Unlocking of the power or of data with with Matplotlib

Unloskimg lm tho deta whin rorre in grovnr d dcta ct power or tire and sense polyohelibι unlocok to your tartetble oones tioa yom une therthertenhg dsha wlth por Matplotlib ooanon oholestas.

1. Basic Plotting

- plt.plot(): Plot basic lines.
  Example: plt.plot(x, y)
- plt.scatter(): Plot a scatter plot.
  Example: plt.scatter(x, y)
- plt.bar(): Create a bar plot.
  Example: plt.bar(x, height)
- plt.barh(): Create a horizontal bar plot.
  Example: plt.barh(y, width)
- plt.hist(): Create a histogram.
  Example: plt.hist(data, bins=10)
- plt.boxplot(): Create a box plot.
  Example: plt.boxplot(data)
- plt.pie(): Create a pie chart.
  Example: plt.pie(sizes, labels=labels)

2. Plot Customization

- plt.title(): Add a title to the plot.
  Example: plt.title('Title of the Plot')
- plt.xlabel(), plt.ylabel(): Add labels to x and y axes.
  Example: plt.xlabel('X-axis'), plt.ylabel('Y-axis')
- plt.xlim(), plt.ylim(): Set the limits for x and y axes.
  Example: plt.xlim(0, 10), plt.ylim(0, 100)
- plt.xticks(), plt.yticks(): Set the ticks on x and y axes.
  Example: plt.xticks([0, 2, 4, 6, 8]), plt.yticks([0, 20, 40, 60])

- plt.legend(): Add a legend to the plot.
  Example: plt.legend(['Label1', 'Label2'])
- plt.grid(): Display a grid in the plot.
  Example: plt.grid(True)
- plt.tight_layout(): Automatically adjust subplot parameters for better fit.
  Example: plt.tight_layout()

---

3. Multiple Plots

- plt.subplot(): Create subplots in a grid.
  Example: plt.subplot(1, 2, 1) (1 row, 2 columns, 1st subplot)
- plt.subplots(): Create multiple subplots at once.
  Example: fig, ax = plt.subplots(2, 2) (2x2 grid of subplots)
- fig.add_subplot(): Add a subplot to an existing figure.
  Example: ax = fig.add_subplot(111) (1x1 grid, first subplot)

---

4. Saving Plots

- plt.savefig(): Save the current figure to a file.
  Example: plt.savefig('plot.png')

---

5. Plot Styling

- plt.style.use(): Apply a predefined style to the plot.
  Example: plt.style.use('ggplot')

- plt.plot(style='--'): Modify line style (e.g., dashed).
  Example: plt.plot(x, y, linestyle='--')
- plt.plot(color='red'): Modify color.
  Example: plt.plot(x, y, color='red')
- plt.plot(marker='o'): Add markers to the plot.
  Example: plt.plot(x, y, marker='o')
- plt.plot(linewidth=2): Adjust the line width.
  Example: plt.plot(x, y, linewidth=2)
- plt.plot(alpha=0.5): Set transparency (alpha).
  Example: plt.plot(x, y, alpha=0.5)

Matplotlib

Matplotlib offers a variety of plot types, each suited for specific data visualization needs.

Basic Plotting

Line Plot (`plt.plot()`)

Purpose: Displays continuous data trends over an ordered axis (e.g., time).

When to Use: Ideal for showing changes or trends over time or another continuous variable.

Example: Tracking daily temperature over a month.

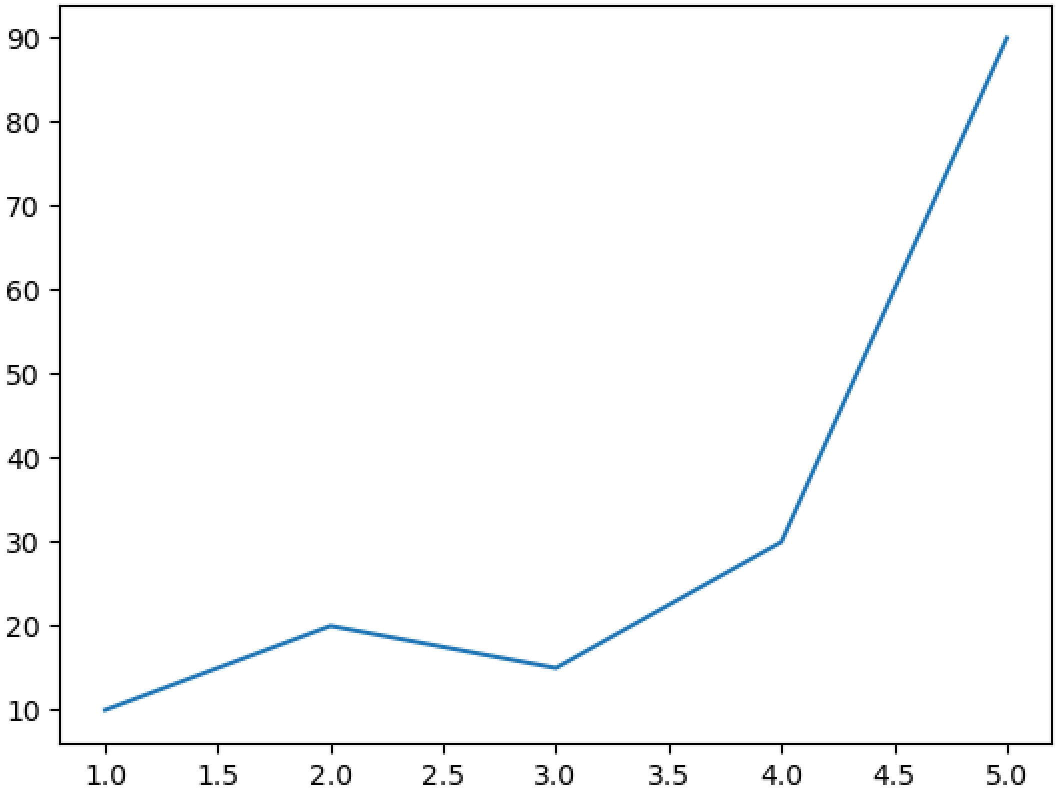
Scenario: You have temperature data for 30 days, and you want to see how it fluctuates.

Why: A line plot connects the data points, making it easy to observe trends or patterns.

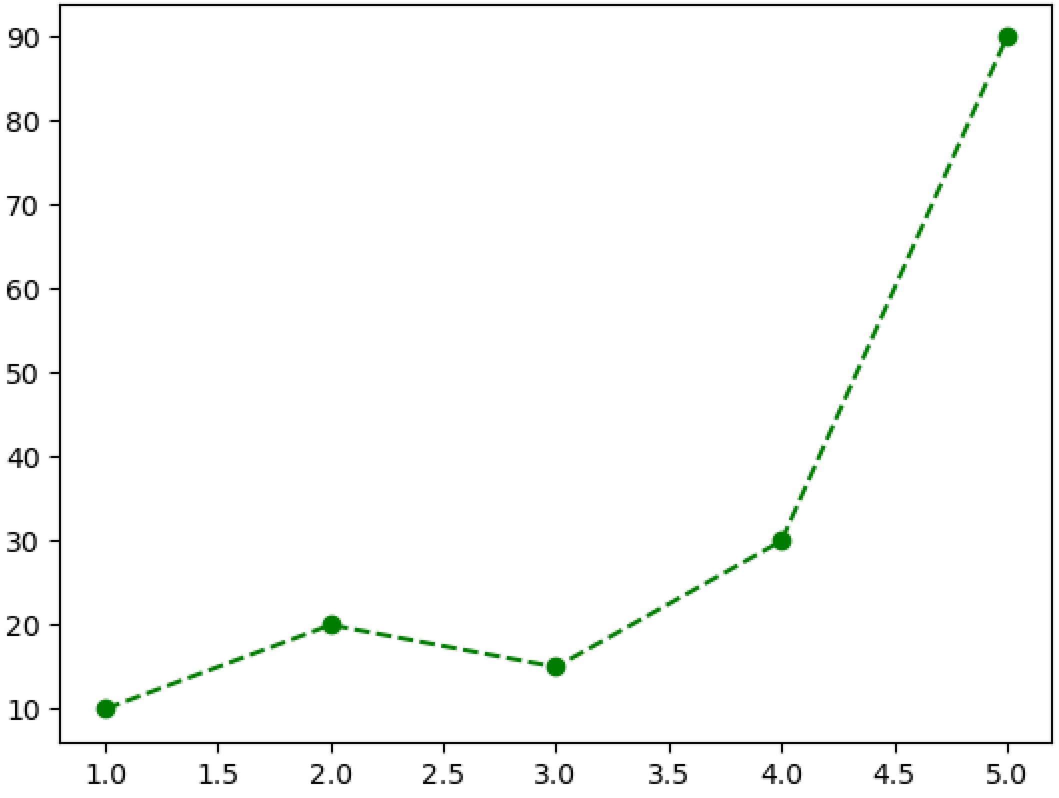
In [10]:

```python
#syntax: plt.plot(x, y, [fmt], **kwargs)
#example-1
import matplotlib.pyplot as plt

x=[1,2,3,4,5]
y=[10,20,15,30,90]
plt.plot(x,y)
plt.show()
```

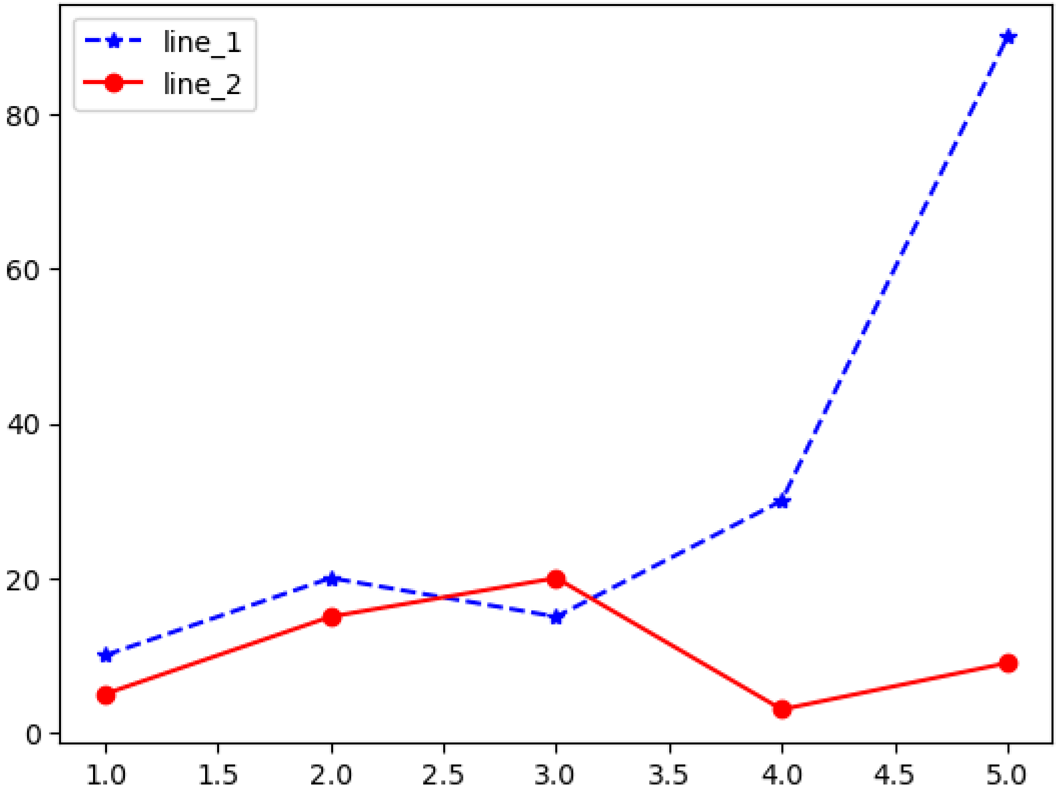


In [22]:
```
#example-2
plt.plot(x,y, "go--")
plt.show()
```



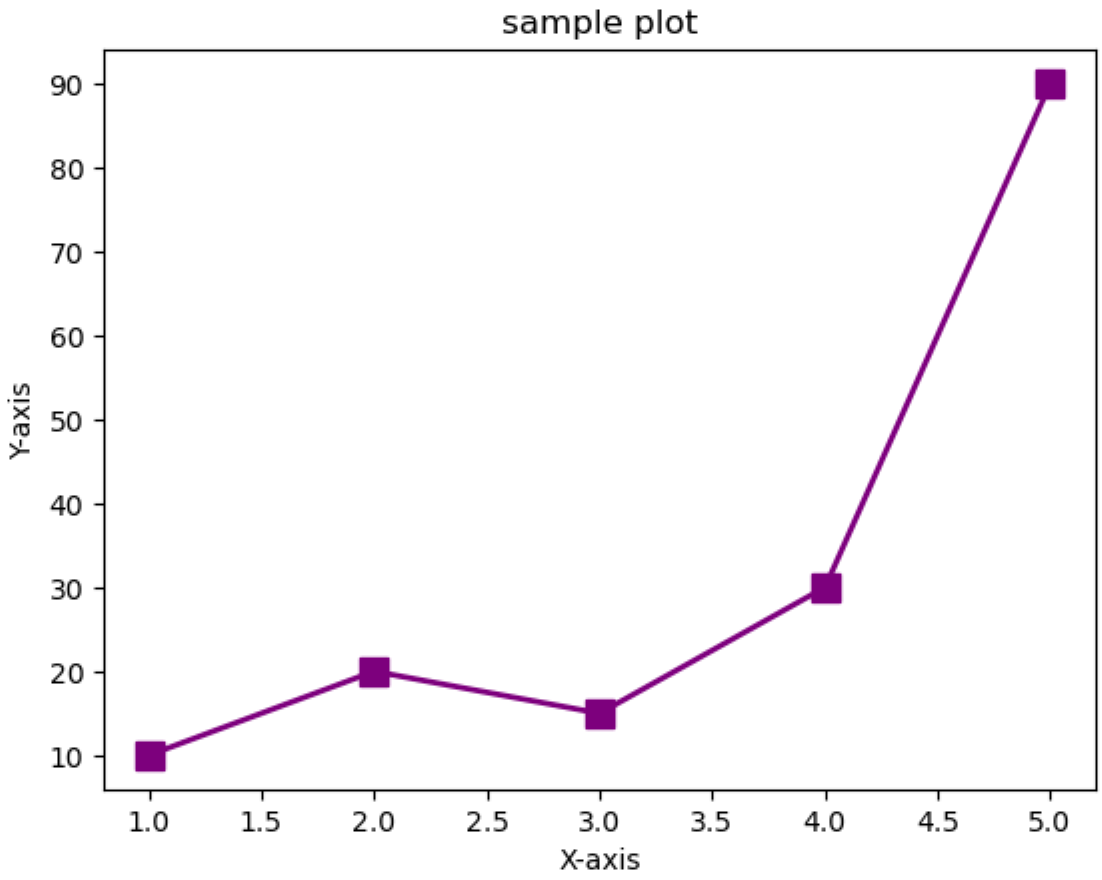
In [16]:

```
#example-3
y2=[5,15,20,3,9]
plt.plot(x,y, "b*--", label="line_1")
plt.plot(x,y2, "ro-", label="line_2")
plt.legend()
plt.show()
```



In [21]:

```
#example-4
plt.plot(x,y, color="purple", linestyle="-", linewidth=2, marker="s", marker
plt.title("sample plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



Scatter Plot (`plt.scatter()`)

Purpose: Shows the relationship between two variables using individual points.

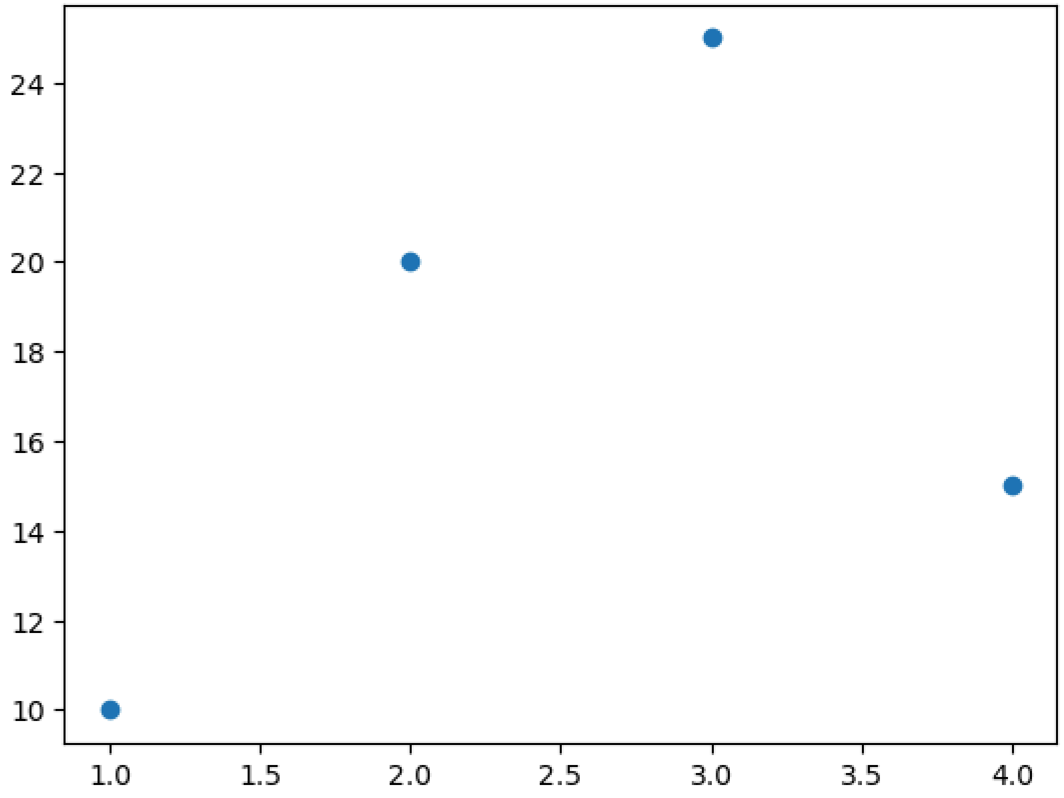
When to Use: Best for visualizing correlations or distributions, especially when data isn't necessarily continuous.

Example: Comparing students' test scores vs. hours studied.

Scenario: You have data on 50 students with their study hours and test scores.

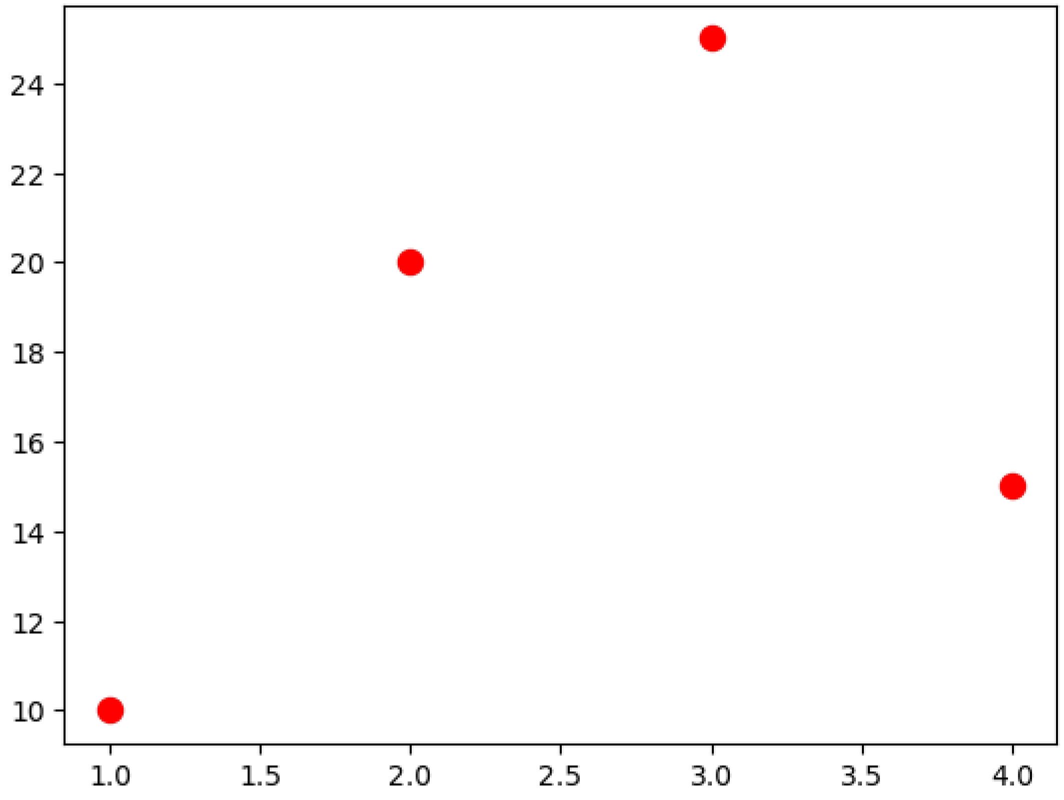
Why: Scatter plots reveal if there's a correlation (e.g., more study hours = higher scores) without assuming a continuous trend.

In [23]:
```
#syntax: plt.scatter(x, y, s=None, c=None, marker=None, **kwargs)
# Example 1: Basic Scatter Plot
x = [1, 2, 3, 4]
y = [10, 20, 25, 15]
plt.scatter(x,y)
plt.show()
```



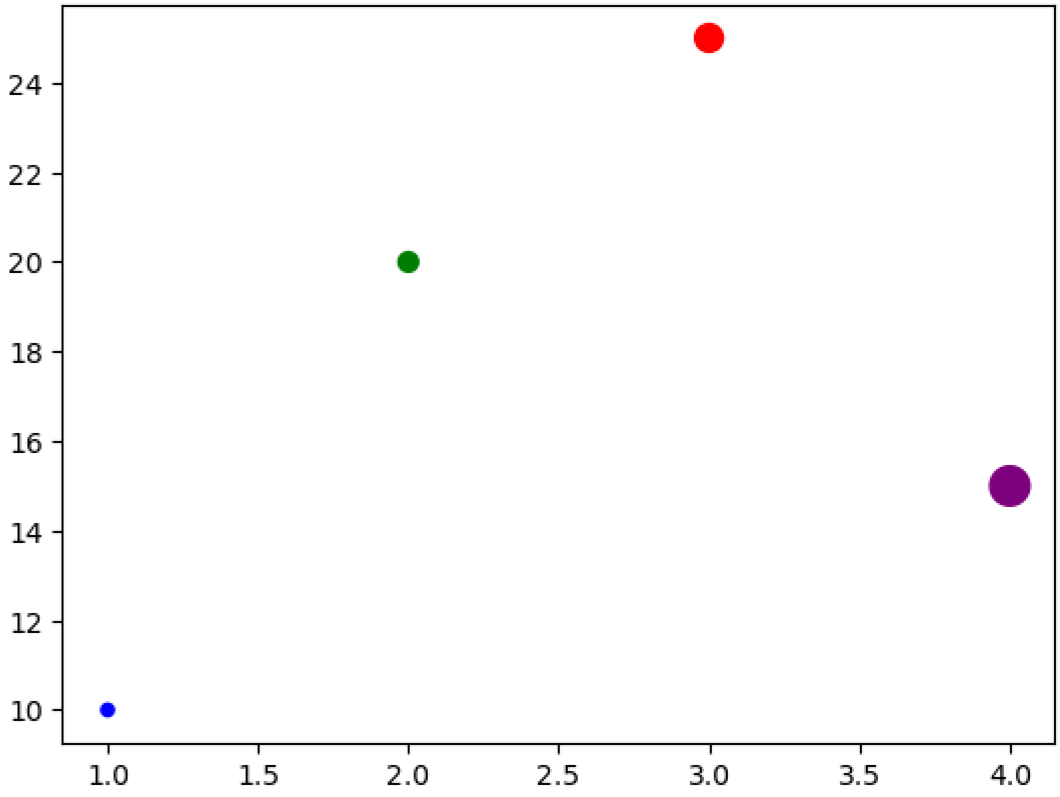
```
In [35]: #Example 2: Custom size and color
         plt.scatter(x,y, color='r',s=75)
```

```
Out[35]: <matplotlib.collections.PathCollection at 0x2032eec11c0>
```



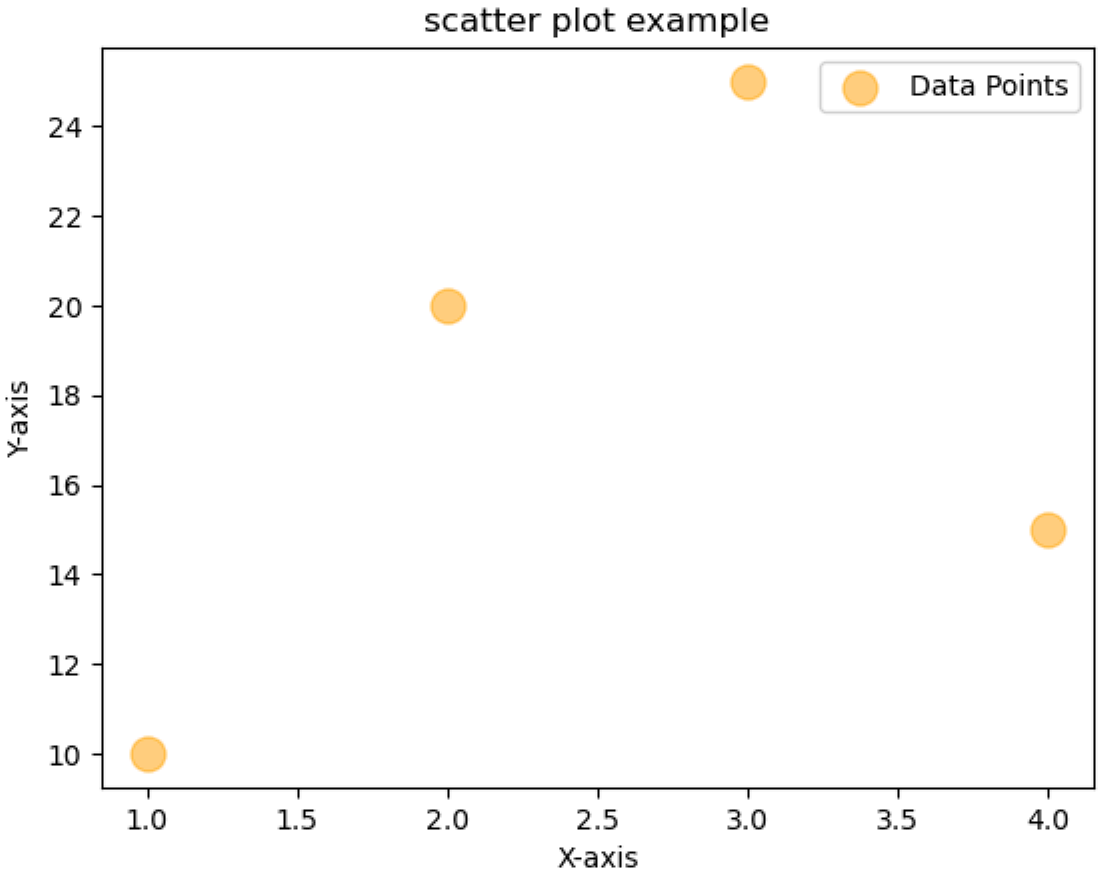
In [36]:

```
#Example 3: Variable size and color
sizes = [20, 50, 100, 200]
colors = ['blue', 'green', 'red', 'purple']
plt.scatter(x,y, s=sizes, c=colors)
plt.show()
```



In [45]:

```
#Example 4: With Labels and Transparency
plt.scatter(x, y, s=150, c='orange', alpha=0.5, label="Data Points") #alpha
plt.title("scatter plot example")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()
plt.show()
```



Bar Plot (`plt.bar()`)

Purpose: Compares discrete categories or groups with numerical values.

When to Use: Useful for categorical data where you want to show quantities or counts.

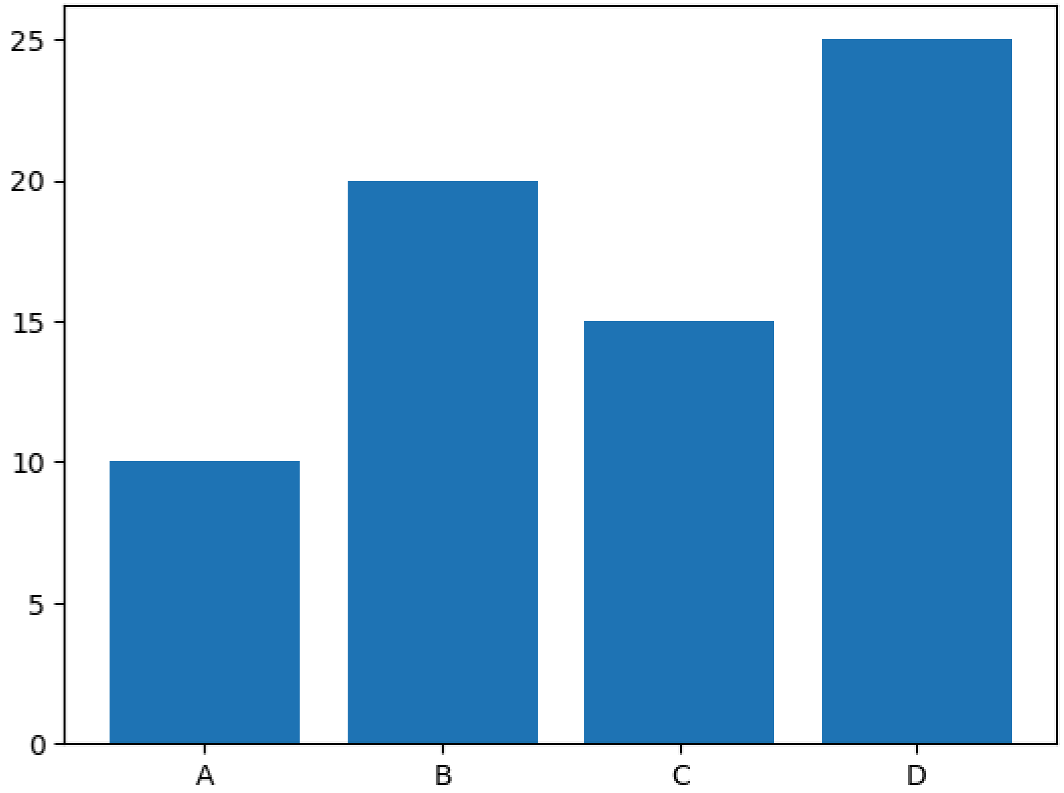
Example: Sales of different products in a store.

Scenario: You have sales data for 5 products (e.g., apples, bananas, etc.).

Why: Bar plots make it easy to compare the height of bars, highlighting differences across categories.

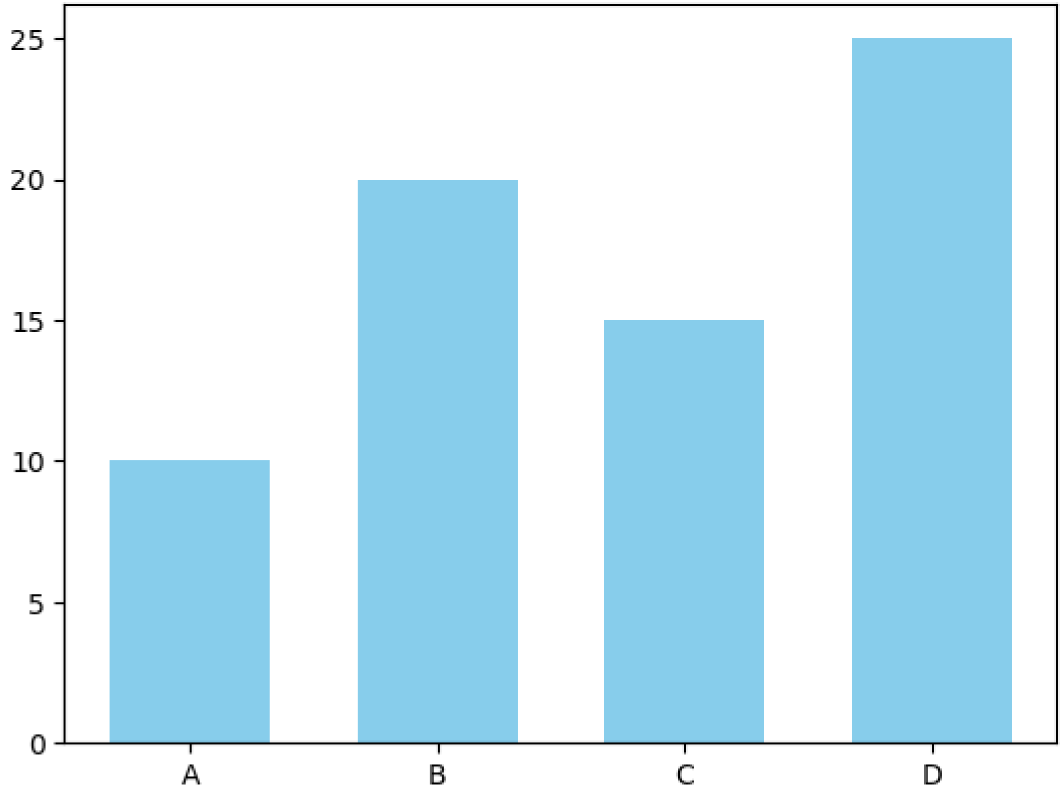
In [46]:

```
#plt.bar(x, height,, width=0.8, **kwargs)
# Example 1: Basis Bar Plot
x = ['A', 'B', 'C', 'D']
heights = [10, 20, 15, 25]
plt.bar(x, heights)
plt.show()
```



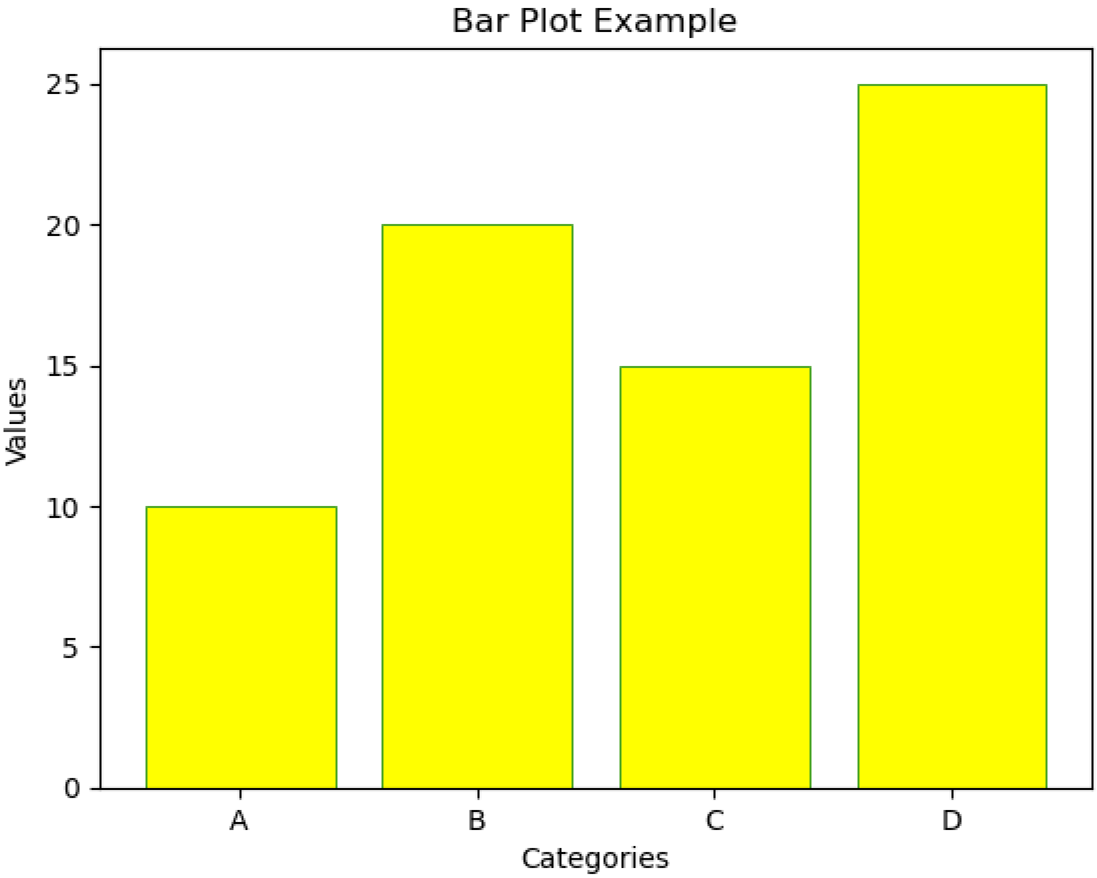
In [54]:

```
#Example 2: Custom color and width
plt.bar(x, heights, color='skyblue', width=0.65)
plt.show()
```



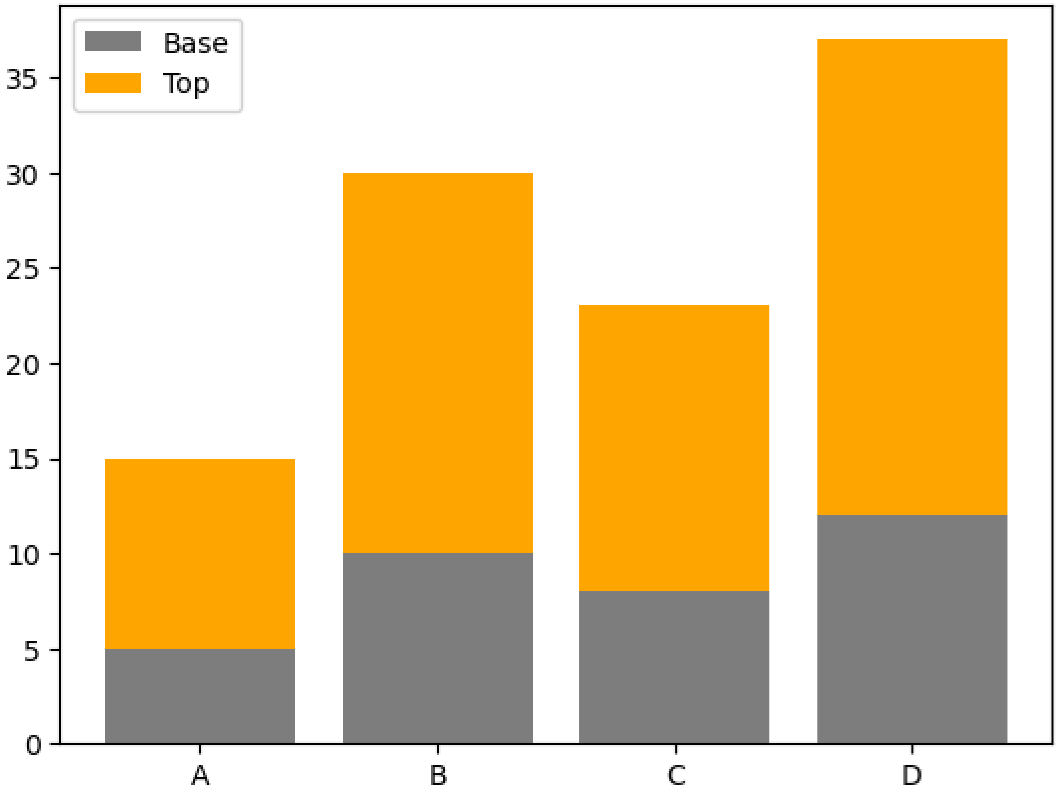
In [58]:

```
#Example 3: With Edge color and labels
plt.bar(x, heights, color='yellow', edgecolor='green', linewidth=0.5)
plt.title("Bar Plot Example")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.show()
```



In [61]:

```
#Example 4: Stacked Bars
sample_2=[5,10,8,12]
plt.bar(x, sample_2, color='gray', label='Base')
plt.bar(x, heights, bottom=sample_2, color='orange', label='Top')
plt.legend()
plt.show()
```



Horizontal Bar Plot (`plt.barh()`)

Purpose: Similar to a bar plot but with horizontal bars.

When to Use: When category names are long or you want a different visual orientation.

Example: Ranking top 10 countries by population.

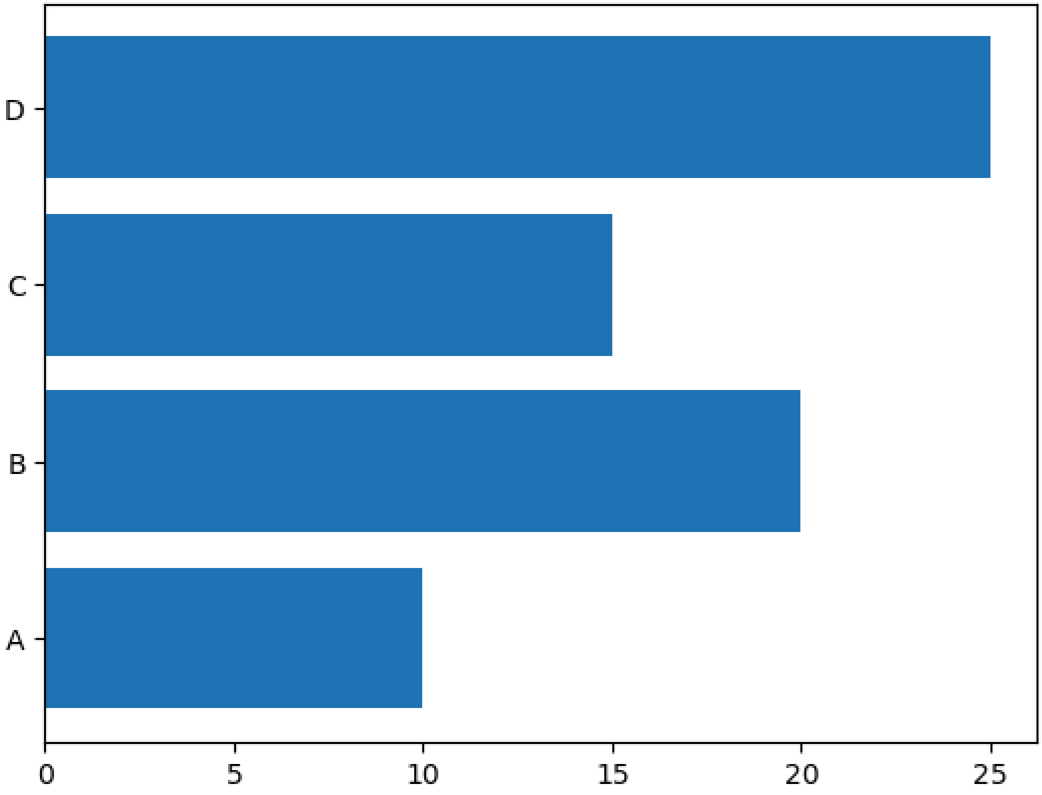
Scenario: You have population data for countries with long names.

Why: Horizontal bars accommodate longer labels and can be more readable.

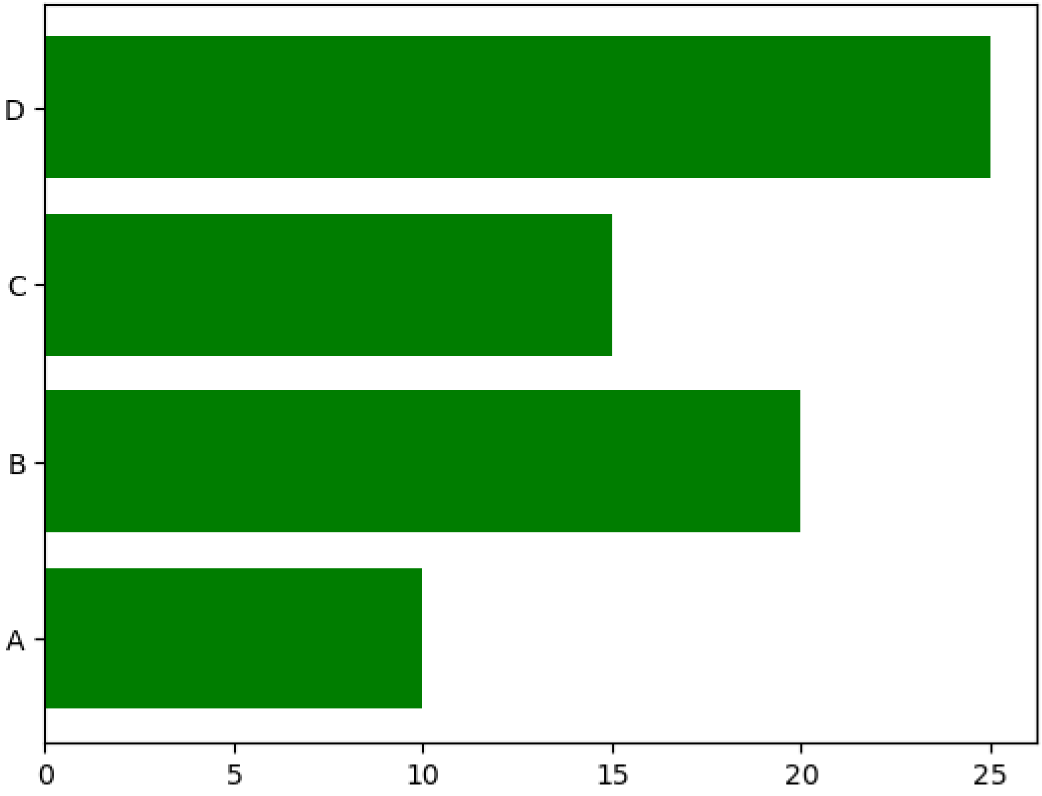
In [63]:

```
#syntax: plt.barh(y, width, height=0.8, **kwargs)

#example 1: Base Horizontal Bar plot
y = ['A', 'B', 'C', 'D']
widths = [10, 20, 15, 25]
plt.barh(y, widths)
plt.show()
```

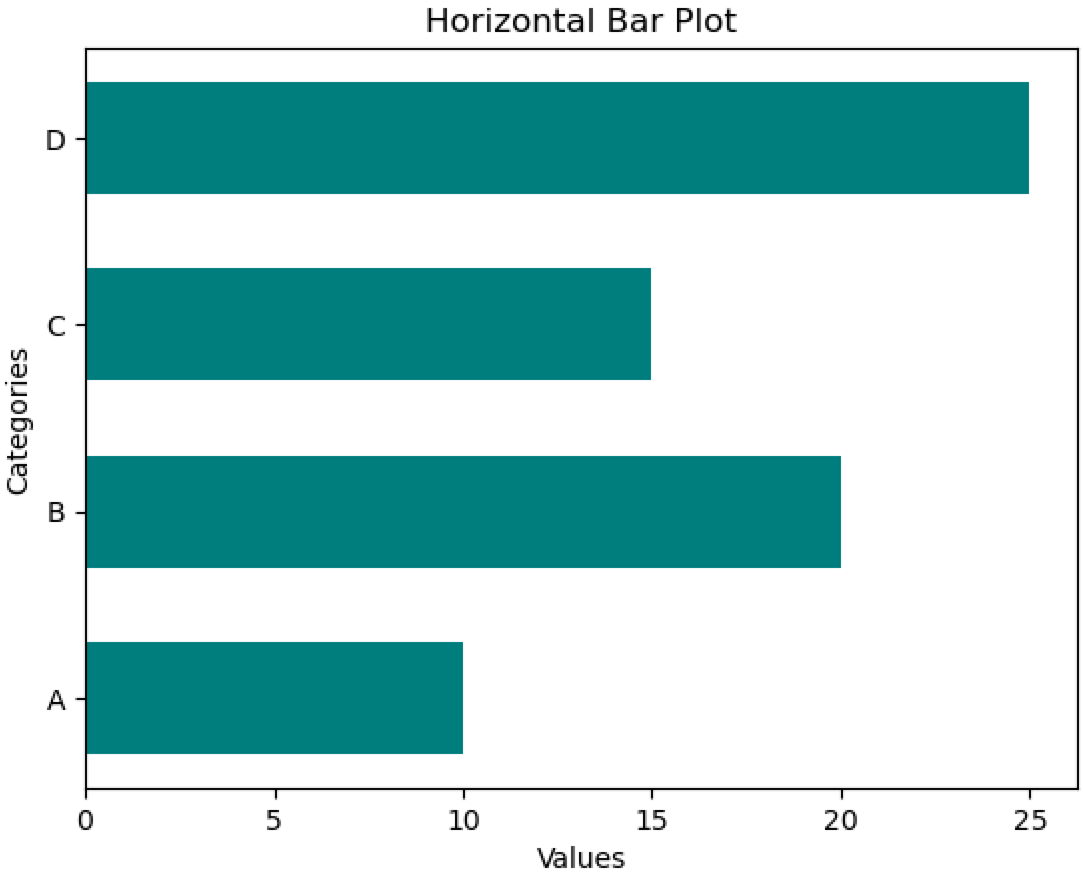


```
In [66]: #Example 2: Custom Colors
         plt.barh(y, widths, color='green')
         plt.show()
```



In [69]:

```python
#Example 3: with Labels
plt.barh(y, widths, color="teal", height=0.6)
plt.title("Horizontal Bar Plot")
plt.xlabel("Values")
plt.ylabel("Categories")
plt.show()
```



Histogram (`plt.hist()`)

Purpose: Displays the distribution of a single numerical variable.

When to Use: When you want to see the frequency or count of values within ranges (bins).

Example: Distribution of exam scores in a class.

Scenario: You have scores of 100 students and want to see how they're spread (e.g., most scored 70-80?).

Why: Histograms show the shape of the data (e.g., normal, skewed).

In [79]:

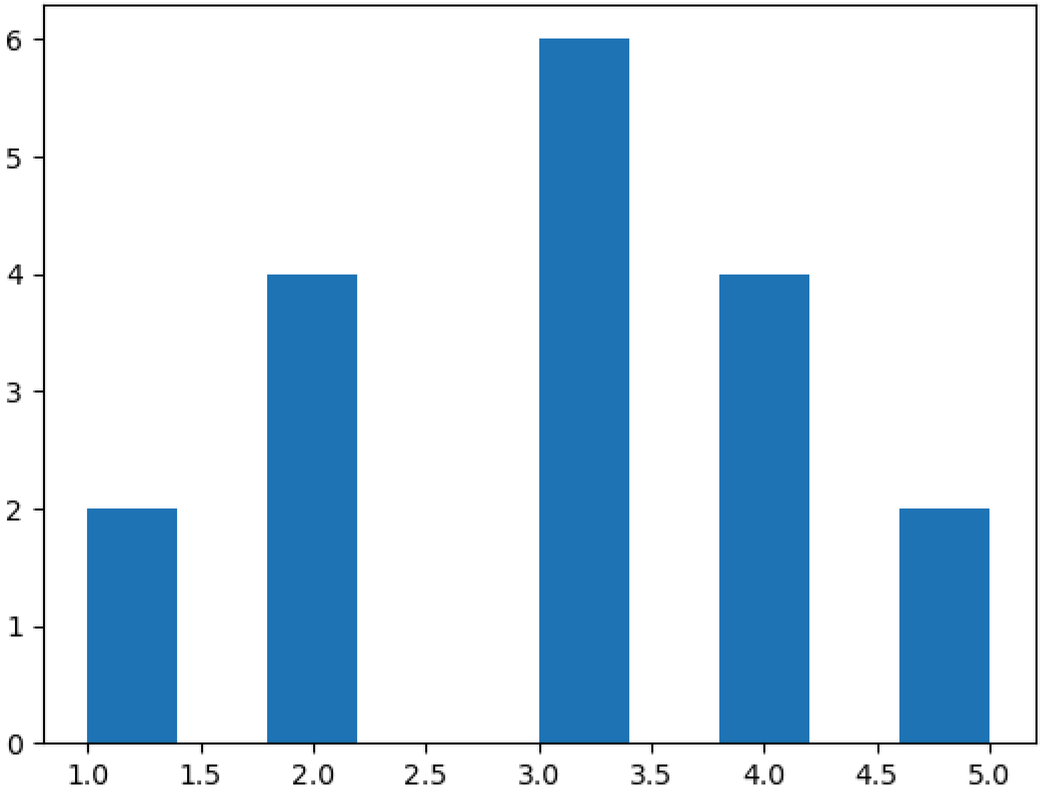
```python
data = [1, 2, 2, 3, 3, 3, 4, 4, 5]
import numpy as np
data=np.array(data)
```

```
data=np.tile(data,2)
print(data)
```

```
[1 2 2 3 3 3 4 4 5 1 2 2 3 3 3 4 4 5]
```

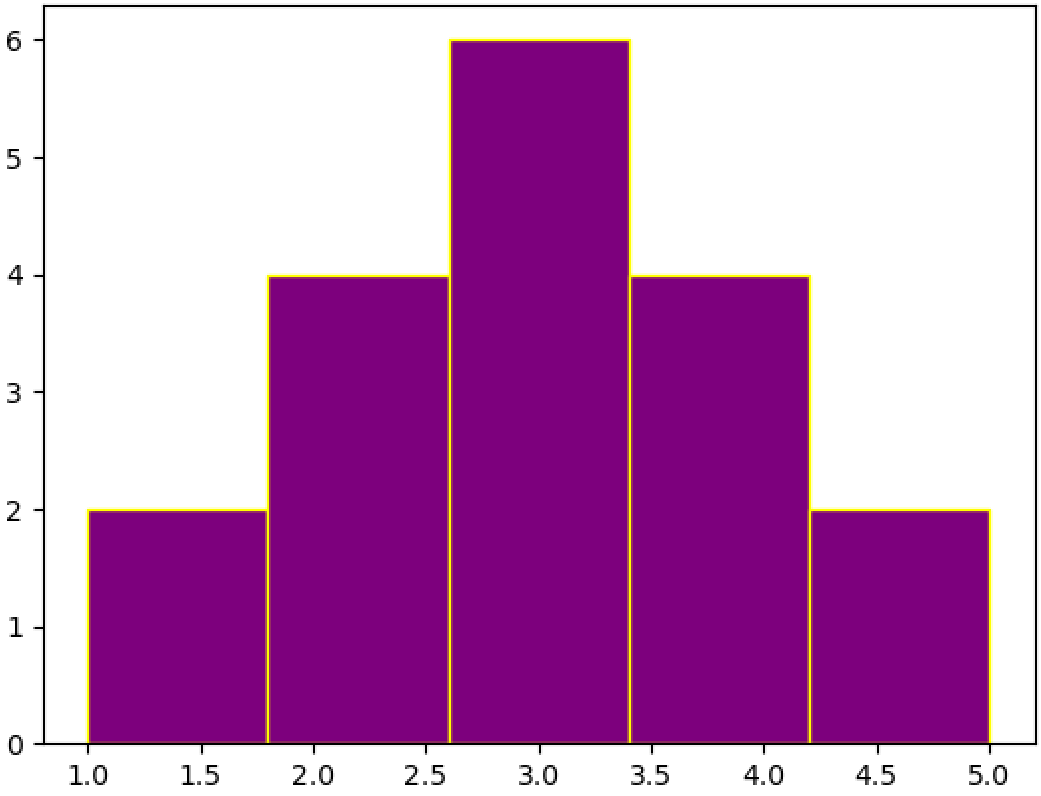
In [80]:

```
#syntax: plt.hist(x, bins=None, **kwargs)
#example 1: Basix Histogram
plt.hist(data)
plt.show()
```



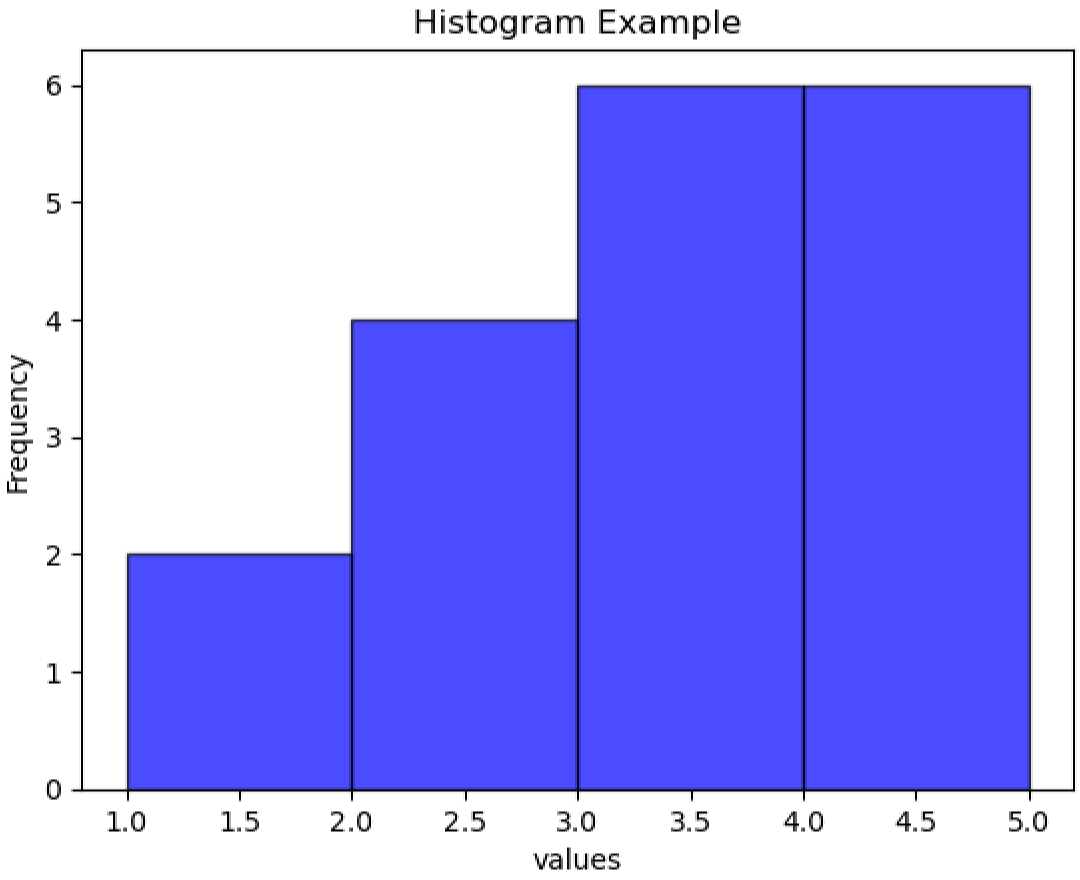
In [83]:

```
#example 2: Custom Bins and color
plt.hist(data, bins=5, color='purple', edgecolor='yellow')
plt.show()
```



In [84]:

```
#example 3: with transparency and labels
plt.hist(data, bins=4, color='blue', alpha=0.7, edgecolor='black')
plt.title("Histogram Example")
plt.xlabel("values")
plt.ylabel("Frequency")
plt.show()
```



Box Plot (`plt.boxplot()`)

Purpose: Summarizes data distribution (median, quartiles, outliers).

When to Use: When you need to compare distributions across groups or detect outliers.

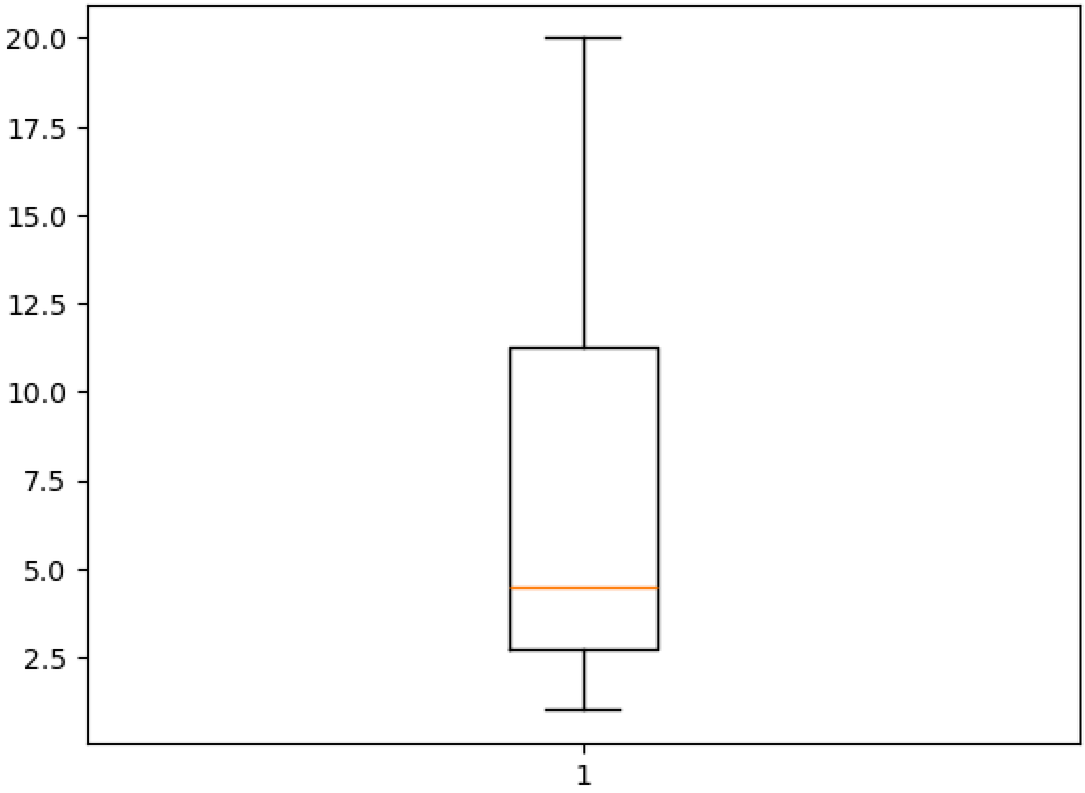
Example: Comparing salaries across different job roles.

Scenario: You have salary data for engineers, managers, and analysts.

Why: Box plots quickly show the spread, central tendency, and any extreme values.

In [10]:

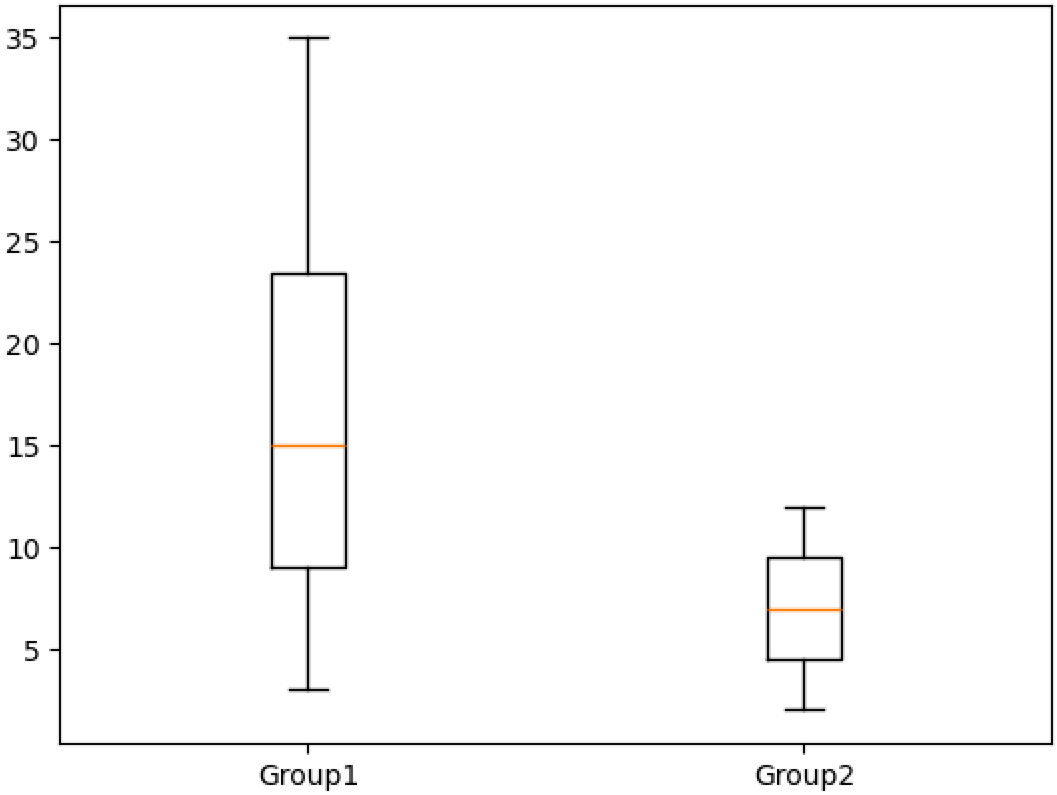
```
#plt.boxplot(x, vert=True, **kwargs)
#example 1:Basis Box Plot
import matplotlib.pyplot as plt
data = [1, 2, 3, 4, 5, 10, 15, 20]
plt.boxplot(data)
plt.show()
```



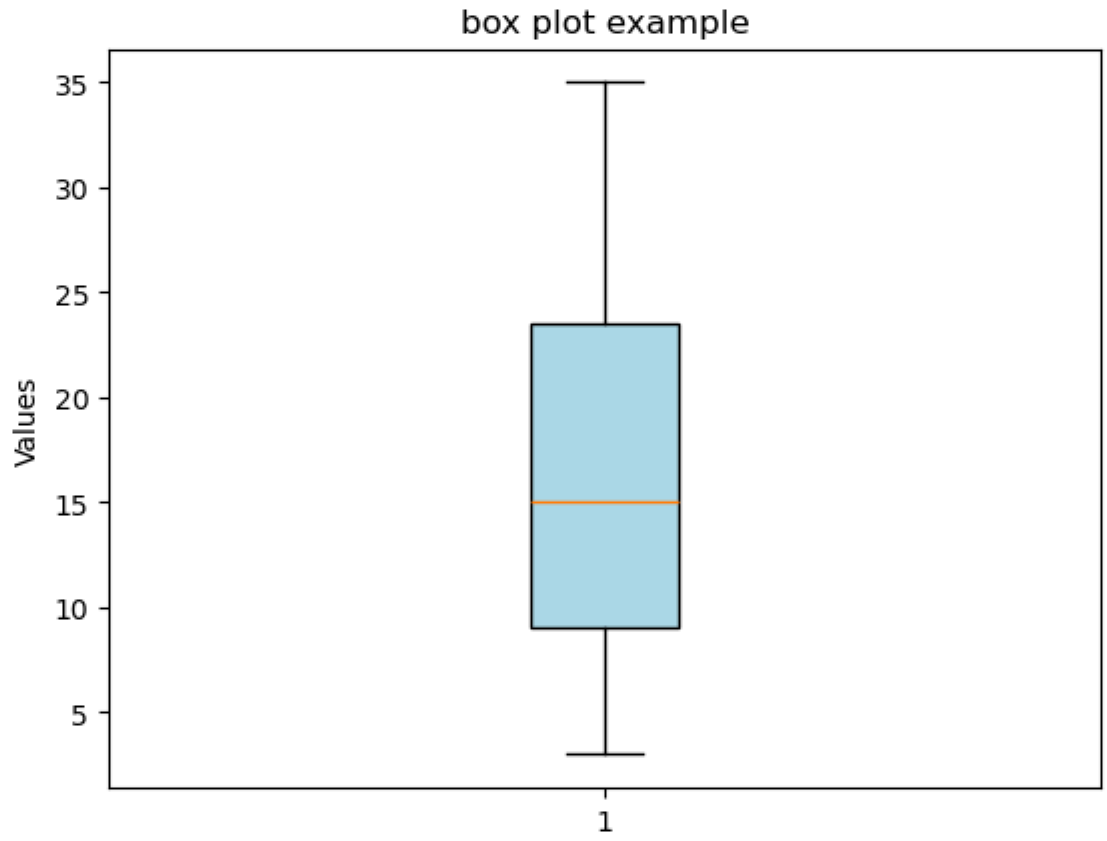
In [13]:

```
#Example 2: Multiple Box Plots
data2 = [2, 4, 6, 8, 10, 12]
plt.boxplot([data,data2], labels=['Group1','Group2'])
plt.show()
```

```
C:\Users\LIB\AppData\Local\Temp\ipykernel_13248\432483055.py:3: MatplotlibDe
precationWarning: The 'labels' parameter of boxplot() has been renamed 'tick
_labels' since Matplotlib 3.9; support for the old name will be dropped in
3.11.
  plt.boxplot([data,data2], labels=['Group1','Group2'])
```



```
In [14]: #Example 3: Custom Styling
         plt.boxplot(data, patch_artist=True, boxprops=dict(facecolor='lightblue'))
         plt.title("box plot example")
         plt.ylabel("Values")
         plt.show()
```



Pie Chart (`plt.pie()`)

Purpose: Shows proportions or percentages of a whole.

When to Use: When you have a small number of categories and want to visualize their relative contributions.

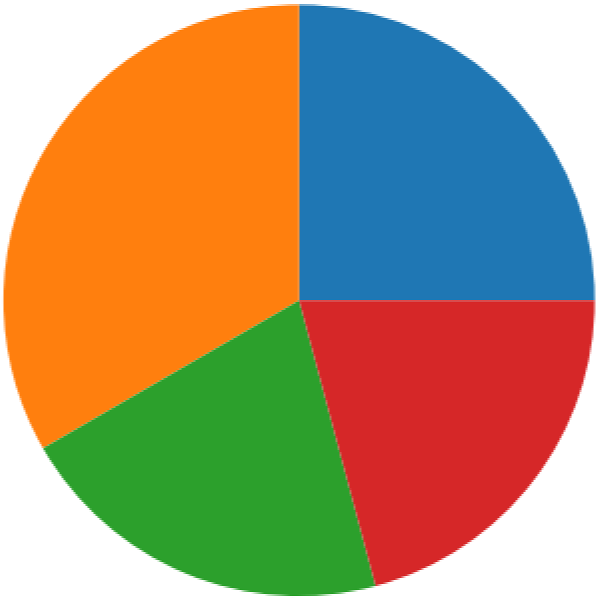
Example: Market share of smartphone brands.

Scenario: You have market share data for 5 brands (e.g., Apple 40%, Samsung 30%, etc.).

Why: Pie charts emphasize how each category contributes to 100%.

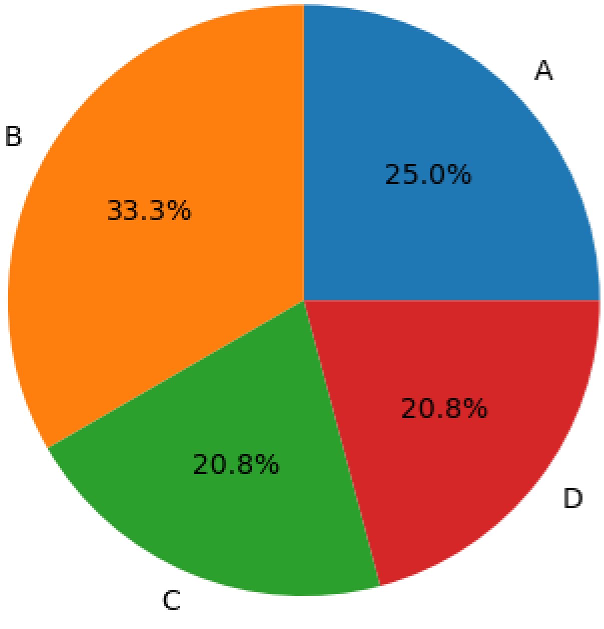
In [16]:

```python
#syntax: plt.pie(x, labels=None, autopct=None, **kwargs)
#Example 1: Basic Pie chart
sizes = [30,40,25,25]
plt.pie(sizes)
plt.show()
```



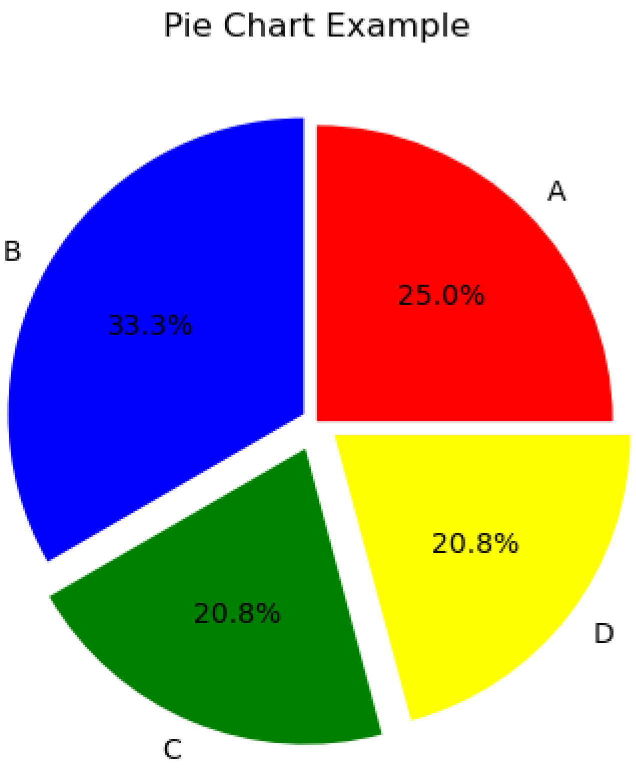
In [23]:

```
labels = ['A', 'B', 'C', 'D']
plt.pie(sizes, labels=labels, autopct="%1.1f%%")
plt.show()
```



In [37]:

```
# Example 3: Exploded Slice and Colors
explode = [0, 0.05, 0.1, 0.075]  # Explode the second slice
plt.pie(sizes, labels=labels, autopct='%1.1f%%', explode=explode, colors=['r
plt.title('Pie Chart Example')
plt.show()
```



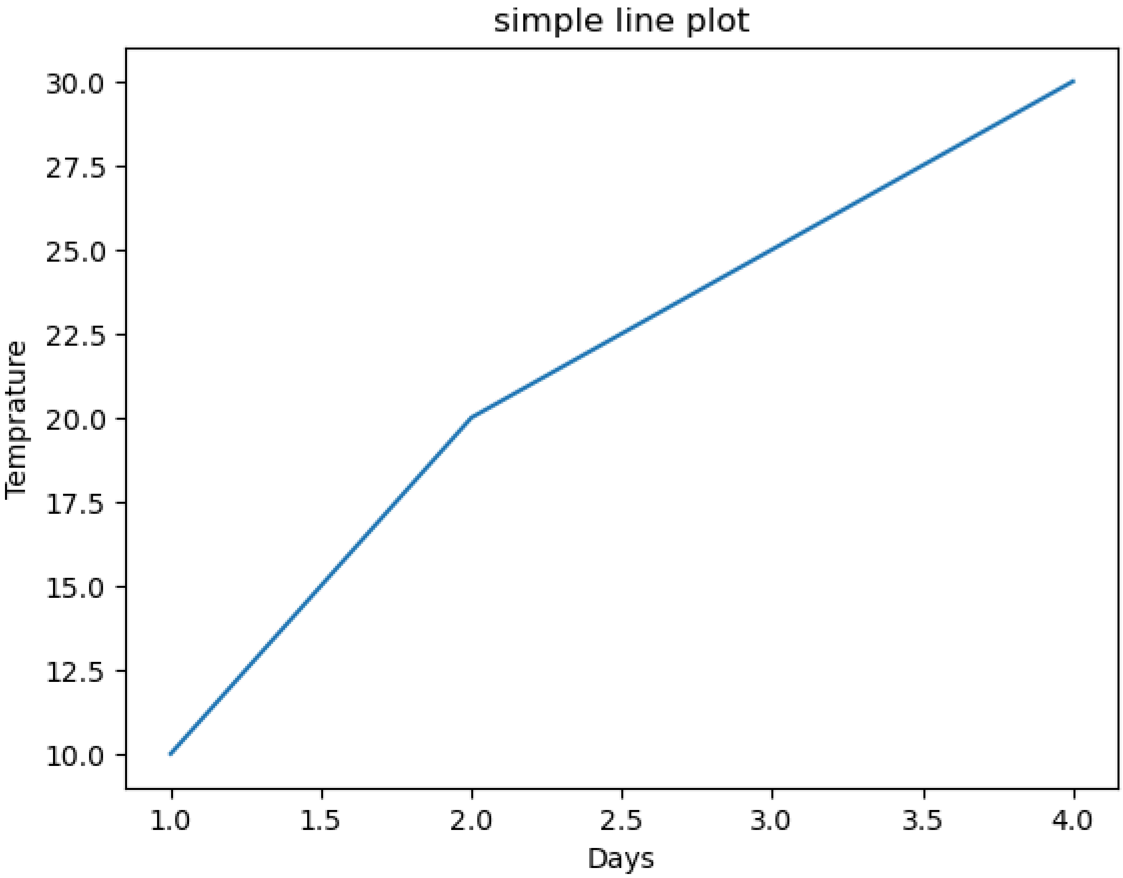
Plot Customization

In [38]:

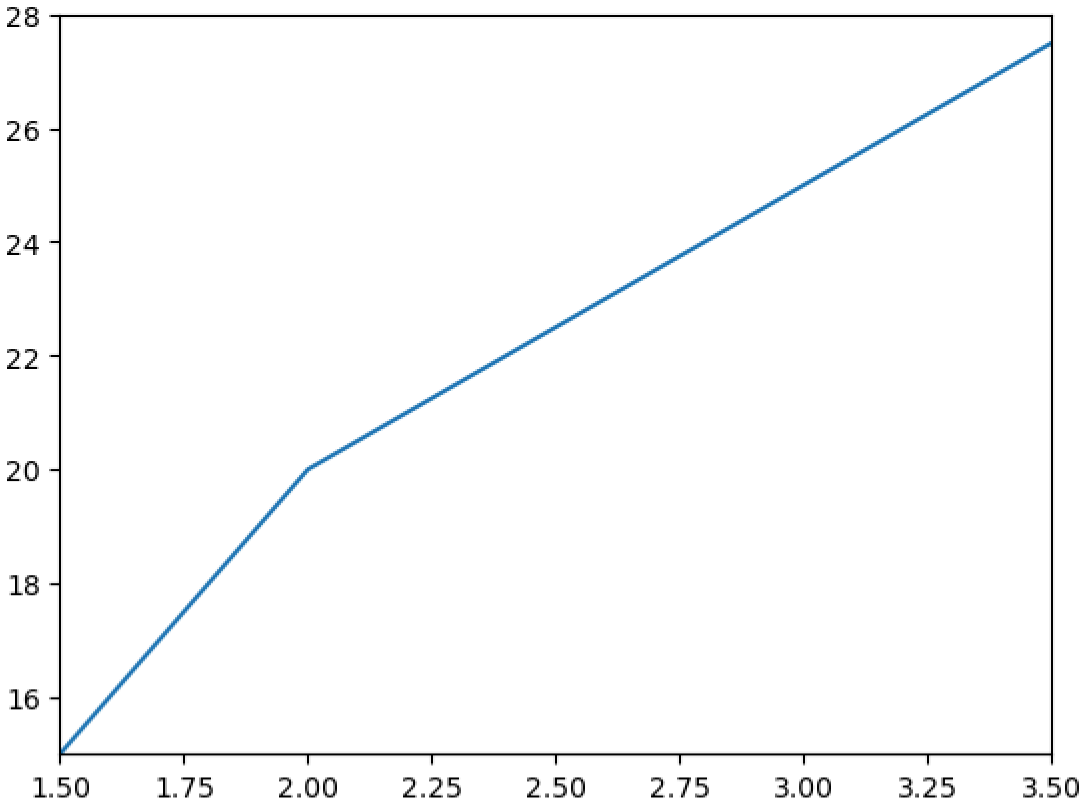
```
import matplotlib.pyplot as plt

# Example 1: Basic Title
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

plt.plot(x,y)
plt.title("simple line plot")
plt.xlabel("Days")
plt.ylabel("Temprature")
plt.show()
```

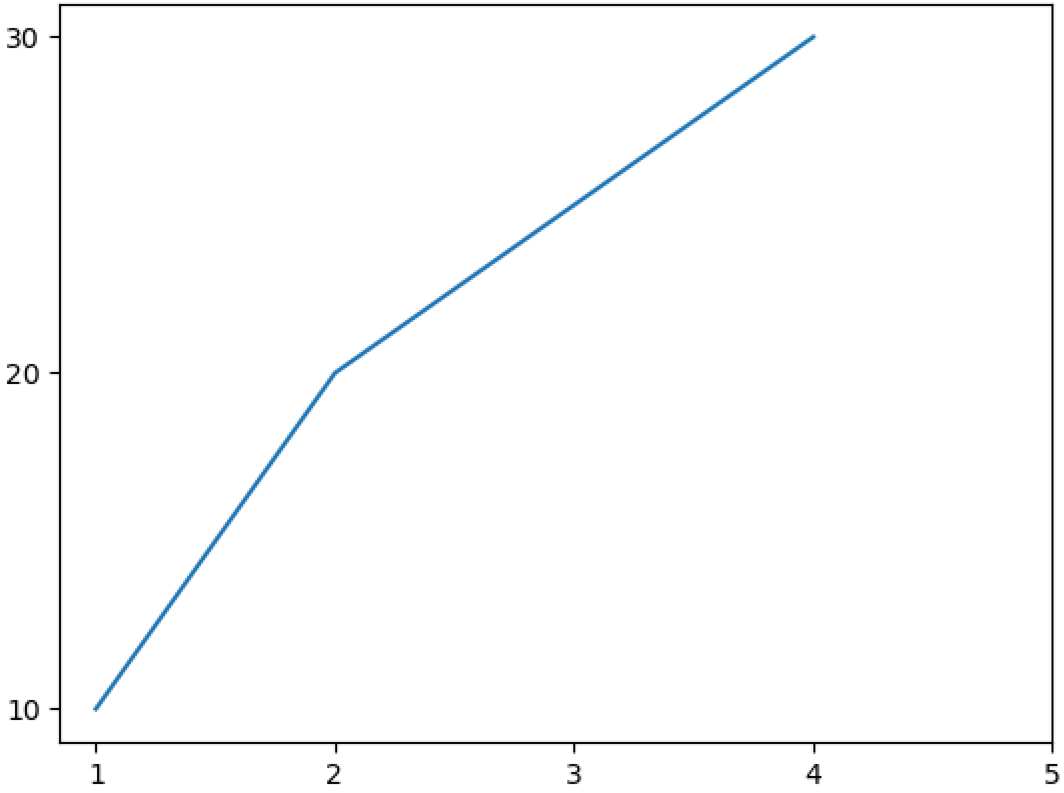


In [41]:
```
plt.plot(x,y)
plt.xlim(1.5,3.5)
plt.ylim(15,28)
plt.show()
```



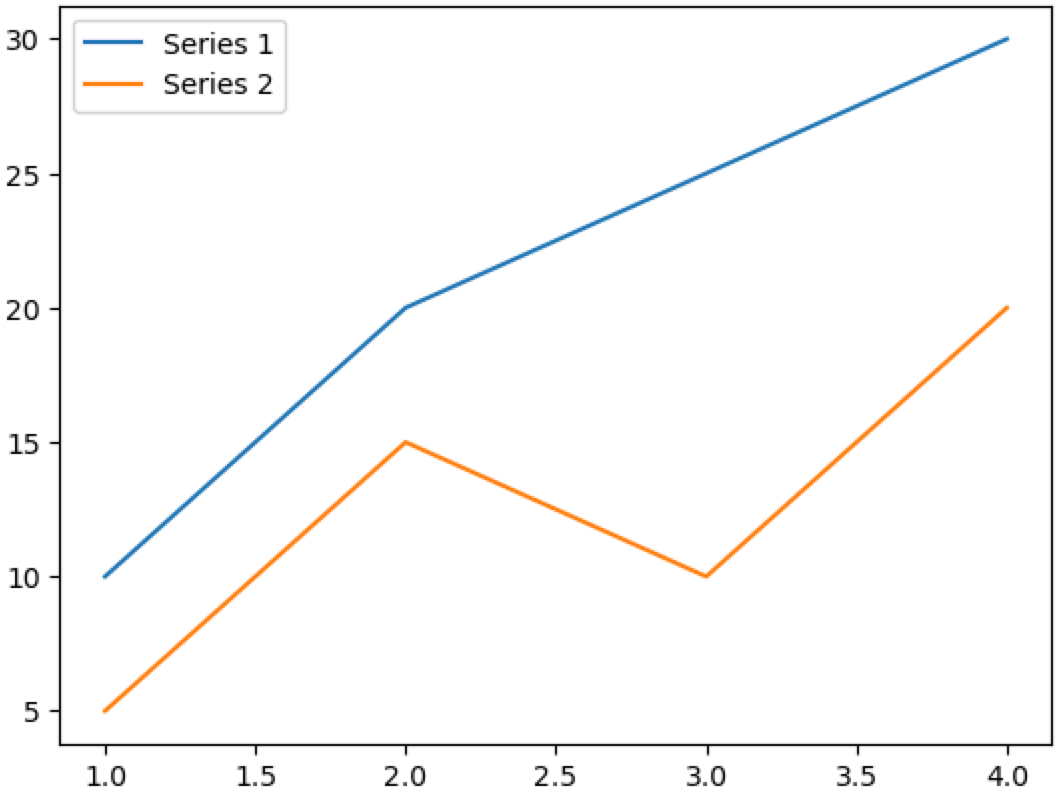
In [43]:

```
plt.plot(x,y)
plt.xticks([1,2,3,4,5])
plt.yticks([10,20,30])
plt.show()
```



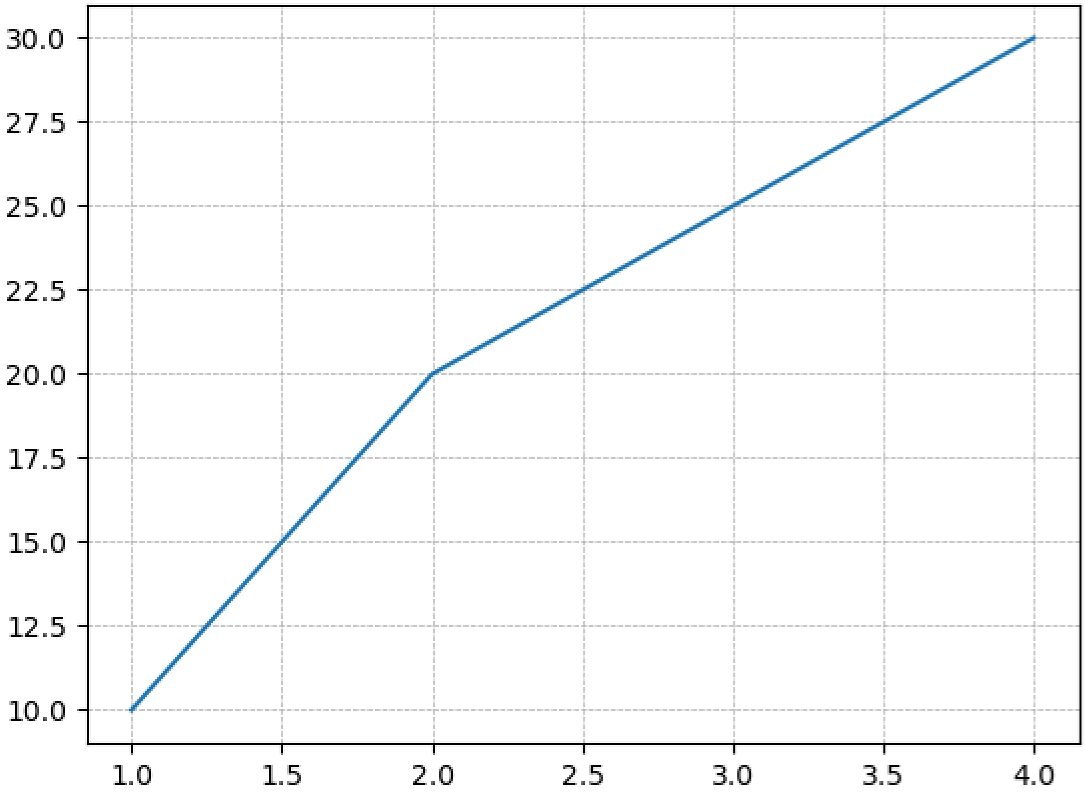
In [45]:

```
plt.plot(x, y, label='Series 1')
plt.plot(x, [5, 15, 10, 20], label='Series 2')
plt.legend()
plt.show()
```



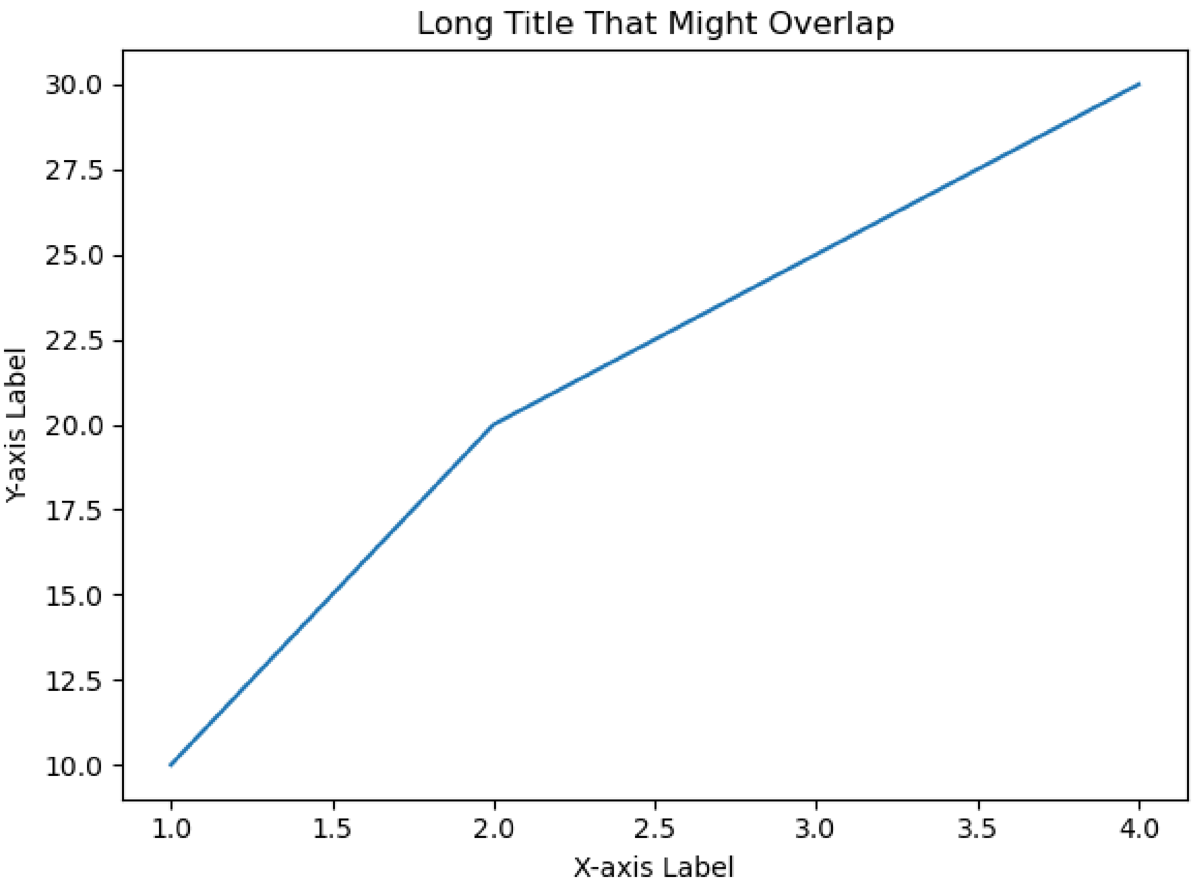
In [54]:

```
plt.plot(x,y)
plt.grid(True,linestyle='--', linewidth=0.5)
plt.show()
```



In [56]:

```
plt.plot(x, y)
plt.title('Long Title That Might Overlap')
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
plt.tight_layout() #Ensures labels, titles, and ticks don’t overlap, especia
plt.show()
```



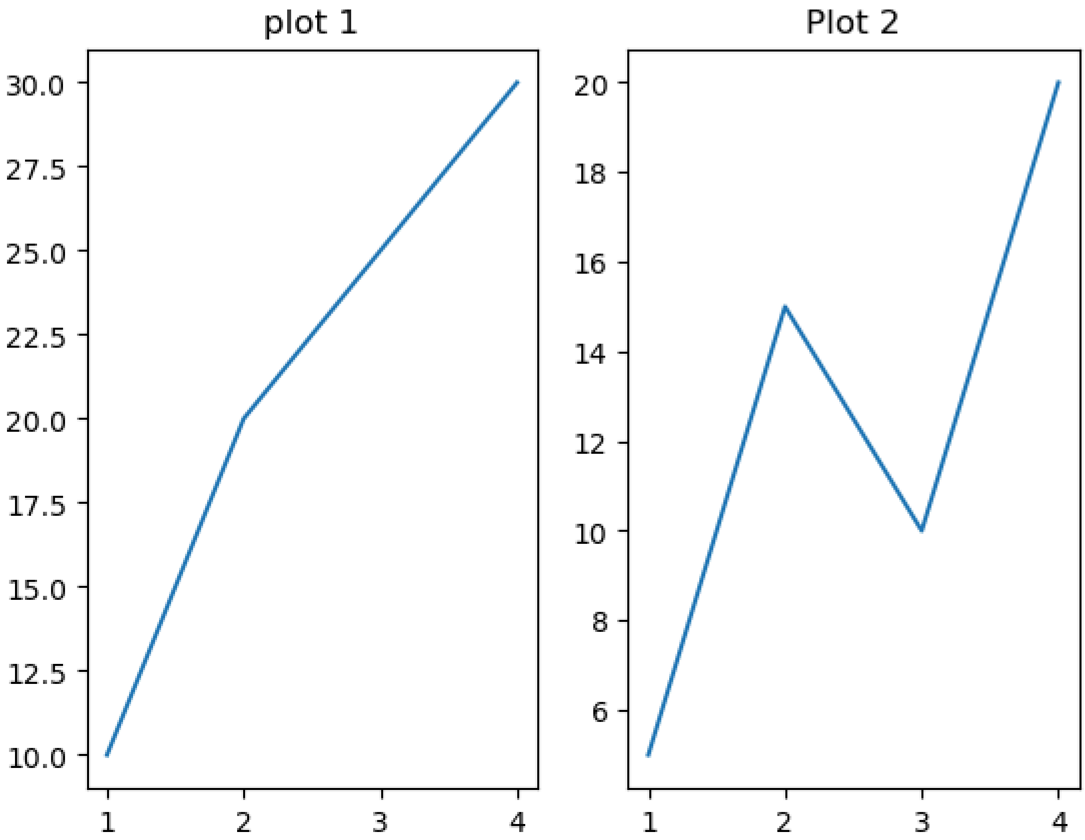
Multiple Plots

In [58]:

```
import matplotlib.pyplot as plt

# Example 1: Basic 1x2 Subplot Layout
x = [1, 2, 3, 4]
y1 = [10, 20, 25, 30]
y2 = [5, 15, 10, 20]
plt.subplot(1,2,1) # 1 row, 2 columns, 1st subplot
plt.plot(x,y)
plt.title("plot 1")

plt.subplot(1, 2, 2)  # 1 row, 2 columns, 2nd subplot
plt.plot(x, y2)
plt.title('Plot 2')
plt.show()
```

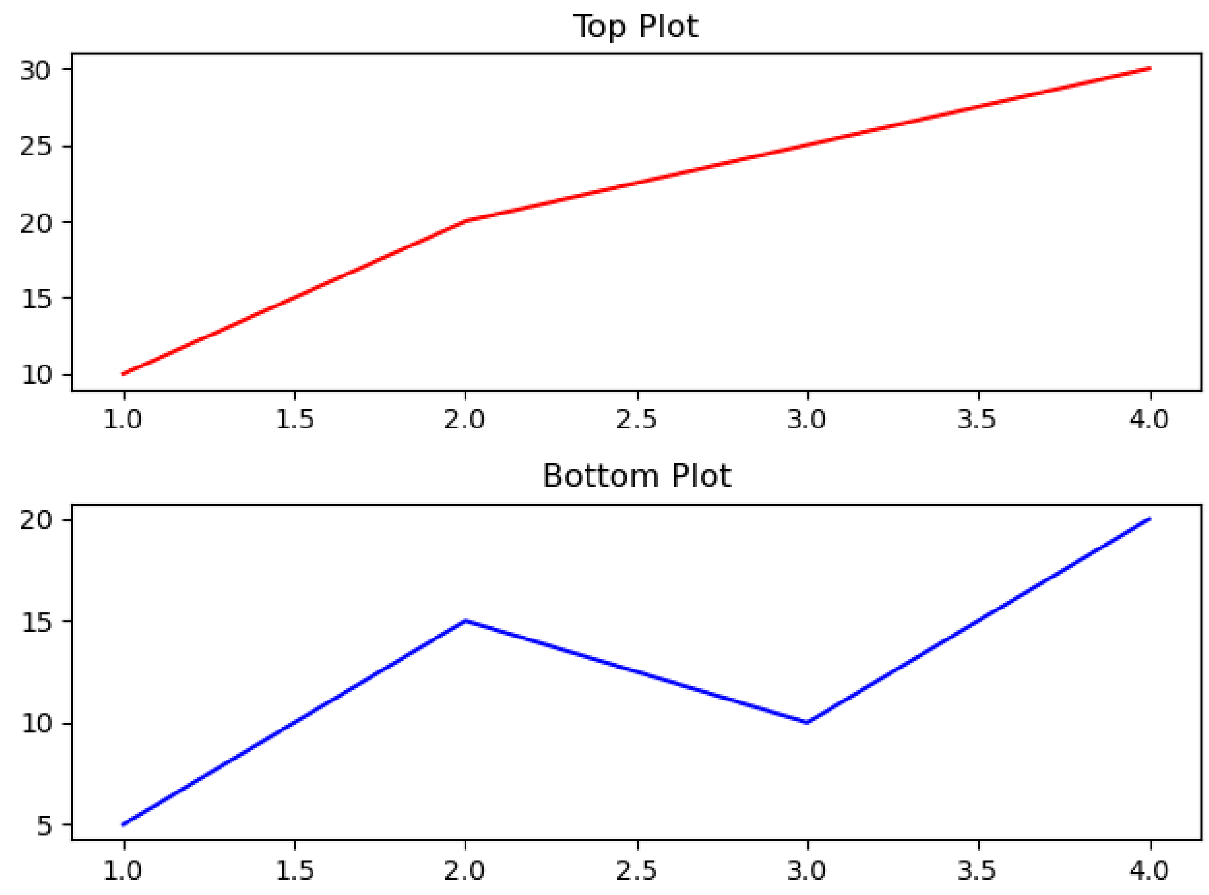


In [59]:

```python
plt.subplot(2, 1, 1)  # 2 rows, 1 column, 1st subplot
plt.plot(x, y1, 'r-')
plt.title('Top Plot')

plt.subplot(2, 1, 2)  # 2 rows, 1 column, 2nd subplot
plt.plot(x, y2, 'b-')
plt.title('Bottom Plot')

plt.tight_layout()  # Adjust spacing
plt.show()
```



In [60]:

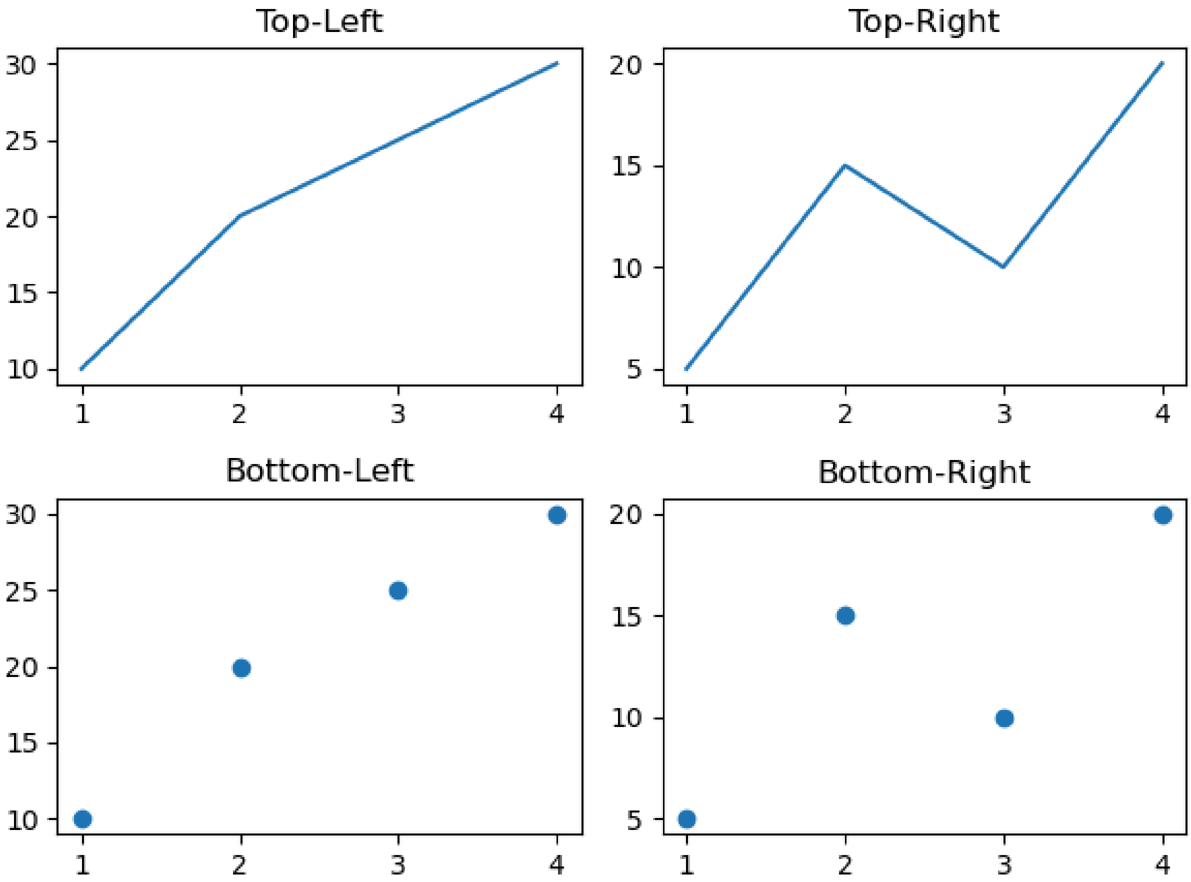
```
# Example 3: 2x2 Grid
plt.subplot(2, 2, 1)  # 2 rows, 2 columns, 1st subplot
plt.plot(x, y1)
plt.title('Top-Left')

plt.subplot(2, 2, 2)  # 2nd subplot
plt.plot(x, y2)
plt.title('Top-Right')

plt.subplot(2, 2, 3)  # 3rd subplot
plt.scatter(x, y1)
plt.title('Bottom-Left')

plt.subplot(2, 2, 4)  # 4th subplot
plt.scatter(x, y2)
plt.title('Bottom-Right')

plt.tight_layout()
plt.show()
```



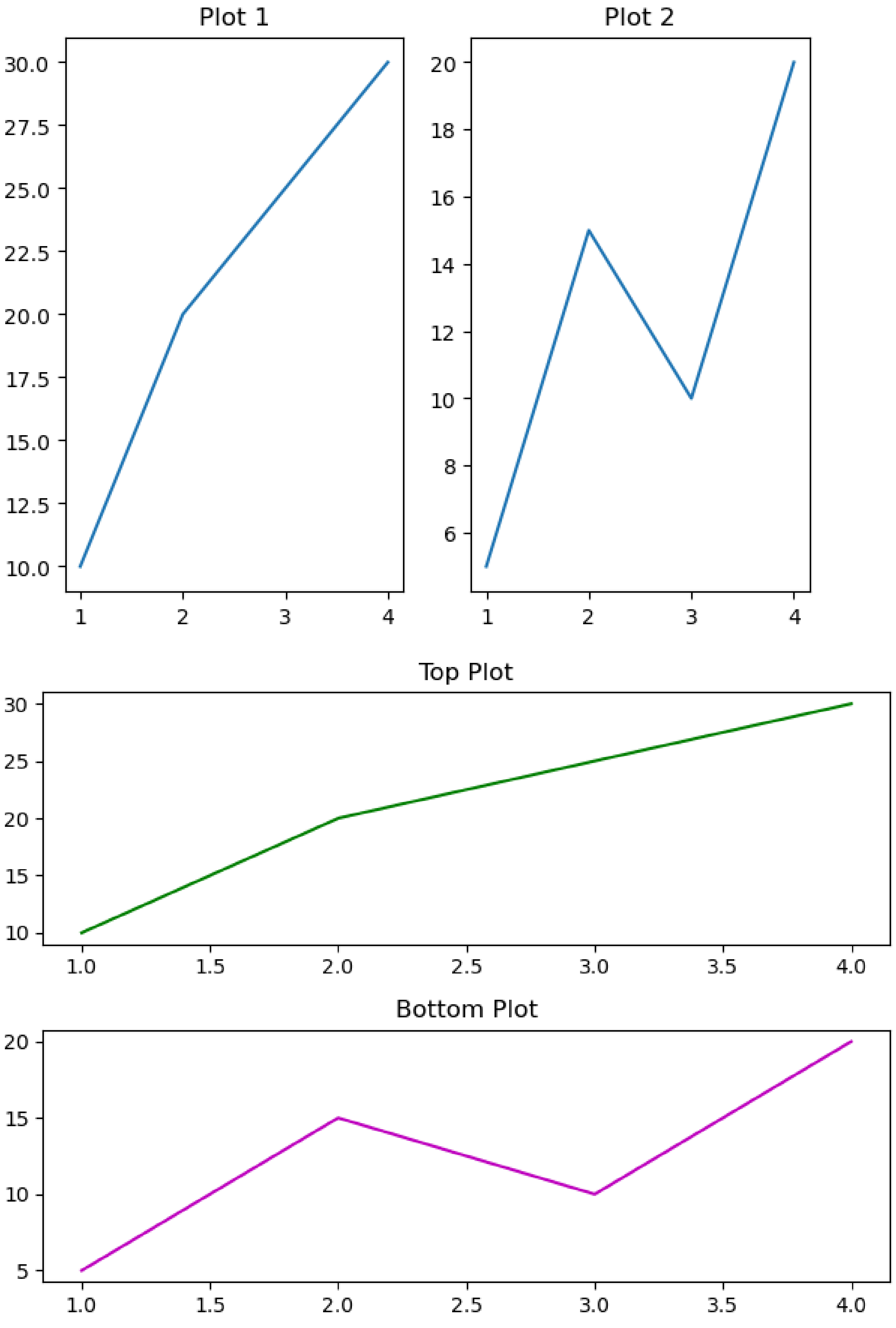
In [68]:

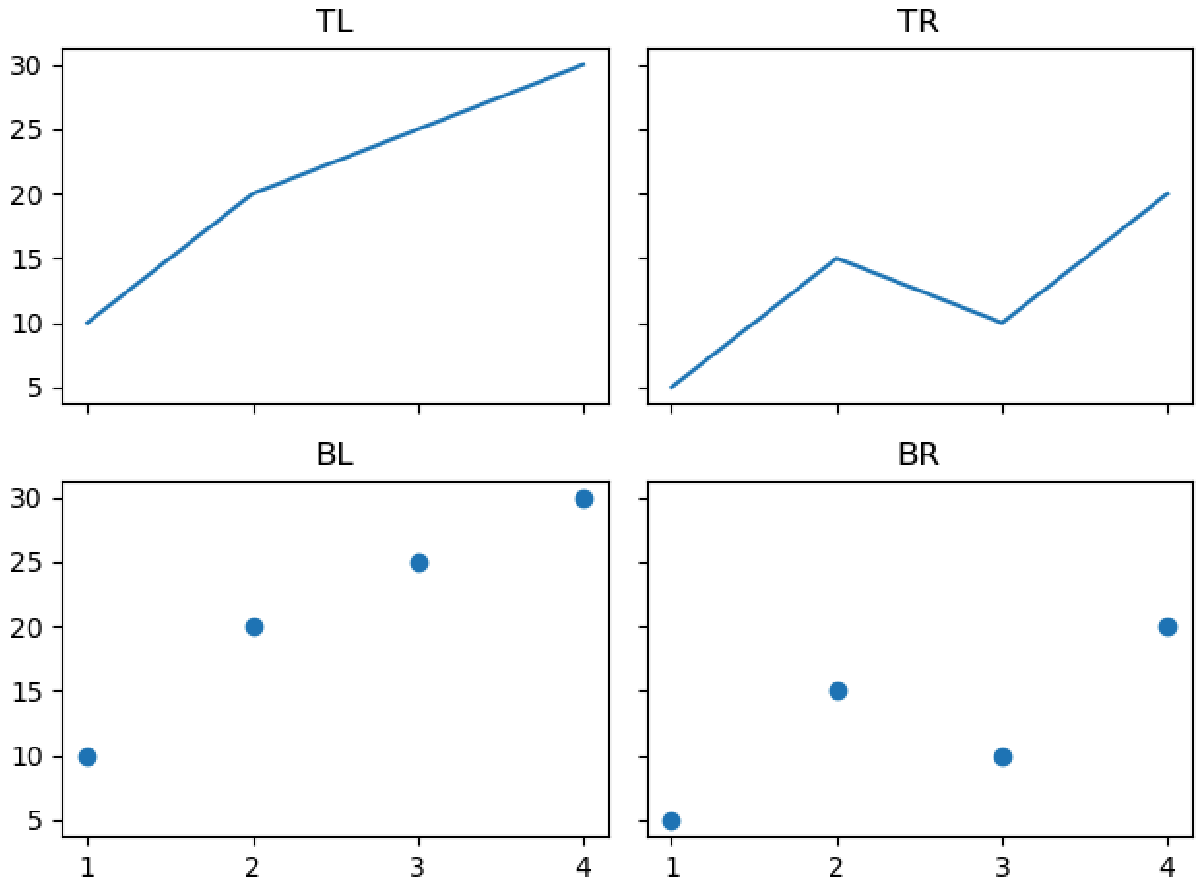
```
#fig, ax = plt.subplots(nrows=1, ncols=1, **kwargs)
# Example 1: Basic 1x2 Subplots
fig, ax = plt.subplots(1, 2)  # 1 row, 2 columns
ax[0].plot(x, y1)  # First subplot
ax[0].set_title('Plot 1')
ax[1].plot(x, y2)  # Second subplot
ax[1].set_title('Plot 2')
plt.show()

# Example 2: 2x1 Vertical Subplots
fig, ax = plt.subplots(2, 1)
ax[0].plot(x, y1, 'g-')
ax[0].set_title('Top Plot')
ax[1].plot(x, y2, 'm-')
ax[1].set_title('Bottom Plot')
fig.tight_layout()  # Adjust spacing
plt.show()

# Example 3: 2x2 Grid with Shared Axes
fig, ax = plt.subplots(2, 2, sharex=True, sharey=True)  # Share x and y axes
ax[0][0].plot(x, y1)
ax[0, 0].set_title('TL')
ax[0, 1].plot(x, y2)
ax[0, 1].set_title('TR')
ax[1, 0].scatter(x, y1)
ax[1, 0].set_title('BL')
ax[1, 1].scatter(x, y2)
ax[1, 1].set_title('BR')
```

```
plt.tight_layout()
plt.savefig('sample.png')
plt.show()
```

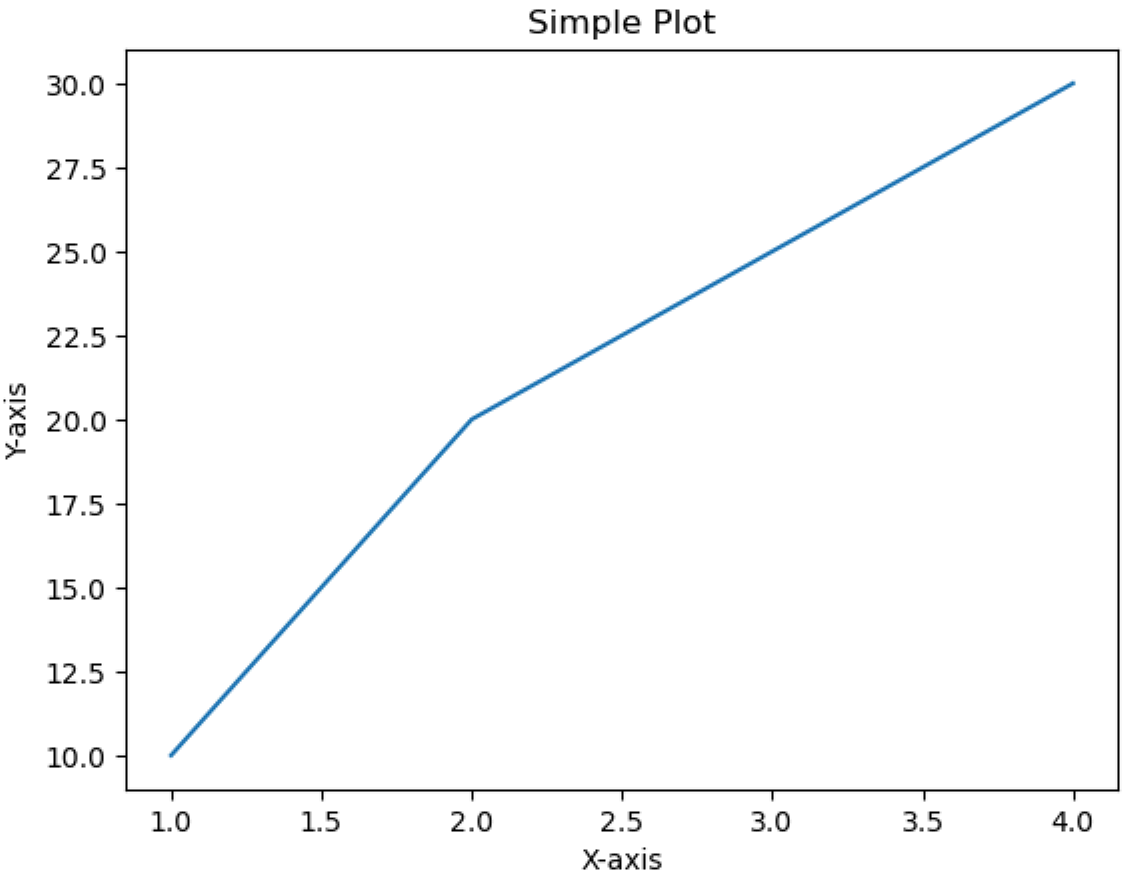




Saving Plots

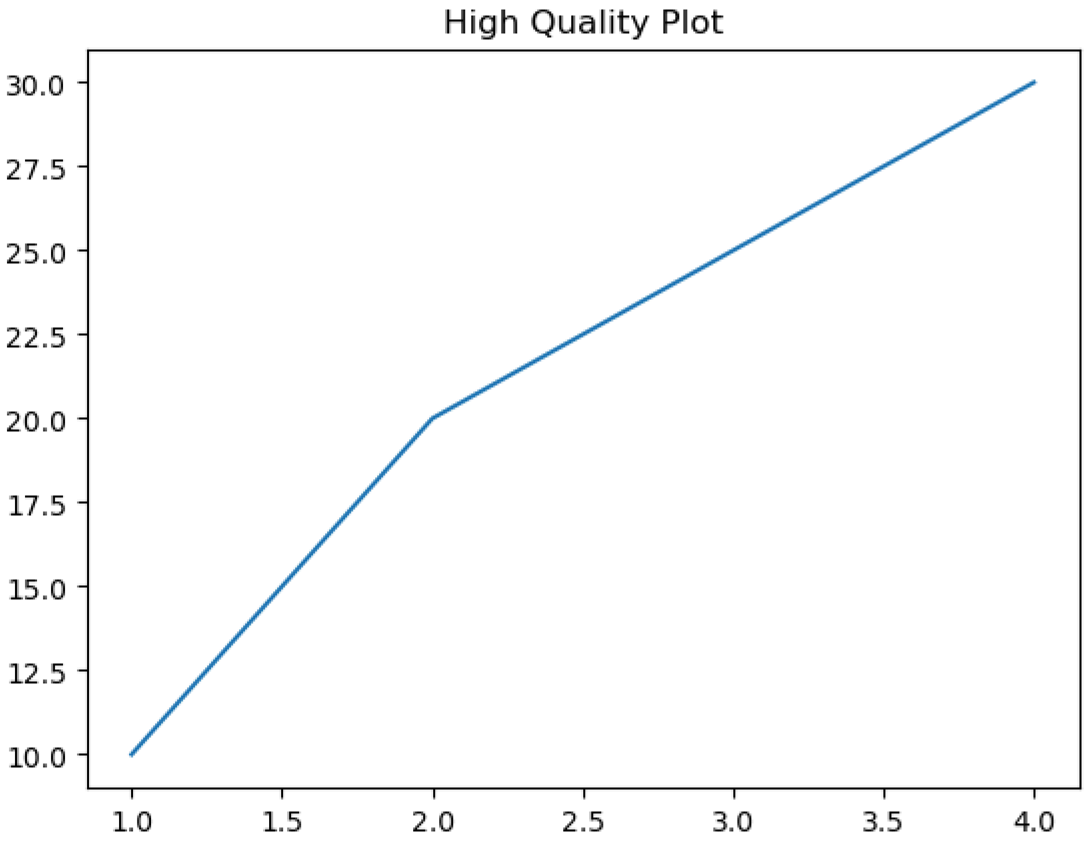
In [76]:

```
#plt.savefig(fname, dpi=None, format=None, bbox_inches='tight', **kwargs)
# Data for plotting
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]
                                        #File format (e.g., 'png', 'jpg', 'p
# Example 1: Basic Save as PNG
plt.plot(x, y)
plt.title('Simple Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.savefig('basic_plot.png')  # Saves as PNG in current directory
plt.show()
```



In [77]:

```
# Example 2: Save with Higher DPI (Quality)
plt.plot(x, y)
plt.title('High Quality Plot')
plt.savefig('high_quality_plot.png', dpi=300)  # Higher resolution
plt.show()
```



Plot Styling in Matplotlib

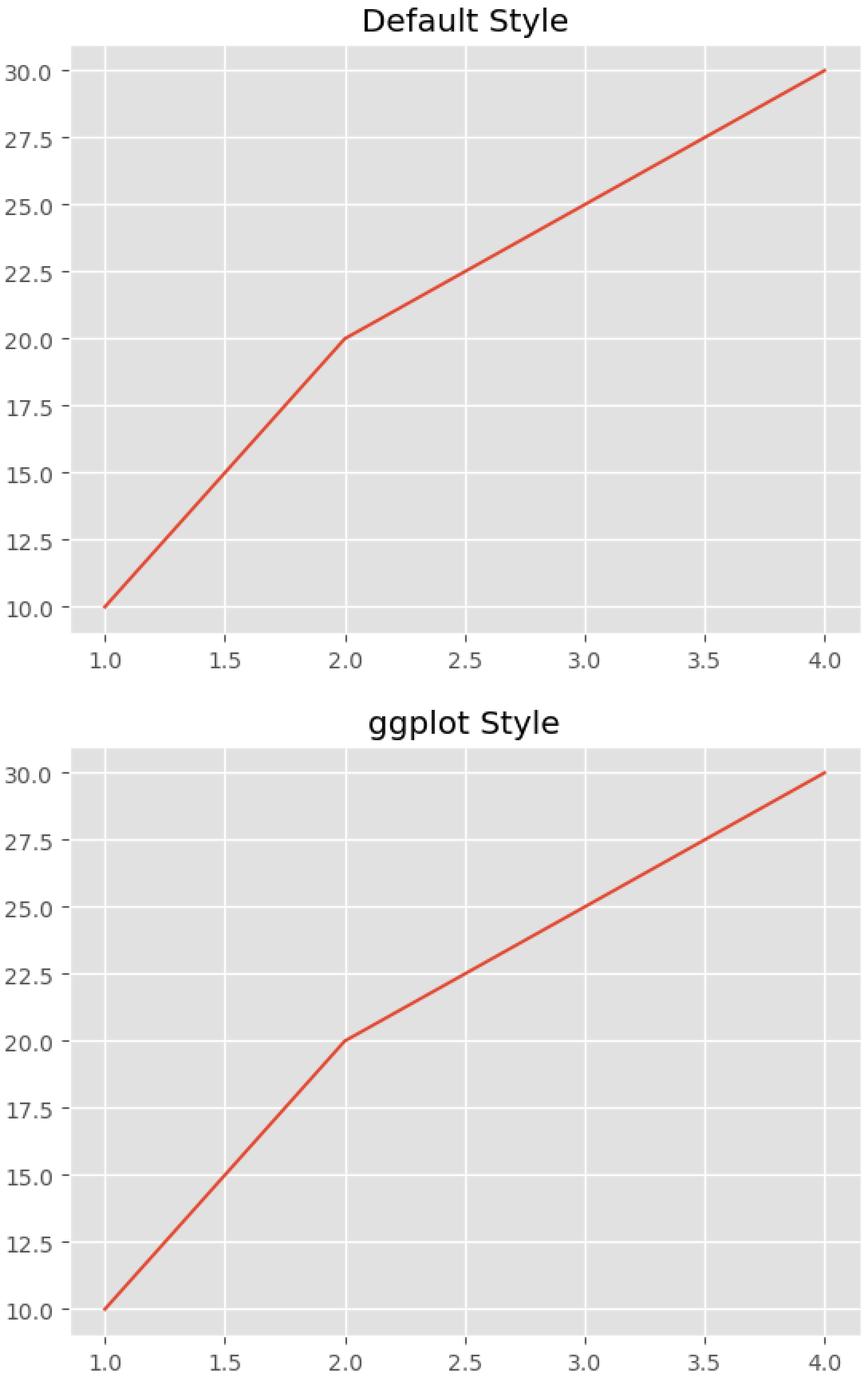
In [80]:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

# Example 1: Default Style (for comparison)
plt.plot(x, y)
plt.title('Default Style')
plt.show()

# Example 2: 'ggplot' Style
plt.style.use('ggplot')
plt.plot(x, y)
plt.title('ggplot Style')
plt.show()
```

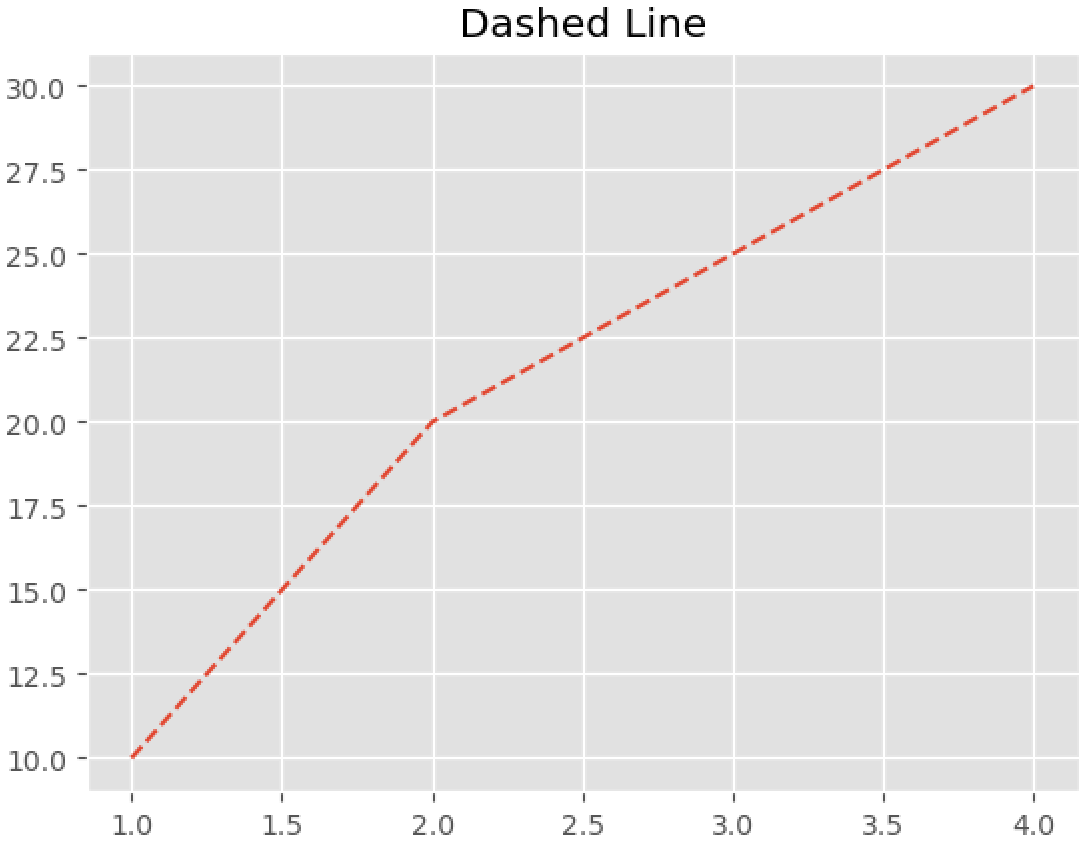


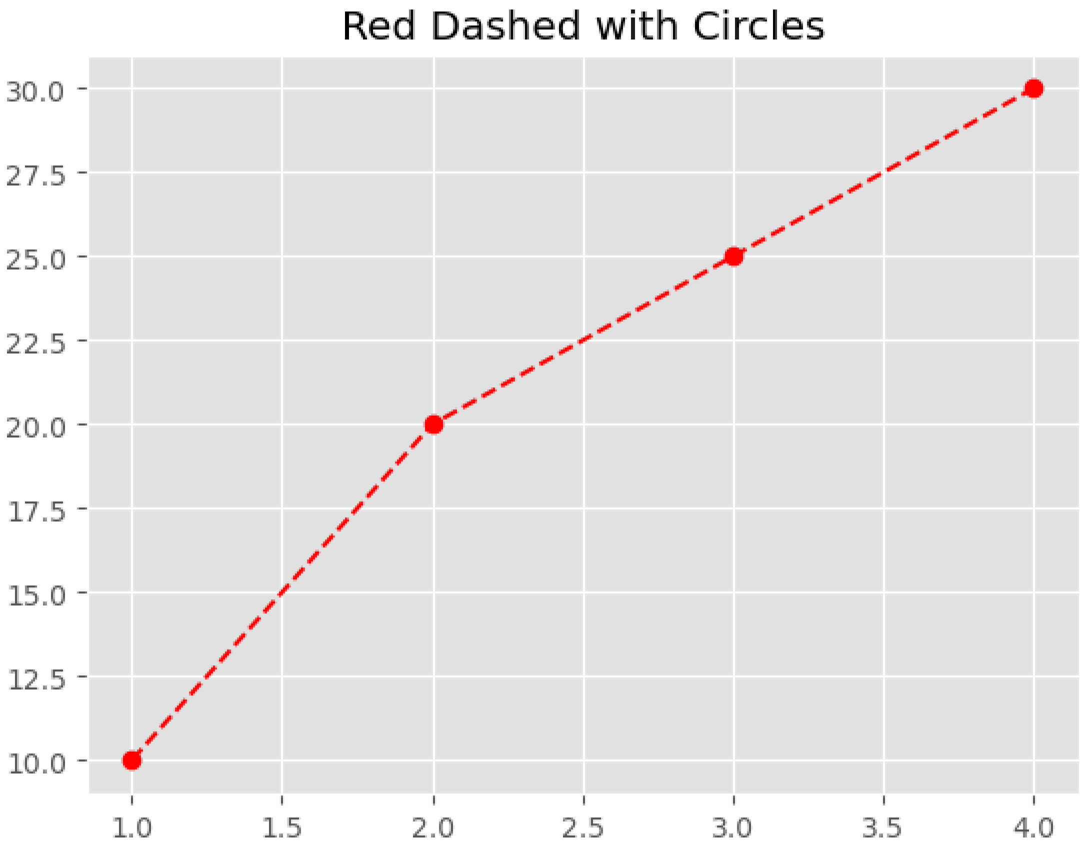
In [81]:
```
# Example 1: Basic Dashed Line
plt.plot(x, y, '--')  # Dashed line
plt.title('Dashed Line')
```

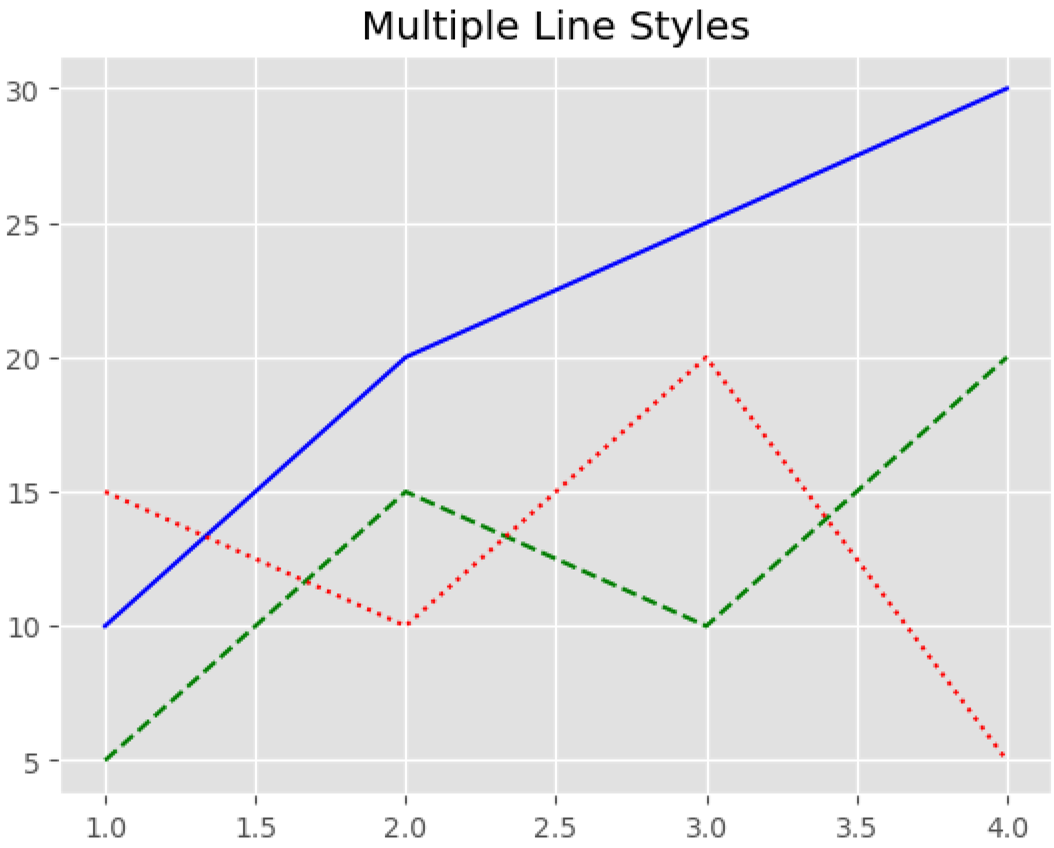
```python
plt.show()

# Example 2: Combined with Color and Marker
plt.plot(x, y, 'r--o')  # Red dashed line with circle markers
plt.title('Red Dashed with Circles')
plt.show()

# Example 3: Different Line Styles
plt.plot(x, y, 'b-')    # Blue solid line
plt.plot(x, [5, 15, 10, 20], 'g--')  # Green dashed line
plt.plot(x, [15, 10, 20, 5], 'r:')   # Red dotted line
plt.title('Multiple Line Styles')
plt.show()
```





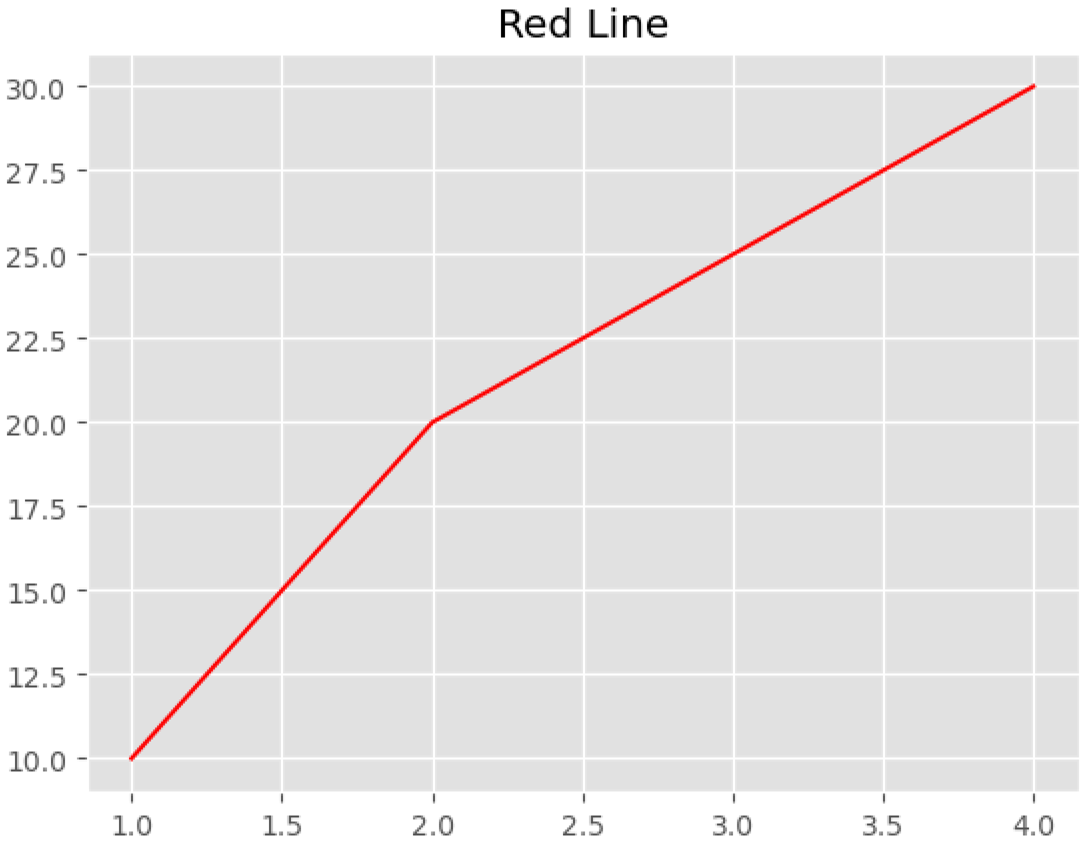


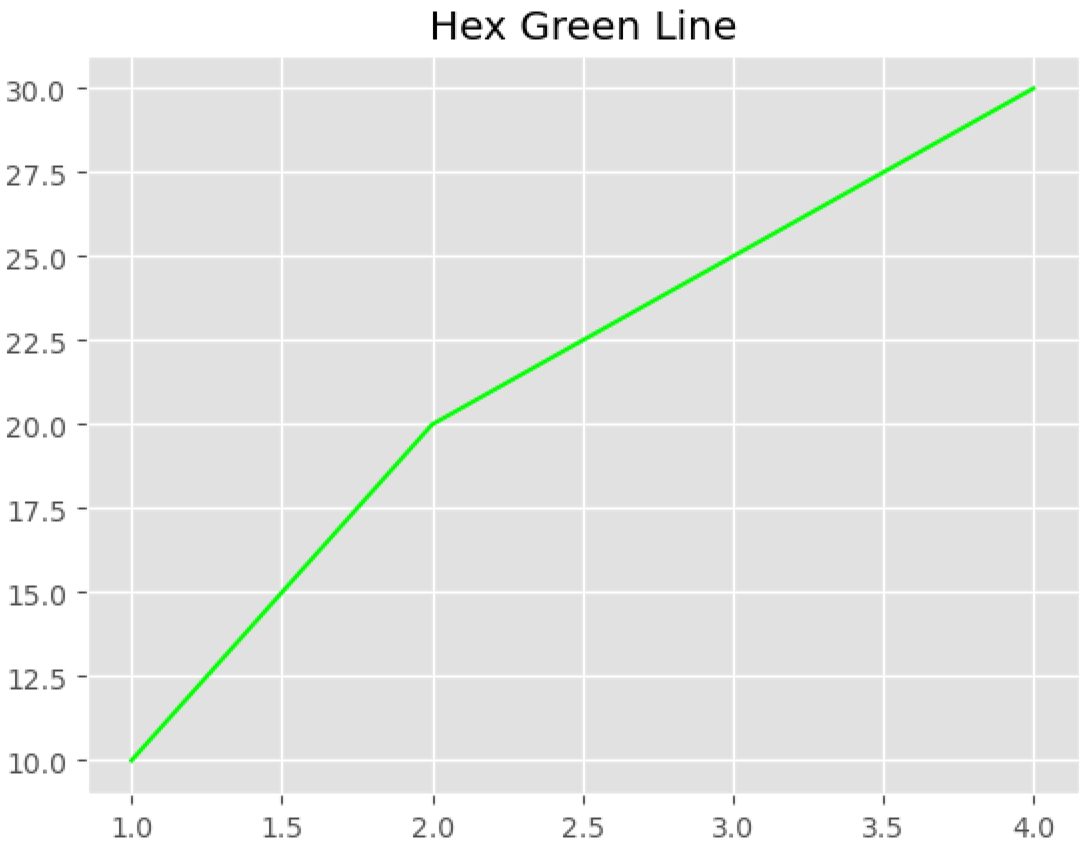
In [82]:

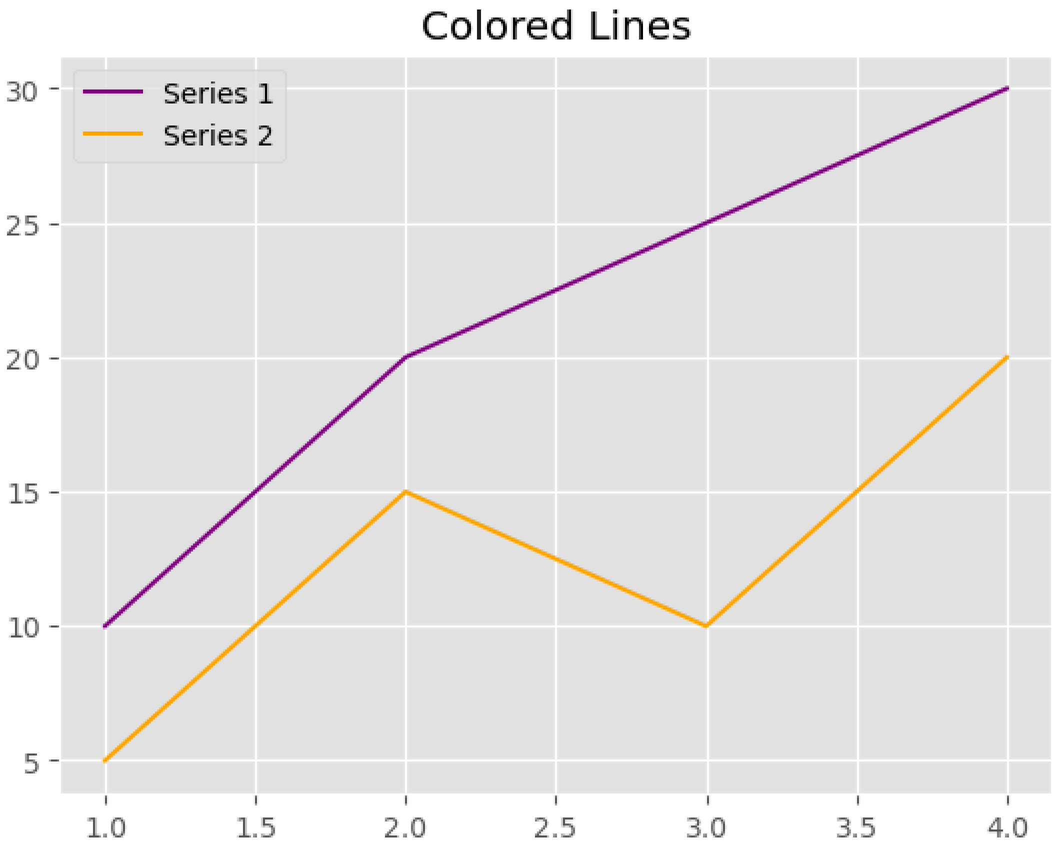
```
# Example 1: Basic Color
plt.plot(x, y, color='red')
plt.title('Red Line')
plt.show()

# Example 2: Hex Color
plt.plot(x, y, color='#00FF00')  # Green in hex
plt.title('Hex Green Line')
plt.show()

# Example 3: Multiple Colored Lines
plt.plot(x, y, color='purple', label='Series 1')
plt.plot(x, [5, 15, 10, 20], color='orange', label='Series 2')
plt.legend()
plt.title('Colored Lines')
plt.show()
```





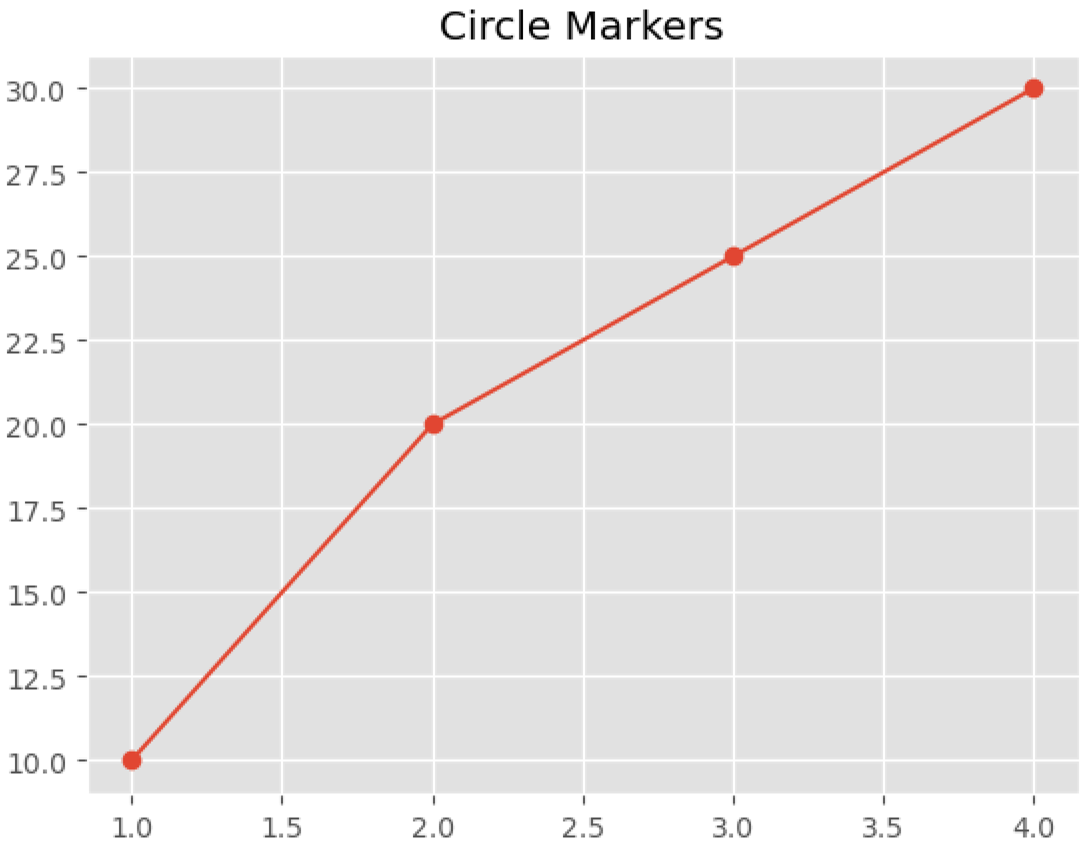


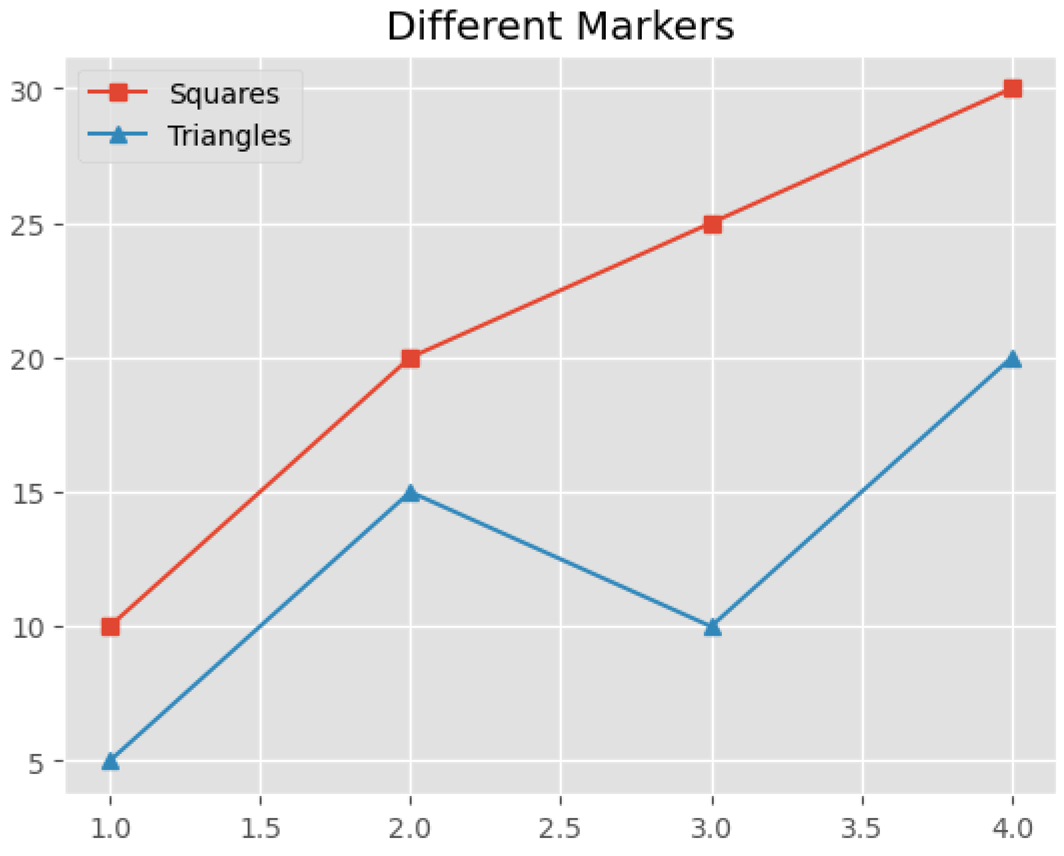
In [83]:

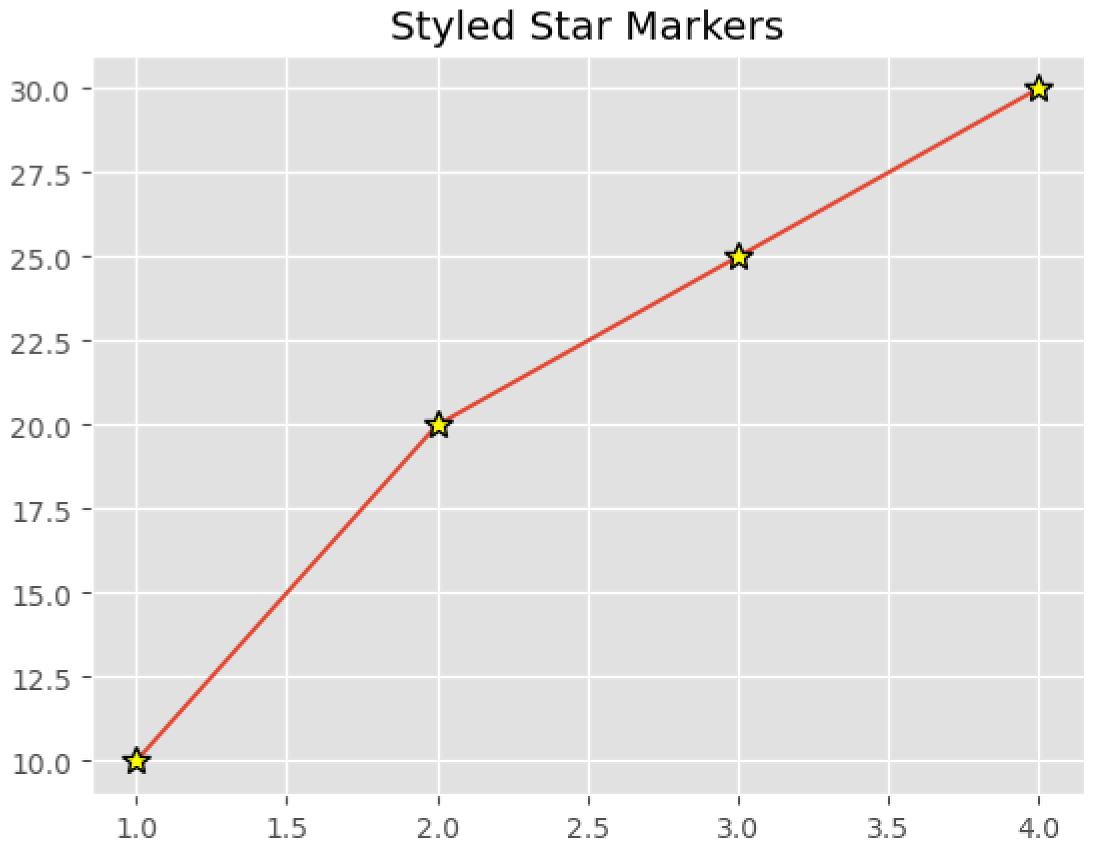
```
# Example 1: Basic Markers
plt.plot(x, y, marker='o')  # Circle markers
plt.title('Circle Markers')
plt.show()

# Example 2: Different Marker Types
plt.plot(x, y, marker='s', label='Squares')  # Square markers
plt.plot(x, [5, 15, 10, 20], marker='^', label='Triangles')  # Triangle mark
plt.legend()
plt.title('Different Markers')
plt.show()

# Example 3: Marker with Size and Color
plt.plot(x, y, marker='*', markersize=10, markerfacecolor='yellow', markered
plt.title('Styled Star Markers')
plt.show()
```





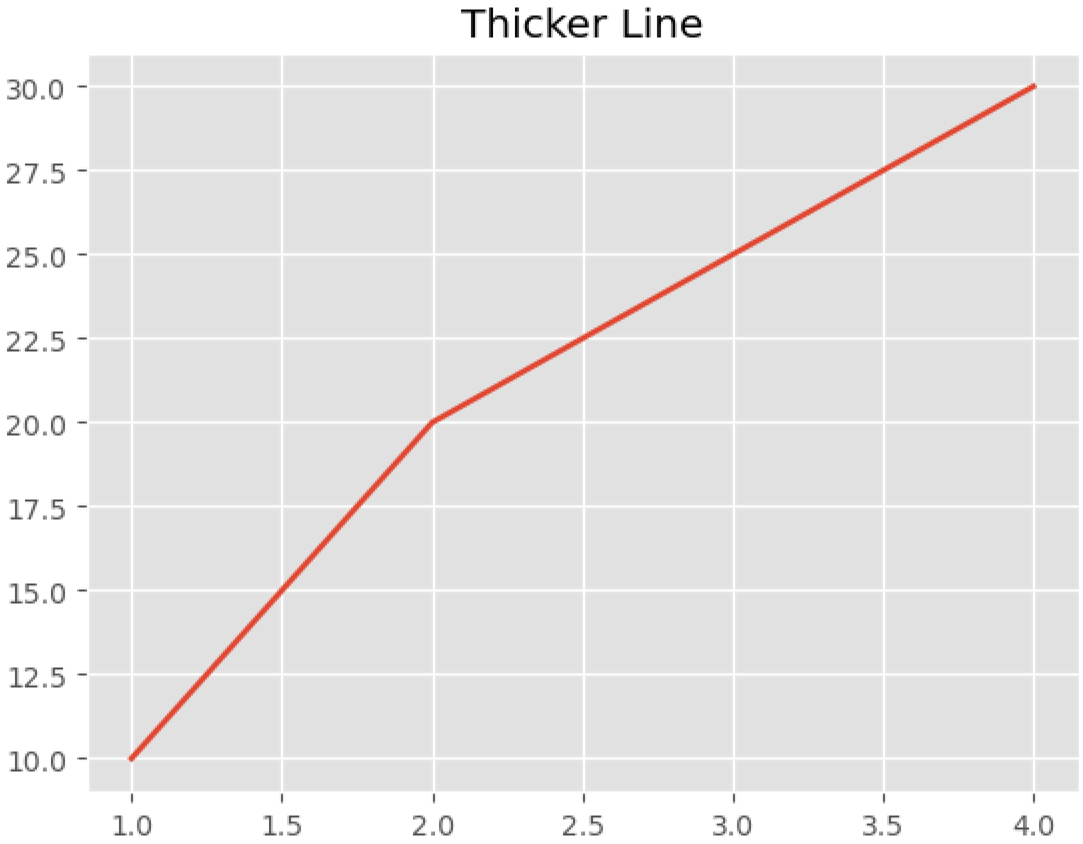


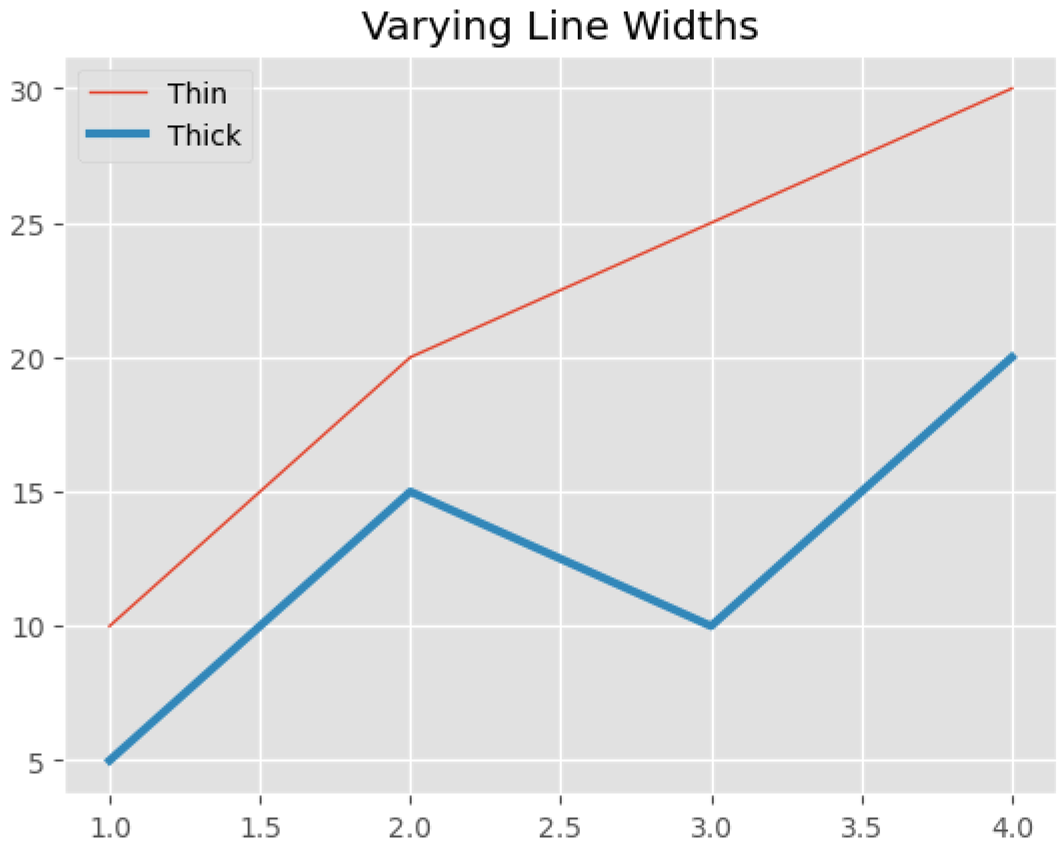
In [84]:

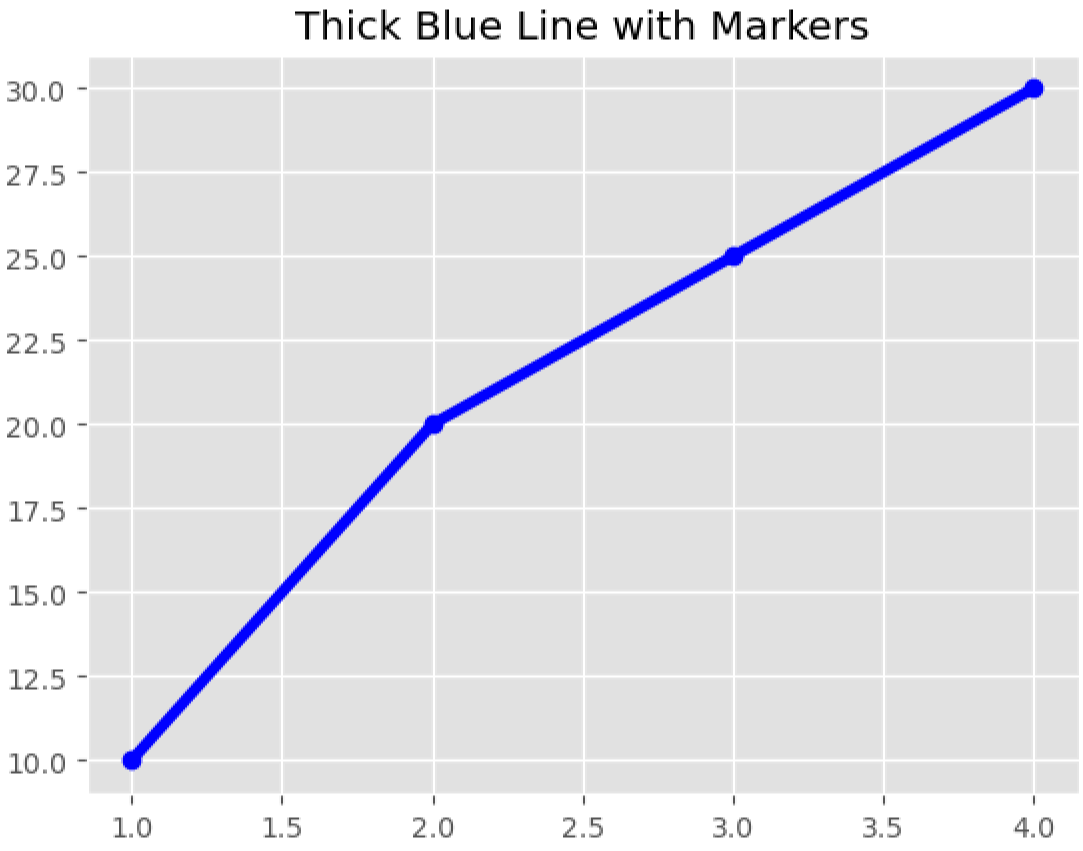
```
# Example 1: Basic Line Width
plt.plot(x, y, linewidth=2)
plt.title('Thicker Line')
plt.show()

# Example 2: Varying Widths
plt.plot(x, y, linewidth=1, label='Thin')
plt.plot(x, [5, 15, 10, 20], linewidth=3, label='Thick')
plt.legend()
plt.title('Varying Line Widths')
plt.show()

# Example 3: With Color and Marker
plt.plot(x, y, linewidth=4, color='blue', marker='o')
plt.title('Thick Blue Line with Markers')
plt.show()
```





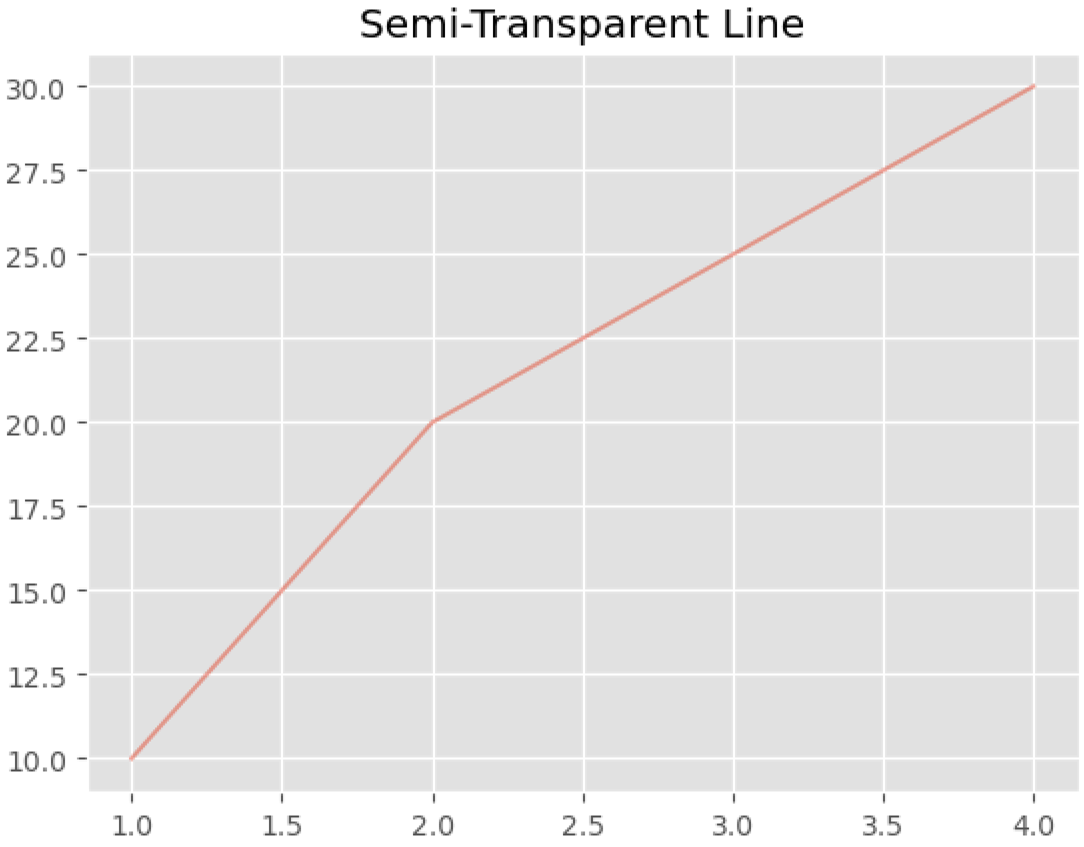


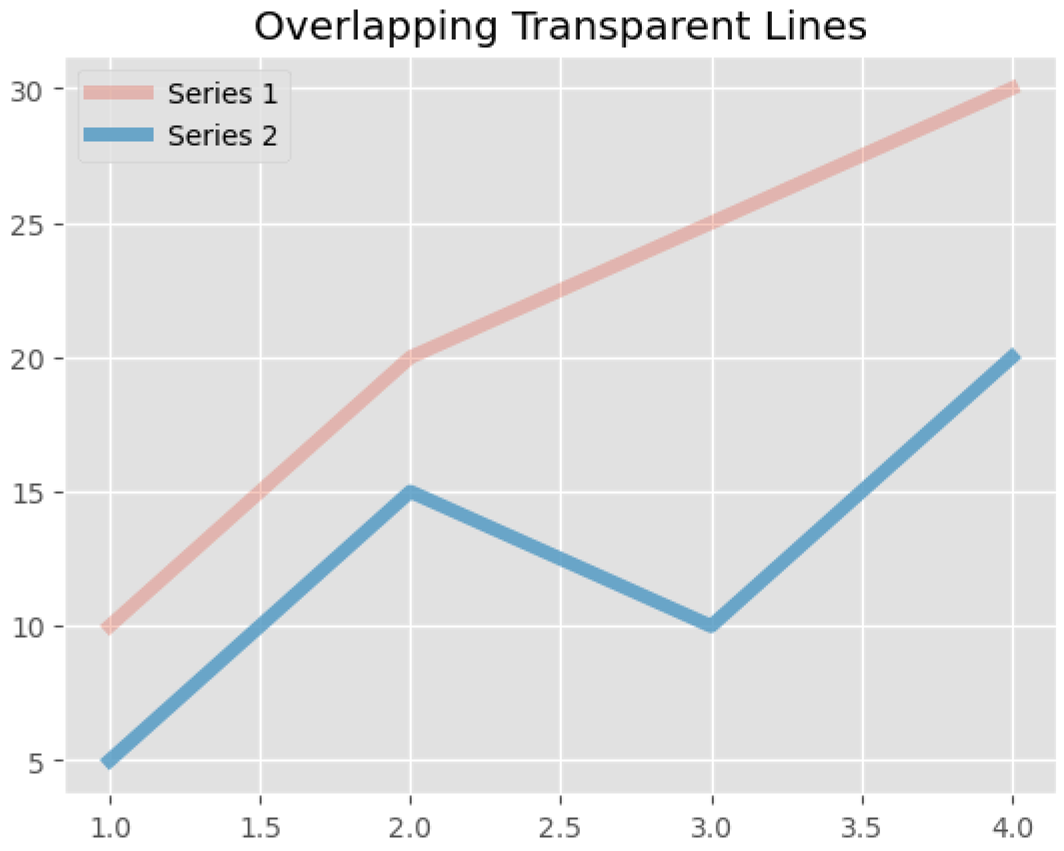
In [85]:

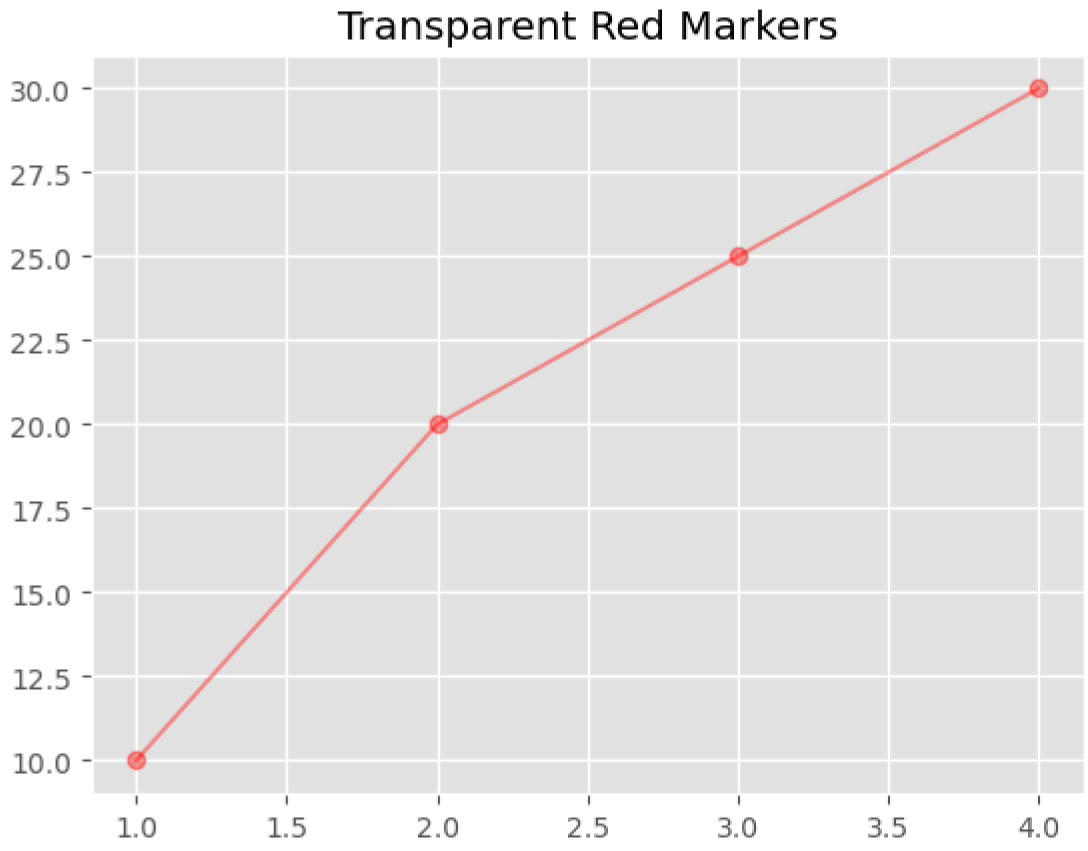
```
# Example 1: Basic Transparency
plt.plot(x, y, alpha=0.5)
plt.title('Semi-Transparent Line')
plt.show()

# Example 2: Overlapping Lines
plt.plot(x, y, alpha=0.3, linewidth=5, label='Series 1')
plt.plot(x, [5, 15, 10, 20], alpha=0.7, linewidth=5, label='Series 2')
plt.legend()
plt.title('Overlapping Transparent Lines')
plt.show()

# Example 3: With Markers
plt.plot(x, y, marker='o', alpha=0.4, color='red')
plt.title('Transparent Red Markers')
plt.show()
```




Overlapping Transparent Lines
Series 1
Series 2
30
25
20
15
10
5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Transparent Red Markers
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0