Matplotlib

Mat

Matplotlib

Matplotlib is a Python library for creating 2D plots and visualizations. It supports a wide range of graph types, is highly customizable, and works well with NumPy for numerical data.

Designed for both simple and complex plotting needs, it's a fundamental tool for data analysis and scientific research, backed by a strong community.

pyplot

Matplotlib's primary features are in the pyplot submodule, offering a MATLAB-like interface for quick and easy plotting.

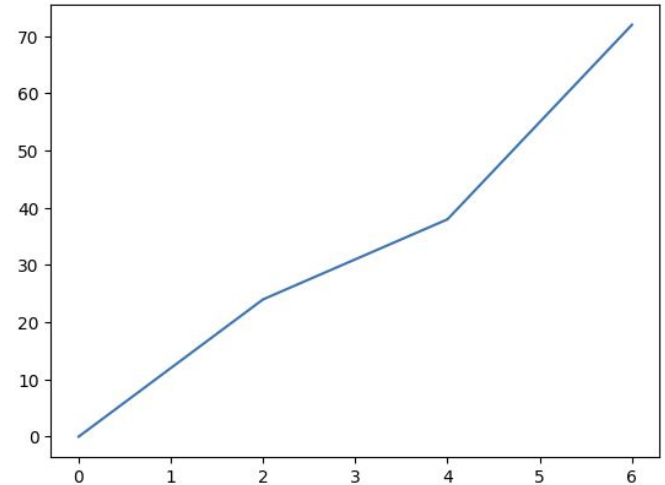
```
import matplotlib.pyplot as plt
```

Example

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

xpoints = np.array([0, 2, 4, 6])
ypoints = np.array([0, 24, 38, 72])

plt.plot(xpoints, ypoints)
plt.show()
```

Result

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

xpoints = np.array([0, 2, 4, 6])
ypoints = np.array([0, 24, 38, 72])

plt.plot(xpoints, ypoints)
plt.show()
```



Plot function

The plot() function creates a diagram by plotting points, connecting them with a line by default.

It accepts parameters to define the points' positions, with the first parameter for the x-axis points and the second for the y-axis points.

Example

```
plt.plot(x, y, marker='o', markerfacecolor='red', linestyle='--', linewidth=2, color='blue')
```

marker='o' specifies the shape of the marker (in this case, a circle).

markerfacecolor='red' sets the color of the markers to red.

linestyle='--' defines the style of the line connecting the points as dashed.

linewidth=2 adjusts the width of the line to 2 units.

color='blue' changes the color of the line to blue.

X axis and Y axis Label & Ttitle

```
plt.title('Customized Plot Example')

plt.xlabel('X axis')

plt.ylabel('Y axis')
```

`plt.show()`

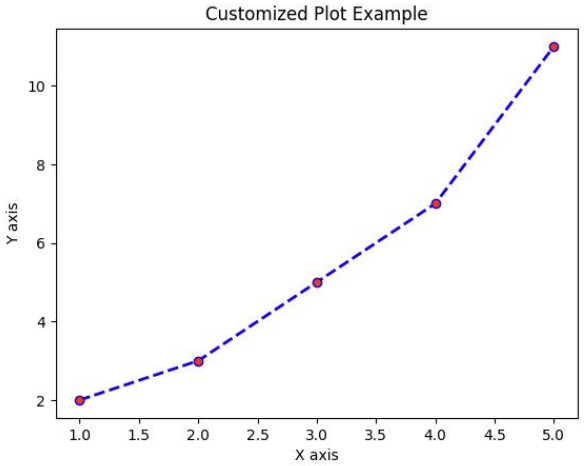
- The `plt.show()` function in Matplotlib displays the plot that you have created.
- It brings up a window that contains the visual representation of your data, as defined by your plotting commands.
- This function is typically called at the end of your plotting commands to render the plot visibly on the screen.

Example

```
import matplotlib.pyplot as plt


# Example data
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]


plt.plot(x, y, marker='o', markerfacecolor='red', linestyle='--', linewidth=2, color='blue')
plt.title('Customized Plot Example')
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.show()
```

Result

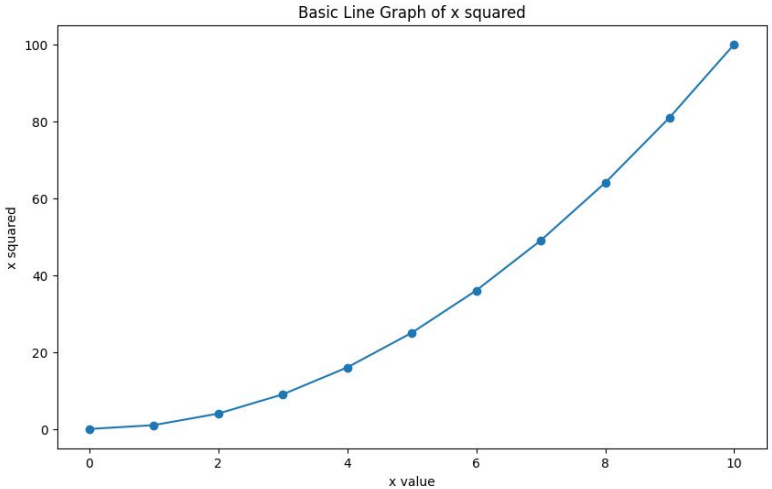
```python
import matplotlib.pyplot as plt

# Example data
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

plt.plot(x, y, marker='o', markerfacecolor='red', linestyle='--', linewidth=2, color='blue')
plt.title('Customized Plot Example')
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.show()
```



Lab 31-1

Generate a line graph with the x-axis ranging from 0 to 10 and the y-axis representing the square of the x values.

Add a title, and labels for the x-axis and y-axis to the graph.



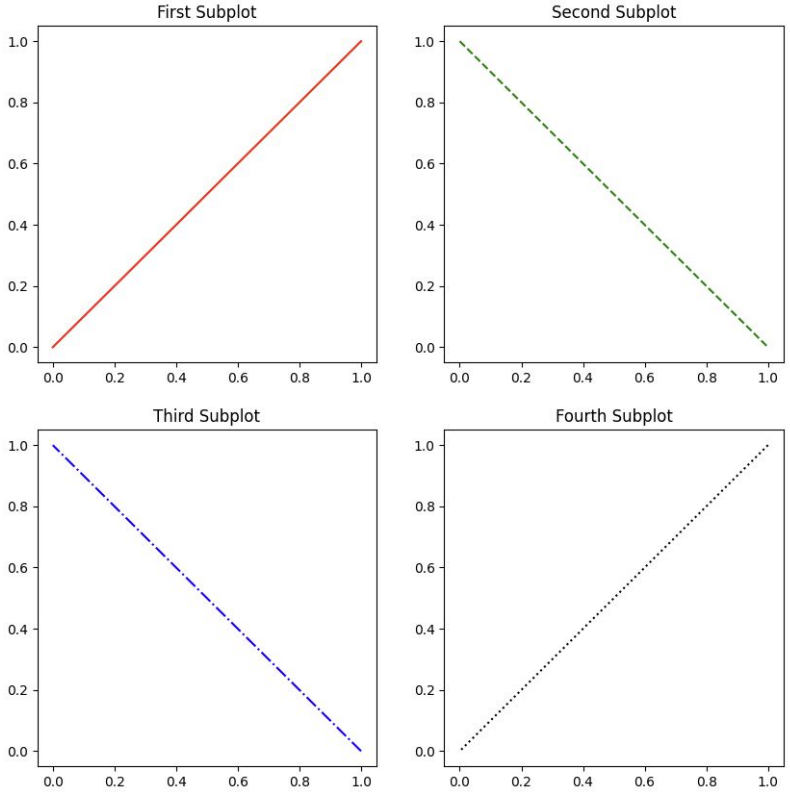
`plt.subplot()`

- The plt.subplot() function in Matplotlib is used to add a subplot to the current figure. It's a way to organize multiple plots in a single figure.
- The plt.subplot() function takes three arguments: nrows, ncols, and index.
  - nrows and ncols divide the figure into a grid with a specified number of rows and columns
  - index specifies the position of the subplot in the grid (indexing starts at 1, from left to right and then top to bottom).

Example

```
import matplotlib.pyplot as plt

# Create a figure

plt.figure(figsize=(10, 10))

# First subplot

plt.subplot(2, 2, 1)  # (rows, columns, panel number)

plt.plot([0, 1], [0, 1], 'r-')  # red line

plt.title('First Subplot')

# Second subplot

plt.subplot(2, 2, 2)

plt.plot([0, 1], [1, 0], 'g--')  # green dashed line

plt.title('Second Subplot')

# Third subplot

plt.subplot(2, 2, 3)

plt.plot([1, 0], [0, 1], 'b-.')  # blue dot-dashed line

plt.title('Third Subplot')

# Fourth subplot

plt.subplot(2, 2, 4)

plt.plot([1, 0], [1, 0], 'k:')  # black dotted line

plt.title('Fourth Subplot')

# Display the figure with subplots

plt.show()
```

Result


First Subplot
Second Subplot
Third Subplot
Fourth Subplot
0.0
0.2
0.4
0.6
0.8
1.0

Other plots

- Scatter
- Bars
- Histograms
- Pie Charts

plt.scatter()

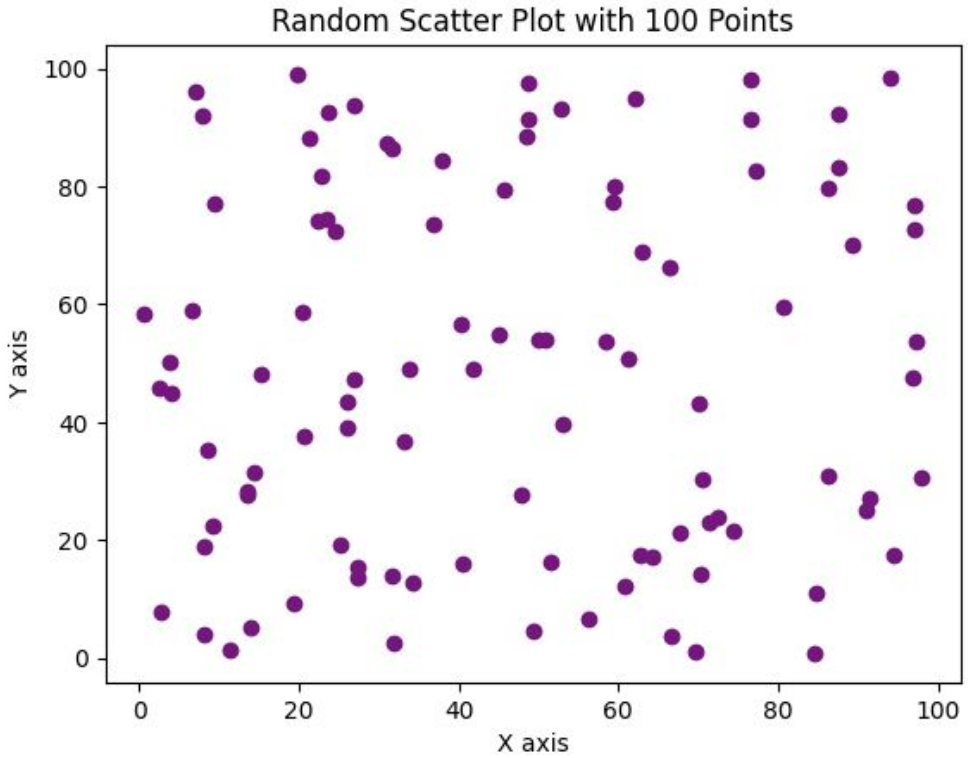
The plt.scatter() function in Matplotlib is used to create scatter plots, which are diagrams where each data point in the dataset is represented as a point on the plot.

This function is particularly useful for visualizing the relationship between two variables, allowing for the observation of how they correlate with each other.

Example

```
import matplotlib.pyplot as plt

import numpy as np

# Generate 100 random data points for x and y

x = np.random.rand(100) * 100  # 100 random values scaled to 0-100

y = np.random.rand(100) * 100  # 100 random values scaled to 0-100

# Create scatter plot

plt.scatter(x, y, color='purple', marker='o')

# Title and labels

plt.title('Random Scatter Plot with 100 Points')

plt.xlabel('X axis')

plt.ylabel('Y axis')

# Show plot

plt.show()
```

Result


Random Scatter Plot with 100 Points
Y axis
X axis
0
20
40
60
80
100

Lab 31-2

Generate Data: Create two sets of 50 random numbers each for both x and y coordinates. These numbers should be between 0 and 1.

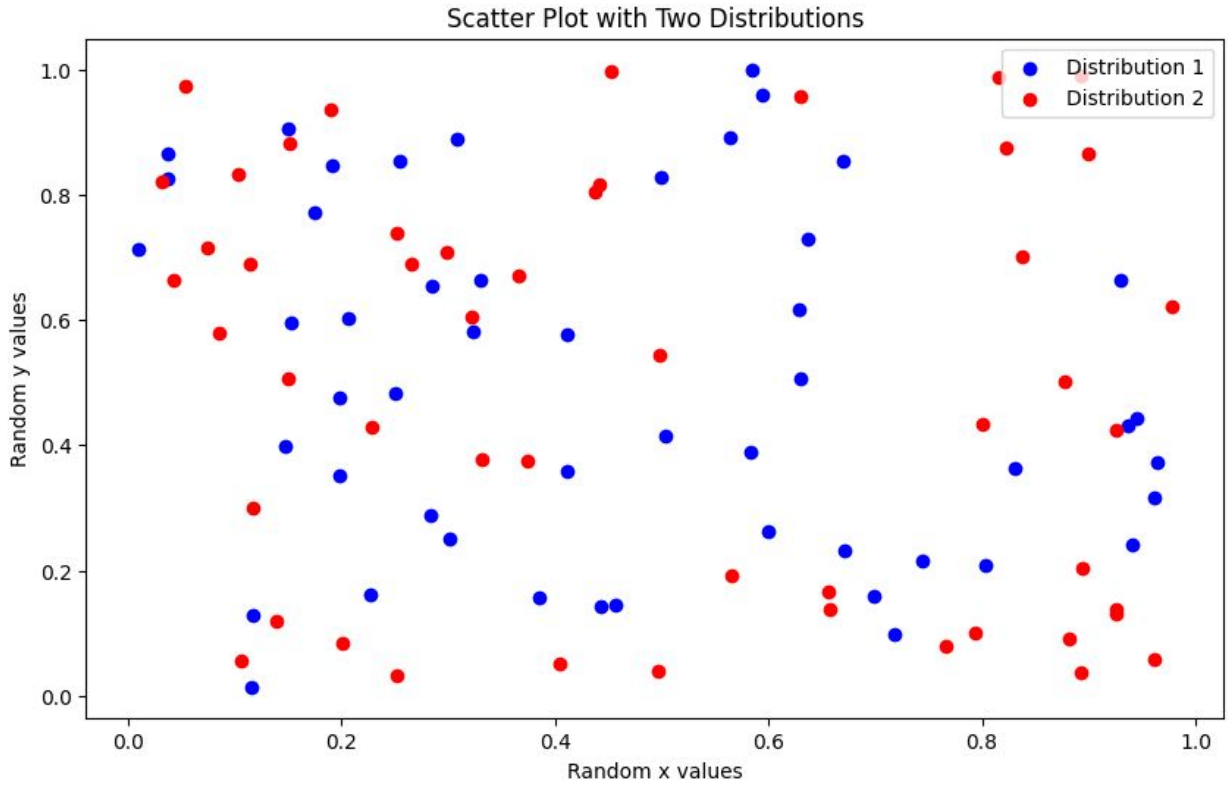
Create Scatter Plot: Plot both sets of data on the same graph. Use blue for the first dataset and red for the second.

Customize Your Plot: Add a title "Scatter Plot with Two Distributions", label both axes as "Random values", and include a legend to identify the datasets.

Add a title and labels for the axes to the graph.

Optionally, try to add a legend using `plt.legend()` too.

Example of Output



`plt.bar()`

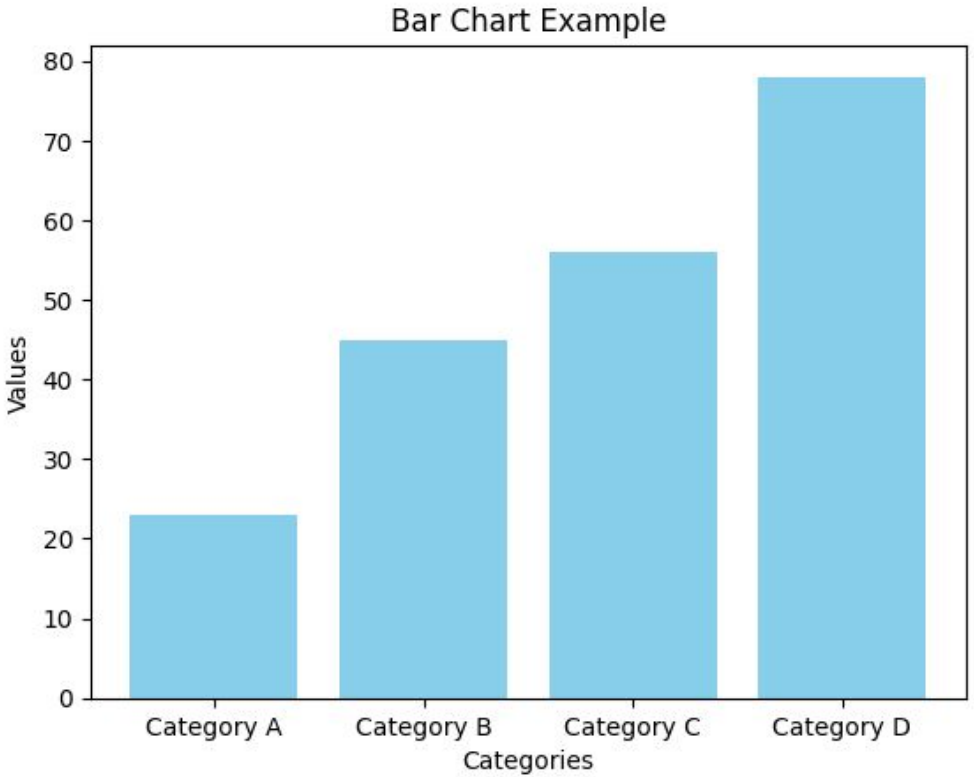
plt.bar() function in Matplotlib is used for creating bar charts, which are graphical representations of data using rectangular bars where the length of each bar is proportional to the value it represents.

Bar charts are useful for comparing different groups or tracking changes over time when the changes are relatively small.

Example

```
import matplotlib.pyplot as plt

# Example categories and values

categories = ['Category A', 'Category B', 'Category C', 'Category D']

values = [23, 45, 56, 78]

# Create bar chart

plt.bar(categories, values, color='skyblue')

# Title and labels

plt.title('Bar Chart Example')

plt.xlabel('Categories')

plt.ylabel('Values')

# Show plot

plt.show()
```

Result


Bar Chart Example
Values
0
10
20
30
40
50
60
70
80
Category A
Category B
Category C
Category D
Categories

Lab 31-3

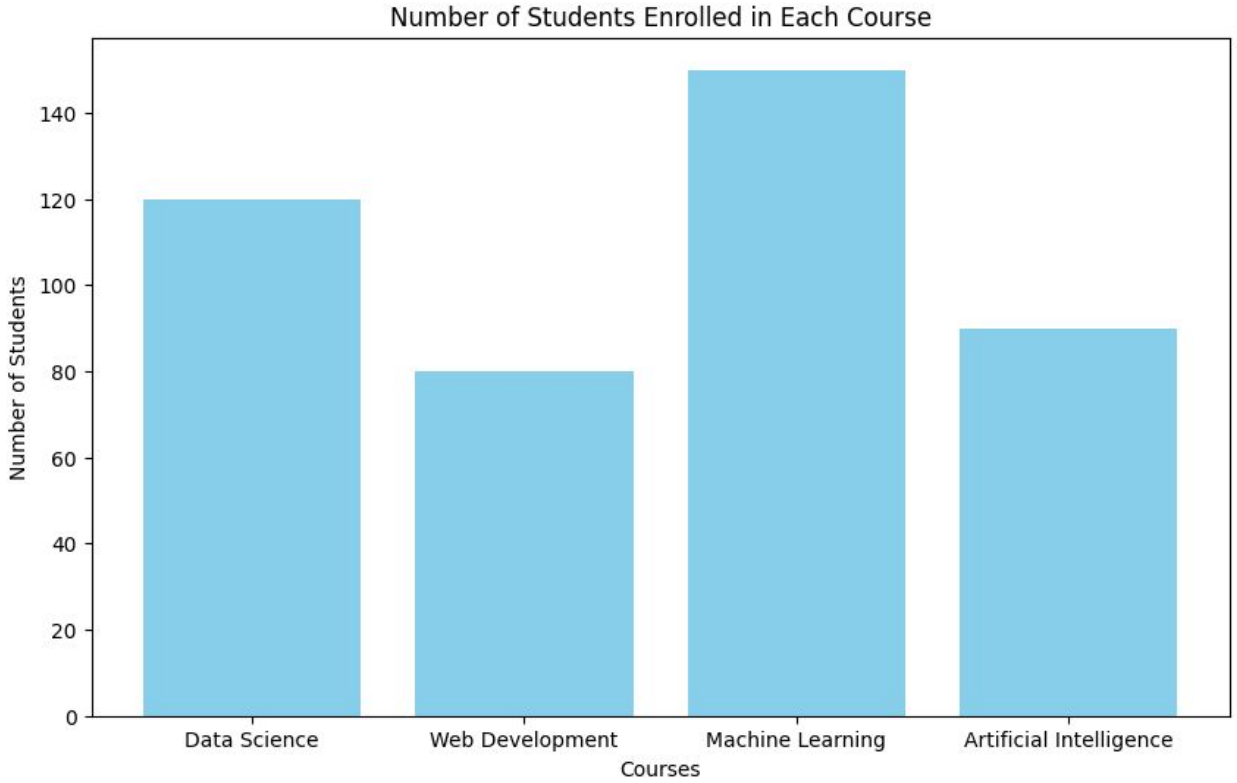
Use the following data to generate a bar chart:

```
Courses = ['Data Science', 'Web Development', 'Machine Learning', 'Artificial Intelligence']

Students = [120, 80, 150, 90]
```

Label each bar with the name of the course.

Add a title and labels for the axes to the graph.

Example of Output



`plt.hist()`

`plt.hist()` function in Matplotlib is used to create histograms, which are a type of bar chart that represents the distribution of numerical data.

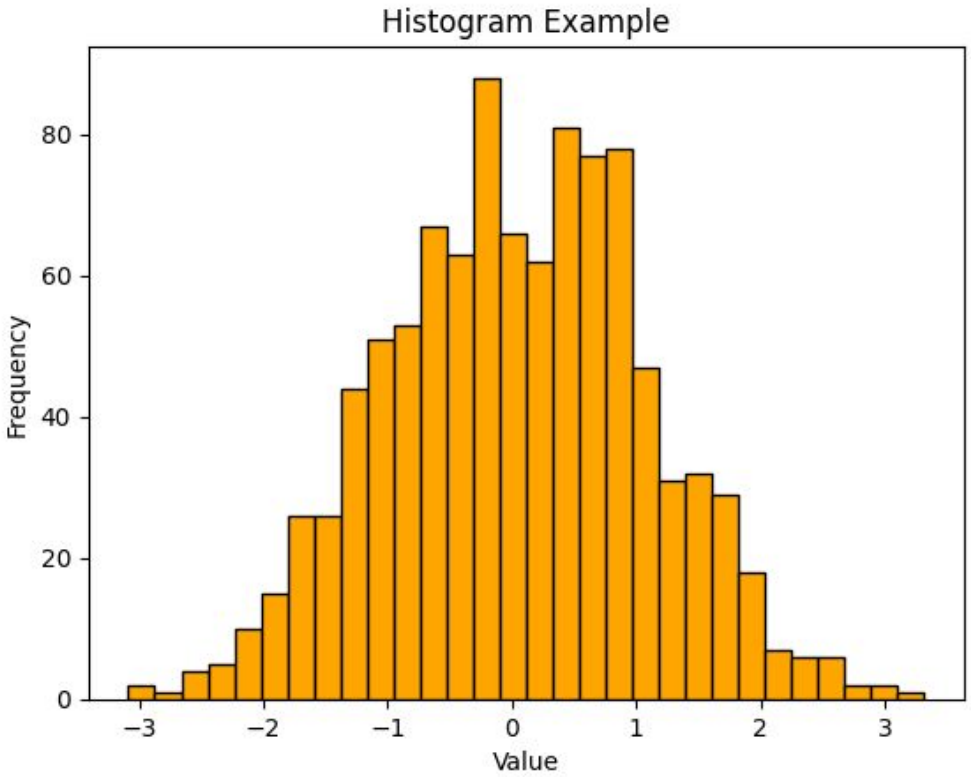
Unlike bar charts that compare different groups, histograms group data points into bins and show the frequency of data points within each bin.

This makes histograms ideal for understanding the distribution, variability, or central tendency of data.

Example

```
import matplotlib.pyplot as plt

import numpy as np

# Generating random data

data = np.random.randn(1000)  # 1000 random data points from a normal distribution

# Create histogram

plt.hist(data, bins=30, color='orange', edgecolor='black')

# Title and labels

plt.title('Histogram Example')

plt.xlabel('Value')

plt.ylabel('Frequency')

# Show plot

plt.show()
```

Result



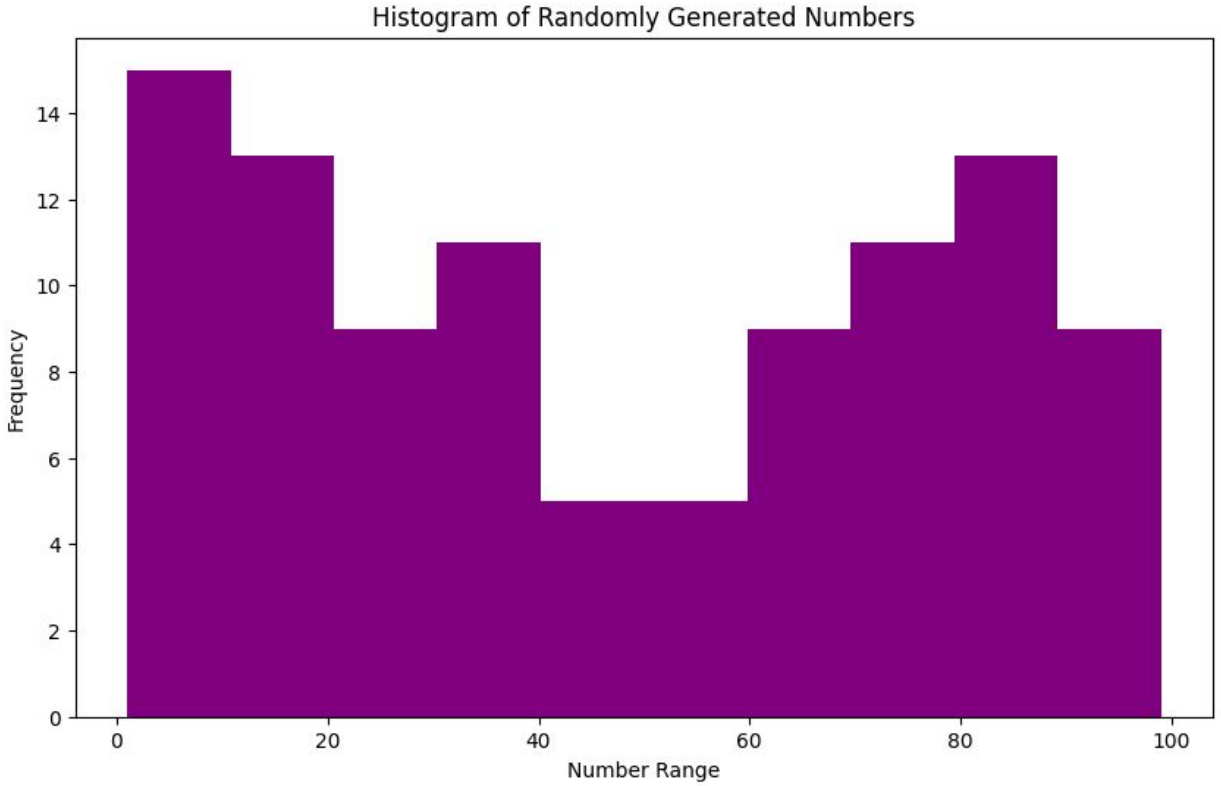
Lab 31-4

Generate 100 random numbers between 0 and 100 and use these to create a histogram.

Use 10 bins in the histogram.

Add a title and labels for the axes to the graph.

Example of Output



`plt.pie()`

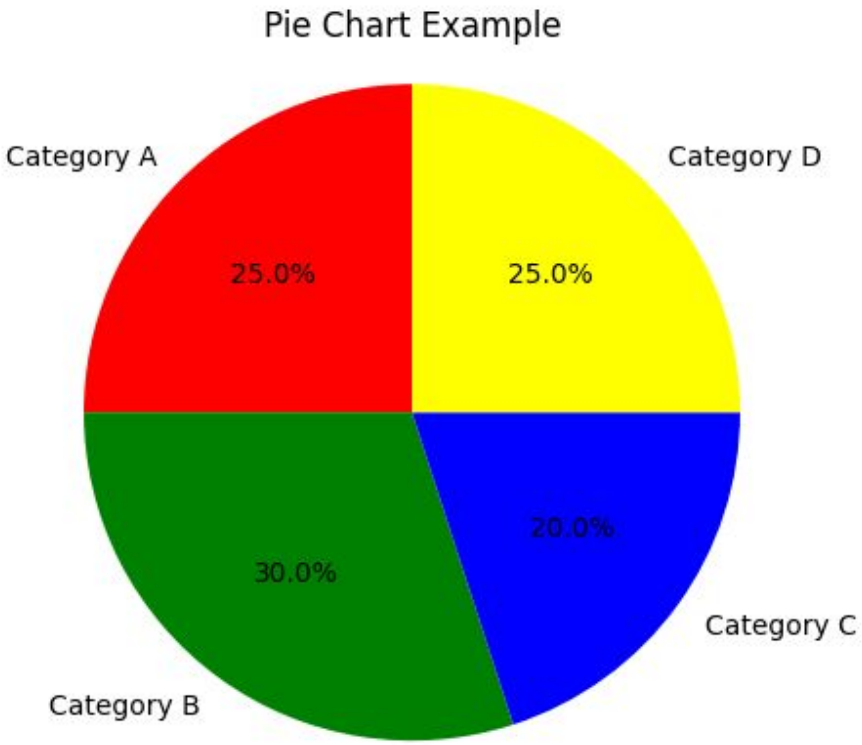
`plt.pie()` function in Matplotlib creates pie charts, which are circular statistical graphics divided into slices to illustrate numerical proportion.

Each slice of the pie chart represents a category's contribution to the whole, with the size of each slice being proportional to the quantity it represents.

Example

```
import matplotlib.pyplot as plt

# Data

sizes = [25, 30, 20, 25]

labels = ['Category A', 'Category B', 'Category C', 'Category D']

# Create pie chart

plt.pie(sizes, labels=labels, autopct='%.1f%%', startangle=90, colors=['red', 'green', 'blue', 'yellow'])

# Equal aspect ratio ensures that pie is drawn as a circle.

plt.axis('equal')

# Title

plt.title('Pie Chart Example')

# Show plot

plt.show()
```

Result


Pie Chart Example
Category A
Category D
25.0%
25.0%
20.0%
30.0%
Category C
Category B