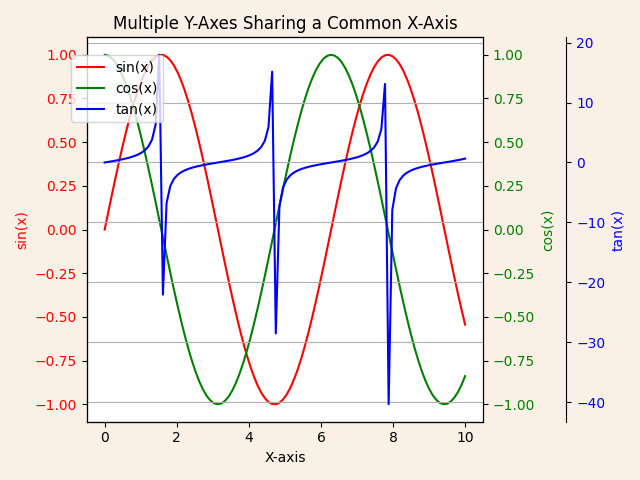
**Task-5:** **Multiple Y-Axes Sharing a Common X-Axis Plot**

1. **Description**

Create a plot with multiple y-axes sharing a common x-axis.

1. **Output**

****

1. **Algorithm Used in Task**

This code creates a plot with a shared x-axis and three distinct y-axes for sin(x), cos(x), and tan(x). Each function is displayed in a unique color for easy differentiation. The y-axes are customized to ensure they do not overlap, making the visualization clear and insightful. The plot is styled with a background color, gridlines, and legends, then saved as a PNG file for future reference.

**Explanation of Libraries and Their Usage:**

1. **Matplotlib (import matplotlib.pyplot as plt)**
   * A Python library for creating static, animated, and interactive visualizations.
   * **Key Functions Used:**
     + plt.subplots(): Creates a figure and a single axis for plotting.
     + ax.twinx(): Creates additional y-axes that share the same x-axis.
     + plt.savefig(): Saves the plot as a PNG file.
2. **NumPy (import numpy as np)**
   * A library for numerical computations.
   * **Key Functions Used:**
     + np.linspace(): Generates an array of evenly spaced values.
     + np.sin(x), np.cos(x), and np.tan(x): Compute trigonometric functions.

**Approach**

1. Generate xxx-values using np.linspace() and compute corresponding yyy-values for sine, cosine, and tangent functions.
2. Create a single x-axis shared by three separate y-axes to display sin(x), cos(x), and tan(x) on different scales.
3. Style the plot with custom colors, legends, gridlines, and labels for clarity.
4. Save the resulting visualization as a PNG file and display it.

**Algorithm:**

1. Import necessary libraries (matplotlib.pyplot and numpy).
2. Generate the xxx-values using np.linspace(0, 10, 100).
3. Compute y1​=sin(x), y2​=cos(x), and y3​=tan(x).
4. Create a figure and axis using plt.subplots().
5. Plot sin(x) on the first y-axis with a red line and label it.
6. Add a second y-axis using twinx() and plot cos(x) with a green line.
7. Add a third y-axis using twinx() and adjust its position outward using spines['right'].set\_position(('outward', 60)). Plot tan(x) with a blue line.
8. Set titles, labels, and tick colors for each axis.
9. Add a grid and a legend for all plots using fig.legend().
10. Save the plot as multiple\_y\_axes\_plot.png and display it using plt.show().