BEGINNER FRIENDLY MANUAL FOR MATPLOTLIB AND PLOTLY

What is Matplotlib?

Here's general definition:

->Matplotlib is a powerful, versatile, and widely-used Python library for creating static, interactive, and animated visualizations. It provides a MATLAB-like interface for plotting data in various formats (line plots, scatter plots, bar charts, histograms, etc.) with extensive customization options.

Key Features:

Flexibility: Fine-tune every aspect of plots (colors, labels, grids, annotations).

Backend Support: Works across different environments (Jupyter, scripts, GUIs).

Integration: Plays well with NumPy, Pandas, and other scientific Python libraries.

Publication-Quality Output: Export plots in multiple formats (PNG, PDF, SVG, etc.).

View of Matplotlib in interesting way:

"Matplotlib is like the Swiss Army knife of Python plotting—it can whip up anything from quick-and-dirty bar charts to publication-ready masterpieces, though sometimes you'll wrestle with its quirks like trying to fold a fitted sheet. Think of it as your grumpy but brilliant artist friend who will make your data look good… but only after you've pleaded with plt.tight_layout() for the 10th time."

Here are several clear, well-commented examples of different types of charts using Matplotlib:

Basic Line Plot

What it is: Connects points with lines, great for trends over time (like stock prices or temperature changes).

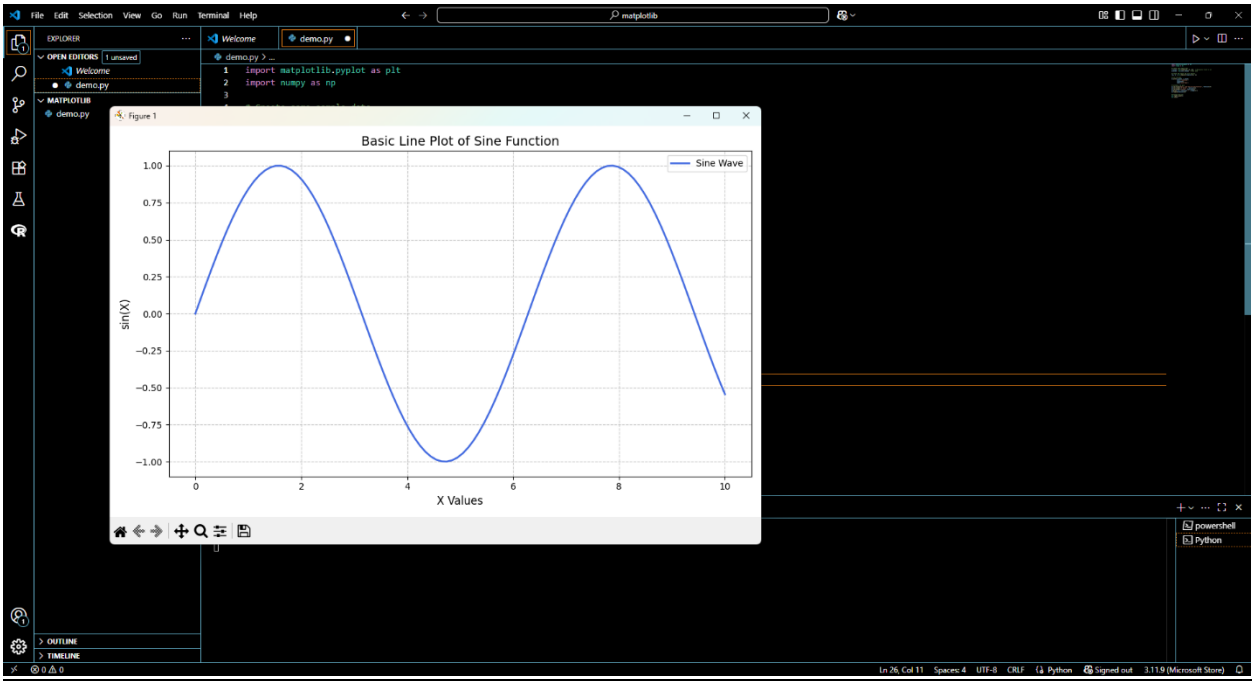
When to use: Tracking changes, trends, or continuous data.

Code Snippet:

```
import matplotlib.pyplot as plt
import numpy as np
# Create some sample data
x_values = np.linspace(0, 10, 100)  # 100 points from 0 to 10
y_values = np.sin(x_values)  # Sine wave
# Create the figure and axis objects
fig, ax = plt.subplots(figsize=(10, 6))
# Plot the data
ax.plot(x_values, y_values,
    color='royalblue',
    linewidth=2,
    label='Sine Wave')
# Customize the plot
ax.set_title('Basic Line Plot of Sine Function', fontsize=14)
ax.set_xlabel('X Values', fontsize=12)
ax.set_ylabel('sin(X)', fontsize=12)
```

```
ax.grid(True, linestyle='--', alpha=0.7)

ax.legend(fontsize=10)

# Display the plot

plt.tight_layout()

plt.show()
```

Output:



2. Bar Chart

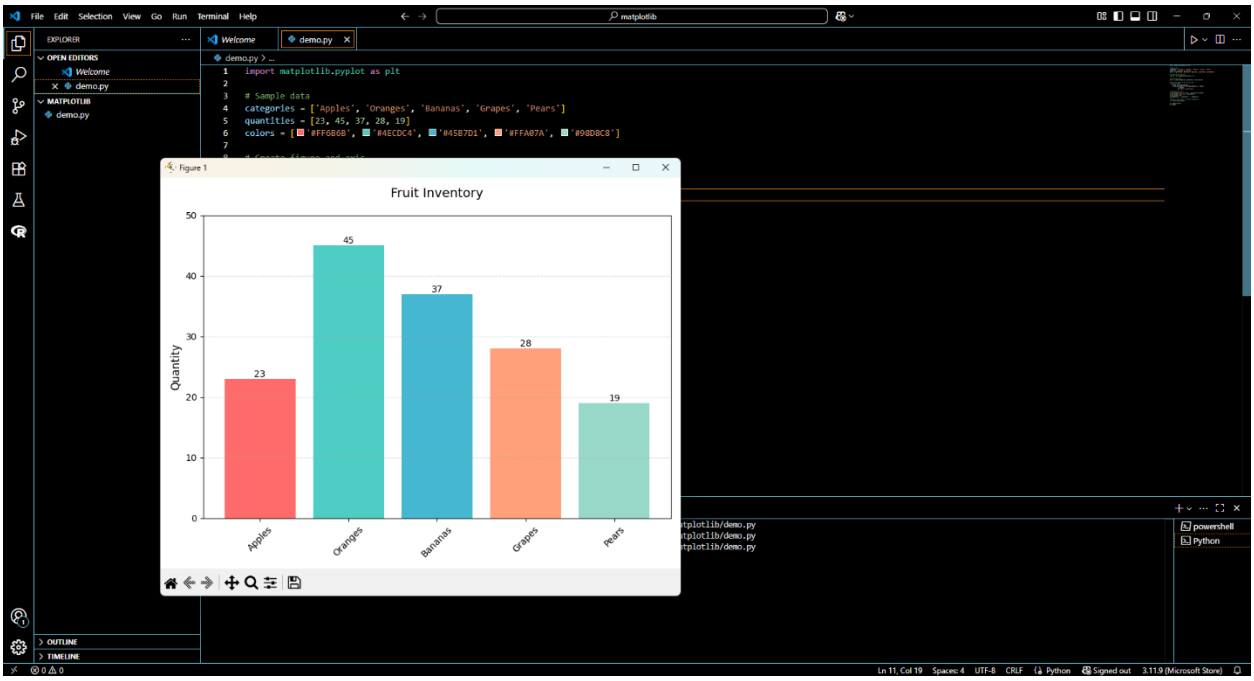
A bar chart is a graph that uses rectangular bars to represent and compare different categories of data. The height (or length) of each bar corresponds to the value it represents, making it easy to see which category is larger or smaller.

Code Snippet:

```
import matplotlib.pyplot as plt


# Sample data
categories = ['Apples', 'Oranges', 'Bananas', 'Grapes', 'Pears']
quantities = [23, 45, 37, 28, 19]
colors = ['#FF6B6B', '#4ECDC4', '#45B7D1', '#FFA07A', '#98D8C8']


# Create figure and axis
fig, ax = plt.subplots(figsize=(8, 6))


# Create bar chart
bars = ax.bar(categories, quantities, color=colors)


# Add value labels on top of each bar
for bar in bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2., height,
```

```
        f'{height}',
        ha='center', va='bottom')

# Customize the chart
ax.set_title('Fruit Inventory', fontsize=14, pad=20)
ax.set_ylabel('Quantity', fontsize=12)
ax.set_ylim(0, 50)
ax.grid(axis='y', linestyle='--', alpha=0.4)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

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

OUTPUT:



3.Pie Chart

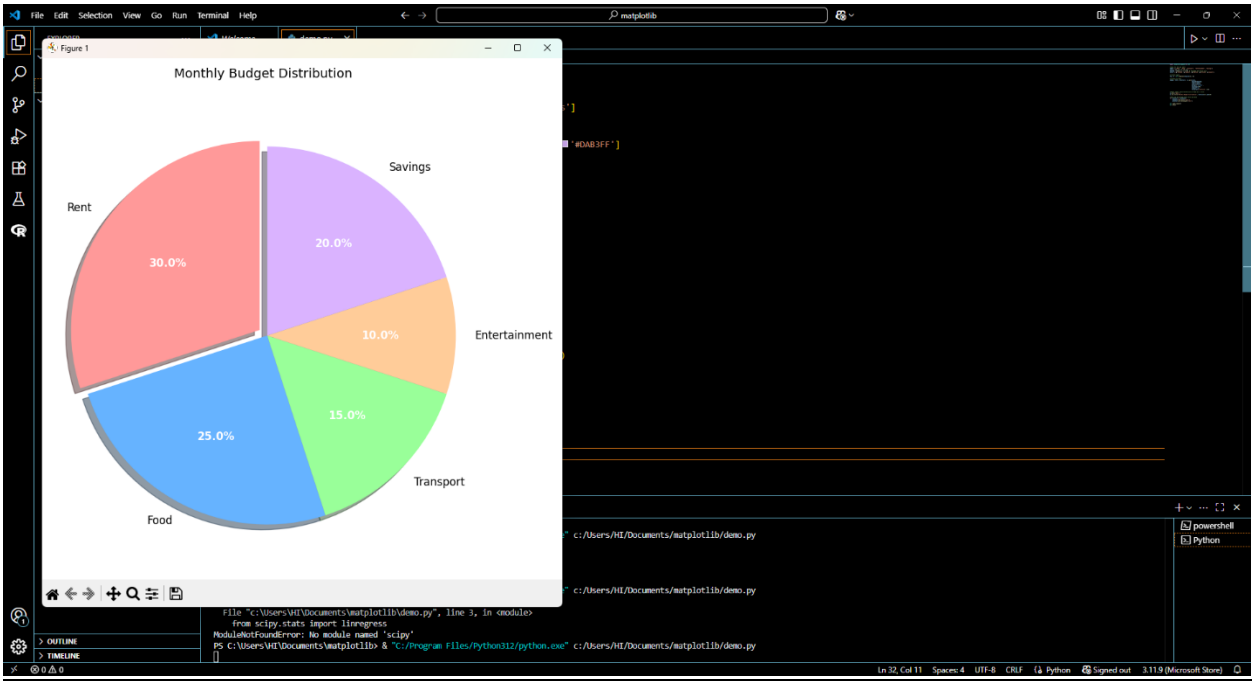
A pie chart is a circular graph that shows proportions of different categories—like slicing a pie into pieces! 🥧 Each "slice" represents a part of the whole, making it great for visualizing percentages (e.g., budget allocation, survey results, or market share).

Code Snippet:

```
import matplotlib.pyplot as plt


# Data for the pie chart
labels = ['Rent', 'Food', 'Transport', 'Entertainment', 'Savings']
sizes = [30, 25, 15, 10, 20]
explode = (0.05, 0, 0, 0, 0)  # "Explode" the first slice
colors = ['#FF9999', '#66B3FF', '#99FF99', '#FFCC99', '#DAB3FF']


# Create figure
fig, ax = plt.subplots(figsize=(8, 8))


# Create pie chart
wedges, texts, autotexts = ax.pie(sizes,
                                  explode=explode,
                                  labels=labels,
                                  colors=colors,
                                  autopct='%1.1f%%',
                                  startangle=90,
                                  shadow=True,
                                  textprops={'fontsize': 12})


# Equal aspect ratio ensures pie is drawn as a circle
ax.axis('equal')
ax.set_title('Monthly Budget Distribution', fontsize=14, pad=20)
```

```
# Make the percentage texts white and bold
for autotext in autotexts:
    autotext.set_color('white')
    autotext.set_fontweight('bold')

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

OUTPUT:



4. Histogram with Multiple Distributions

A histogram visualizes the distribution of numerical data by grouping values into "bins" and showing their frequency. When comparing multiple distributions (e.g., heights of men vs. women, exam scores across classes), you can overlay histograms to see differences clearly.

Code Snippet:

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

# Generate random data
np.random.seed(42)
data1 = np.random.normal(50, 10, 1000)
data2 = np.random.normal(70, 15, 800)
data3 = np.random.normal(30, 5, 1200)

# Create figure
fig, ax = plt.subplots(figsize=(10, 6))

# Plot histograms
ax.hist(data1, bins=30, alpha=0.6, label='Group A', color='skyblue')
ax.hist(data2, bins=30, alpha=0.6, label='Group B', color='salmon')
```

```
ax.hist(data3, bins=30, alpha=0.6, label='Group C', color='lightgreen')

# Customize
ax.set_title('Distribution Comparison', fontsize=14)
ax.set_xlabel('Values', fontsize=12)
ax.set_ylabel('Frequency', fontsize=12)
ax.grid(True, linestyle='--', alpha=0.3)
ax.legend(fontsize=10)

# Add mean lines
for data, color in zip([data1, data2, data3], ['blue', 'red', 'green']):
    ax.axvline(data.mean(), color=color, linestyle='dashed', linewidth=2)

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

OUTPUT:

---

5. Box Plot

A box plot (or "box-and-whisker plot") is a compact way to visualize the distribution of a dataset, showing its median, quartiles, outliers, and spread. It’s perfect for comparing multiple groups or detecting skewed data!

Code Snippet:

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

# Generate sample data
np.random.seed(42)
data = [np.random.normal(0, std, 100) for std in range(1, 5)]
labels = ['Group 1', 'Group 2', 'Group 3', 'Group 4']

# Create figure
fig, ax = plt.subplots(figsize=(8, 6))

# Create boxplot
box = ax.boxplot(data,
                 labels=labels,
                 patch_artist=True,
                 medianprops={'color': 'black', 'linewidth': 2})

# Customize colors
colors = ['lightblue', 'lightgreen', 'pink', 'wheat']
for patch, color in zip(box['boxes'], colors):
    patch.set_facecolor(color)

# Customize plot
ax.set_title('Box Plot Example', fontsize=14)
```

```
ax.set_ylabel('Values', fontsize=12)

ax.grid(True, linestyle='--', alpha=0.4)


plt.tight_layout()

plt.show()
```

OUTPUT:

6. Stacked Area Chart

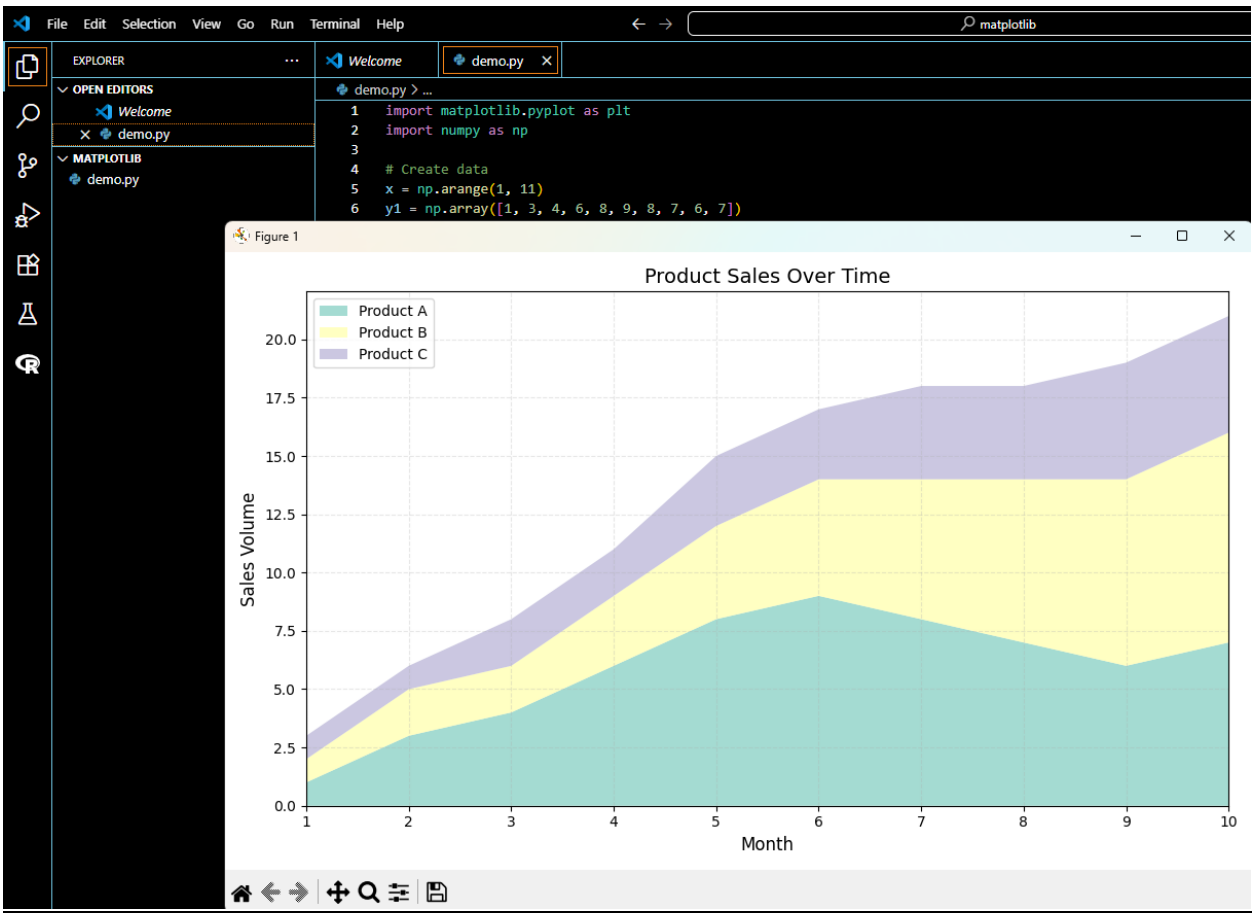
A stacked area chart visualizes how different components contribute to a whole over time (or categories). Each layer represents a category, and the height shows its cumulative value—great for trends like sales by product, population by age group, or energy sources over years.

Code Snippet:

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

# Create data
x = np.arange(1, 11)
y1 = np.array([1, 3, 4, 6, 8, 9, 8, 7, 6, 7])
y2 = np.array([1, 2, 2, 3, 4, 5, 6, 7, 8, 9])
y3 = np.array([1, 1, 2, 2, 3, 3, 4, 4, 5, 5])

# Create figure
fig, ax = plt.subplots(figsize=(10, 6))

# Create stacked area
ax.stackplot(x, y1, y2, y3,
        labels=['Product A', 'Product B', 'Product C'],
```

```
        colors=['#8dd3c7', '#ffffb3', '#bebada'],
        alpha=0.8)

# Customize
ax.set_title('Product Sales Over Time', fontsize=14)
ax.set_xlabel('Month', fontsize=12)
ax.set_ylabel('Sales Volume', fontsize=12)
ax.legend(loc='upper left', fontsize=10)
ax.grid(True, linestyle='--', alpha=0.3)
ax.set_xlim(1, 10)

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

OUTPUT:


File Edit Selection View Go Run Terminal Help
matplotlib
EXPLORER
OPEN EDITORS
Welcome
demo.py
MATPLOTLIB
demo.py
Welcome
demo.py
demo.py > ...
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # Create data
5 x = np.arange(1, 11)
6 y1 = np.array([1, 3, 4, 6, 8, 9, 8, 7, 6, 7])
Figure 1
Product Sales Over Time
Product A
Product B
Product C
Sales Volume
Month
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
1
2
3
4
5
6
7
8
9
10

What is Plotly?

Here's general definition:

Plotly is a powerful, interactive data visualization library that lets you create beautiful, dynamic charts and graphs for the web. Think of it like a digital artist that turns boring numbers into clickable, zoomable, hover-friendly visuals—perfect for dashboards, reports, or any project where you want users to explore data, not just stare at it.

Key Features:

✓ Interactive by default – Hover, zoom, pan, and click on elements.

✓ Works in browsers & notebooks – Great for Jupyter, Dash (web apps), and embedded websites.

✓ Fancy chart types – 3D plots, maps, animations, and more.

✓ Easy to use – Simple syntax (especially with plotly.express).

✓ Collaborative – Save/share charts online (like a "Google Docs for graphs").

View of Matplotlib in interesting way:

Imagine if Excel graphs and video games had a baby—that's Plotly!

It's a flashy, interactive charting tool that lets you:

Hover over data points to see details ("Ah, this dot is John's 6ft tall, 180lbs!").

Zoom in/out ("Let me check that tiny spike in 2020...").

Click legends to hide/show stuff ("Bye, 2019 data!").

Even spin 3D plots like you're in a sci-fi movie 🚀.

Where it shines:

Dashboards (like live stock trackers).

Websites/blogs (readers can play with the data).

When you want to "wow" people ("Wait, this chart MOVES?!").

Downside? Slightly heavier than Matplotlib (like comparing a gaming PC to a calculator).

Here are several clear, well-commented examples of different types of charts using Plotly:

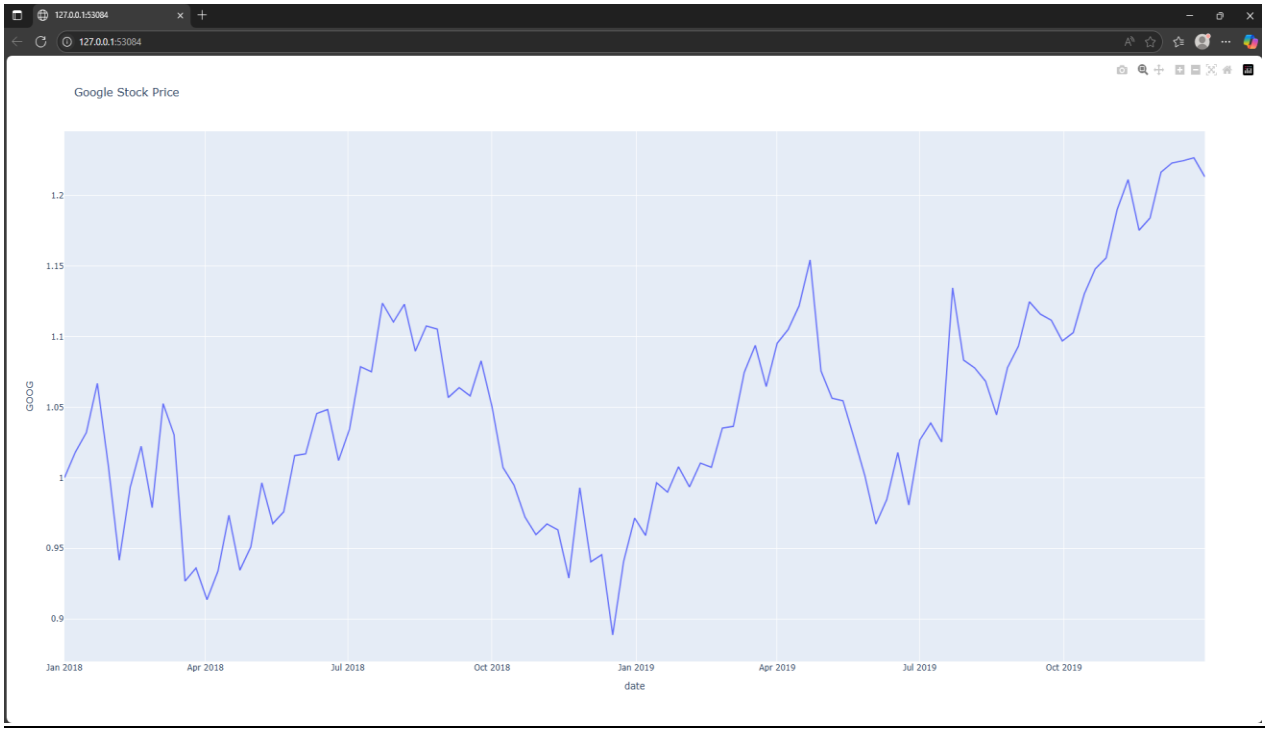
1. Basic Line Plot

A line plot in Plotly displays data points connected by straight lines, ideal for showing trends over time (e.g., stock prices, temperature changes, or website traffic). Plotly makes these plots interactive—zoom, hover for values, and toggle legends with a click!

Code Snippet:

```
import plotly.express as px

df = px.data.stocks()
fig = px.line(df, x='date', y='GOOG', title='Google Stock Price')
fig.show()
```

OUTPUT:


127.0.0.1:53084
127.0.0.1:53084
Google Stock Price
GOOG
1.2
1.15
1.1
1.05
1
0.95
0.9
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Jan 2019
Apr 2019
Jul 2019
Oct 2019
date

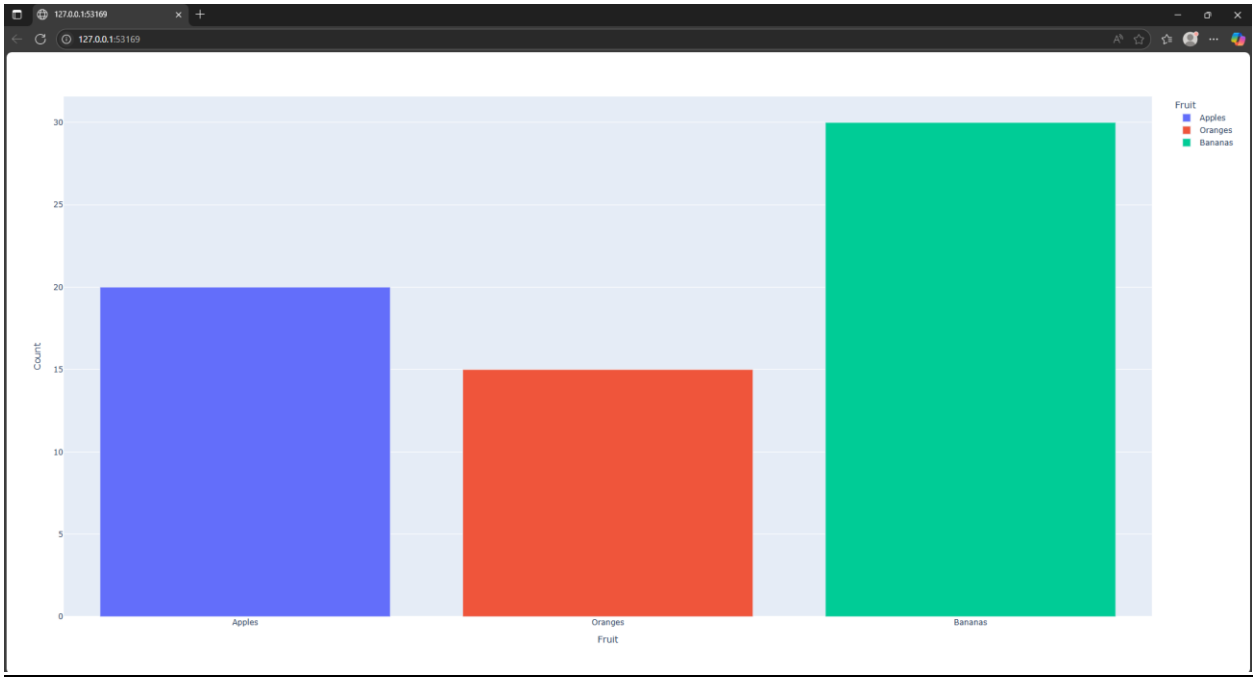
2. Bar Chart

A bar chart in Plotly displays categorical data with rectangular bars, where the height/length represents the value of each category. It's perfect for comparisons (e.g., sales by product, survey results, or population by country). Plotly adds interactivity—hover for values, zoom, and toggle bars!

Code Snippet:

```
import plotly.express as px
data = {'Fruit': ['Apples', 'Oranges', 'Bananas'], 'Count': [20, 15, 30]}
fig = px.bar(data, x='Fruit', y='Count', color='Fruit')
fig.show()
```

OUTPUT:



---

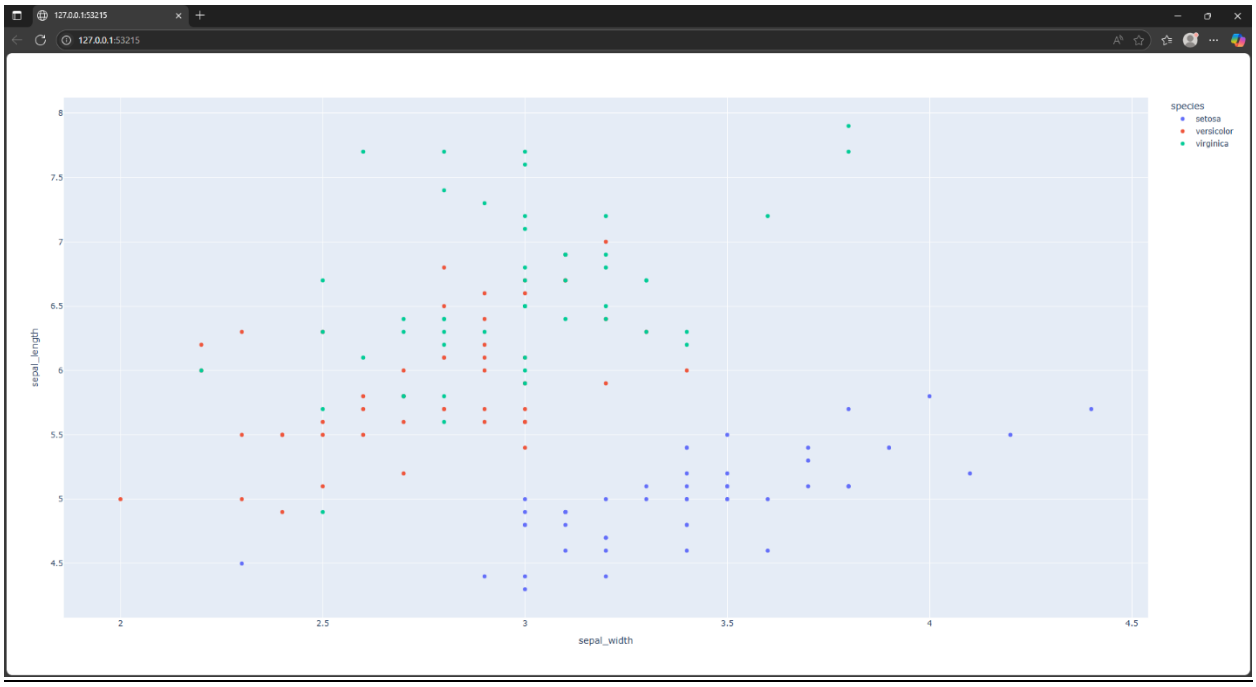
3. Scatter Plot

A scatter plot in Plotly displays individual data points as dots on a 2D (or 3D) plane, ideal for revealing relationships between two variables (e.g., correlation, clusters, or outliers). Plotly adds hover tools, zooming, and animations for deeper exploration.

Code Snippet:

```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x='sepal_width', y='sepal_length',
color='species')
fig.show()
```

OUTPUT:



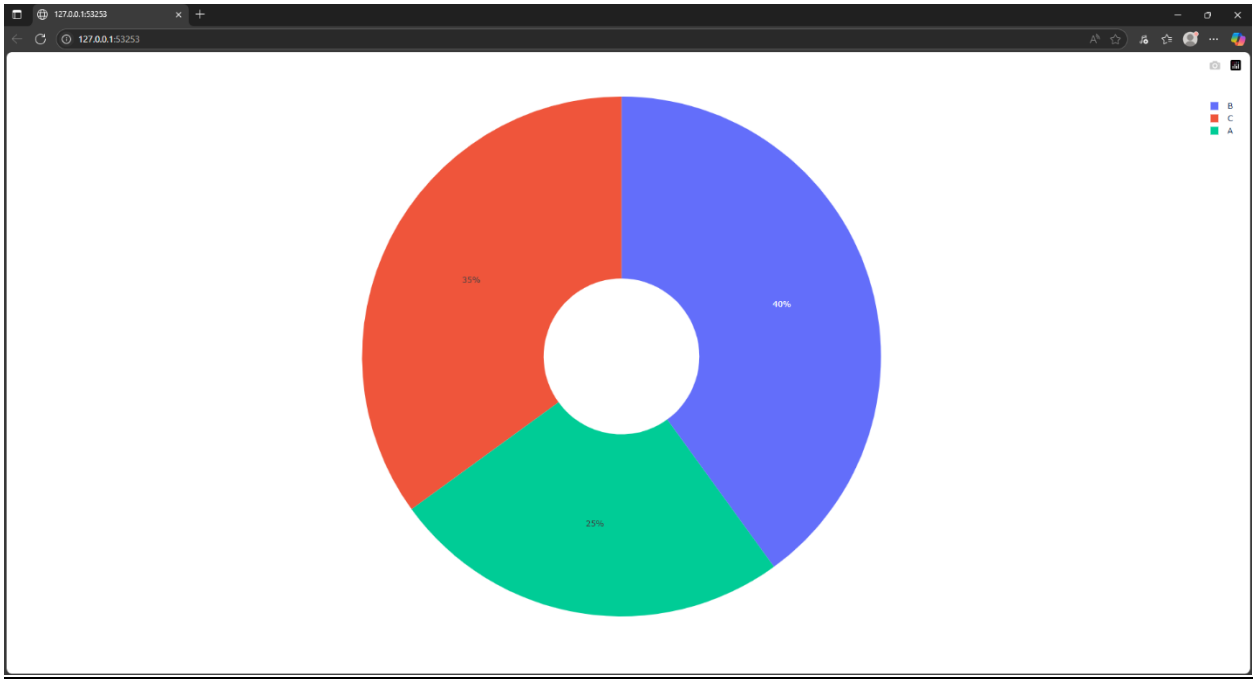
4. Pie Chart

Purpose: Show proportions of a whole (e.g., market share, survey results).

Code Snippet:

```
import plotly.express as px

data = {'Category': ['A', 'B', 'C'], 'Value': [25, 40, 35]}

fig = px.pie(data, values='Value', names='Category', hole=0.3)

fig.show()
```

OUTPUT:



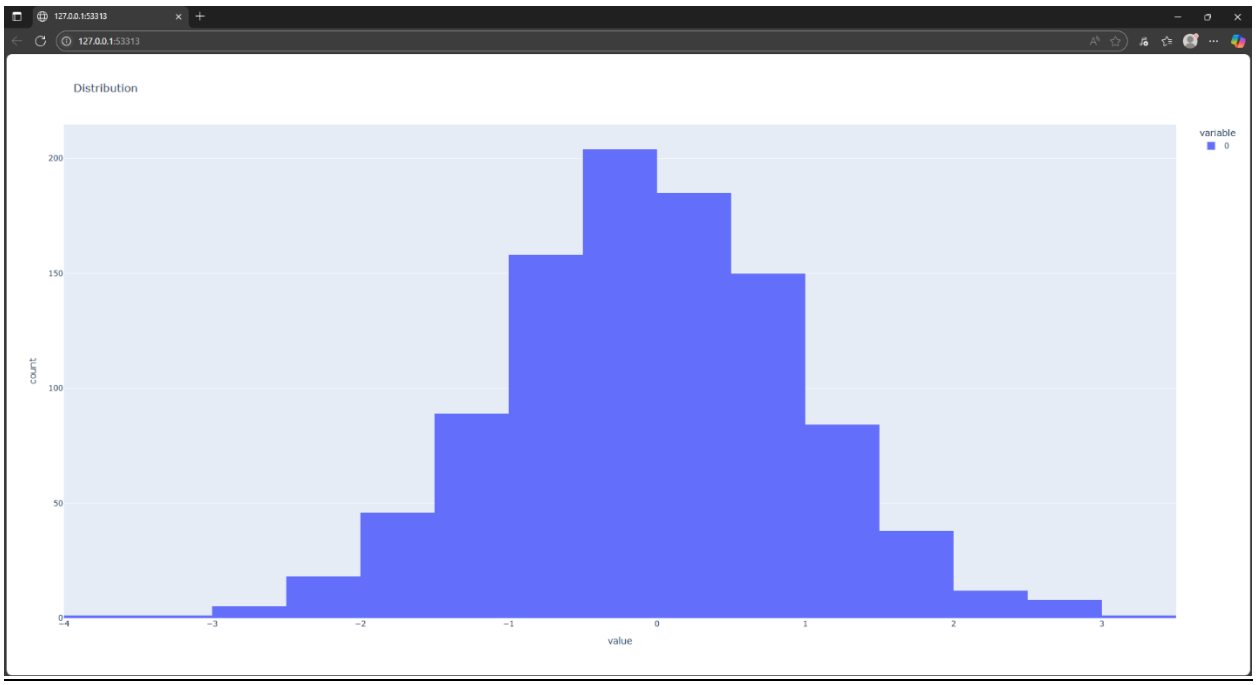
5. Histogram

Purpose: Display distribution of numerical data (e.g., ages, scores).

Code Snippet:

```
import plotly.express as px

import numpy as np

x = np.random.randn(1000)

fig = px.histogram(x, nbins=30, title='Distribution')

fig.show()
```

OUTPUT:



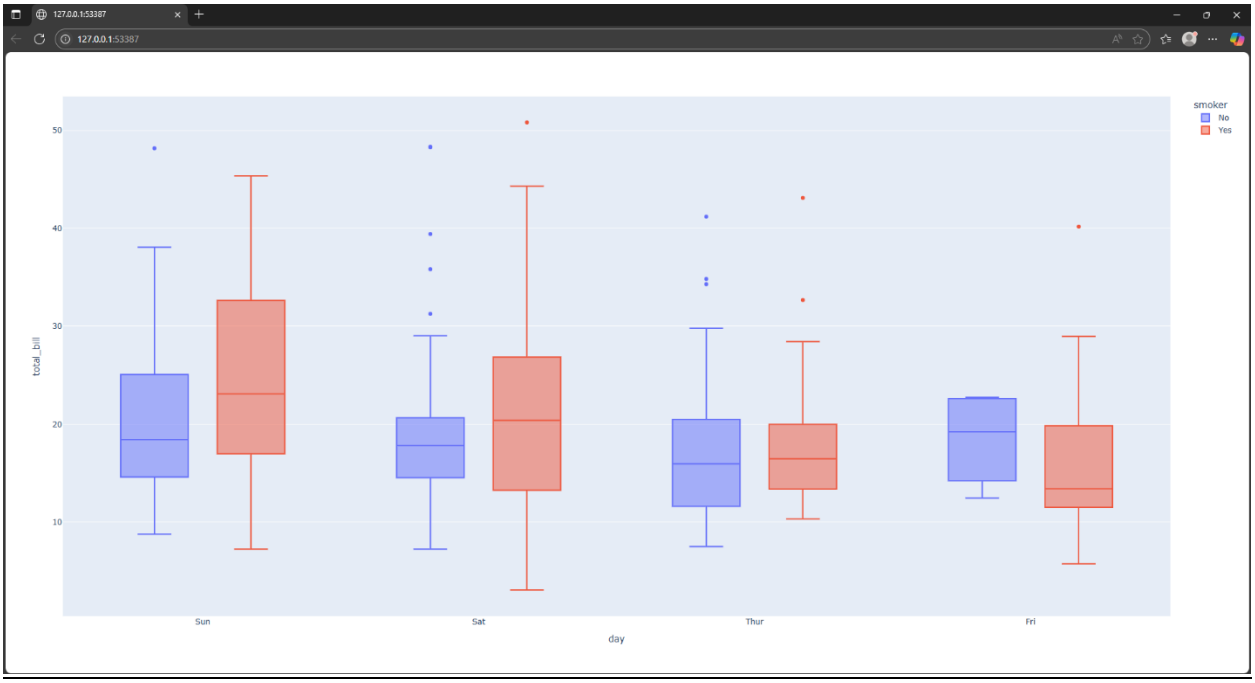
6. Box Plot

Purpose: Compare distributions (median, quartiles, outliers).

Code Snippet:

```
import plotly.express as px
df = px.data.tips()
fig = px.box(df, x='day', y='total_bill', color='smoker')
fig.show()
```

OUTPUT:



7. Heatmap

Purpose: Visualize matrix data (e.g., correlations, activity grids).

Code Snippet:

```
import plotly.express as px

import numpy as np

z = np.random.rand(5,5)

fig = px.imshow(z, text_auto=True,
color_continuous_scale='Viridis')

fig.show()
```

OUTPUT:

---

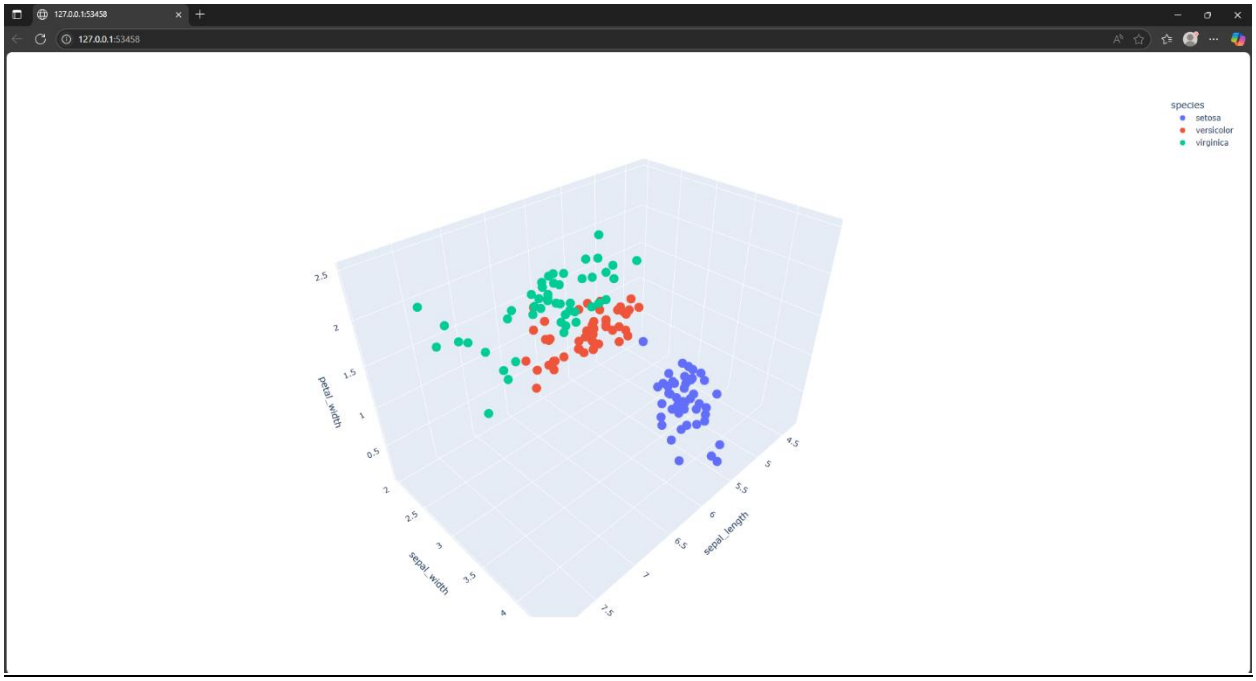
8. 3D Scatter

Purpose: Explore 3D relationships.

Code Snippet

```
import plotly.express as px

df = px.data.iris()

fig = px.scatter_3d(df, x='sepal_length', y='sepal_width',
z='petal_width', color='species')

fig.show()
```

OUTPUT:


127.0.0.1:53458
species
setosa
versicolor
virginica
petal_width
sepal_width
sepal_length

Comparison of Matplotlib and Plot

Feature	Matplotlib	Plotly
Type of Plots	Static (images, PDFs)	Interactive (HTML, dashboards)
Learning Curve	Moderate (more manual customization)	Easier for interactive plots
Popularity	Very popular in academic research	Popular in business & dashboards
Customization	Highly customizable	Pre-built templates and themes
Output Format	PNG, PDF, SVG	HTML, Jupyter, Dash apps
Community Support	Very large	Growing fast
Integration	Works well with NumPy, Pandas	Works well with Dash, web apps

Official Matplotlib resource:

https://matplotlib.org/stable/users/explain/quick_start.html#quick-start

Official Plotly resource:

https://plotly.com/python/distplot/