Ex: Simulate Dice Throws and Visualize Distribution

```
import numpy as np

# ------------------------------
# 1. Parameters
# ------------------------------
num_throws = 1000   # Number of throws
sides = 6           # Number of sides on each dice

# ------------------------------
# 2. Simulate Two Dice Throws
# ------------------------------
# np.random.randint(low, high, size) generates random integers
dice1 = np.random.randint(1, sides + 1, size=num_throws)
dice2 = np.random.randint(1, sides + 1, size=num_throws)

# Calculate sum of two dice
sums = dice1 + dice2

# ------------------------------
# 3. Count Frequencies of Sums
# ------------------------------
min_sum = 2
max_sum = 2 * sides
freq = {i: 0 for i in range(min_sum, max_sum + 1)}

for total in sums:
    freq[total] += 1

# ------------------------------
# 4. Display Frequency Table
# ------------------------------
print("\nSum | Frequency | Percentage")
print("----------------------------")
for value in range(min_sum, max_sum + 1):
    percentage = (freq[value] / num_throws) * 100
    print(f"{value:^3} | {freq[value]:^9} | {percentage:>7.2f}%")

# ------------------------------
# 5. ASCII Histogram
# ------------------------------
print("\nSum Distribution (ASCII Histogram)")
for value in range(min_sum, max_sum + 1):
    bar = '*' * (freq[value] * 50 // num_throws)
```

```
    print(f"{value:2}: {bar}")
```

Output:

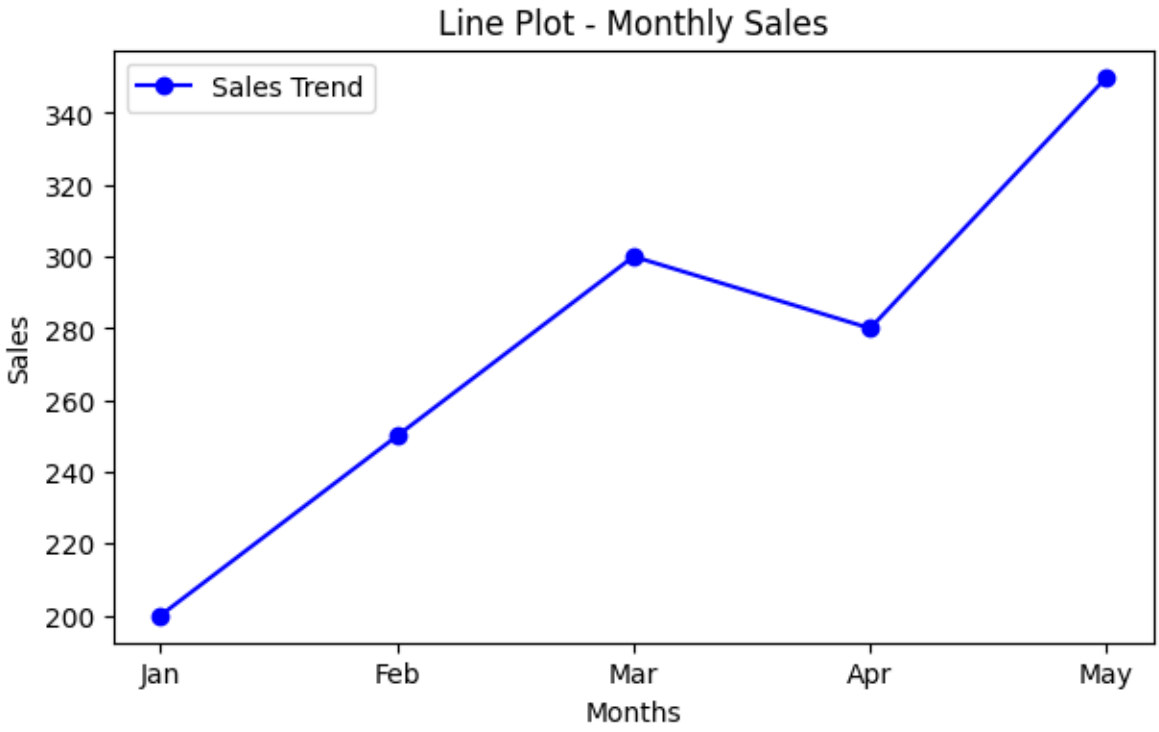
```
Sum | Frequency | Percentage
----------------------------
 2 |    27     |   2.70%
 3 |    65     |   6.50%
 4 |    67     |   6.70%
 5 |    125    |  12.50%
 6 |    130    |  13.00%
 7 |    162    |  16.20%
 8 |    136    |  13.60%
 9 |    109    |  10.90%
10 |    89     |   8.90%
11 |    55     |   5.50%
12 |    35     |   3.50%

Sum Distribution (ASCII Histogram)
 2: *
 3: ***
 4: ***
 5: ******
 6: ******
 7: ********
 8: ******
 9: *****
10: ****
11: **
12: *
```

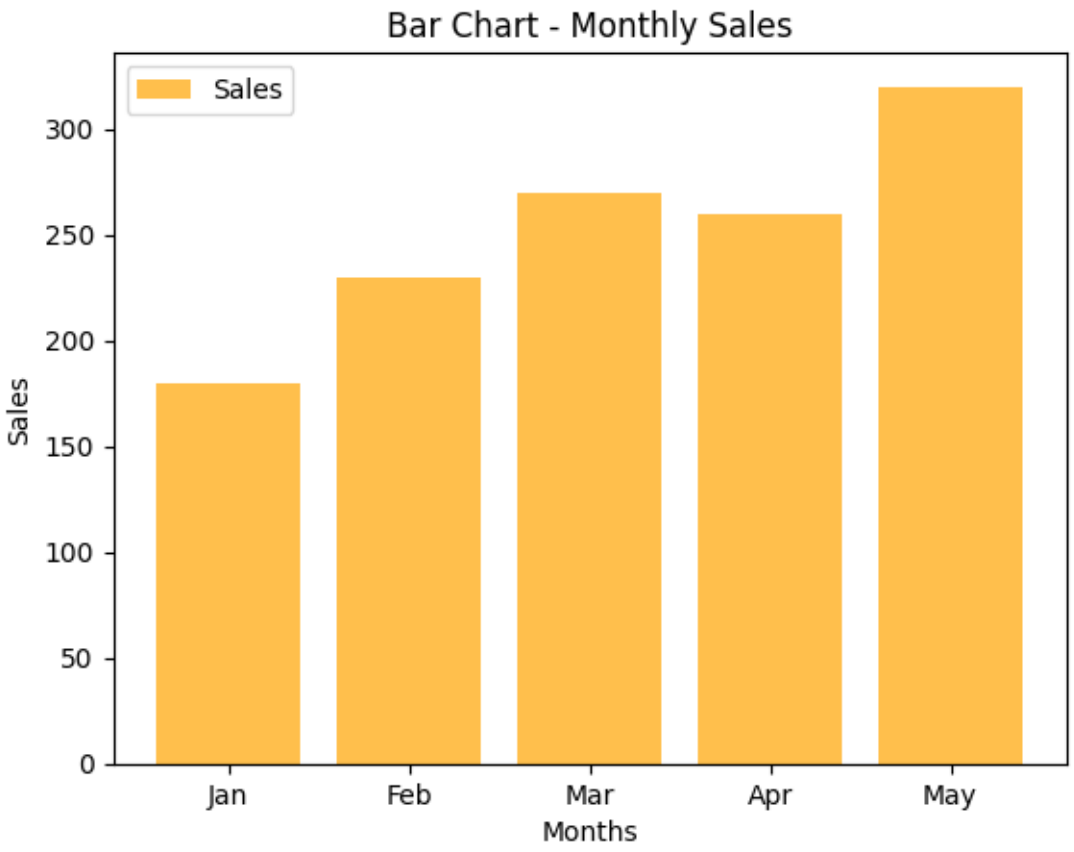
Ex: Matplotlib Basics – Monthly Sales Data

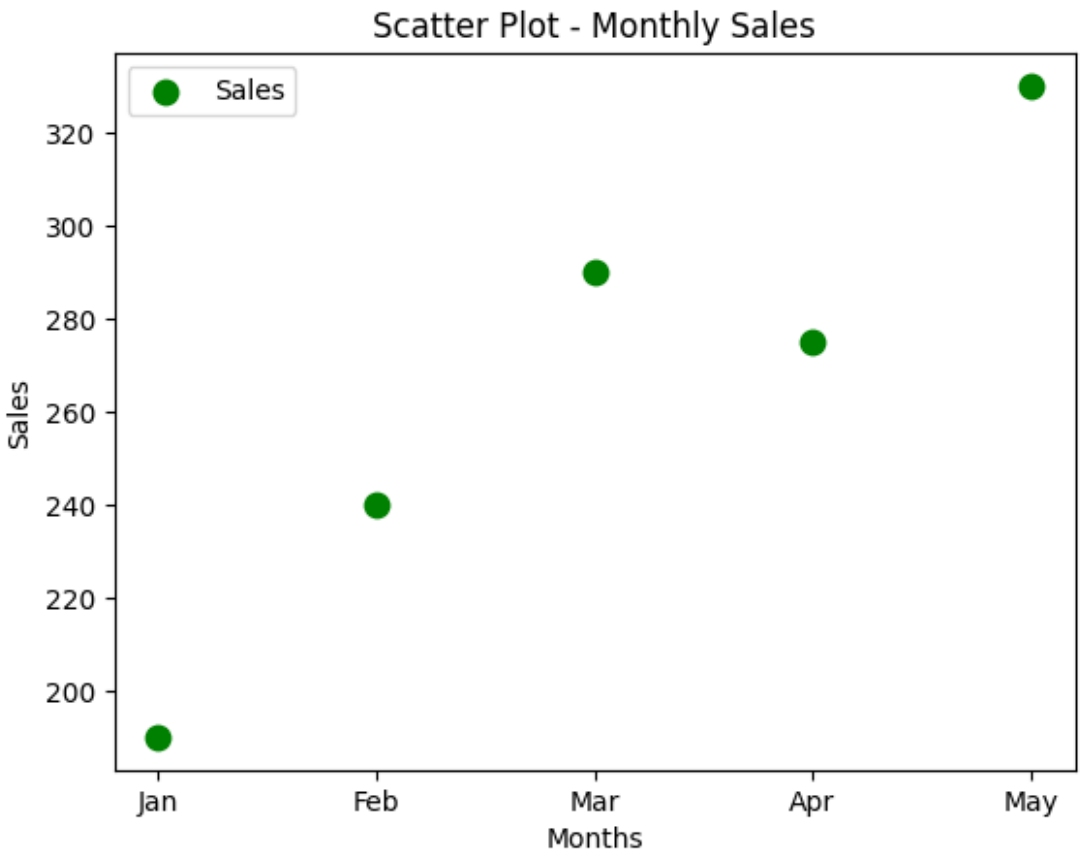
```
import matplotlib.pyplot as plt

# Sample Data
months = ["Jan", "Feb", "Mar", "Apr", "May"]
sales_line = [200, 250, 300, 280, 350]    # For Line Plot
sales_bar = [180, 230, 270, 260, 320]     # For Bar Chart
sales_scatter = [190, 240, 290, 275, 330] # For Scatter Plot

# ------------------------------------------
# 1. Line Plot
# ------------------------------------------
plt.figure(figsize=(7, 4))
plt.plot(months, sales_line, label="Sales Trend", color="blue", marker="o")
plt.title("Line Plot - Monthly Sales")
plt.xlabel("Months")
plt.ylabel("Sales")
plt.legend()
plt.show()

# ------------------------------------------
# 2. Bar Chart
# ------------------------------------------
plt.bar(months, sales_bar, label="Sales", color="orange", alpha=0.7)
plt.title("Bar Chart - Monthly Sales")
plt.xlabel("Months")
plt.ylabel("Sales")
plt.legend()
plt.show()
```

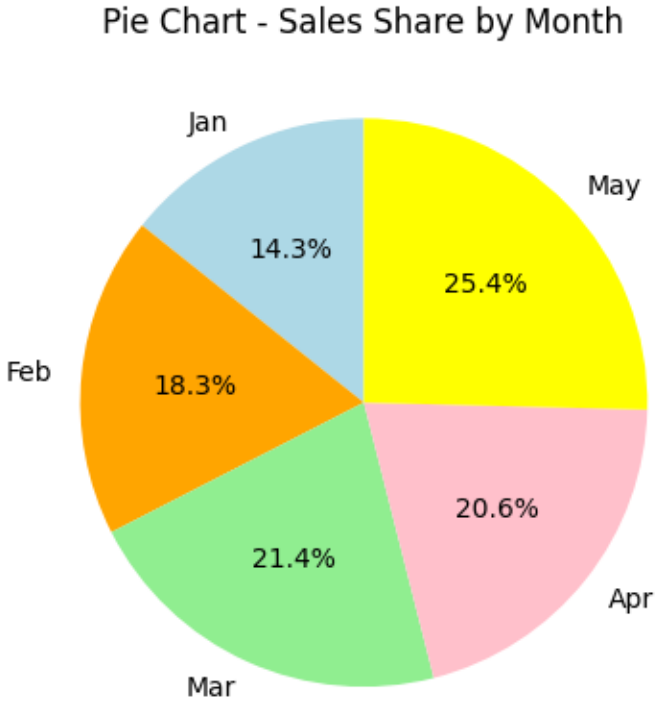
```
# ----------------------------------------
# 3. Scatter Plot
# ----------------------------------------
plt.scatter(months, sales_scatter, label="Sales", color="green", s=80)
plt.title("Scatter Plot - Monthly Sales")
plt.xlabel("Months")
plt.ylabel("Sales")
plt.legend()
plt.show()

# ----------------------------------------
# 4. Pie Chart
# ----------------------------------------
plt.pie(sales_bar, labels=months, autopct="%1.1f%%", startangle=90,
        colors=["lightblue", "orange", "lightgreen", "pink", "yellow"])
plt.title("Pie Chart - Sales Share by Month")
plt.show()

# ----------------------------------------
# 5. Histogram
# ----------------------------------------
# Example: Distribution of daily sales in a single month
daily_sales = [180, 200, 190, 250, 300, 280, 220, 210, 230, 260,
               270, 240, 210, 220, 200, 290, 310, 280, 300, 330]
plt.hist(daily_sales, bins=6, color="purple", alpha=0.7, edgecolor="black")
plt.title("Histogram - Daily Sales Distribution")
plt.xlabel("Sales")
plt.ylabel("Frequency")
plt.show()
```

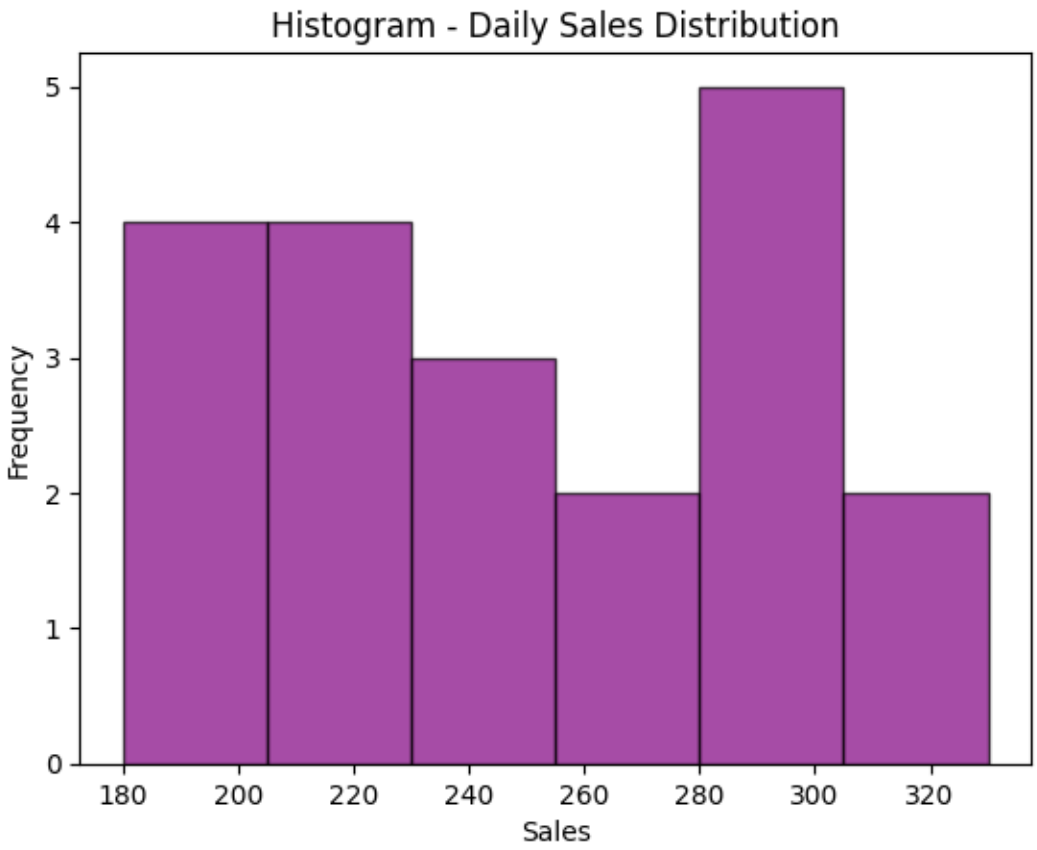
Output


Line Plot - Monthly Sales
Sales Trend
340
320
300
280
260
240
220
200
Sales
Jan
Feb
Mar
Apr
May
Months


Bar Chart - Monthly Sales
Sales
300
250
200
150
100
50
0
Sales
Jan
Feb
Mar
Apr
May
Months


Scatter Plot - Monthly Sales
Sales
320
300
280
260
240
220
200
Sales
Jan
Feb
Mar
Apr
May
Months


Pie Chart - Sales Share by Month
Jan
May
14.3%
25.4%
Feb
18.3%
20.6%
21.4%
Apr
Mar

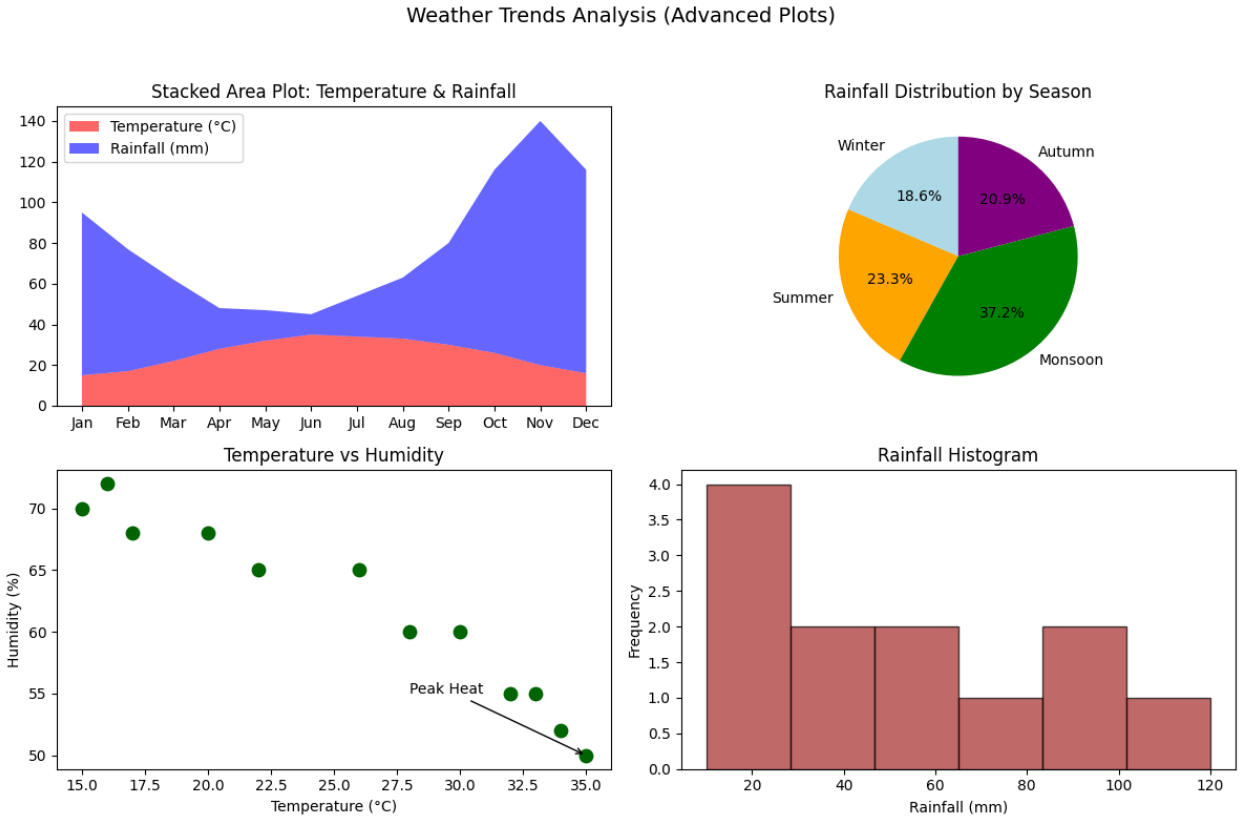

Histogram - Daily Sales Distribution
Frequency
Sales
0
1
2
3
4
5
180
200
220
240
260
280
300
320

Ex.No. Matplotlib Advanced – Weather Trends

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

# Sample Data (Weather Trends)
months = np.array(["Jan", "Feb", "Mar", "Apr", "May", "Jun",
                   "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])
temperature = [15, 17, 22, 28, 32, 35, 34, 33, 30, 26, 20, 16]  # °C
rainfall = [80, 60, 40, 20, 15, 10, 20, 30, 50, 90, 120, 100]   # mm
humidity = [70, 68, 65, 60, 55, 50, 52, 55, 60, 65, 68, 72]     # %

# Create subplots
fig, axs = plt.subplots(2, 2, figsize=(12, 8))

# ------------------------------------------
# 1. Stacked Area Plot – Temperature & Rainfall
# ------------------------------------------
axs[0, 0].stackplot(months, temperature, rainfall,
                    labels=["Temperature (°C)", "Rainfall (mm)"],
                    colors=["red", "blue"], alpha=0.6)
axs[0, 0].set_title("Stacked Area Plot: Temperature & Rainfall")
axs[0, 0].legend(loc="upper left")

# ------------------------------------------
# 2. Pie Chart – Seasonal Rainfall Distribution
# ------------------------------------------
seasons = ["Winter", "Summer", "Monsoon", "Autumn"]
rainfall_season = [200, 250, 400, 225]
axs[0, 1].pie(rainfall_season, labels=seasons, autopct="%1.1f%%",
```

```
        startangle=90, colors=["lightblue", "orange", "green", "purple"])
axs[0, 1].set_title("Rainfall Distribution by Season")

# ------------------------------------------
# 3. Scatter Plot – Temperature vs Humidity
# ------------------------------------------
axs[1, 0].scatter(temperature, humidity, color="darkgreen", s=80)
axs[1, 0].set_title("Temperature vs Humidity")
axs[1, 0].set_xlabel("Temperature (°C)")
axs[1, 0].set_ylabel("Humidity (%)")
axs[1, 0].annotate("Peak Heat", xy=(35, 50), xytext=(28, 55),
            arrowprops=dict(facecolor="black", arrowstyle="->"))

# ------------------------------------------
# 4. Histogram – Rainfall Distribution
# ------------------------------------------
axs[1, 1].hist(rainfall, bins=6, color="brown", alpha=0.7, edgecolor="black")
axs[1, 1].set_title("Rainfall Histogram")
axs[1, 1].set_xlabel("Rainfall (mm)")
axs[1, 1].set_ylabel("Frequency")

# Adjust layout
plt.suptitle("Weather Trends Analysis (Advanced Plots)", fontsize=14)
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

Output:


Weather Trends Analysis (Advanced Plots)
Stacked Area Plot: Temperature & Rainfall
Temperature (°C)
Rainfall (mm)
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Rainfall Distribution by Season
Winter
Autumn
Summer
Monsoon
18.6%
20.9%
23.3%
37.2%
Temperature vs Humidity
Humidity (%)
Temperature (°C)
Peak Heat
Rainfall Histogram
Frequency
Rainfall (mm)

Ex: Basic Dashboard in Google Colab – Website Analytics

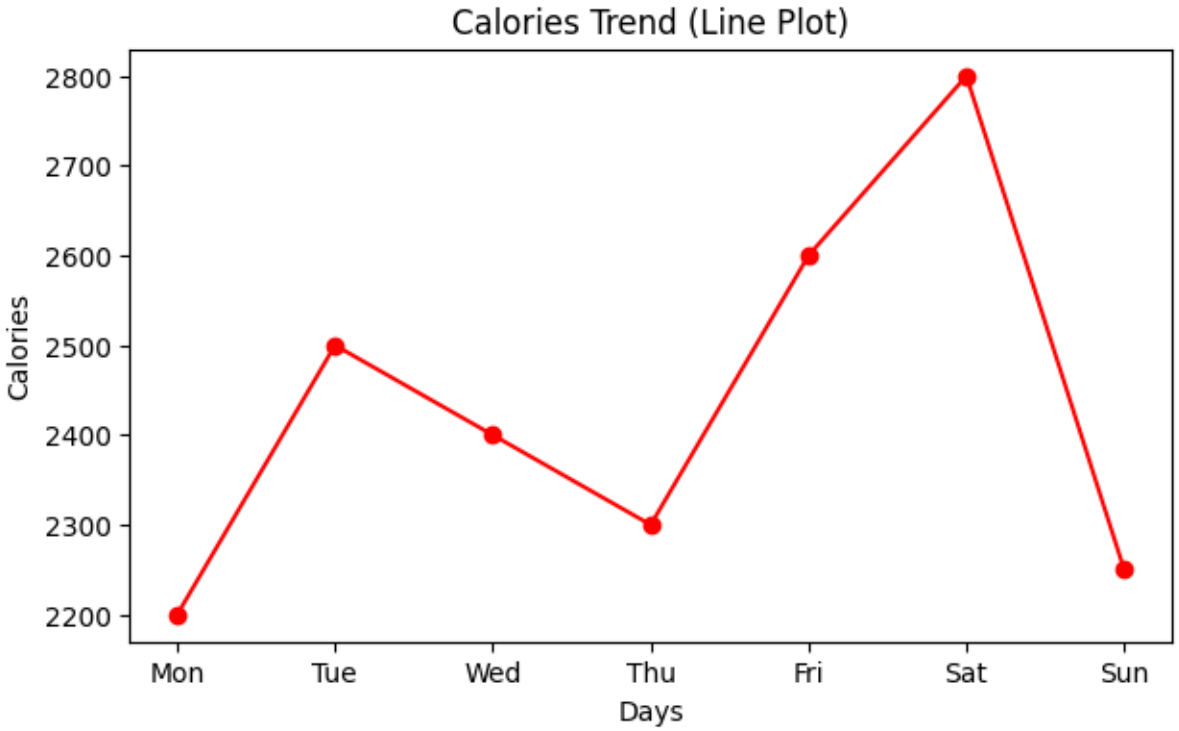
```
import matplotlib.pyplot as plt
import numpy as np
from ipywidgets import interact

# ------------------------------
# 1. Sample Data
# ------------------------------
days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
steps = [5000, 7500, 8000, 6500, 9000, 10000, 7000]
# Steps walked
calories = [2200, 2500, 2400, 2300, 2600, 2800, 2250]
# Calories burned
sleep_hours = [7, 6.5, 8, 7.5, 6, 8, 7]
# Hours slept

# ------------------------------
# 2. Plotting Function
# ------------------------------
def plot_dashboard(plot_type, metric):
    plt.figure(figsize=(7,4))

    # Select data based on metric
    if metric == "Steps":
        data = steps
        ylabel = "Steps"
        color = "blue"
    elif metric == "Calories":
        data = calories
        ylabel = "Calories"
        color = "red"
    else:
        data = sleep_hours
        ylabel = "Hours"
        color = "green"

    # Plot types
    if plot_type == "Line Plot":
        plt.plot(days, data, marker="o", color=color)
        plt.title(f"{metric} Trend (Line Plot)")
        plt.xlabel("Days")
        plt.ylabel(ylabel)
    elif plot_type == "Bar Chart":
        plt.bar(days, data, color=color, alpha=0.7)
```

```
        plt.title(f"{metric} Trend (Bar Chart)")
        plt.xlabel("Days")
        plt.ylabel(ylabel)
    elif plot_type == "Scatter Plot":
        plt.scatter(days, data, color=color, s=80)
        plt.title(f"{metric} Trend (Scatter Plot)")
        plt.xlabel("Days")
        plt.ylabel(ylabel)
    elif plot_type == "Pie Chart":
        plt.pie(data, labels=days, autopct="%1.1f%%", startangle=90)
        plt.title(f"{metric} Share by Day (Pie Chart)")
    elif plot_type == "Histogram":
        plt.hist(data, bins=6, color=color, alpha=0.7, edgecolor="black")
        plt.title(f"{metric} Distribution (Histogram)")
        plt.xlabel(ylabel)
        plt.ylabel("Frequency")

    plt.show()

# ------------------------------
# 3. Interactive Dashboard
# ------------------------------
interact(plot_dashboard,
         plot_type=["Line Plot", "Bar Chart", "Scatter Plot", "Pie Chart", "Histogram"],
         metric=["Steps", "Calories", "Sleep Hours"])
```

Output

plot_type Line Plot

metric Calories



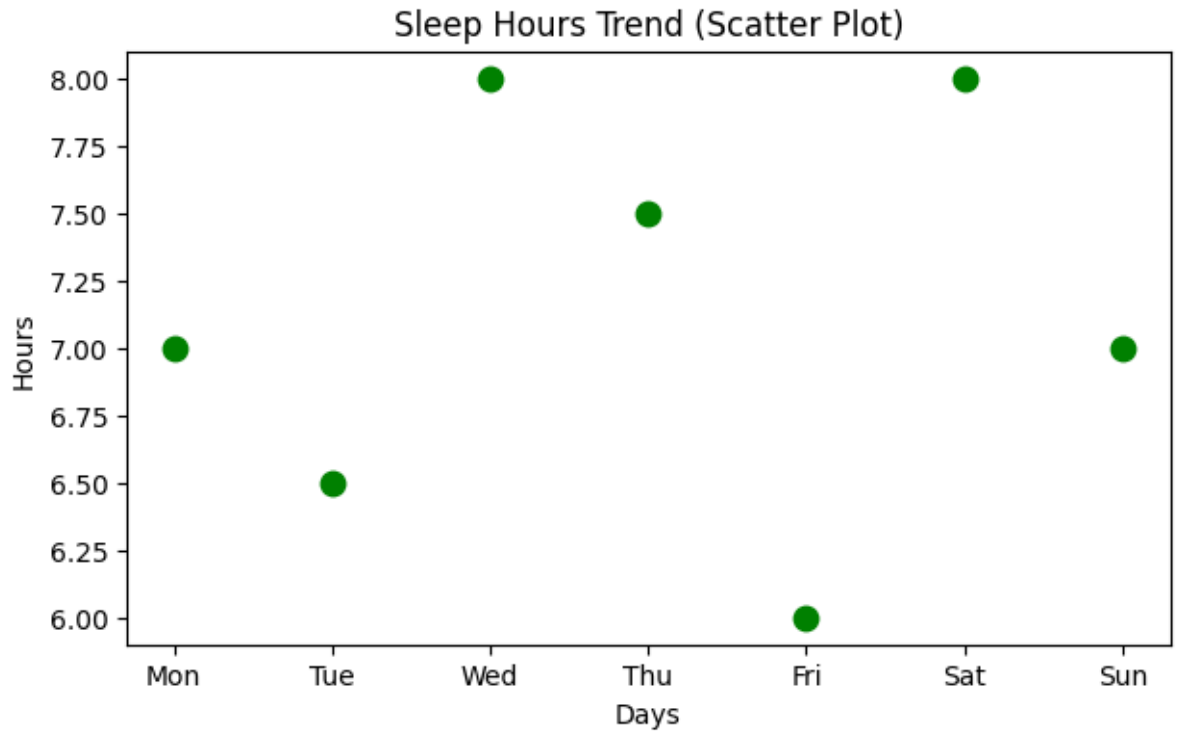
plot_dashboard

def plot_dashboard(plot_type, metric)

<no docstring>

plot_type Scatter Plot

metric Sleep Hours



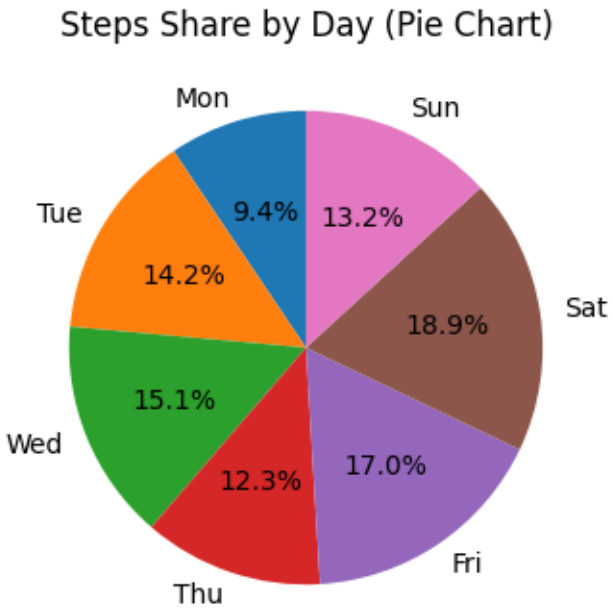
plot_dashboard

def plot_dashboard(plot_type, metric)

<no docstring>

plot_type Pie Chart

metric Steps



plot_dashboard
def plot_dashboard(plot_type, metric)
<no docstring>