VISUALIZATION LIBRAY DOCUMENTATION: MATPLOTLIB & SEABORN

Introduction: Data visualization is the art of turning numbers and statistics into pictures that tell a story. In Python, there's a rich ecosystem of libraries designed to help you create everything from simple line graphs to interactive dashboards. Whether you're exploring a dataset for the first time, presenting findings to a client, or building a live data app, the right visualization tool can make all the difference.

This guide introduces you to the most popular Python visualization libraries—**Matplotlib, Seaborn** explaining what each does best, how they differ, and where to start. By the end, you'll know which library to pick for your project and how to begin crafting clear, compelling visuals that bring your data to life.

MATPLOTLIB

Matplotlib, introduced in 2002 by John Hunter. It is for plotting 2D plots which is built on numpy. A plot of matplotlib contains:

1 Figure

2 Axes

3 Axis

4 Artists

Figure

Figure is the container in which plots are present. It can have a single plot or multiple plots.

Axes

Axes are the plots, and they are artist attached to the figure. A plot can have 2 or 3 axes. set_xlabel(), set_ylabel() are the functions used to set the labels of x and y respectively.

Axis

It is responsible for generating ticks or the limits on the axes.

Artists

The whatever content visible in a figure are the artists.

PYPLOT

Pyplot is a sub library of matplotlib where all the utilities lie under. It has different types of plots including bar graphs, scatter plots, pie charts, histograms, area charts.

We import pyplot from matplotlib as following:

```
1 import matplotlib.pyplot as plt
```

We also import numpy as a support for the matplotlib :

```
import numpy as np
```

Some of the plots in matplotlib:

LINE CHART

Line plots are the basic charts where dot consecutive points are connected by a continuous line.

The default plot() is used for line charts.

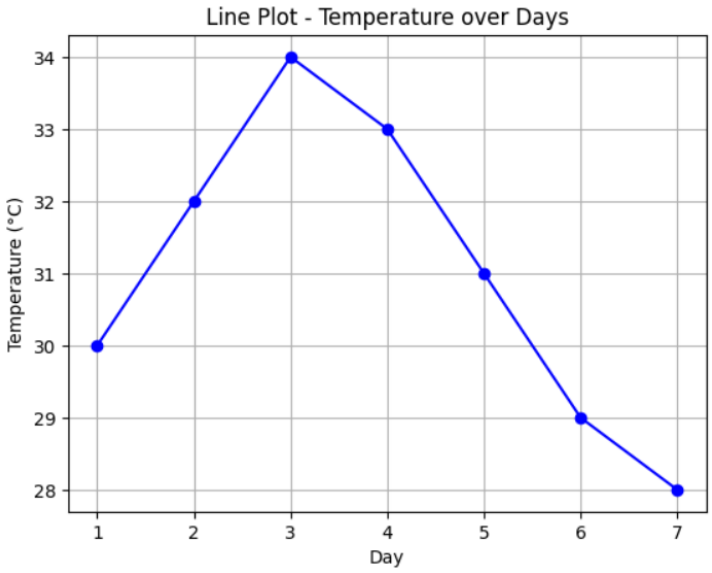
Code snippet:

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

days = np.arange(1, 8)
temperature = [30, 32, 34, 33, 31, 29, 28]

plt.plot(days, temperature, marker='o', color='blue')
plt.title("Line Plot - Temperature over Days")
plt.xlabel("Day")
plt.ylabel("Temperature (°C)")
plt.grid(True)
plt.show()
```

Output:



Description:

np.array() is used to store the temperature and days data in array format.
plot() is used to plot the line chart showing temperature variation over days, with markers for each data point.
title() sets the graph title as *"Line Plot - Temperature over Days"*.
xlabel() and ylabel() label the x-axis as *Day* and the y-axis as *Temperature (°C)* respectively.
show() displays the plotted graph.

BAR GRAPH

Rectangular bars are used in bar chart as the height represent the frequency of a particular element.
bar() is used for plotting bar graphs, it has parameters like x, y, width, color.

bar(x, y, color, width)

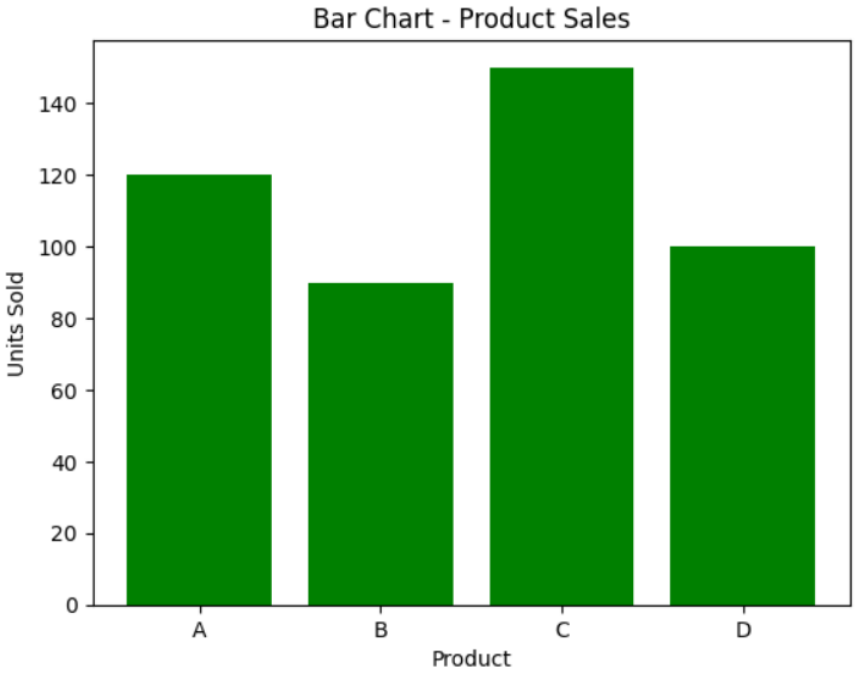
Code snippet:

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

products = ["A", "B", "C", "D"]
sales = [120, 90, 150, 100]

plt.bar(products, sales, color='green')
plt.title("Bar Chart - Product Sales")
plt.xlabel("Product")
plt.ylabel("Units Sold")
plt.show()
```

Output:



Description:
np.array() is used to store product names and corresponding sales data.
bar() is used to create a vertical bar chart showing units sold for each product.
title() sets the chart title as *"Bar Chart - Product Sales"*.
xlabel() and ylabel() label the x-axis as *Product* and the y-axis as *Units Sold* respectively.
show() displays the chart.

HISTOGRAM

Histogram is a type of bar graph where the graph is represented in groups. hist() is used for plotting histogram with parameters x, bins, color, edgecolor.

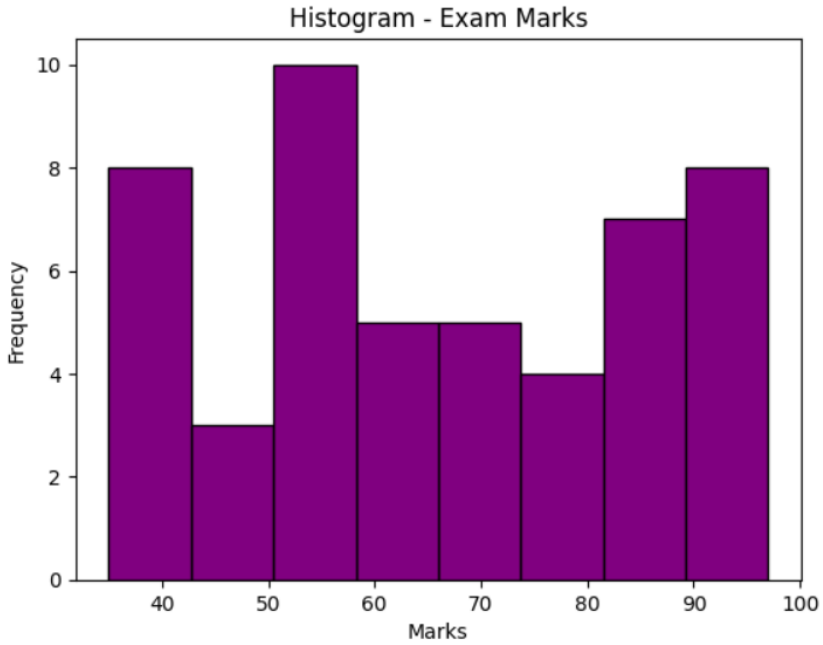
Code snippet:

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

marks = np.random.randint(35, 100, 50)

plt.hist(marks, bins=8, color='purple', edgecolor='black')
plt.title("Histogram - Exam Marks")
plt.xlabel("Marks")
plt.ylabel("Frequency")
plt.show()
```

Output:



Description:

hist() is used to plot the histogram, where parameter x contains the exam marks data.
bins specifies the number of intervals into which the marks are grouped along the x-axis.
color defines the fill color of the rectangular bars, and edgecolor specifies the color of the borders separating the bars.
title() sets the plot title, while xlabel() and ylabel() label the axes.
show() displays the histogram.

SCATTER PLOT

Scatter plot uses dots to depict the relationship between the data. scatter() is used for plotting scatter plot and scatter plot can be done for multiple datasets at the same time by differentiating it with colors. Legends help to achieve this difference.

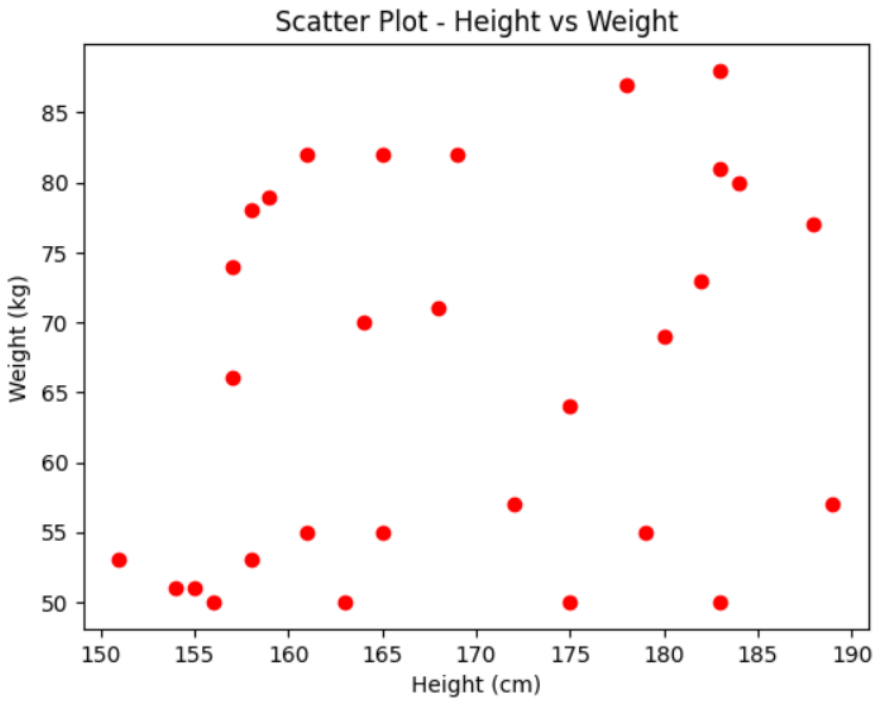
Code snippet:

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

height = np.random.randint(150, 190, 30)
weight = np.random.randint(50, 90, 30)

plt.scatter(height, weight, color='red')
plt.title("Scatter Plot - Height vs Weight")
plt.xlabel("Height (cm)")
plt.ylabel("Weight (kg)")
plt.show()
```

Output:



Description:

x1, y1 represent one dataset and x2, y2 represent another dataset.
scatter() is used to plot the scatter graph for each dataset, where parameters x and y specify the coordinates, and c specifies the color of the points.
legend() adds labels to help differentiate between the datasets in the plot.
title() sets the plot title, while xlabel() and ylabel() label the axes.
show() displays the final scatter plot.

PIECHART

Pie charts are used to plot data of same kind which means the same series of data where the different elements are divide based on their percentage.

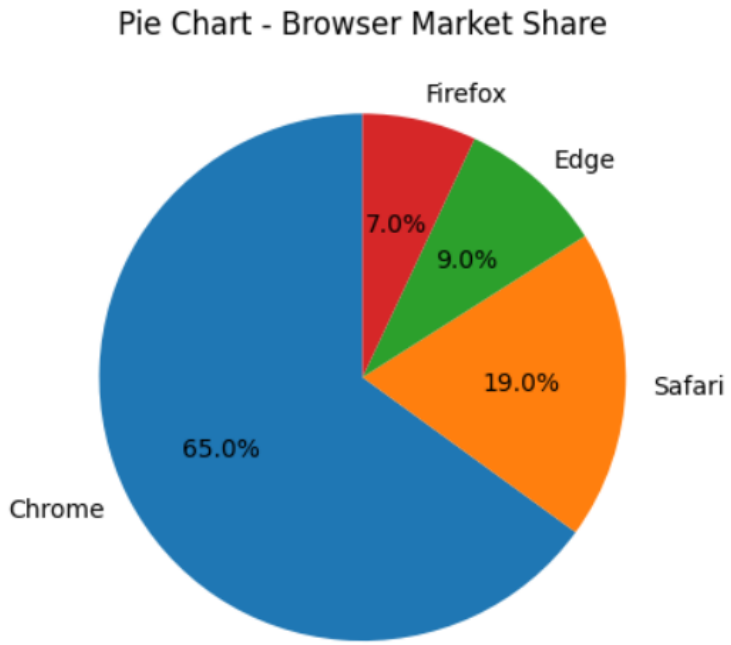
Code snippet:

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

browsers = ["Chrome", "Safari", "Edge", "Firefox"]
market_share = [65, 19, 9, 7]

plt.pie(market_share, labels=browsers, autopct='%1.1f%%', startangle=90)
plt.title("Pie Chart - Browser Market Share")
plt.show()
```

Output:



Description:

A pie chart is a circular graph divided into slices, where each slice represents a proportion of the whole. It's commonly used to show percentage or proportional data, making it easy to compare parts to the total. In Matplotlib, pie charts are created using the plt.pie() function, which allows customization of colors, labels, percentages, and slice separation for emphasis.

SEABORN

Seaborn is a visualization library in python which is built on top of matplotlib. It is advanced than matplotlib and have different features which are default like style, color palette. Seaborn has different categories of plots like relational plot, categorical plot, distribution plot, regression plot, matrix plot.

Let us see the different plots in each category.

1 Relational Plots:

- scatterplot()
- lineplot()
- relplot()

2 Categorical Plots:

- barplot()
- countplot()
- boxplot()
- violinplot()
- swarmplot()
- pointplot()
- catplot()

3 Distribution Plots:

- histplot()
- kdeplot()
- rugplot()
- distplot()

4 Relational Plots:

- regplot()
- lmplot()

5 Matrix Plots:

- heatmap()
- clustermap()

Here are some sample codes for some of the graphs.

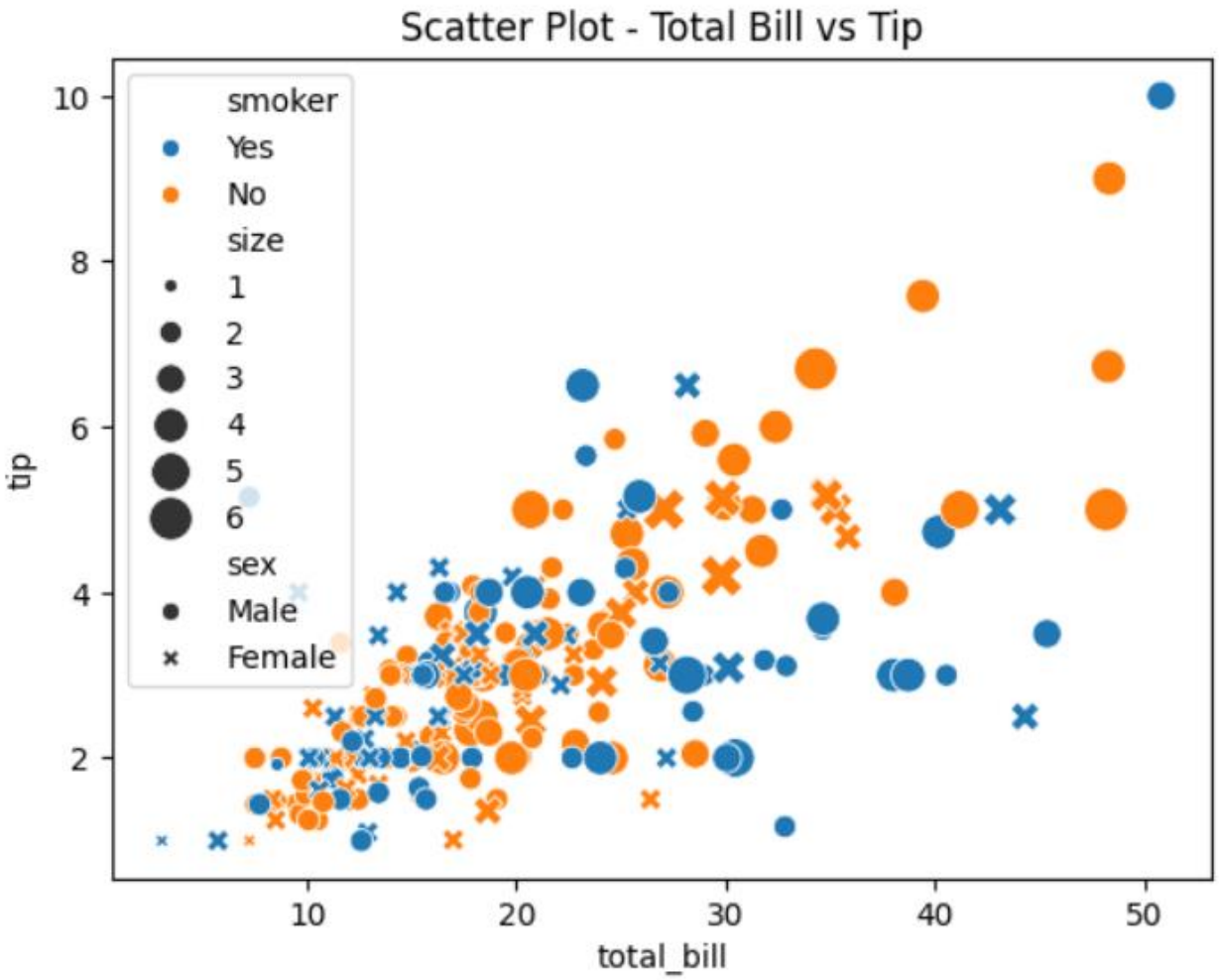
Scatter plot:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

tips = sns.load_dataset("tips")

sns.scatterplot(data=tips, x="total_bill", y="tip", hue="smoker", style="sex", size="size", sizes=(20,200))
plt.title("Scatter Plot - Total Bill vs Tip")
plt.show()
```

Output:



Description:

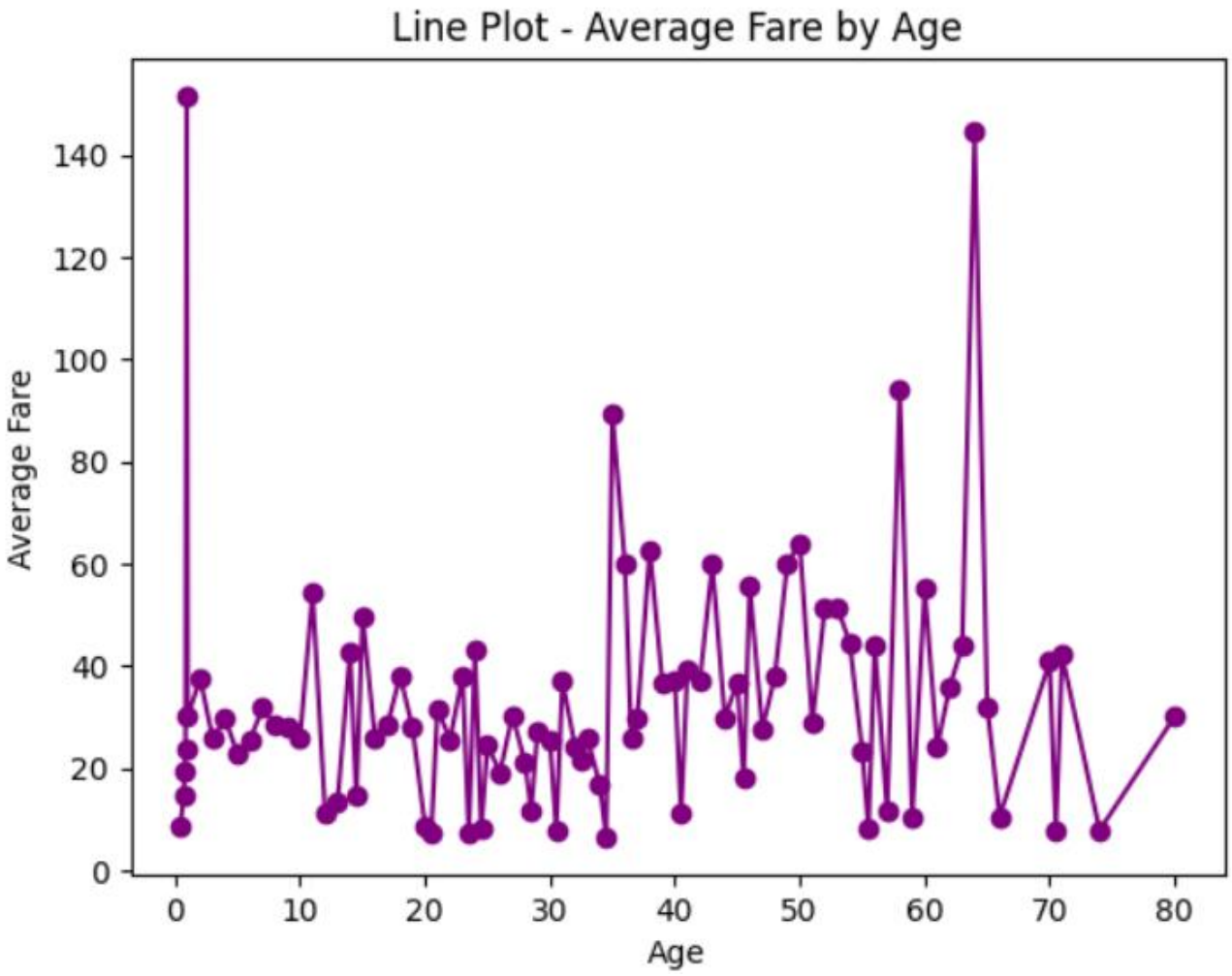
scatterplot() is used to plot total_bill on the x-axis and tip on the y-axis.
The parameter hue is set to the "sex" column, which assigns different colors to Male and Female categories.
The size parameter maps the "size" column to the marker sizes, and style can be used to vary marker shapes if needed.
legend() displays the mapping for hue, size, and style to help interpret the plot.
title() sets the plot title, and xlabel()/ylabel() label the axes.
show() displays the scatter plot.

LINE PLOT

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

titanic = sns.load_dataset("titanic")
avg_fare_age = titanic.groupby("age")["fare"].mean().dropna()
avg_fare_age.plot(kind="line", marker="o", color="purple", title="Line Plot - Average Fare by Age")
plt.xlabel("Age")
plt.ylabel("Average Fare")
plt.show()
```

Output:



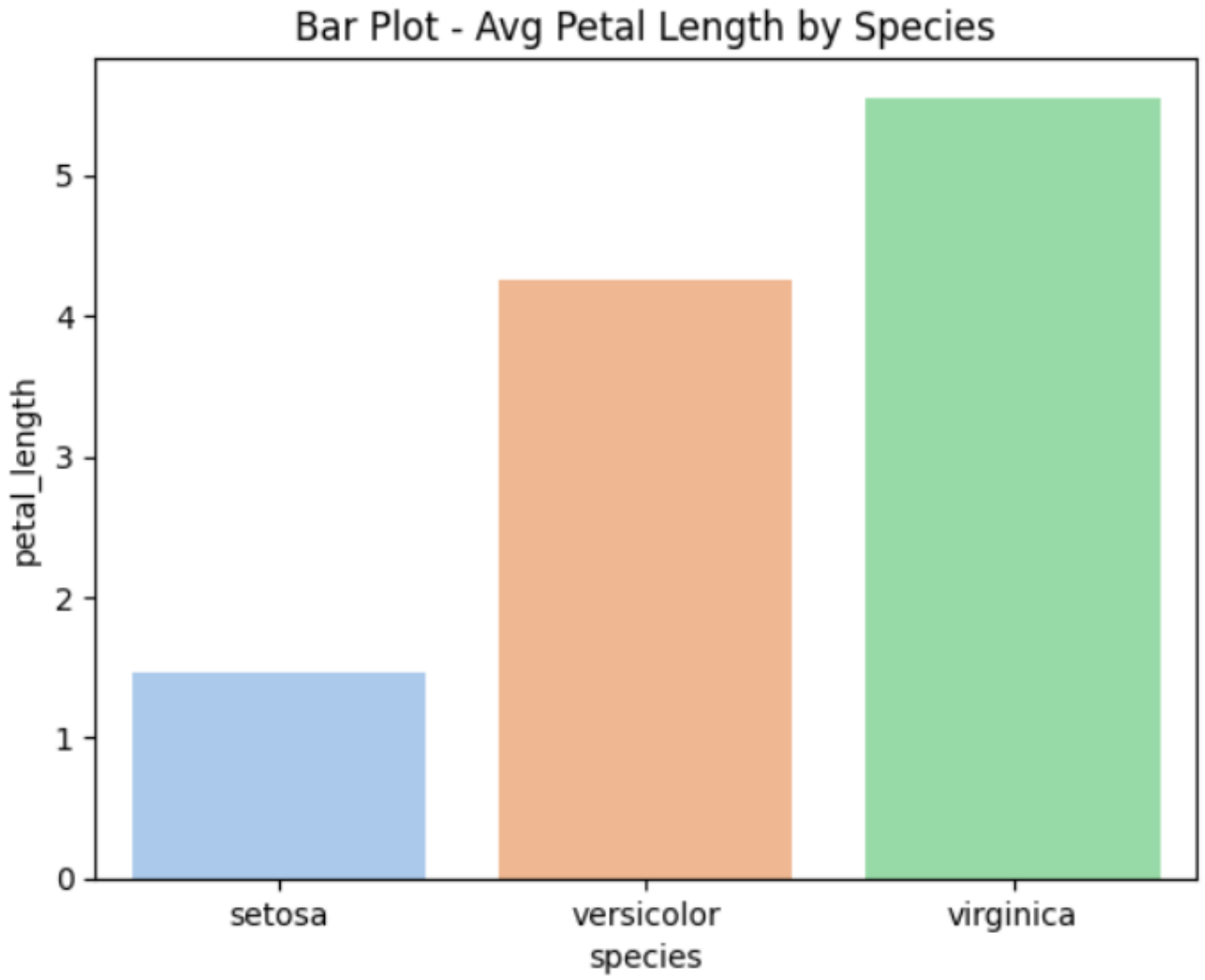
BAR PLOT

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset("iris")

sns.barplot(data=iris, x="species", y="petal_length", ci=None, palette="pastel")
plt.title("Bar Plot - Avg Petal Length by Species")
plt.show()
```

Output:

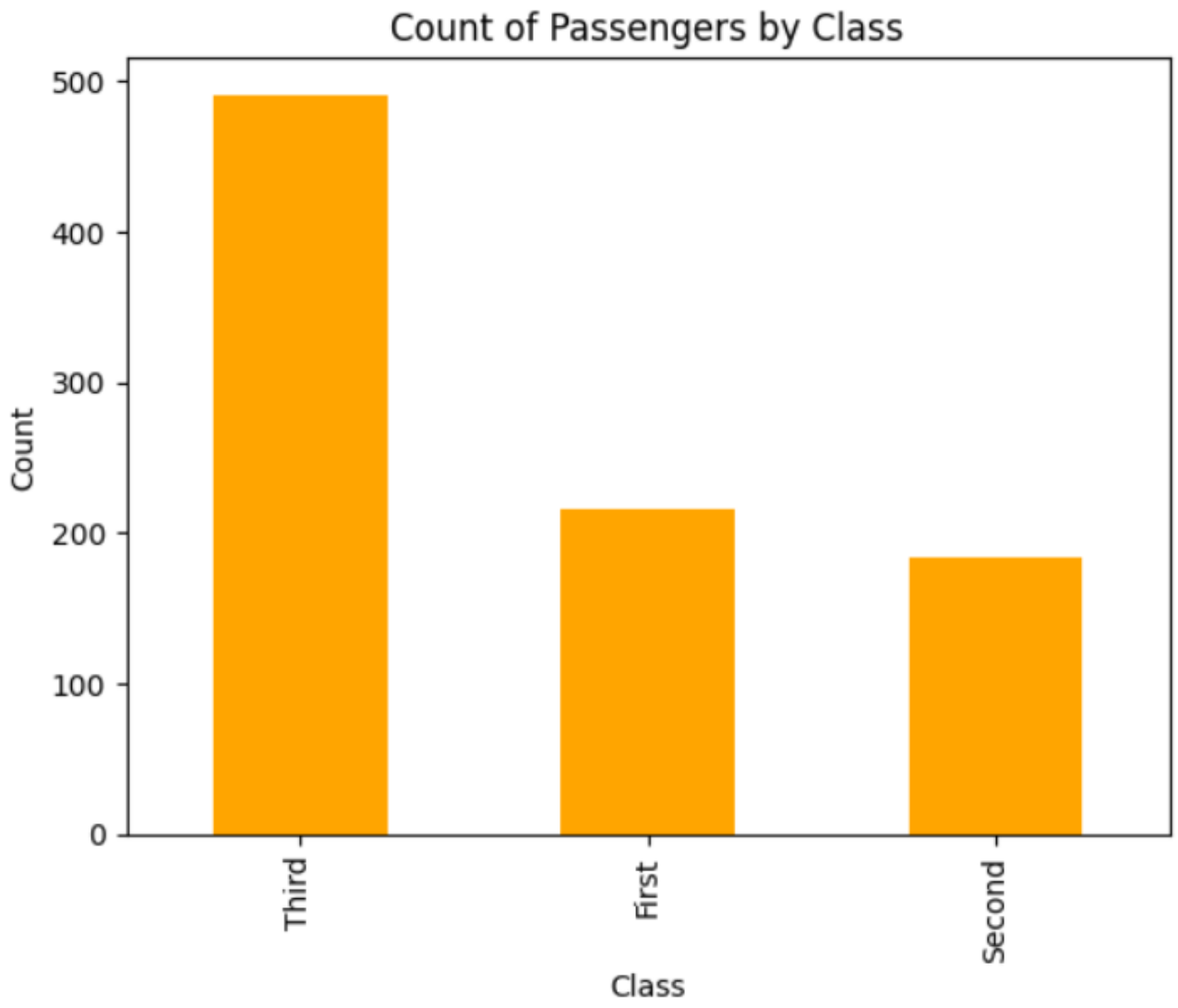


COUNT PLOT

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

titanic["class"].value_counts().plot(kind="bar", color="orange", title="Count of Passengers by Class")
plt.xlabel("Class")
plt.ylabel("Count")
plt.show()
```

Output:



Description:

countplot() is used to display the count of passengers in each class category from the dataset.
The x parameter is set to "class" to plot passenger classes on the x-axis, and color is set to orange for the bars.
The y-axis represents the count of passengers in each class.
title() sets the plot title, and xlabel()/ylabel() label the axes.
show() displays the count plot.

BOX PLOT

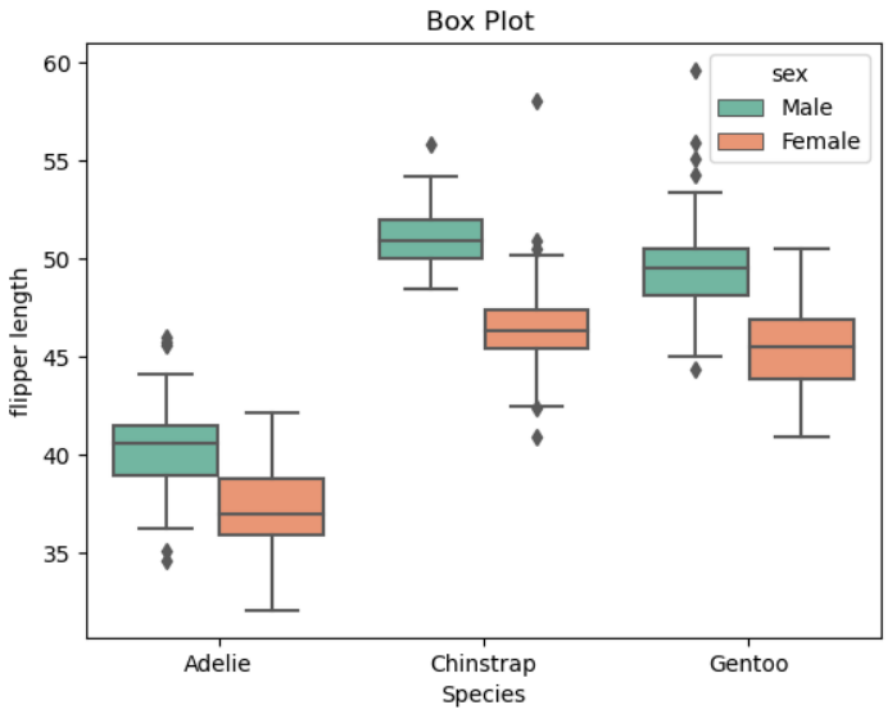
Box plot shows the quartiles where the data lies in interquartile range will be inside the box. The points which are away from the whiskers are outliers.

Code snippet:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

tips.boxplot(column="total_bill", by="day")
plt.title("Box Plot - Total Bill by Day")
plt.suptitle("")  # Remove default subtitle
plt.show()
```

Output:



Description:

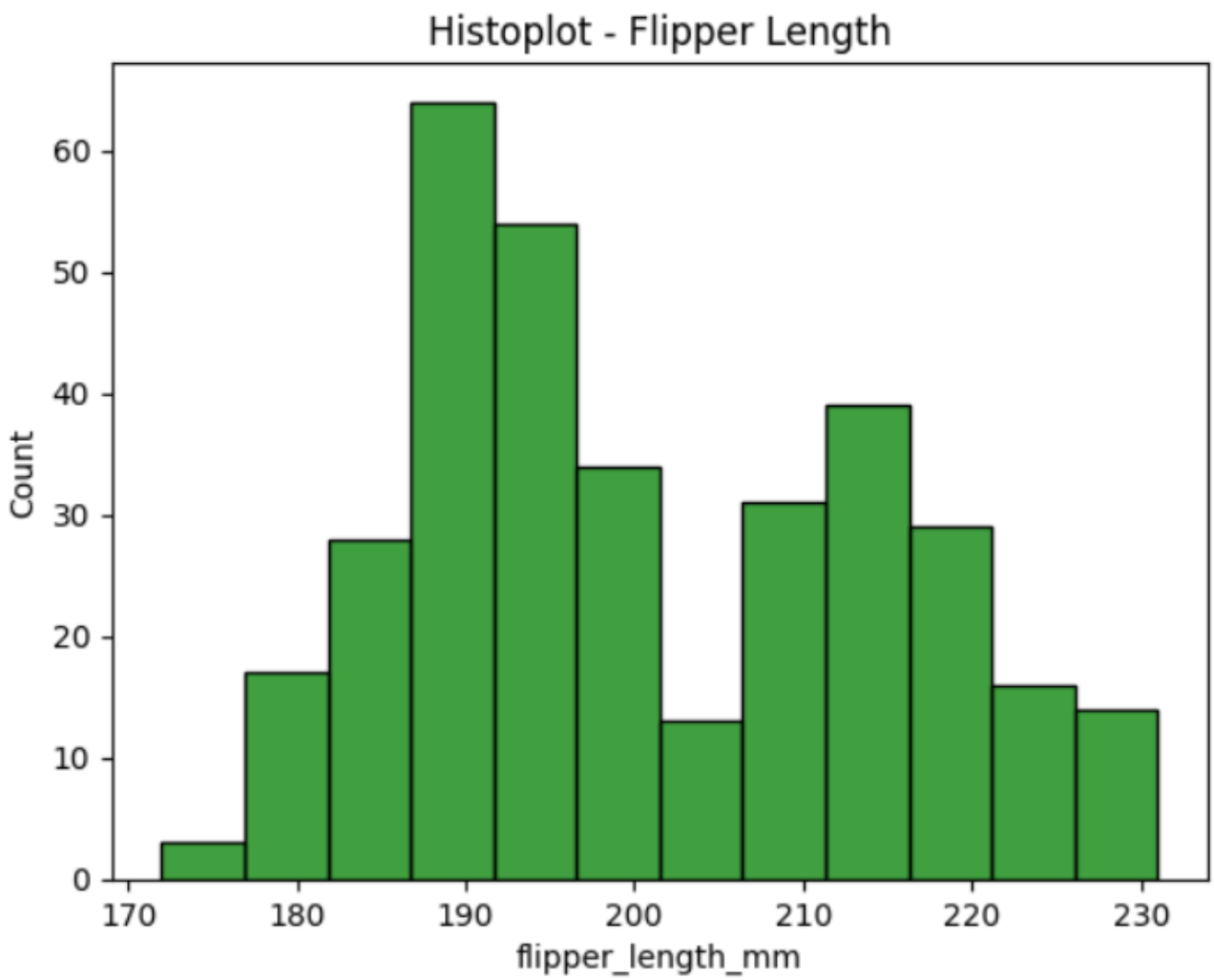
boxplot() is used to visualize the distribution of flipper_length across different penguin species.
The x parameter is set to "species" and the y parameter to "flipper_length".
The hue parameter is set to "sex", which uses different colors to distinguish Male and Female penguins within each species.
The plot shows the median (central line), interquartile range (box), and possible outliers (dots outside the whiskers).
legend() helps identify the colors for each sex.
title() sets the plot title, and show() displays the box plot.

HISTOPLOT

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

penguins = sns.load_dataset("penguins")
sns.histplot(data=penguins, x="flipper_length_mm", bins=12, color="green")
plt.title("Histoplot - Flipper Length")
plt.show()
```

Output:



HEATMAP

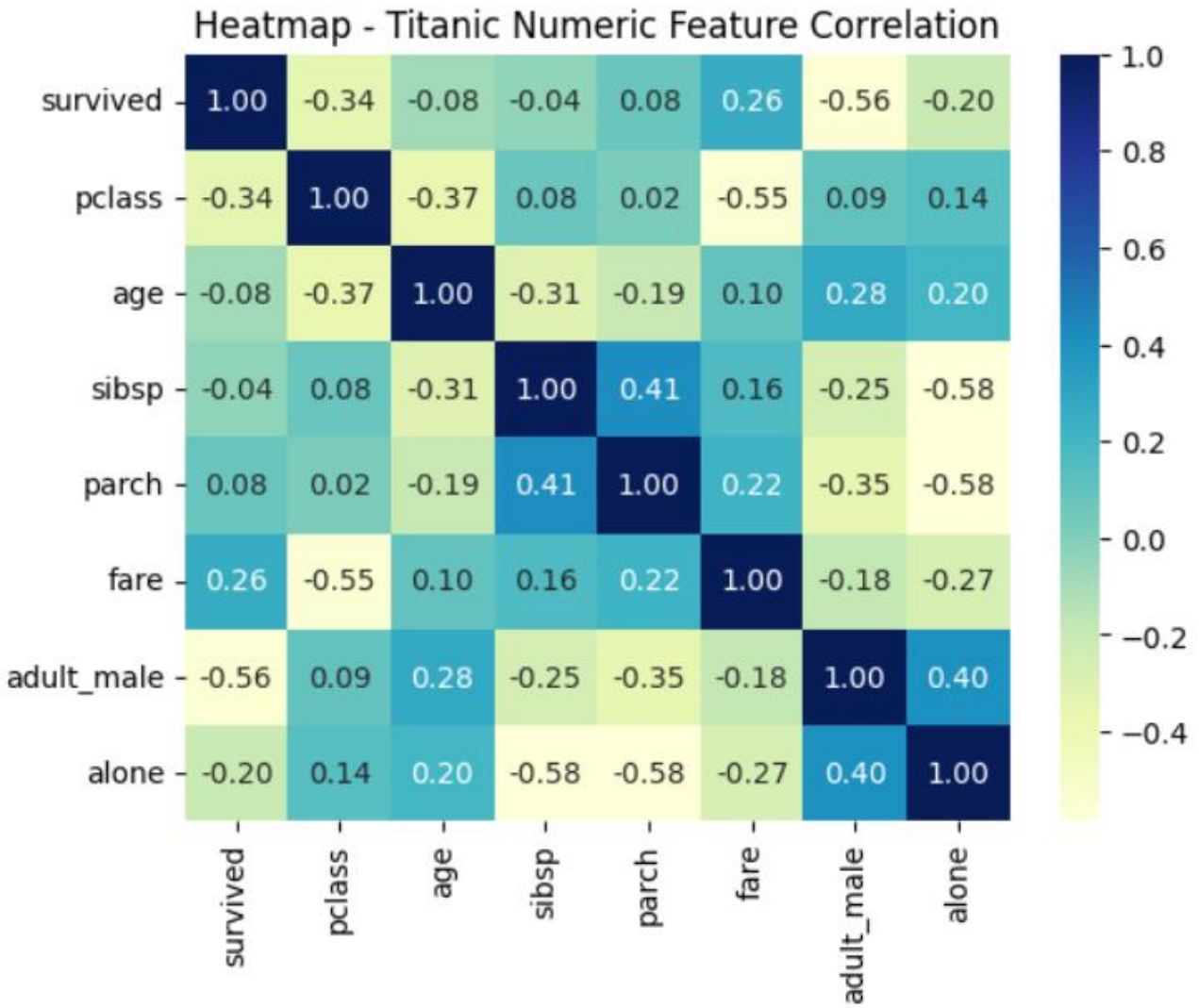
Heatmaps uses correlation matrix and visualizes data. The datapoints where the higher values get brighter colors and lower values get darker colors.

Code snippet:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(titanic.corr(numeric_only=True), annot=True, cmap="YlGnBu", fmt=".2f")
plt.title("Heatmap - Titanic Numeric Feature Correlation")
plt.show()
```

Output:



PAIRPLOT

ChatGPT said:

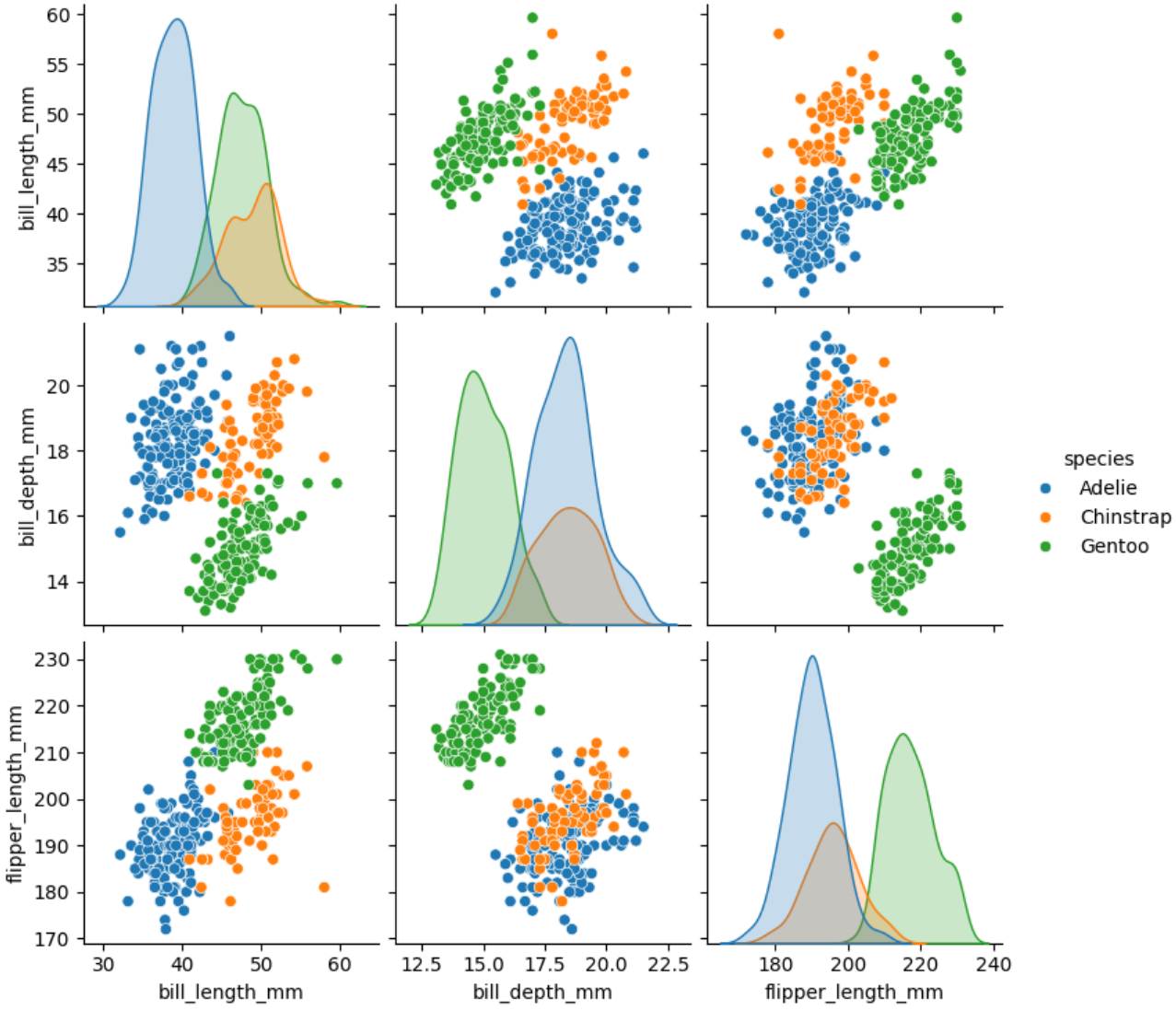
A **Pair plot** in Seaborn is a grid of plots that shows scatter plots for every pair of numerical variables and histograms for individual variables, helping visualize relationships and distributions.

Code snippet:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(penguins.dropna(), vars=["bill_length_mm", "bill_depth_mm", "flipper_length_mm"], hue="species")
plt.show()
```

Output:



COMPARISON OF MATPLOTLIB AND SEABORN

Matplotlib

1. Core Library – A comprehensive Python library for creating static, animated, and interactive visualizations with fine-grained control.
2. Flexibility – Can produce any type of plot, from basic charts to complex 3D and geographical maps.
3. Mature – Well-established, long-standing library; many other visualization tools (including Seaborn) are built on top of it.
4. Customization – Allows detailed customization of every aspect of a plot (colors, line styles, markers, fonts, annotations, etc.).
5. Integration – Works smoothly with NumPy, Pandas, SciPy, and other Python libraries.

Advantages of Matplotlib:

- High degree of customization and control.
- Wide variety of plot types and styles.
- Strong community support and documentation.
- Suitable for publication-quality graphics.

Seaborn

1. High-Level Interface – Built on Matplotlib; simplifies the creation of attractive, informative statistical graphics.
2. Statistical Plotting – Specializes in plots that show data relationships (scatter, bar, box, violin, pair plots, etc.).
3. Aesthetics – Comes with themes and color palettes for visually appealing results by default.
4. Integration with Pandas – Works seamlessly with Pandas DataFrames, reducing the need for data reshaping before plotting.

Advantages of Seaborn:

- Simplified syntax and high-level functions.
- Strong statistical plotting capabilities.
- Attractive default aesthetics.
- Easy integration with Pandas DataFrames.
- Ideal for quick exploratory data analysis.

Overall

- Matplotlib → Best for flexibility, detailed customization, and complex plots.
- Seaborn → Best for ease of use, statistical visualization, and attractive default designs.
- Many data scientists use both — Seaborn for quick plotting, Matplotlib for fine-tuning.