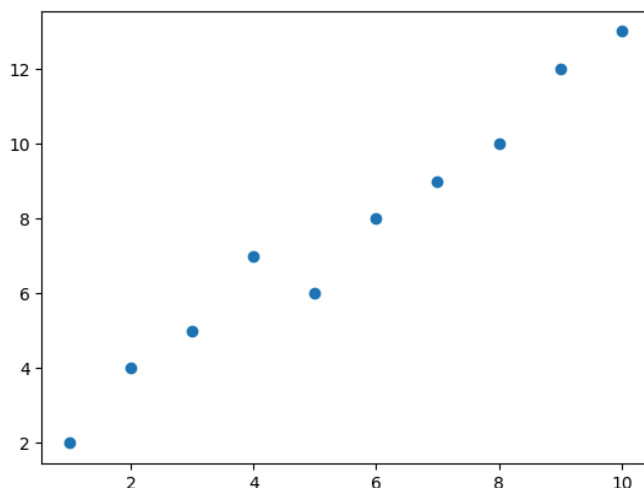


```
#matplotlib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

#1. Create a scatter plot using Matplotlib to visualize the relationship between x and y.

```
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [2, 4, 5, 7, 6, 8, 9, 10, 12, 13]
plt.scatter(x,y)
```

 <matplotlib.collections.PathCollection at 0x7f444f3b67a0>



#2. Generate a line plot to visualize the trend of values for the data.

```
data = np.array([3, 7, 9, 15, 22, 29, 35])
plt.plot(data)
```

Gemini

NameError X

...



You

Please explain the error:

NameError: name 'data_heatmap' is not defined



Gemini

Hypothesis

The error message "NameError: name 'data_heatmap' is not defined" indicates that the variable `data_heatmap` is being used in the code but hasn't been assigned a value yet. Looking at the code, it seems like `data_heatmap` was intended to be defined in the commented-out section:

```
"""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)"""
```



[Use code with caution](#)

Suggested Changes

```
from bokeh.plotting import figure,
from bokeh.models import LinearColorMapper
from bokeh.palettes import Viridis256
import numpy as np # Import numpy
```

```
# Reshape data
data_heatmap = np.random.rand(10, 10)
data_array = data_heatmap.flatten()
```

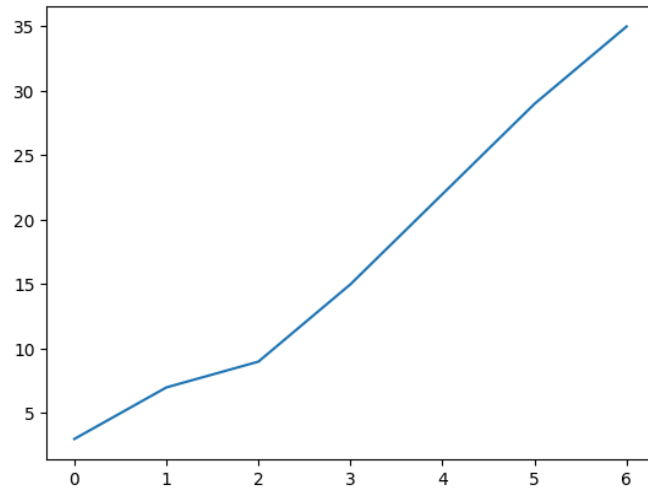
```
# Create color mapper
color_mapper = LinearColorMapper(palette=Viridis256)
```

```
# Create figure
p = figure(title="Heatmap", x_range=0, y_range=0)
```

```
# Plot heatmap
p.rect(x=xx.flatten(), y=yy.flatten(), width=1, height=1)
```

```
# Add color bar
color_bar = ColorBar(color_mapper=color_mapper, location='right')
```

↗ [matplotlib.lines.Line2D at 0x7f444c0f37c0]



```
p.add_layout(color_bar, 'right')
```

```
show(p)
```

[Use code with caution](#)

By uncommenting the line

```
data_heatmap =
```

```
np.random.rand(10, 10) and
```

importing the numpy module, you define the variable and resolve the

NameError .

Rate this answer  

#3.Display a bar chart to represent the frequency of each item

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

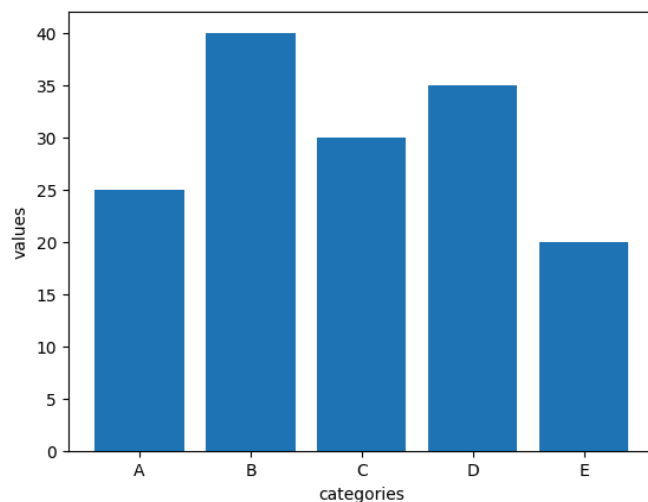
```
values = [25, 40, 30, 35, 20]
```

```
plt.xlabel("categories")
```

```
plt.ylabel("values")
```

```
plt.bar(categories,values)
```

↗ <BarContainer object of 5 artists>

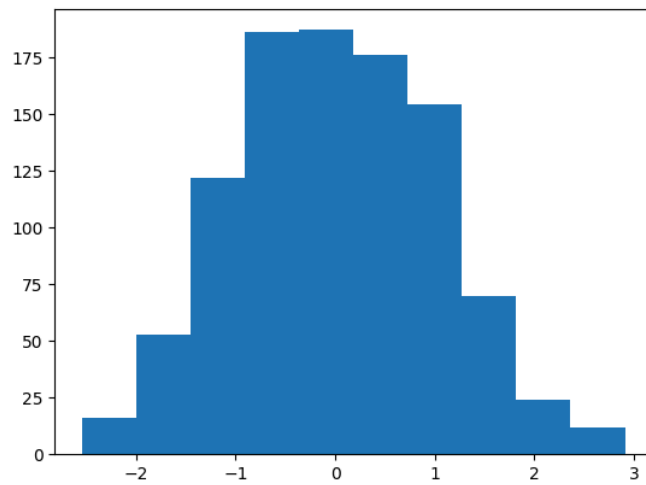


#4.. Create a histogram to visualize the distribution of values

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

```
plt.hist(data)
```

```
➦ (array([ 16.,  53., 122., 186., 187., 176., 154.,
  70.,  24.,  12.]),
  array([-2.54723276, -2.00151152, -1.45579027,
 -0.91006903, -0.36434779,
    0.18137345,  0.72709469,  1.27281594,
  1.81853718,  2.36425842,
    2.90997966])),
  <BarContainer object of 10 artists>)
```

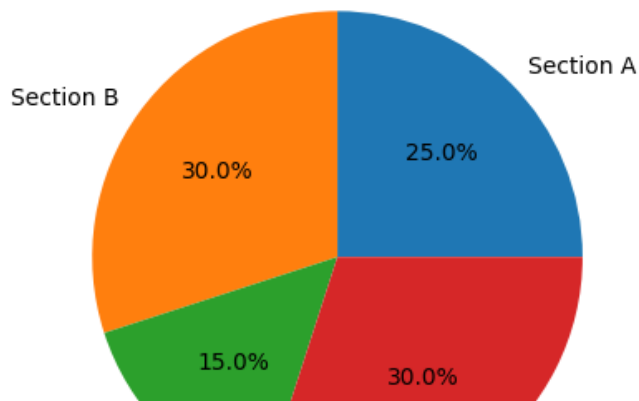


```
#5. Show a pie chart to represent the percentage distribution o
sections = ['Section A', 'Section B', 'Section C', 'Section D']
sizes = [25, 30, 15, 30]
plt.pie(sizes,labels=sections,autopct='%1.1f%%')
```

```

[<matplotlib.patches.Wedge at 0x7f4447dce650>,
 <matplotlib.patches.Wedge at 0x7f4447dce560>,
 <matplotlib.patches.Wedge at 0x7f4447dcf220>,
 <matplotlib.patches.Wedge at 0x7f4447dcf8b0>],
 [Text(0.7778174593052024, 0.7778174593052023,
 'Section A'),
  Text(-0.8899187180267095, 0.6465637441936395,
 'Section B'),
  Text(-0.7778173864806728, -0.777817532129725,
 'Section C'),
  Text(0.646563910834035, -0.8899185969553609,
 'Section D')],
 [Text(0.4242640687119285, 0.42426406871192845,
 '25.0%'),
  Text(-0.48541020983275057, 0.3526711331965306,
 '30.0%'),
  Text(-0.4242640289894578, -0.42426410843439544,
 '15.0%'),
  Text(0.35267122409129176, -0.4854101437938332,
 '30.0%')]]

```



```

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

```

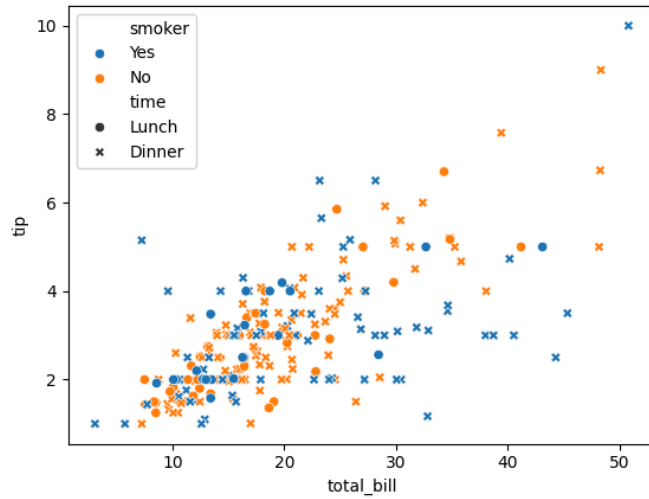
#1. Create a scatter plot to visualize the relationship between

```

tips = sns.load_dataset("tips")
sns.scatterplot(data=tips, x="total_bill", y="tip", hue="smoker

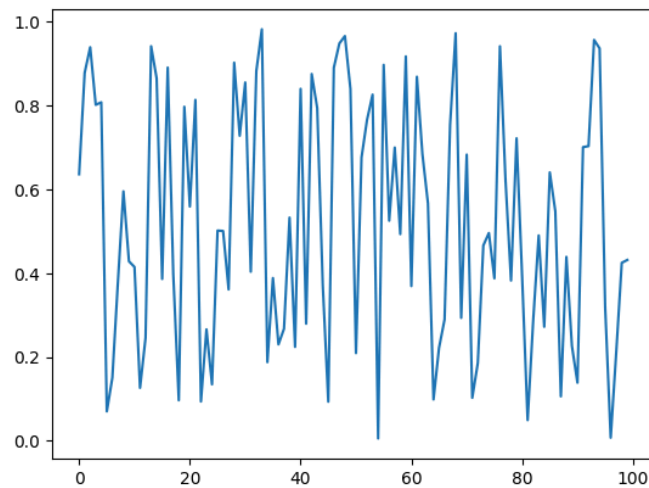
```

 <Axes: xlabel='total_bill', ylabel='tip'>



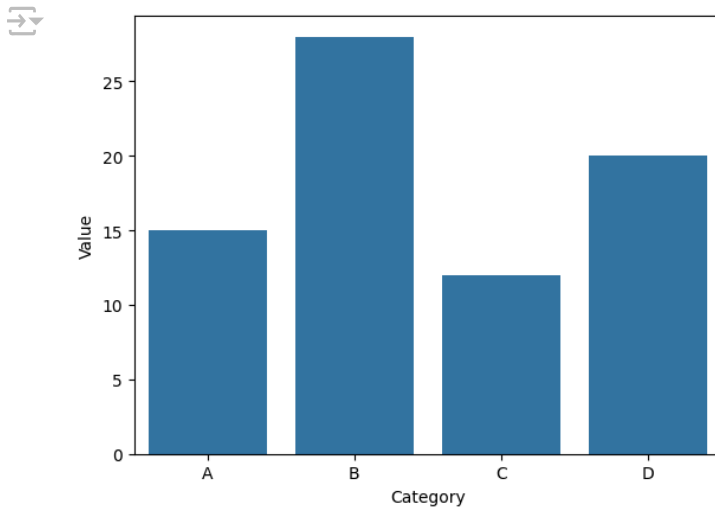
#2.Generate a dataset of random numbers. Visualize the distribu
random_numbers = np.random.rand(100)
plt.plot(random_numbers)

 [



```
#3.Create a dataset representing categories and their corresponding values
data = {'Category': ['A', 'B', 'C', 'D'],
        'Value': [15, 28, 12, 20]}
df = pd.DataFrame(data)

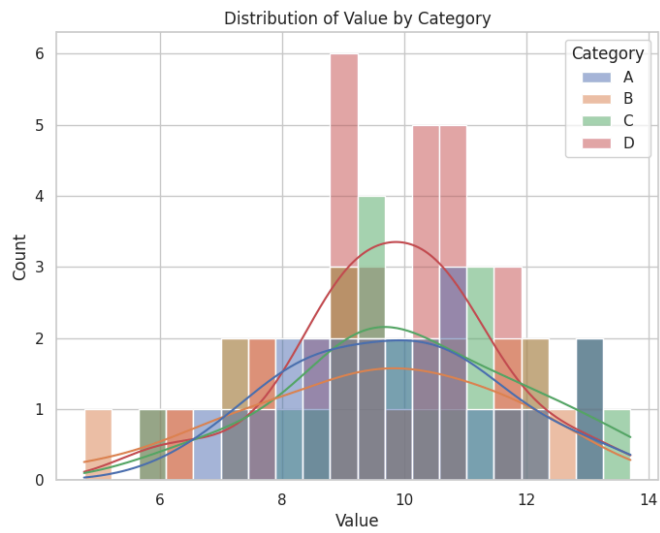
sns.barplot(x='Category', y='Value', data=df)
plt.show()
```



```
#4.Generate a dataset with categories and numerical values. Visualize the distribution of values for each category.
categories = ['A', 'B', 'C', 'D']
num_samples = 100
np.random.seed(42)
values = np.random.normal(loc=10, scale=2, size=num_samples)
category_column = np.random.choice(categories, size=num_samples)

df = pd.DataFrame({'Category': category_column, 'Value': values})

sns.set(style='whitegrid')
plt.figure(figsize=(8, 6))
sns.histplot(data=df, x='Value', hue='Category', bins=20, kde=True)
plt.xlabel('Value')
plt.ylabel('Count')
plt.title('Distribution of Value by Category')
plt.show()
```



#5.Generate a synthetic dataset with correlated features. Visua

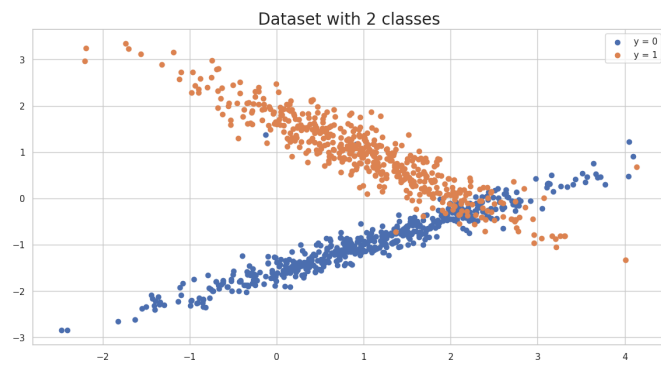
```
from sklearn.datasets import make_classification
```

```
X, y = make_classification(  
    n_samples=1000,  
    n_features=2,  
    n_redundant=0,  
    n_clusters_per_class=1,  
    random_state=42  
)
```

```
df = pd.concat([pd.DataFrame(X), pd.Series(y)], axis=1)  
df.columns = ['x1', 'x2', 'y']
```

```
def plot(df, x1, x2, y, title='', save=False, filename='figure.p  
    plt.figure(figsize=(14, 7))  
    plt.scatter(x=df[df[y] == 0][x1], y=df[df[y] == 0][x2], lab  
    plt.scatter(x=df[df[y] == 1][x1], y=df[df[y] == 1][x2], lab  
    plt.title(title, fontsize=20)  
    plt.legend()  
    if save:  
        plt.savefig(filename, dpi=300, bbox_inches='tight', pad_  
    plt.show()
```

```
plot(df, x1='x1', x2='x2', y='y', title='Dataset with 2 classes
```



```
#plotly
import plotly.graph_objects as go
import plotly.express as px

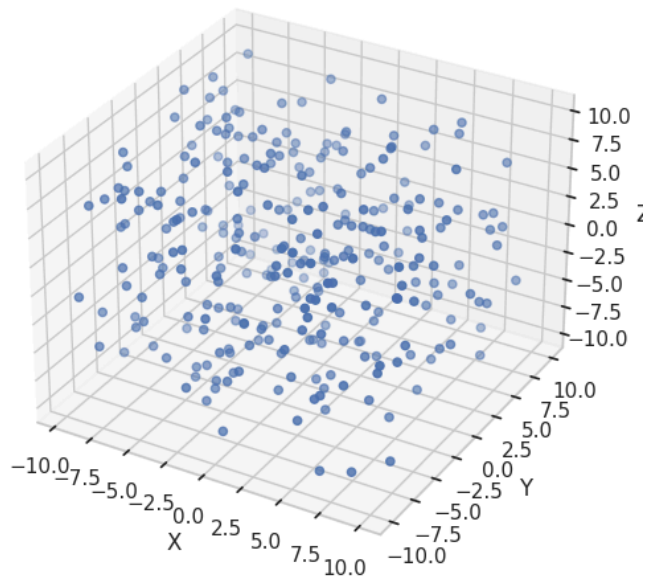
#1. Using the given dataset, to generate a 3D scatter plot to v
"""np.random.seed(30)
data = {
    'X': np.random.uniform(-10, 10, 300),
    'Y': np.random.uniform(-10, 10, 300),
    'Z': np.random.uniform(-10, 10, 300)
}
df = pd.DataFrame(data)"""

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)
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['X'], df['Y'], df['Z'], c='b', marker='o')

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('3D Scatter Plot')
plt.show()
```




3D Scatter Plot



```

""" Using the Student Grades, create a violin plot to display
categories.
2p.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)

## Using the sales data, generate a heatmap to visualize the va
different months and days.

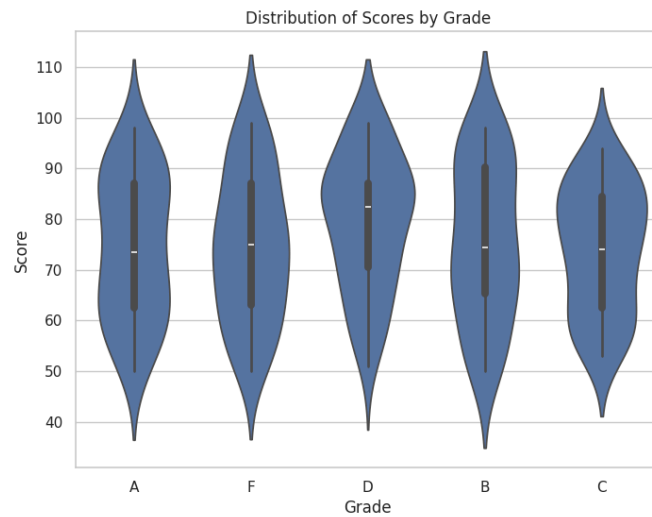
np.random.seed(20)
data = {
    'Month': np.random.choice(['Jan', 'Feb', 'Mar', 'Apr', 'May
    'Day': np.random.choice(range(1, 31), 100),
    'Sales': np.random.randint(1000, 5000, 100)
}
df = pd.DataFrame(data)"""

np.random.seed(15)
data = {
    'Grade': np.random.choice(['A', 'B', 'C', 'D', 'F'], 200),
    'Score': np.random.randint(50, 100, 200)
}
df_grades = pd.DataFrame(data) # Changed df to df_grades

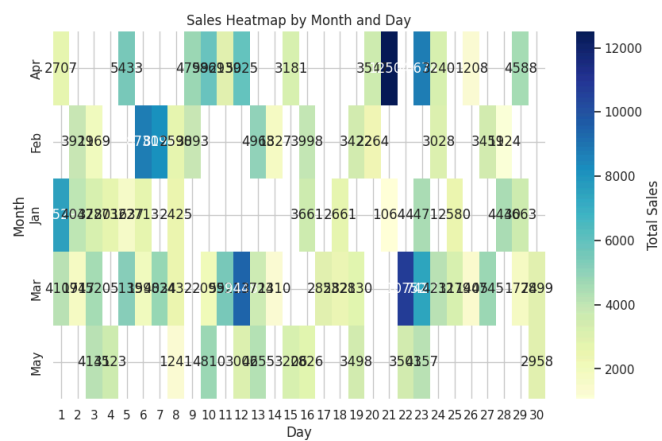
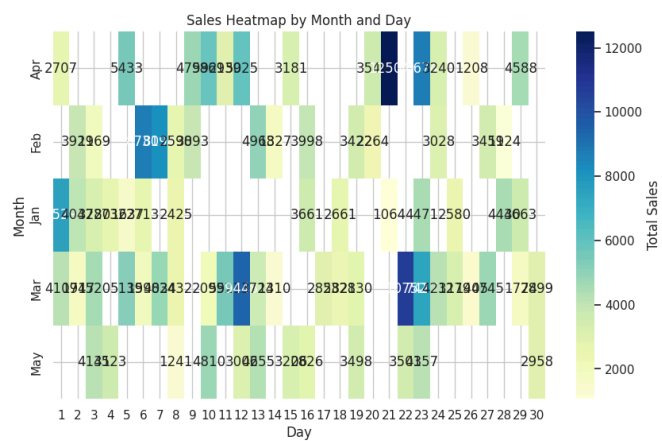
np.random.seed(20)
data = {
    'Month': np.random.choice(['Jan', 'Feb', 'Mar', 'Apr', 'May
    'Day': np.random.choice(range(1, 31), 100),
    'Sales': np.random.randint(1000, 5000, 100)
}
df_sales = pd.DataFrame(data) # Changed df to df_sales

plt.figure(figsize=(8, 6))
sns.violinplot(x='Grade', y='Score', data=df_grades) # Changed
plt.title('Distribution of Scores by Grade')
plt.xlabel('Grade')
plt.ylabel('Score')
plt.show()

```



```
#3. Using the sales data, generate a heatmap to visualize the v.
"""np.random.seed(20)
data = {
    'Month': np.random.choice(['Jan', 'Feb', 'Mar', 'Apr', 'May
    '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
    'Day': np.random.choice(range(1, 31), 100),
    'Sales': np.random.randint(1000, 5000, 100)
}
df = pd.DataFrame(data)
sales_matrix = df.pivot_table(index='Month', columns='Day', val
plt.figure(figsize=(10, 6))
sns.heatmap(sales_matrix, cmap='YlGnBu', annot=True, fmt='.0f',
plt.title('Sales Heatmap by Month and Day')
plt.xlabel('Day')
plt.ylabel('Month')
plt.show()
```



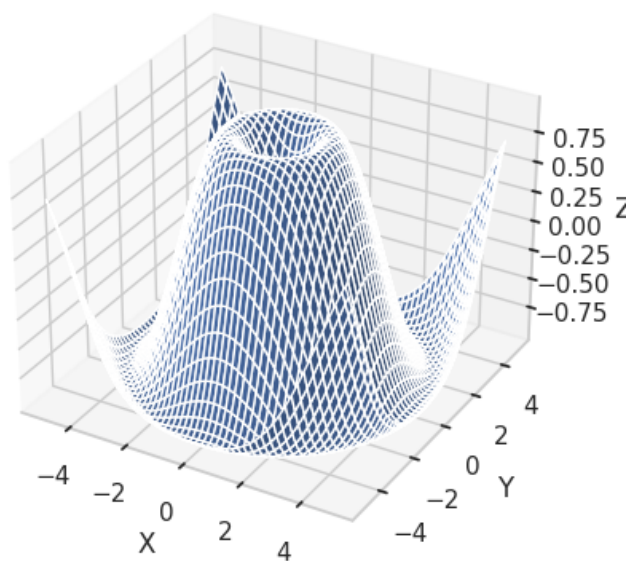
```
#4.Using the given x and y data, generate a 3D surface plot to
"""x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = np.sin(np.sqrt(x**2 + y**2))
data = {
    'X': x.flatten(),
    'Y': y.flatten(),
    'Z': z.flatten()
}
df = pd.DataFrame(data)"""
x = np.linspace(-5,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

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, y, z)

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.title('3D Surface Plot of  $z=\sin(x^2+y^2)^{1/2}$ ')
plt.show()
```



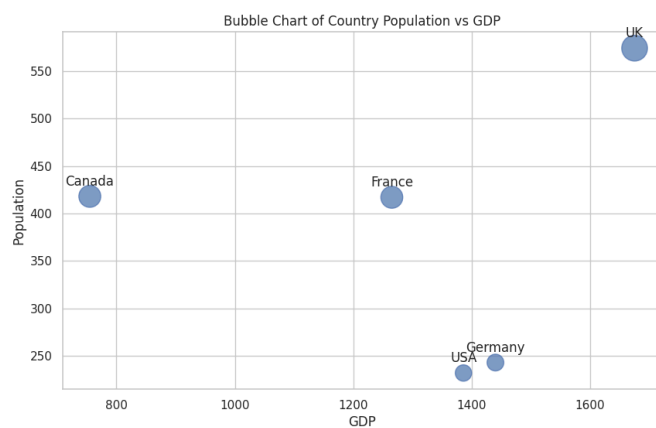
3D Surface Plot of $z=\sin(x^2+y^2)^{1/2}$



```
#5.create a bubble chart to represent each country's 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)"""
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)
plt.figure(figsize=(10, 6))
plt.scatter(x=df['GDP'], y=df['Population'], s=df['Population'])
plt.xlabel('GDP')
plt.ylabel('Population')
plt.title('Bubble Chart of Country Population vs GDP')

# Add country labels
for i, row in df.iterrows():
    plt.annotate(row['Country'], (row['GDP'], row['Population'])

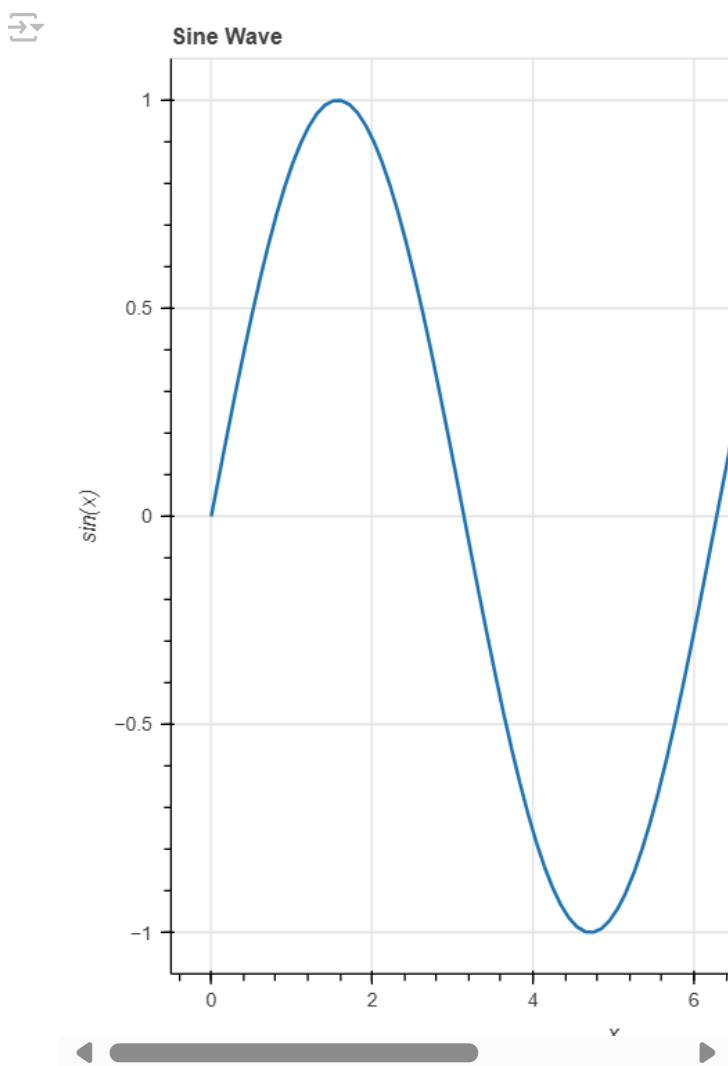
plt.show()
```



```
#bokeh.  
import bokeh.io  
import bokeh.plotting  
bokeh.io.output_notebook()
```

#1.1.Create a Bokeh plot displaying a sine wave. Set x-values f

```
import numpy as np  
from bokeh.io import output_notebook, show  
from bokeh.plotting import figure  
  
x = np.linspace(0, 10, 100)  
y = np.sin(x)  
output_notebook() # Call the imported function  
p = figure(title="Sine Wave", x_axis_label='x', y_axis_label='s'  
  
p.line(x, y, legend_label="sin(x)", line_width=2)  
  
show(p)
```



```
#2.Create a Bokeh scatter plot using randomly generated x and y
from bokeh.plotting import figure, output_file, show
import random

# Create random data
x = [random.random() for _ in range(100)]
y = [random.random() for _ in range(100)]

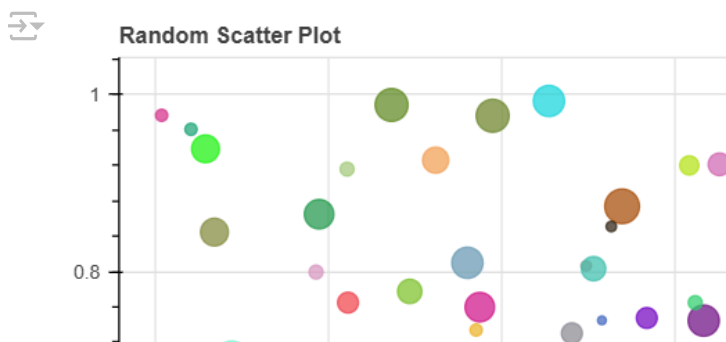
# Generate random sizes and colors
sizes = [random.randint(5, 20) for _ in range(100)]
colors = [f"#{random.randint(0, 0xFFFFFF):06x}" for _ in range(100)]

# Create a Bokeh figure
p = figure(title="Random Scatter Plot")

# Plot the scatter points
p.scatter(x, y, size=sizes, color=colors, alpha=0.7)

# Save the plot to an HTML file
output_file("scatter_plot.html")

# Show the plot
show(p)
```



Gemini access in Colab is available for a limited time for unsubscribed users. [Upgrade to Colab Pro](#)

Enter a prompt here

0 / 400

Responses may display inaccurate or offensive information that doesn't represent Google's views. [Learn more](#)