Python Libraries

1. NumPy

Purpose:

- NumPy (Numerical Python) is a powerful library for numerical computing in Python. It provides support for arrays (multidimensional arrays or matrices) and a collection of mathematical functions to operate on these arrays efficiently.

Key Features:

- **N-dimensional array object:** Core to NumPy is the `ndarray`, a fast and space-efficient multidimensional array.
- **Mathematical functions:** Comprehensive collection of functions to perform operations like linear algebra, Fourier transforms, and random number generation.
- **Broadcasting:** Allows you to perform operations on arrays of different shapes.
- **Integration with C/C++ and Fortran code:** Facilitates performance-critical tasks.

Example:

python

```
import numpy as np

# Creating a 2D array
array = np.array([[1, 2, 3], [4, 5, 6]])

# Performing element-wise operations
array_squared = array ** 2
print(array_squared)
```

2. Pandas

Purpose:

- Pandas is a powerful and flexible library for data manipulation and analysis. It introduces data structures like Series (1D) and DataFrame (2D) that are ideal for handling and analyzing structured data.

Key Features:

- **DataFrame:** A 2D table of data with labeled axes (rows and columns).
- **Handling missing data:** Pandas offers robust tools for detecting, filtering, and filling in missing data.
- **Data alignment and reshaping:** Automatic alignment of data in DataFrame and easy reshaping via pivoting or stacking/unstacking.
- **GroupBy:** Split-apply-combine strategy for aggregating data.
- **Time series:** Tools for time series analysis and operations.

Example:

python

```
import pandas as pd

# Creating a DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [25, 30,
35]}
df = pd.DataFrame(data)

# Accessing data
print(df['Name'])

# Filtering data
adults = df[df['Age'] > 30]
print(adults)
```

3. Matplotlib

Purpose:

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is highly customizable and can produce plots, histograms, power spectra, bar charts, error charts, and more.

Key Features:

- **Plotting 2D graphs:** Line plots, scatter plots, bar charts, histograms, etc.
- **Customization:** Full control over every element of a plot, including line styles, font properties, axes properties, etc.
- **Integration with other libraries:** Works well with NumPy and Pandas, making it easy to visualize data stored in these structures.
- **Subplots and complex layouts:** Create complex visualizations with multiple subplots.

Example:

python

```
import matplotlib.pyplot as plt

# Creating simple line plot
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

plt.plot(x, y)
plt.title('Simple Line Plot')
plt.xlabel('X Axis')
plt.ylabel('Y Axis')
plt.show()
```

Summary

- **NumPy** is ideal for numerical computations and array manipulation.
- **Pandas** is the go-to library for handling and analyzing structured data, particularly in tabular form.
- **Matplotlib** excels at creating a wide variety of static and interactive visualizations, making it a powerful tool for data visualization.