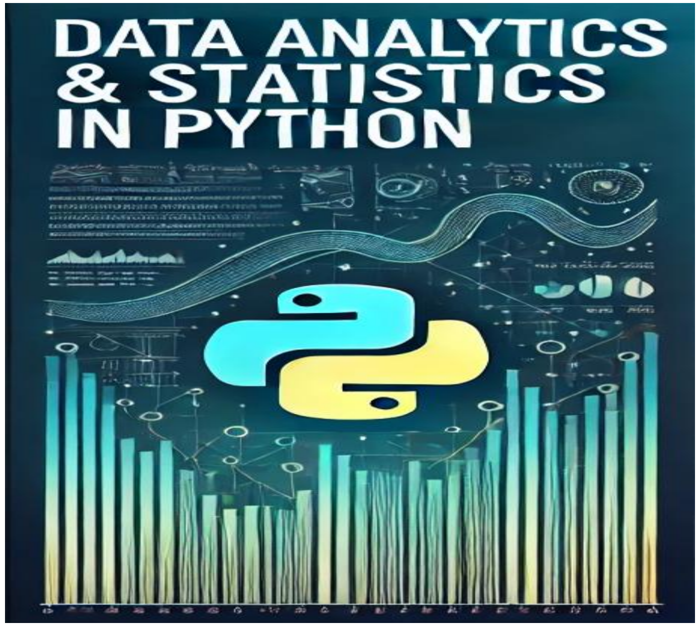
Data Analytics & Statistics in Python
Session 6: Data Visualisation



Metropolia
University of Applied Sciences

Learning data-driven decision-making with Python

Instructor: Hamed Ahmadinia, Ph.D.

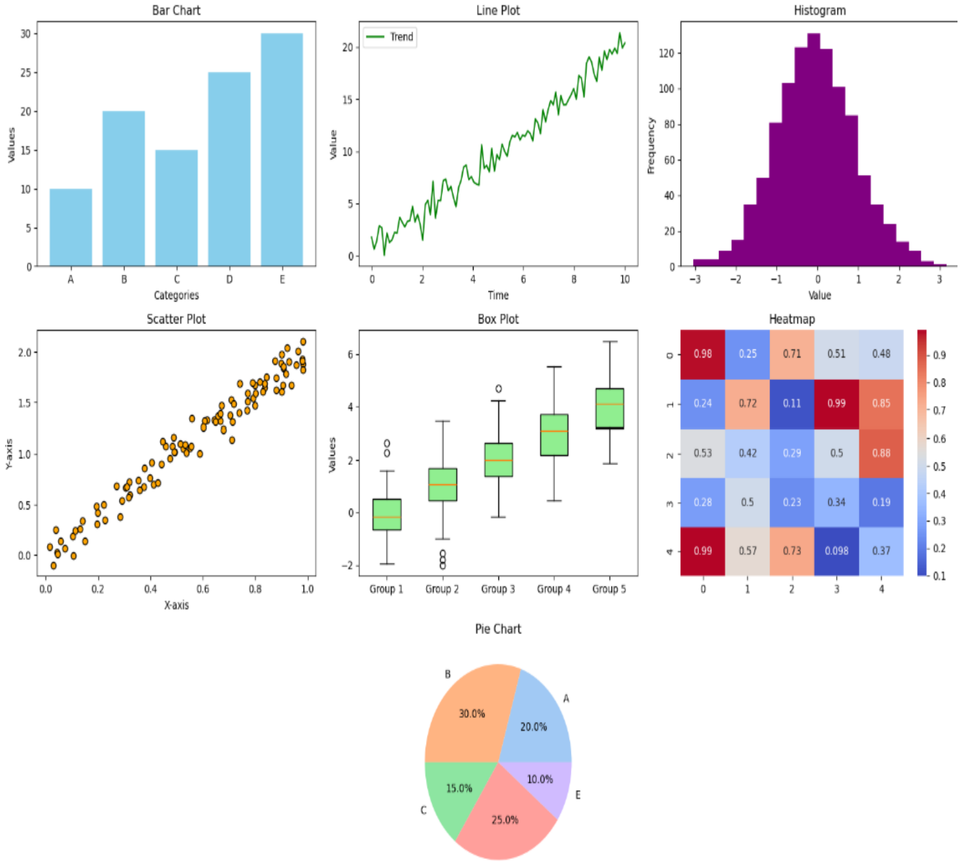
Email: hamed.ahmadinia@metropolia.fi

Concepts of Today

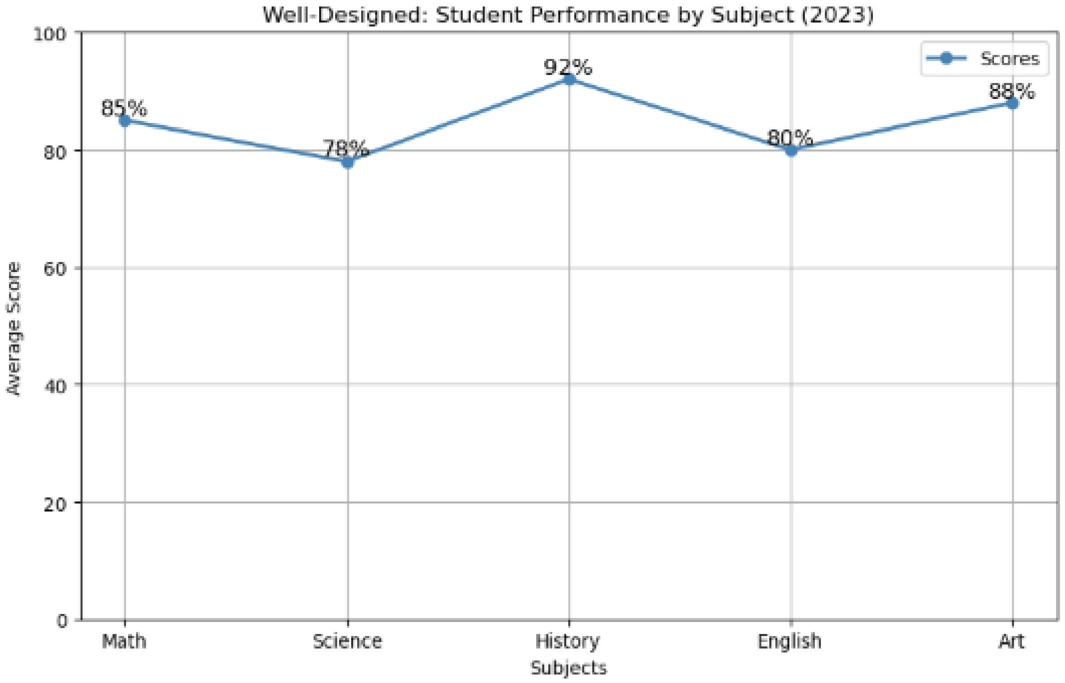
- **Key Topics:**
  1. **Introduction to Data Visualization**
  2. **Plotting Data**
  3. **Matplotlib Basics:**
     - Pyplot basics: creating plots and subplots
     - Annotating plots with labels and titles
  4. **Seaborn for Statistical Visualization:**
     - Histograms, scatter plots, and box plots
     - Heatmaps and pair plots
  5. **Advanced Plots:**
     - Pie charts and Residual plots

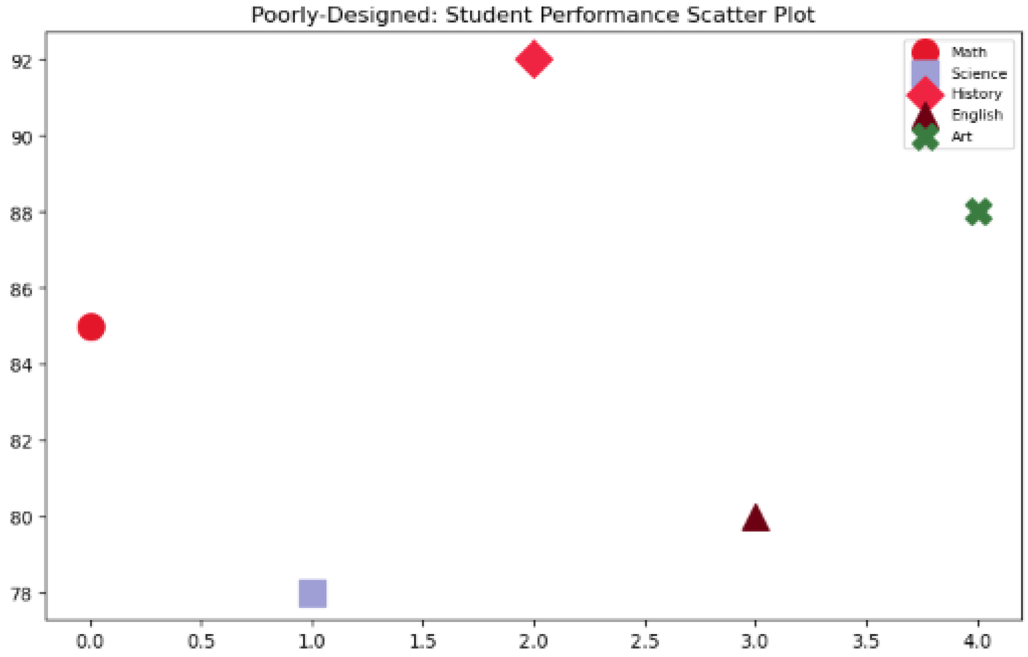
Introduction to Plotting Data

- **Why Visualize Data?**
  - Helps analyze and communicate data effectively.
  - Identifies patterns, trends, and outliers in data.
- **Common Data Visualization Types:**
  1. **Bar Charts**: Compare quantities across categories.
  2. **Line Plots**: Show trends over time or continuous data.
  3. **Histograms**: Show data distribution (bins).
  4. **Scatter Plots**: Visualize relationships between two variables.
  5. **Box Plots**: Summarize data distribution and highlight outliers.
  6. **Heatmaps**: Display values' intensity using color gradients.
  7. **Pie Charts**: Show proportions as slices of a whole.



Good vs Poor Data Visualization





Key Takeaways:

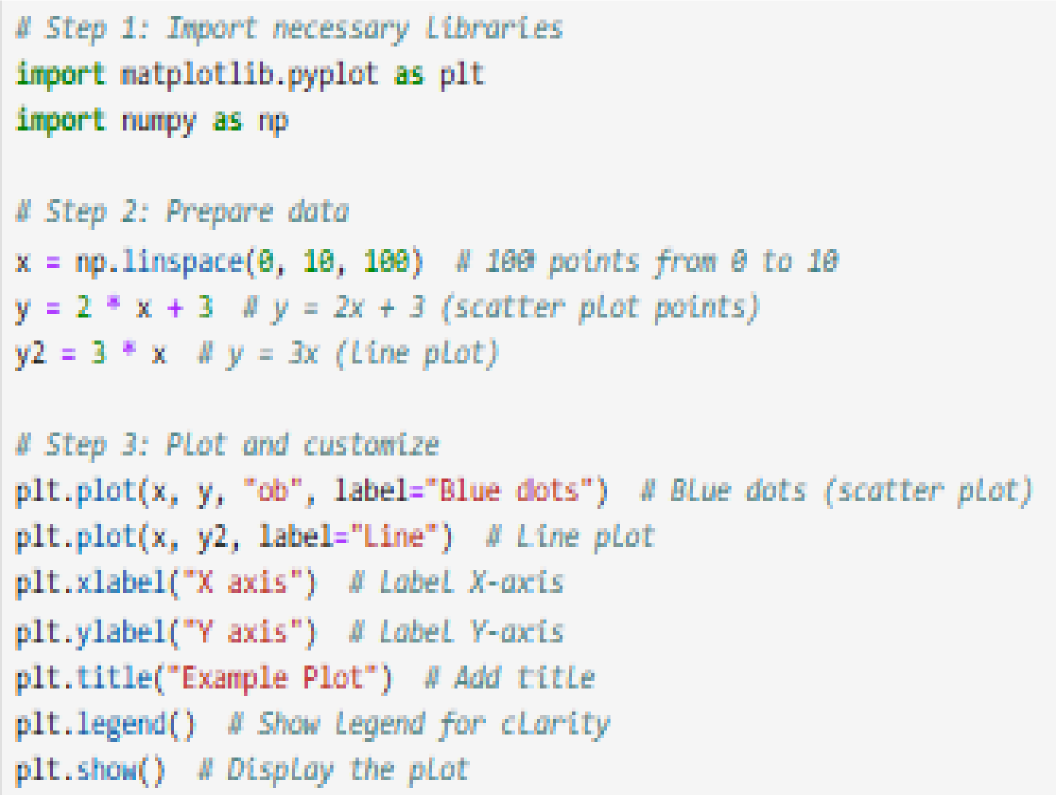
1. Choose the right chart type for your data.
2. Focus on clarity and avoid unnecessary elements.
3. Ensure titles, legends, and labels enhance understanding.

Matplotlib

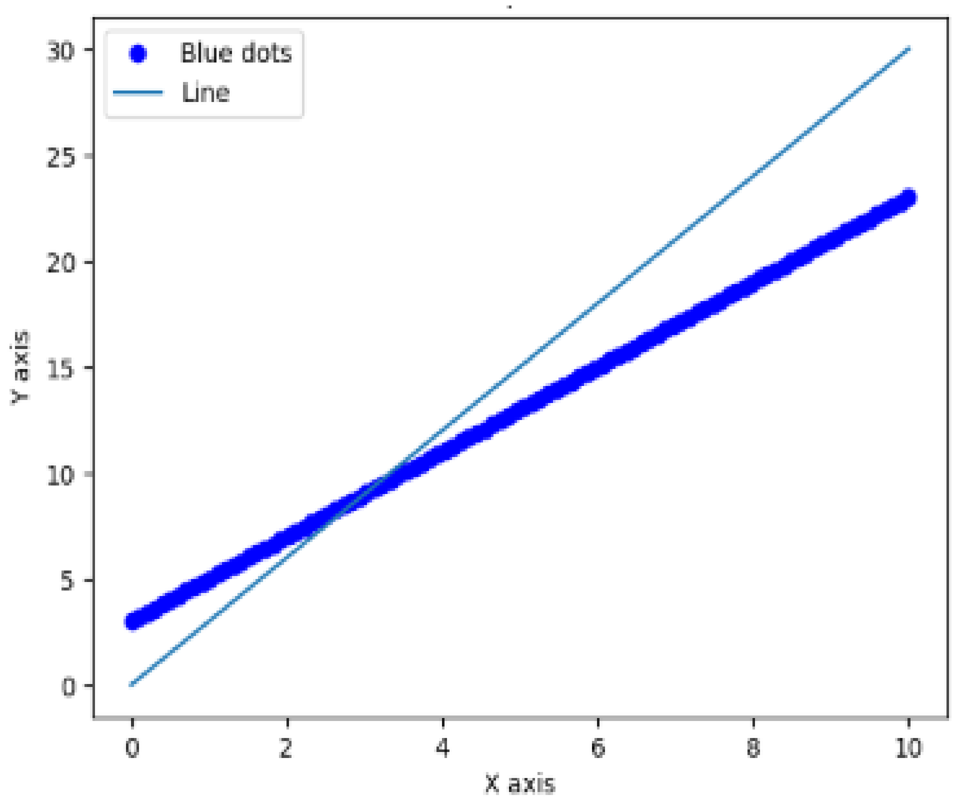
- **What is Matplotlib?**
- Matplotlib is the primary Python library for plotting data.
- It allows the creation of a wide range of plots such as line plots, scatter plots, histograms, and subplots.

- **Why Matplotlib?**
- Beginner-friendly for simple plots.
- Supports advanced customization for detailed data visualizations.
- Works well with other libraries like NumPy, Pandas, and Seaborn.

Pyplot Basics

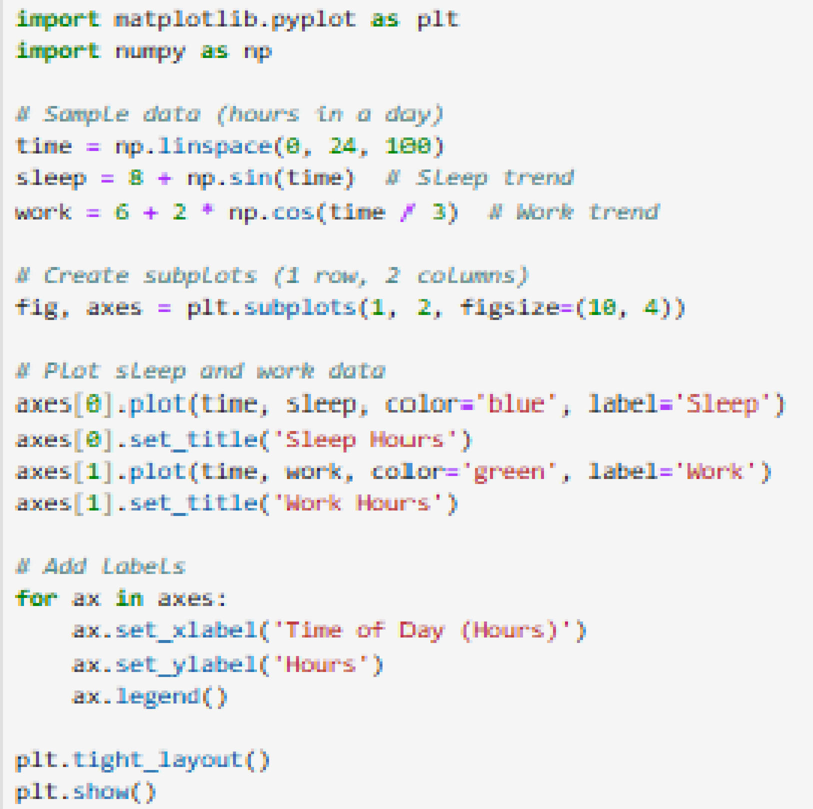


```
# Step 1: Import necessary libraries
import matplotlib.pyplot as plt
import numpy as np

# Step 2: Prepare data
x = np.linspace(0, 10, 100)  # 100 points from 0 to 10
y = 2 * x + 3  # y = 2x + 3 (scatter plot points)
y2 = 3 * x  # y = 3x (line plot)

# Step 3: Plot and customize
plt.plot(x, y, "ob", label="Blue dots")  # Blue dots (scatter plot)
plt.plot(x, y2, label="Line")  # Line plot
plt.xlabel("X axis")  # Label X-axis
plt.ylabel("Y axis")  # Label Y-axis
plt.title("Example Plot")  # Add title
plt.legend()  # Show legend for clarity
plt.show()  # Display the plot
```

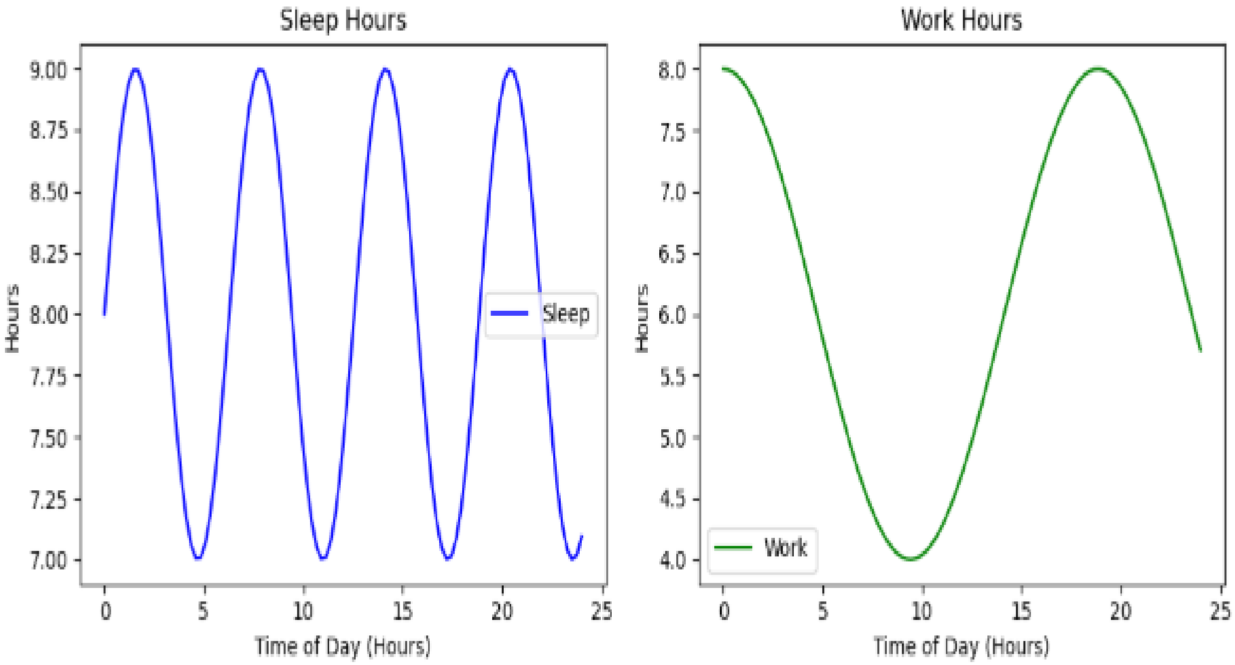


Creating Subplots

Subplots allow multiple plots in one figure—ideal for comparisons of trends and data visualizations.

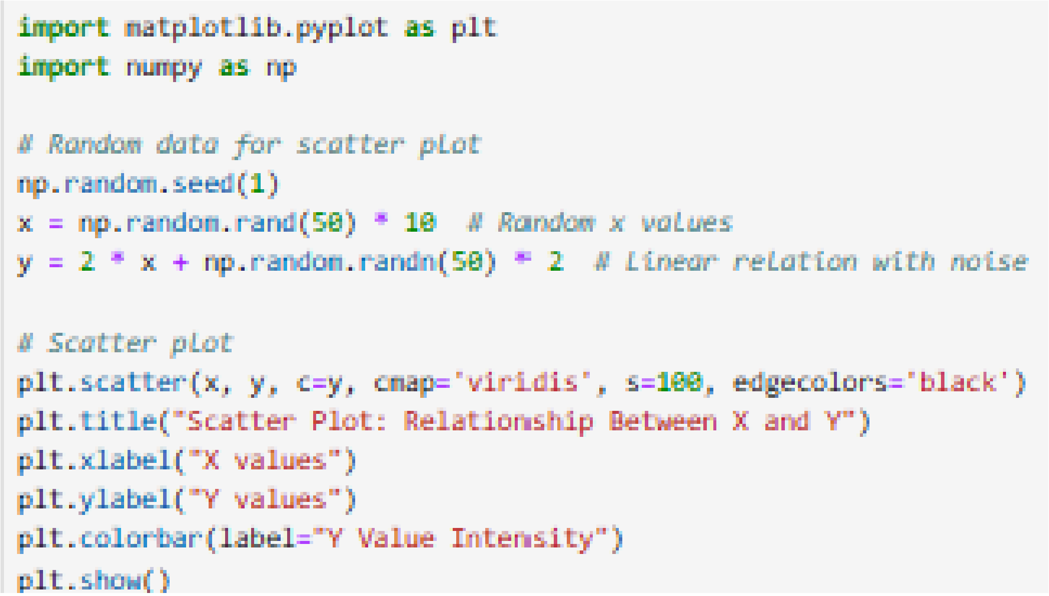


```
import matplotlib.pyplot as plt
import numpy as np

# Sample data (hours in a day)
time = np.linspace(0, 24, 100)
sleep = 8 + np.sin(time)  # Sleep trend
work = 6 + 2 * np.cos(time / 3)  # Work trend

# Create subplots (1 row, 2 columns)
fig, axes = plt.subplots(1, 2, figsize=(10, 4))

# Plot sleep and work data
axes[0].plot(time, sleep, color='blue', label='Sleep')
axes[0].set_title('Sleep Hours')
axes[1].plot(time, work, color='green', label='Work')
axes[1].set_title('Work Hours')

# Add labels
for ax in axes:
    ax.set_xlabel('Time of Day (Hours)')
    ax.set_ylabel('Hours')
    ax.legend()

plt.tight_layout()
plt.show()
```

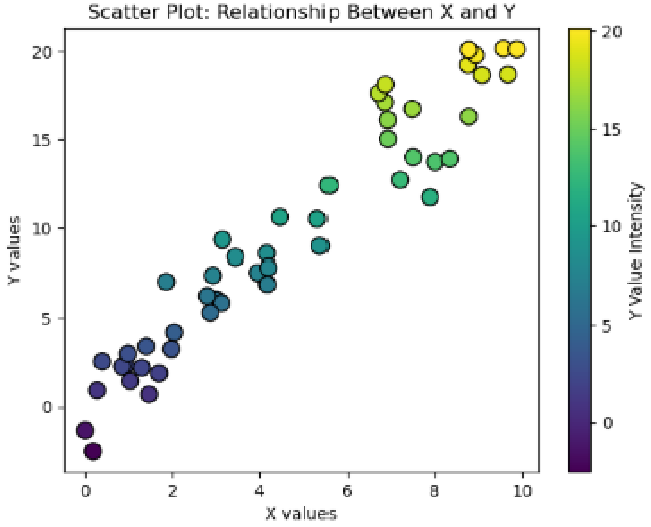


Scatter Plots (Visualizing Relationships)

- **Purpose**: Shows the relationship between two variables.
- **Best for**: Detecting trends, clusters, and outliers.

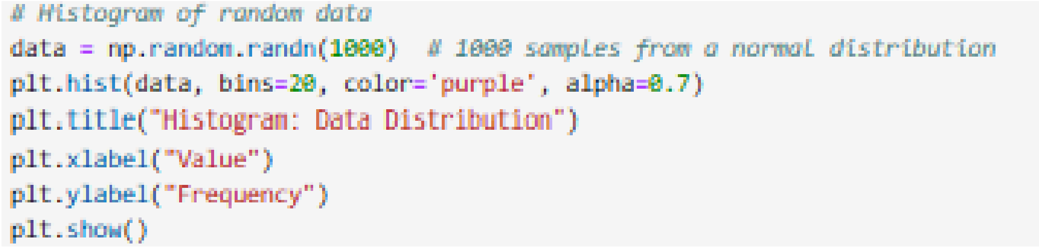


```python
import matplotlib.pyplot as plt
import numpy as np

# Random data for scatter plot
np.random.seed(1)
x = np.random.rand(50) * 10  # Random x values
y = 2 * x + np.random.randn(50) * 2  # Linear relation with noise

# Scatter plot
plt.scatter(x, y, c=y, cmap='viridis', s=100, edgecolors='black')
plt.title("Scatter Plot: Relationship Between X and Y")
plt.xlabel("X values")
plt.ylabel("Y values")
plt.colorbar(label="Y Value Intensity")
plt.show()
```

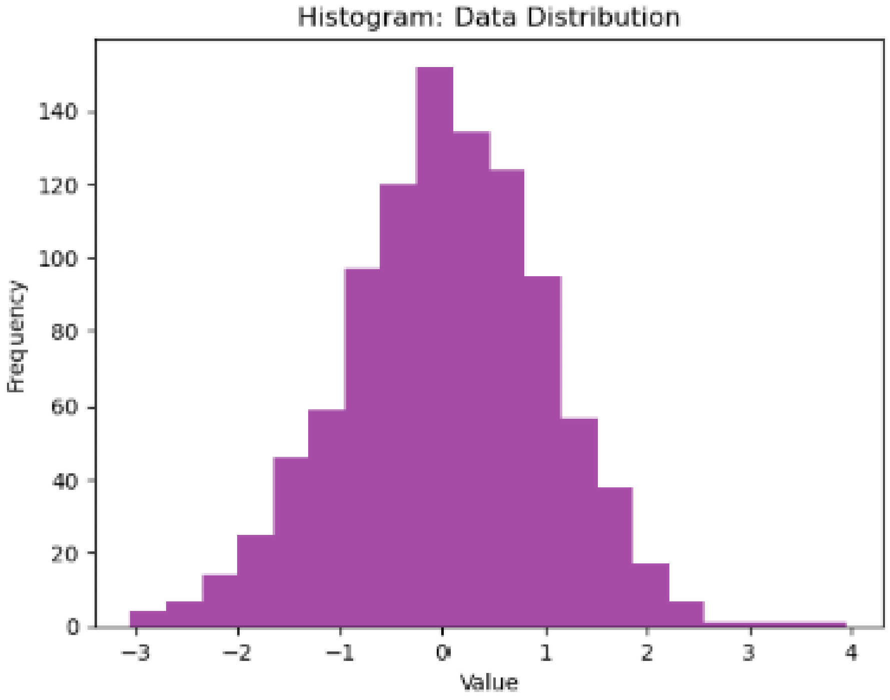


Histograms (Distribution of Data)

- **Purpose**: Visualizes the frequency of values in a dataset.
- **Best for**: Analyzing distributions (e.g., normal, skewed).

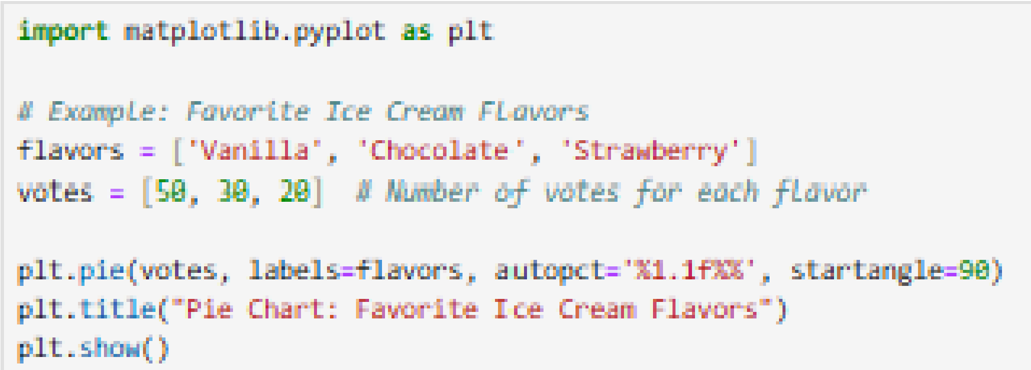


```
# Histogram of random data
data = np.random.randn(1000)  # 1000 samples from a normal distribution
plt.hist(data, bins=20, color='purple', alpha=0.7)
plt.title("Histogram: Data Distribution")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

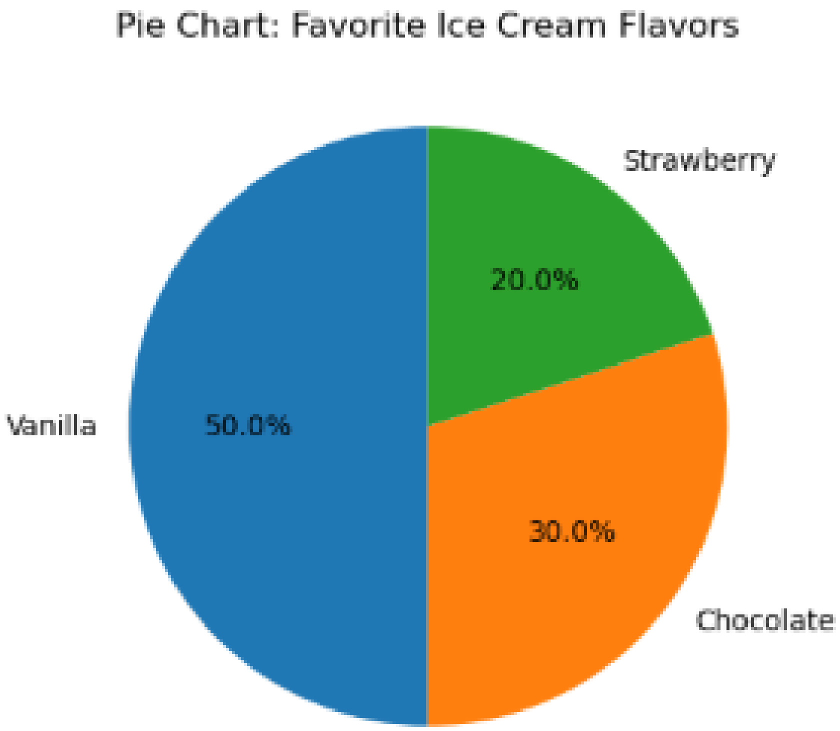


Pie Charts (Proportion Visualization)

- **Purpose:** Displays proportions of categories in a whole.
- **Best for:** Showing relative contributions.



```
import matplotlib.pyplot as plt

# Example: Favorite Ice Cream Flavors
flavors = ['Vanilla', 'Chocolate', 'Strawberry']
votes = [50, 30, 20]  # Number of votes for each flavor

plt.pie(votes, labels=flavors, autopct='%1.1f%%', startangle=90)
plt.title("Pie Chart: Favorite Ice Cream Flavors")
plt.show()
```



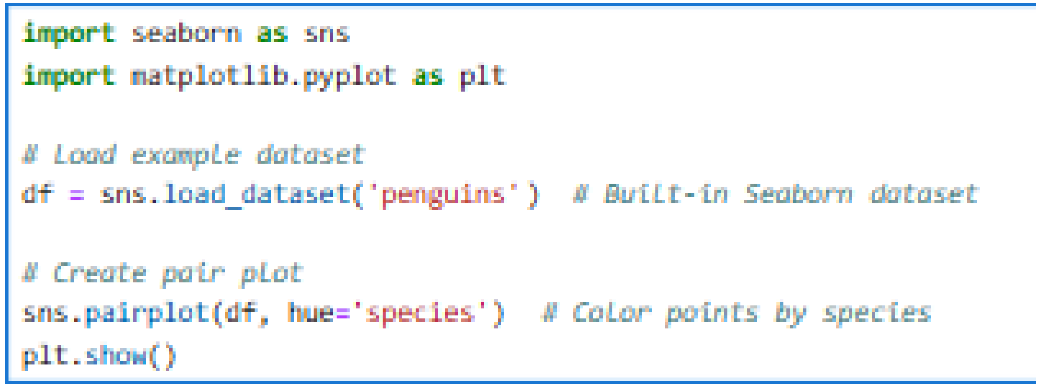
Vanilla has the most votes, followed by chocolate and strawberry.

Seaborn for Statistical Plots

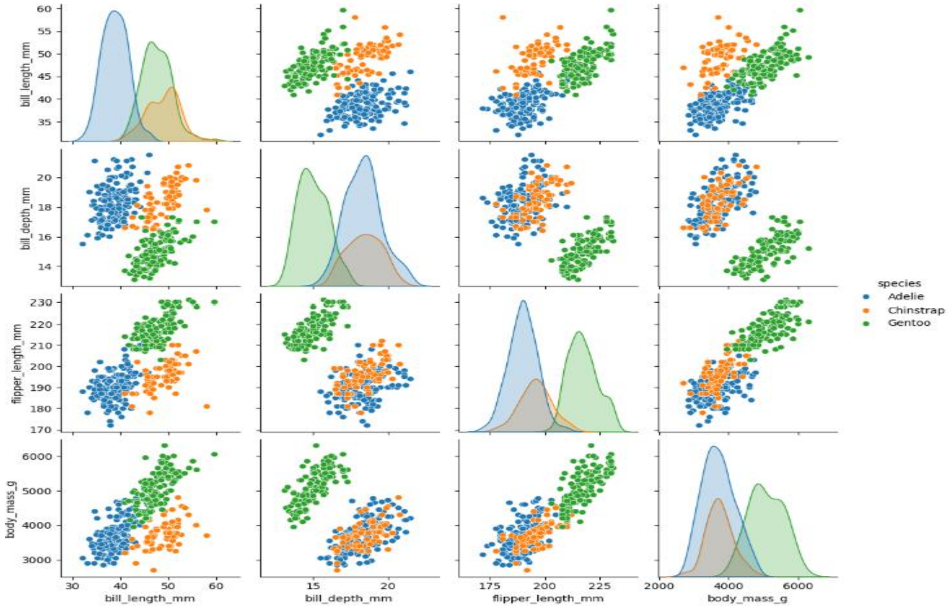
- **Seaborn Overview:**
  - **Purpose**: A high-level data visualization library based on Matplotlib.
  - **Key Features**:
    - Integration with pandas for effortless DataFrame handling.
    - Creates visually appealing plots with minimal code.
    - Great for statistical analysis and exploratory data visualization.
- **Why Seaborn?**
  - Compact code: Create detailed plots with just 1–2 lines of code.
  - Supports complex statistical plots such as **pair plots, boxplots, heatmaps**, and **residual plots**.

Plotting Distributions with Seaborn

- **1. Pair Plots: Compare Relationships Between Variables**
- **What is a Pair Plot?**
  - A **grid of scatter plots** showing relationships between numerical features.
  - The **diagonal** contains histograms that show the distribution of each feature.
  - Useful for spotting patterns, trends, and outliers.

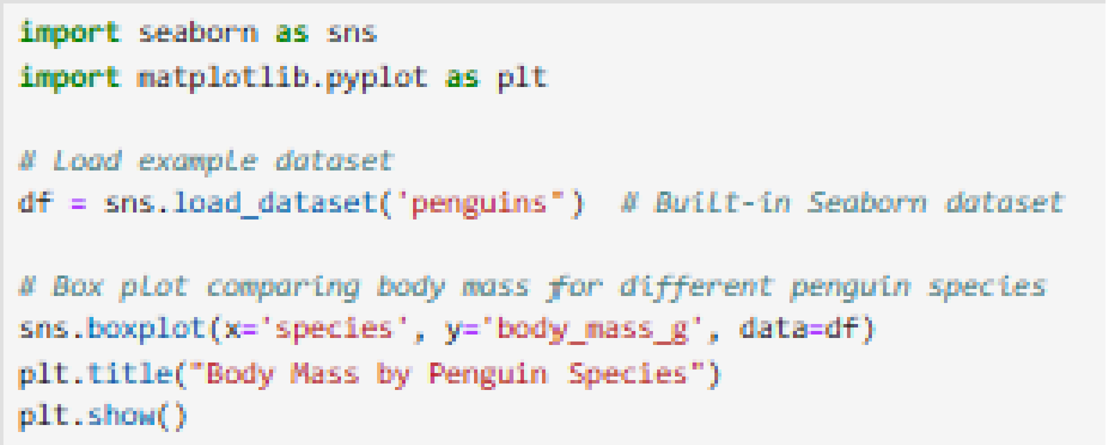


```
import seaborn as sns
import matplotlib.pyplot as plt

# Load example dataset
df = sns.load_dataset('penguins')  # Built-in Seaborn dataset

# Create pair plot
sns.pairplot(df, hue='species')  # Color points by species
plt.show()
```

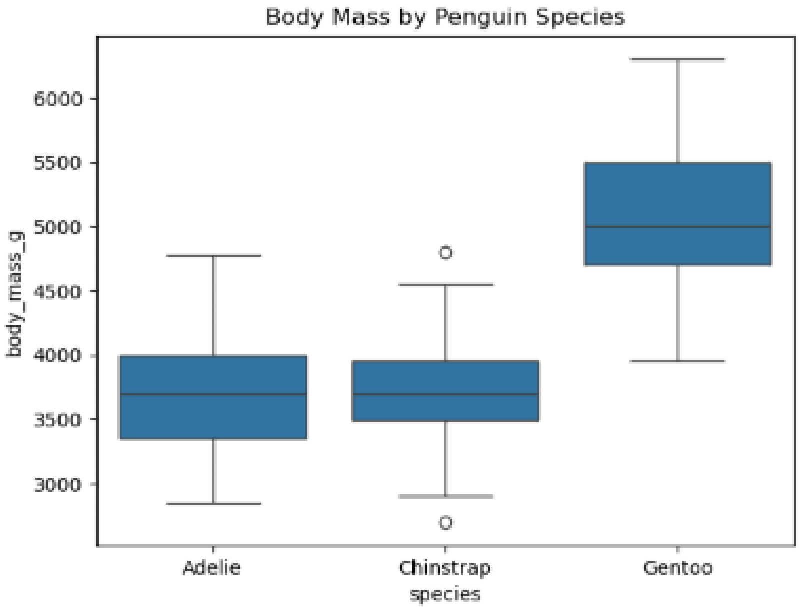


Plotting Distributions with Seaborn

- **2. Box Plots: Summarize Data Distribution and Identify Outliers**
- **What is a Box Plot?**
  - Shows the **median**, **quartiles**, and **outliers** of the data.
  - Great for comparing distributions across categories (e.g., species).

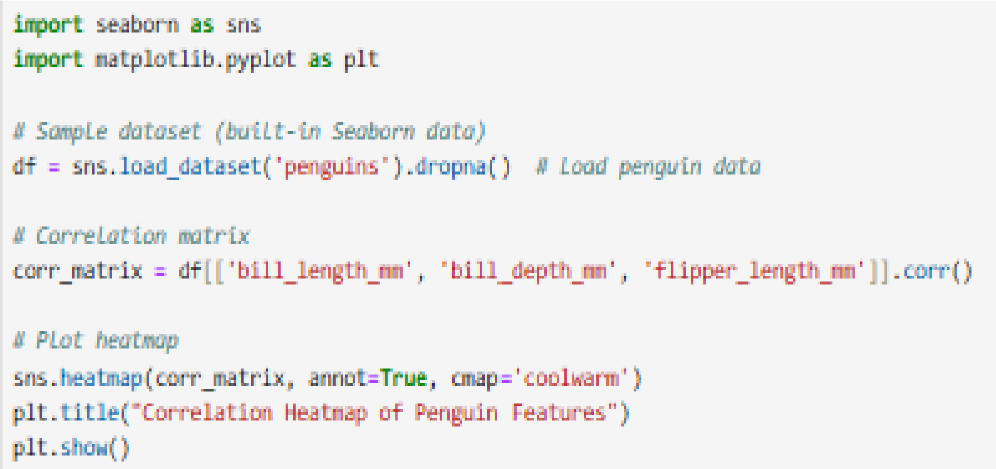


```
import seaborn as sns
import matplotlib.pyplot as plt

# Load example dataset
df = sns.load_dataset('penguins')  # Built-in Seaborn dataset

# Box plot comparing body mass for different penguin species
sns.boxplot(x='species', y='body_mass_g', data=df)
plt.title("Body Mass by Penguin Species")
plt.show()
```

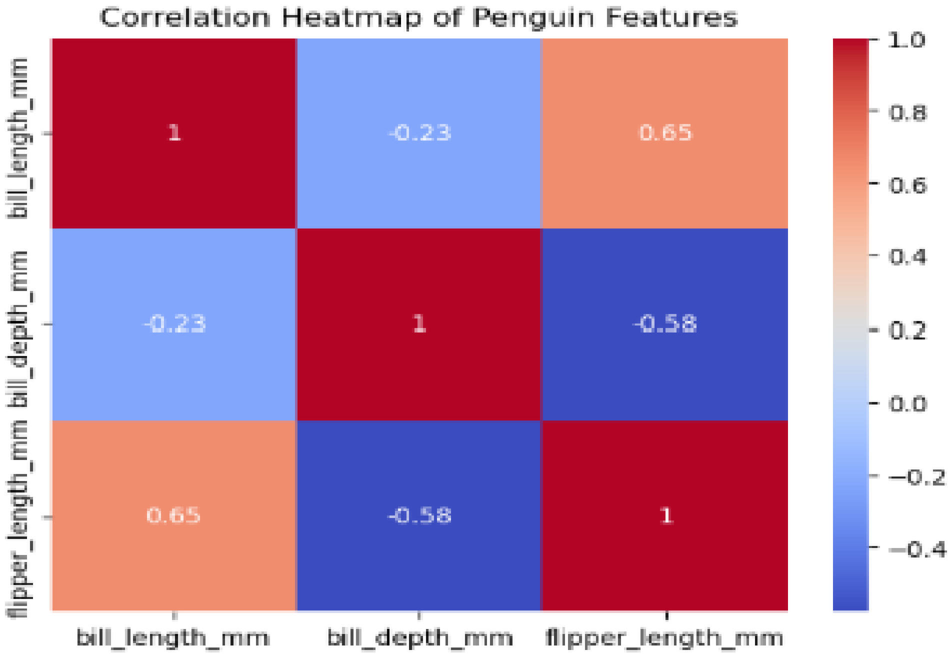


Heatmaps (Correlation Matrix)

- **Purpose:** A heatmap visualizes how strongly features are related (correlated) using colors.



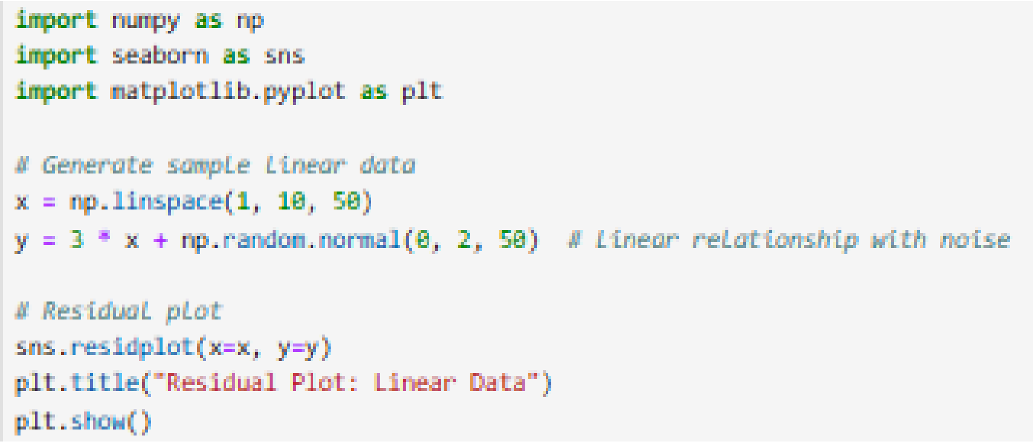
```
import seaborn as sns
import matplotlib.pyplot as plt

# Sample dataset (built-in Seaborn data)
df = sns.load_dataset('penguins').dropna()  # Load penguin data

# Correlation matrix
corr_matrix = df[['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm']].corr()

# Plot heatmap
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap of Penguin Features")
plt.show()
```



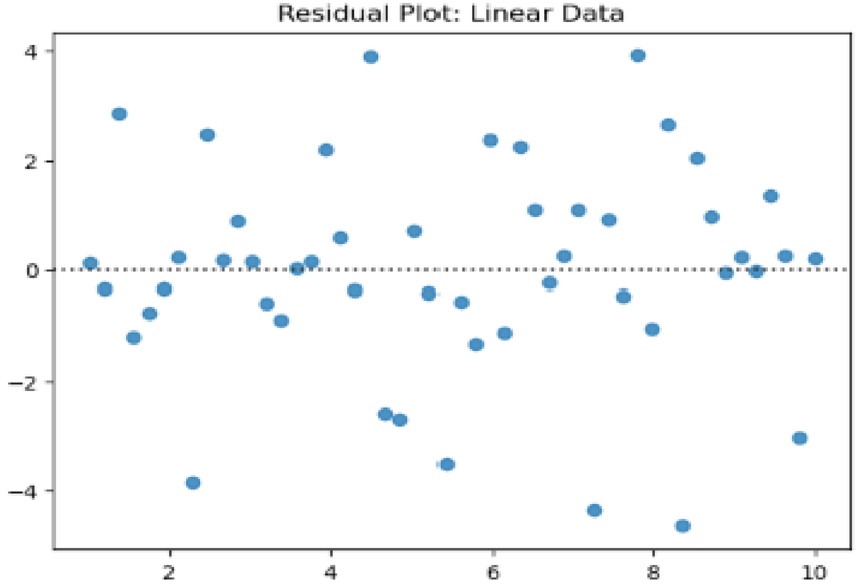
Dark red = strong positive correlation (increase together).
Dark blue = strong negative correlation (one increases, the other decreases).
Helps spot which features are related (e.g., bill length vs flipper length).

Residual Plots (Check Regression Line Fit)

- **Purpose:** Residual plots help check if a linear regression model fits the data well.



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

# Generate sample linear data
x = np.linspace(1, 10, 50)
y = 3 * x + np.random.normal(0, 2, 50)  # Linear relationship with noise

# Residual plot
sns.residplot(x=x, y=y)
plt.title("Residual Plot: Linear Data")
plt.show()
```

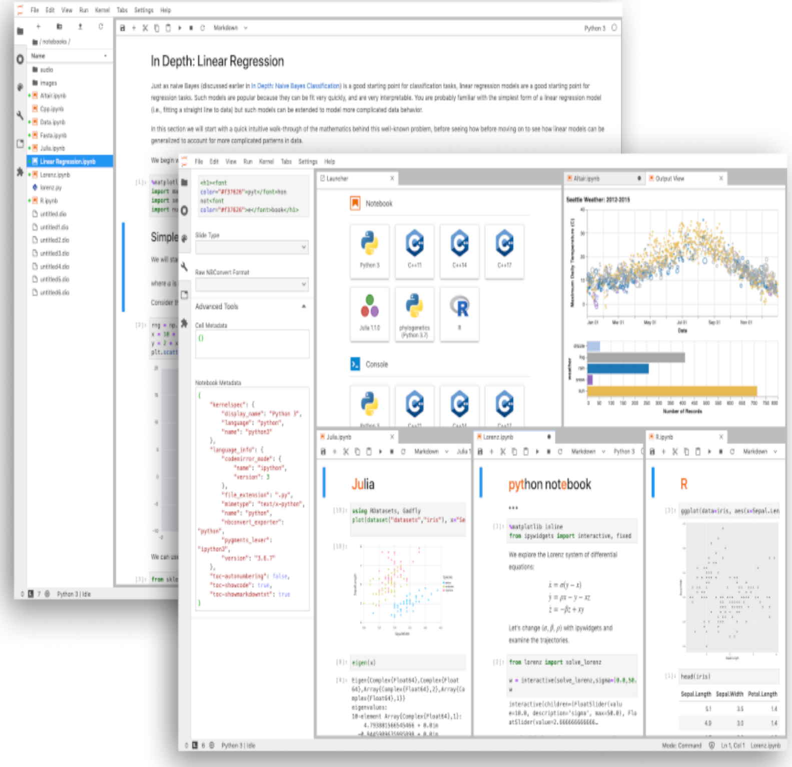


- Residuals: The difference between predicted and actual values.
- Ideal Plot: Randomly scattered points around 0 → good fit.
- Non-linear Fit Issue: If a pattern (e.g., curve) appears, the relationship might not be linear.

Notebook Review

Walk through how to apply key Python concepts in a Jupyter Notebook:

- Matplotlib Basics
- Pyplot basics
- Seaborn for Statistical Visualization
- Histograms, scatter plots, and box plots
- Heatmaps and pair plots
- Pie charts and Residual plots



Kahoot Quiz Time!

Let’s Test Our Knowledge!

Hands-on Exercise

Form groups (2–3 members).

- Download *Hands-on Exercise #6* from the course page.
- Complete the coding tasks and discuss your solutions.
- Don’t forget to add the names of your group members to the file.
- Submit your completed *Hands-on Exercise* to the course Moodle page or send it to the teacher's email address.

Reference

- Vohra, M., & Patil, B. (2021). A Walk Through the World of Data Analytics. , 19-27. https://doi.org/10.4018/978-1-7998-3053-5.ch002.
- VanderPlas, J. (2016). Python data science handbook: Essential tools for working with data. O'Reilly Media. Available at https://jakevdp.github.io/PythonDataScienceHandbook/
- Severance, C. (2016). Python for everybody: Exploring data using Python 3. Charles Severance. Available at https://www.py4e.com/html3/
- McKinney, W. (2017). *Python for data analysis: Data wrangling with pandas, NumPy, and Jupyter*. O'Reilly Media.