3b. Write a Python program to illustrate Stacked Area Plot and Subplot using Matplotlib.

1. Introduction.

- This Python program demonstrates how to create a **stacked area plot** and **line plot** side-by-side using Matplotlib. It visualizes the monthly sales of three different products over six months.
- The stacked area plot highlights the cumulative sales of all products, while the line plot offers a comparison of individual product sales.
- These visualizations help analyze trends and compare performance effectively. Basic color names and clear labels are used for ease of interpretation.

Stacked Area Plot

A **stacked area plot** is a type of plot that visualizes multiple data series stacked on top of each other. It shows the cumulative values of different categories over time or another continuous variable. Each category is represented by an area under a line, and the total height of the stacked areas at any given point reflects the combined value of all categories. This type of plot is useful for displaying part-to-whole relationships over time, as it allows you to see both the total and the individual contributions of each category.

Subplot

A **subplot** refers to a smaller plot that is part of a larger figure containing multiple plots. It allows you to display multiple visualizations side by side or in a grid within the same figure, enabling comparisons of different datasets or perspectives on the same data. In Matplotlib, subplots are created using the `plt.subplots()` function, which divides the figure into a grid of rows and columns, where each section is an independent plot (subplot).

2. Program code.

```
# Import necessary libraries

import matplotlib.pyplot as plt
import numpy as np

# Sample data for stacked plot
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
```

```
product_A_sales = [100, 120, 130, 150, 160, 180]
product_B_sales = [80, 100, 90, 110, 120, 130]
product_C_sales = [60, 70, 60, 80, 90, 100]

# Stack the data
product_A = np.array(product_A_sales)
product_B = np.array(product_B_sales)
product_C = np.array(product_C_sales)

# Create a figure with 1 row and 2 columns of subplots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 6))

# 1st subplot - Stacked Area Plot
ax1.stackplot(months, product_A, product_B, product_C,
              labels=['Product A', 'Product B', 'Product C'],
              colors=['red', 'green', 'blue'])

ax1.set_title('Stacked Area Plot')
ax1.set_xlabel('Months')
ax1.set_ylabel('Sales')
ax1.legend(loc='upper left')

# 2nd subplot - Line Plot (Sales comparison by product)
ax2.plot(months, product_A, label='Product A', marker='o', color='red')
ax2.plot(months, product_B, label='Product B', marker='s',
color='green')
ax2.plot(months, product_C, label='Product C', marker='^', color='blue')

ax2.set_title('Sales Comparison (Subplot)')
ax2.set_xlabel('Months')
ax2.set_ylabel('Sales')
ax2.legend(loc='upper left')

# Adjust layout for clarity
plt.tight_layout()
plt.show()
```

3. Explanation of the code

1. ***Importing the necessary libraries***

```
import matplotlib.pyplot as plt
import numpy as np
```

- matplotlib.pyplot: This is the sub-library of Matplotlib that is used for creating visualizations. It's commonly imported as plt for simplicity.
- numpy: A powerful library for numerical computing. Here, it is used to create arrays (np.array), which are more efficient and versatile for handling data in numerical computations.

2. ***Sample data for the stacked plot***

```
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
product_A_sales = [100, 120, 130, 150, 160, 180]
product_B_sales = [80, 100, 90, 110, 120, 130]
product_C_sales = [60, 70, 60, 80, 90, 100]
```

- **months**: A list representing the x-axis labels (months from January to June).
- **product_A_sales, product_B_sales, product_C_sales**: These lists store sales data for three products (A, B, and C) over the six months. Each element in these lists corresponds to the sales for a particular month.

3. ***Convert the data into NumPy arrays***

```
product_A = np.array(product_A_sales)
product_B = np.array(product_B_sales)
product_C = np.array(product_C_sales)
```

- These lines convert the sales data (which are currently in Python lists) into NumPy arrays. NumPy arrays allow for faster computations and are compatible with many plotting functions in matplotlib. In this case, it makes it easier to stack and plot the data.

4. ***Create a figure with subplots***

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 6))
```

- plt.subplots(1, 2): This creates a figure with 1 row and 2 columns of subplots, meaning two plots will be displayed side-by-side.
- fig: Represents the entire figure (a container for the plots).

- ax1, ax2: These are the axes for the individual subplots. ax1 corresponds to the first plot (left), and ax2 corresponds to the second plot (right).
- figsize=(10, 6): Specifies the overall size of the figure, with a width of 10 inches and a height of 6 inches.

*5. **First subplot - Stacked Area Plot***

```
ax1.stackplot(months, product_A, product_B, product_C,
          labels=['Product A', 'Product B', 'Product C'],
          colors=['red', 'green', 'blue'])
```

- **ax1.stackplot()**: Creates a stacked area plot.
    - **months**: Represents the x-axis values (the months).
    - **product_A, product_B, product_C**: These represent the y-axis values for the three products. The plot will stack the values on top of each other to show the cumulative sales of all three products for each month.
    - **labels=['Product A', 'Product B', 'Product C']**: These labels correspond to each product and will be shown in the legend.
    - **colors=['red', 'green', 'blue']**: Basic color names for each product's area (red for Product A, green for Product B, and blue for Product C).

*6. **Add title, labels, and legend to the stacked area plot***

```
ax1.set_title('Stacked Area Plot')
ax1.set_xlabel('Months')
ax1.set_ylabel('Sales')
ax1.legend(loc='upper left')
```

- **set_title('Stacked Area Plot')**: Sets the title of the first subplot.
- **set_xlabel('Months')**: Labels the x-axis as "Months".
- **set_ylabel('Sales')**: Labels the y-axis as "Sales".
- **legend(loc='upper left')**: Displays the legend in the upper-left corner, showing which color corresponds to which product.

*7. **Second subplot - Line Plot (Sales comparison by product)***

```
ax2.plot(months, product_A, label='Product A', marker='o', color='red')
ax2.plot(months, product_B, label='Product B', marker='s', color='green')
```

ax2.plot(months, product_C, label='Product C', marker='^', color='blue')

- **ax2.plot()**: Plots individual line graphs for each product.
  - **months**: The x-axis values (months).
  - **product_A, product_B, product_C**: The y-axis values (sales data for each product).
  - **label**: Specifies the label for each product in the legend.
  - **marker**: Adds markers for each data point in the plot ('o' for Product A, 's' for Product B, and '^' for Product C).
  - **color**: Sets the color for each product's line (red, green, and blue).

8. *Add title, labels, and legend to the line plot*

ax2.set_title('Sales Comparison (Subplot)')

ax2.set_xlabel('Months')

ax2.set_ylabel('Sales')

ax2.legend(loc='upper left')

- **set_title('Sales Comparison (Subplot)')**: Sets the title for the second subplot (the line plot).
- **set_xlabel('Months')**: Labels the x-axis as "Months".
- **set_ylabel('Sales')**: Labels the y-axis as "Sales".
- **legend(loc='upper left')**: Displays the legend in the upper-left corner, similar to the first plot.

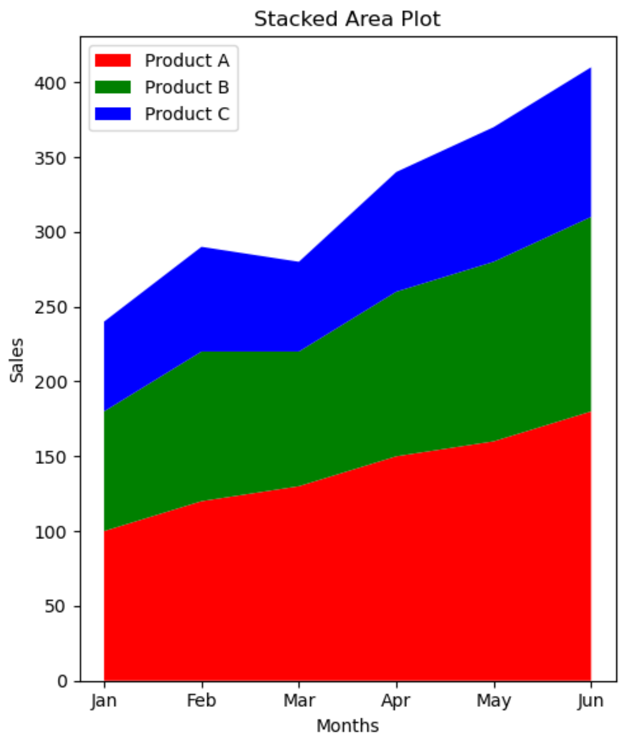
9. *Adjust layout for clarity and display the plot*

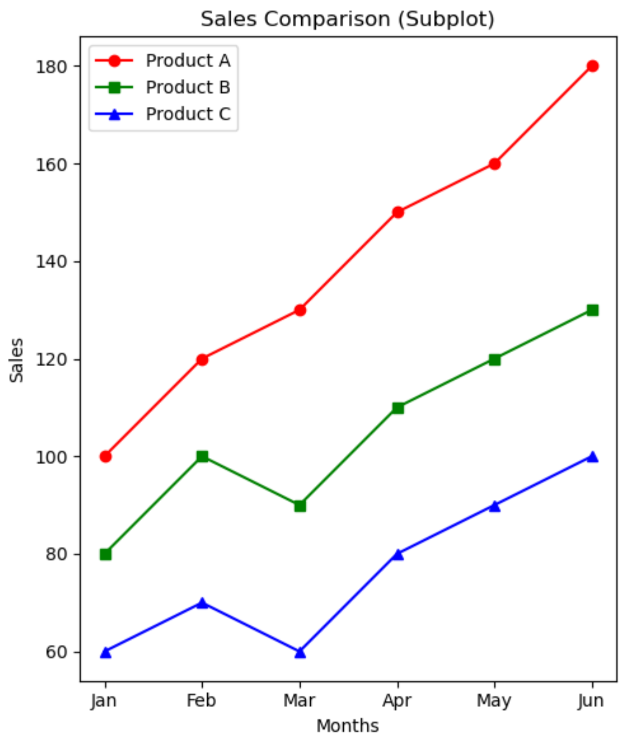
plt.tight_layout()

plt.show()

- **plt.tight_layout()**: Automatically adjusts the spacing between subplots for better clarity, ensuring that labels and titles don't overlap.
- **plt.show()**: Finally, this function displays the figure with both subplots.

Output


Stacked Area Plot
Product A
Product B
Product C
Sales
Months
0
50
100
150
200
250
300
350
400
Jan
Feb
Mar
Apr
May
Jun


Sales Comparison (Subplot)
Product A
Product B
Product C
Sales
Months
60
80
100
120
140
160
180
Jan
Feb
Mar
Apr
May
Jun