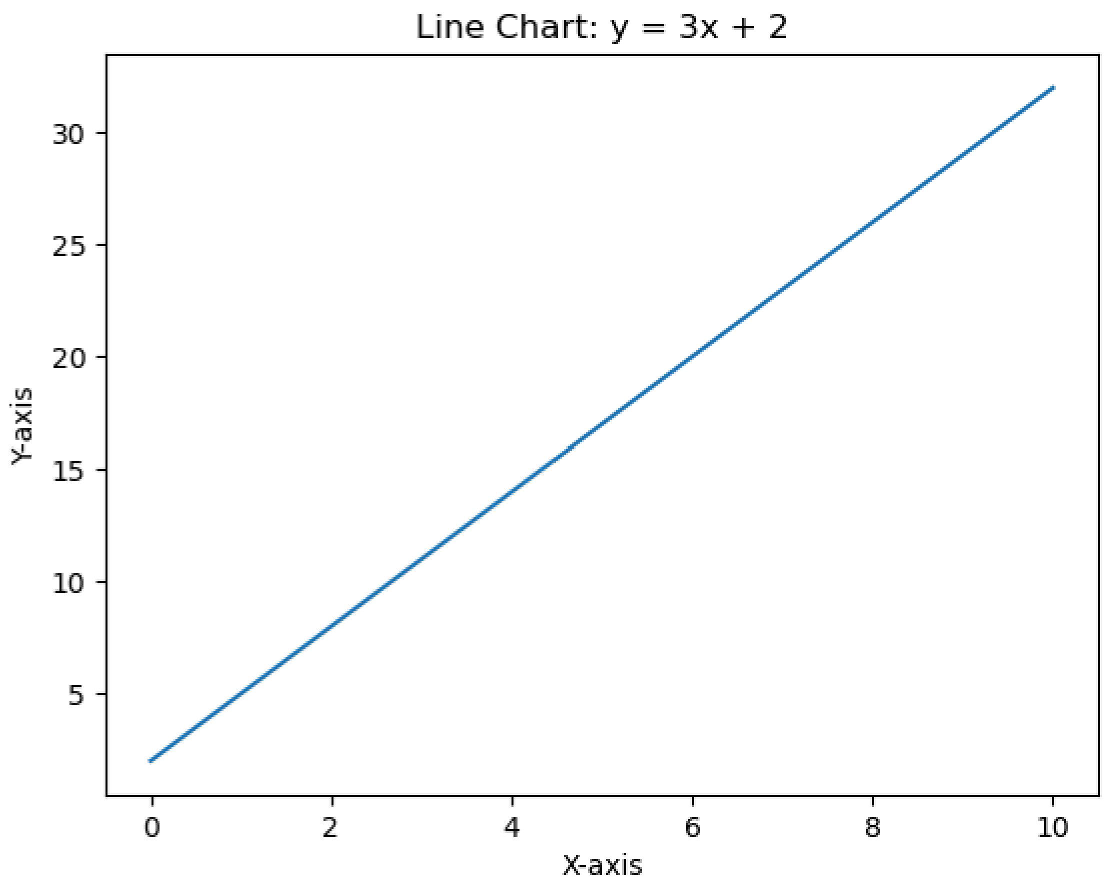
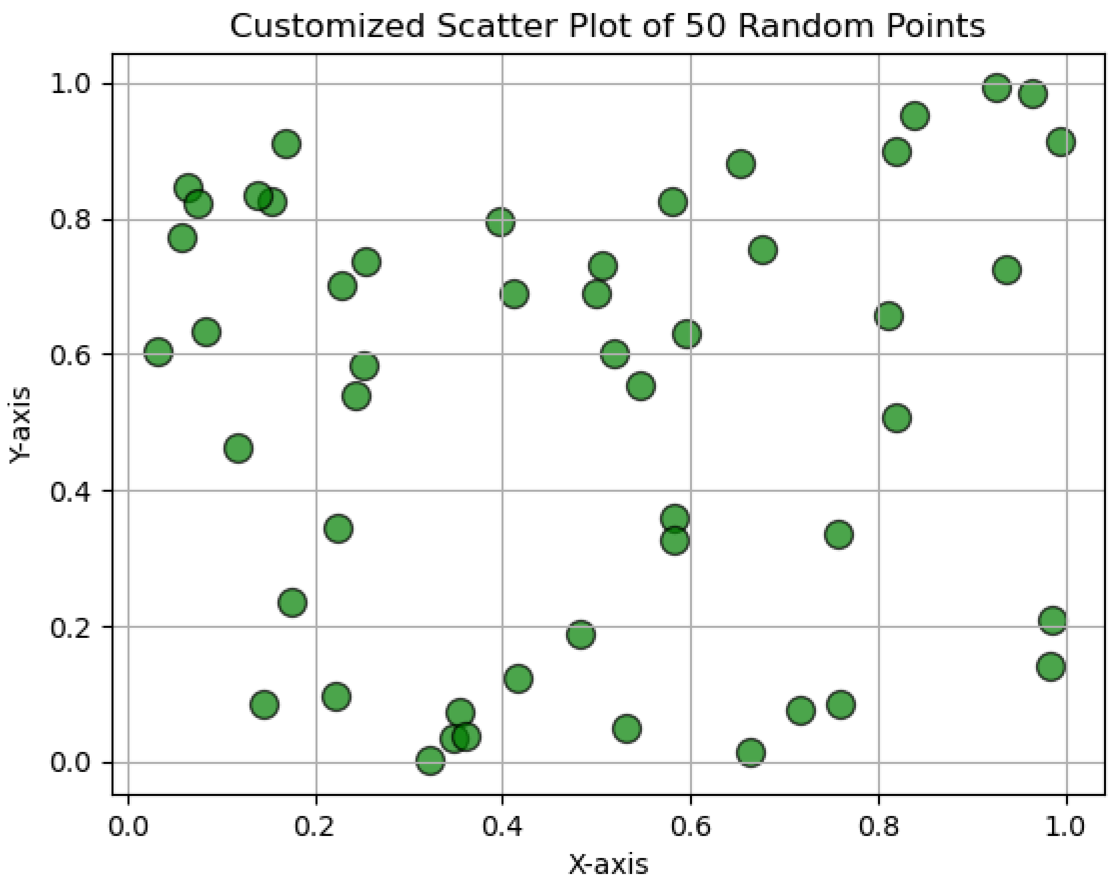
1)Plot a Line Chart: Plot y = 3x + 2 for x in the range 0 to 10. Set the x-axis and y-axis labels, and give the plot a title.

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)
y = 3 * x + 2
plt.plot(x, y, label='y = 3x + 2', color='blue')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Line Chart of y = 3x + 2')
plt.grid(True)
plt.legend()
plt.show()
```



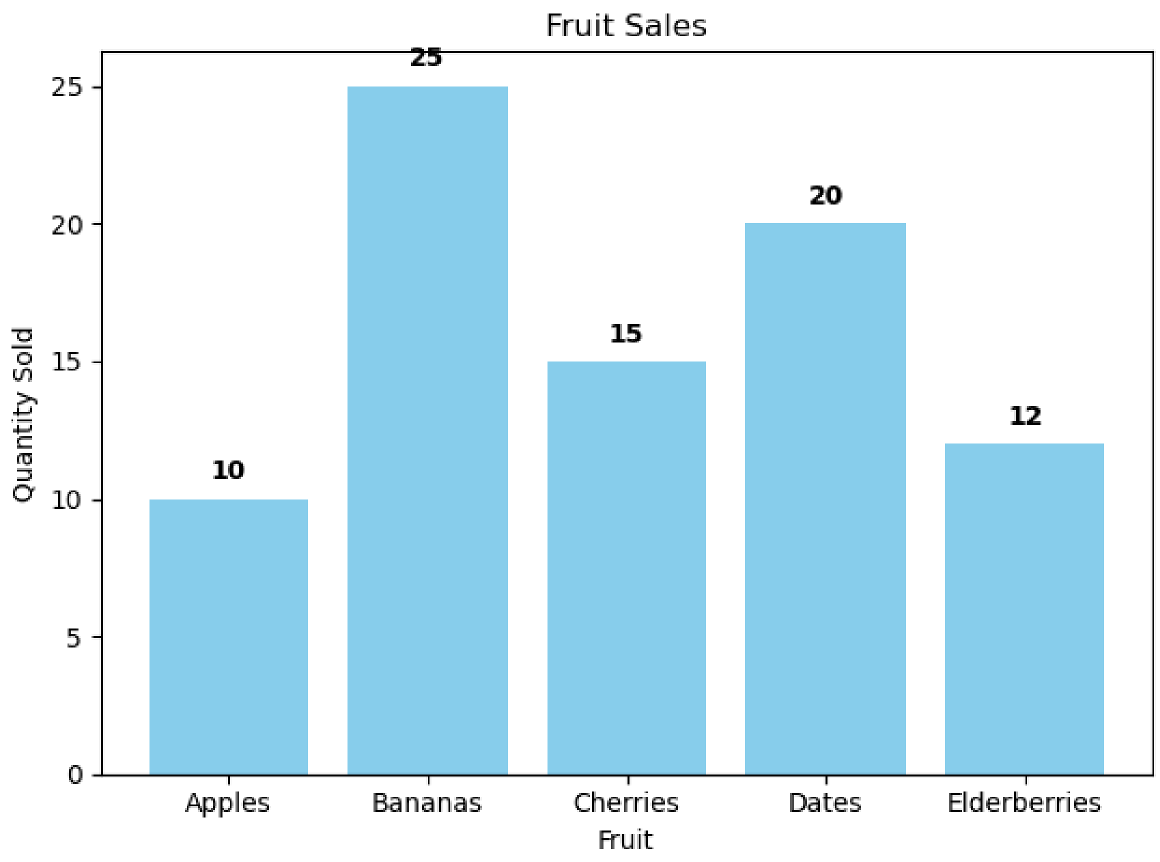
2)Scatter Plot with Customization: Create a scatter plot of 50 random points. Change the color of the points, increase their size, and add a grid to the plot.

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.rand(50)
y = np.random.rand(50)
plt.scatter(x, y, color='green', s=100, alpha=0.7, edgecolors='black')
plt.title('Customized Scatter Plot of 50 Random Points')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.grid(True)
plt.show()
```



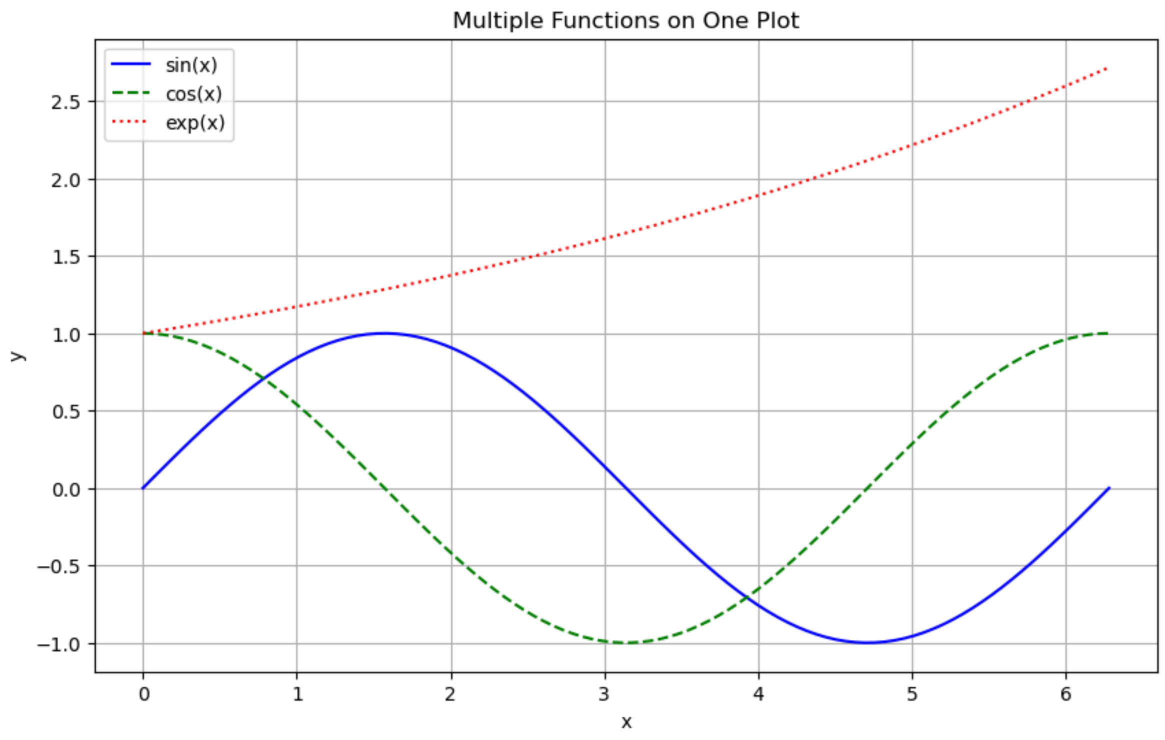
3)Bar Chart with Annotations: Draw a vertical bar chart for five categories of your choice. Annotate the value on top of each bar.

In [2]:
```
import matplotlib.pyplot as plt
categories = ['Apples', 'Bananas', 'Cherries', 'Dates', 'Elderberries']
values = [10, 25, 15, 20, 12]
plt.bar(categories, values, color='skyblue')
plt.title('Fruit Sales')
plt.xlabel('Fruit')
plt.ylabel('Quantity Sold')
for i, value in enumerate(values):
    plt.text(i, value + 0.5, str(value), ha='center', va='bottom', fontweight='b
plt.tight_layout()
plt.show()
```



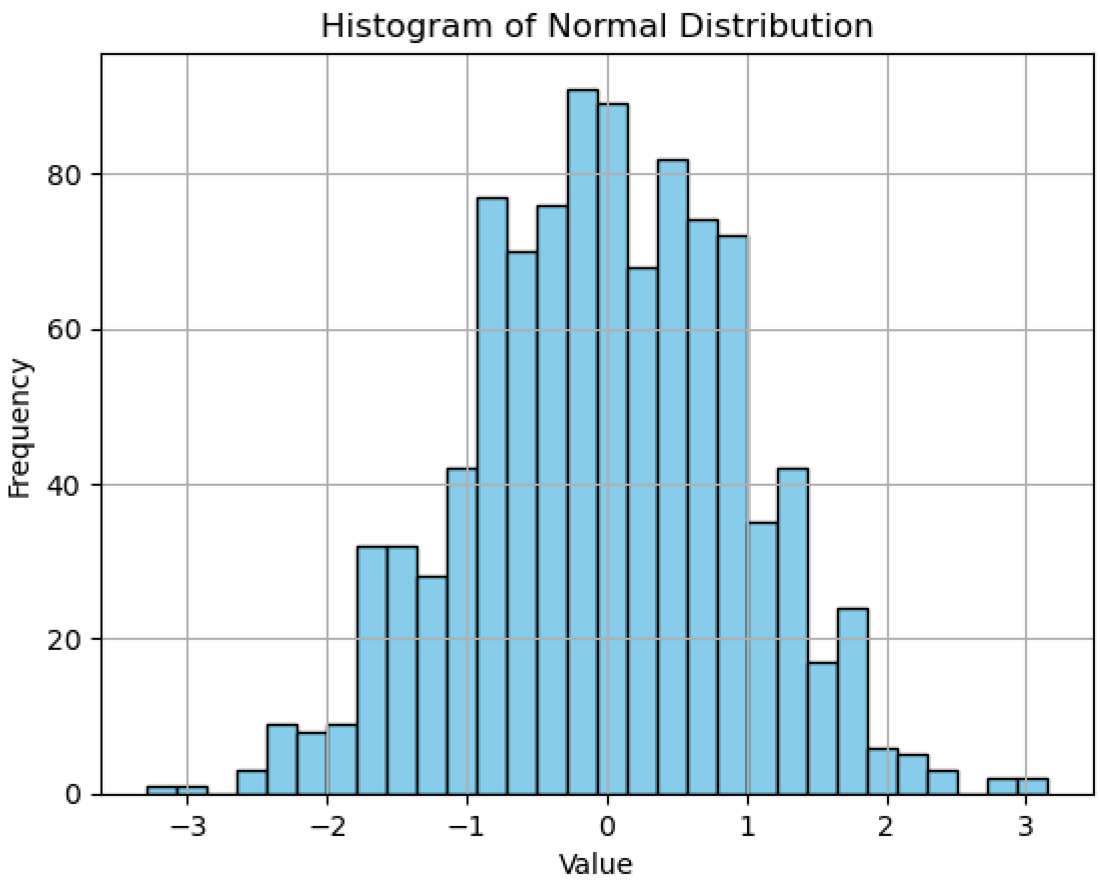
4)Multiple Lines in One Plot: Plot three mathematical functions (sin, cos, exp) on the same axes, each with different colors and line styles. Add a legend.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 2 * np.pi, 100)
y_sin = np.sin(x)
y_cos = np.cos(x)
y_exp = np.exp(x / (2 * np.pi))
plt.figure(figsize=(10, 6))
plt.plot(x, y_sin, label='sin(x)', color='blue', linestyle='-')
plt.plot(x, y_cos, label='cos(x)', color='green', linestyle='--')
plt.plot(x, y_exp, label='exp(x)', color='red', linestyle=':')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Multiple Functions on One Plot')
plt.grid(True)
plt.show()
```



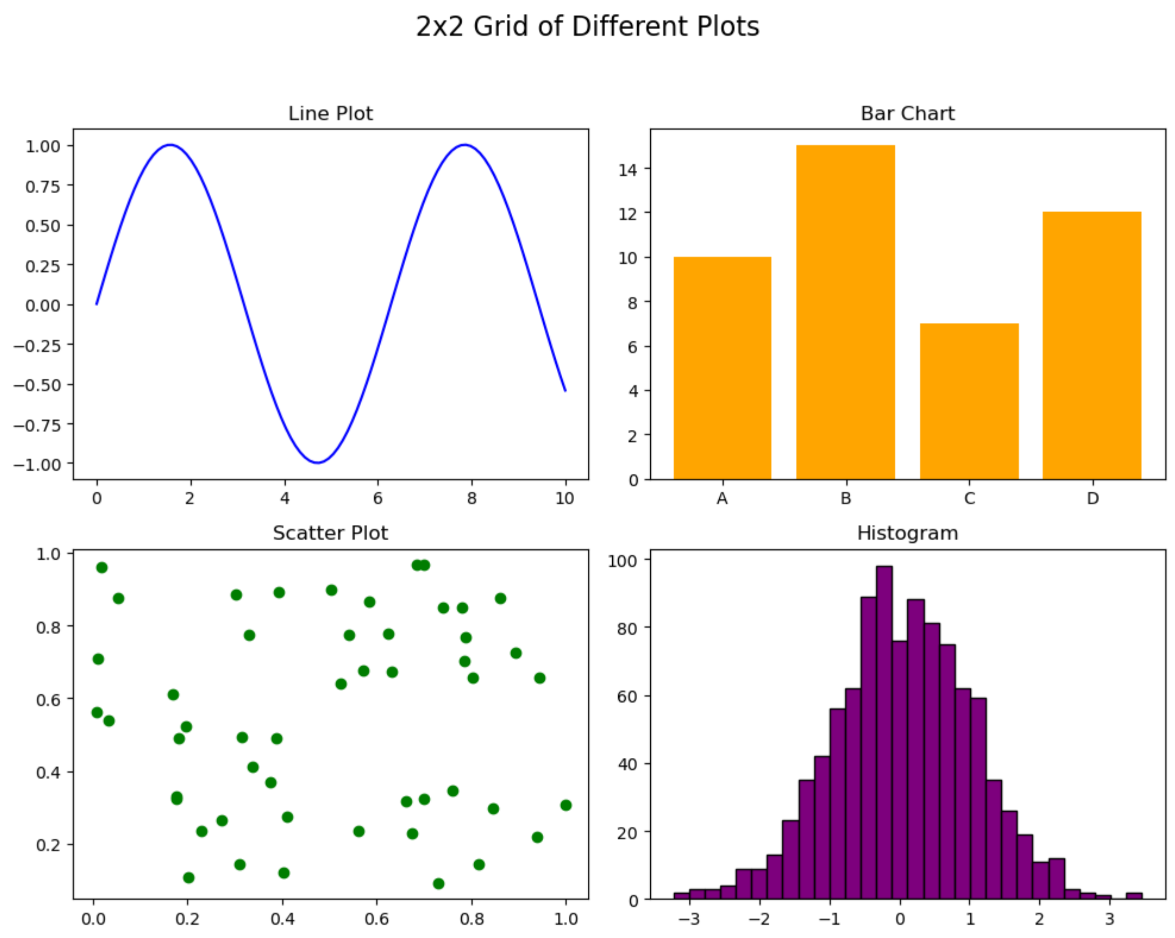
5)Histogram with Custom Bins: Generate 1000 random numbers from a normal distribution and plot their histogram with 30 bins. Change the bar color and add axis labels

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
data = np.random.randn(1000)
plt.hist(data, bins=30, color='skyblue', edgecolor='black')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram of Normal Distribution')
plt.show()
```



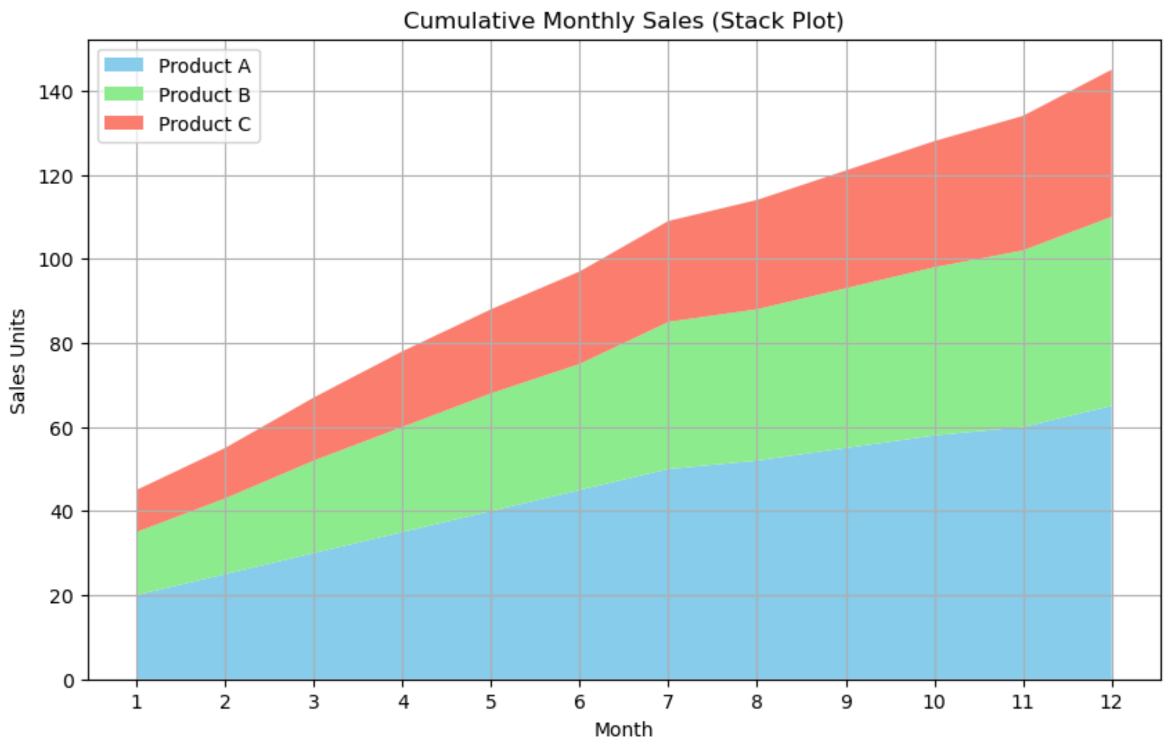
6)Subplots Grid: Create a 2x2 grid of subplots: line plot, bar chart, scatter plot, and histogram, each in a different subplot. Add an overall (supertitle) for the figure.

In [3]:
```python
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 10, 100)
y = np.sin(x)
categories = ['A', 'B', 'C', 'D']
values = [10, 15, 7, 12]
x_scatter = np.random.rand(50)
y_scatter = np.random.rand(50)
hist_data = np.random.randn(1000)
fig, axs = plt.subplots(2, 2, figsize=(10, 8))
axs[0, 0].plot(x, y, color='blue')
axs[0, 0].set_title('Line Plot')
axs[0, 1].bar(categories, values, color='orange')
axs[0, 1].set_title('Bar Chart')
axs[1, 0].scatter(x_scatter, y_scatter, color='green')
axs[1, 0].set_title('Scatter Plot')
axs[1, 1].hist(hist_data, bins=30, color='purple', edgecolor='black')
axs[1, 1].set_title('Histogram')
fig.suptitle('2x2 Grid of Different Plots', fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```



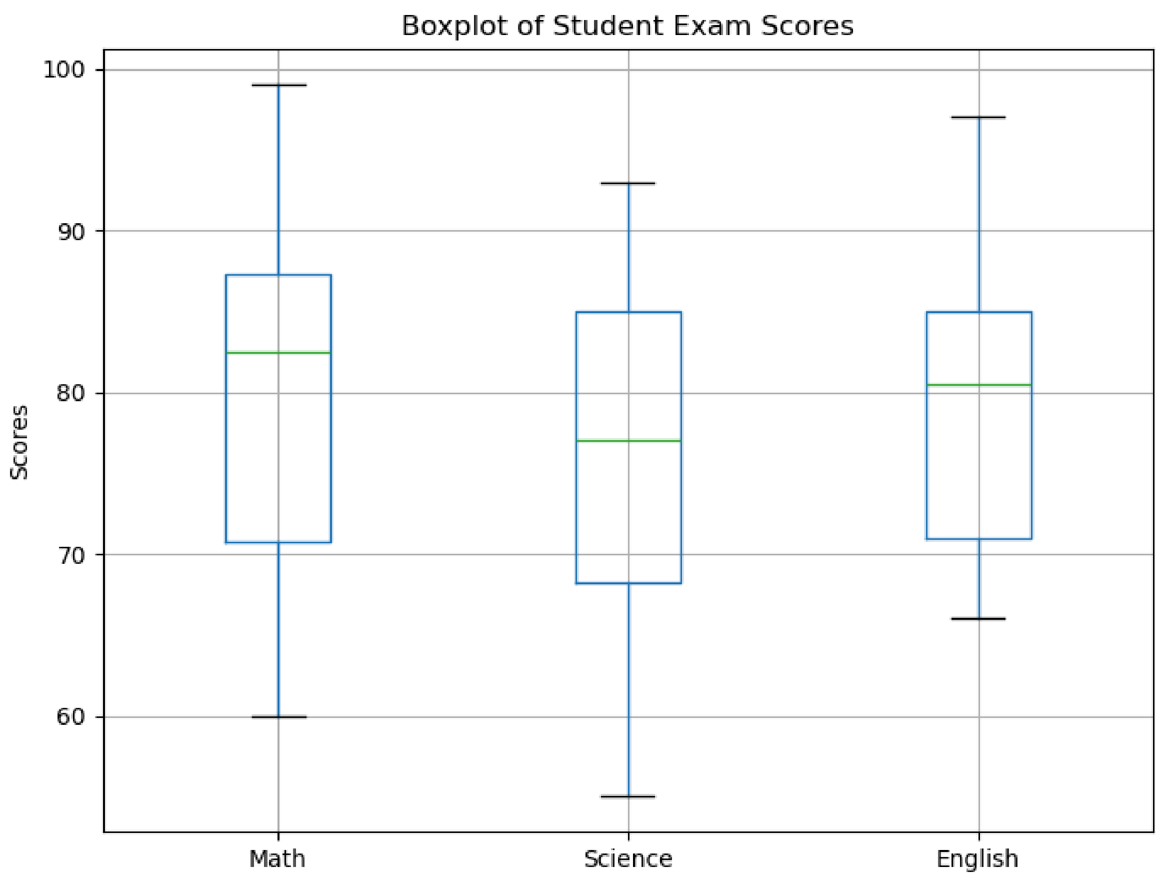
7)Stack Plot: Plot the cumulative sales of 3 products over 12 months as a stack plot using made-up data.

In [4]:
```
import matplotlib.pyplot as plt
import numpy as np
months = np.arange(1, 13)
product_A = [20, 25, 30, 35, 40, 45, 50, 52, 55, 58, 60, 65]
product_B = [15, 18, 22, 25, 28, 30, 35, 36, 38, 40, 42, 45]
product_C = [10, 12, 15, 18, 20, 22, 24, 26, 28, 30, 32, 35]
plt.figure(figsize=(10, 6))
plt.stackplot(months, product_A, product_B, product_C,
              labels=['Product A', 'Product B', 'Product C'],
              colors=['skyblue', 'lightgreen', 'salmon'])
plt.title('Cumulative Monthly Sales (Stack Plot)')
plt.xlabel('Month')
plt.ylabel('Sales Units')
plt.legend(loc='upper left')
plt.xticks(months)
plt.grid(True)
plt.show()
```



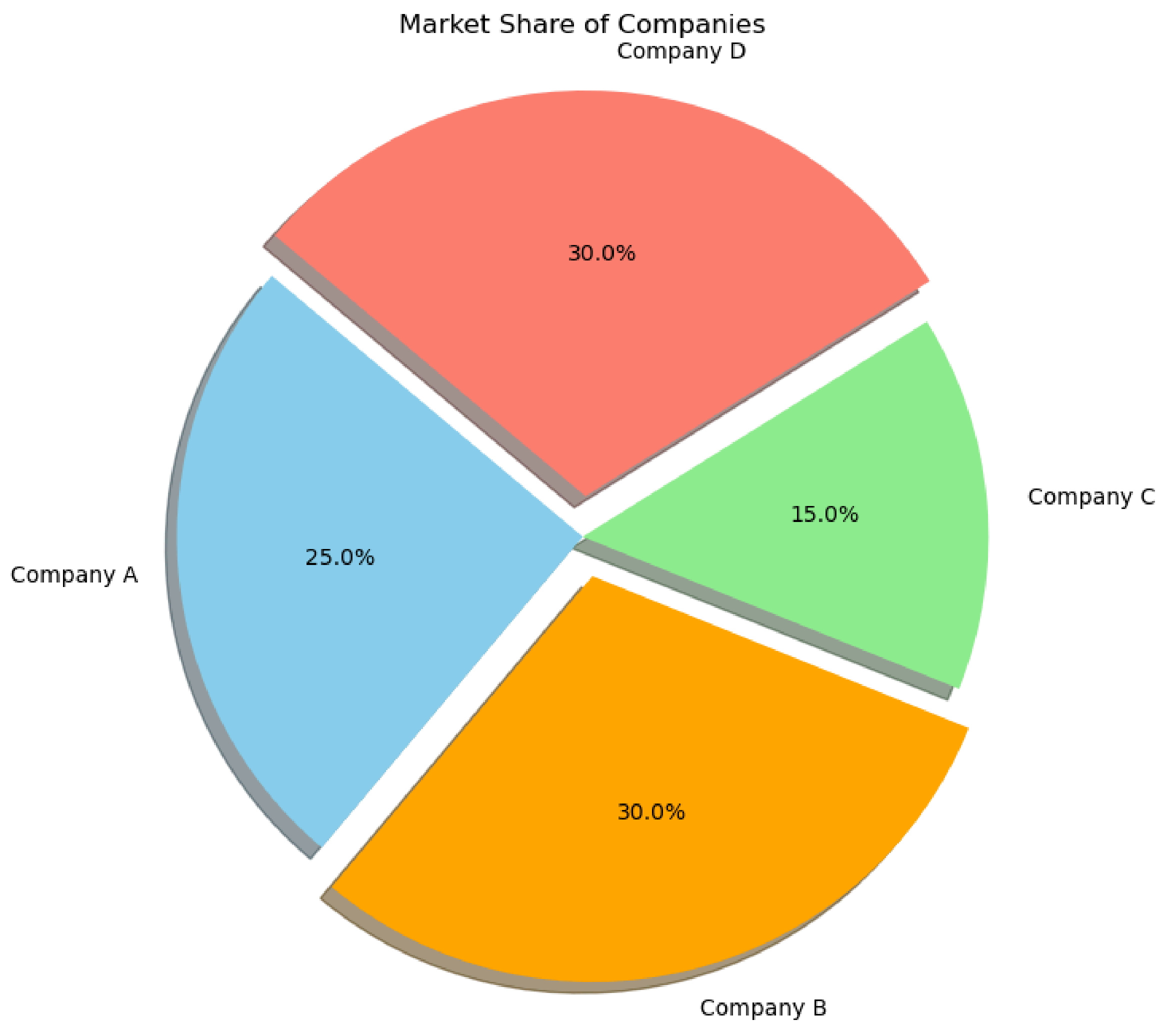
8)Boxplot Using DataFrame: Given a DataFrame of student exam scores for three subjects, visualize the data distribution with a boxplot.

```
In [5]: import pandas as pd
        import matplotlib.pyplot as plt
        import numpy as np
        data = {
            'Math': np.random.randint(60, 100, 30),
            'Science': np.random.randint(55, 95, 30),
            'English': np.random.randint(65, 100, 30)
        }

        df = pd
        plt.figure(figsize=(8, 6))
        df.boxplot(column=['Math', 'Science', 'English'])
        plt.title('Boxplot of Student Exam Scores')
        plt.ylabel('Scores')
        plt.grid(True)
        plt.show()
```



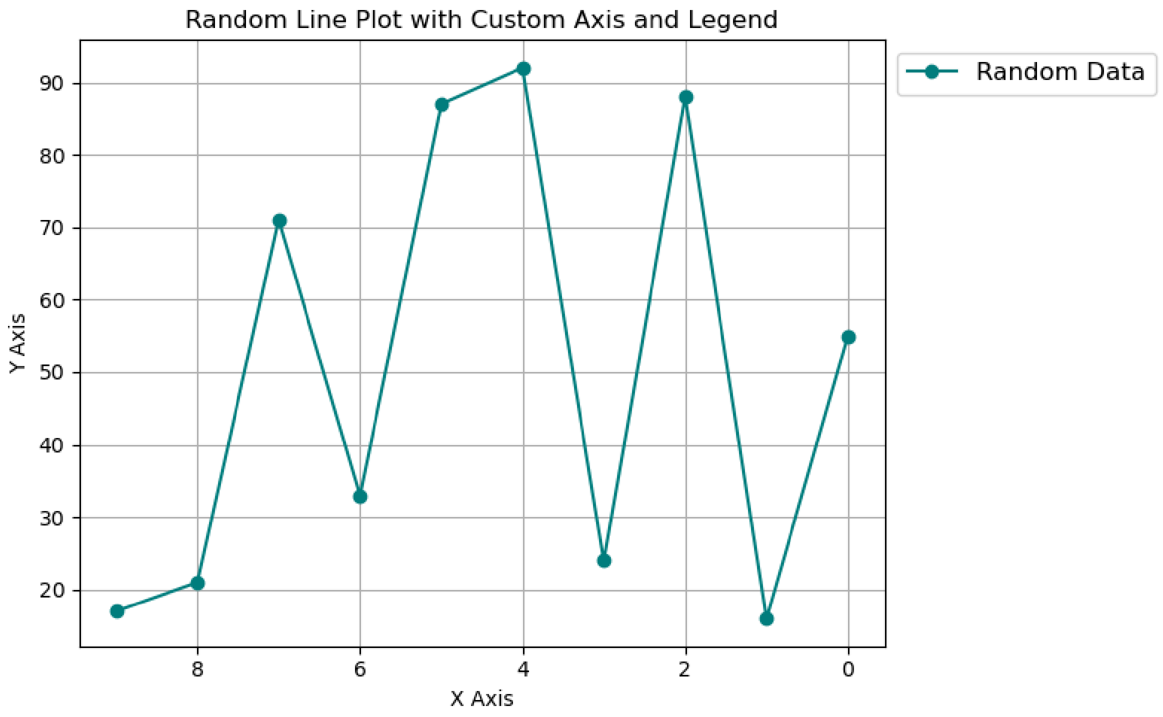
9)Pie Chart with Explode: Create a pie chart showing the market share of four companies. Explode the largest segment and display percentage values on the chart.

```
In [6]: import matplotlib.pyplot as plt
        companies = ['Company A', 'Company B', 'Company C', 'Company D']
        market_share = [25, 30, 15, 30]
        plt.figure(figsize=(8, 8))
        plt.pie(market_share, labels=companies, explode=explode, autopct='%1.1f%%',
                shadow=True, startangle=140, colors=['skyblue', 'orange', 'lightgreen',
        plt.axis('equal')
        plt.title('Market Share of Companies')

        plt.show()
```



0)Axis and Legend Customization: For a random line plot, reverse the x-axis, move the legend outside the plot, and increase legend font size.

In [7]:
```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0, 10)
y = np.random.randint(10, 100, size=10)
plt.figure(figsize=(8, 5))
plt.plot(x, y, label='Random Data', color='teal', marker='o')
plt.gca().invert_xaxis()
plt.legend(loc='upper left', bbox_to_anchor=(1, 1), fontsize=12)

plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.title('Random Line Plot with Custom Axis and Legend')

plt.tight_layout()

plt.grid(True)
plt.show()
```


Random Line Plot with Custom Axis and Legend
Random Data
Y Axis
X Axis
90
80
70
60
50
40
30
20
8
6
4
2
0

```
In [ ]:
```