Q: What is NumPy, and why is it widely used in Python? A: NumPy is a Python library for numerical computing. It provides support for multi-dimensional arrays, mathematical functions, and linear algebra operations, making it essential for scientific computing and data analysis. It's widely used because of its speed, efficient memory usage, and ability to handle large datasets.

Q: How does broadcasting work in NumPy? A: Broadcasting allows NumPy to perform element-wise operations on arrays of different shapes by "stretching" the smaller array along the larger one's dimensions. This avoids explicit looping and improves performance.

Q: What are the differences between NumPy arrays and Python lists? A:

Speed: NumPy arrays are faster than Python lists due to optimized C implementation. Memory: Arrays are memory-efficient, storing data in contiguous blocks. Functionality: NumPy provides extensive mathematical and logical operations that are not available for Python lists. Q: What does the term "vectorized operation" mean in NumPy? A: A vectorized operation is one where operations are applied element-wise to entire arrays without using explicit loops, improving performance and readability.

Q: How does NumPy handle multidimensional arrays? A: NumPy represents multidimensional arrays as nested structures of fixed dimensions, with tools to perform operations like reshaping, slicing, and transposing.

Q: Why is NumPy's array slicing faster than Python's list slicing? A: NumPy slices are views of the original array, meaning no new memory is allocated. In contrast, Python list slicing creates a copy, which takes more time and memory.

Q: What are some advanced features of NumPy? A:Fancy indexing and masking. Linear algebra functions. Fourier transforms. Random sampling. Handling structured data using structured arrays.

Q: What is a Pandas DataFrame? A: A DataFrame is a two-dimensional, tabular data structure in Pandas with labeled rows and columns, similar to a spreadsheet or SQL table.

Q: Explain the use of the groupby() method in Pandas. A: groupby() is used to split a dataset into groups based on some criteria, apply a function to each group, and combine the results. It's commonly used for aggregation and transformation tasks.

Q: What is the purpose of the describe() function in Pandas? A: describe() generates summary statistics of numerical data, including count, mean, standard deviation, min, max, and quartiles.

Q: Why is handling missing data important in Pandas? A: Missing data can skew analysis and lead to incorrect results. Pandas offers tools like fillna(), dropna(), and interpolate() to manage missing values efficiently.

Q: What is the significance of hierarchical indexing in Pandas? A: Hierarchical indexing allows you to work with multi-level indexes, making it easier to manage and analyze multi-dimensional data.

Q: What is the role of a pivot table in Pandas? A: A pivot table is used to summarize, sort, and reorganize data. It allows you to compute aggregations (like sum or mean) for specific combinations of data features.

Q: Explain the difference between apply() and map() in Pandas. A:

apply() is used to apply a function along an axis (rows or columns) of a DataFrame. map() is applied element-wise to Series objects. Q: How does Pandas simplify time series analysis? A: Pandas provides specialized data structures and functions like resampling, shifting, rolling windows, and datetime indexing to efficiently analyze time-series data.

Q: Why is Seaborn preferred for statistical visualizations? A: Seaborn offers a high-level interface for creating attractive and informative statistical plots. It simplifies visualizations like heatmaps, pair plots, and categorical plots, which are harder to achieve with Matplotlib alone.

Q: What is a heatmap, and when should it be used? A: A heatmap is a graphical representation of data where individual values are represented as colors. It's commonly used for showing correlations or data density in a matrix format.

Q: What is the role of Seaborn's pairplot() function? A: pairplot() creates a grid of scatter plots for all numerical pairwise combinations of variables in a dataset, making it useful for exploratory data analysis.

Q: What are some common use cases for Seaborn? A:

Correlation heatmaps. Visualizing distributions (e.g., histograms, KDE plots). Comparing categories using box plots or violin plots. Showing relationships between variables with scatter or line plots.

Q: How does Matplotlib differ from Plotly? A:

Matplotlib: Great for static, publication-quality plots. Plotly: Best for interactive and dynamic visualizations. Q: What are the benefits of using Plotly for data visualization? A: Plotly offers interactive, browser-based visualizations, 3D plots, and dashboards, making it ideal for dynamic data exploration.

Q: What is the role of Bokeh in data visualization? A: Bokeh is a Python library for creating interactive, web-ready plots and dashboards. It's suitable for large datasets and supports streaming data.

```
# How do you create a 2D NumPy array and calculate the sum of each row?

import numpy as np
array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
row_sums = array.sum(axis=1)
print("Row sums:", row_sums)

```

```
Row sums: [ 6 15 24]
```

```python
# Perform matrix multiplication using NumPy.

import numpy as np
matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])
result = np.dot(matrix1, matrix2)
print("Matrix multiplication result:\n", result)

```

```
Matrix multiplication result:
 [[19 22]
 [43 50]]
```

```python
# Write a Pandas script to find the mean of a specific column in a DataFrame.

import pandas as pd
data = {'A': [10, 20, 30], 'B': [40, 50, 60]}
df = pd.DataFrame(data)
mean_value = df['A'].mean()
print("Mean of column A:", mean_value)

```

```
Mean of column A: 20.0
```

```python
# Create a DataFrame and add a new column based on an existing column.

import pandas as pd
data = {'A': [1, 2, 3], 'B': [4, 5, 6]}
df = pd.DataFrame(data)
df['C'] = df['A'] + df['B']
print(df)

```

```
   A  B  C
0  1  4  5
1  2  5  7
2  3  6  9
```

Generate   10 random numbers using numpy   Close

```python
# Generate a Pandas DataFrame and filter rows where a column value is greater than a t

import pandas as pd
data = {'A': [10, 20, 30], 'B': [40, 50, 60]}
df = pd.DataFrame(data)
filtered_df = df[df['A'] > 15]
print(filtered_df)


```

```
    A   B
1  20  50
2  30  60
```

```python
# Use Pandas to load a CSV file and display its first 5 rows.

import pandas as pd
df = pd.read_csv('your_file.csv')


print(df.head())

```

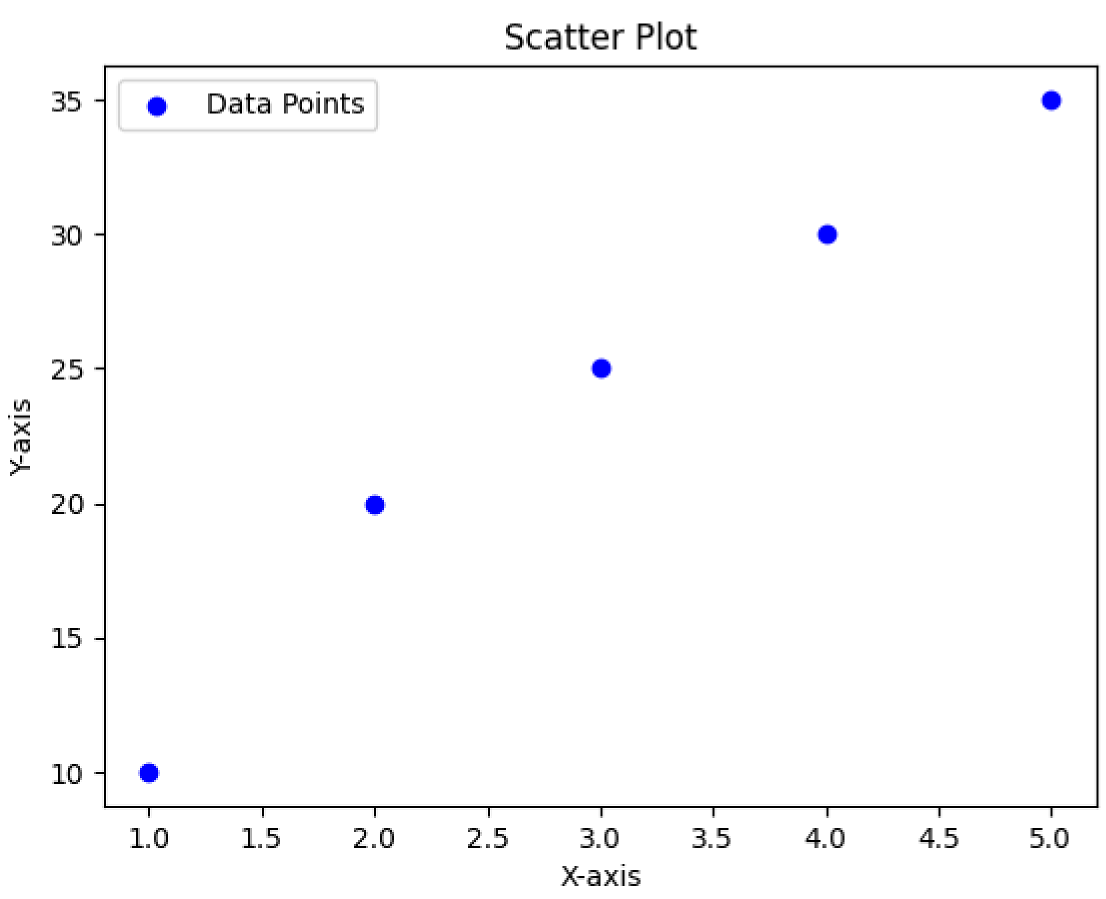
```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-17-601059b938d6> in <cell line: 5>()
      3 import pandas as pd
      4 
----> 5 df = pd.read_csv('your_file.csv')  # Replace 'your_file.csv' with the path
to your CSV file
      6 
      7 
```

4 frames

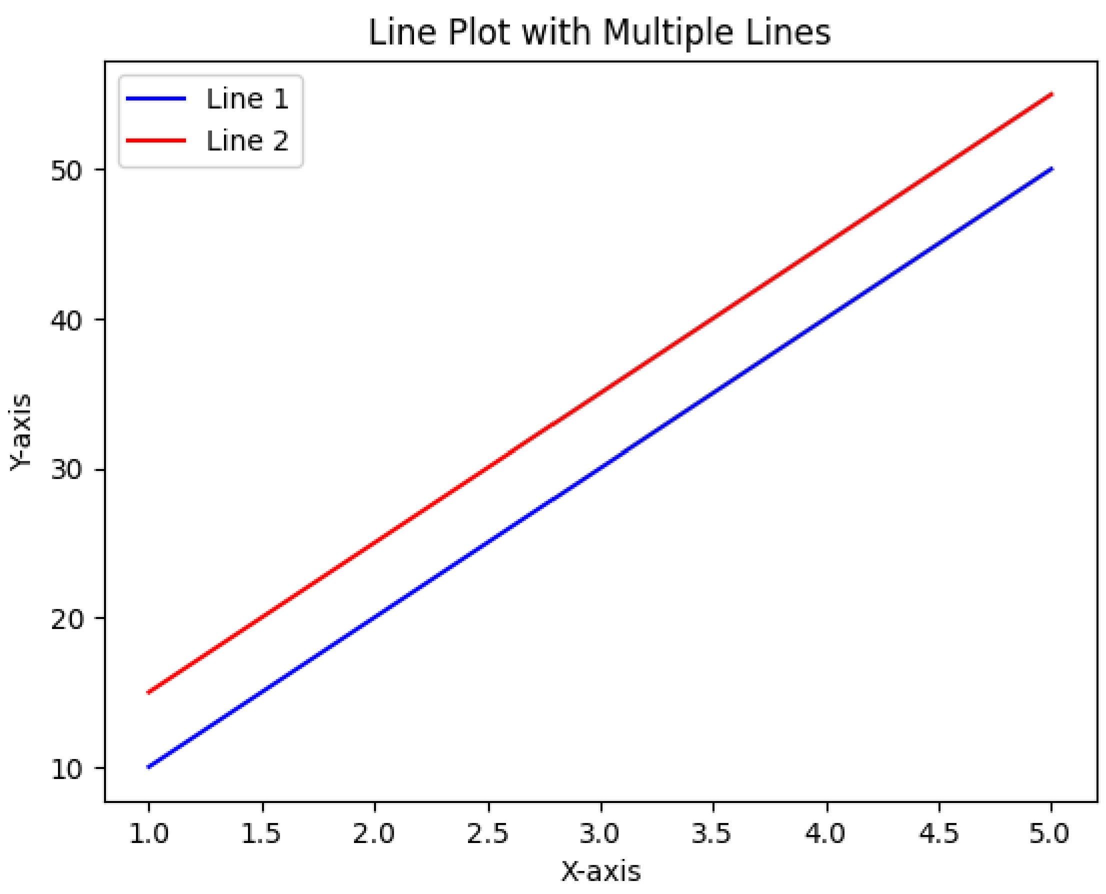
```
/usr/local/lib/python3.10/dist-packages/pandas/io/common.py in
get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors,
storage_options)
    871         if ioargs.encoding and "b" not in ioargs.mode:
    872             # Encoding
--> 873             handle = open(
    874                 handle,
    875                 ioargs.mode,
```

Next steps: Explain error

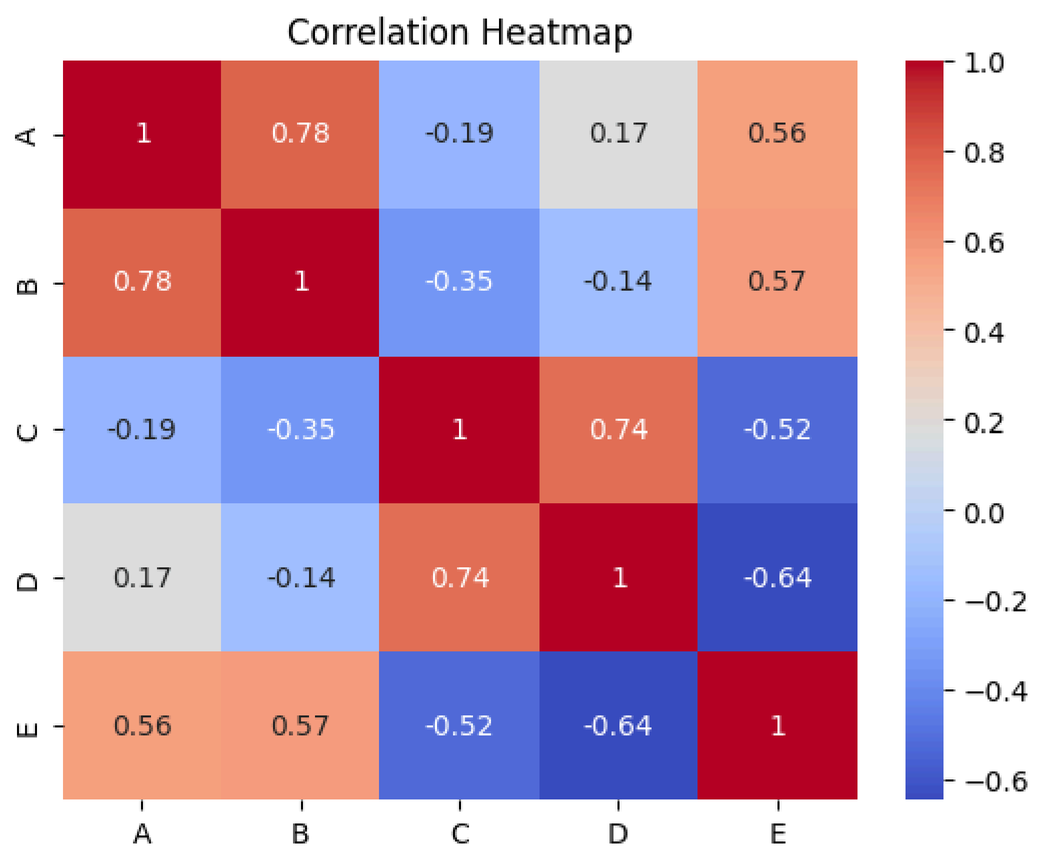
```python
# Create a scatter plot using Matplotlib.

import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 35]
plt.scatter(x, y, color='blue', label='Data Points')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot')
plt.legend()
plt.show()

```



```
# Create a line plot with multiple lines using Matplotlib.

import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y1 = [10, 20, 30, 40, 50]
y2 = [15, 25, 35, 45, 55]
plt.plot(x, y1, label='Line 1', color='blue')
plt.plot(x, y2, label='Line 2', color='red')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Line Plot with Multiple Lines')
plt.legend()
plt.show()

```

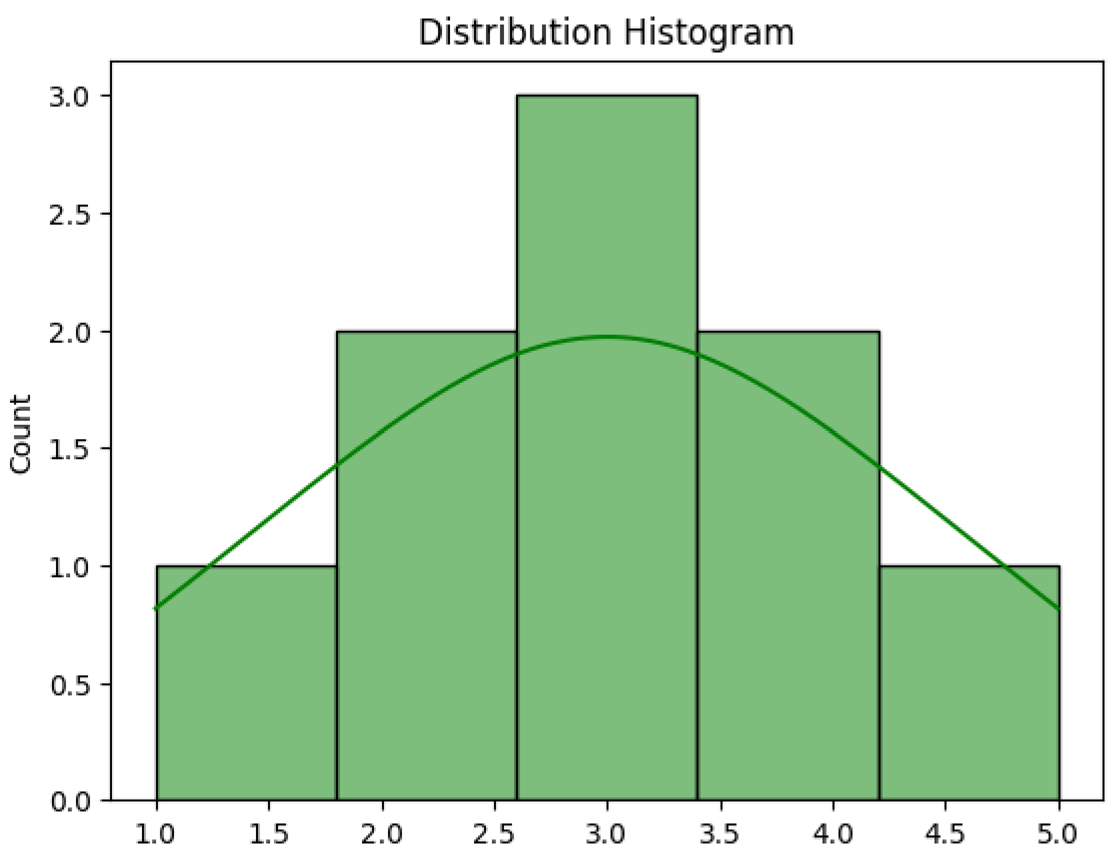


```
# How do you calculate the correlation matrix using Seaborn and visualize it with a he

import seaborn as sns
import pandas as pd
import numpy as np
data = np.random.rand(5, 5)
df = pd.DataFrame(data, columns=['A', 'B', 'C', 'D', 'E'])
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

```



	A	B	C	D	E
A	1	0.78	-0.19	0.17	0.56
B	0.78	1	-0.35	-0.14	0.57
C	-0.19	-0.35	1	0.74	-0.52
D	0.17	-0.14	0.74	1	-0.64
E	0.56	0.57	-0.52	-0.64	1

```
# Create a histogram using Seaborn to visualize a distribution.

import seaborn as sns
import matplotlib.pyplot as plt
data = [1, 2, 2, 3, 3, 3, 4, 4, 5]
sns.histplot(data, kde=True, color='green')
plt.title('Distribution Histogram')
plt.show()

```



```python
# Generate a bar plot using Plotly.

import plotly.express as px
data = {'Fruits': ['Apples', 'Bananas', 'Cherries'], 'Quantity': [10, 15, 7]}
fig = px.bar(data, x='Fruits', y='Quantity', title='Fruit Quantities')
fig.show()

```

Fruit Quantities

Generate print hello world using rot13 Close

```
# Create a 3D scatter plot using Plotly.
```