### **Data Toolkit Assignment**

### Creating Identical 2D Arrays in NumPy

#### Method 1: Using `np.full`

import numpy as np

array1 = np.full((3, 3), 7)

print(array1)

**Output:**

[[7 7 7]

[7 7 7]

[7 7 7]]

#### Method 2: Using `np.ones` and Multiplication

array2 = np.ones((3, 3)) \* 7

print(array2)

**Output:**

[[7. 7. 7.]

[7. 7. 7.]

[7. 7. 7.]]

#### Method 3: Using `np.tile`

array3 = np.tile(7, (3, 3))

print(array3)

**Output:**

[[7 7 7]

[7 7 7]

[7 7 7]]

### Generating and Reshaping an Array

array = np.linspace(1, 10, 100).reshape(10, 10)

print(array)

### Differences Between `np.array`, `np.asarray`, and `np.asanyarray`

* **`np.array`**: Always creates a new array.
* **`np.asarray`**: Converts input to an array, but does not copy if the input is already an array.
* **`np.asanyarray`**: Similar to np.asarray, but passes through subclasses of ndarray.

### Deep Copy vs Shallow Copy

* **Deep Copy**: Creates a new object and recursively copies all objects found in the original.
* **Shallow Copy**: Creates a new object, but inserts references into it to the objects found in the original.

### Generating and Rounding a 3x3 Array

array = np.random.uniform(5, 20, (3, 3))

rounded\_array = np.round(array, 2)

print(rounded\_array)

### Extracting Even and Odd Integers

array = np.random.randint(1, 11, (5, 6))

even\_integers = array[array % 2 == 0]

odd\_integers = array[array % 2 != 0]

print("Even integers:", even\_integers)

print("Odd integers:", odd\_integers)

### 3D Array Operations

array = np.random.randint(1, 11, (3, 3, 3))

indices\_max = np.argmax(array, axis=2)

print("Indices of max values:", indices\_max)

array1 = np.random.randint(1, 11, (3, 3, 3))

array2 = np.random.randint(1, 11, (3, 3, 3))

elementwise\_multiplication = array1 \* array2

print("Element-wise multiplication:", elementwise\_multiplication)

### Cleaning and Transforming 'Phone' Column

import pandas as pd

# Load the dataset

df = pd.read\_csv('People Data (1).csv')

# Clean and transform 'Phone' column

df['Phone'] = df['Phone'].str.replace(r'\D', '', regex=True).astype(float)

# Display table attributes and data types

print(df.dtypes)

print(df.head())

### Tasks Using People Dataset

#### a) Read the 'data.csv' file using pandas, skipping the first 50 rows.

df = pd.read\_csv('People Data (1).csv', skiprows=50)

#### b) Only read the columns: 'Last Name', ‘Gender’,’Email’,‘Phone’ and ‘Salary’ from the file.

df\_filtered = df[['Last Name', 'Gender', 'Email', 'Phone', 'Salary']]

#### c) Display the first 10 rows of the filtered dataset.

print(df\_filtered.head(10))

#### d) Extract the ‘Salary’ column as a Series and display its last 5 values.

salary\_series = df\_filtered['Salary']

print(salary\_series.tail(5))

### Filtering and Selecting Rows

filtered\_df = df[(df['Last Name'].str.contains('Duke')) &

(df['Gender'] == 'Female') &

(df['Salary'] < 85000)]

print(filtered\_df)

### Creating a 7x5 DataFrame

random\_series = pd.Series(np.random.randint(1, 7, 35))

df\_7x5 = random\_series.values.reshape(7, 5)

print(df\_7x5)

### Creating Two Different Series and Joining Them

series1 = pd.Series(np.random.randint(10, 51, 50))

series2 = pd.Series(np.random.randint(100, 1001, 50))

df\_series = pd.DataFrame({'col1': series1, 'col2': series2})

print(df\_series)

### Operations Using People Dataset

#### a) Delete the 'Email', 'Phone', and 'Date of birth' columns from the dataset.

df\_dropped = df.drop(columns=['Email', 'Phone', 'Date of birth'])

#### b) Delete the rows containing any missing values.

df\_cleaned = df\_dropped.dropna()

print(df\_cleaned)

### Scatter Plot Using Matplotlib and NumPy

import matplotlib.pyplot as plt

x = np.random.rand(100)

y = np.random.rand(100)

plt.scatter(x, y, color='red', marker='o', label='Scatter Points')

plt.axhline(y=0.5, color='blue', linestyle='--', label='y = 0.5')

plt.axvline(x=0.5, color='green', linestyle=':', label='x = 0.5')

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.title('Advanced Scatter Plot of Random Values')

plt.legend()

plt.show()

### Time-Series Dataset and Plotting

date\_rng = pd.date\_range(start='1/1/2020', end='1/01/2021', freq='D')

df\_time\_series = pd.DataFrame(date\_rng, columns=['Date'])

df\_time\_series['Temperature'] = np.random.randint(20, 35, size=(len(date\_rng)))

df\_time\_series['Humidity'] = np.random.randint(30, 70, size=(len(date\_rng)))

fig, ax1 = plt.subplots()

ax1.set\_xlabel('Date')

ax1.set\_ylabel('Temperature', color='tab:red')

ax1.plot(df\_time\_series['Date'], df\_time\_series['Temperature'], color='tab:red')

ax2 = ax1.twinx()

ax2.set\_ylabel('Humidity', color='tab:blue')

ax2.plot(df\_time\_series['Date'], df\_time\_series['Humidity'], color='tab:blue')

plt.title('Temperature and Humidity Over Time')

plt.show()

### Histogram with PDF Overlay

data = np.random.normal(0, 1, 1000)

count, bins, ignored = plt.hist(data, 30, density=True, alpha=0.6, color='g')

mu, sigma = 0, 1

pdf = 1/(sigma \* np.sqrt(2 \* np.pi)) \* np.exp( - (bins - mu)\*\*2 / (2 \* sigma\*\*2) )

plt.plot(bins, pdf, linewidth=2, color='r')

plt.xlabel('Value')

plt.ylabel('Frequency/Probability')

plt.title('Histogram with PDF Overlay')

plt.show()

### Seaborn Scatter Plot

import seaborn as sns

x = np.random.rand(100)

y = np.random.rand(100)

quadrants = ['Q1' if (i > 0.5 and j > 0.5) else 'Q2' if (i <= 0.5 and j > 0.5) else 'Q3' if (i <= 0.5 and j <= 0.5) else 'Q4' for i, j in zip(x, y)]

df\_scatter = pd.DataFrame({'x': x, 'y': y, 'quadrant': quadrants})

sns.scatterplot(data=df\_scatter, x='x', y='y', hue='quadrant')

plt.axhline(0.5, ls='--', color='grey')

plt.axvline(0.5, ls='--', color='grey')

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.title('Quadrant-wise Scatter Plot')

plt.legend()

plt.show()

### Creating Two Different Series and Joining Them

import pandas as pd

import numpy as np

# Creating the first Series with random numbers ranging from 10 to 50

series1 = pd.Series(np.random.randint(10, 51, 50))

# Creating the second Series with random numbers ranging from 100 to 1000

series2 = pd.Series(np.random.randint(100, 1001, 50))

# Creating a DataFrame by joining these Series by column

df\_series = pd.DataFrame({'col1': series1, 'col2': series2})

print(df\_series)

### Operations Using People Dataset

#### a) Delete the 'Email', 'Phone', and 'Date of birth' columns from the dataset.

# Load the dataset

df = pd.read\_csv('People Data (1).csv')

# Delete the specified columns

df\_dropped = df.drop(columns=['Email', 'Phone', 'Date of birth'])

#### b) Delete the rows containing any missing values.

# Delete rows with any missing values

df\_cleaned = df\_dropped.dropna()

#### d) Print the final output

print(df\_cleaned)

### Scatter Plot Using Matplotlib and NumPy

import matplotlib.pyplot as plt

# Creating two NumPy arrays with 100 random float values between 0 and 1

x = np.random.rand(100)

y = np.random.rand(100)

# Creating the scatter plot

plt.scatter(x, y, color='red', marker='o', label='Scatter Points')

# Adding a horizontal line at y = 0.5

plt.axhline(y=0.5, color='blue', linestyle='--', label='y = 0.5')

# Adding a vertical line at x = 0.5

plt.axvline(x=0.5, color='green', linestyle=':', label='x = 0.5')

# Labeling the axes

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

# Setting the title of the plot

plt.title('Advanced Scatter Plot of Random Values')

# Displaying the legend

plt.legend()

# Showing the plot

plt.show()

### Time-Series Dataset and Plotting

# Creating a time-series dataset

date\_rng = pd.date\_range(start='1/1/2020', end='1/01/2021', freq='D')

df\_time\_series = pd.DataFrame(date\_rng, columns=['Date'])

df\_time\_series['Temperature'] = np.random.randint(20, 35, size=(len(date\_rng)))

df\_time\_series['Humidity'] = np.random.randint(30, 70, size=(len(date\_rng)))

# Plotting Temperature and Humidity on the same plot with different y-axes

fig, ax1 = plt.subplots()

ax1.set\_xlabel('Date')

ax1.set\_ylabel('Temperature', color='tab:red')

ax1.plot(df\_time\_series['Date'], df\_time\_series['Temperature'], color='tab:red')

ax2 = ax1.twinx()

ax2.set\_ylabel('Humidity', color='tab:blue')

ax2.plot(df\_time\_series['Date'], df\_time\_series['Humidity'], color='tab:blue')

plt.title('Temperature and Humidity Over Time')

plt.show()

### Histogram with PDF Overlay

# Creating a NumPy array with 1000 samples from a normal distribution

data = np.random.normal(0, 1, 1000)

# Plotting the histogram

count, bins, ignored = plt.hist(data, 30, density=True, alpha=0.6, color='g')

# Overlaying the PDF

mu, sigma = 0, 1

pdf = 1/(sigma \* np.sqrt(2 \* np.pi)) \* np.exp( - (bins - mu)\*\*2 / (2 \* sigma\*\*2) )

plt.plot(bins, pdf, linewidth=2, color='r')

# Labeling the axes

plt.xlabel('Value')

plt.ylabel('Frequency/Probability')

# Setting the title of the plot

plt.title('Histogram with PDF Overlay')

# Showing the plot

plt.show()

### Seaborn Scatter Plot

import seaborn as sns

# Creating two random arrays

x = np.random.rand(100)

y = np.random.rand(100)

# Determining the quadrants

quadrants = ['Q1' if (i > 0.5 and j > 0.5) else 'Q2' if (i <= 0.5 and j > 0.5) else 'Q3' if (i <= 0.5 and j <= 0.5) else 'Q4' for i, j in zip(x, y)]

# Creating a DataFrame for the scatter plot

df\_scatter = pd.DataFrame({'x': x, 'y': y, 'quadrant': quadrants})

# Plotting the scatter plot

sns.scatterplot(data=df\_scatter, x='x', y='y', hue='quadrant')

# Adding horizontal and vertical lines

plt.axhline(0.5, ls='--', color='grey')

plt.axvline(0.5, ls='--', color='grey')

# Labeling the axes

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

# Setting the title of the plot

plt.title('Quadrant-wise Scatter Plot')

# Displaying the legend

plt.legend()

# Showing the plot

plt.show()

### Bokeh: Sine Wave Function Line Chart

from bokeh.plotting import figure, show

from bokeh.io import output\_notebook

import numpy as np

output\_notebook()

# Generate data

x = np.linspace(0, 4 \* np.pi, 100)

y = np.sin(x)

# Create a new plot

p = figure(title="Sine Wave Function", x\_axis\_label='x', y\_axis\_label='y')

# Add a line renderer with legend and line thickness

p.line(x, y, legend\_label="Sine Wave", line\_width=2)

# Add grid lines

p.xgrid.grid\_line\_color = 'gray'

p.ygrid.grid\_line\_color = 'gray'

# Show the results

show(p)

### Bokeh: Random Categorical Bar Chart

from bokeh.plotting import figure, show

from bokeh.io import output\_notebook

from bokeh.models import ColumnDataSource, HoverTool

import numpy as np

import pandas as pd

output\_notebook()

# Generate random categorical data

categories = ['A', 'B', 'C', 'D', 'E']

values = np.random.randint(1, 100, size=len(categories))

# Create a DataFrame

data = pd.DataFrame({'categories': categories, 'values': values})

# Create a ColumnDataSource

source = ColumnDataSource(data)

# Create a new plot

p = figure(x\_range=categories, title="Random Categorical Bar Chart", x\_axis\_label='Categories', y\_axis\_label='Values')

# Add bars

p.vbar(x='categories', top='values', width=0.9, source=source, legend\_field="categories",

line\_color='white', fill\_color='blue')

# Add hover tool

hover = HoverTool()

hover.tooltips = [("Category", "@categories"), ("Value", "@values")]

p.add\_tools(hover)

# Show the results

show(p)

### Plotly: Simple Line Plot

import plotly.graph\_objects as go

import numpy as np

# Generate random data

x = np.arange(0, 10, 0.1)

y = np.random.randn(len(x))

# Create a line plot

fig = go.Figure(data=go.Scatter(x=x, y=y, mode='lines', name='Random Data'))

# Update layout

fig.update\_layout(title='Simple Line Plot', xaxis\_title='X-axis', yaxis\_title='Y-axis')

# Show the plot

fig.show()

### Plotly: Interactive Pie Chart

import plotly.graph\_objects as go

import numpy as np

# Generate random data

labels = ['A', 'B', 'C', 'D', 'E']

values = np.random.randint(10, 100, size=len(labels))

# Create a pie chart

fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])

# Update layout

fig.update\_layout(title='Interactive Pie Chart')

# Show the plot

fig.show()