**Beginners Guide To Python Visualisation Libraries:**

**Introduction :** In Python, visualization libraries are tools that let you turn data into charts, graphs, and other visual representations making it easier to analyse and communicate insights.

Python offers many visualization libraries to help turn raw data into meaningful visual insights.  
Let’s walk through two widely used ones — **plotly** and **Pandas** — to understand how they help create different types of plots and charts for data analysis.

Plotly:

Library overview :

In Python, **Plotly** is an interactive visualization library that allows you to create dynamic, publication-quality charts and dashboards. Unlike static libraries, Plotly’s charts let you **zoom, hover, and filter** data points directly in the browser, making it easier to explore and present insights interactively.

Plotly supports a wide range of visualizations — from basic **line, bar, and scatter plots** to advanced **heatmaps, choropleth maps, 3D plots, and animations**. It works seamlessly with popular data analysis tools like **Pandas** and **NumPy**, making it easy to turn raw datasets into rich visual experiences.

Whether you’re creating quick **exploratory visuals** or polished **interactive dashboards**, Plotly offers both a **Python API** (plotly.graph\_objects and plotly.express) and integration with **Dash**, a framework for building complete web applications for data visualization.

**✨ Key Features:**

* **Interactive charts** with zoom, pan, hover tooltips, and legends.
* Supports **2D, 3D, and geographic visualizations**.
* **High customization** with colors, styles, and annotations.
* Works in **Jupyter notebooks, web apps, and standalone HTML files**.
* Can export charts as **static images or interactive HTML**.

**🔹 Common Uses:**

* Data analysis and exploration
* Business dashboards
* Scientific research visualization
* Real-time data monitoring

Graph types :

**Line Plot**

* **Purpose:** Used to visualize trends over time or continuous data.
* **Key Feature in Plotly:** Interactive by default, with hover tooltips, zoom, and pan options.
* **Function:** px.line()

  
 code :

import plotly.express as px

import pandas as pd

data = {

'Year': [2018, 2019, 2020, 2021, 2022],

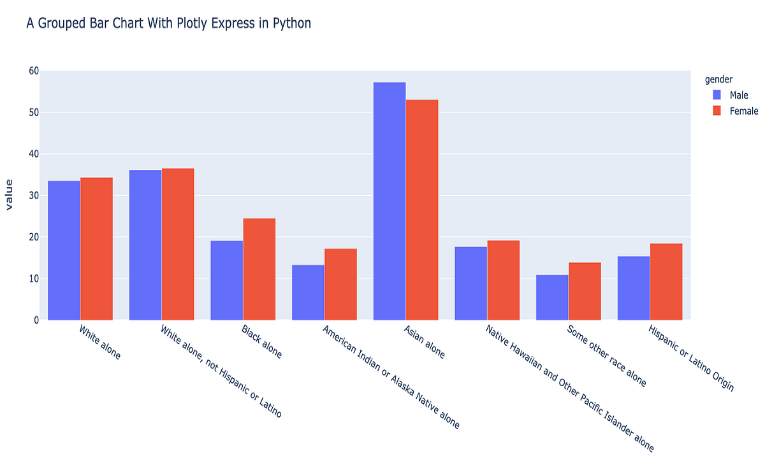
'Sales': [100, 150, 130, 170, 2000,

df = pd.DataFrame(data)

fig = px.line(df, x='Year', y='Sales', title='Sales Over Years', markers=True)

BAR CHART :

A **bar chart** is a type of graph that represents **categorical data** with rectangular bars, where the length or height of each bar corresponds to the value of that category.



Code :

import plotly.express as px

import pandas as pd

data = {

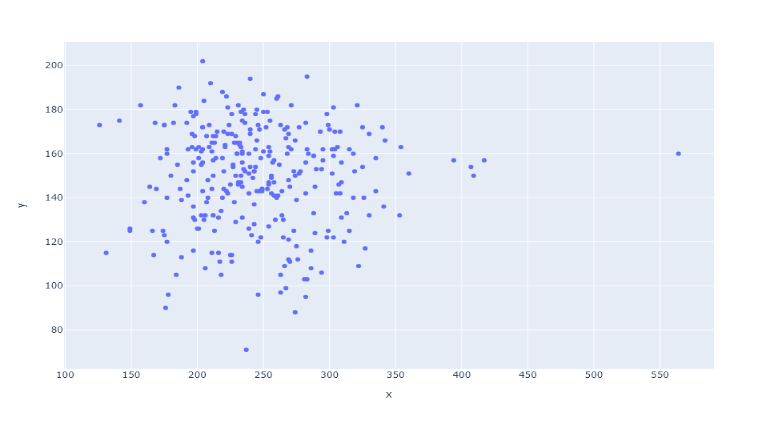
'Fruits': ['Apples', 'Oranges', 'Bananas', 'Grapes'],

'Quantity': [10, 15, 7, 12]

}

Scatter plot :

A **scatter plot** displays data points on a two-dimensional plane, showing the relationship between two variables. Each point represents one observation, with its position determined by the values of the variables on the X and Y axes.



Code :

import plotly.express as px

import pandas as pd

data = {

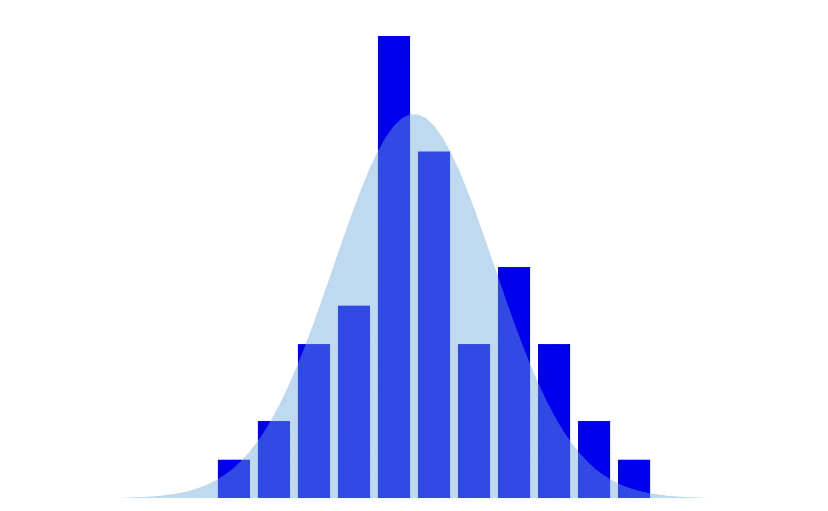
'Height': [150, 160, 170, 180, 190],

'Weight': [50, 60, 65, 80, 90],

'Name': ['A', 'B', 'C', 'D', 'E') }

Histogram :

A **histogram** is a type of chart that shows the distribution of a dataset by grouping data into **intervals (bins)** and counting how many values fall into each bin.



Code :

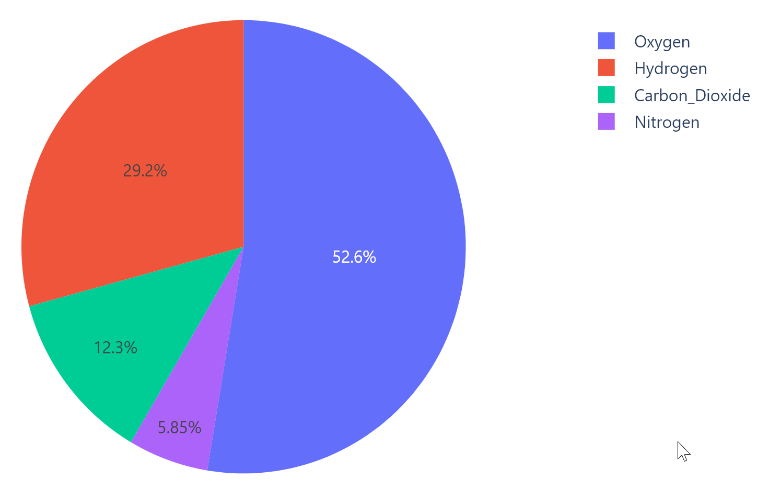
data = {

'Scores': [55, 60, 65, 70, 70, 75, 80, 85, 85, 90, 92, 95, 100, 100)

}

Pie chart :

A **pie chart** displays proportions of categories as slices of a circle, where each slice’s angle corresponds to its percentage of the whole.



PANDAS :

Pandas is a powerful Python library designed for data manipulation, analysis, and visualization. It provides two main data structures -Series (1D) and DataFrame (2D) - which make it easy to work with structured data like tables.

While Pandas is primarily used for data cleaning, transformation, and analysis, it also has built-in visualization capabilities that allow quick creation ofplots directly from Series or DataFrames, making it ideal for exploratory data analysis.

**Key Features of Pandas**

1. **Powerful Data Structures** – Offers **Series** (1D) and **DataFrame** (2D) for handling structured data.
2. **Easy Data Manipulation** – Supports filtering, grouping, merging, joining, and reshaping datasets.
3. **Handling Missing Data** – Provides tools to detect, remove, or fill in missing values.
4. **Data Import & Export** – Reads and writes data from formats like CSV, Excel, JSON, SQL, etc.
5. **Built-in Visualization** – Creates quick plots directly from Series/DataFrames using .plot().
6. **Integration with Other Libraries** – Works seamlessly with NumPy, Matplotlib, and other Python data tools.
7. **High Performance** – Optimized for large datasets through efficient indexing and operations.

**GRAPH TYPES** :

Top of Form

**Line Plot :**

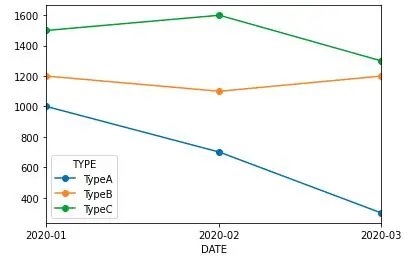
**Description:** Plots data points connected by lines, showing trends over continuous data.  
**Use Case:** Sales growth over months.

**Sample Code:**

import pandas as pd

data = pd.Series([10, 20, 15, 25, 30])

data.plot(title="Line Plot", marker='o')



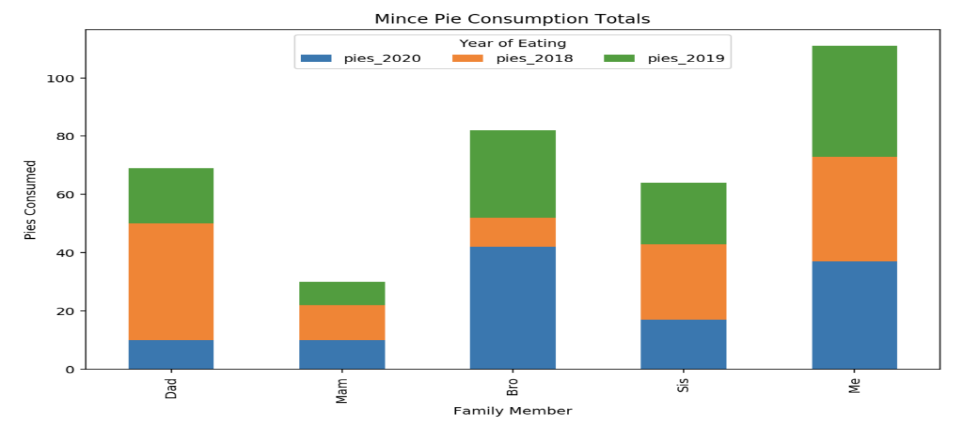
**Bar Chart :**

**Description:** Displays data as rectangular bars for categorical comparison.  
**Use Case:** Comparing product sales.

**Sample Code:**

df = pd.Series([5, 8, 12], index=['A', 'B', 'C'])

df.plot(kind='bar', title="Bar Chart", color='skyblue')

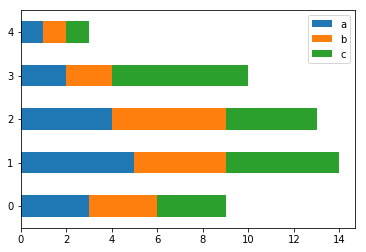


**Horizontal Bar Chart :**

**Description:** Same as a bar chart but horizontal.  
**Use Case:** Comparing values when category labels are long.

**Sample Code:**

df.plot(kind='barh', title="Horizontal Bar Chart", color='lightgreen')



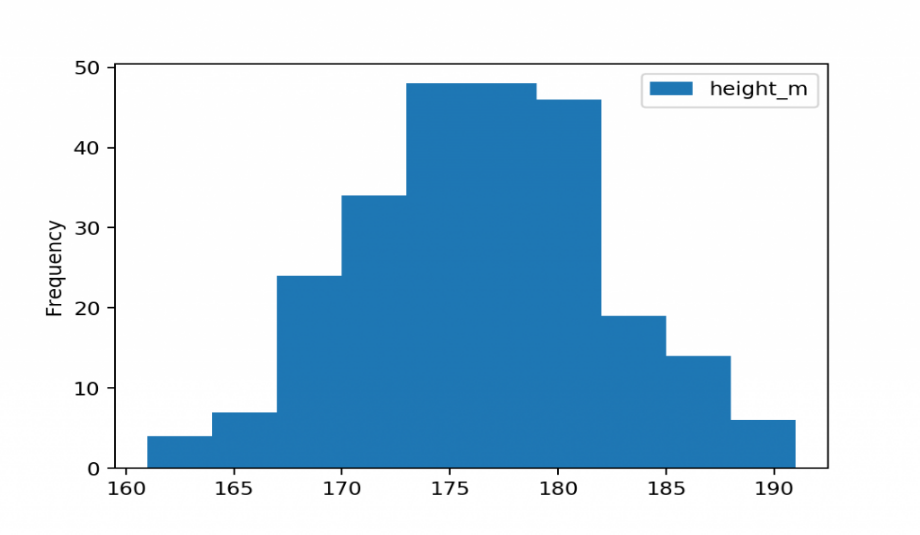
**Histogram**

**Description:** Groups numeric data into bins to show frequency distribution.  
**Use Case:** Analyzing exam score distributions

**Sample Code:**

data = pd.Series([3, 5, 5, 6, 7, 8, 8, 9, 10])

data.plot(kind='hist', bins=5, title="Histogram", color='orange', edgecolor='black')



**Area Plot :**

**Description:** Like a line plot but the area under the line is filled.  
**Use Case:** Showing cumulative trends.

**Sample Code:**

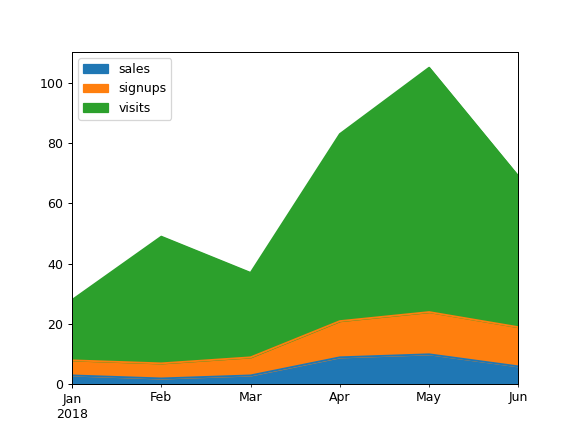
df = pd.DataFrame({

'A': [1, 3, 4],

'B': [2, 4, 6]

})

df.plot(kind='area', alpha=0.5, title="Area Plot")



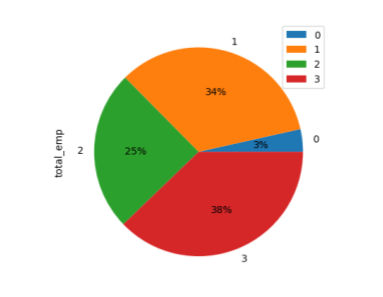
**Pie Chart :**

**Description:** Shows proportions of a whole as slices of a circle.  
**Use Case:** Visualizing budget distribution.

**Sample Code:**

df = pd.Series([30, 20, 50], index=['A', 'B', 'C']

df.plot(kind='pie', autopct='%1.1f%%', title="Pie Chart")



|  |  |  |
| --- | --- | --- |
| **Comparison: Plotly vs Pandas Plotting** |  |  |
| | **Feature** | **Plotly** | **Pandas Plotting** | | --- | --- | --- | | **Interactivity** | Fully interactive (zoom, hover tooltips, pan, legend click) | Static plots (no interactivity by default) | | **Ease of Use** | High-level API (plotly.express) makes it easy for quick plots; more complex customization via graph\_objects | Very simple for quick plotting from DataFrames (df.plot()), minimal setup | | **Chart Variety** | Wide range: 2D, 3D, maps, animations, heatmaps, advanced charts | Basic charts: line, bar, histogram, box, area, scatter, pie, hexbin | | **Customization** | Highly customizable (colors, annotations, styles, layouts) | Limited customization compared to Plotly | | **Integration** | Works well with Pandas, NumPy, Dash (for dashboards) | Built directly into Pandas, no extra library needed | | **Output** | Web-friendly HTML charts, can export as images | Static PNG/SVG via Matplotlib backend | | **Performance** | May lag with very large datasets | Faster for small to medium datasets | | **Installation** | Requires separate installation (pip install plotly) | Comes with Pandas (depends on Matplotlib) | | | | |  |  |
| Resources :  : <https://plotly.com/python/>  : <https://www.educative.io/answers/plotly-graph-objects-and-its-methods>  : <https://pandas.pydata.org/>  : <https://en.wikipedia.org/wiki/Pandas_(software)>  Conclusion :  **Plotly vs Pandas Plotting**  While both Plotly and Pandas plotting are valuable tools for data visualization in Python, they serve different purposes:   * **Pandas plotting** is ideal for **quick, simple, static** visualizations directly from DataFrames. It’s fast, easy to use, and great for exploratory analysis when interactivity isn’t required. * **Plotly** excels in creating **interactive, customizable, and presentation-ready** charts. It’s perfect when you need engaging visuals for reports, dashboards, or sharing insights online. | | | |  |  |
|  | | | |  |  |
|  | | | |  |  |
|  | | | |  |  |
|  | | | |  |  |
|  | | | |  |  |
|  | | | |  |  |
|  | | | |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |