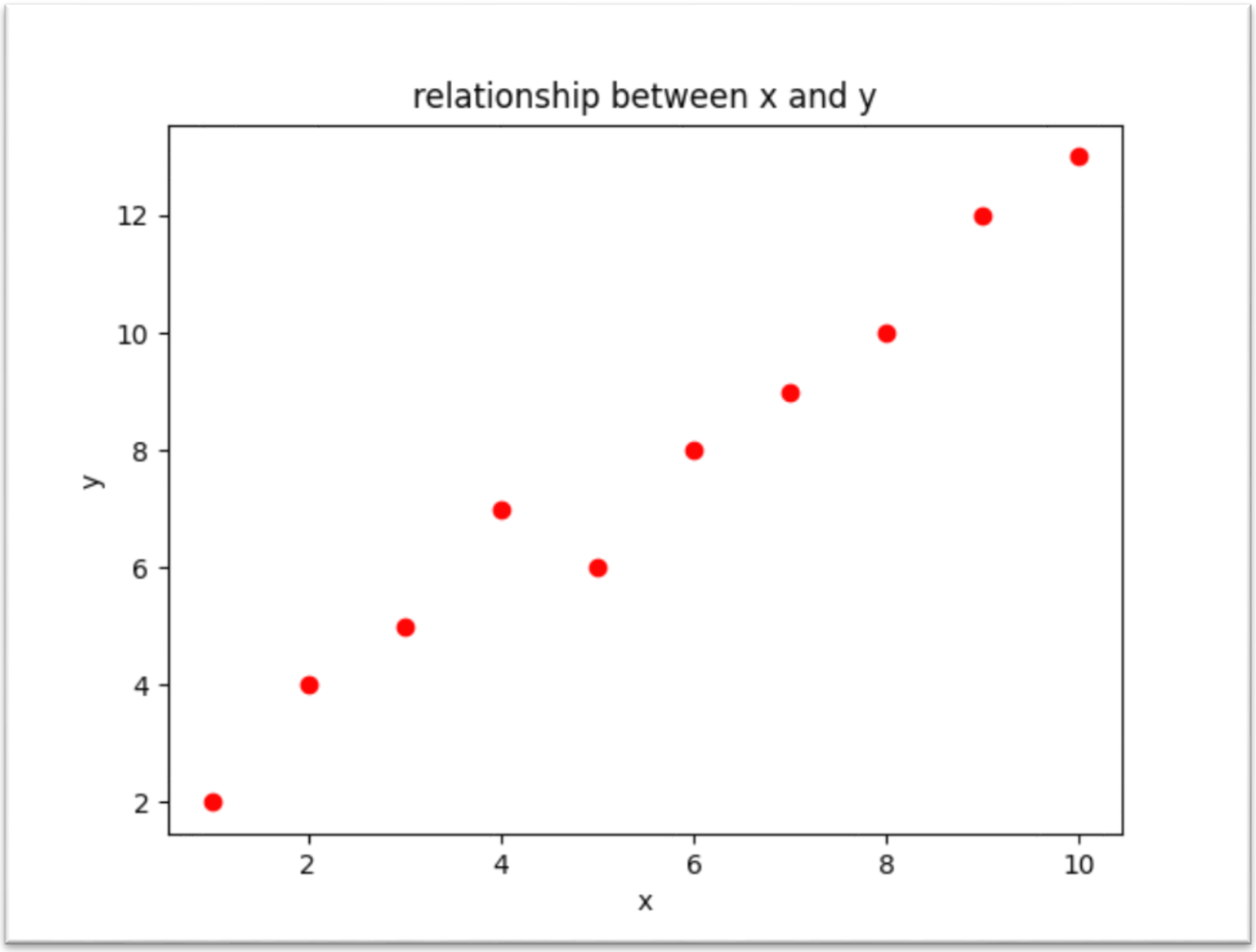
MATPLOTLIB ASSIGNMENT:

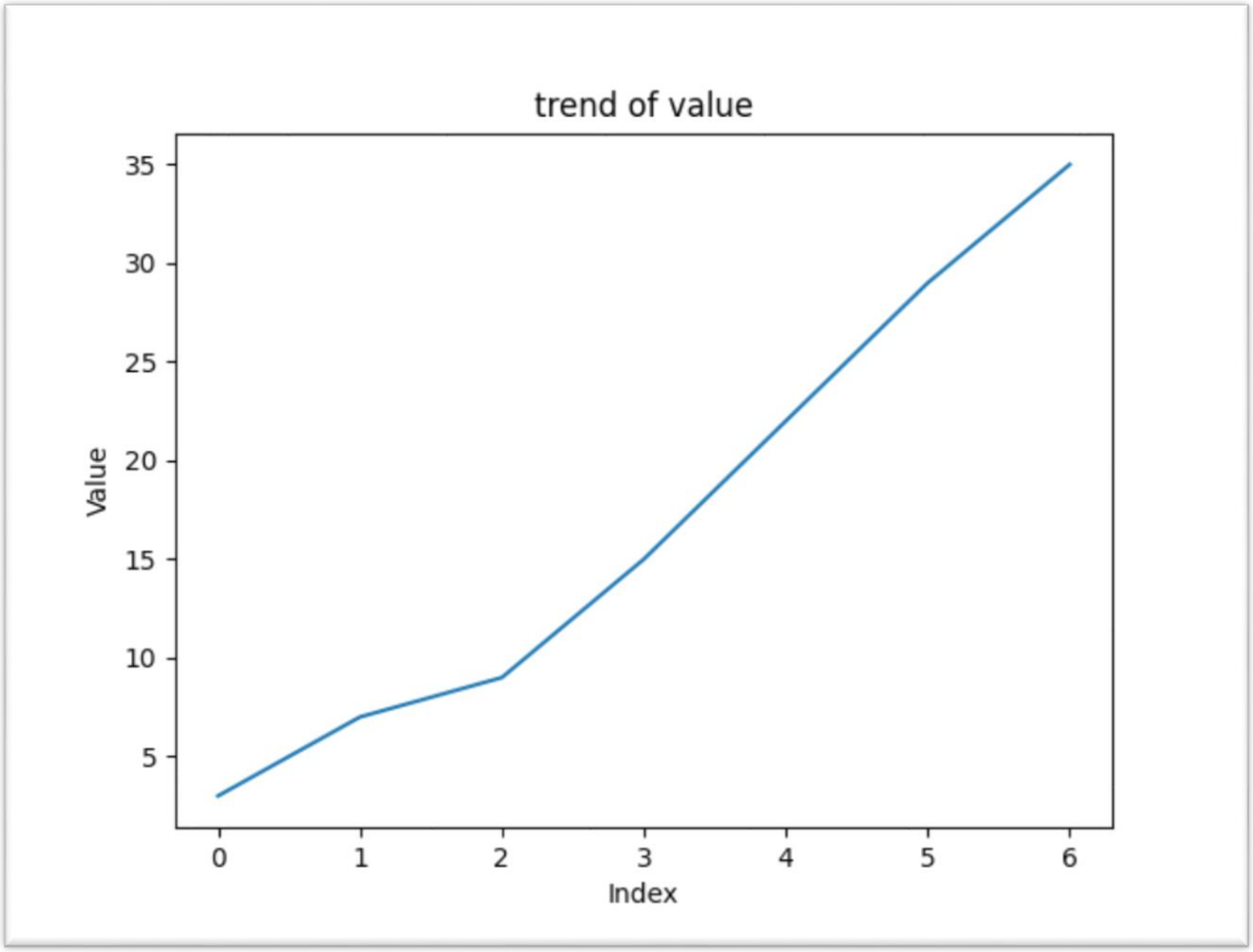
1. Create a scatter plot using Matplotlib to visualize the relationship between two arrays, x and y for the given data. x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] y = [2, 4, 5, 7, 6, 8, 9, 10, 12, 13].

```python

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.scatter(x,y,color ='r')
plt.title('relationship between x and y')
plt.xlabel('x')
plt.ylabel('y')

plt.show()
```



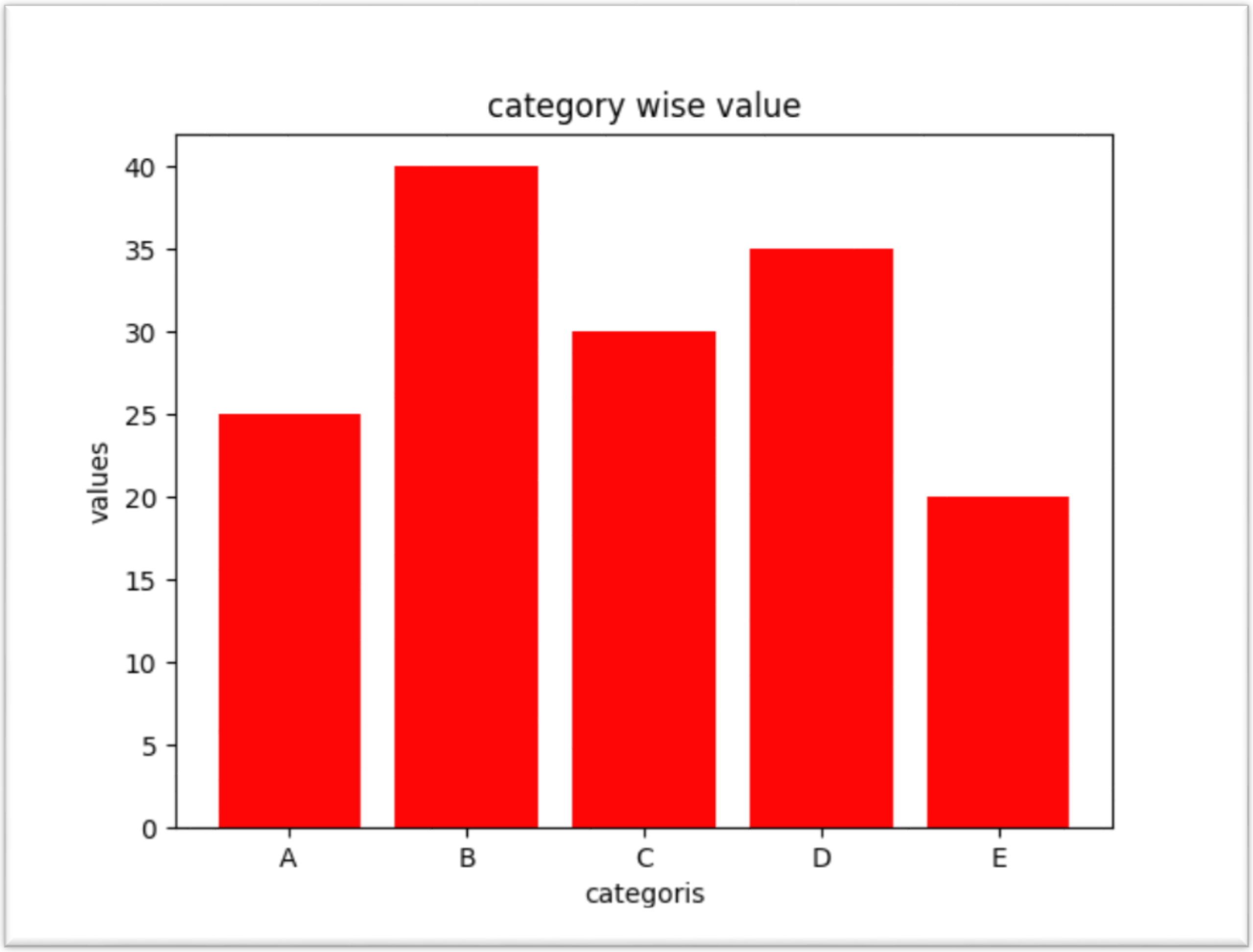
2. Generate a line plot to visualize the trend of values for the given data.
data = np.array([3, 7, 9, 15, 22, 29, 35]).

```
import matplotlib.pyplot as plt
plt.plot(data)
plt.title('trend of value')
plt.xlabel('Index')
plt.ylabel('Value')
plt.show()
```



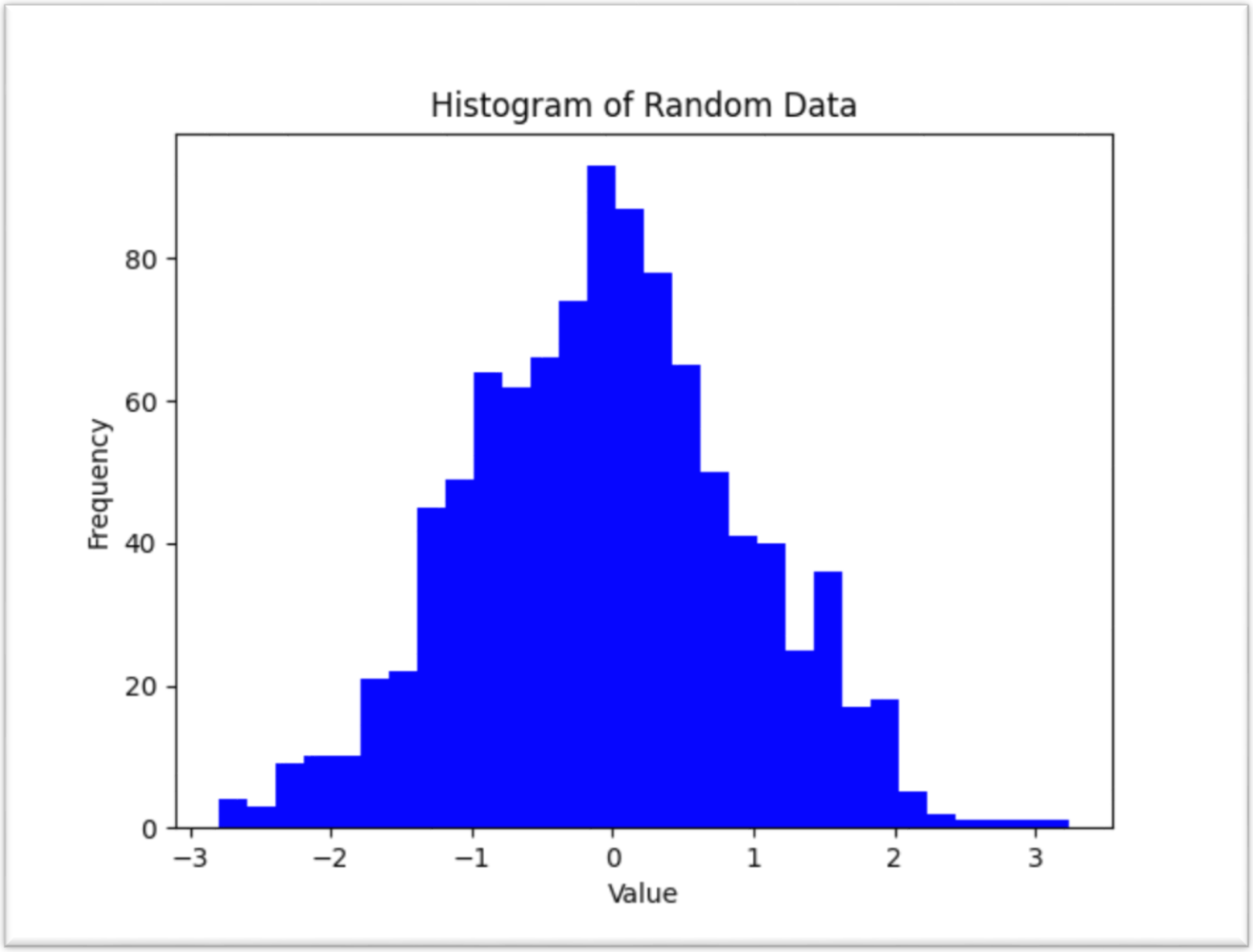
3. Display a bar chart to represent the frequency of each item in the given array categories. categories = ['A', 'B', 'C', 'D', 'E'] values = [25, 40, 30, 35, 20].

```
import matplotlib.pyplot as plt
plt.bar(categories,values,color='r',align='center')
plt.title('category wise value')
plt.xlabel('categoris')
plt.ylabel('values')
plt.show()
```



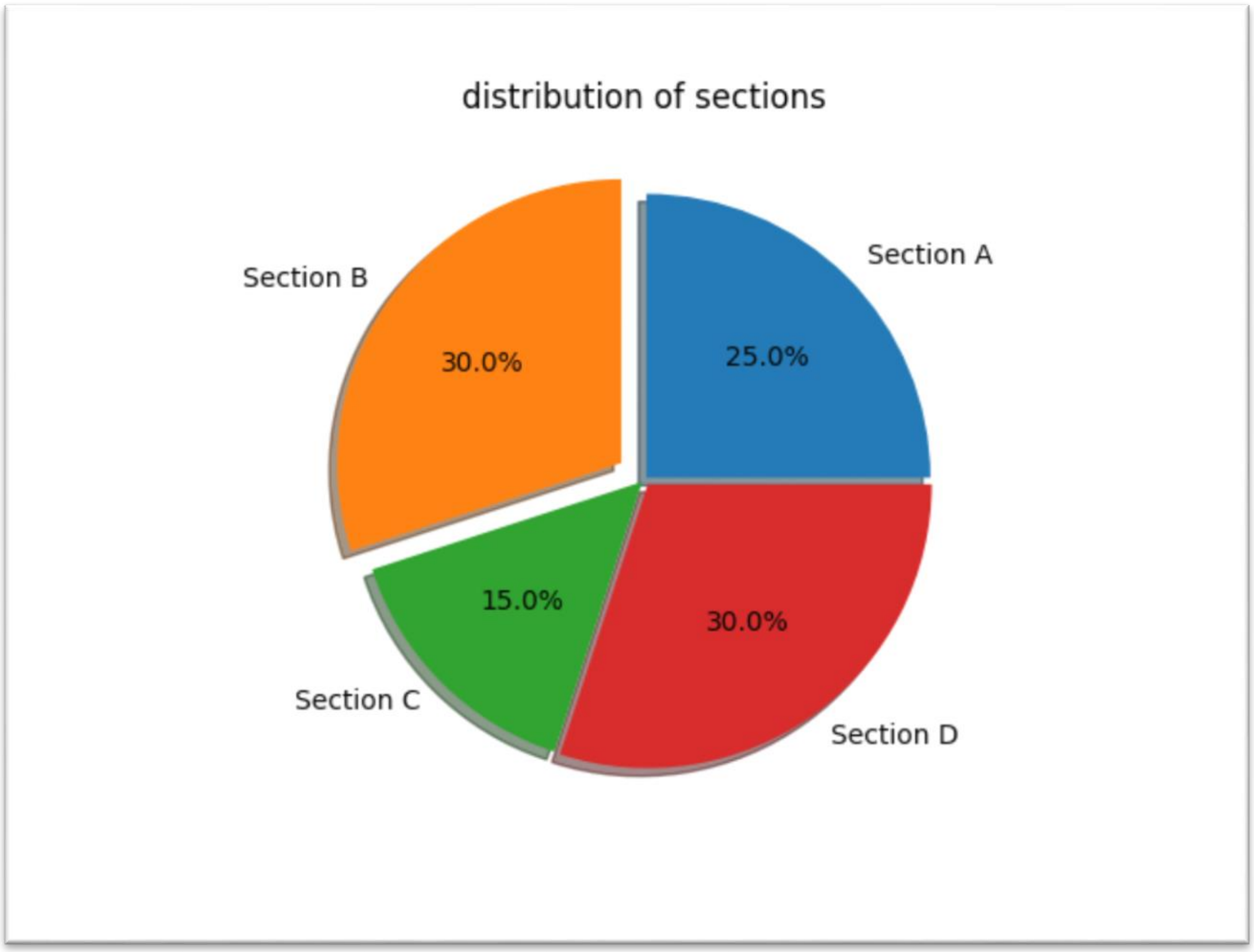
4. Create a histogram to visualize the distribution of values in the array data. data = np.random.normal(0, 1, 1000).

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

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

# Create the histogram
plt.hist(data, bins=30, color='blue')

# Add labels and a title
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram of Random Data')

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



5. Show a pie chart to represent the percentage distribution of different sections in the array `sections`. sections = ['Section A', 'Section B', 'Section C', 'Section D'] sizes = [25, 30, 15, 30].

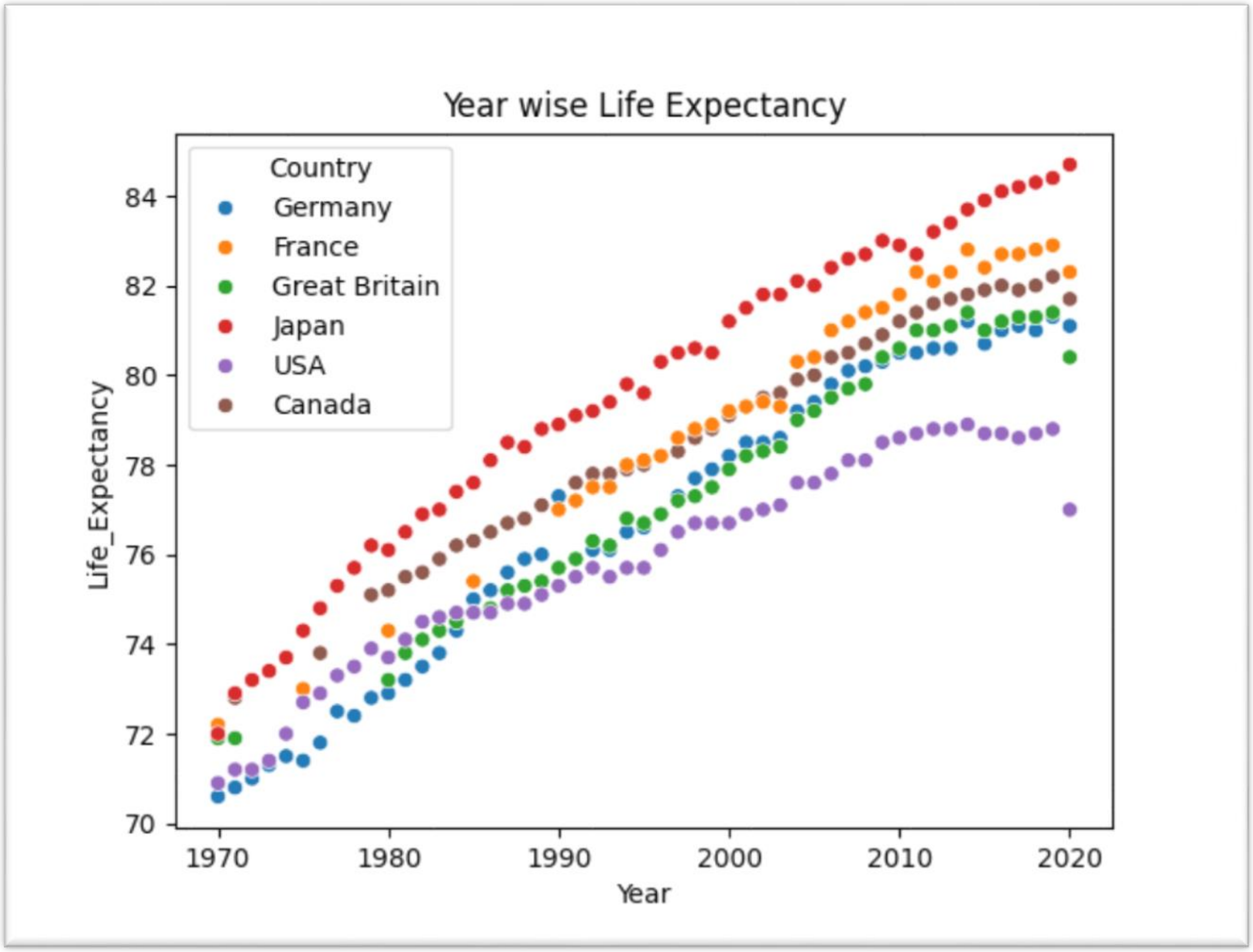
```
explode =[0.01,0.1,0.01,0.02]
plt.pie(sizes,labels=sections,autopct= '%1.1f%%',shadow=True,explode=explode,)
plt.show()
```


distribution of sections
Section A
25.0%
Section B
30.0%
Section C
15.0%
Section D
30.0%

SEABORN ASSIGNMENT:

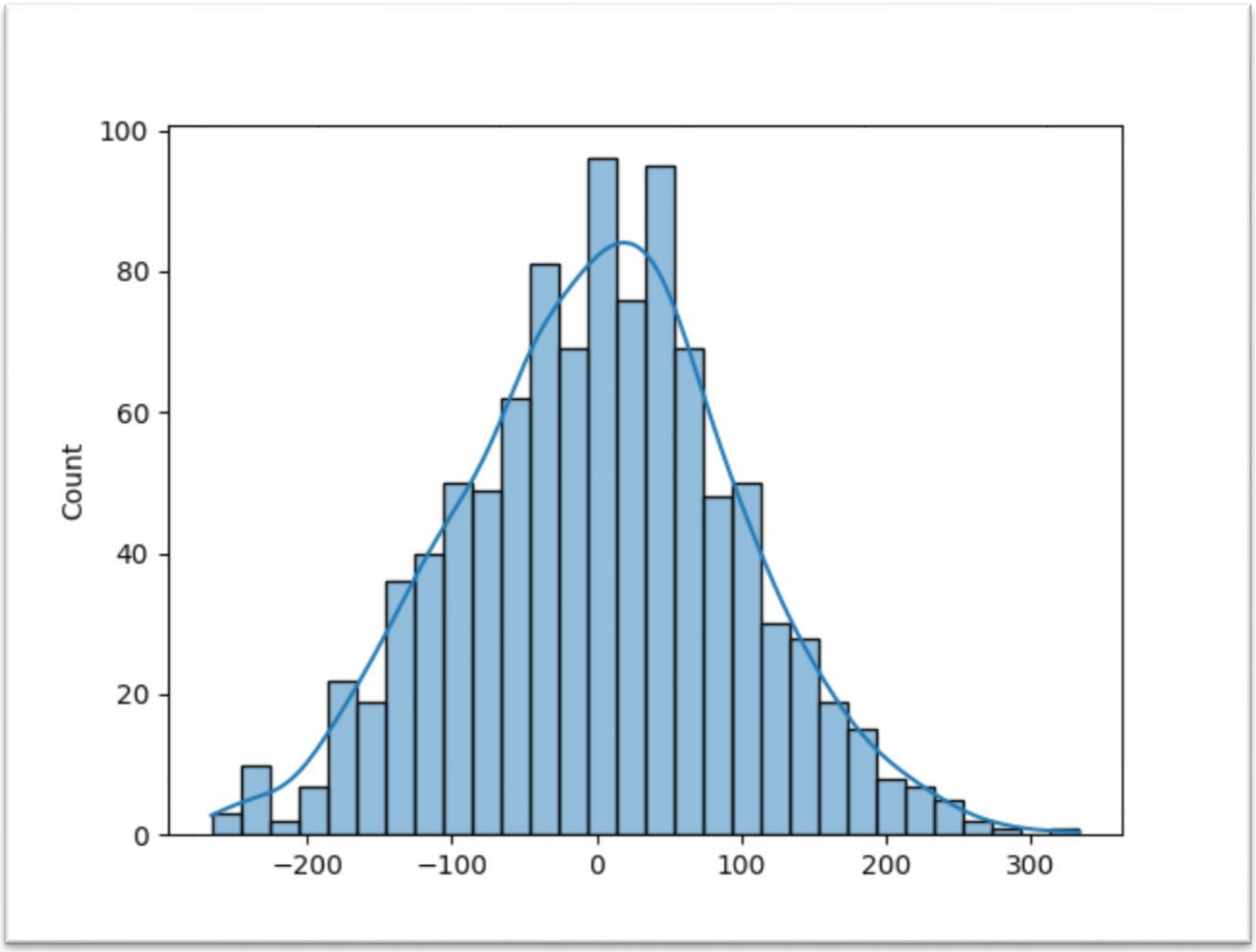
1. Create a scatter plot to visualize the relationship between two variables, by generating a synthetic dataset.

```
import seaborn as sns
sns.get_dataset_names()
health = sns.load_dataset('healthexp')
health.head()
sns.scatterplot(x=health['Year'],
y=health['Life_Expectancy'],hue=health['Country'])
plt.title('Year wise Life Expectancy')
plt.show()
```



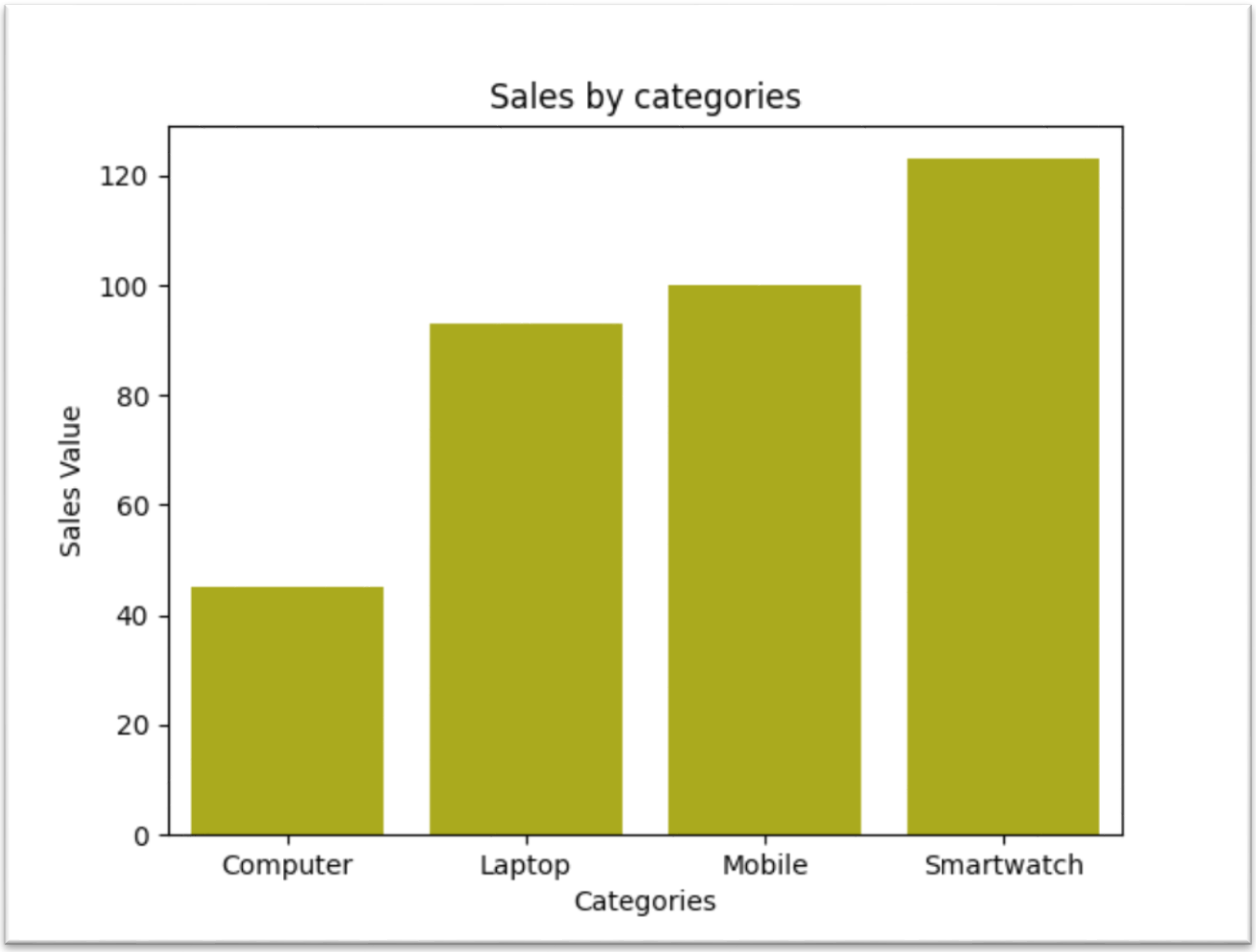
2. Generate a dataset of random numbers. Visualize the distribution of a numerical variable.

```python
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
data = np.random.normal(1,100,1000)

sns.histplot(data, kde=True, bins=30)
plt.show()
```



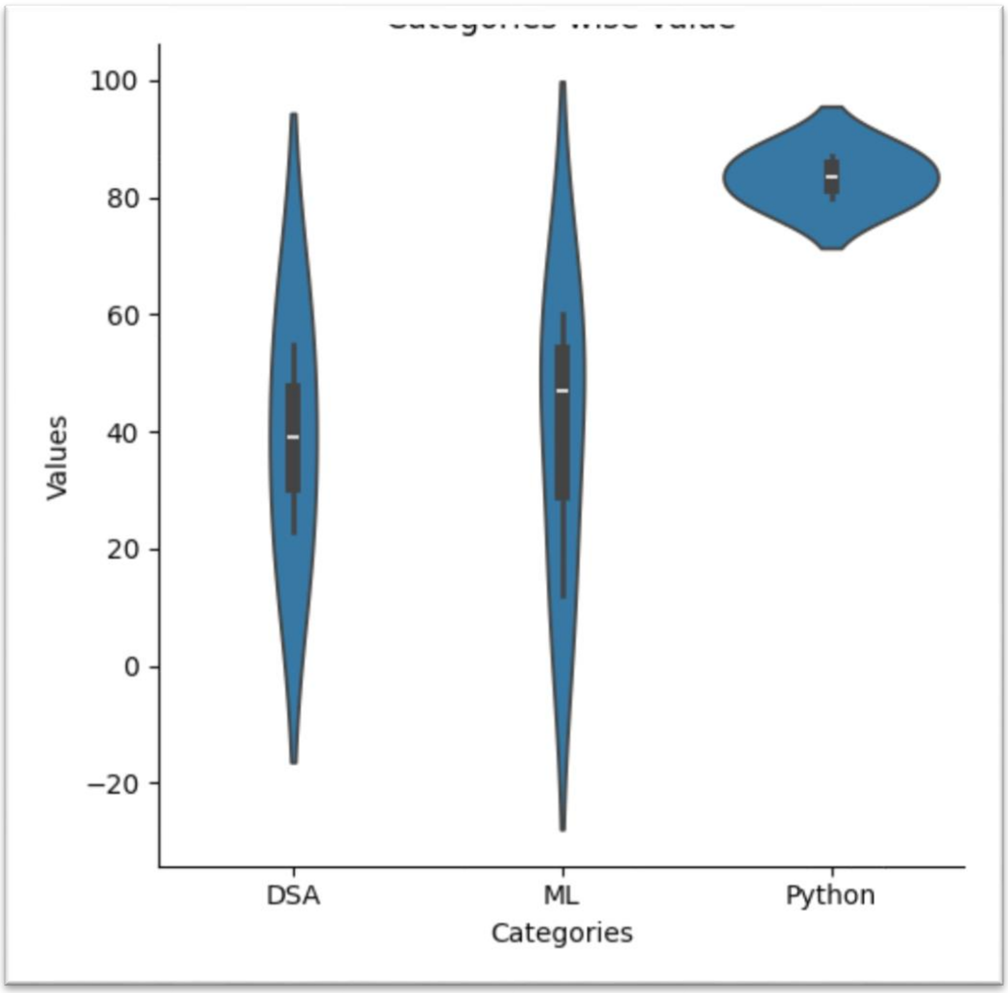
3. Create a dataset representing categories and their corresponding values. Compare different categories based on numerical values.

```
import pandas as pd
data ={
'categories' : ['Computer','Laptop','Mobile','Smartwatch'],
'Sales' : [45,93,100,123] }
data = pd.DataFrame(data)
data
sns.barplot(x=data['categories'],y=data['Sales'],color='y')
plt.title('Sales by categories')
plt.xlabel('Categories')
plt.ylabel('Sales Value')
plt.show()
```



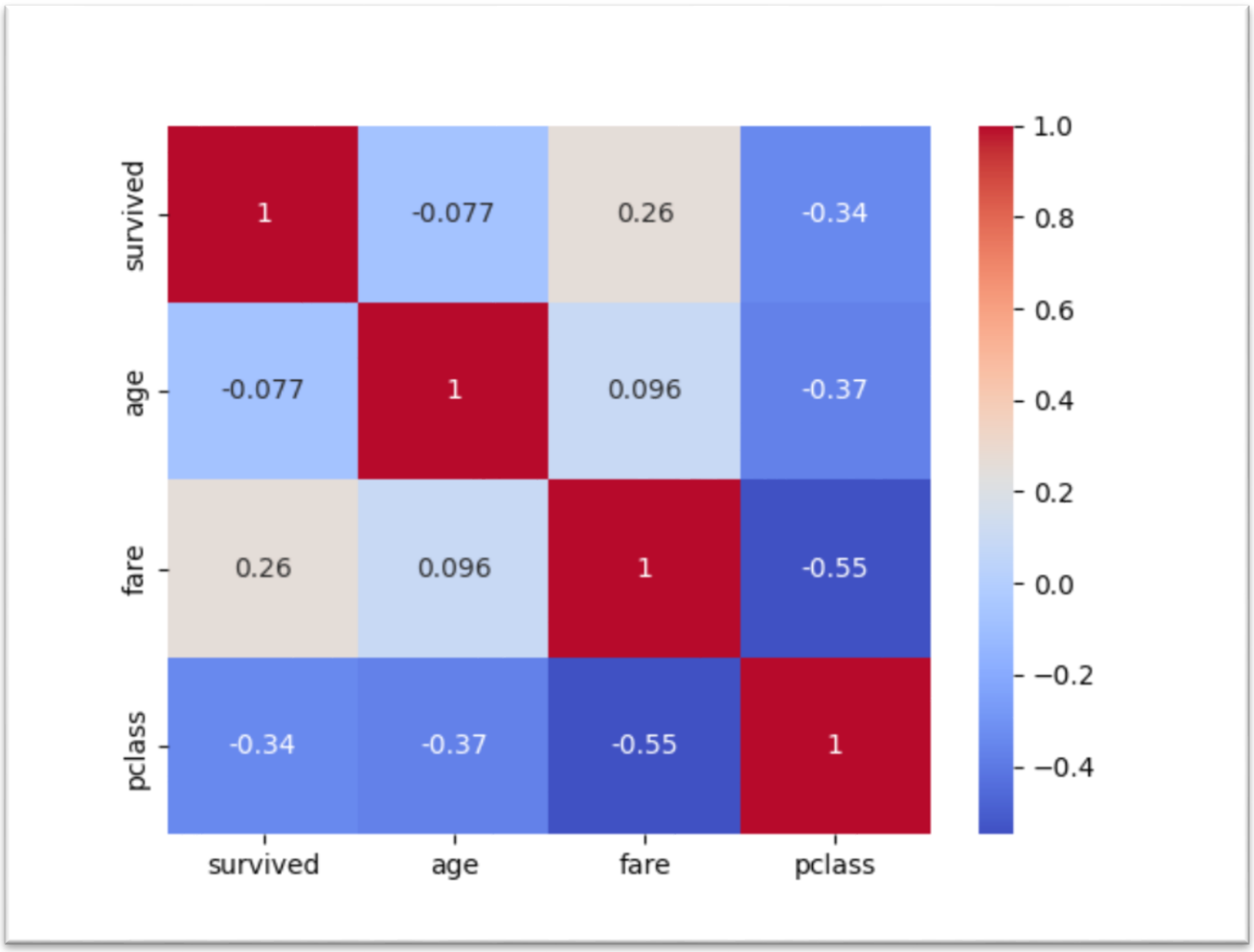
4.Generate a dataset with categories and numerical values. Visualize the distribution of a numerical variable across different categories.

```python
import numpy as np
import pandas as pd
import seaborn as sns

data = {
    'Categories' : ['DSA','ML','Python','ML','DSA','ML','Python'],
    'Values' : np.random.randint(1,100,7)
}
df = pd.DataFrame(data)
sns.catplot(x=df['Categories'],y=df['Values'],kind='violin')
plt.title('Categories wise value')
plt.show()
```



5. Generate a synthetic dataset with correlated features. Visualize the correlation matrix of a dataset using a heatmap.

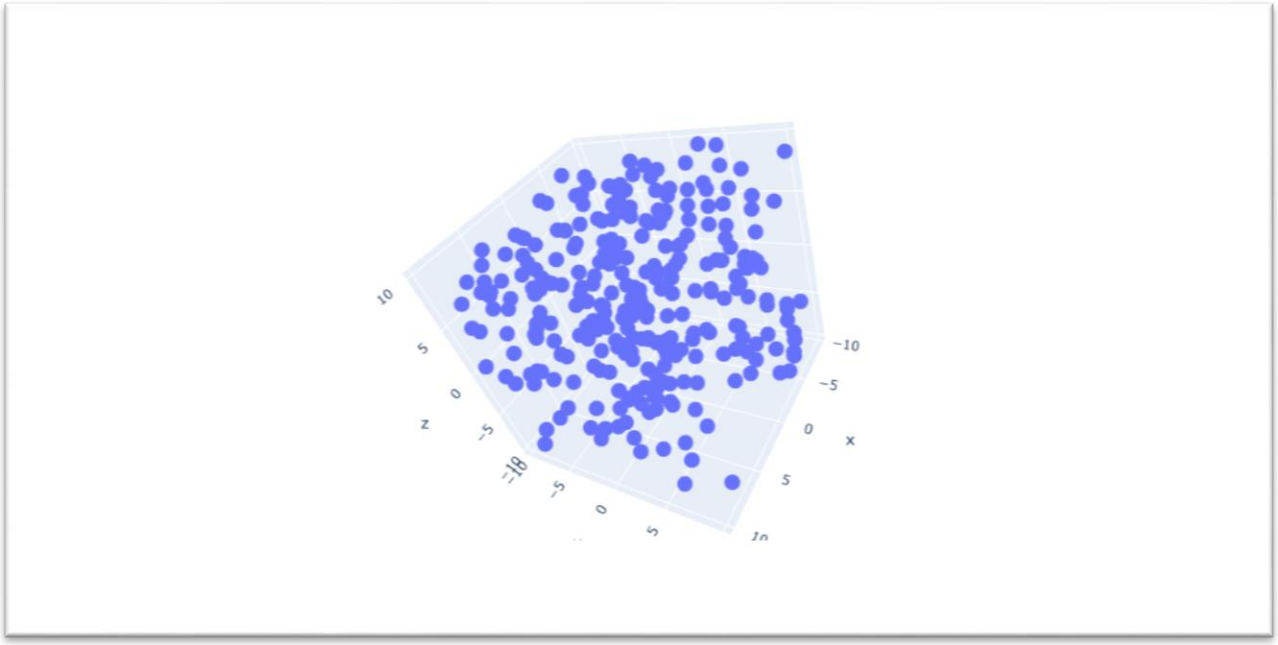
```
import seaborn as sns
sns.get_dataset_names()
 df = sns.load_dataset('titanic')
df.head()
df =df[['survived','age','fare','pclass']]
df.corr()
sns.heatmap(df.corr(),cmap='coolwarm',annot=True)
plt.show()
```



PLOTLY ASSIGNMENT:

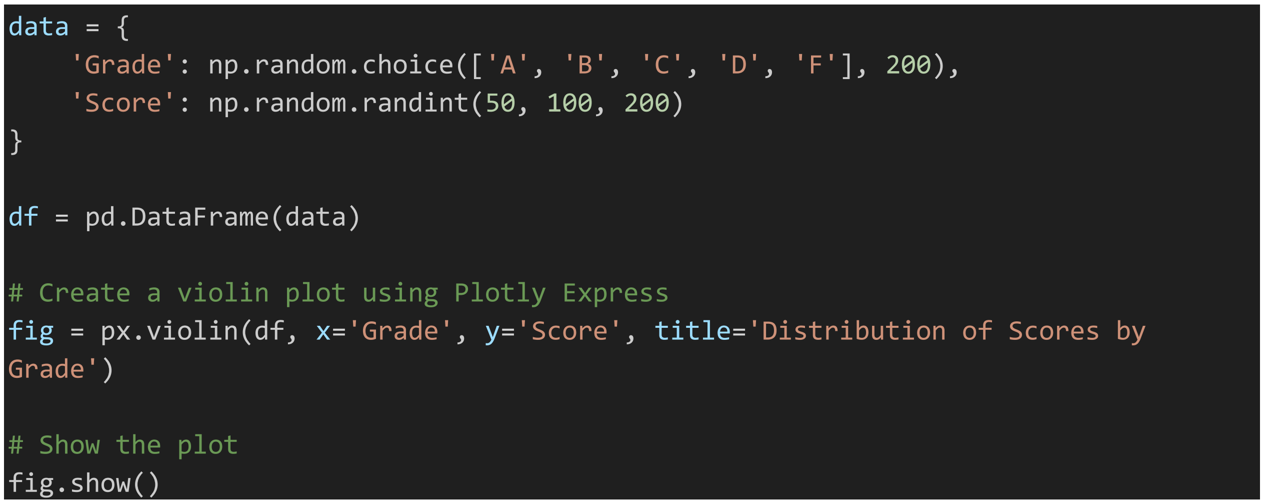
1. Using the given dataset, to generate a 3D scatter plot to visualize the distribution of data points in a three dimensional space. np.random.seed(30)data = { 'X': np.random.uniform(-10, 10, 300), 'Y': np.random.uniform(-10, 10, 300),'Z': np.random.uniform(-10, 10, 300)} df = pd.DataFrame(data).

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

fig = go.Figure()
fig =px.scatter_3d(x=df['X'],y=df['Y'],z=df['Z'])
fig.show()
```

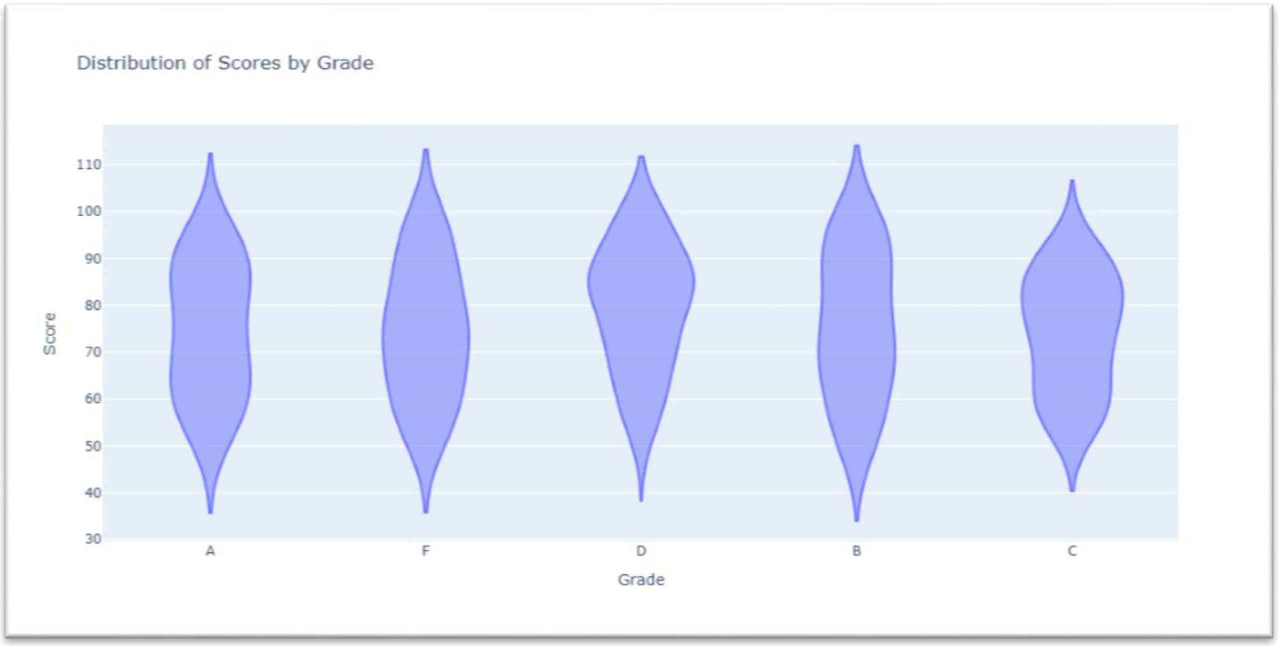


2. Using the Student Grades, create a violin plot to display the distribution of scores across different grade categories. np.random.seed(15) data = { 'Grade': np.random.choice(['A', 'B', 'C', 'D', 'F'], 200), 'Score': np.random.randint(50, 100, 200) } df = pd.DataFrame(data).

```

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

np.random.seed(15)

# Create a DataFrame with random grades and scores
```

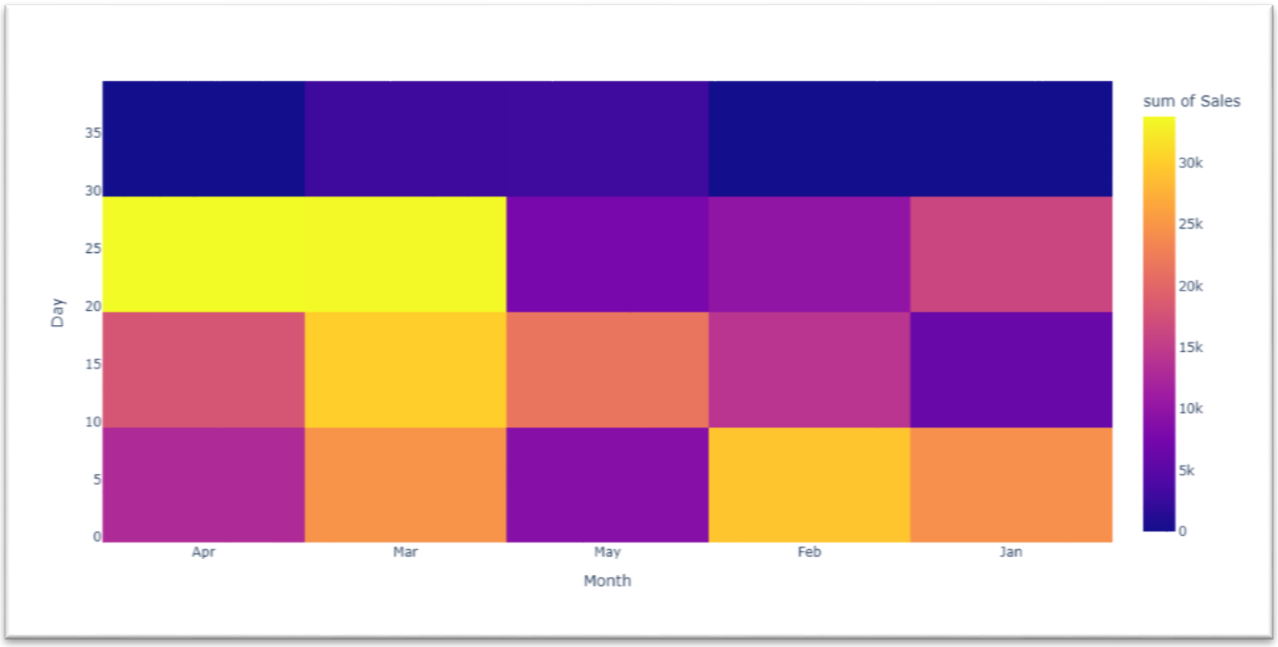


```
data = {
    'Grade': np.random.choice(['A', 'B', 'C', 'D', 'F'], 200),
    'Score': np.random.randint(50, 100, 200)
}

df = pd.DataFrame(data)

# Create a violin plot using Plotly Express
fig = px.violin(df, x='Grade', y='Score', title='Distribution of Scores by
Grade')

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



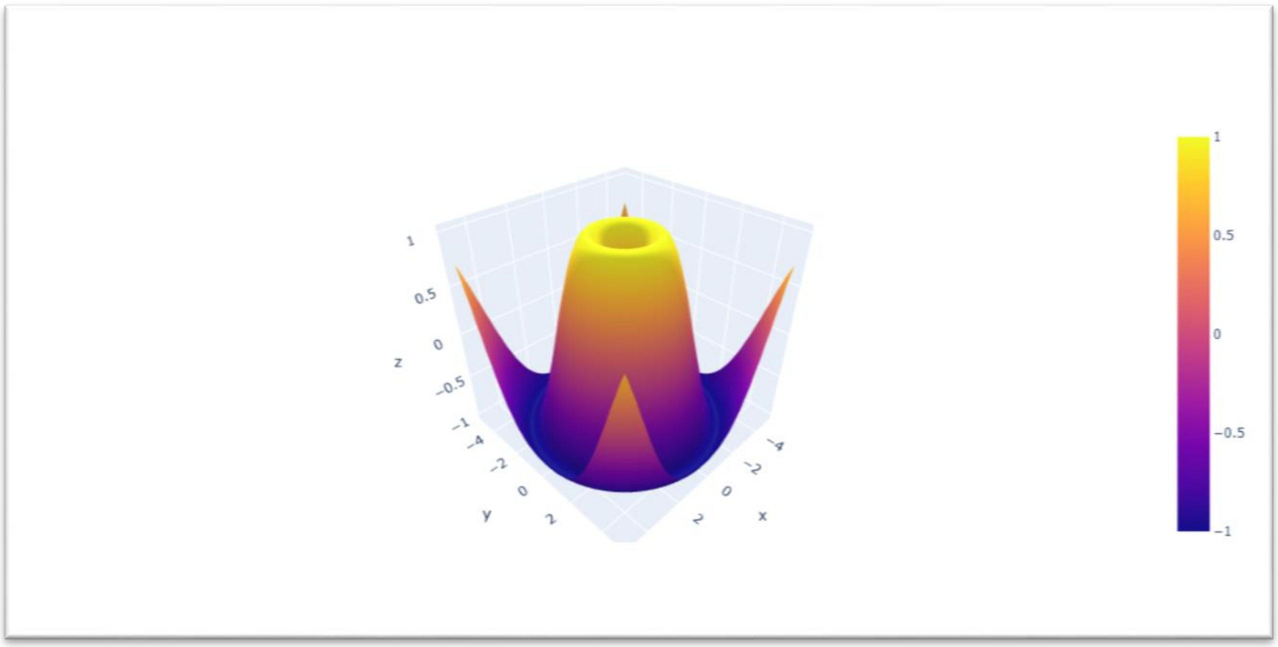
3. Using the sales data, generate a heatmap to visualize the variation in sales across different months and days. np.random.seed(20) data = { 'Month': np.random.choice(['Jan', 'Feb', 'Mar', 'Apr', 'May'], 100), 'Day': np.random.choice(range(1, 31), 100), 'Sales': np.random.randint(1000, 5000, 100) } df = pd.DataFrame(data).

```
 np.random.seed(20)
 data = {
 'Month': np.random.choice(['Jan', 'Feb', 'Mar', 'Apr', 'May'], 100),
 'Day': np.random.choice(range(1, 31), 100),
 'Sales': np.random.randint(1000, 5000, 100)
 }
df = pd.DataFrame(data)
import plotly.graph_objects as go
import plotly.express as px
import pandas as pd
fig = px.density_heatmap(df,x='Month',y='Day',z='Sales',title='Heat map char of
data',)
fig.show()
```



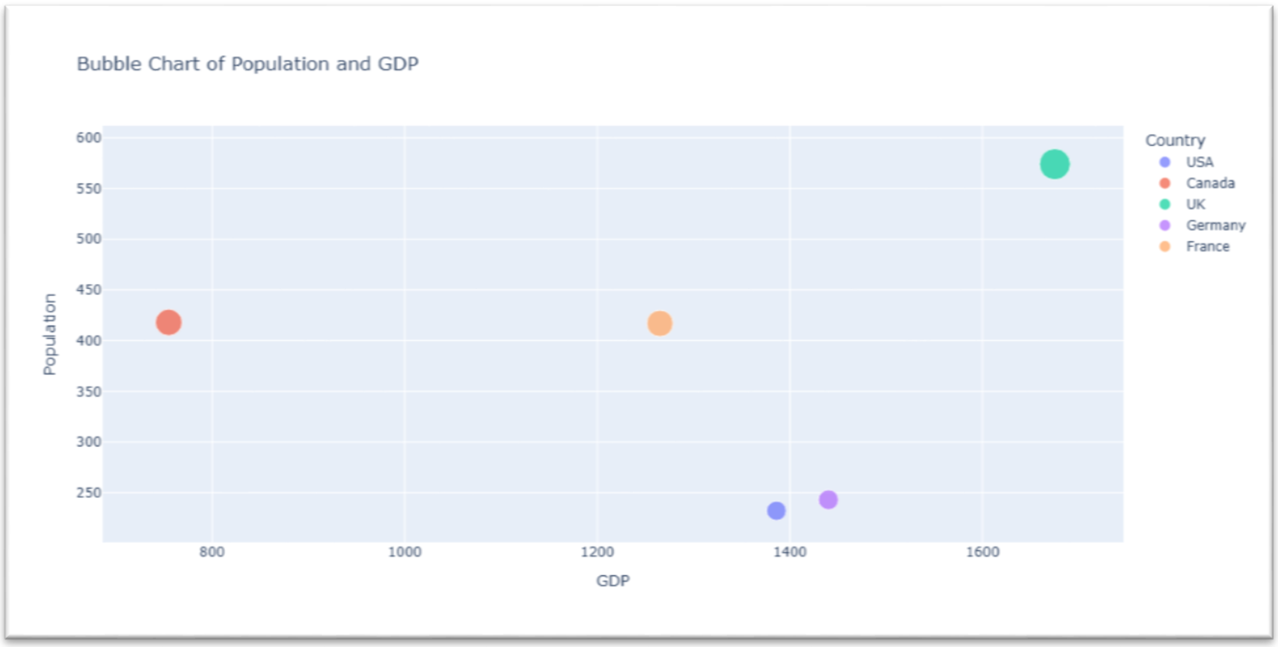
4. Using the given x and y data, generate a 3D surface plot to visualize the function x = np.linspace(-5, 5, 100) y = np.linspace(-5, 5, 100) x, y = np.meshgrid(x, y) z = np.sin(np.sqrt(x**2 + y**2)) data = { 'X': x.flatten(), 'Y': y.flatten(), 'Z': z.flatten() } df = pd.DataFrame(data).

```
df = pd.DataFrame(data)
import numpy as np
import plotly.graph_objects as go

# Generate the data
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = np.sin(np.sqrt(x**2 + y**2))

# Create a 3D surface plot
fig = go.Figure(data=[go.Surface(x=x, y=y, z=z)])

# Set the title and axis labels
fig.update_layout(title='3D Surface Plot', xaxis_title='X', yaxis_title='Y',
zaxis_title='Z')

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



4. Using the given dataset, create a bubble chart to represent each country's population (y-axis), GDP (x axis), and bubble size proportional to the population. np.random.seed(25) data = { 'Country': ['USA', 'Canada', 'UK', 'Germany', 'France'], 'Population': np.random.randint(100, 1000, 5), 'GDP': np.random.randint(500, 2000, 5) } df = pd.DataFrame(data).

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

np.random.seed(25)

# Create a DataFrame with random data
data = {
    'Country': ['USA', 'Canada', 'UK', 'Germany', 'France'],
    'Population': np.random.randint(100, 1000, 5),
    'GDP': np.random.randint(500, 2000, 5)
}

df = pd.DataFrame(data)

# Create a bubble chart using Plotly
fig = px.scatter(df, x='GDP', y='Population', size='Population',
color='Country',
                 title='Bubble Chart of Population and GDP')

# Show the bubble chart
fig.show()
```

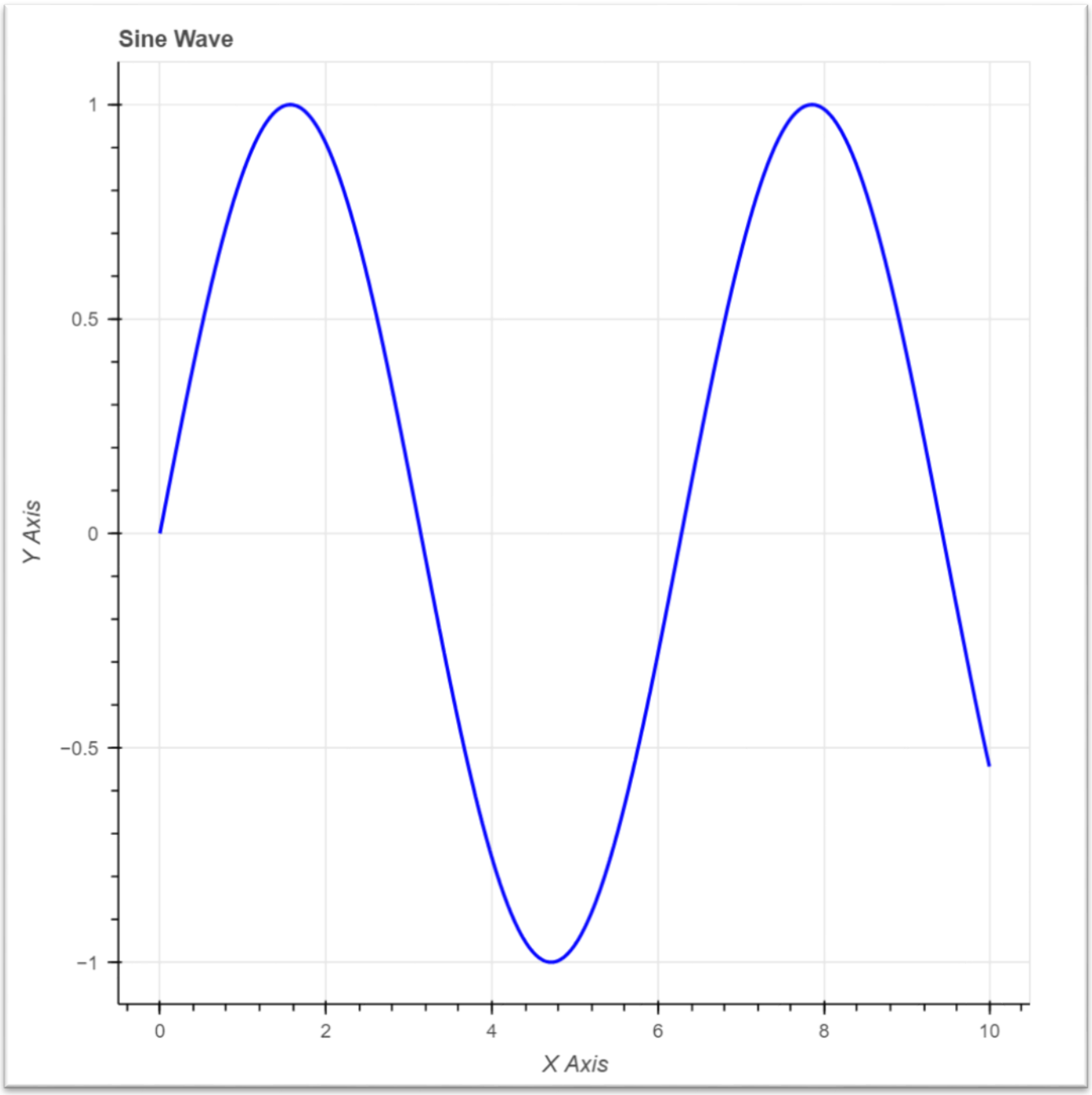


BOKEH ASSIGNMENT:

1.Create a Bokeh plot displaying a sine wave. Set x-values from 0 to 10 and y-values as the sine of x.

```
# Import necessary modules

import numpy as np

import bokeh.plotting as bp

from bokeh.io import output_file, show


# Generate x and y values for the sine wave

x = np.linspace(0, 10, 1000)  # 1000 points between 0 and 10

y = np.sin(x)


# Specify the output file to save the plot

output_file("sine_wave_plot.html")
```

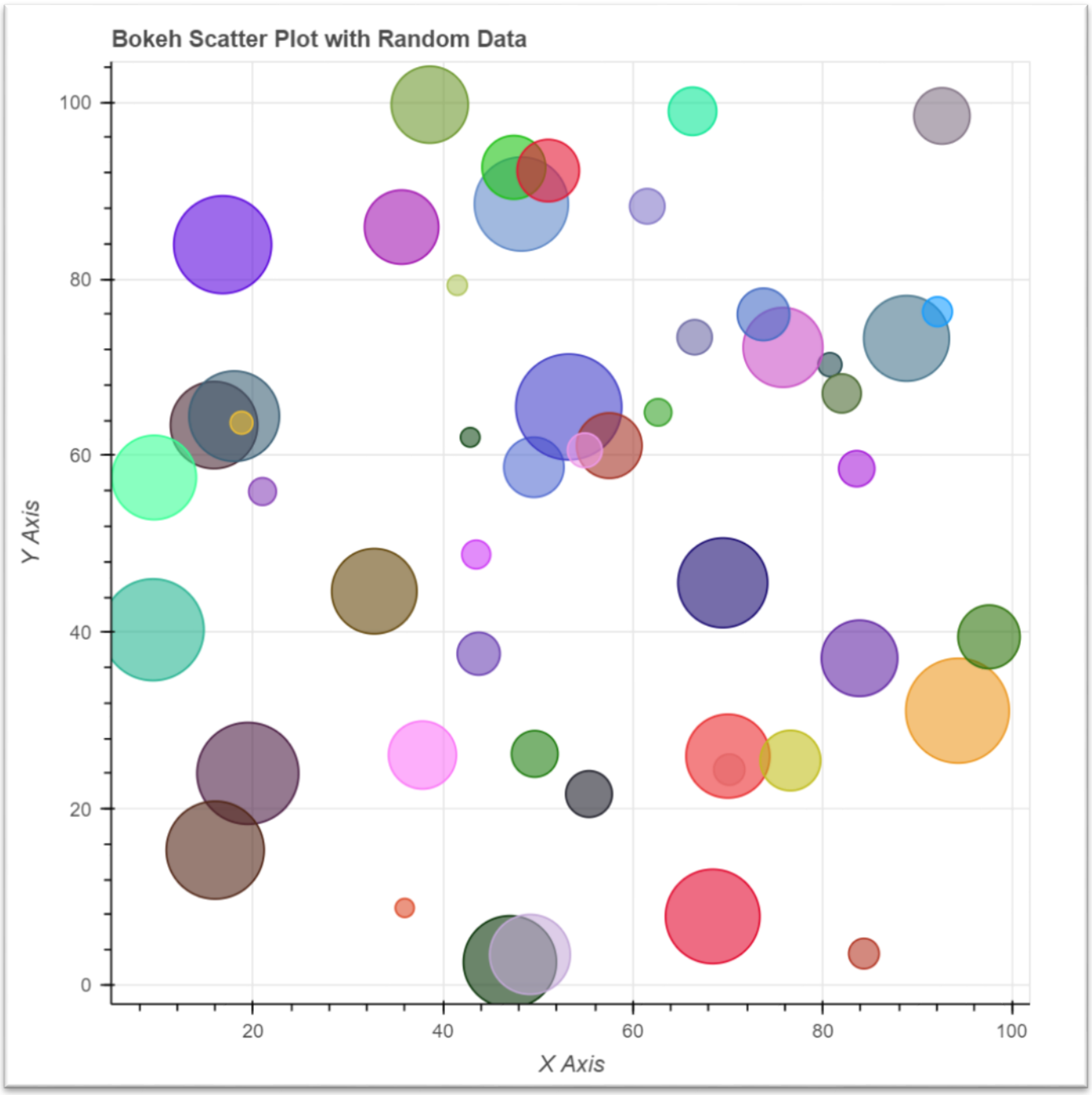
```
# Create a Bokeh figure

fig = bp.figure(title="Sine Wave", x_axis_label='X Axis', y_axis_label='Y Axis')


# Add a line plot to the figure

fig.line(x, y, line_width=2, color="blue")


# Show the plot

show(fig)
```



2.Create a Bokeh scatter plot using randomly generated x and y values. Use different sizes and colors for the markers based on the 'sizes' and 'colors' columns.

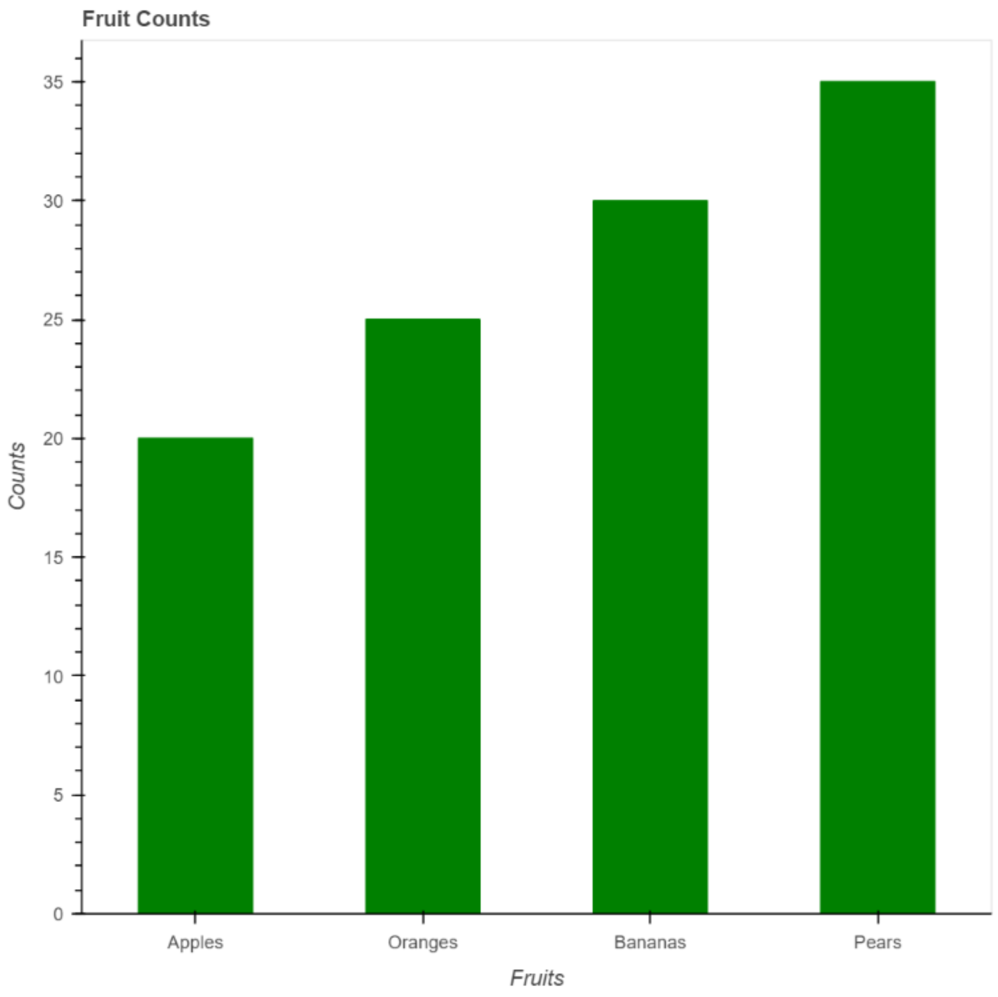
```
from bokeh.plotting import figure, show

from bokeh.io import output_notebook
```

```
import numpy as np

import random


# Prepare output to be displayed in the notebook

output_notebook()


# Generate random data for x and y values

x = np.random.rand(50) * 100

y = np.random.rand(50) * 100


# Generate random sizes and colors

sizes = np.random.rand(50) * 50 + 10  # sizes between 10 and 60

colors = ["#" + ''.join([random.choice('0123456789ABCDEF') for _ in range(6)]) for _ in range(50)]


# Create a scatter plot

p = figure(title="Bokeh Scatter Plot with Random Data", x_axis_label='X Axis', y_axis_label='Y Axis')


# Add scatter renderer

p.scatter(x, y, size=sizes, color=colors, fill_alpha=0.6)


# Show the plot

show(p)
```



3. Generate a Bokeh bar chart representing the counts of different fruits using the following dataset. fruits = ['Apples', 'Oranges', 'Bananas', 'Pears'] counts = [20, 25, 30, 35].

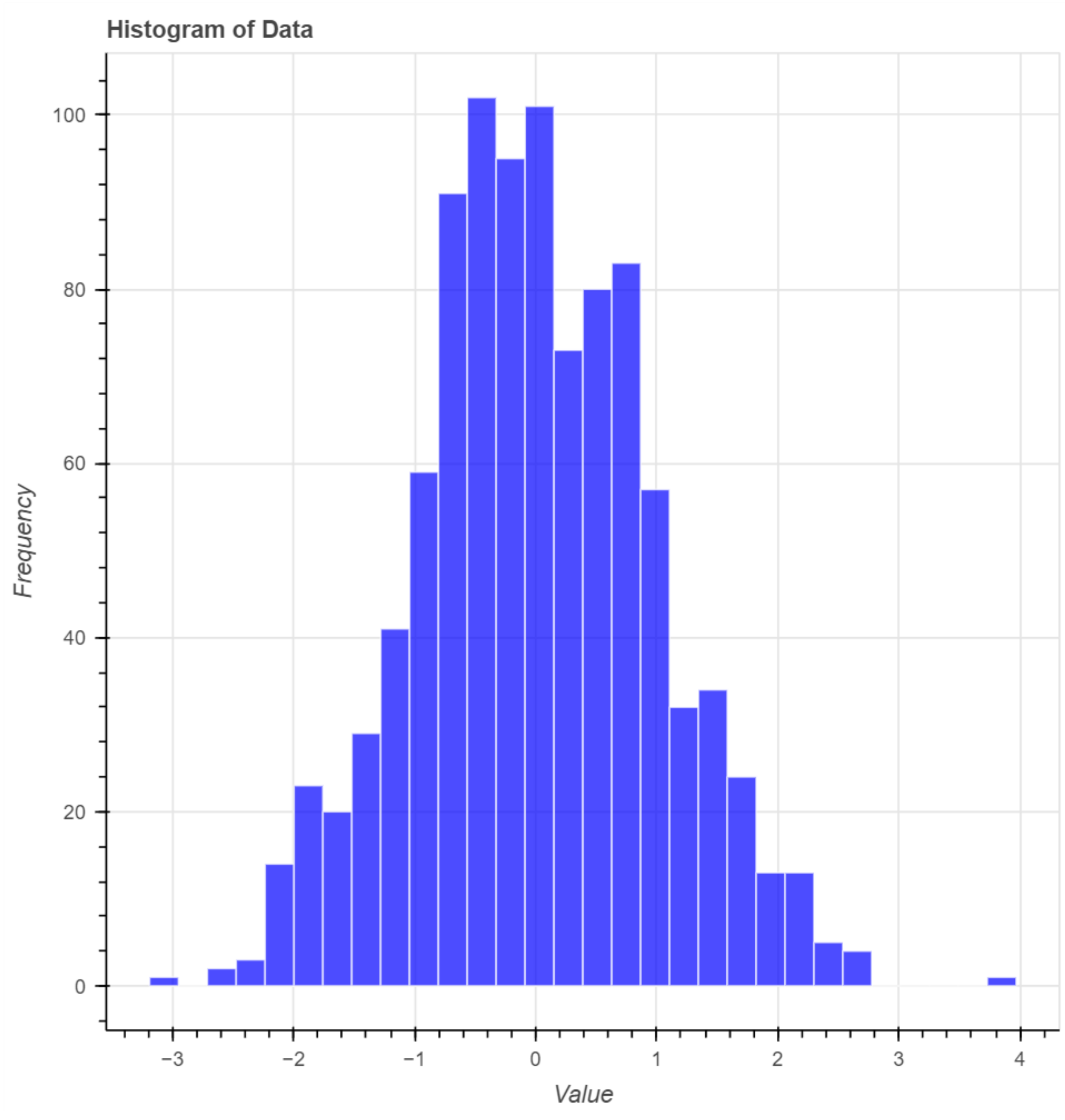
```
# Import necessary modules
from bokeh.plotting import figure, show
from bokeh.io import output_file

# Dataset
fruits = ['Apples', 'Oranges', 'Bananas', 'Pears']
counts = [20, 25, 30, 35]

# Specify the output file to save the plot
output_file("fruit_counts_bar_chart.html")

# Create a Bokeh figure
p = figure(x_range=fruits, title="Fruit Counts", x_axis_label='Fruits', y_axis_label='Counts',
        toolbar_location=None, tools="")

# Add a bar renderer
p.vbar(x=fruits, top=counts, width=0.5, color="green")

# Customize appearance
p.y_range.start = 0
p.xgrid.grid_line_color = None
p.ygrid.grid_line_color = None

# Show the plot
show(p)
```



4. Create a Bokeh histogram to visualize the distribution of the given data. data_hist = np.random.randn(1000) hist, edges = np.histogram(data_hist, bins=30).

```
import numpy as np

from bokeh.plotting import figure, show

from bokeh.io import output_file


# Generate random data for the histogram

data_hist = np.random.randn(1000)
```

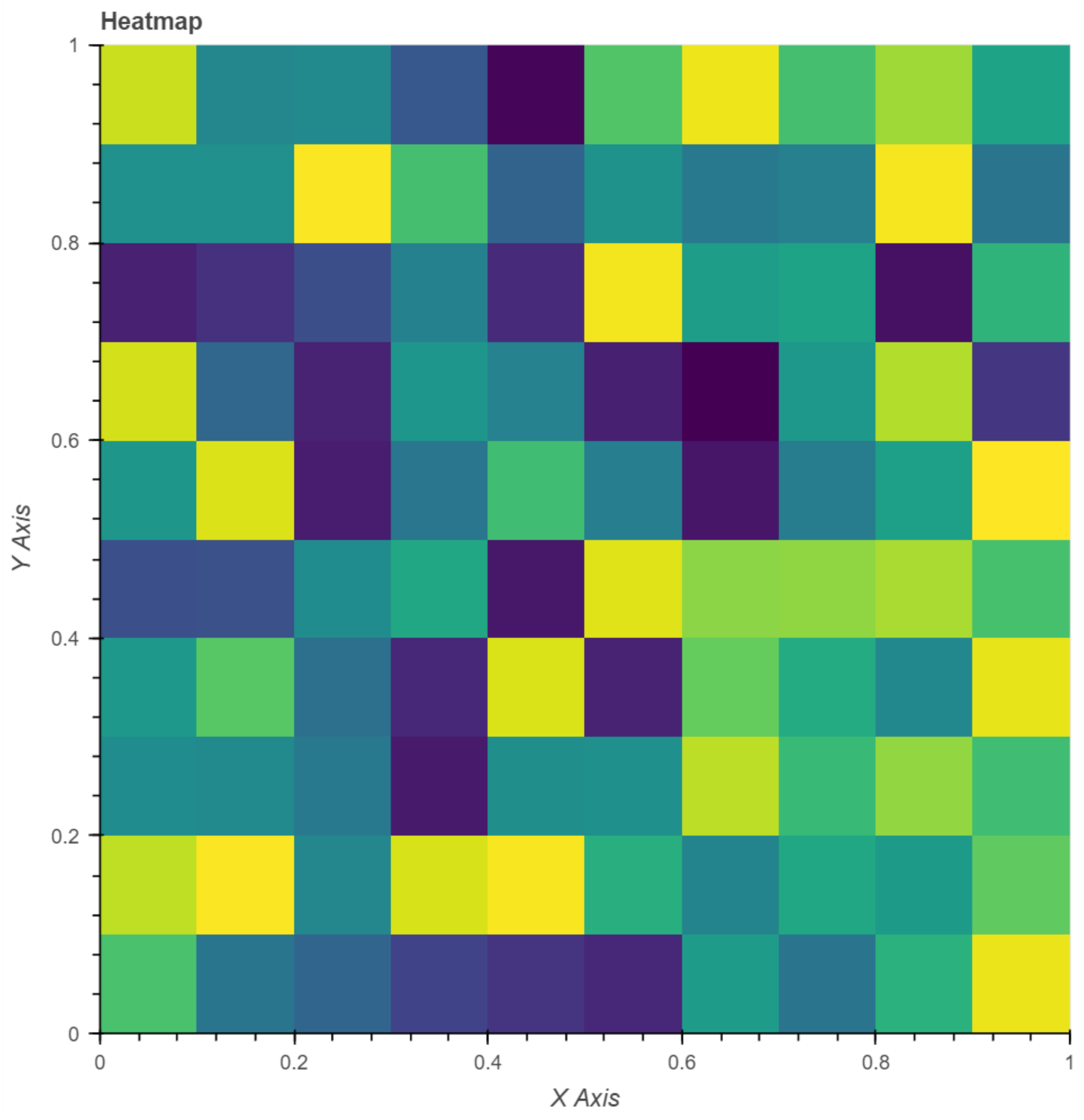
```
# Compute the histogram

hist, edges = np.histogram(data_hist, bins=30)


# Specify the output file to save the plot

output_file("histogram_plot.html")


# Create a Bokeh figure

p = figure(title="Histogram of Data", x_axis_label='Value', y_axis_label='Frequency')


# Add a quad renderer for the histogram

p.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:], fill_color="blue", line_color="white",
alpha=0.7)


# Show the plot

show(p)
```



5. Create a Bokeh heatmap using the provided dataset. data_heatmap = np.random.rand(10, 10) x = np.linspace(0, 1, 10) y = np.linspace(0, 1, 10) xx, yy = np.meshgrid(x, y).

```
import numpy as np

from bokeh.plotting import figure, show

from bokeh.io import output_file
```

```
# Provided data for the heatmap
data_heatmap = np.random.rand(10, 10)
x = np.linspace(0, 1, 10)
y = np.linspace(0, 1, 10)
xx, yy = np.meshgrid(x, y)

# Specify the output file to save the plot
output_file("heatmap_plot.html")

# Create a Bokeh figure
p = figure(title="Heatmap", x_axis_label='X Axis', y_axis_label='Y Axis',
        x_range=(0, 1), y_range=(0, 1))

# Add image glyph for heatmap
# Bokeh expects a list of 2D arrays for the image parameter
p.image(image=[data_heatmap], x=0, y=0, dw=1, dh=1, palette="Viridis256")

# Show the plot
show(p)
```


Heatmap
1
0.8
0.6
0.4
0.2
0
Y Axis
0
0.2
0.4
0.6
0.8
1
X Axis