Male	30	20	50
Female	40	10	50
Total	70	30	100

70 × 50 ÷ 100

1. Examine the Table :

> Observe the counts in each cell, noting the distribution of responses.

2. Row & column Margins:

> Calculate row & column Totals to identify the marginal distributions.

> Row Marginal (Total in each row): 50, 50, 100.

> Column Margins (" column): 70, 30, 100.

3. Calculate Percentages :

> compute Percentages For each cell, row, & column.

> cell Percentage (eg. Male who voted A)

$30/100 \times 100 = 30\%$.

> Row Percentage (eg. Male who voted A out of Male Total) %.

$30/50 \times 100 = 60\%$

> Column Percentage (eg. Male who voted A out of total who voted A):

$30/70 \times 100 = 42.86\%$.

4. Assess Independence:

> Examine whether the distribution of one variable is independent of the other or if there is an association.

5. Chi-square Test:

> use the chi-square test to determine if there is a significant association between the variables.

> calculate the chi-square statistic & compare it to the critical value.

6. Interpretation Results:

> If the p-value from the chi-square test is significant, reject the null hypothesis of independence.

7. Visualization:

$dof = (r-1)(c-1)$

$dof =$

> create visualizations like a clustered bar chart or a stacked bar chart to illustrate the relationships visually.

O	E	O − E	$(O-E)^2$	$\frac{(O-E)^2}{E}$
30	35	−5	25	0.71
40	35	5	25	0.71
20	15	+5	25	1.66
10	15	−5	25	1.66
				4.74