## National University of Computer and Emerging Sciences



**Lab 10**

Department of Computer Science

FAST-NU, Lahore, Pakistan

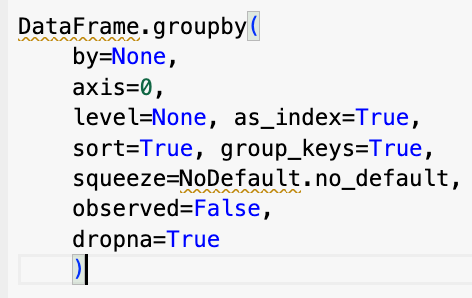
**Data Aggregation**

Data aggregation is a core step in data science that involves summarizing and combining data. Pandas provides powerful tools to aggregate and reshape data using functions like groupby, pivot\_table, and melt. These functions allow you to rearrange and condense data for easier analysis, visualization, and reporting.

## Group By Method

This function in Pandas groups data by one or more columns and applies aggregate functions, such as mean, sum, count, etc., on the grouped data. It allows you to split a DataFrame into groups based on values in one or more columns, perform operations on each group, and combine the results.

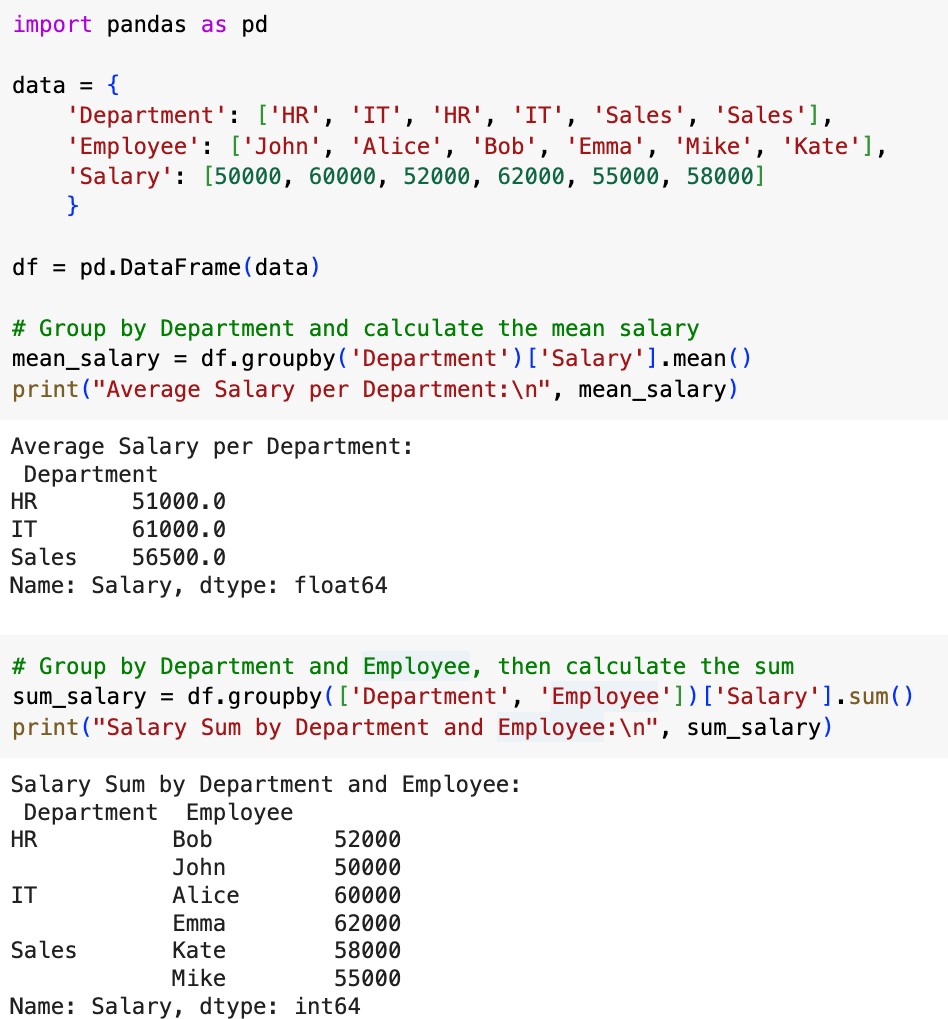
## Syntax:



**Parameters:**

* **by**: Column(s) or index level(s) by which to group the data.
* **axis**: Determines the axis to group by (0 for rows, 1 for columns).
* **level**: Groups by a specific level if the DataFrame has a MultiIndex.
* **as\_index**: If True (default), the grouping columns are set as the index of the result.
* **sort**: Sorts group keys, affecting the output order.
* **group\_keys**: Adds group keys to the result if True.

## Example Usage:

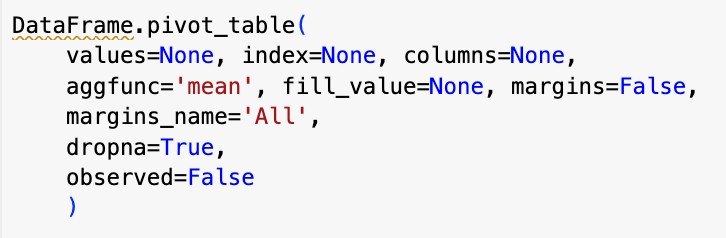


**Pivot Table Method**

This function creates a spreadsheet-style pivot table. It allows you to create summaries of data across multiple dimensions, similar to Excel’s pivot table functionality. It’s especially helpful for organizing large datasets, performing aggregations across multiple levels, and displaying data in a more structured format.

A pivot table allows you to calculate and display aggregated values, such as mean, sum, or count, across rows and columns based on one or more keys.

## Syntax:

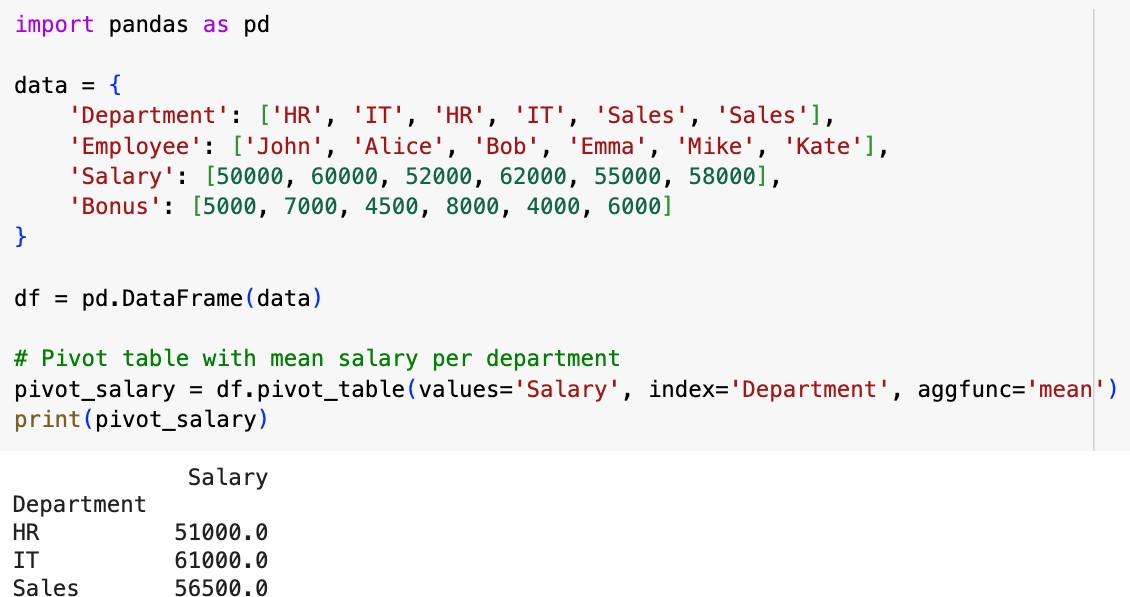


**Key Parameters:**

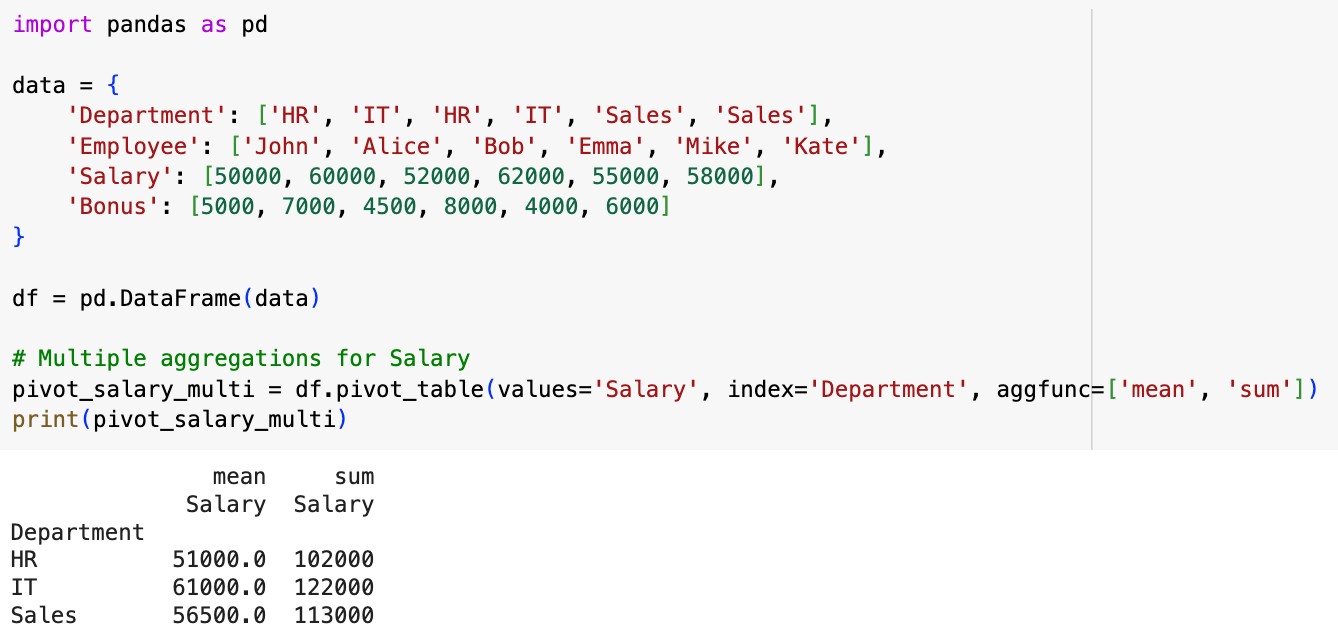
* **values**: Specifies the column(s) to aggregate.
* **index**: Sets the rows for the pivot table.
* **columns**: Sets the columns for the pivot table.
* **aggfunc**: Defines the aggregation function to apply (e.g., mean, sum, count). It defaults to 'mean'.
* **ﬁll\_value**: Replaces missing values with a specified value.
* **margins**: Adds row and column totals when set to True.
* **margins\_name**: Changes the name of the totals row/column if margins is True.

## Example Usage:

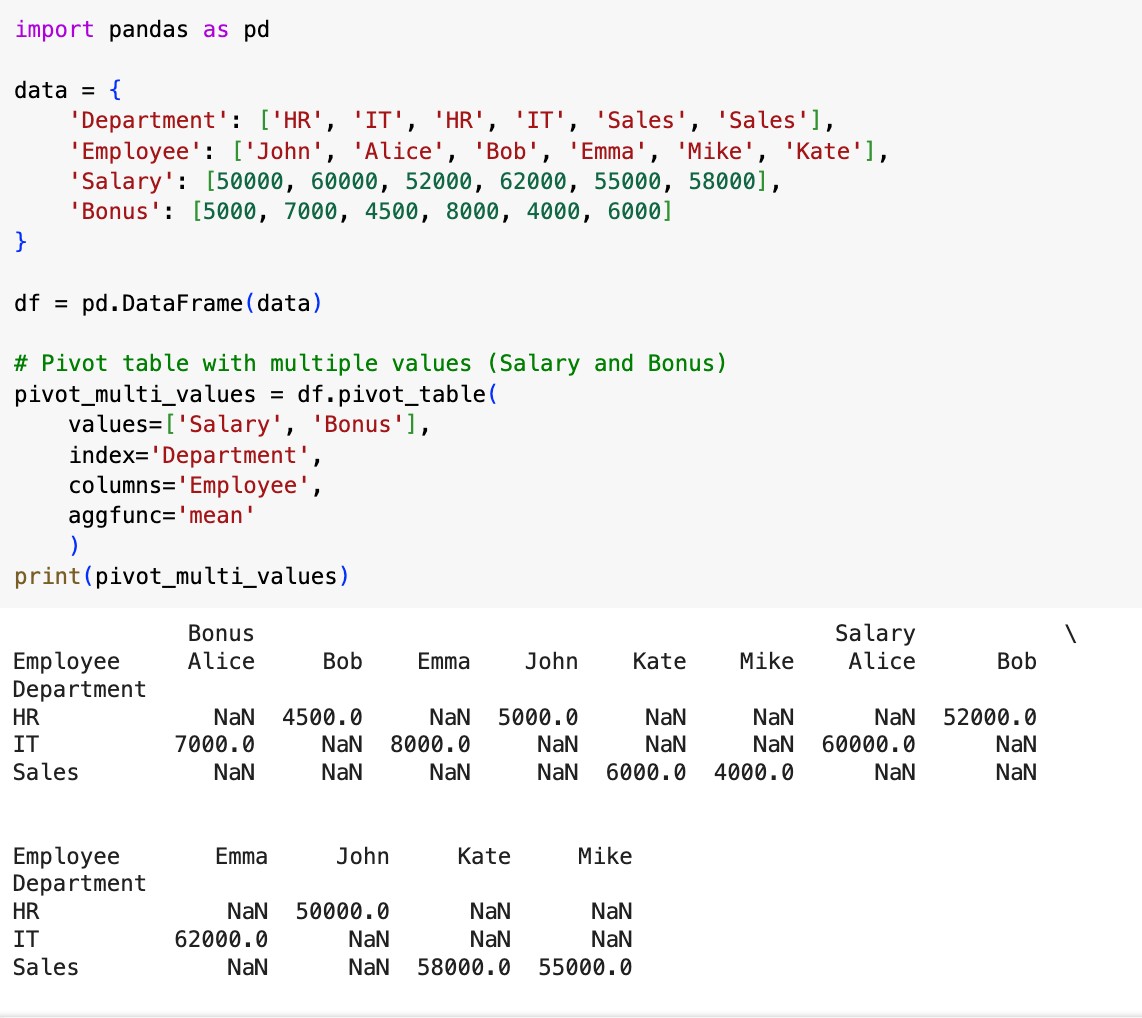
To calculate the average salary per department, you can create a pivot table with 'Department' as the index and 'Salary' as the values:



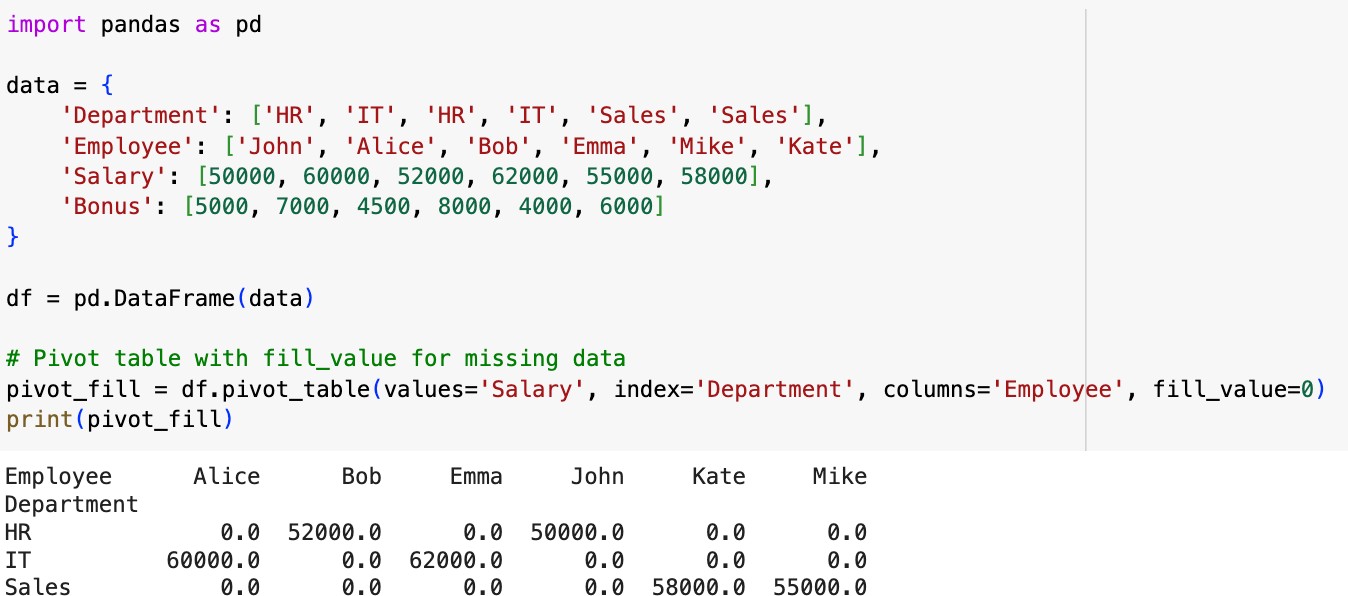
You can apply multiple aggregation functions to the values by passing a list to aggfunc. Here’s an example that calculates both the sum and mean of the salaries for each department:



You can also include multiple columns in the values and columns parameters. Here’s an example where we calculate the mean of both Salary and Bonus, broken down by Department and Employee.



If some combinations in your pivot table are missing values, you can use the fill\_value parameter to replace them with a specific value.



## Summary of pivot\_table:

* **Single Aggregation**: Use pivot\_table(values, index, aggfunc='mean') for simple summaries.
* **Multiple Aggregations**: Specify aggfunc=['sum', 'mean'] for multiple calculations.
* **Multiple Columns/Values**: Use index and columns parameters to organize complex data.
* **Handling Missing Data**: Use fill\_value to replace missing values.
* **Row and Column Totals**: Use margins=True to add totals for rows/columns.

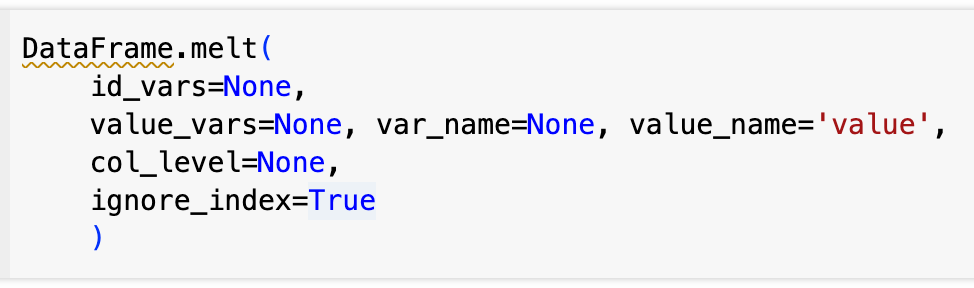
## Melt Method

This method in Pandas is used to transform or "unpivot" data, making it longer rather than wider. This is useful for reshaping data, especially when you need to convert wide-format tables (where multiple values are in separate columns) into a tidy long-format table, with one column for variable names and one for values.

The melt() method reshapes the DataFrame by:

* Transforming specified columns into rows.
* Stacking selected columns under a single "variable" column while placing their corresponding values under a "value" column.

## Syntax:

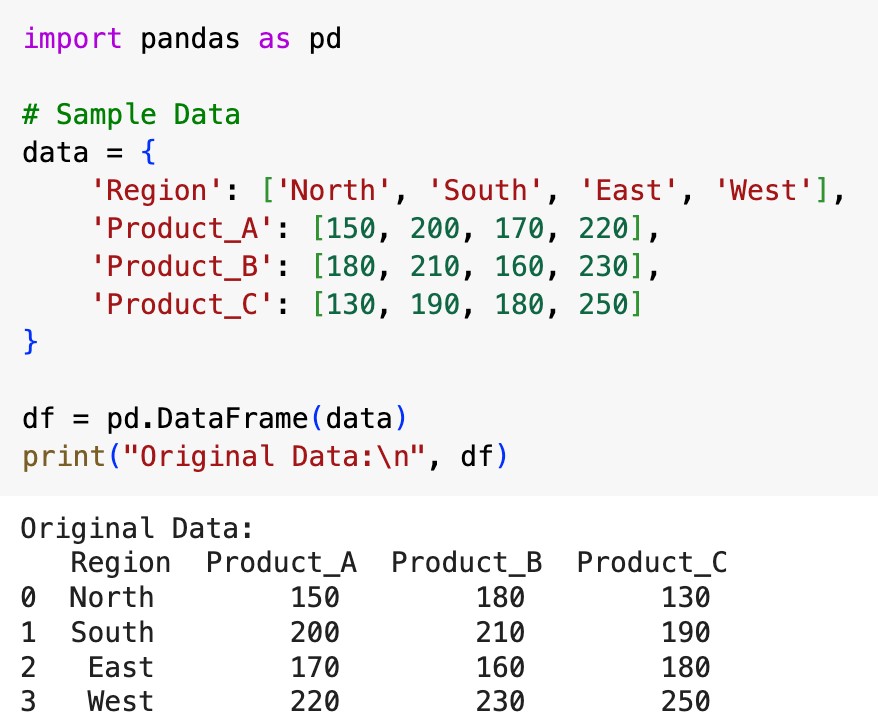


**Key Paramters:**

* **id\_vars**: Specifies the columns that should remain unchanged (the identifiers).
* **value\_vars**: Specifies the columns to "melt" into rows (the variables).
* **var\_name**: Sets the name for the new column that holds the names of the "melted" columns (variable names).
* **value\_name**: Sets the name for the new column that holds the values from the "melted" columns.
* **ignore\_index**: If set to True, the index will be reset; otherwise, it retains the original DataFrame's index.

## Example Usage:

Consider a simple example dataset with sales data across diﬀerent products and regions:



To reshape this DataFrame and make it longer (tall), we can "melt" the Product\_A, Product\_B, and Product\_C columns into rows while keeping Region as an identifier:

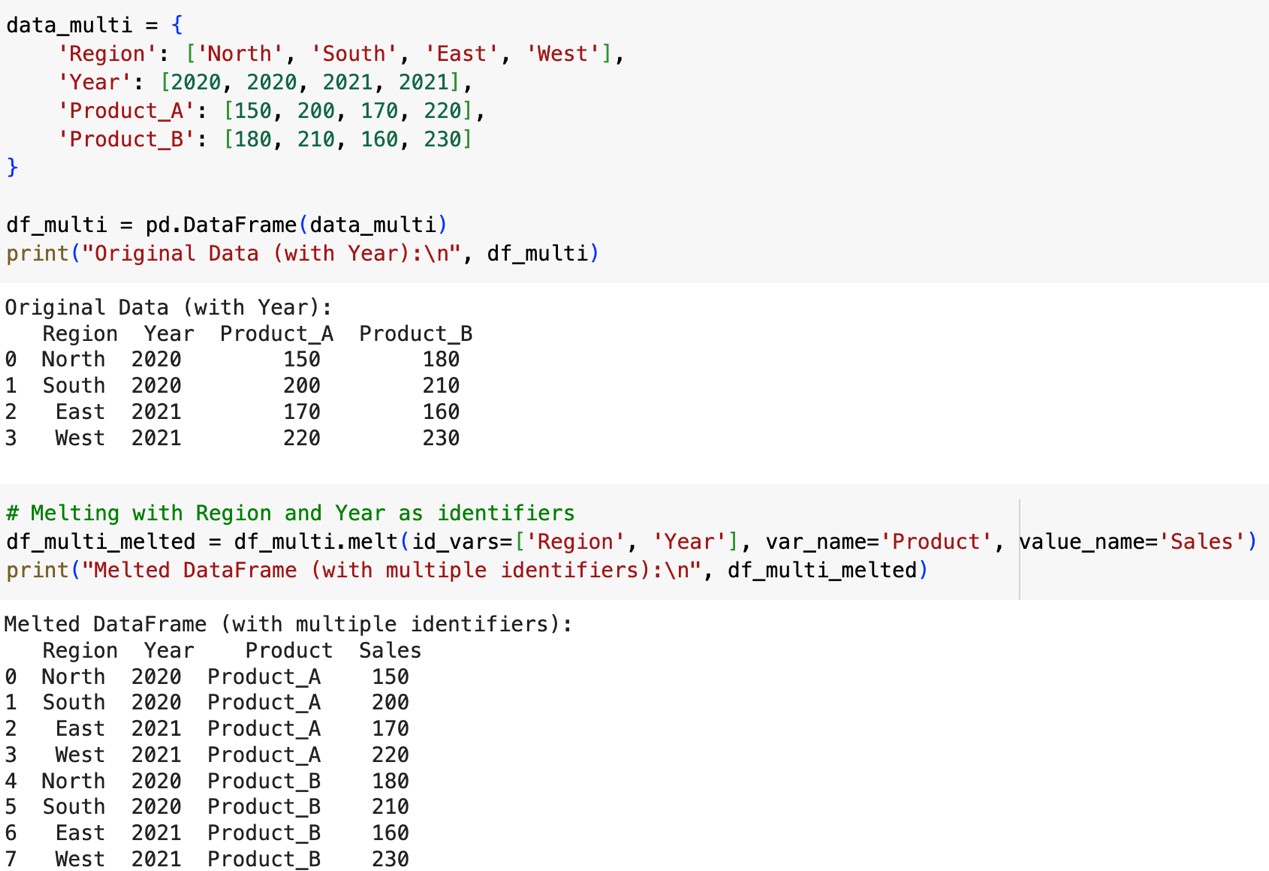


## Explanation of Parameters in This Example

* **id\_vars = 'Region'**: Keeps the Region column as-is, without transforming it.
* **value\_vars = ['Product\_A', 'Product\_B', 'Product\_C']**: Specifies which columns to "melt" into rows.
* **var\_name = 'Product'**: Renames the column that holds the original column names (the product names).
* **value\_name = 'Sales'**: Renames the column that holds the values of the melted columns (sales data).

## Melt method with Multiple Identifiers:

If your DataFrame has multiple columns that should remain as identifiers, you can specify them all in id\_vars.



Now, both Region and Year are retained as identifiers, while the product columns are melted into rows.

## Summary of melt:

* **Basic Reshaping**: Use id\_vars to specify the identifier column and value\_vars for the columns to melt.
* **All Columns Except Identifier**: Omit value\_vars to melt all other columns.
* **Multiple Identifiers**: Specify a list in id\_vars to keep multiple columns as identifiers.
* **Custom Column Names**: Use var\_name and value\_name to rename the resulting columns.

# Statistical Analysis

Intro to Statistical Terms: Introduction to EDA | Descriptive Statistics Intro to Matplotlib: Chart & Graphs | Matplotlib

### Exercises:

# Melt Exercises:

# Task 1: Use the melt function to transform the dataset, keeping PassengerId and Name as identifier variables, and melting the columns Sex, Age, and Fare into a long format.

# Task 2: Use the melt function to reshape the dataset, keeping PassengerId as the identifier, and melting Pclass, Embarked, and Survived into a long format.

# Task 3: Melt the columns SibSp and Parch while keeping PassengerId as the identifier to analyze the number of siblings/spouses and parents/children for each passenger.

# GroupBy Exercises:

# Task 1: Group the data by Sex and calculate the survival rate for each gender.

# Task 2: Group the data by Pclass and calculate the average fare for each class.

# Task 3: Group the data by Embarked and count the number of survivors for each port.

# ****Pivot Table Exercises:****

# ****Task 1:**** Create a pivot table to show the survival rate, indexed by Sex and with columns asPclass.

# Task 2: Create a pivot table showing the average fare for each combination of Sex and Embarked.

# Task 3: Create a pivot table that shows the total count of passengers, indexed by Pclass and with columns as Survived.

# MatplotLib

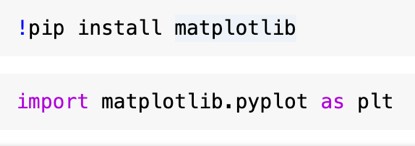
**Matplotlib** is a widely used plotting library for the Python programming language, designed to provide an easy and efficient way to create static, animated, and interactive visualizations.

## Why do we need it?

The primary purpose of Matplotlib is to allow users to create high-quality graphs, charts, and plots that can help convey complex data insights visually. It is particularly well-suited for:

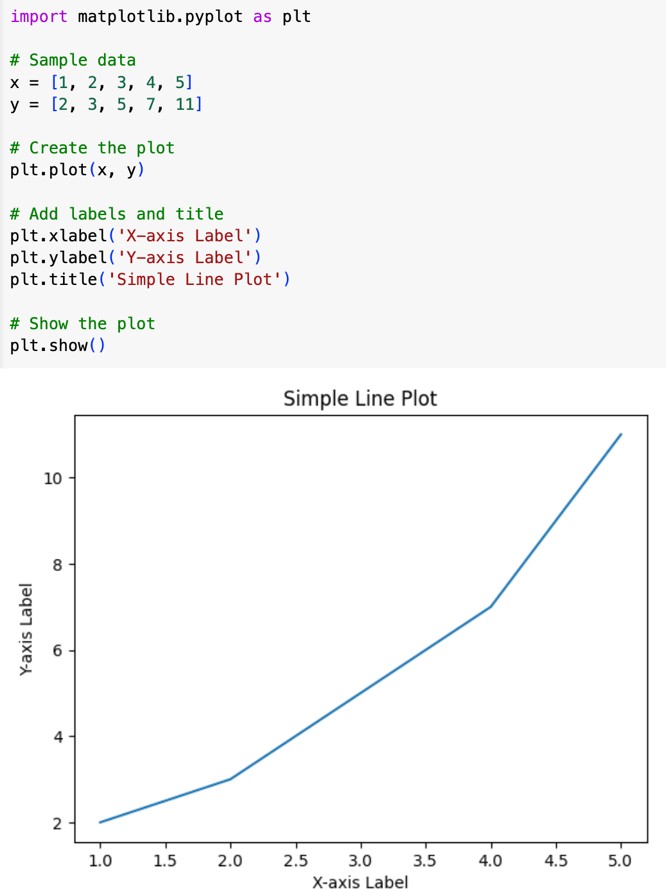
* **Exploratory Data Analysis (EDA)**: Quickly visualizing data distributions and relationships.
* **Statistical Visualization**: Creating plots that summarize or analyze data, such as histograms, box plots, and scatter plots.
* **Publishing and Presentation**: Generating publication-quality figures for research papers, reports, or presentations.

## Installation and Importing:



**Examples:**

To create a simple line plot, follow these steps:

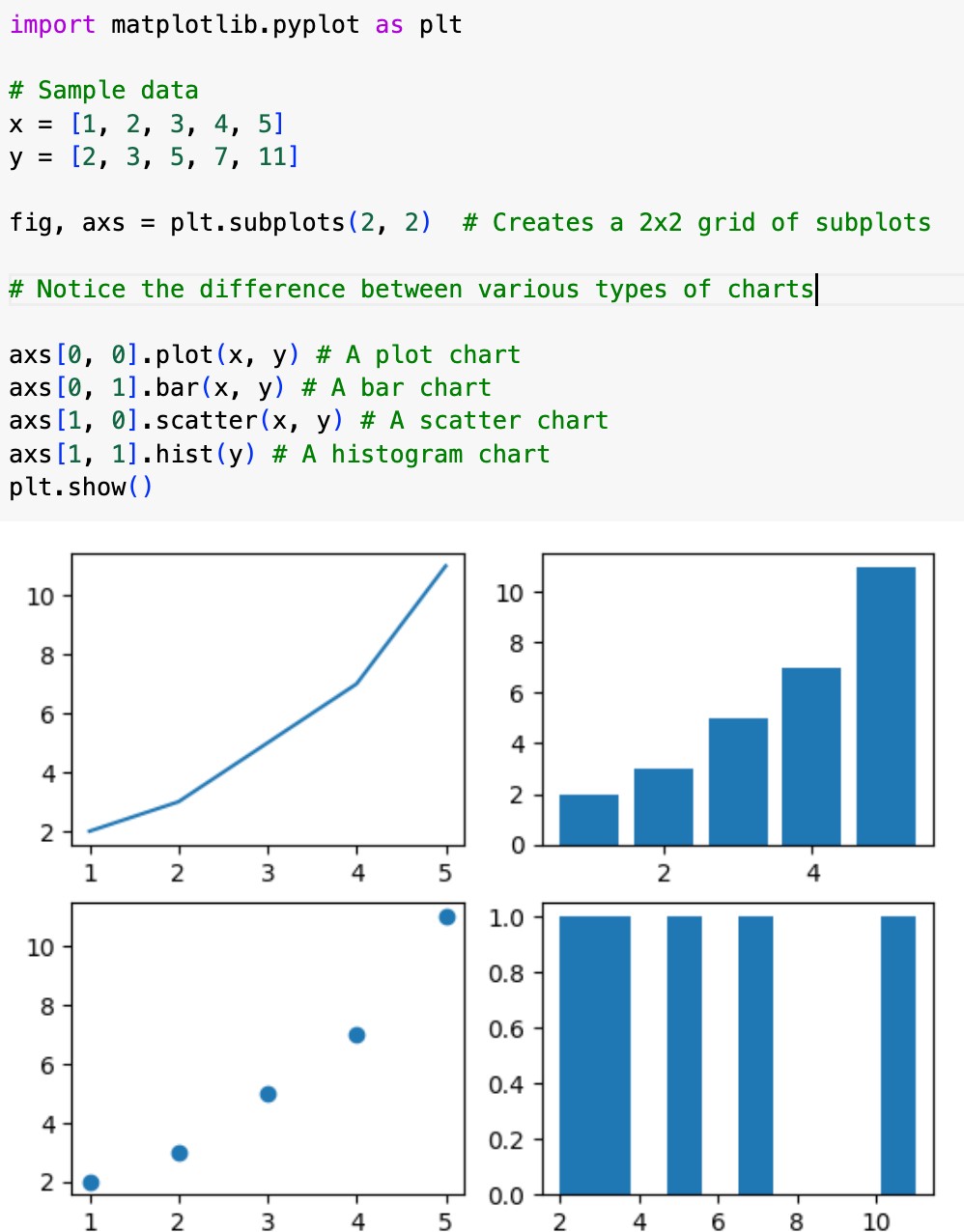


## Figures and Axes

* **Figure**: The entire window or page that contains all the plots.
* **Axes**: A single plot or graph in a figure. A figure can contain multiple axes.

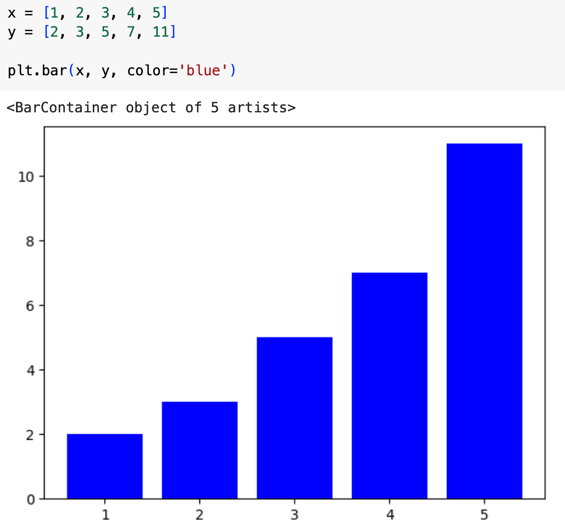
## Subplots:

You can create multiple subplots in a single figure using subplot() or subplots():

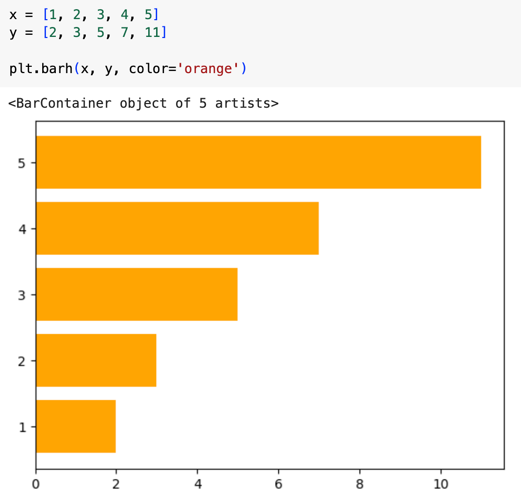


## Common Types of Plots Line Plot:

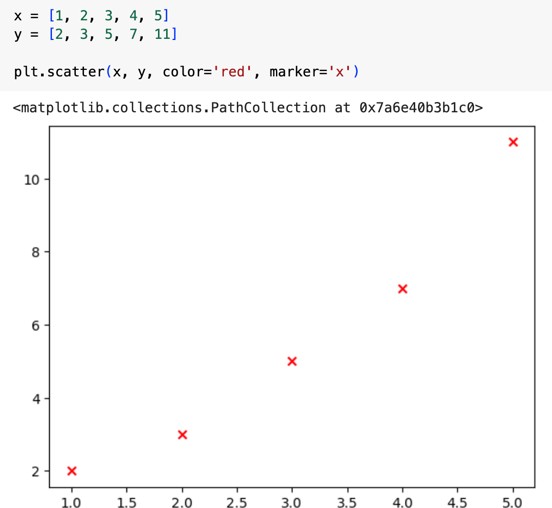
**Bar Plot:**



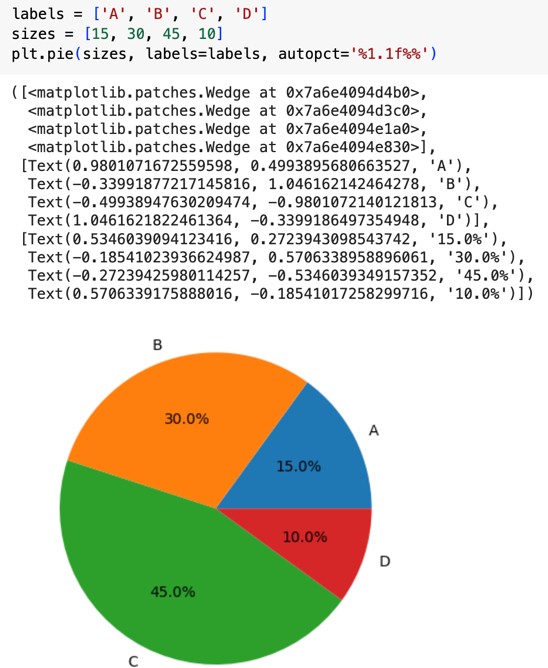
## Horizontal Bar Plot:



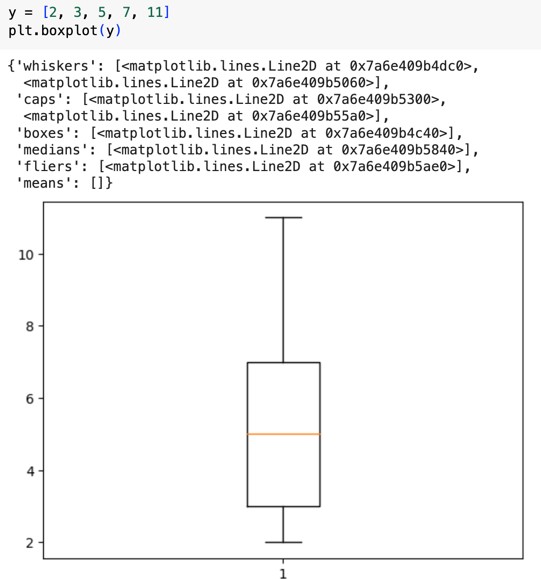
**Scatter Plot:**



## Pie Chart:



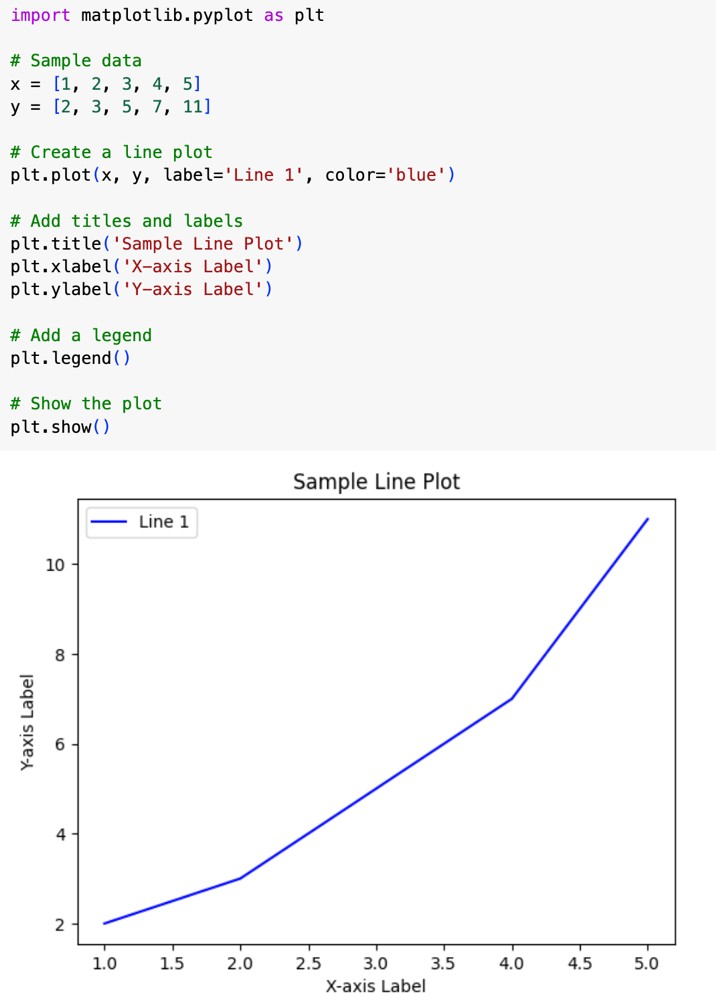
**Box Plot:**



## Customizations

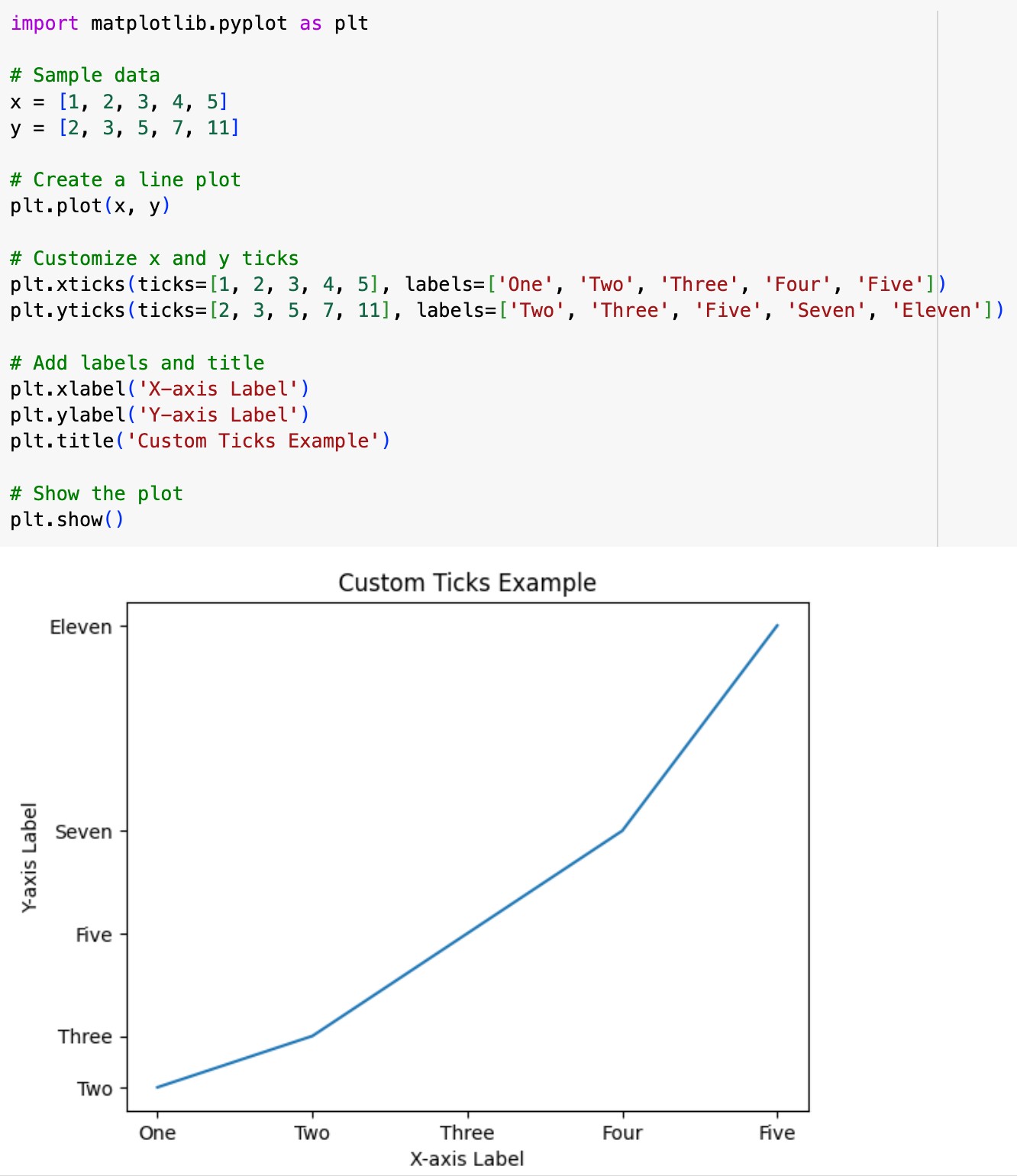
**Adding titles and labels:**

You can add titles, labels, and legends to your plots:



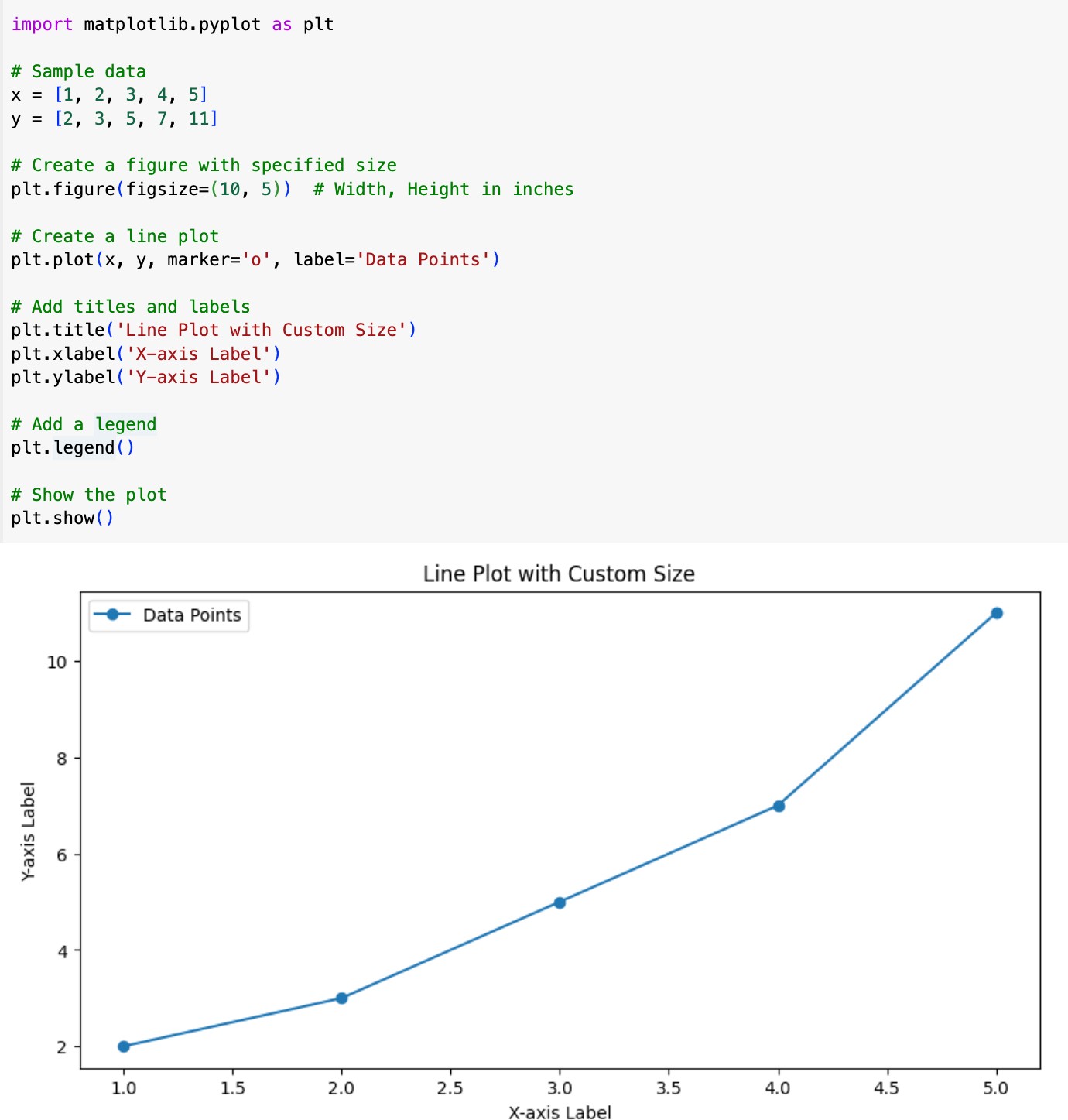
## Customizing Ticks:

You can customize the ticks on the axes, including their positions and labels, to make the data easier to interpret:



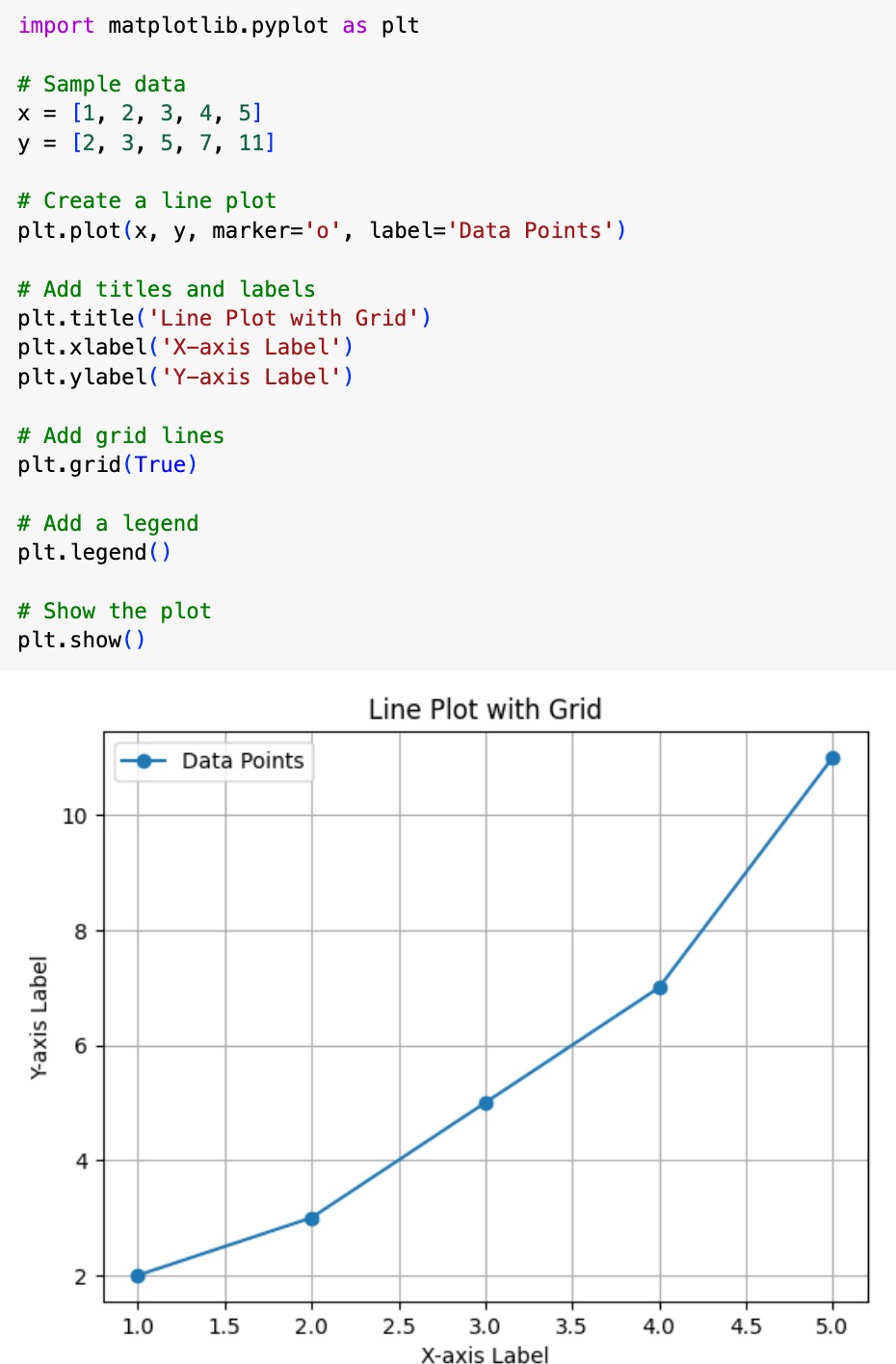
## Changing Figure Size:

You can specify the figure size when creating a figure, which allows for better layout control depending on your data visualization needs.



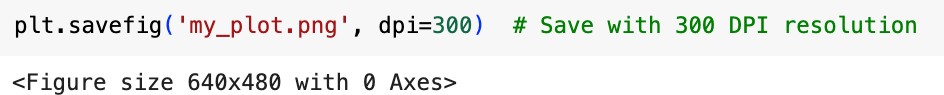
## Add Grid Lines:

Adding grid lines can improve readability by helping viewers trace values across the plot.



## Saving Your plotted Grid into Disk:

You can save your plots in various formats, such as PNG, PDF, or SVG. Increase the dpi to get more sharper image.



**Visualization Exercise:**

**Perform an Exploratory Data Analysis (EDA) on the provided Titanic dataset to gain insights into the passenger demographics and factors influencing survival rates.**

1. **Data Overview:**
   * What is the size of the dataset? How many columns and rows does it contain?
   * Are there any missing values in the dataset? If so, which columns have missing data and what percentage of data is missing?
2. **Passenger Demographics:**
   * What is the distribution of passengers across different passenger classes (Pclass)?
   * How is the age distribution of passengers? Are there any outliers?
   * What is the gender distribution among passengers?
3. **Survival Analysis:**
   * What is the overall survival rate of passengers?
   * Is there any relationship between passenger class and survival rate?
   * How does age correlate with survival? Are children more likely to survive?
   * Is there any significant difference in survival rates based on gender?
   * Does having family members (siblings/spouses or parents/children) aboard affect survival chances?
4. **Ticket Fare and Cabin:**
   * What is the distribution of ticket fares? Are there any outliers?
   * Is there a correlation between ticket fare and passenger class?
   * How does the availability of cabin data relate to survival rates?
5. **Embarkation Port:**
   * What is the distribution of passengers based on the port of embarkation?
   * Is there any correlation between the embarkation port and survival rates?
6. **Additional Insights:**
   * Are there any interesting patterns or insights that stand out in the data?
   * Can you identify any potential biases or limitations in the dataset?
7. **Visualization:**
   * Create visualizations (e.g., histograms, box plots, bar plots, etc.) to better understand the relationships between different variables and survival outcomes.