1. How does broadcasting work in NumPy?

Ans. Broadcasting allows NumPy to perform operations on arrays of different shapes without explicitly replicating data. Smaller arrays are "broadcast" over larger ones so that element-wise operations can be performed efficiently.

Example:

python

import numpy as np

a = np.array([1, 2, 3])

b = np.array([[1], [2], [3]])

result = a + b

2. What is a Pandas DataFrame?

A **DataFrame** is a 2-dimensional labeled data structure in Pandas, similar to a spreadsheet or SQL table. It can hold different types of data (int, float, string) and is ideal for data manipulation and analysis.

---

3. Explain the use of the `groupby()` method in Pandas.

`groupby()` is used to group data based on some criteria and perform aggregate operations like `sum()`, `mean()`, etc.

Example:

python

CopyEdit

```
df.groupby('Category')['Sales'].sum()
```

---

4. Why is Seaborn preferred for statistical visualizations?

Seaborn is built on top of Matplotlib and integrates closely with Pandas. It offers:

- Beautiful default styles
- Built-in statistical plots like boxplots, violin plots, and KDEs

- Automatic estimation and plotting of linear regressions

---

5. What are the differences between NumPy arrays and Python lists?

Feature	NumPy Array	Python List
Performance	Faster	Slower
Data type	Homogeneous	Heterogeneous
Functionality	Supports vectorized ops	No vectorization
Memory usage	Lower	Higher

---

6. What is a heatmap, and when should it be used?

A **heatmap** is a data visualization where individual values are represented as colors. It's useful for visualizing correlations, confusion matrices, or any 2D data relationships.

---

7. What does the term “vectorized operation” mean in NumPy?

Vectorized operations allow you to perform element-wise operations on entire arrays without using loops, which makes computations faster and more efficient.

Example:

python

CopyEdit

```
a = np.array([1, 2, 3])

b = a * 2  # Vectorized operation
```

---

8. How does Matplotlib differ from Plotly?

Feature	Matplotlib	Plotly
Interactivity	Limited	High
Output	Static	Interactive (HTML)
Ease of use	More complex	User-friendly
3D plots	Limited support	Strong support

---

9. What is the significance of hierarchical indexing in Pandas?

Hierarchical indexing (MultiIndex) allows for indexing and subsetting of data with multiple keys. It's especially useful for working with panel data or grouped time series.

---

10. What is the role of Seaborn's `pairplot()` function?

`pairplot()` automatically plots pairwise relationships between numeric columns in a dataset, showing scatter plots, histograms, and optional class distinctions.

---

11. What is the purpose of the `describe()` function in Pandas?

`describe()` generates summary statistics (count, mean, std, min, max, quartiles) for numerical (or categorical) columns.

---

12. Why is handling missing data important in Pandas?

Missing data can skew analysis, models, or visualizations. Pandas provides functions like `isna()`, `fillna()`, and `dropna()` to handle them effectively.

---

13. What are the benefits of using Plotly for data visualization?

- Interactive visualizations
- Export to HTML
- Integration with Dash for building dashboards
- Supports 3D and geographic plots

---

14. How does NumPy handle multidimensional arrays?

NumPy uses `ndarray` objects that support n-dimensional arrays. Operations are optimized and extend naturally from 1D to multi-D (like matrices and tensors).

---

15. What is the role of Bokeh in data visualization?

Bokeh is a Python library for creating interactive plots and dashboards for web browsers. It supports real-time streaming and high-performance rendering of large datasets.

---

16. Explain the difference between `apply()` and `map()` in Pandas.

- `map()` is used on Series and is typically for element-wise transformations.
- `apply()` can be used on both Series and DataFrames for applying a function along an axis (rows/columns).

---

17. What are some advanced features of NumPy?

- Broadcasting
- Masking and boolean indexing

- Memory mapping
- FFTs and linear algebra
- Custom data types
- Structured arrays

---

18. How does Pandas simplify time series analysis?

- Date parsing and conversion with `to_datetime()`
- Resampling with `resample()`
- Shifting and lagging with `shift()`
- Rolling statistics with `rolling()`

---

19. What is the role of a pivot table in Pandas?

A pivot table in Pandas summarizes data with aggregate functions, allowing you to reshape datasets for better analysis.

Example:

python

CopyEdit

```
df.pivot_table(values='Sales', index='Region', columns='Month', aggfunc='sum')
```

---

20. Why is NumPy's array slicing faster than Python's list slicing?

NumPy arrays are stored in contiguous memory blocks and operate at the C level, making slicing and data access much faster compared to Python lists, which are objects referencing pointers.

---

21. What are some common use cases for Seaborn?

- Correlation analysis with heatmaps
- Categorical data visualization with barplots, violin plots
- Distribution analysis with histograms and KDEs
- Pairwise relationships using `pairplot()` or `jointplot()`**1. Create a 2D NumPy array and calculate the sum of each row**

`python`

`CopyEdit`

- `import numpy as np`
-
- `arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])`
- `row_sum = arr.sum(axis=1)`
- `print(row_sum)`

✅ 2. Pandas: Find the mean of a specific column in a DataFrame

`python`

`CopyEdit`

- `import pandas as pd`
-
- `df = pd.DataFrame({'A': [10, 20, 30], 'B': [5, 15, 25]})`
- `mean_value = df['B'].mean()`
- `print(mean_value)`

✅ 3. Create a scatter plot using Matplotlib

python

CopyEdit

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

plt.scatter(x, y)
plt.title('Scatter Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```

✅ 4. Calculate the correlation matrix using Seaborn and visualize with a heatmap

python

CopyEdit

```
import seaborn as sns
import pandas as pd

df = pd.DataFrame({
    'A': [1, 2, 3, 4],
    'B': [10, 20, 30, 40],
    'C': [5, 10, 15, 20]
})

corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

✅ 5. Generate a bar plot using Plotly

python

CopyEdit

```
import plotly.express as px
import pandas as pd

data = pd.DataFrame({'Fruits': ['Apple', 'Banana', 'Cherry'], 'Quantity': [10, 20, 15]})
fig = px.bar(data, x='Fruits', y='Quantity', title='Fruit Quantities')
fig.show()
```

✅ 6. Create a DataFrame and add a new column based on an existing column

python

CopyEdit

```
import pandas as pd

df = pd.DataFrame({'Price': [100, 200, 300]})
df['Discounted'] = df['Price'] * 0.9
print(df)
```

✅ 7. Element-wise multiplication of two NumPy arrays

python

CopyEdit

```
import numpy as np

a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
result = a * b
```

```
print(result)
```

✅ 8. Line plot with multiple lines using Matplotlib

python

CopyEdit

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y1 = [1, 4, 9, 16]
y2 = [2, 5, 10, 17]

plt.plot(x, y1, label='Line 1')
plt.plot(x, y2, label='Line 2')
plt.legend()
plt.title('Multiple Line Plot')
plt.show()
```

✅ 9. Generate a Pandas DataFrame and filter rows where column value > threshold

python

CopyEdit

```
import pandas as pd

df = pd.DataFrame({'Age': [15, 25, 35, 45]})
filtered_df = df[df['Age'] > 30]
print(filtered_df)
```

✅ 10. Histogram using Seaborn to visualize a distribution

python

CopyEdit

```
import seaborn as sns
import numpy as np

data = np.random.normal(loc=0, scale=1, size=100)
sns.histplot(data, kde=True)
```

✅ 11. Perform matrix multiplication using NumPy

python

CopyEdit

```
import numpy as np

a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6], [7, 8]])
result = np.dot(a, b)
print(result)
```

✅ 12. Use Pandas to load a CSV and display first 5 rows

python

CopyEdit

```
import pandas as pd

df = pd.read_csv('your_file.csv')  # replace with your file path
```

```
print(df.head())
```

✅ 13. Create a 3D scatter plot using Plotly

python

CopyEdit

```
import plotly.express as px
import pandas as pd

df = pd.DataFrame({
    'x': [1, 2, 3, 4],
    'y': [10, 20, 30, 40],
    'z': [100, 200, 300, 400]
})

fig = px.scatter_3d(df, x='x', y='y', z='z')
fig.show()
```

✅ 14. Java + Data Structures (Quick Overview)

Commonly used **Java Data Structures**:

- **Array / ArrayList**
- **LinkedList**
- **Stack**
- **Queue / Deque**
- **HashMap / TreeMap**
- **HashSet / TreeSet**

Example: Using a HashMap in Java

java

CopyEdit

```
import java.util.HashMap;

public class Main {
    public static void main(String[] args) {
        HashMap<String, Integer> map = new HashMap<>();
        map.put("Apple", 2);
        map.put("Banana", 5);
        System.out.println(map.get("Banana"));
    }

}
```