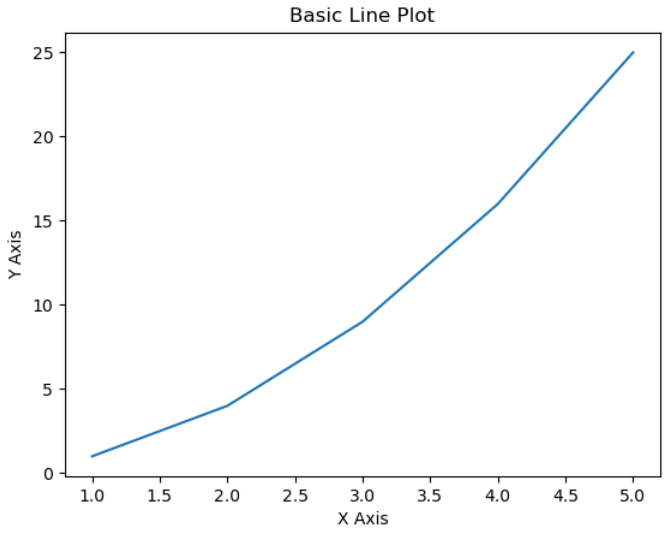
Data Visualization with Matplotlib

1. Basic Line Plot

The following code snippet demonstrates how to create a basic line plot using Matplotlib.

```
import matplotlib.pyplot as plt x = [1, 2, 3, 4, 5] y = [1, 4, 9, 16, 25] plt.plot(x, y) plt.xlabel('X Axis') plt.ylabel('Y Axis') plt.title('Basic Line Plot') plt.show()
```

This code creates a simple line plot with labeled axes and a title.

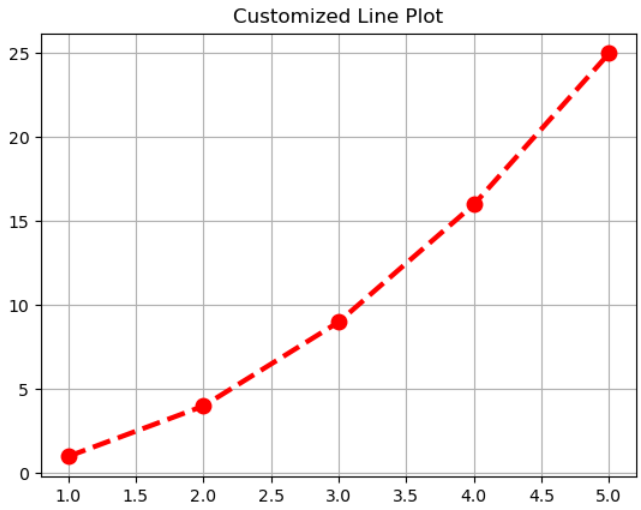


2. Customized Line Plot

You can customize the appearance of the plot using various parameters.

```
plt.plot(x, y, color='Red', linestyle='--', marker='o', linewidth=3, markersize=9) plt.grid(True) plt.show()
```

This code customizes the line plot with color, line style, markers, and grid.

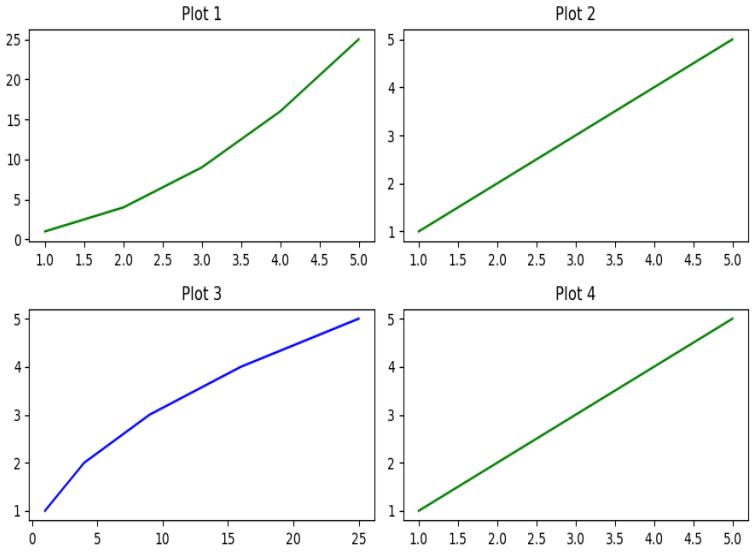


3. Multiple Plots

You can create multiple subplots in a single figure.

```
plt.figure(figsize=(9, 5)) plt.subplot(2, 2, 1) plt.plot(x, y1,
color='green') plt.title('Plot 1') plt.subplot(2, 2, 2) plt.plot(x, y2,
color='green') plt.title('Plot 2') plt.subplot(2, 2, 3) plt.plot(y1, x,
color='blue') plt.title('Plot 3') plt.subplot(2, 2, 4) plt.plot(y2, x,
color='green') plt.title('Plot 4') plt.tight_layout() plt.show()
```

This code creates a 2x2 grid of subplots, each with its own title.

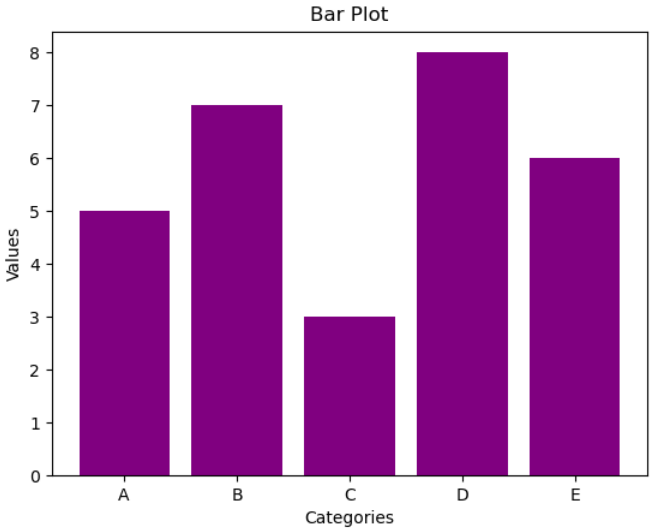


4. Bar Plot

Bar plots are useful for comparing quantities across different categories.

```
categories = ['A', 'B', 'C', 'D', 'E'] values = [5, 7, 3, 8, 6]
plt.bar(categories, values, color='purple') plt.xlabel('Categories')
plt.ylabel('Values') plt.title('Bar Plot') plt.show()
```

This code creates a bar plot with labeled axes and a title.

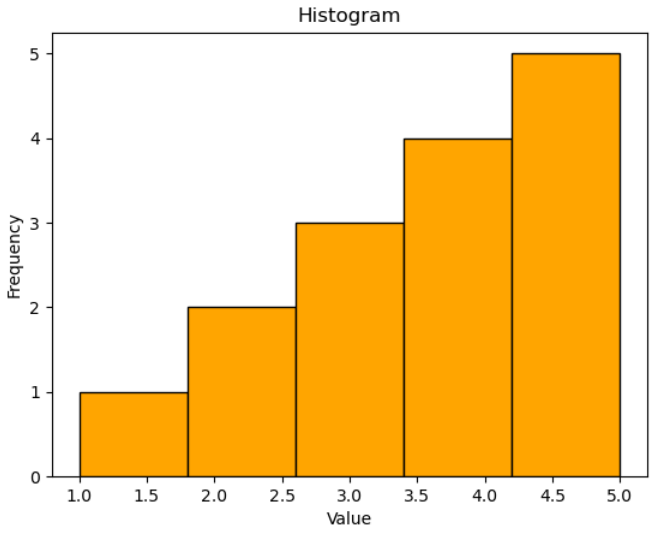


5. Histograms

Histograms represent the distribution of a dataset.

```
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5] plt.hist(data, bins=5, color='orange', edgecolor='black') plt.title('Histogram') plt.xlabel('Value') plt.ylabel('Frequency') plt.show()
```

This code creates a histogram with specified bins and colors.

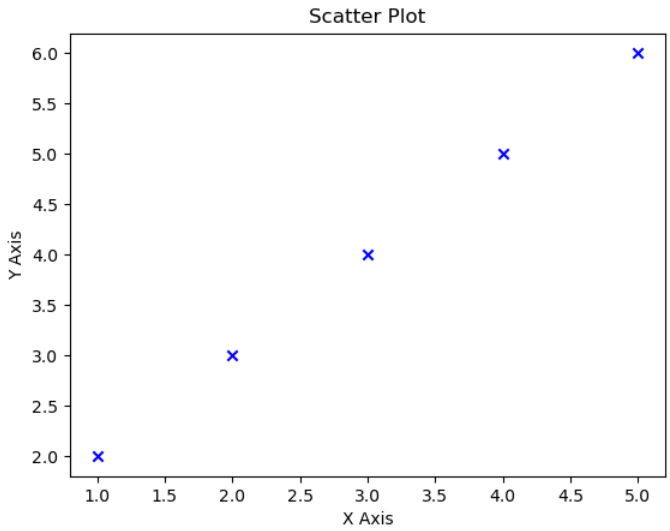


6. Scatter Plot

Scatter plots are used to observe relationships between variables.

```
x = [1, 2, 3, 4, 5] y = [2, 3, 4, 5, 6] plt.scatter(x, y, color='blue', marker='x') plt.title('Scatter Plot') plt.xlabel('X Axis') plt.ylabel('Y Axis') plt.show()
```

This code creates a scatter plot with specified colors and markers.

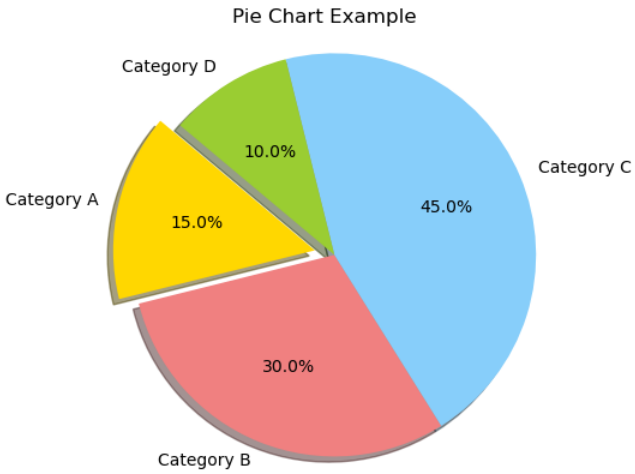


7. Pie Chart

Pie charts are useful for showing the proportions of a whole.

```
labels = ['Category A', 'Category B', 'Category C', 'Category D'] sizes = [15, 30, 45, 10] colors = ['gold', 'lightcoral', 'lightskyblue', 'yellowgreen'] explode = (0.1, 0, 0, 0) # explode the 1st slice plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140) plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle. plt.title('Pie Chart Example') plt.show()
```

This code creates a pie chart with labeled slices and percentages.

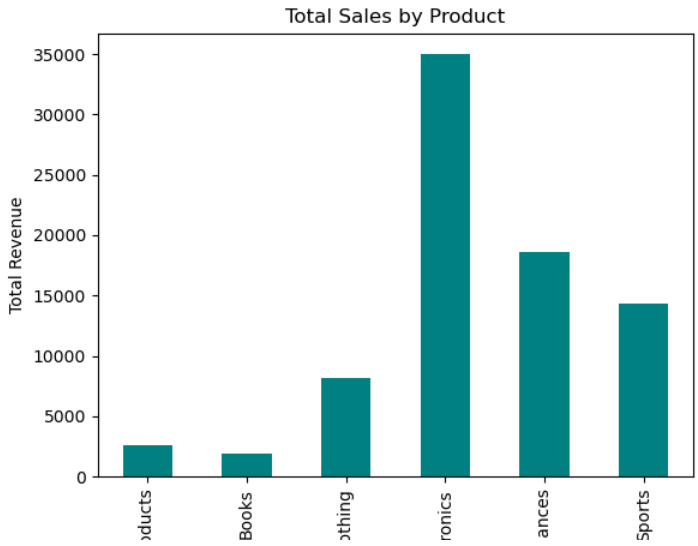


8. Total Sales by Product

This section visualizes total sales by product category.

```
# plot total sales by products
total_sales_by_product=sales_data_df.groupby('Product Category')['Total
Revenue'].sum() total_sales_by_product.plot(kind='bar', color='teal')
plt.xlabel('Product Category') plt.ylabel('Total Revenue') plt.title('Total
Sales by Product') plt.show()
```

This code creates a bar plot showing total sales for each product category.



9. Sales Trend Over Time

This section visualizes the sales trend over time.

```
sales_trend = sales_data_df.groupby('Date')['Total
Revenue'].sum().reset_index() plt.plot(sales_trend['Date'],
sales_trend['Total Revenue']) plt.xlabel('Date') plt.ylabel('Total Revenue')
plt.title('Sales Trend Over Time') plt.xticks(rotation=45) plt.show()
```

This code creates a line plot showing the sales trend over time.