Assignment 44 Solutions

Name: Ashesh Jyoti Majumdar

Assignment files -> Assignment_44

1. Demonstrate three different methods for creating identical 2D arrays in NumPy. Provide the code for each method and the final output after each method.

->

```
import numpy as np

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


# Method 2: Using np.ones and scalar multiplication
array2 = np.ones((2, 3)) * np.array([1, 2, 3])
print("Method 2:", array2)
'''
Output 2:
Method 2: [[1. 2. 3.]
[1. 2. 3.]]
'''


# Method 3: Using np.full
array3 = np.full((2, 3), [1, 2, 3])
print("Method 3:", array3)
'''
Output 3:
```

```
Method 3: [[1 2 3]
[1 2 3]]
'''
```

2. Using the Numpy function, generate an array of 100 evenly spaced numbers between 1 and 10 and reshape that 1D array into a 2D array.

->

```
import numpy as np
array = np.linspace(1, 10, 100).reshape(10, 10)
print("2D Array:", array)
'''
Output:
$ python3 100_evenlyspaced_number_2d_3d.py
2D Array: [[ 1.          1.09090909  1.18181818  1.27272727
1.36363636  1.45454545
  1.54545455  1.63636364  1.72727273  1.81818182]
[ 1.90909091  2.          2.09090909  2.18181818  2.27272727
2.36363636
  2.45454545  2.54545455  2.63636364  2.72727273]
[ 2.81818182  2.90909091  3.          3.09090909  3.18181818
3.27272727
  3.36363636  3.45454545  3.54545455  3.63636364]
[ 3.72727273  3.81818182  3.90909091  4.          4.09090909
4.18181818
  4.27272727  4.36363636  4.45454545  4.54545455]
[ 4.63636364  4.72727273  4.81818182  4.90909091  5.
5.09090909
  5.18181818  5.27272727  5.36363636  5.45454545]
[ 5.54545455  5.63636364  5.72727273  5.81818182  5.90909091  6.
  6.09090909  6.18181818  6.27272727  6.36363636]
[ 6.45454545  6.54545455  6.63636364  6.72727273  6.81818182
6.90909091
  7.          7.09090909  7.18181818  7.27272727]
[ 7.36363636  7.45454545  7.54545455  7.63636364  7.72727273
7.81818182
  7.90909091  8.          8.09090909  8.18181818]
[ 8.27272727  8.36363636  8.45454545  8.54545455  8.63636364
8.72727273
  8.81818182  8.90909091  9.          9.09090909]
[ 9.18181818  9.27272727  9.36363636  9.45454545  9.54545455
9.63636364
  9.72727273  9.81818182  9.90909091 10.        ]]
asheshjyoti@ubuntu:~/house/asheshjyoti/workspace/Learning/Python_and_
machineLearning_guided/PW_DataSciencewithML/assignment44 <master>
'''
```

3. Explain the following terms:

- The difference between np.array, np.asarray, and np.asanyarray:
    - np.array creates a new array object.
    - np.asarray converts the input to an array but does not copy if the input is already an ndarray.
    - np.asanyarray is similar to np.asarray, but it passes through subclasses of ndarray.
- The difference between deep copy and 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.

4. Generate a 3x3 array with random floating-point numbers between 5 and 20. Then, round each number in the array to 2 decimal places.

->

```
import numpy as np
array = np.random.uniform(5, 20, (3, 3))
```

```
rounded_array = np.round(array, 2)
print("Rounded Array:", rounded_array)
'''
Output:
$ python3 3x3_array_random_floating_point_number.py
Rounded Array: [[14.75  5.06 18.89]
[ 6.46 18.5   8.32]
[10.61 18.87 11.27]]
asheshjyoti@ubuntu:~/house/asheshjyoti/workspace/Learning/Python_and_
machineLearning_guided/PW_DataSciencewithML/assignment44 <master>
'''
```

5. Create a NumPy array with random integers between 1 and 10 of shape (5, 6). After creating the array perform the following operations:

a)Extract all even integers from array.

b)Extract all odd integers from arrayX

->

```
import numpy as np
array = np.random.randint(1, 11, (5, 6))
print("Original Array:", array)

# a) Extract all even integers from array
even_integers = array[array % 2 == 0]
print("Even Integers:", even_integers)

# b) Extract all odd integers from array
odd_integers = array[array % 2 != 0]
print("Odd Integers:", odd_integers)
'''
Output:
$ python3 Q5_Random_integer.py
Original Array: [[ 5  5  8  5  2  6]
[ 9  1  9  3  3  6]
[ 8  3 10  3  2 10]
[ 2  8  1  9 10  8]
[ 9  7  3  8  6  1]]
Even Integers: [ 8  2  6  6  8 10  2 10  2  8 10  8  8  6]
```

```
Odd Integers: [5 5 5 9 1 9 3 3 3 3 1 9 9 7 3 1]
asheshjyoti@ubuntu:~/house/asheshjyoti/workspace/Learning/Python_and_machin
eLearning_guided/PW_DataSciencewithML/assignment44 <master>
'''
```

6. Create a 3D NumPy array of shape (3, 3, 3) containing random integers between 1 and 10. Perform the following operations:

a) Find the indices of the maximum values along each depth level (third axis).

b) Perform element-wise multiplication of between both arrayX

->

```
import numpy as np
array = np.random.randint(1, 11, (3, 3, 3))
print("3D Array:", array)

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

# b) Element-wise multiplication between two arrays
array2 = np.random.randint(1, 11, (3, 3, 3))
elementwise_multiplication = array * array2
print("Element-wise Multiplication:", elementwise_multiplication)
'''
Output:
$ python3 Q6.py
3D Array: [[[ 5  5  9]
  [ 9  7  1]
  [ 8  6  4]]

 [[10  9  5]
  [ 2  8  3]
  [ 8  4  6]]
```

```
[[ 4  4  5]
 [10  7  9]
 [ 4  2  2]]]
Indices of Maximum Values: [[2 0 0]
[0 1 0]
[2 0 0]]
Element-wise Multiplication: [[[ 50  45  18]
 [ 18  70   3]
 [ 16  54  36]]

[[ 40  72  35]
 [ 18   8   6]
 [ 24  20  18]]

[[ 20  40  20]
 [100  14  27]
 [ 12  16   6]]]
asheshjyoti@ubuntu:~/house/asheshjyoti/workspace/Learning/Python_and_machin
eLearning_guided/PW_DataSciencewithML/assignment44 <master>
'''
```

7. Clean and transform the 'Phone' column in the sample dataset to remove non-numeric characters and convert it to a numeric data type. Also display the table attributes and data types of each column.

->

```
import pandas as pd

# Load dataset
data = pd.read_csv('People Data.csv')

# Clean and transform the 'Phone' column
data['Phone'] = data['Phone'].str.replace(r'\D', '',
regex=True).astype(float)

# Display table attributes and data types
print(data.dtypes)
'''
Output:
$ python3 Q7.py
```

```
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
asheshjyoti@ubuntu:~/house/asheshjyoti/workspace/Learning/Python_and_machineLearning_guided/PW_DataSciencewithML/assignment44 <master>
'''
```

8)Perform the following tasks using people dataset:

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

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

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

d) Extract the ‘Salary’' column as a Series and display its last 5 valuesX

->

```
import pandas as pd

# Load the dataset, ensuring the header is included even when skipping the first 50 rows
data = pd.read_csv('People Data.csv', skiprows=range(1, 51))

# Only read the columns: 'Last Name', 'Gender', 'Email', 'Phone', and 'Salary'
data_filtered = data[['Last Name', 'Gender', 'Email', 'Phone', 'Salary']]

# Display the first 10 rows of the filtered dataset
print(data_filtered.head(10))

# Extract the 'Salary' column as a Series and display its last 5 values
```

```
salary_series = data_filtered['Salary']
print(salary_series.tail(5))


# Only read the columns: 'Last Name', 'Gender', 'Email', 'Phone', and
'Salary'
data_filtered = data[['Last Name', 'Gender', 'Email', 'Phone', 'Salary']]
print(data_filtered.head(10))

'''
Output:
asheshjyoti@ubuntu:~/house/asheshjyoti/workspace/Learning/Python_and_machin
eLearning_guided/PW_DataSciencewithML/assignment44 <master>
$ python3 Q8.py
 Last Name  Gender  ...                  Phone  Salary
0    Zavala    Male  ...  001-859-448-9935x54536   80000
1     Carey  Female  ...   001-274-739-8470x814   70000
2     Hobbs  Female  ...       241.179.9509x498   60000
3    Reilly    Male  ...      207.797.8345x6177  100000
4    Conrad    Male  ...   001-599-042-7428x143   50000
5      Cole    Male  ...           663-280-5834   85000
6   Donovan    Male  ...                    NaN   65000
7    Little  Female  ...      125.219.3673x0076   60000
8    Dawson  Female  ...     650-748-3069x64529   60000
9      Page    Male  ...       849.500.6331x717   60000

[10 rows x 5 columns]
945     90000
946     50000
947     60000
948    100000
949     90000
Name: Salary, dtype: int64
 Last Name  Gender  ...                  Phone  Salary
0    Zavala    Male  ...  001-859-448-9935x54536   80000
1     Carey  Female  ...   001-274-739-8470x814   70000
2     Hobbs  Female  ...       241.179.9509x498   60000
3    Reilly    Male  ...      207.797.8345x6177  100000
4    Conrad    Male  ...   001-599-042-7428x143   50000
5      Cole    Male  ...           663-280-5834   85000
6   Donovan    Male  ...                    NaN   65000
7    Little  Female  ...      125.219.3673x0076   60000
8    Dawson  Female  ...     650-748-3069x64529   60000
```

```
9      Page     Male  ...        849.500.6331x717   60000

[10 rows x 5 columns]
asheshjyoti@ubuntu:~/house/asheshjyoti/workspace/Learning/Python_and_machin
eLearning_guided/PW_DataSciencewithML/assignment44 <master>
'''
```

9. Filter and select rows from the People_Dataset, where the “Last Name' column contains the name 'Duke', 'Gender' column contains the word Female and ‘Salary’ should be less than 85000.

->

```
import pandas as pd

# Load the dataset, ensuring the header is included even when skipping the
first 50 rows
data = pd.read_csv('People Data.csv', skiprows=range(1, 51))
filtered_data = data[(data['Last Name'].str.contains('Duke')) &
(data['Gender'] == 'Female') & (data['Salary'] < 85000)]
print(filtered_data)
'''
Output:
$ python3 Q9.py
    Index          User Id  ...        Job Title Salary
160    211  DF17975CC0a0373  ...  Producer, radio  50000
407    458  dcE1B7DE83c1076  ...        Herbalist  50000
679    730  c9b482D7aa3e682  ...     Nurse, adult  70000

[3 rows x 10 columns]
asheshjyoti@ubuntu:~/house/asheshjyoti/workspace/Learning/Python_and_machin
eLearning_guided/PW_DataSciencewithML/assignment44 <master>
'''
```

10. Create a 7*5 DataFrame in Pandas using a series generated from 35 random integers between 1 to 6.

->

```
import pandas as pd
import numpy as np

# Load the dataset, ensuring the header is included even when skipping the first 50 rows
data = pd.read_csv('People Data.csv', skiprows=range(1, 51))
df = pd.DataFrame(np.random.randint(1, 7, size=(7, 5)))
print(df)
'''
Output:
$ python3 Q10.py
  0  1  2  3  4
0  5  6  4  3  3
1  4  4  6  6  5
2  3  1  3  6  5
3  1  1  5  2  5
4  1  3  2  4  6
5  5  5  3  1  5
6  6  2  1  4  1
asheshjyoti@ubuntu:~/house/asheshjyoti/workspace/Learning/Python_and_machineLearning_guided/PW_DataSciencewithML/assignment44 <master>
'''
```

11. Create two different Series, each of length 50, with the following criteria:

a) The first Series should contain random numbers ranging from 10 to 50.

b) The second Series should contain random numbers ranging from 100 to 1000.

c) Create a DataFrame by joining these Series by column, and, change the names of the columns to 'col1', 'col2', etc.

->

```
import numpy as np
import pandas as pd
```

```
# a) The first Series should contain random numbers ranging from 10 to 50.
series1 = pd.Series(np.random.randint(10, 51, size=50))

# b) The second Series should contain random numbers ranging from 100 to
1000.
series2 = pd.Series(np.random.randint(100, 1001, size=50))

# c) Create a DataFrame by joining these Series by column, and, change the
names of the columns to 'col1', 'col2', etc
df = pd.DataFrame({'col1': series1, 'col2': series2})
print(df)
'''
Output:
$ python3 Q11.py
   col1  col2
0     43   806
1     45   164
2     21   280
3     20   639
4     37   948
5     15   920
6     20   628
7     26   797
8     32   555
9     48   679
10    22   991
11    34   242
12    26   356
13    15   712
14    32   200
15    43   811
16    27   512
17    21   963
18    48   252
19    16   708
20    35   975
21    21   205
22    13   379
23    15   507
24    50   111
25    39   972
26    14   283
```

```
27    31   261
28    12   345
29    35   135
30    48   884
31    26   904
32    13   121
33    28   365
34    35   780
35    28   938
36    50   647
37    40   900
38    40   873
39    49   117
40    40   288
41    28   231
42    36   478
43    48   375
44    39   154
45    20   784
46    31   354
47    39   570
48    20   679
49    40   492
asheshjyoti@ubuntu:~/house/asheshjyoti/workspace/Learning/Python_and_machin
eLearning_guided/PW_DataSciencewithML/assignment44 <master>
'''
```

12. Perform the following operations using people data set:

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

b) Delete the rows containing any missing values.

d) Print the final output also.

->

```
import pandas as pd

# Load the dataset, ensuring the header is included even when skipping the
first 50 rows
data = pd.read_csv('People Data.csv', skiprows=range(1, 51))
# a) Delete the 'Email', 'Phone', and 'Date of birth' columns from the
```

```
dataset.
data = data.drop(columns=['Email', 'Phone', 'Date of birth'])

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

# c) Print the final output also
print(data)
'''
Output:
$ python3 Q12.py
    Index  ...  Salary
0      51  ...   80000
1      52  ...   70000
2      53  ...   60000
3      54  ...  100000
4      55  ...   50000
..    ...  ...     ...
945   996  ...   90000
946   997  ...   50000
947   998  ...   60000
948   999  ...  100000
949  1000  ...   90000

[950 rows x 7 columns]
asheshjyoti@ubuntu:~/house/asheshjyoti/workspace/Learning/Python_and_machin
eLearning_guided/PW_DataSciencewithML/assignment44 <master>
'''
```

13. Create two NumPy arrays, x and y, each containing 100 random float values between 0 and 1. Perform the following tasks using Matplotlib and NumPy:

a) Create a scatter plot using x and y, setting the color of the points to red and the marker style to 'o'.

b) Add a horizontal line at y = 0.5 using a dashed line style and label it as 'y = 0.5'.

c) Add a vertical line at x = 0.5 using a dotted line style and label it as 'x =

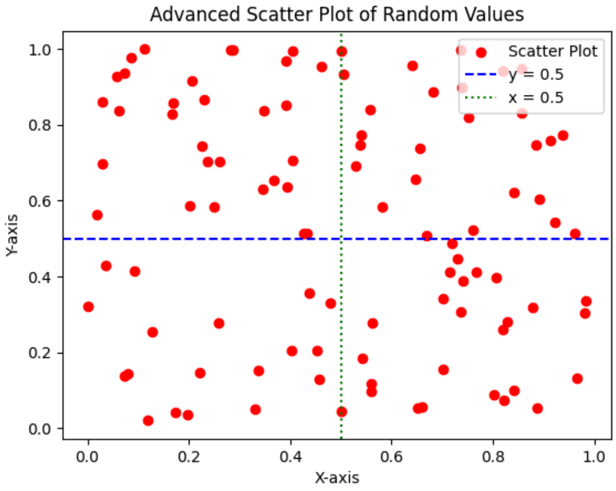
0.5'.

d) Label the x-axis as 'X-axis' and the y-axis as 'Y-axis'.

e) Set the title of the plot as 'Advanced Scatter Plot of Random Values'.

f) Display a legend for the scatter plot, the horizontal line, and the vertical line.
->

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

x = np.random.rand(100)
y = np.random.rand(100)

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



14. Create a time-series dataset in a Pandas DataFrame with columns:

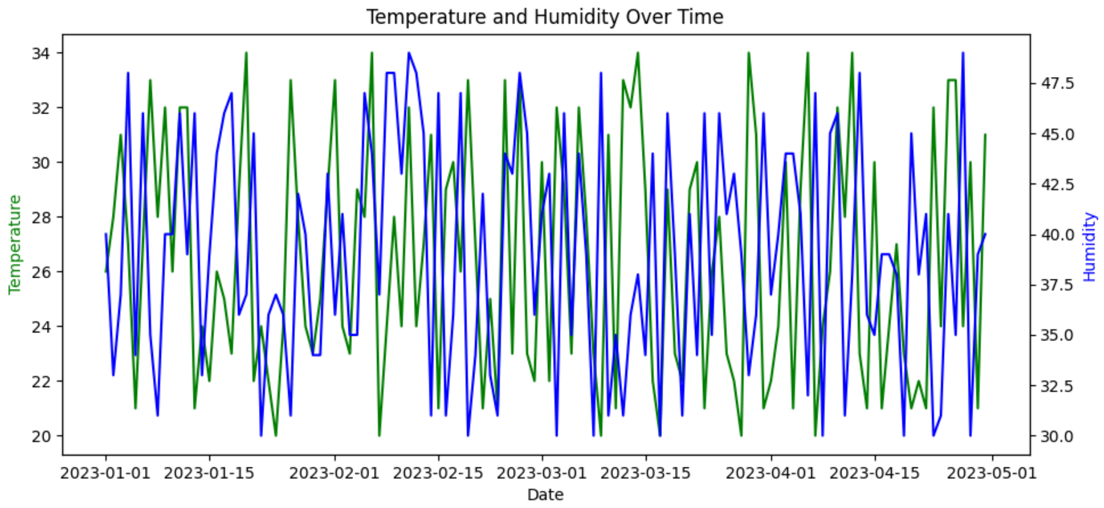
'Date', 'Temperature', 'Humidity' and perform the following tasks using Matplotlib:

a) Plot the 'Temperature' and 'Humidity' on the same plot with different y-axes (left y-axis for 'Temperature' and right y-axis for 'Humidity').

b) Label the x-axis as 'Date'.

c) Set the title of the plot as 'Temperature and Humidity Over Time'.
->

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
date_rng = pd.date_range(start='2023-01-01', end='2023-04-30', freq='D')
df = pd.DataFrame(date_rng, columns=['Date'])
df['Temperature'] = np.random.randint(20, 35, size=(len(date_rng)))
df['Humidity'] = np.random.randint(30, 50, size=(len(date_rng)))

# a) Plot the 'Temperature' and 'Humidity' on the same plot with different
y-axes (left y-axis for 'Temperature' and right y-axis for 'Humidity').
fig, ax1 = plt.subplots()

ax2 = ax1.twinx()
ax1.plot(df['Date'], df['Temperature'], 'g-')
ax2.plot(df['Date'], df['Humidity'], 'b-')

ax1.set_xlabel('Date')
ax1.set_ylabel('Temperature', color='g')
ax2.set_ylabel('Humidity', color='b')
plt.title('Temperature and Humidity Over Time')

plt.show()
```



15. Create a NumPy array data containing 1000 samples from a normal distribution. Perform the following tasks using Matplotlib:

a) Plot a histogram of the data with 30 bins.

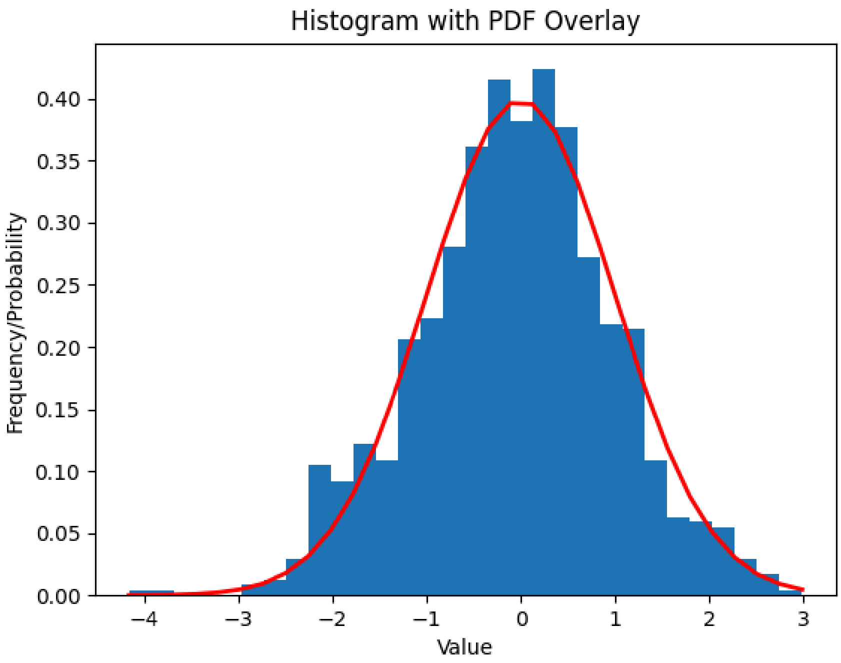
b) Overlay a line plot representing the normal distribution's probability density function (PDF).

c) Label the x-axis as 'Value' and the y-axis as 'Frequency/Probability'.

d) Set the title of the plot as 'Histogram with PDF Overlay'.

->

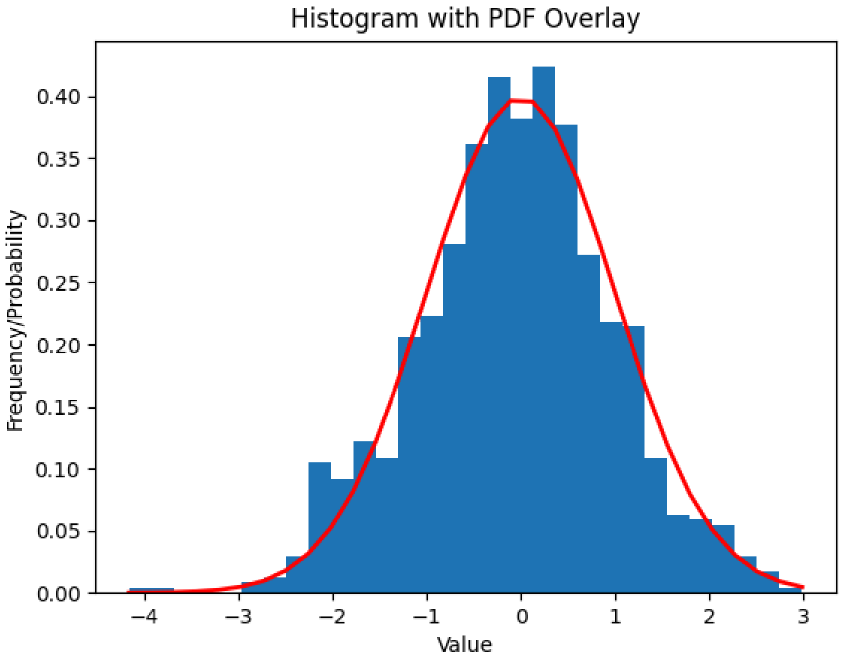
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
data = np.random.randn(1000)

# a) Plot a histogram of the data with 30 bins.
count, bins, ignored = plt.hist(data, 30, density=True)

# b) Overlay a line plot representing the normal distribution's probability
density function (PDF).
pdf = (1/(np.sqrt(2 * np.pi))) * np.exp(-(bins**2) / 2)
plt.plot(bins, pdf, linewidth=2, color='r')
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')
plt.title('Histogram with PDF Overlay')
plt.show()
```



16. Set the title of the plot as 'Histogram with PDF Overlay'.

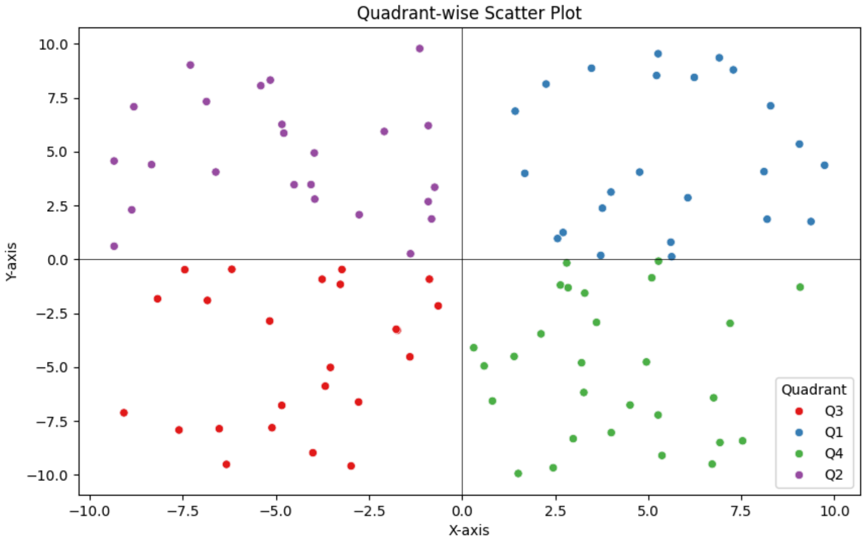
->

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
data = np.random.randn(1000)

# a) Plot a histogram of the data with 30 bins.
count, bins, ignored = plt.hist(data, 30, density=True)

# b) Overlay a line plot representing the normal distribution's probability density function (PDF).
pdf = (1/(np.sqrt(2 * np.pi))) * np.exp(-(bins**2) / 2)
plt.plot(bins, pdf, linewidth=2, color='r')
plt.xlabel('Value')
plt.ylabel('Frequency/Probability')
plt.title('Histogram with PDF Overlay')
plt.show()
```



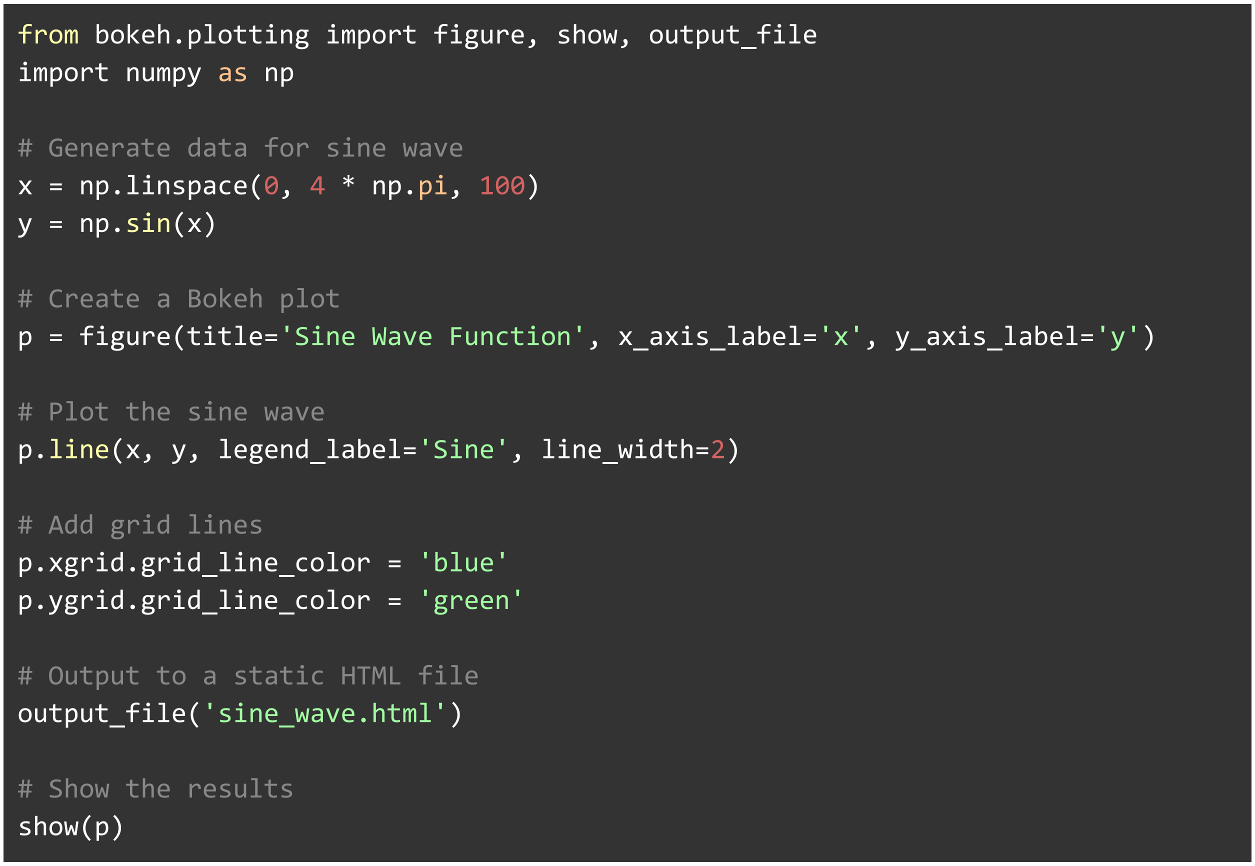
17)Create a Seaborn scatter plot of two random arrays, color points based on their position relative to the origin (quadrants), add a legend, label the axes, and set the title as 'Quadrant-wise Scatter Plot'.
->

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

# Generate two random arrays
x = np.random.uniform(-10, 10, 100)
y = np.random.uniform(-10, 10, 100)

# Determine the quadrant for each point
def get_quadrant(x, y):
   if x > 0 and y > 0:
       return 'Q1'
   elif x < 0 and y > 0:
       return 'Q2'
   elif x < 0 and y < 0:
       return 'Q3'
   elif x > 0 and y < 0:
       return 'Q4'

quadrants = [get_quadrant(xi, yi) for xi, yi in zip(x, y)]

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

# Create a Seaborn scatter plot
plt.figure(figsize=(10, 6))
scatter_plot = sns.scatterplot(data=df, x='x', y='y', hue='quadrant',
palette='Set1')

# Add labels, legend, and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Quadrant-wise Scatter Plot')
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.legend(title='Quadrant')
plt.show()
```

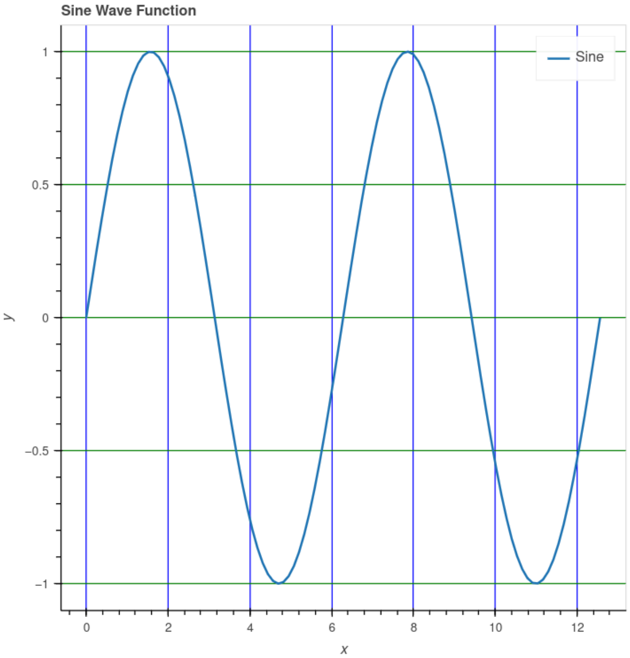


18. With Bokeh, plot a line chart of a sine wave function, add grid lines, label the axes, and set the title as 'Sine Wave Function.

->



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

# Generate data for sine wave
x = np.linspace(0, 4 * np.pi, 100)
y = np.sin(x)

# Create a Bokeh plot
p = figure(title='Sine Wave Function', x_axis_label='x', y_axis_label='y')

# Plot the sine wave
p.line(x, y, legend_label='Sine', line_width=2)

# Add grid lines
p.xgrid.grid_line_color = 'blue'
p.ygrid.grid_line_color = 'green'

# Output to a static HTML file
output_file('sine_wave.html')

# Show the results
show(p)
```

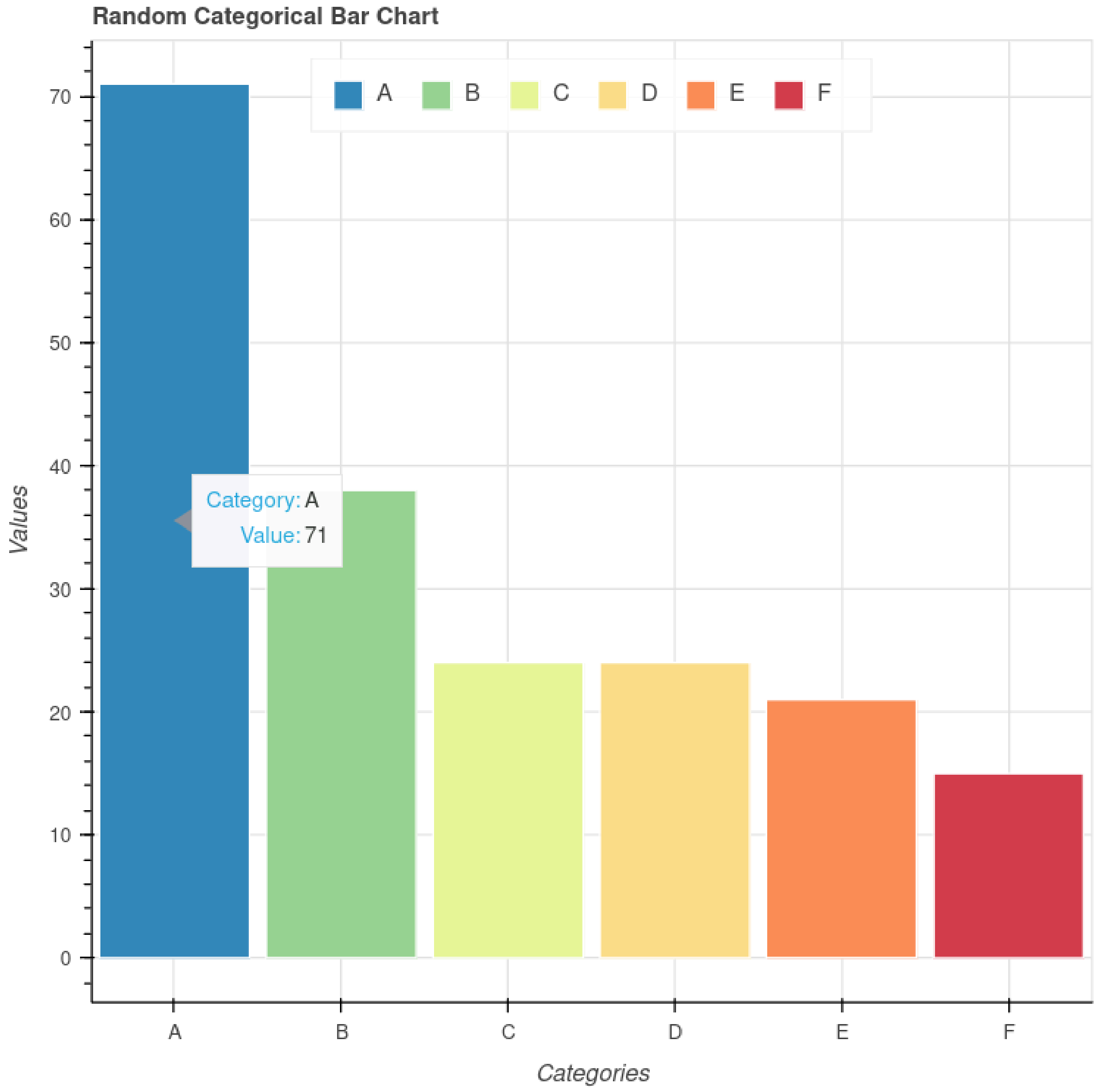


19. Using Bokeh, generate a bar chart of randomly generated categorical data, color bars based on their values, add hover tooltips to display exact values, label the axes, and set the title as 'Random Categorical Bar Chart'

->

```
from bokeh.plotting import figure, show, output_file
from bokeh.models import ColumnDataSource, HoverTool
from bokeh.transform import factor_cmap
from bokeh.palettes import Spectral6
import numpy as np
import pandas as pd

# Generate random categorical data
categories = ['A', 'B', 'C', 'D', 'E', 'F']
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 Bokeh figure
p = figure(x_range=categories, title="Random Categorical Bar Chart",
          toolbar_location=None, tools="")

# Add a bar chart
p.vbar(x='categories', top='values', width=0.9, source=source,
      legend_field='categories', line_color='white',
      fill_color=factor_cmap('categories', palette=Spectral6,
factors=categories))

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

# Label the axes
p.xaxis.axis_label = "Categories"
p.yaxis.axis_label = "Values"

# Customize the legend
```

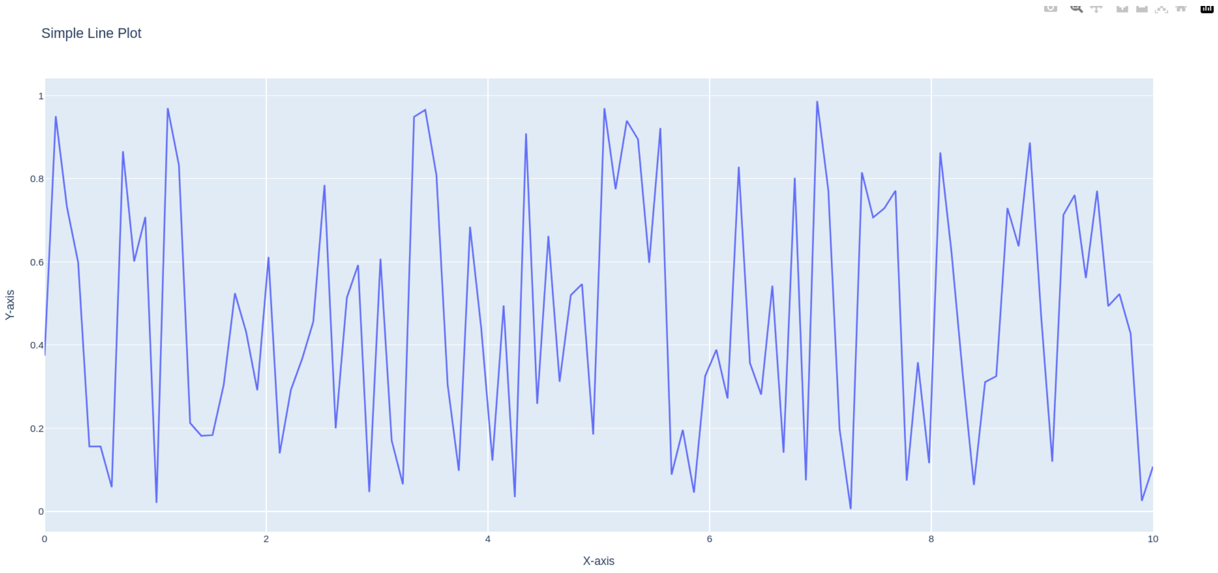
```
p.legend.orientation = "horizontal"
p.legend.location = "top_center"

# Output to a static HTML file
output_file('random_categorical_bar_chart.html')

# Show the results
show(p)
```



20. Using Plotly, create a basic line plot of a randomly generated dataset, label the axes, and set the title as 'Simple Line Plot'.

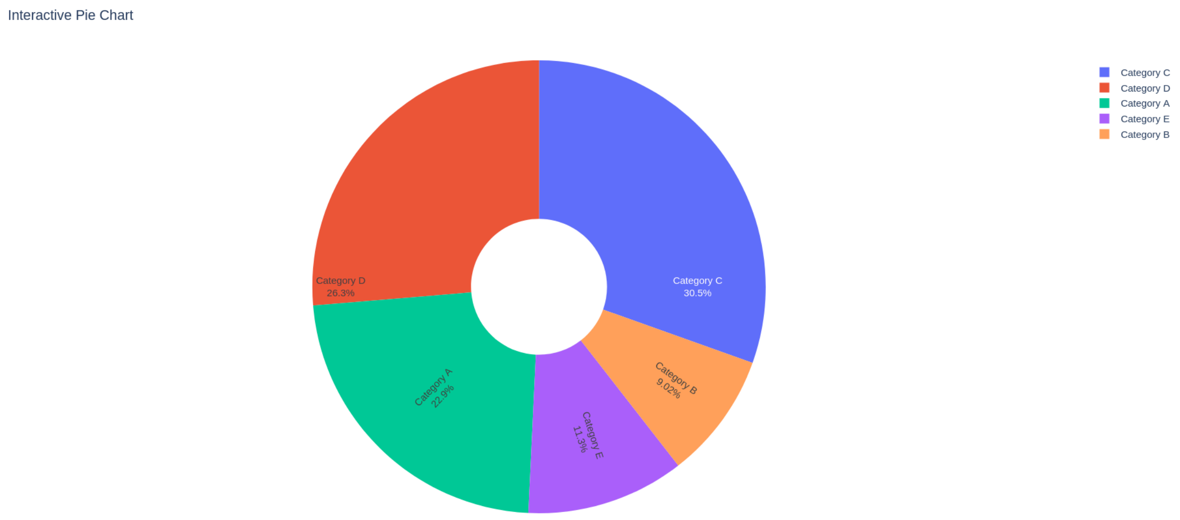
->

```
import plotly.graph_objects as go
import numpy as np

# Generate a random dataset
np.random.seed(42)
x = np.linspace(0, 10, 100)
y = np.random.rand(100)

# Create the Plotly line plot
fig = go.Figure()

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

# Add labels to the axes and set the title
fig.update_layout(
   title='Simple Line Plot',
   xaxis_title='X-axis',
   yaxis_title='Y-axis'
)

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



21. Using Plotly, create an interactive pie chart of randomly generated data, add labels and percentages, set the title as 'Interactive Pie Chart'.
->

```
import plotly.graph_objects as go
import numpy as np

# Generate random data for the pie chart
np.random.seed(42)
categories = ['Category A', 'Category B', 'Category C', 'Category D', 'Category
E']
values = np.random.randint(10, 100, size=len(categories))

# Create the Plotly pie chart
fig = go.Figure(data=[go.Pie(labels=categories, values=values, hole=0.3,
textinfo='label+percent', insidetextorientation='radial')])

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

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


Interactive Pie Chart
Category C
Category B
Category E
Category A
Category D
🟦 Category C
🟥 Category D
🟩 Category A
🟪 Category E
🟧 Category B