People Data set Assignment

1. Three Methods to Create Identical 2D Arrays in NumPy

```
import numpy as np

# Method 1: Using np.array
array1 = np.array([[1, 2, 3], [4, 5, 6]])
print(array1)

# Method 2: Using np.zeros and then filling values
array2 = np.zeros((2, 3))
array2[0, :] = [1, 2, 3]
array2[1, :] = [4, 5, 6]
print(array2)

# Method 3: Using np.full
array3 = np.full((2, 3), 0)
array3[0, :] = [1, 2, 3]
array3[1, :] = [4, 5, 6]
print(array3)
```

Output for all methods:

```
[[1 2 3]
 [4 5 6]]
```

2. Generate and Reshape an Array

```
# Generate 100 evenly spaced numbers between 1 and 10
array_1d = np.linspace(1, 10, 100)
# Reshape to a 2D array of shape (10, 10)
array_2d = array_1d.reshape(10, 10)
print(array_2d)
```

3. Differences Between np.array, np.asarray, and np.asanyarray

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

4. 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.

5. Generate and Round a 3x3 Array

```
# Generate a 3x3 array with random floats between 5 and 20
random_array = np.random.uniform(5, 20, (3, 3))
# Round each number to 2 decimal places
rounded_array = np.round(random_array, 2)
print(rounded_array)
```

6. Create and Extract Integers from a 5x6 Array

```
# Create a 5x6 array with random integers between 1 and 10
random_int_array = np.random.randint(1, 11, (5, 6))
print("Original Array:\n", random_int_array)

# Extract all even integers
even_integers = random_int_array[random_int_array % 2 == 0]
print("Even Integers:\n", even_integers)

# Extract all odd integers
odd_integers = random_int_array[random_int_array % 2 != 0]
print("Odd Integers:\n", odd_integers)
```

7. Create a 3D NumPy Array and Perform Operations

```
# Create a 3D NumPy array of shape (3, 3, 3) containing random
integers between 1 and 10
array_3d = np.random.randint(1, 11, (3, 3, 3))
print("3D Array:\n", array_3d)

# a) Find the indices of the maximum values along each depth level
(third axis)
max_indices = np.argmax(array_3d, axis=2)
print("Indices of maximum values along each depth level:\n",
max_indices)

# b) Perform element-wise multiplication of the array with itself
multiplied_array = array_3d * array_3d
print("Element-wise multiplied array:\n", multiplied_array)
```

8. Clean and Transform the 'Phone' Column

```
import pandas as pd
import re

# Function to clean phone numbers
def clean_phone_number(phone):
    return int(re.sub(r'\D', '', phone)) if phone else None
```

```
# Load the dataset
df = pd.read_csv('People Data.csv')

# Clean the 'Phone' column
df['Phone'] = df['Phone'].apply(clean_phone_number)

# Display the table attributes and data types of each column
print("Table attributes and data types:\n", df.dtypes)
```

Output:

```
Table attributes and data types:
 Index            int64
User Id          object
First Name       object
Last Name        object
Gender           object
Email            object
Phone           float64
Date of birth    object
Job Title        object
Salary            int64
dtype: object
```

9. Perform Tasks Using the People Dataset

```
# Read the 'data.csv' file using pandas, skipping the first 50 rows
df_filtered = pd.read_csv('People Data.csv', skiprows=50,
usecols=['Last Name', 'Gender', 'Email', 'Phone', 'Salary'])

# Display the first 10 rows of the filtered dataset
print("First 10 rows of the filtered dataset:\n",
df_filtered.head(10))

# Extract the ‘Salary’ column as a Series and display its last 5
values
salary_series = df_filtered['Salary']
print("Last 5 values of the 'Salary' column:\n",
salary_series.tail(5))

# Filter and select rows where 'Last Name' contains 'Duke', 'Gender'
contains 'Female', and 'Salary' is less than 85000
filtered_rows = df_filtered[(df_filtered['Last
Name'].str.contains('Duke')) &

(df_filtered['Gender'].str.contains('Female')) &
                            (df_filtered['Salary'] < 85000)]
print("Filtered rows:\n", filtered_rows)
```

10. Create a 7x5 DataFrame with Random Integers

```
# Create a 7x5 DataFrame in Pandas using a series generated from 35
random integers between 1 to 6
random_integers = np.random.randint(1, 7, 35)
df_random = pd.DataFrame(random_integers.reshape(7, 5))
print("7x5 DataFrame with random integers between 1 and 6:\n",
df_random)
```

11. Create Two Different Series and a DataFrame

```
# Create the first Series with random numbers ranging from 10 to 50
series1 = pd.Series(np.random.randint(10, 51, 50))

# Create the second Series with random numbers ranging from 100 to
1000
series2 = pd.Series(np.random.randint(100, 1001, 50))

# Create a DataFrame by joining these Series by column
df_series = pd.DataFrame({'col1': series1, 'col2': series2})
print("DataFrame with two Series:\n", df_series)
```

12. Perform Operations Using the People Dataset

```
# Delete the 'Email', 'Phone', and 'Date of birth' columns from the
dataset
df_cleaned = df.drop(columns=['Email', 'Phone', 'Date of birth'])

# Delete the rows containing any missing values
df_cleaned = df_cleaned.dropna()

# Print the final output
print("Final cleaned dataset:\n", df_cleaned)
```

13. Create Two NumPy Arrays and Plot Using Matplotlib

```
import matplotlib.pyplot as plt

# Create two NumPy arrays, x and y, each containing 100 random float
values between 0 and 1
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot using x and y
plt.scatter(x, y, color='red', marker='o', label='Random Points')

# Add a horizontal line at y = 0.5
plt.axhline(y=0.5, color='blue', linestyle='--', label='y = 0.5')

# Add a vertical line at x = 0.5
plt.axvline(x=0.5, color='green', linestyle=':', label='x = 0.5')
```

```
# Label the axes
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Set the title of the plot
plt.title('Advanced Scatter Plot of Random Values')

# Display a legend
plt.legend()

# Show the plot
plt.show()
```

14. Create a Time-Series Dataset and Plot Using Matplotlib

```
import pandas as pd
import matplotlib.pyplot as plt

# Create a time-series dataset
data = {
    'Date': pd.date_range(start='2023-01-01', periods=100,
freq='D'),
    'Temperature': np.random.uniform(15, 35, 100),
    'Humidity': np.random.uniform(40, 80, 100)
}
df_time_series = pd.DataFrame(data)

# Plot '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')
ax1.tick_params(axis='y', labelcolor='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')
ax2.tick_params(axis='y', labelcolor='tab:blue')

plt.title('Temperature and Humidity Over Time')
fig.tight_layout()
plt.show()
```

15. Create a Histogram with PDF Overlay Using Matplotlib

```
import scipy.stats as stats
```

```
# Create a NumPy array containing 1000 samples from a normal
distribution
data = np.random.normal(loc=0, scale=1, size=1000)

# Plot a histogram of the data with 30 bins
plt.hist(data, bins=30, density=True, alpha=0.6, color='g')

# Overlay a line plot representing the normal distribution's PDF
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, 0, 1)
plt.plot(x, p, 'k', linewidth=2)

# Label the axes and set the title
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')
plt.title('Histogram with PDF Overlay')
plt.show()
```

16. Create a Seaborn Scatter Plot

```
import seaborn as sns

# Create two random arrays
x = np.random.randn(100)
y = np.random.randn(100)

# Determine the quadrant for each point
quadrant = np.where((x >= 0) & (y >= 0), 'Q1',
                    np.where((x < 0) & (y >= 0), 'Q2',
                             np.where((x < 0) & (y < 0), 'Q3',
'Q4')))

# Create a DataFrame
df_scatter = pd.DataFrame({'x': x, 'y': y, 'quadrant': quadrant})

# Create a scatter plot
sns.scatterplot(data=df_scatter, x='x', y='y', hue='quadrant',
palette='viridis')

# Label the axes and set the title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Quadrant-wise Scatter Plot')
plt.legend(title='Quadrant')
plt.show()
```

17. Create a Line Chart Using Bokeh

```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
import numpy as np
```

```
output_notebook()

# Generate data for a sine wave
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)

# Show the results
show(p)
```

18. Create a Bar Chart Using Bokeh

```
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import ColumnDataSource, HoverTool
import pandas as pd

output_notebook()

# Generate random categorical data
categories = ['A', 'B', 'C', 'D', 'E']
values = np.random.randint(1, 10, size=len(categories))

# Create a DataFrame
df_bar = pd.DataFrame({'categories': categories, 'values': values})

# Create a ColumnDataSource
source = ColumnDataSource(df_bar)

# Create a new plot
p = figure(x_range=categories, title="Random Categorical Bar Chart",
toolbar_location=None, tools="")

# Add bars
p.vbar(x='categories', top='values', width=0.9, source=source,
legend_field="categories",
       line_color='white', fill_color='navy')

# Add hover tool
hover = HoverTool()
hover.tooltips = [("Category", "@categories"), ("Value", "@values")]
p.add_tools(hover)

# Show the results
show(p)
```

19. Create a Basic Line Plot Using Plotly

```
import plotly.express as px

# Generate random data
x = np.linspace(0, 10, 100)
y = np.random.randn(100)

# Create a line plot
fig = px.line(x=x, y=y, labels={'x': 'X-axis', 'y': 'Y-axis'},
title='Simple Line Plot')

# Show the plot
fig.show()
```

20. Create an Interactive Pie Chart Using Plotly

```
import plotly.graph_objects as go

# 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)])

# Set the title
fig.update_layout(title_text='Interactive Pie Chart')

# Show the plot
fig.show()
```