THeroy que and answer

1.What is NumPy, and why is it widely used in Python? ANs.NumPy (Numerical Python) is a library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on them. It is widely used because of its efficiency, speed, and powerful features like broadcasting, vectorized operations, and integration with other libraries.

2.How does broadcasting work in NumPy? ANS.Broadcasting allows NumPy to perform arithmetic operations on arrays of different shapes by automatically expanding the smaller array to match the larger array's shape. This eliminates the need for explicit looping and makes operations more efficient.

3.What is a Pandas DataFrame? ANs.A Pandas DataFrame is a two-dimensional, tabular data structure with labeled rows and columns, similar to an Excel spreadsheet or SQL table. It is widely used for data analysis and manipulation.

4.Explain the use of the groupby() method in Pandas. Ans.The groupby() method is used to group data based on one or more columns and apply aggregation functions such as sum(), mean(), or count() to summarize the grouped data.

5.Why is Seaborn preferred for statistical visualizations? ANs.Seaborn is preferred because it provides high-level functions for creating attractive and informative statistical graphics. It integrates well with Pandas and simplifies complex visualizations like heatmaps, violin plots, and pair plots.

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

ans.NumPy arrays are more memory-efficient and faster than Python lists. NumPy supports vectorized operations, while Python lists require explicit loops. NumPy arrays support multi-dimensional data structures, while Python lists are primarily one-dimensional.

7.What is a heatmap, and when should it be used? A heatmap is a data visualization technique that uses colors to represent values in a matrix or dataset. It is commonly used to show correlations, relationships, or density variations in data.

8.What does the term "vectorized operation" mean in NumPy? A vectorized operation refers to performing an operation on an entire array without using explicit loops. This makes computations faster and more efficient.

9.How does Matplotlib differ from Plotly?

ans. Matplotlib is a static plotting library primarily used for simple visualizations. Plotly is an interactive visualization library that allows zooming, panning, and tooltips for better data exploration.

10.What is the significance of hierarchical indexing in Pandas? Hierarchical indexing allows multiple levels of row or column labels, enabling complex data representation and multi-level data aggregation.

11.What is the role of Seaborn’s pairplot() function? ans.pairplot() creates pairwise scatter plots for all numerical variables in a dataset, making it useful for visualizing relationships and patterns.

12.What is the purpose of the describe() function in Pandas? ans.The describe() function provides summary statistics of numerical columns, including mean, median, standard deviation, and quartiles.

13.Why is handling missing data important in Pandas? ans.Handling missing data prevents errors in analysis, ensures data consistency, and improves the accuracy of statistical models.

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

ans.Interactive charts Built-in support for complex visualizations (3D plots, choropleths) Customization and dashboard integration

15.How does NumPy handle multidimensional arrays? NumPy represents multidimensional arrays as ndarray objects, where each dimension is an axis. It provides operations like reshaping, slicing, and broadcasting for efficient computations.

16.What is the role of Bokeh in data visualization? Bokeh is a Python library for creating interactive, web-based visualizations with high performance and customization.

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

apply() applies a function to an entire DataFrame or Series. map() applies a function element-wise but only works on Series.

18.What are some advanced features of NumPy?

Broadcasting Memory mapping Universal functions (ufuncs) Fast Fourier Transform (FFT) Random number generation

19.How does Pandas simplify time series analysis? Pandas provides built-in support for time-based indexing, resampling, shifting, and rolling window calculations, making time series analysis easier.

20.What is the role of a pivot table in Pandas? A pivot table reshapes data by summarizing values based on specified columns, similar to an Excel pivot table.

21.Why is NumPy’s array slicing faster than Python’s list slicing? NumPy slices return views (not copies), meaning no new memory is allocated. Python lists create copies, which increases overhead.

22.What are some common use cases for Seaborn?

Correlation heatmaps Distribution plots Pairwise relationship visualizations Regression plots Categorical data visualization

Practical que and answer

1.Create a 2D NumPy array and calculate the sum of each row

import numpy as np

```
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) row_sums = np.sum(arr, axis=1) print(row_sums)
```

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

```
import pandas as pd

df = pd.DataFrame({'A': [10, 20, 30], 'B': [40, 50, 60]}) mean_value = df['A'].mean()
print(mean_value)
```

3.Create a scatter plot using Matplotlib

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

x = np.random.rand(50) y = np.random.rand(50)

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

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

```
import seaborn as sns import pandas as pd import numpy as np import matplotlib.pyplot as plt

data = pd.DataFrame(np.random.rand(10, 4), columns=['A', 'B', 'C', 'D']) correlation_matrix =
data.corr()

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm') plt.title("Correlation Matrix
Heatmap") plt.show()
```

5.Generate a bar plot using Plotly

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

df = pd.DataFrame({'Category': ['A', 'B', 'C'], 'Values': [10, 20, 15]}) fig = px.bar(df, x='Category',
y='Values', title="Bar Plot") fig.show()
```

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

```
import pandas as pd

df = pd.DataFrame({'A': [1, 2, 3, 4]}) df['B'] = df['A'] * 2 # Creating new column B based on A
print(df)
```

7.Element-wise multiplication of two NumPy arrays

```
import numpy as np

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

result = a * b print(result)
```

8.Create a line plot with multiple lines using Matplotlib

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

x = np.linspace(0, 10, 100) y1 = np.sin(x) y2 = np.cos(x)
```

```
plt.plot(x, y1, label="sin(x)") plt.plot(x, y2, label="cos(x)") plt.legend() plt.title("Line Plot with
Multiple Lines") plt.show()
```

8.Filter rows where a column value is greater than a threshold in Pandas

```
import pandas as pd
```

```
df = pd.DataFrame({'A': [5, 10, 15, 20]}) filtered_df = df[df['A'] > 10] # Filtering rows where A > 10
print(filtered_df)
```

9.Create a histogram using Seaborn to visualize a distribution

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

```
data = np.random.randn(1000) # Generating random normal distribution sns.histplot(data,
bins=30, kde=True) plt.title("Histogram") plt.show()
```

10.Perform matrix multiplication using NumPy

```
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)
```

11.Use Pandas to load a CSV file and display its first 5 rows

```
import pandas as pd
```

```
df = pd.read_csv('your_file.csv') # Replace 'your_file.csv' with the actual file path print(df.head())
```
Create a 3D scatter plot using Plotly

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

```
df = pd.DataFrame({'x': np.random.rand(50), 'y': np.random.rand(50), 'z': np.random.rand(50)})
fig = px.scatter_3d(df, x='x', y='y', z='z', title="3D Scatter Plot") fig.show()
```