Al-Mustaqbal University

College of Sciences

Intelligent Medical Systems Department

Advanced Plotting with Matplotlib

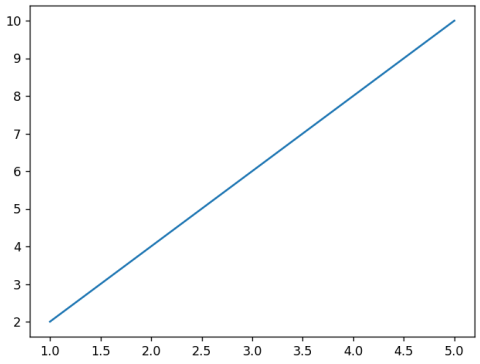
Advanced Plot Customization:

Let's break down the code provided and explore the advanced customization options used:

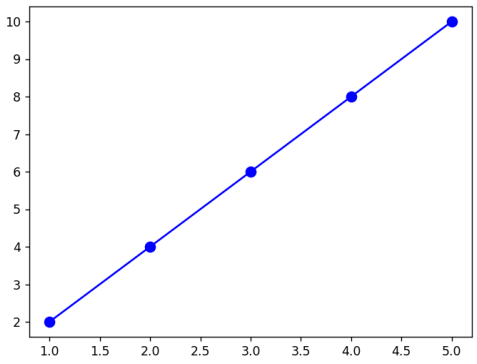
Color, Line Style, and Marker:

The `color`, `linestyle`, and `marker` parameters in the `plt.plot()` function allow us to specify the color, line style, and marker style for the plot.

```
plt.plot(x, y, color='blue', linestyle='-', marker='o', markersize=8, label='Data')
```



Before



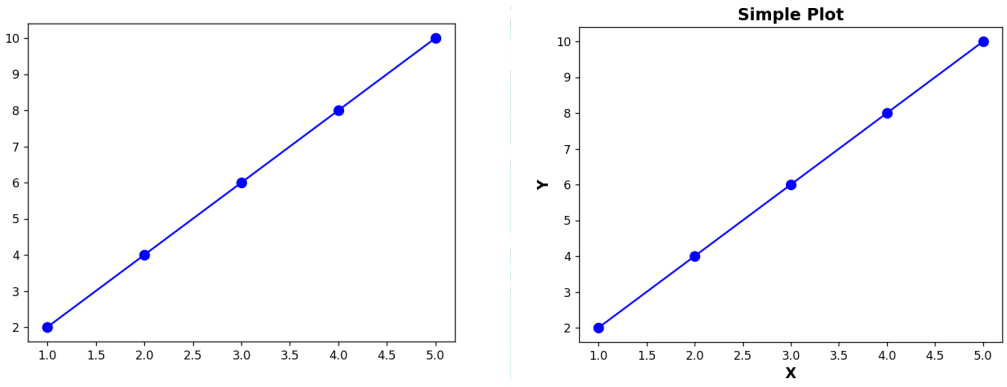
After

Labels and Title:

The `plt.xlabel()`, `plt.ylabel()`, and `plt.title()` functions are used to add labels to the x and y axes and a title to the plot.

Additional parameters like `fontsize`, `fontweight`, and color can be specified to customize the appearance of the text.

```
plt.xlabel('X', fontsize=12, fontweight='bold', color='black')
plt.ylabel('Y', fontsize=12, fontweight='bold', color='black')
plt.title('Simple Plot', fontsize=14, fontweight='bold', color='black')
```

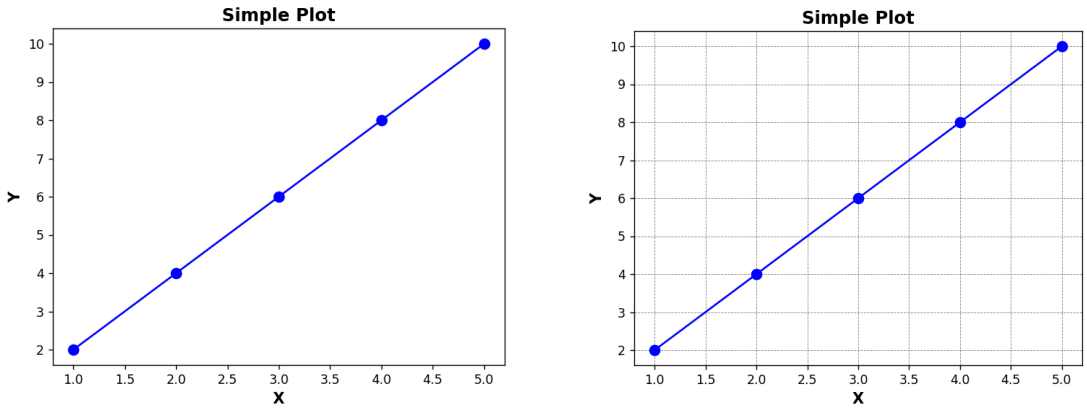


Before After

Gridlines:

The `plt.grid()` function adds gridlines to the plot, with options to specify the linestyle, linewidth, and color.

```
plt.grid(True, linestyle='--', linewidth=0.5, color='gray')
```



Before After

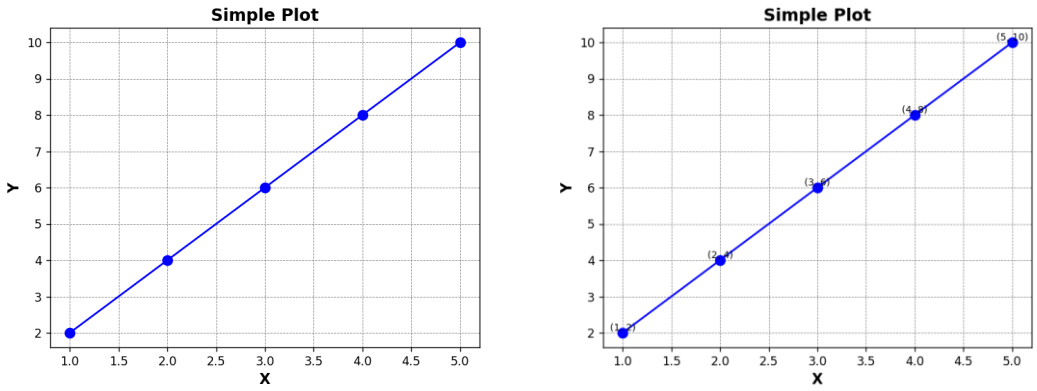
Text Annotation:

Text annotations are added to each data point using a loop and the `plt.text()` function.

Annotations include the coordinates of each point and are customized with fontsize, horizontal alignment (`ha`), and vertical alignment (`va`).

```
for i in range(len(x)):
    plt.text(x[i], y[i], f'({x[i]}, {y[i]})', fontsize=8, ha='center', va='bottom')

```



Before After

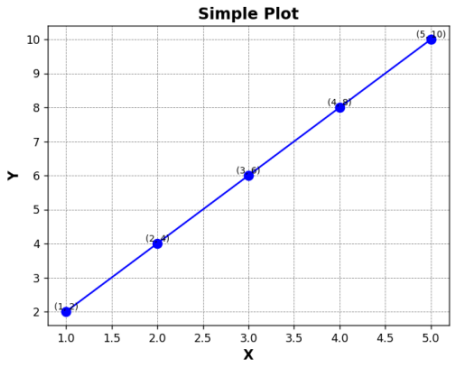
Adding Horizontal and Vertical Lines:

Horizontal and vertical lines are added using `plt.axhline()` and `plt.axvline()` functions, respectively.

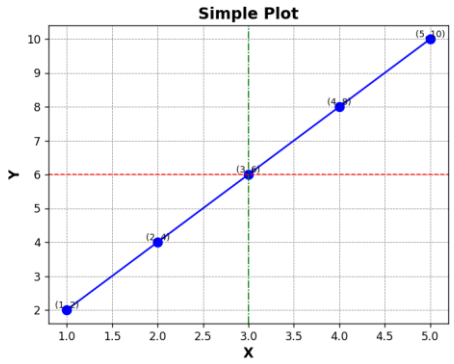
These lines provide additional reference points on the plot.

```python
# Adding a horizontal line
plt.axhline(y=6, color='red', linestyle='--', linewidth=1)

# Adding a vertical line
plt.axvline(x=3, color='green', linestyle='-.', linewidth=1)
```



Before



After

Saving the Plot:

The `plt.savefig()` function saves the plot as an image file, specifying the filename, dpi (dots per inch).

```
plt.savefig('simple_plot.png', dpi=300)
```

Conclusion:

Matplotlib offers a wide range of customization options for creating visually appealing and informative plots. By exploring the advanced plotting techniques demonstrated in this lecture, you can effectively convey data insights and enhance the presentation of your visualizations. Experiment with different parameters and features to create plots tailored to your specific needs and preferences.