Basic Plotting and Customization

Basic plotting involves the fundamental techniques of representing data visually. It serves as the first step in transforming raw data into meaningful insights.

There are two types of basic plots:

a. Line Chart
b. Bar Chart

a. Line Chart: A line chart is a type of chart that displays information using a series of data points called markers, connected by straight line segments. It is often used to visualize trends over a continuous interval or time span.

Line charts are particularly useful for tracking changes over time, illustrating the progression of variables, or identifying correlations between different datasets. In a line chart, data points are connected by lines, making it easy to observe the overall trajectory.

- **Purpose:** Line charts are employed to showcase trends and patterns over a continuous set of data points.
- **Application:** Useful for tracking changes over time, illustrating the progression of variables, or identifying correlations.
- **Matplotlib Function:** The **plot()** function is utilized to create line charts, providing flexibility in terms of styling and representation.

b. Bar Chart: This type of chart presents categorical data with rectangular bars. Each bar's length or height is proportional to the data it represents. Bar charts can be used to compare the values of different categories.

Bar charts are particularly effective in highlighting disparities between categories, making them ideal for tasks such as comparing sales figures across various regions or displaying the frequency of

pecific categories. In a bar chart, each category is represented by a bar, the length of which corresponds to the value it represents.

- **Purpose:** Bar charts are designed for comparing categorical data or displaying the distribution of values across different groups.
- **Application:** Ideal for visualizing disparities between groups, such as comparing sales figures across regions or displaying the frequency of specific categories.
- **Matplotlib Function:** The **bar()** function, along with its variations, enables the creation of effective bar charts.

Let's learn to create line and bar charts using simple Python code..

a. **Creating Line Charts:** We can create line charts using **plot()** function.

Example Code:

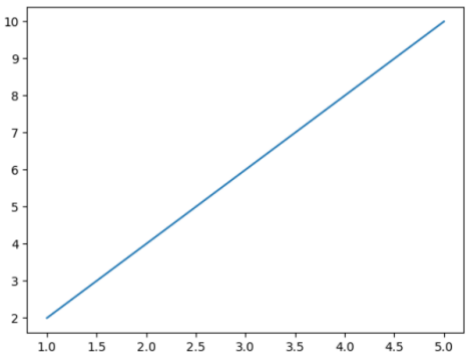
```
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Create a line chart
plt.plot(x, y)

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

Output:



The code begins by importing the pyplot module from the matplotlib library and aliasing it as plt.

This code creates a simple line chart using Matplotlib with the given sample data and then displays the chart. The X-axis values are [1, 2, 3, 4, 5], and the corresponding Y-axis values are [2, 4, 6, 8, 10].

b. Creating Bar Charts: Bar charts are useful for comparing values across different categories. The **bar()** function is used for making Bar charts.

Example Code:

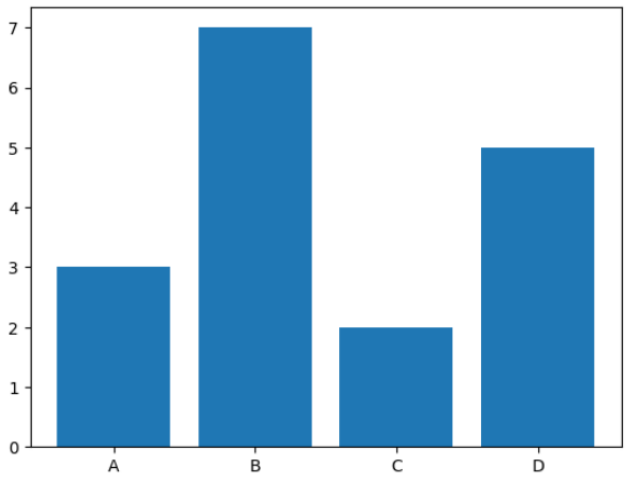
```
import matplotlib.pyplot as plt

# Sample data
categories = ['A', 'B', 'C', 'D']
values = [3, 7, 2, 5]

# Create a bar chart
plt.bar(categories, values)

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

Output:



In the code, two lists named **categories** and **values**, are defined. The list **categories** contains labels for different categories (e.g., 'A', 'B', 'C', 'D'), and **values** contains corresponding numerical values. The **bar** function from the **pyplot** module is used to create a bar chart. It takes **categories** as the x-axis labels and **values** as the heights of the bars.

As visible in the bar chart, each category is associated with a bar, and the height of the bar represents the value for that category. The bar function from the pyplot module is used to create a bar chart. It takes categories as the x-axis labels and values as the heights of the bars. Each category is associated with a bar, and the height of the bar represents the value for that category.

Customizing Plots with Matplotlib:

Customizing plots refers to the process of modifying the visual appearance and layout of graphs and charts to better suit the specific requirements, preferences, or storytelling goals of the data analyst, scientist, or audience. This involves adjusting various elements of a plot, such as colors, line styles, markers, fonts, labels, titles, axes etc.

Customization plays a pivotal role in data visualization, allowing you to tailor plots to convey information more effectively and make them visually appealing. The goal is to enhance the clarity, readability, and visual appeal of the data visualization.

Importance of Customizing Plots:

1. **Enhancing Interpretation:** Customization enhances the interpretability of visualizations. Distinct colours, markers, and line styles make it easier for viewers to differentiate between data series.
2. **Aesthetics and Readability:** Aesthetically pleasing visualizations are more likely to engage and inform the audience. Customization improves the readability of the chart.
3. **Contextual Information:** Titles, axis labels, and legends provide contextual information. Viewers can quickly understand the purpose of the chart, the variables being represented, and the significance of different elements.
4. **Communication of Insights:** By customizing elements, you can draw attention to specific details, emphasize trends, or highlight outliers, effectively communicating insights derived from the data.
5. **Tailoring to Audience:** Customization allows you to tailor visualizations to the preferences and expectations of your audience. Different audiences may require different levels of detail or specific stylistic choices.

Let's now explore the various customization options.

Customization Options:

a. **Line Chart Customization:**
   - **Marker Style (marker='o'):** Adding markers at data points enhances visibility and emphasizes individual values. Circular markers ('o') are placed at each data point in the line chart created using **plt.plot(x, y, marker='o')**. This makes each data point easily distinguishable.
   - **Line Style (linestyle='--'):** Choosing a specific line style, such as dashed (**--**), adds variety to the representation of the data. The line connecting the markers has a dashed line style, as specified by **plt.plot(x, y, linestyle='--')**. This introduces a visual element that makes the line stand out.

b. **Color Customization (color='b'):** Selecting a distinct color for the line improves the chart's visual appeal. In this case, **'b'** represents blue. For example, the color of the line is set to blue using **plt.plot(x, y, color='b')**. This enhances the aesthetics of the chart.

c. **Labeling (label='Line A'):** Assigning a label to the line enables the creation of a legend, providing context for the data series.

d. **Title and Axis Labels:**
   - **Title (plt.title('Customized Line Chart')):** A clear and descriptive title summarizes the chart's purpose, aiding interpretation.
   - **Axis Labels (plt.xlabel('X-axis Label'), plt.ylabel('Y-axis Label')):** Labels on the x-axis and y-axis provide information about the variables being represented.

e. **Legend Display (plt.legend()):** Including a legend is crucial when multiple data series are present. It clarifies the association between labels and corresponding lines.

Let us consider one example code to customise the Line chart.

Example Code:

```
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Create a line chart with customization
plt.plot(x, y, marker='o', linestyle='--', color='b',
label='Line A')

# Add title and labels
plt.title('Customized Line Chart')
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')

# Show a legend
plt.legend()

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

Output:

The code demonstrates how to create a customized line chart using Matplotlib. By incorporating marker styles, line styles, colours, labels, a title, and a legend, the visualization becomes not only informative but also visually appealing. These customization techniques enhance the interpretability of the chart and provide a clear narrative for the viewer.