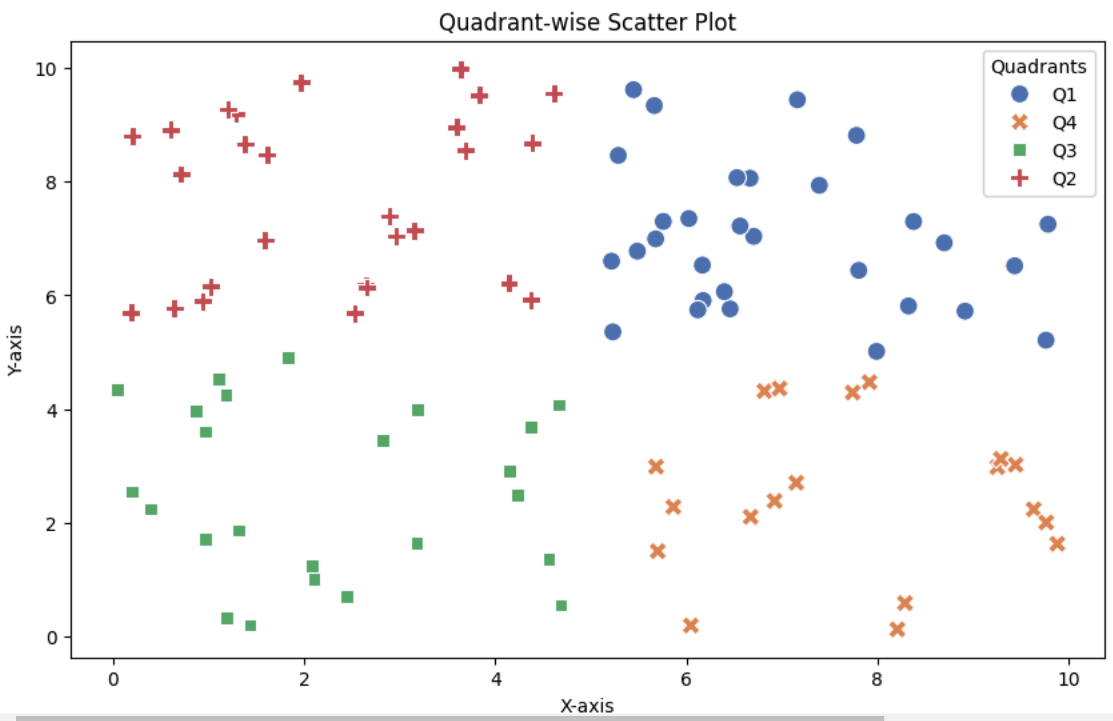
ques 16.Create a Seaborn scatter plot of two random arrays, color points based on their position relative to the origin (quadrants), add a legend, label the axes, and set the title as 'Quadrant-wise Scatter Plot'.

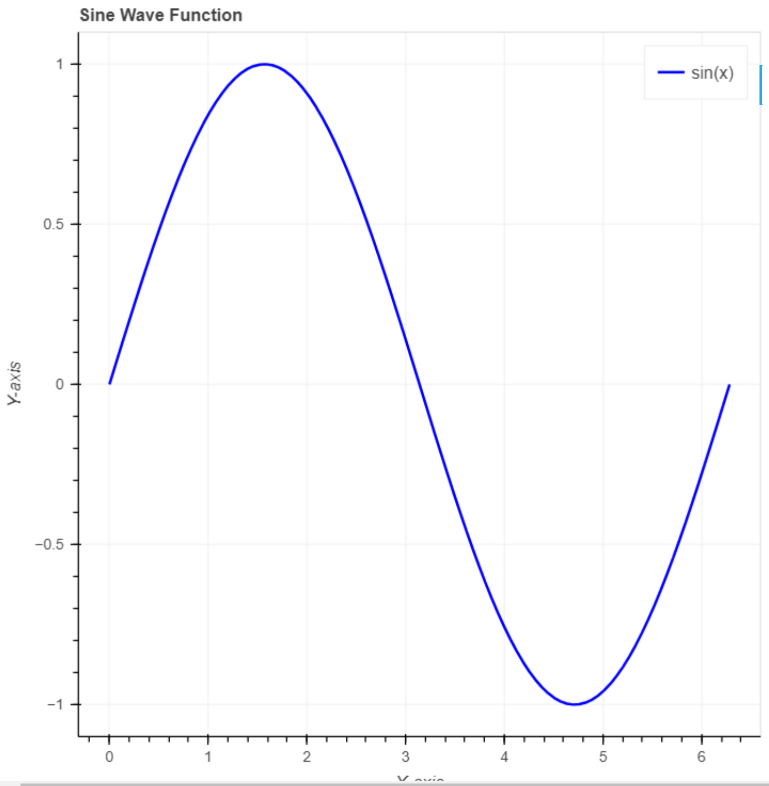
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Create two random arrays
np.random.seed(0)  # For reproducibility
x = np.random.rand(100) * 10  # Random values for x-axis
y = np.random.rand(100) * 10  # Random values for y-axis

# Create a DataFrame and define a column for quadrant
df = pd.DataFrame({'x': x, 'y': y})

# Determine the quadrant for each point
df['Quadrant'] = np.where((df['x'] >= 5) & (df['y'] >= 5), 'Q1',  # Quadrant 1
                          np.where((df['x'] < 5) & (df['y'] >= 5), 'Q2',  # Quadrant 2
                                   np.where((df['x'] < 5) & (df['y'] < 5), 'Q3',  # Quadrant 3
                                            'Q4')))  # Quadrant 4
# Create a scatter plot
plt.figure(figsize=(10, 6))
scatter = sns.scatterplot(data=df, x='x', y='y', hue='Quadrant', palette='deep', style='Quadrant', s=100)
plt.legend(title='Quadrants')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Quadrant-wise Scatter Plot')
plt.show()
```



ques 17.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
from bokeh.io import output_notebook
import numpy as np

# Prepare the data
x = np.linspace(0, 2 * np.pi, 100)  # 100 points from 0 to 2π
y = np.sin(x)  # Sine wave values

# Output to notebook (use output_file('sine_wave.html') for a standalone HTML file)
output_notebook()
```

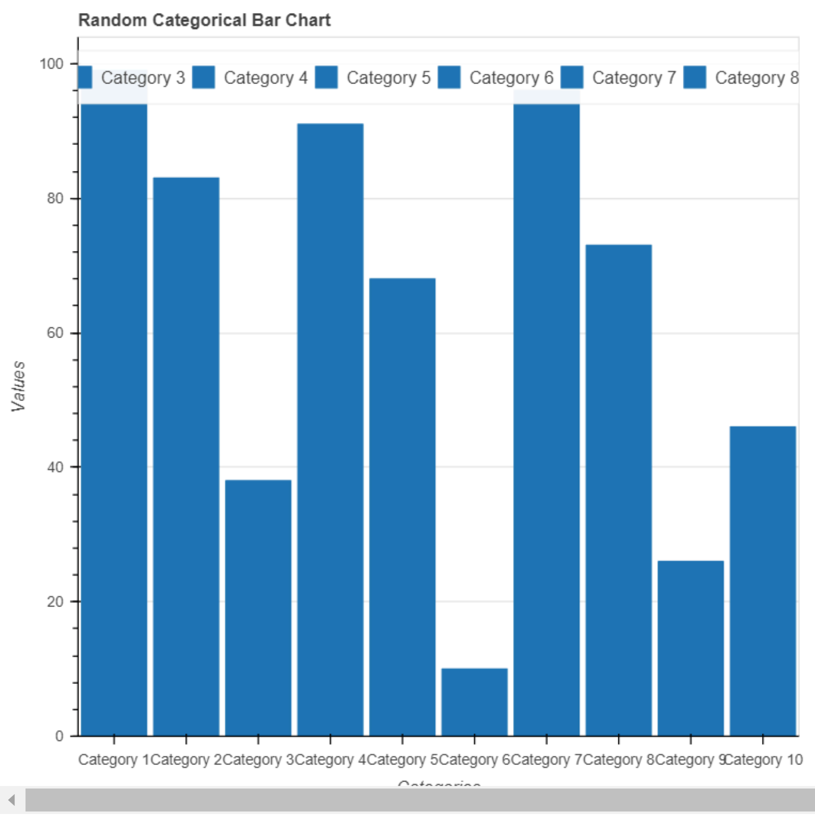
```
# Create a new plot with a title and axis labels
p = figure(title='Sine Wave Function', x_axis_label='X-axis', y_axis_label='Y-axis')
p.line(x, y, legend_label='sin(x)', line_width=2, color='blue')
p.grid.grid_line_alpha = 0.3  # Set the transparency of grid lines
show(p)
```



ques 18.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

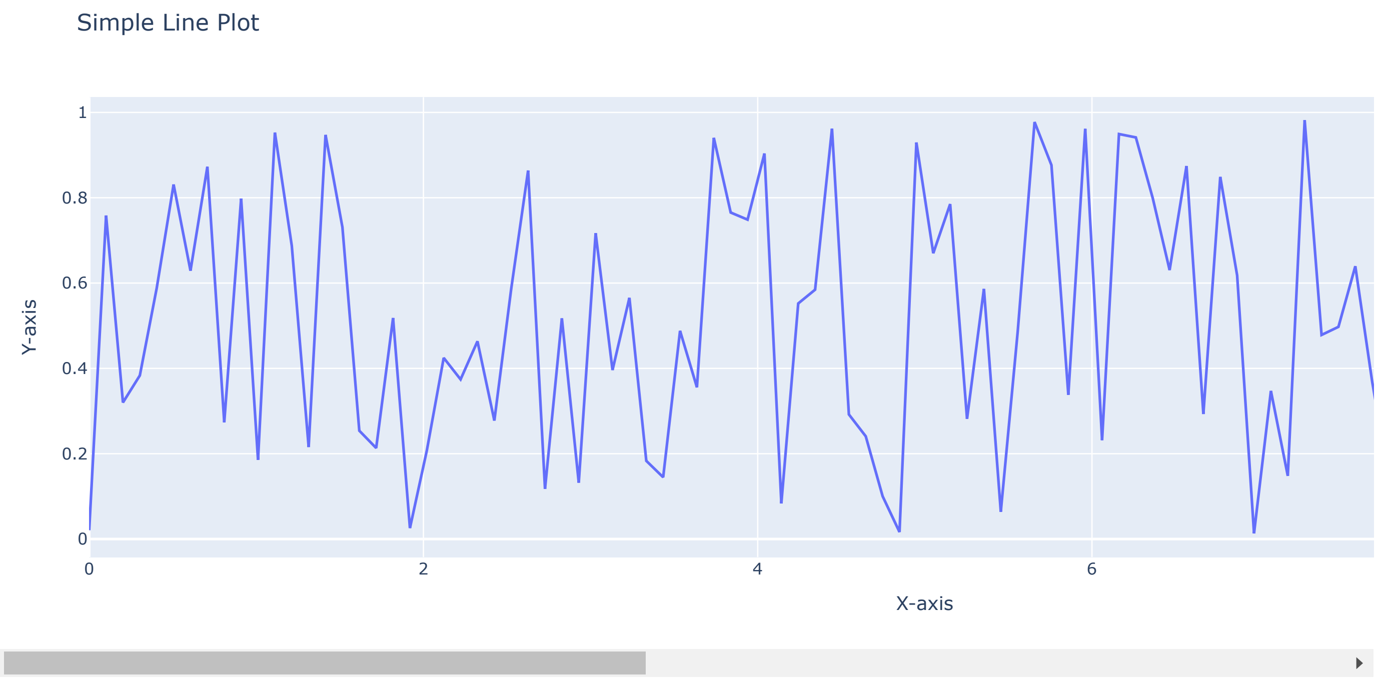
```
import numpy as np
import pandas as pd
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.models import ColumnDataSource, HoverTool

# Prepare the data
categories = [f'Category {i}' for i in range(1, 11)]  # 10 categories
values = np.random.randint(10, 100, size=len(categories))  # Random values for each category

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

# Create a ColumnDataSource
source = ColumnDataSource(data=data)

# Define a color mapping based on values
colors = ['#%02x%02x%02x' % (int(255 * (value / max(values))), 0, 0) for value in values]

# Output to notebook (use output_file('bar_chart.html') for a standalone HTML file)
output_notebook()

# Create a new plot
p = figure(x_range=categories, title='Random Categorical Bar Chart',
           x_axis_label='Categories', y_axis_label='Values', toolbar_location=None)

# Add bars with color mapping and hover tooltips
bars = p.vbar(x='Category', top='Value', source=source, width=0.9,legend_field='Category')
source.data['colors'] =colors

# Add hover tooltips
hover = HoverTool()
```

```
hover.tooltips = [("Category", "@Category"), ("Value", "@Value")]
p.add_tools(hover)

# Customize the plot
p.y_range.start = 0
p.xgrid.grid_line_color = None
p.legend.orientation = "horizontal"
p.legend.location = "top_center"
show(p)
```



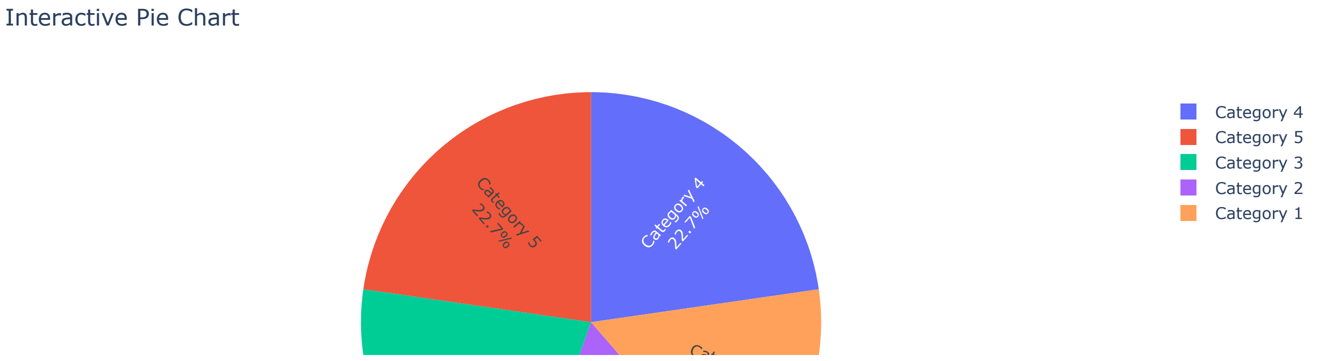
ques 19.Using Plotly, create a basic line plot of a randomly generated dataset, label the axes, and set the title as 'Simple Line Plot'

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

# Generate random data
x = np.linspace(0, 10, 100)  # 100 points from 0 to 10
y = np.random.rand(100)  # Random values for y

# Create a line plot
fig = go.Figure()

# Add a line trace
fig.add_trace(go.Scatter(x=x, y=y, mode='lines', name='Random Data'))
fig.update_layout(
    title='Simple Line Plot',
    xaxis_title='X-axis',
    yaxis_title='Y-axis'
)
fig.show()
```



ques 20. using Plotly, create an interactive pie chart of randomly generated data, add labels and percentages, set the title as 'Interactive Pie Chart'.

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

# Generate random data
np.random.seed(0)  # For reproducibility
labels = [f'Category {i+1}' for i in range(5)]  # Create 5 categories
values = np.random.randint(10, 100, size=len(labels))  # Random values for each category

# Create a pie chart
fig = go.Figure(data=[go.Pie(labels=labels, values=values,
                             textinfo='label+percent',
                             insidetextorientation='radial')])
fig.update_layout(title='Interactive Pie Chart')
fig.show()
```


Interactive Pie Chart
Category 5
22.7%
Category 4
22.7%
Category 4
Category 5
Category 3
Category 2
Category 1