**SHADOWFOX INTERSHIP**

Task Level (Beginner)

Visualization Library Documentation

Introduction:

Data visualization is the graphical representation of data and information. It involves the process of creating visual representations of data. Data visualization can take many different forms, from basic charts and graphs. Basic charts and graphs are the simplest form of data visualization to represent numerical data. We need data visualization for several reasons. Data visualization helps us to easily understand complex datasets that might be difficult to comprehend in their raw form. It can highlight patterns, trends, and relationships that might not be immediately apparent from looking at the data. Visualization enables us to communicate insights and findings to stakeholders in a more compelling and understandable way. Data visualization is an essential tool for anyone dealing with data. Companies can gain insights into market trends, financial performance, and customer behavior. Healthcare professionals can identify patterns in patient data and develop targeted treatment plants. To create effective visualizations, it's important to follow best practices that can help ensure that the data is accurately represented and the message is clearly communicated. Data visualization is a vital tool for gaining insights, and effectively communicating complex information. In the world of data visualization, there are several plot libraries in Python that provide unique features and capabilities. Some of the popular libraries are Matplotlib, Pandas, Seaborn, Folium, Plotly, and PyWaffle. Each plot library has its own strengths and use cases. By harnessing the power of these plot libraries, we can unlock insights from data and effectively communicate our findings. Let's now explore the fifth plot library. If you want to present your data in an interactive way, plotly is a great choice to explore. It offers highly interactive plots and dashboards. With Plotly, we can create line plots, scatter plots, bar charts, pie charts, 3D plots, and Choropleth maps etc…

Objectives:

* To get know about Python visualization libraries: Matplotlib, Plotly, Bokeh.
* Implement data visualization techniques and plots using Python libraries.
* Create different types of charts and plots such as line, area, histograms, bar, pie, box, scatter, and bubble.
* Provide a brief overview of both libraries.
* Explain their relevance in data visualization.
* Highlight their key features and use cases.
* Describe the variety of charts each library can generate.
* Include practical examples with code snippets for common chart types (e.g., line plots, scatter plots, bar charts, pie charts).
* Emphasize how both libraries seamlessly integrate with pandas dataframes.
* Show examples of loading data from pandas and creating visualizations.

**Library Overview:**

Plotly

It’s is a powerful graphing library that supports interactive, publication-quality visualizations. It is available for Python, R, MATLAB, and JavaScript, making it versatile for various programming environments.

Unique Features:

* Interactivity: Plotly is renowned for its ability to create highly interactive plots that can be embedded in web applications. Users can zoom, pan, hover, and click to interact with the data.
* Wide Range of Chart Types: It supports a vast array of chart types, including basic plots (like line, bar, and scatter), complex 3D charts, and specialized visualizations like geographical maps and financial charts.
* Ease of Use: Plotly’s syntax is straightforward, making it easy to convert data into stunning visualizations with minimal code.
* Integration: It integrates well with popular web frameworks like Dash, enabling the creation of full-fledged data-driven web applications.

Typical Use Cases:

* Dashboard Development: Frequently used for creating interactive dashboards that allow users to explore data in real-time.
* Data Exploration and Analysis: Useful in scenarios where interactive data exploration is required, such as in Jupyter Notebooks.
* Presentations and Reports: Ideal for creating engaging, interactive charts that can be shared in reports and presentations.

**Simple Example for Plotly Library:**

import plotly.graph\_objects as go

# Sample data

x\_values = [1, 2, 3, 4, 5]

y\_values = [10, 11, 12, 13, 14]

#To Create a line plot

fig = go.Figure()

#trace

fig.add\_trace(go.Scatter(x=x\_values, y=y\_values, mode='lines', name='Line Plot'))

#title and labels

fig.update\_layout(

title='Simple Line Plot',

xaxis\_title='X Axis',

yaxis\_title='Y Axis'

)

#plot

fig.show()

Bokeh:

It’s is a Python library specialized in creating interactive and versatile visualizations for web applications. It provides elegant and high-performance charts and plots.

Unique Features:

* Interactivity: Like Plotly, Bokeh excels in creating interactive visualizations. Users can interact with the data through zooming, panning, and hover tools.
* High-Level and Low-Level Interfaces: Bokeh offers both a high-level interface for quickly generating common plot types and a low-level interface for creating complex, custom visualizations.
* Real-Time Data: It supports streaming and real-time data updates, making it suitable for live data applications.
* Server Capabilities: Bokeh can be used to create interactive web applications through its Bokeh server, which allows for the building of rich dashboards and data applications.

Typical Use Cases:

* Interactive Web Applications: Often used to create interactive plots within web applications, particularly those requiring real-time data interaction.
* Data Science and Analysis: Employed in exploratory data analysis, where users need to interact with data to uncover insights.
* Financial and Scientific Visualization: Used in domains that require detailed and interactive visualizations, such as finance and scientific research.

**Simple Example for Bokeh Library:**

from bokeh.plotting import figure, show, output\_file

from bokeh.models import HoverTool

from bokeh.io import curdoc

# Sample data

x = [1, 2, 3, 4, 5]

y = [6, 7, 2, 4, 5]

# To Create a figure object

p = figure(title="Simple Line Plot with Hover Tool", x\_axis\_label='X-Axis', y\_axis\_label='Y-Axis')

#To Add a line renderer with legend and line thickness

p.line(x, y, legend\_label="Sample Line", line\_width=2)

#To Add circles at data points

p.circle(x, y, size=10, color="navy", alpha=0.5)

# To Add hover tool

hover = HoverTool()

hover.tooltips = [("Index", "$index"), ("(x,y)", "($x, $y)")]

p.add\_tools(hover)

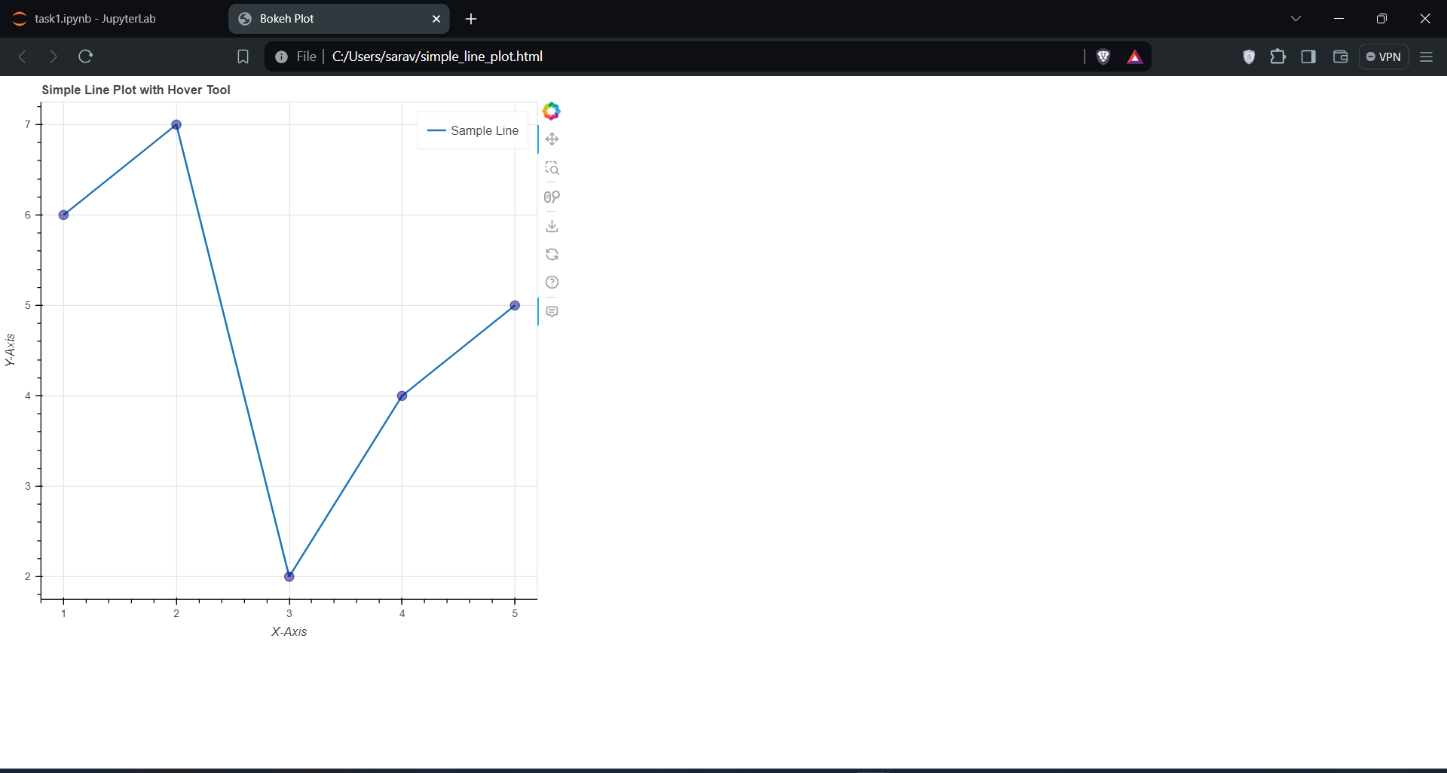
# Output to a static HTML file

output\_file("simple\_line\_plot.html")

# Show the results

show(p)

|  |
| --- |
| **Note:**  **Adding Hover Tool:**  **hover = HoverTool() initializes the hover tool.**  **hover.tooltips specifies what information to display when hovering over data points.**  **Output File and Display:**  **output\_file("simple\_line\_plot.html") specifies the output HTML file.**  **show(p) displays the plot in a web browser.**  **Bokeh Server:**  **If you want to serve the plot using a Bokeh server, you can add the figure to the current document with curdoc().add\_root(p) and run the script with bokeh serve --show script\_name.py.**  **This example creates a simple line plot with interactive hover information, which you can save as an HTML file and view in a web browser.** |

**Output of Bokeh:**

**Graph Types:**

The Implementation of different types graphs using Plotly library, such as line plots, scatter plots, bar charts, histograms, pie charts, etc.

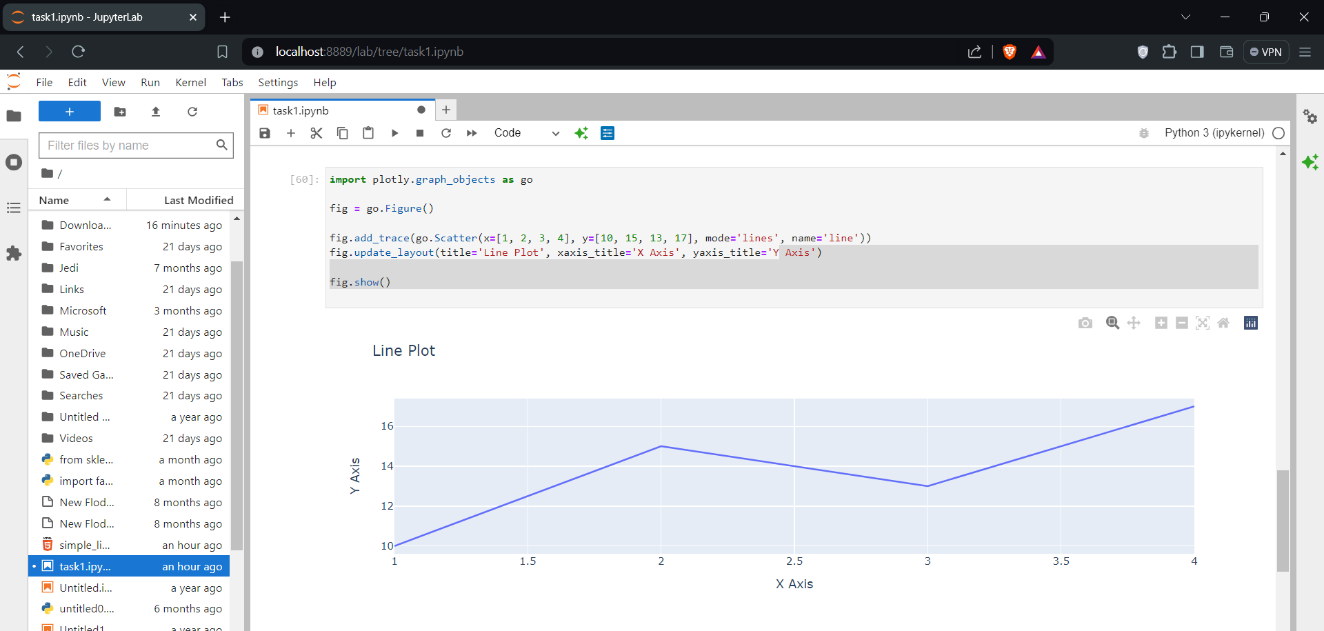
**Plotly:**

**1. Line Plots**

**Description: Line plots display data points connected by straight lines, often used to visualize trends over time.**

**Use Case: Tracking stock prices over time, monitoring website traffic.**

|  |
| --- |
| import plotly.graph\_objects as go  fig = go.Figure()  fig.add\_trace(go.Scatter(x=[1, 2, 3, 4], y=[10, 15, 13, 17], mode='lines', name='line'))  fig.update\_layout(title='Line Plot', xaxis\_title='X Axis', yaxis\_title='Y Axis')  fig.show() |

Output

**2. Scatter Plots**

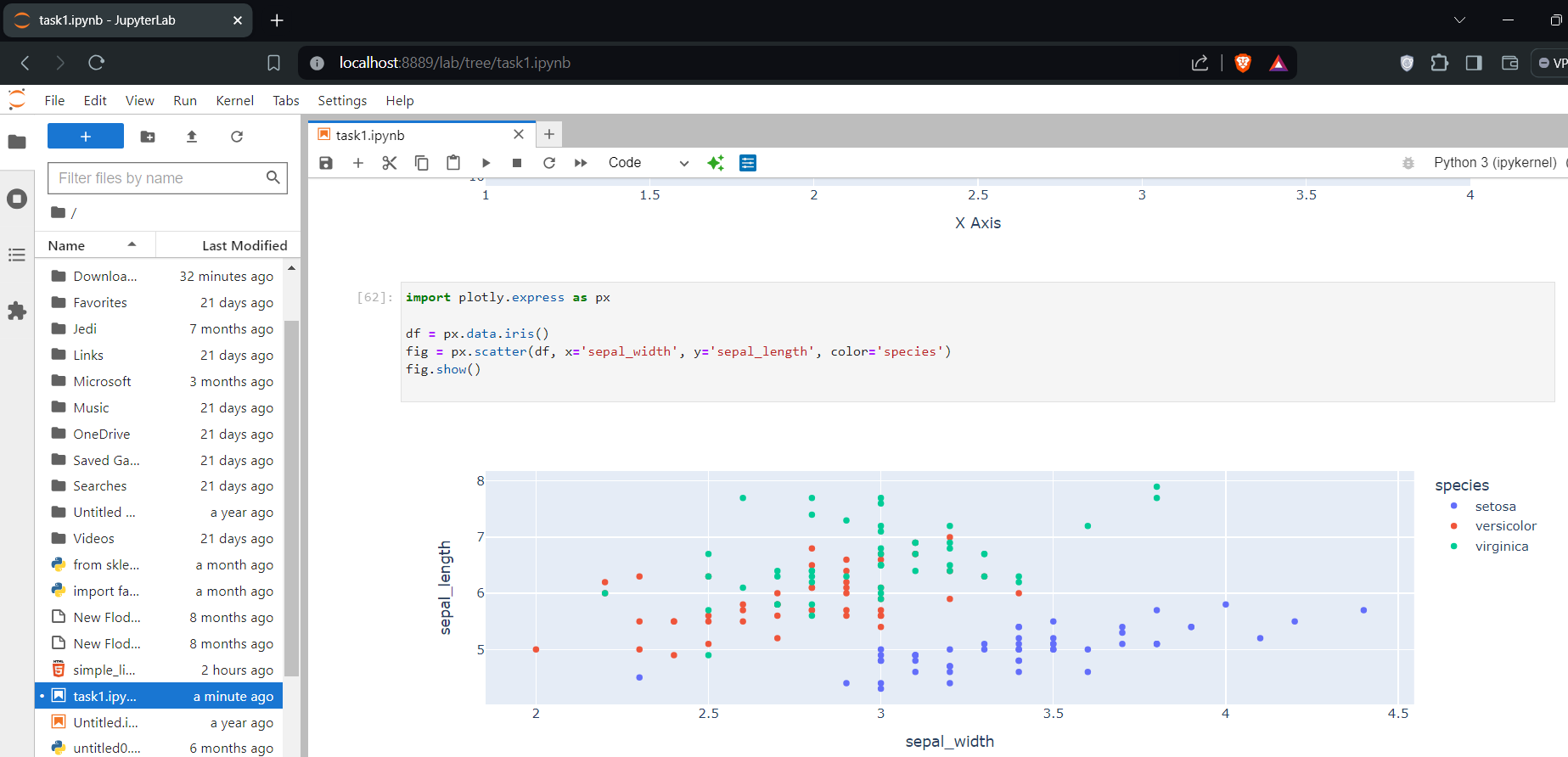
Description: Scatter plots display individual data points, often used to identify relationships between variables.

Use Case: Comparing two variables to see if they correlate, such as height and weight.

**Code Example:**

|  |
| --- |
| **import plotly.express as px**  **df = px.data.iris()**  **fig = px.scatter(df, x='sepal\_width', y='sepal\_length', color='species')**  **fig.update\_layout(title='Scatter Plot')**  **fig.show()** |

**Output:**

****

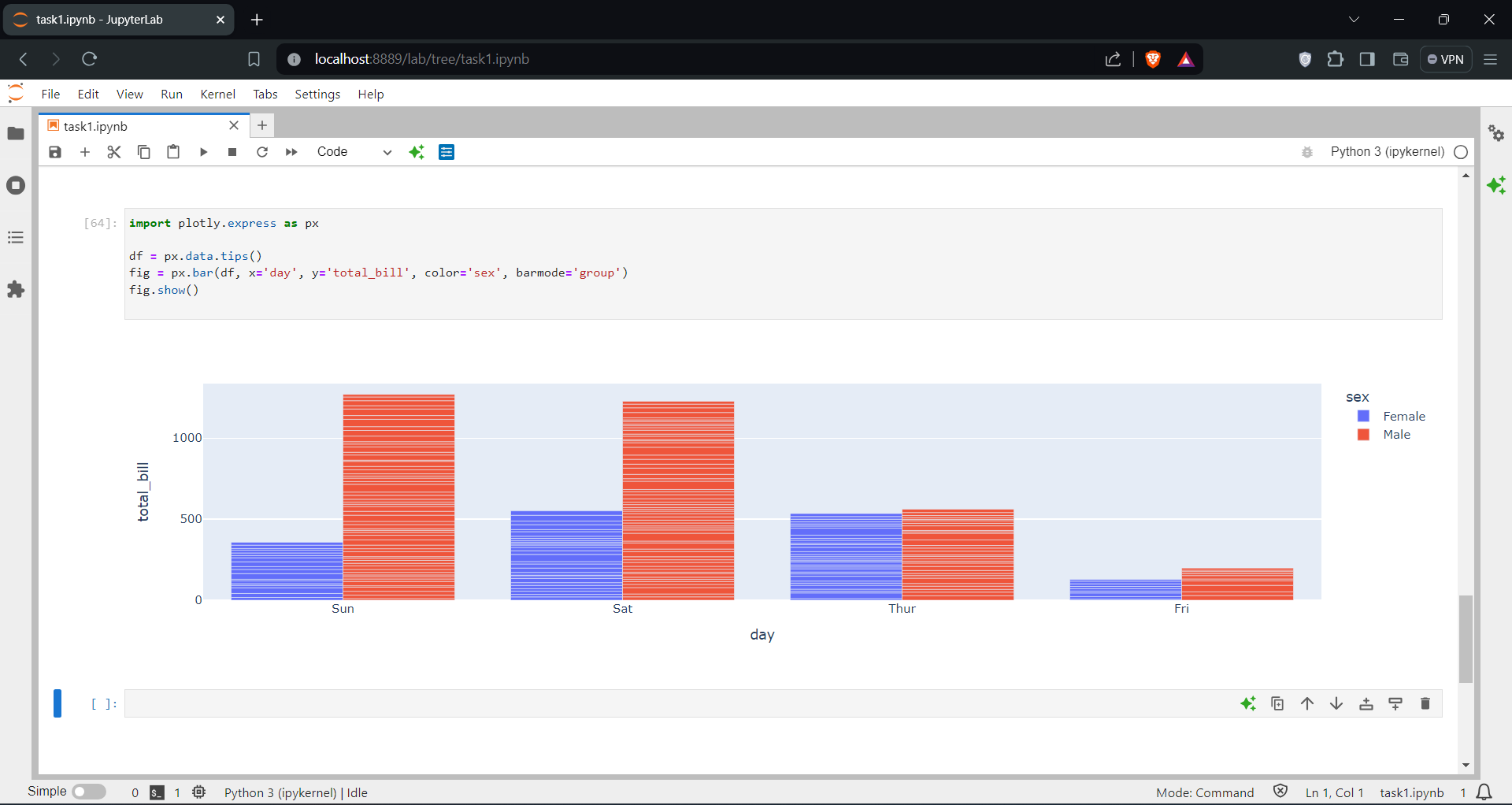
**3. Bar Chart**

**Description:** A bar chart presents categorical data with rectangular bars.

Use Case: Comparing quantities across categories.

**Code Example**:

|  |
| --- |
| import plotly.express as px  df = px.data.tips()  fig = px.bar(df, x='day', y='total\_bill', color='sex', barmode='group')  fig.show() |



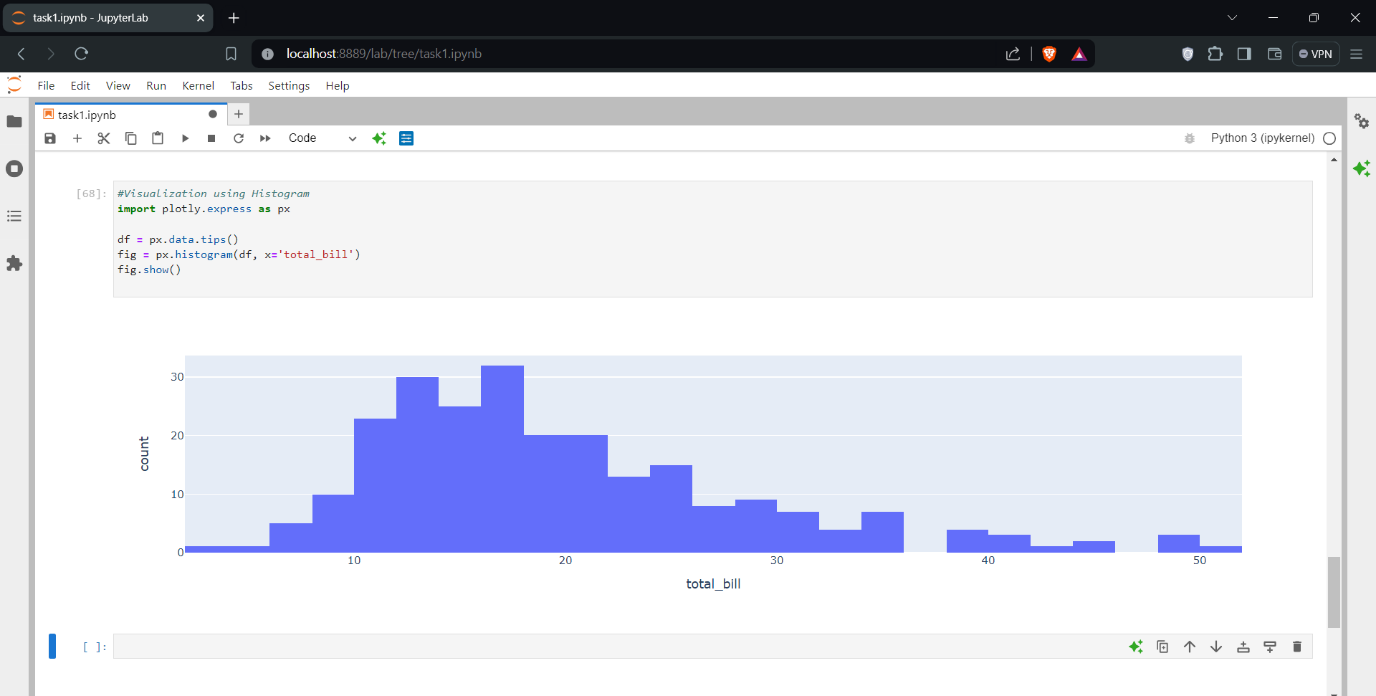
4. Histogram

Description: A histogram represents the distribution of numerical data.

Use Case: Visualizing the frequency distribution of a dataset.

**Code Example:**

|  |
| --- |
| import plotly.express as px  df = px.data.tips()  fig = px.histogram(df, x='total\_bill')  fig.show() |



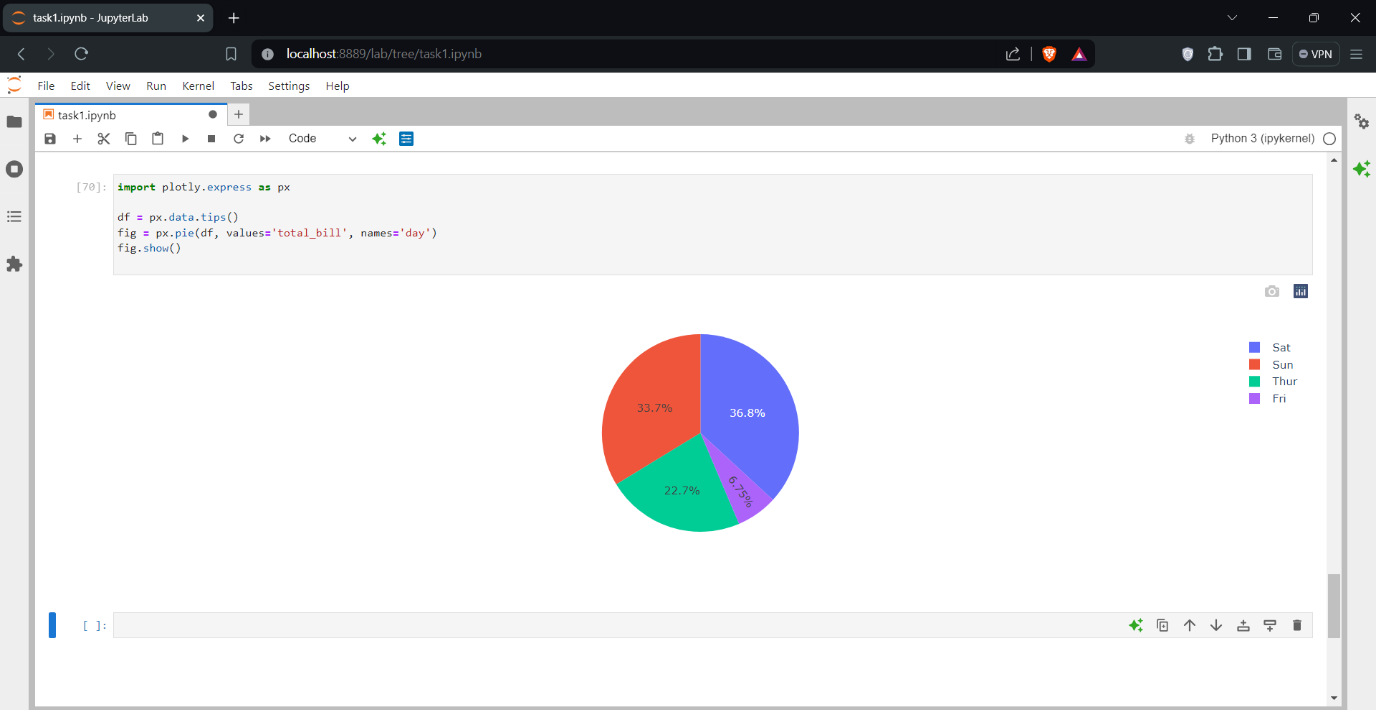
**5. Pie Chart**

**Description:** A pie chart displays data as slices of a circular pie.

Use Case: Showing proportions of a whole.

**Code Example:**

|  |
| --- |
| import plotly.express as px  df = px.data.tips()  fig = px.pie(df, values='total\_bill', names='day')  fig.show() |



**Bokeh**

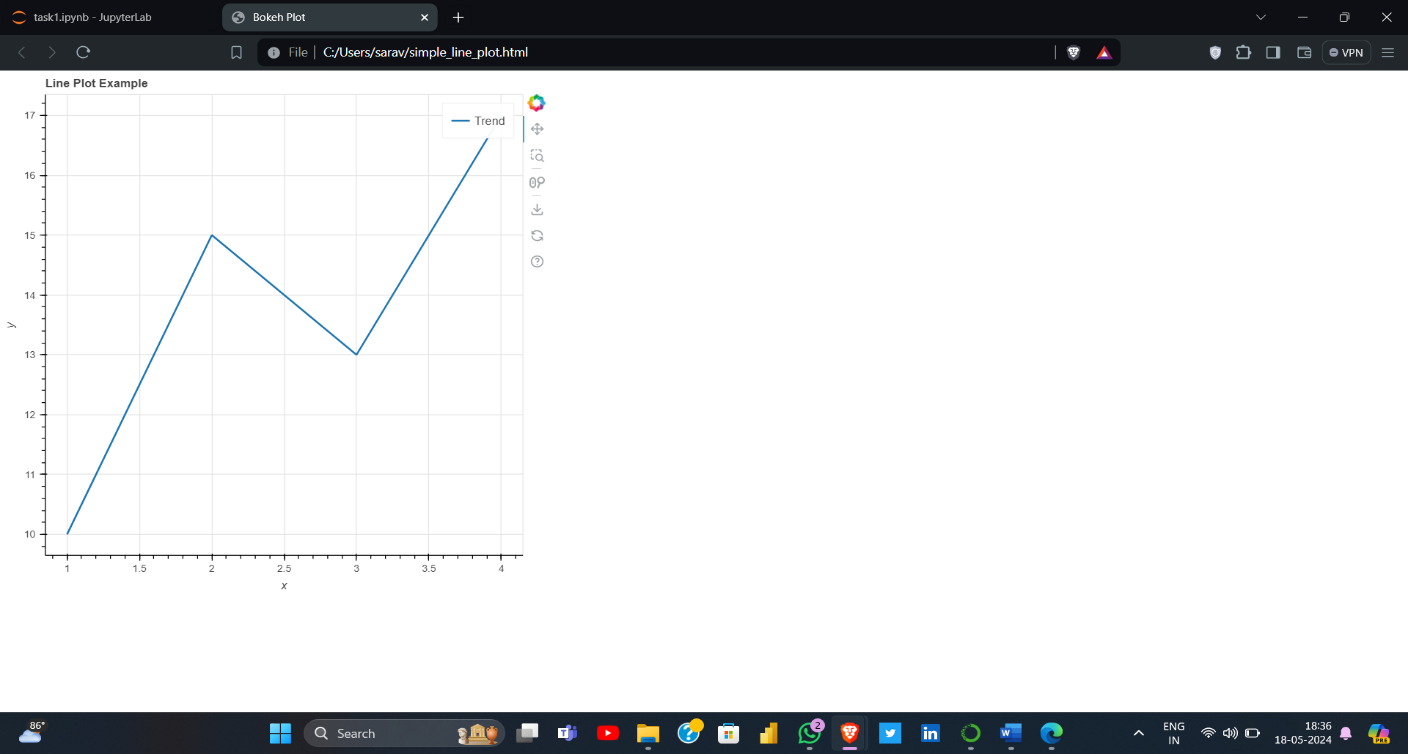
1. Line Plot

Description: A line plot displays information as a series of data points connected by straight line segments.

Use Case: Visualizing trends over time.

Code Example:

|  |
| --- |
| from bokeh.plotting import figure, show  p = figure(title="Line Plot Example", x\_axis\_label='x', y\_axis\_label='y')  p.line(x=[1, 2, 3, 4], y=[10, 15, 13, 17], legend\_label="Trend", line\_width=2)  show(p) |



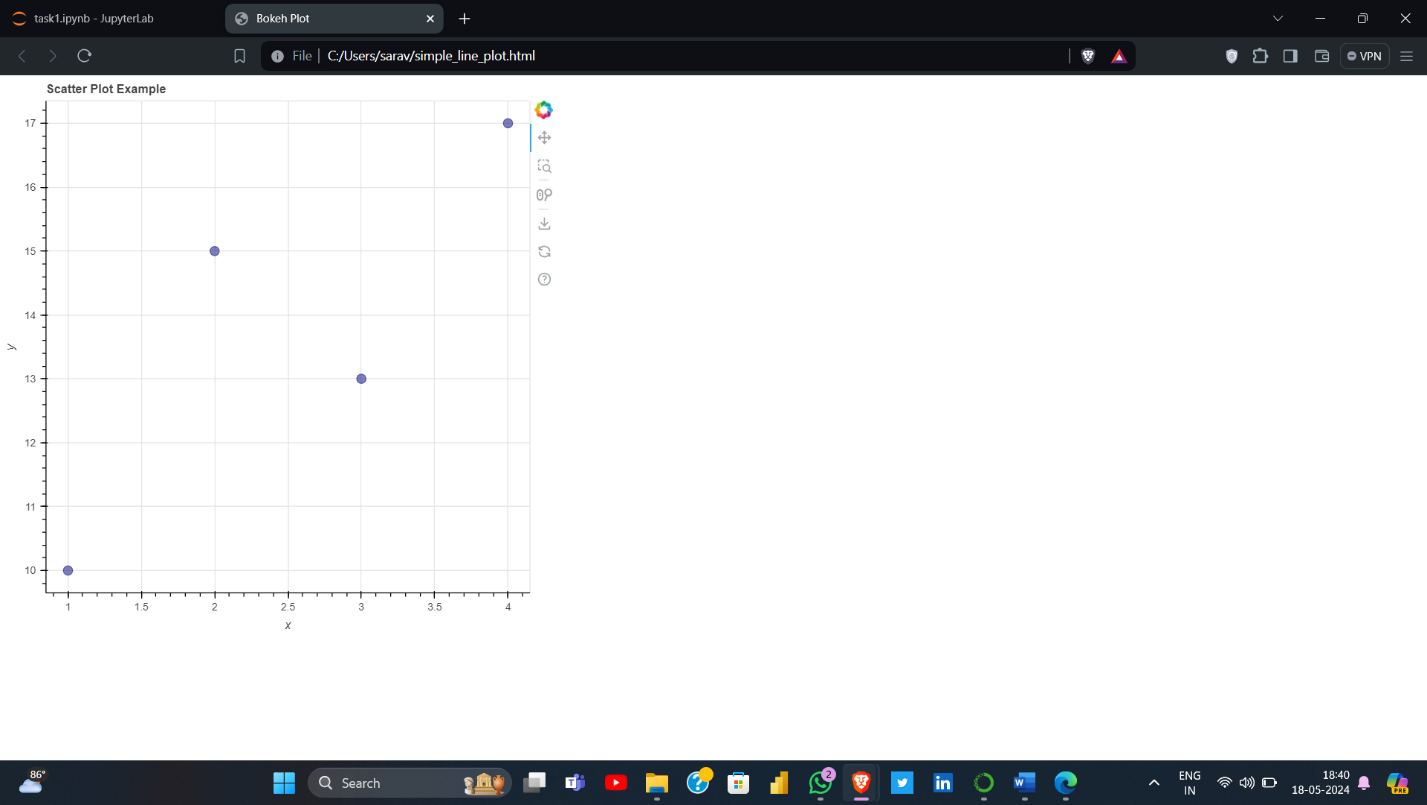
2. Scatter Plot

Description: A scatter plot displays values for two variables for a set of data points.

Use Case: Analyzing relationships between variables.

Code Example:

|  |
| --- |
| from bokeh.plotting import figure, show  p = figure(title="Scatter Plot Example", x\_axis\_label='x', y\_axis\_label='y')  p.circle(x=[1, 2, 3, 4], y=[10, 15, 13, 17], size=10, color="navy", alpha=0.5)  show(p) |



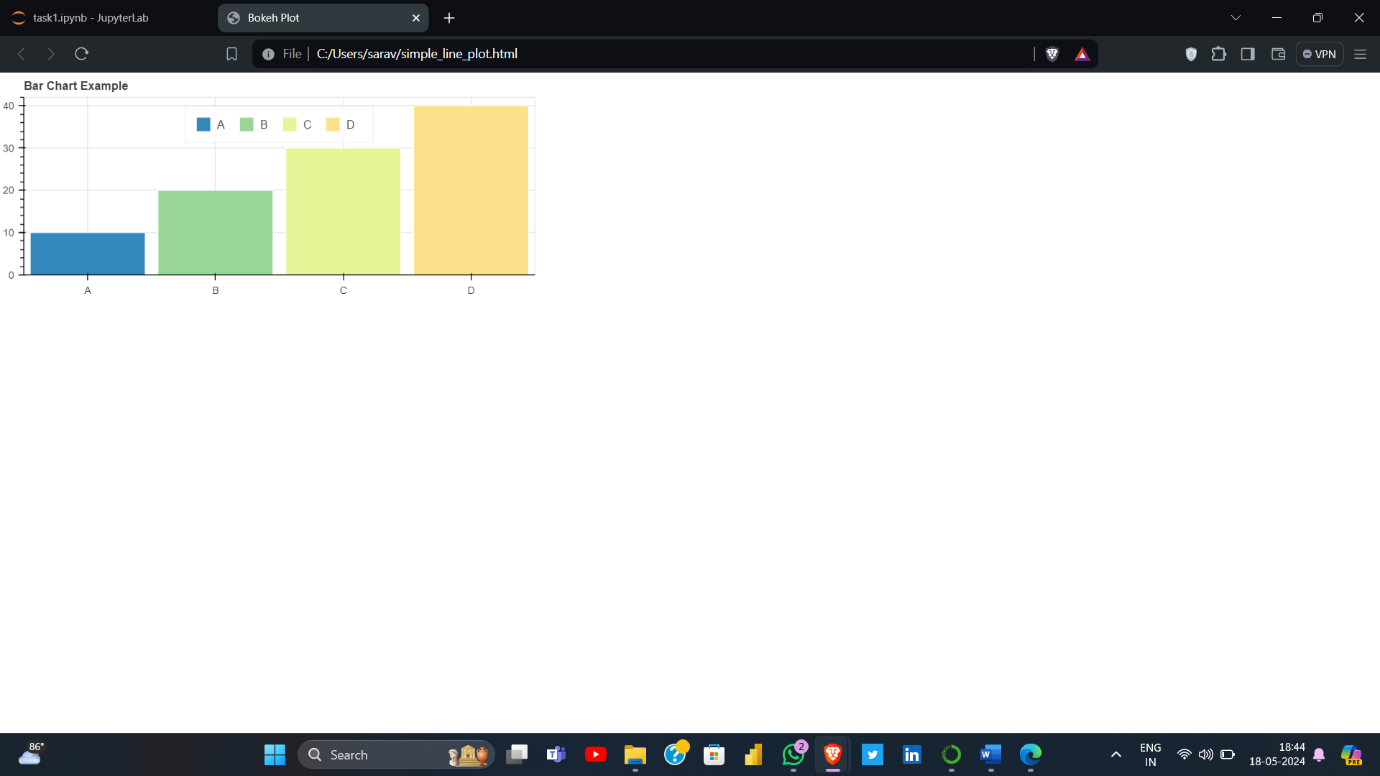
3. Bar Chart

Description: A bar chart presents categorical data with rectangular bars.

Use Case: Comparing quantities across categories.

Code Example:

|  |
| --- |
| from bokeh.plotting import figure, show  from bokeh.transform import factor\_cmap  from bokeh.io import output\_notebook  from bokeh.palettes import Spectral6  output\_notebook()  categories = ['A', 'B', 'C', 'D']  values = [10, 20, 30, 40]  p = figure(x\_range=categories, height=250, title="Bar Chart Example", toolbar\_location=None, tools="")  p.vbar(x=categories, top=values, width=0.9, legend\_field="x", line\_color='white', fill\_color=factor\_cmap('x', palette=Spectral6, factors=categories))  p.y\_range.start = 0  p.legend.orientation = "horizontal"  p.legend.location = "top\_center"  show(p) |



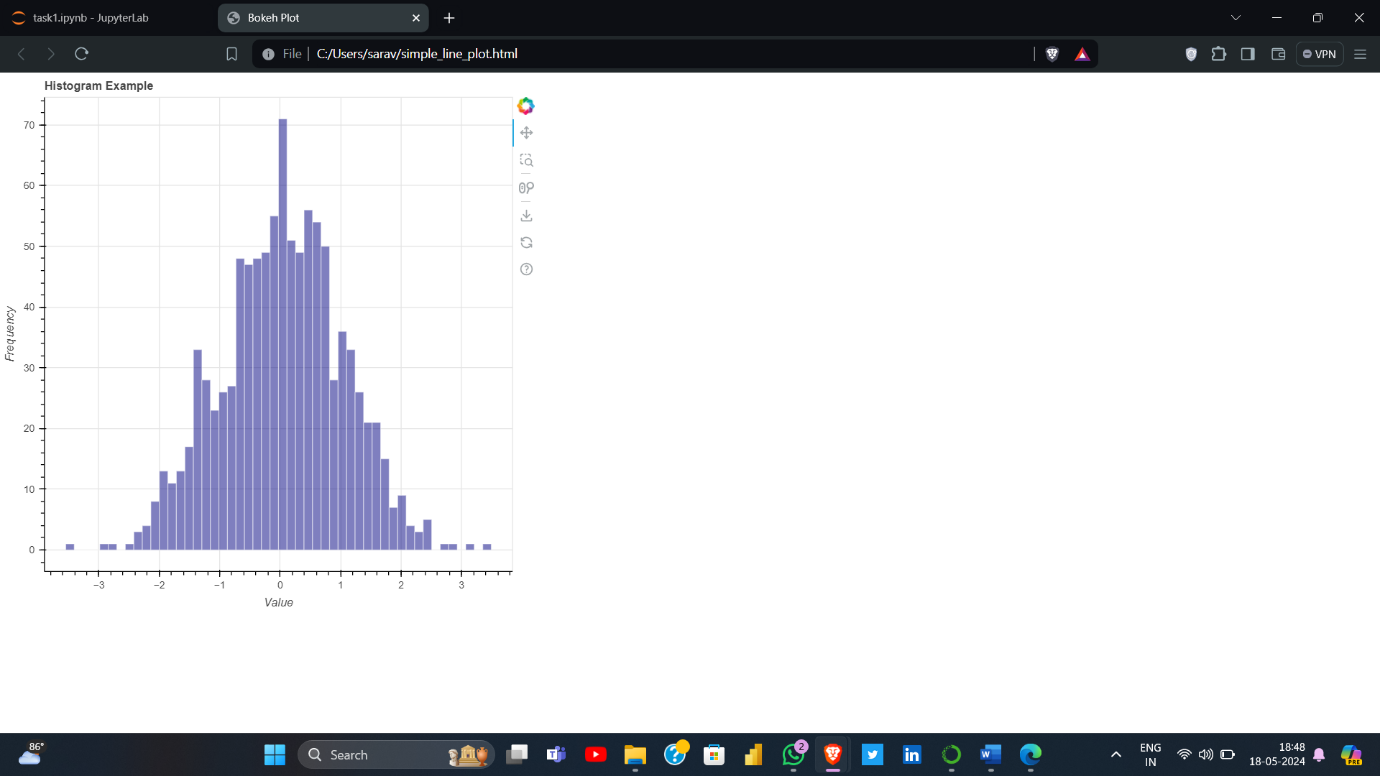
4. Histogram

Description: A histogram represents the distribution of numerical data.

Use Case: Visualizing the frequency distribution of a dataset.

Code Example:

|  |
| --- |
| import numpy as np  from bokeh.plotting import figure, show  data = np.random.randn(1000)  hist, edges = np.histogram(data, bins=50)  p = figure(title="Histogram Example", x\_axis\_label='Value', y\_axis\_label='Frequency')  p.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:], fill\_color="navy", line\_color="white", alpha=0.5)  show(p) |



5. Pie Chart

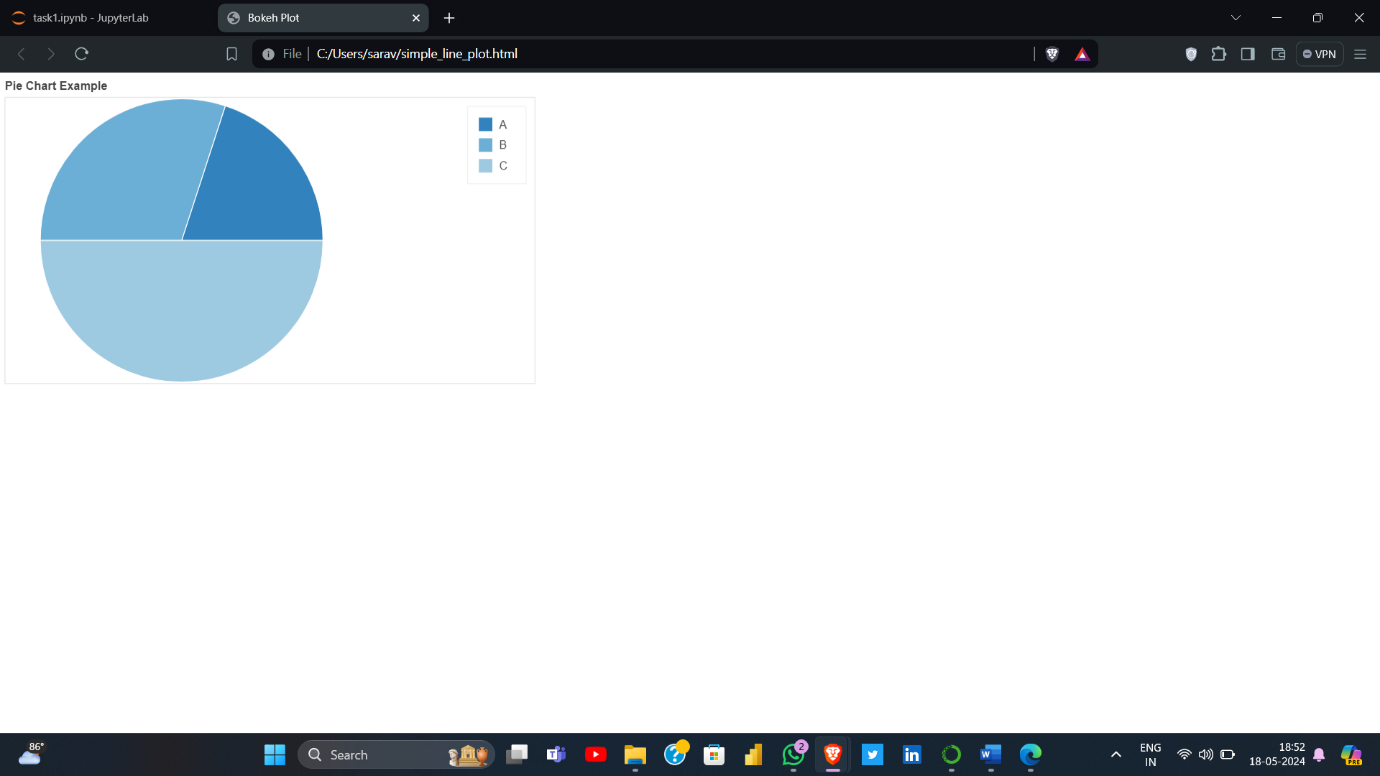
Description: A pie chart displays data as slices of a circular pie.

Use Case: Showing proportions of a whole.

Note: Bokeh doesn't directly support pie charts, but you can use wedges to create a pie chart.

Code Example:

|  |
| --- |
| from math import pi  from bokeh.palettes import Category20c  from bokeh.plotting import figure, show  from bokeh.transform import cumsum  import pandas as pd  x = pd.Series([20, 30, 50], index=["A", "B", "C"]).reset\_index(name='value').rename(columns={'index': 'category'})  x['angle'] = x['value'] / x['value'].sum() \* 2 \* pi  x['color'] = Category20c[len(x)]  p = figure(height=350, title="Pie Chart Example", toolbar\_location=None, tools="hover", tooltips="@category: @value", x\_range=(-0.5, 1.0))  p.wedge(x=0, y=1, radius=0.4, start\_angle=cumsum('angle', include\_zero=True), end\_angle=cumsum('angle'), line\_color="white", fill\_color='color', legend\_field='category', source=x)  p.axis.axis\_label = None  p.axis.visible = False  p.grid.grid\_line\_color = None  show(p) |



**Comparison**

**Outline:**

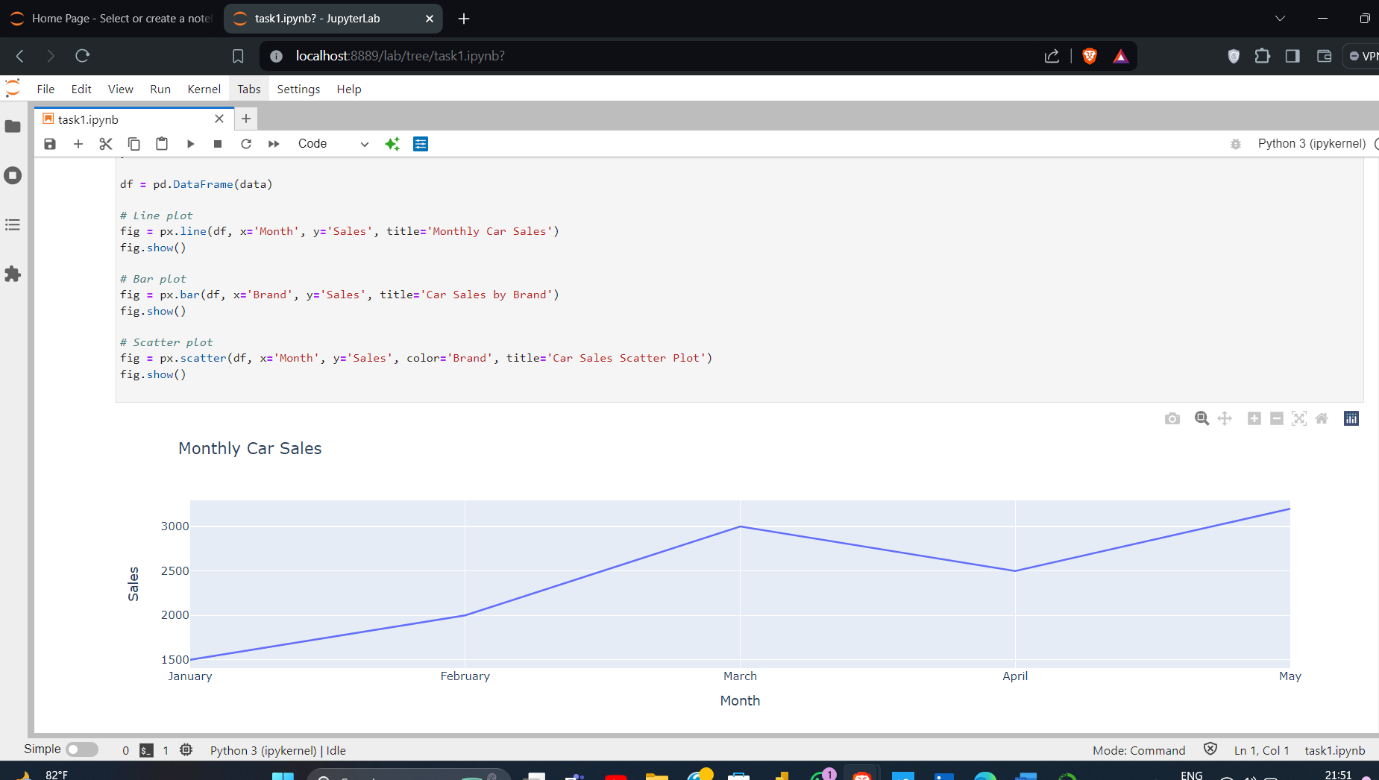
|  |
| --- |
| * **Ease of Use** * **Customization Options** * **Interactivity** * **Performance with Large Datasets** * **Code and Visualization** |

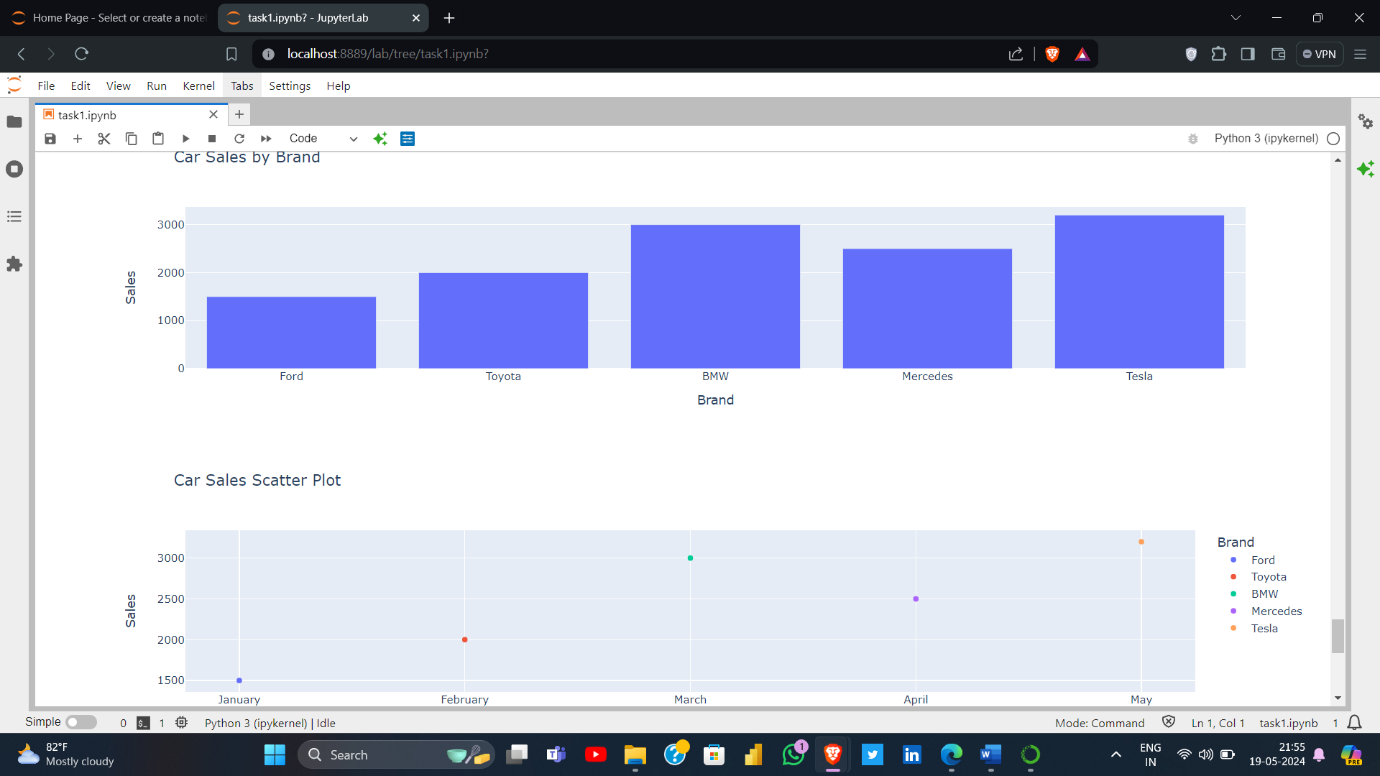
1. **Comparison of Plotly Library**

* **Ease of use:**
* **Strengths: Plotly is known for its user-friendly API. It has a simple syntax that makes it easy to create complex visualizations quickly. The documentation is comprehensive and there are numerous examples available online.**
* **Weaknesses: Plotly can sometimes be overwhelming for beginners due to the multitude of options and settings. The learning curve can be steep when diving into more advanced features.**
* **Customization Options:**
* **Strengths: Offers extensive customization options for almost every aspect of a plot. Users can easily customize the layout, axes, markers, colors, and annotations.**
* **Weaknesses: The extensive customization options can be daunting and lead to verbose code. Sometimes finding the right attribute to customize can be challenging.**
* **Interactivity:**
* **Strengths: Highly interactive out of the box. It supports zooming, panning, tooltips, and more without additional configuration.**
* **Weaknesses: For extremely large datasets, interactivity can become sluggish.**
* **Performance with Large Datasets**
* **Strengths: Can handle moderately large datasets efficiently. It leverages WebGL for rendering, which can significantly improve performance for large datasets.**
* **Weaknesses: Performance can degrade with very large datasets, especially if there are many interactive features.**
* **Code and Visualization**

**Let's create visualizations for Plotly libraries using a hypothetical "carsales" dataset.**

|  |
| --- |
| **import plotly.express as px**  **import pandas as pd**  **# Example dataset**  **data = {**  **'Month': ['January', 'February', 'March', 'April', 'May'],**  **'Sales': [1500, 2000, 3000, 2500, 3200],**  **'Brand': ['Ford', 'Toyota', 'BMW', 'Mercedes', 'Tesla']**  **}**  **df = pd.DataFrame(data)**  **# Line plot**  **fig = px.line(df, x='Month', y='Sales', title='Monthly Car Sales')**  **fig.show()**  **# Bar plot**  **fig = px.bar(df, x='Brand', y='Sales', title='Car Sales by Brand')**  **fig.show()**  **# Scatter plot**  **fig = px.scatter(df, x='Month', y='Sales', color='Brand', title='Car Sales Scatter Plot')**  **fig.show()** |

**Line Plot**

Bar Chart & Scatter Plot

Note:

**For visualization, run the code snippets at any python integrated environment. Plotly's visualizations will appear as interactive charts in your browser or notebook**.

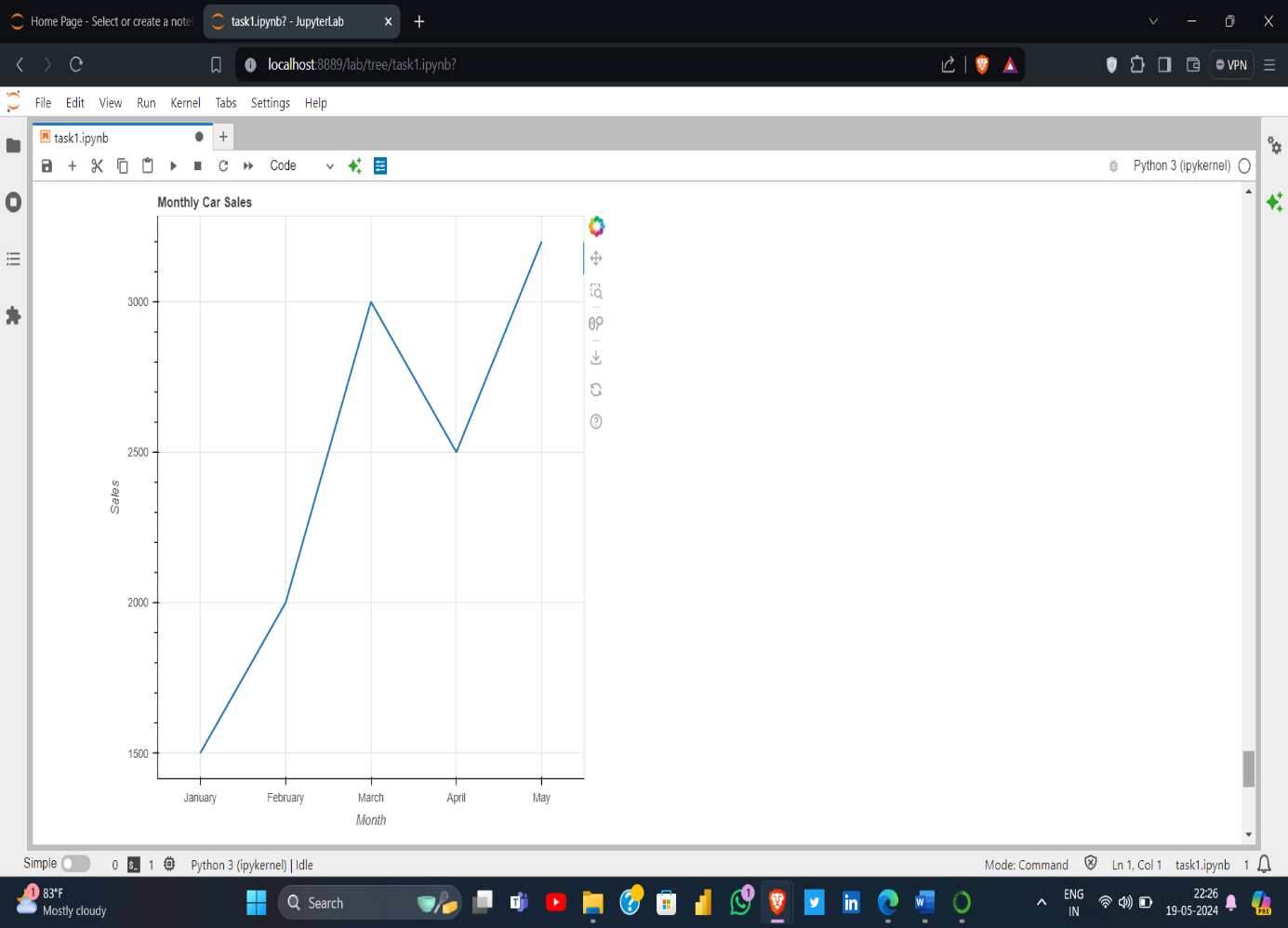
1. **Comparison of Bokeh Library**

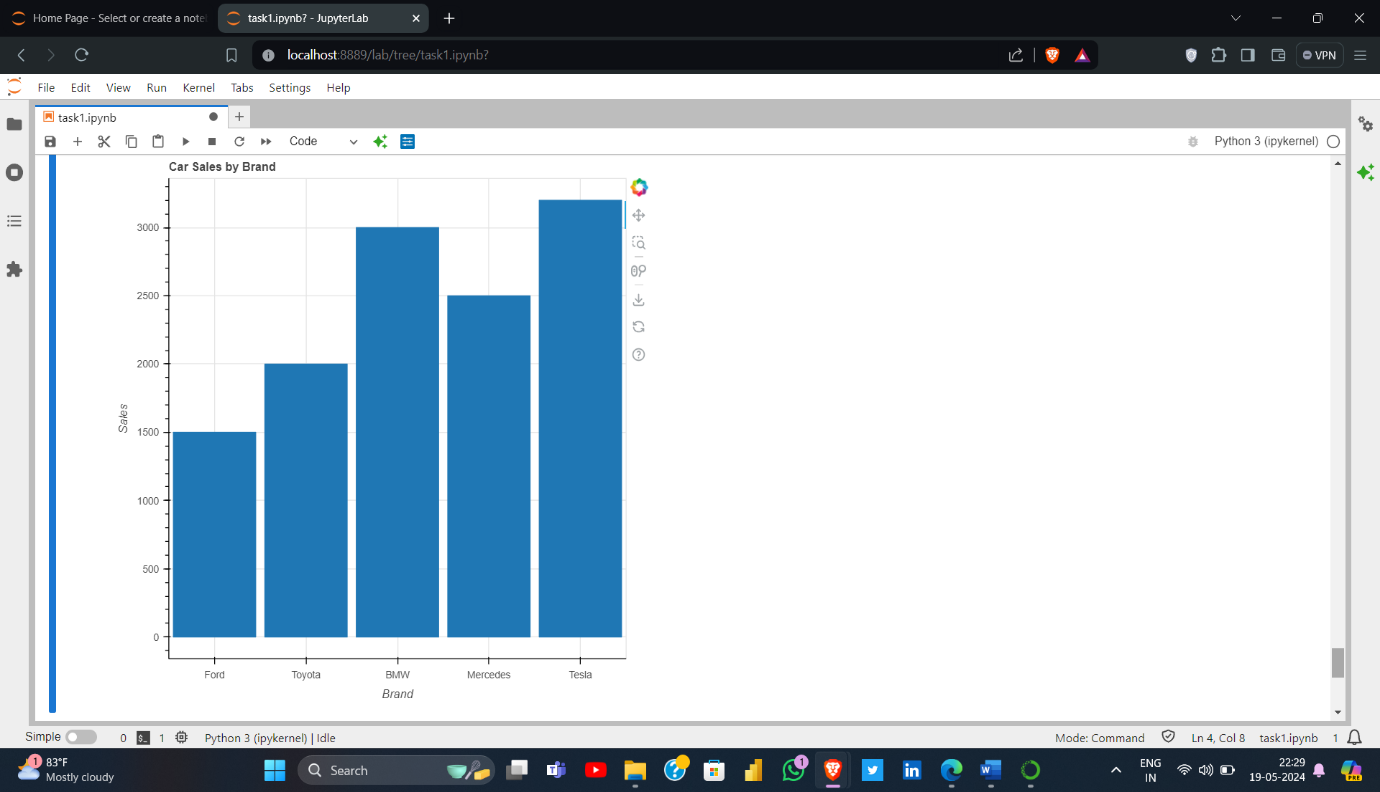
* **Ease of use**
* **Strengths: Bokeh is also relatively easy to use, with an intuitive interface for basic plots. It integrates well with Jupyter Notebooks, making it convenient for data exploration.**
* **Weaknesses: Bokeh can be less intuitive for creating complex, highly customized visualizations. The documentation, while good, is not as extensive as Plotly’s, and some features require a deeper understanding of the library’s architecture.**
* **Customization Options**
* **Strengths: Provides robust customization options, particularly for plots embedded in web applications. Its use of a declarative syntax makes it easier to manage complex customizations.**
* **Weaknesses: Customizing certain plot aspects can be less straightforward compared to Plotly. Advanced customizations might require more in-depth knowledge of Bokeh’s API.**
* **Interactivity**
* **Strengths: Excellent interactivity, particularly for custom and highly specific interactive behaviors. It’s designed with web-based applications in mind, making it powerful for interactive dashboards.**
* **Weaknesses: Setting up interactivity can be more complex and may require writing custom JavaScript for advanced interactions.**
* **Performance with Large Datasets**
* **Designed to handle large and streaming datasets well. It uses efficient rendering techniques and can handle large datasets more gracefully.**
* **While it handles large datasets better than Plotly, there might still be performance issues with extremely large data volumes, particularly in highly interactive plots.**
* **Code and Visualization**

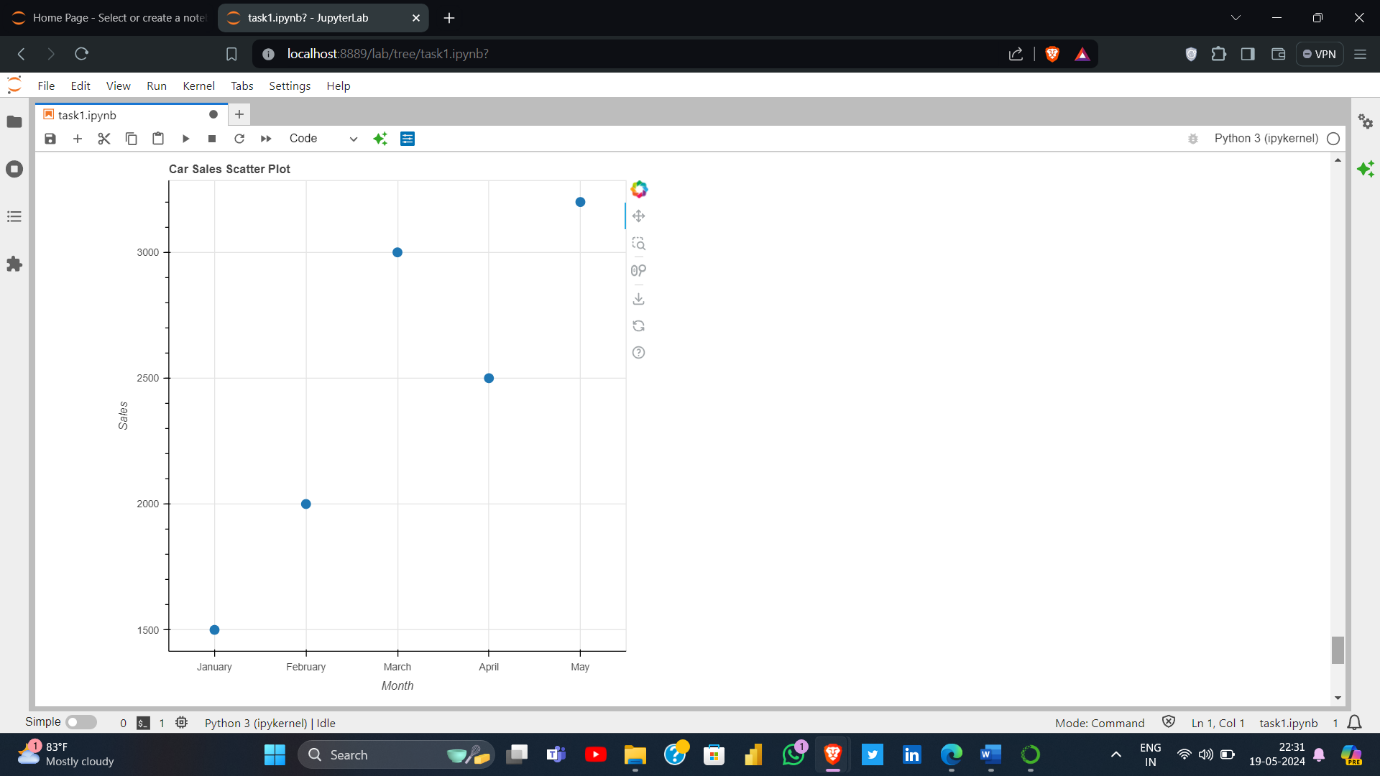
**Let's create example visualizations for Bokeh libraries using a hypothetical "carsales" dataset.**

|  |
| --- |
| **from bokeh.plotting import figure, show**  **from bokeh.models import ColumnDataSource**  **from bokeh.io import output\_notebook**  **import pandas as pd**  **output\_notebook()**  **# Example dataset**  **data = {**  **'Month': ['January', 'February', 'March', 'April', 'May'],**  **'Sales': [1500, 2000, 3000, 2500, 3200],**  **'Brand': ['Ford', 'Toyota', 'BMW', 'Mercedes', 'Tesla']**  **}**  **df = pd.DataFrame(data)**  **source = ColumnDataSource(df)**  **# Line plot**  **p = figure(title='Monthly Car Sales', x\_axis\_label='Month', y\_axis\_label='Sales', x\_range=df['Month'])**  **p.line(x='Month', y='Sales', source=source, line\_width=2)**  **show(p)**  **# Bar plot**  **p = figure(x\_range=df['Brand'], title='Car Sales by Brand', x\_axis\_label='Brand', y\_axis\_label='Sales')**  **p.vbar(x='Brand', top='Sales', source=source, width=0.9)**  **show(p)**  **# Scatter plot**  **p = figure(title='Car Sales Scatter Plot', x\_axis\_label='Month', y\_axis\_label='Sales', x\_range=df['Month'])**  **p.scatter(x='Month', y='Sales', source=source, size=10)**  **show(p)** |

* **Visualization of graphs**
* **Line graph**



**Bar chart**

**Scatter Plot**

**Conclusion**

**Both Plotly and Bokeh are powerful tools for data visualization, each with its strengths and weaknesses. Plotly is generally easier to use and offers a wide range of customization and interactivity features out of the box, but can struggle with very large datasets. Bokeh, while potentially more complex to customize and make interactive, excels with large datasets and provides robust tools for embedding plots in web applications. Your choice between the two will depend on your specific needs, including ease of use, the level of customization required, and the size of the datasets you are working with.**

|  |
| --- |
| **Note:**  **Performed of Task link:** [**http://localhost:8888/lab/tree/task(Beginner).ipynb**](http://localhost:8888/lab/tree/task(Beginner).ipynb) |