**Title:** Multiple Y-Axes Sharing a Common X-Axis Plot.

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

**Task 2 Explanation**

**Introduction**

Visualizing data with multiple y-axes on a common x-axis is essential when comparing variables with different units or scales. For instance, when analysing temperature, pressure, and humidity over time, each has a distinct range and unit, making it impractical to combine them into a single y-axis. This report provides a detailed explanation of how to create such plots using **Matplotlib** in Python.

**Objective**

To create a professional-quality plot with:

* A single x-axis shared among all data.
* Multiple y-axes for datasets with different scales.
* Proper alignment, labelling, and readability.

**Implementation**

This visualization is implemented using Python's Matplotlib library, leveraging the twinx functionality for multiple y-axes.

**1. Setting Up the Environment**

Start by importing the necessary libraries:

* **matplotlib.pyplot**: The primary library for creating plots.
* **numpy**: Used to generate sample data.

**2. Generating Data**

Create datasets for visualization:

* **np.linspace(0, 10, 100)** generates 100 equally spaced points between 0 and 10 for the x-axis.
* Each y variable represents a different dataset (sine wave, exponential growth, and logarithmic growth).

**3. Creating the First Y-Axis**

Start by initializing a figure and adding the primary y-axis:

The **ax1** object represents the main axis.

* The sine wave (y1) is plotted in blue ('b-').
* The y-axis label and tick colors are set to blue for clarity.

**4. Adding the Second Y-Axis**

Use **twinx** to create a second y-axis that shares the x-axis:

* The second y-axis is represented by **ax2**.
* The exponential growth dataset (y2) is plotted in red ('r-').
* Its label and ticks are colored red to differentiate it from the first axis.

**5. Adding the Third Y-Axis**

To add a third axis, offset it from the second y-axis:

* **ax3** is created using **twinx**, like the second axis.
* The **spines['right'].set\_position(('outward', 60))** method moves the third y-axis outward by 60 pixels to prevent overlap with the second axis.
* The logarithmic dataset (y3) is plotted in green ('g-'), with its label and ticks also in green.

**6. Adding Legends**

Combine legends from all axes for clarity:

This ensures that the legend displays labels for all datasets.

**7. Final Adjustments**

Optimize the layout and add a title:

* **tight\_layout** adjusts the spacing to prevent overlaps.
* The title describes the plot’s purpose.

**Output**

The resulting plot displays:

* A common x-axis representing the range of values shared by all datasets.
* Three distinct y-axes with appropriate labels and tick colors.
* A legend to identify the datasets.