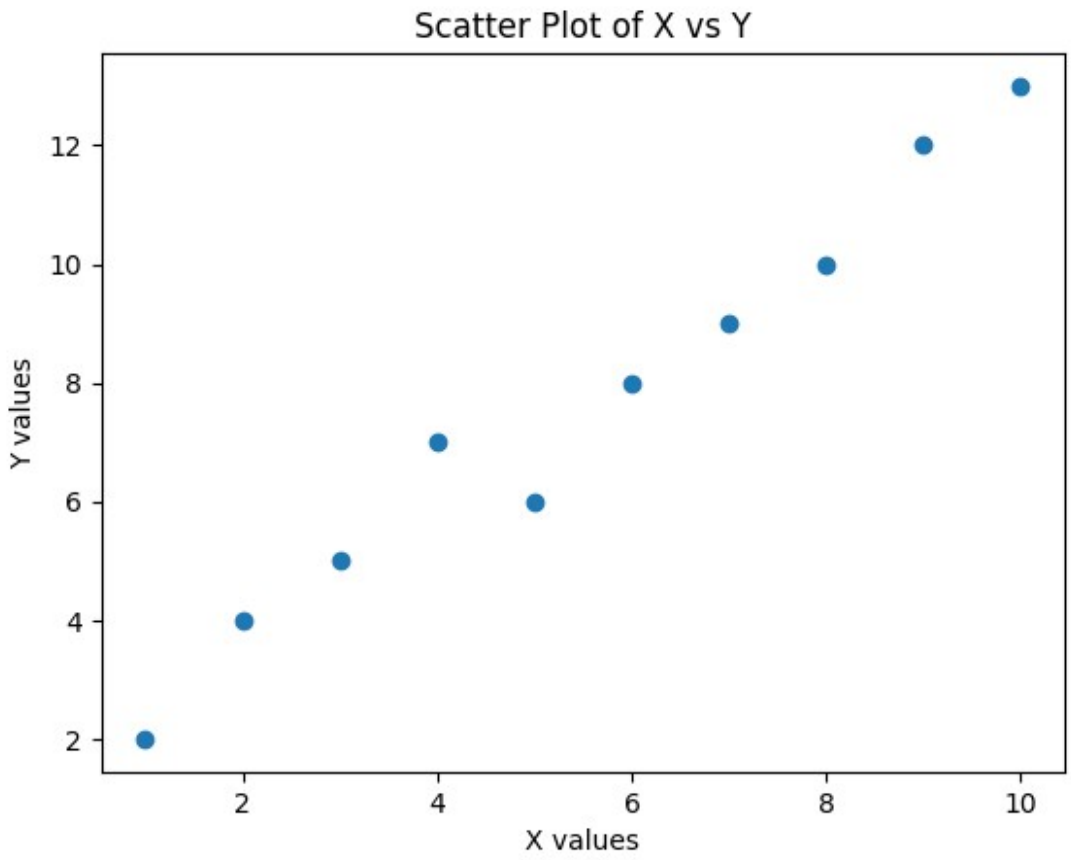
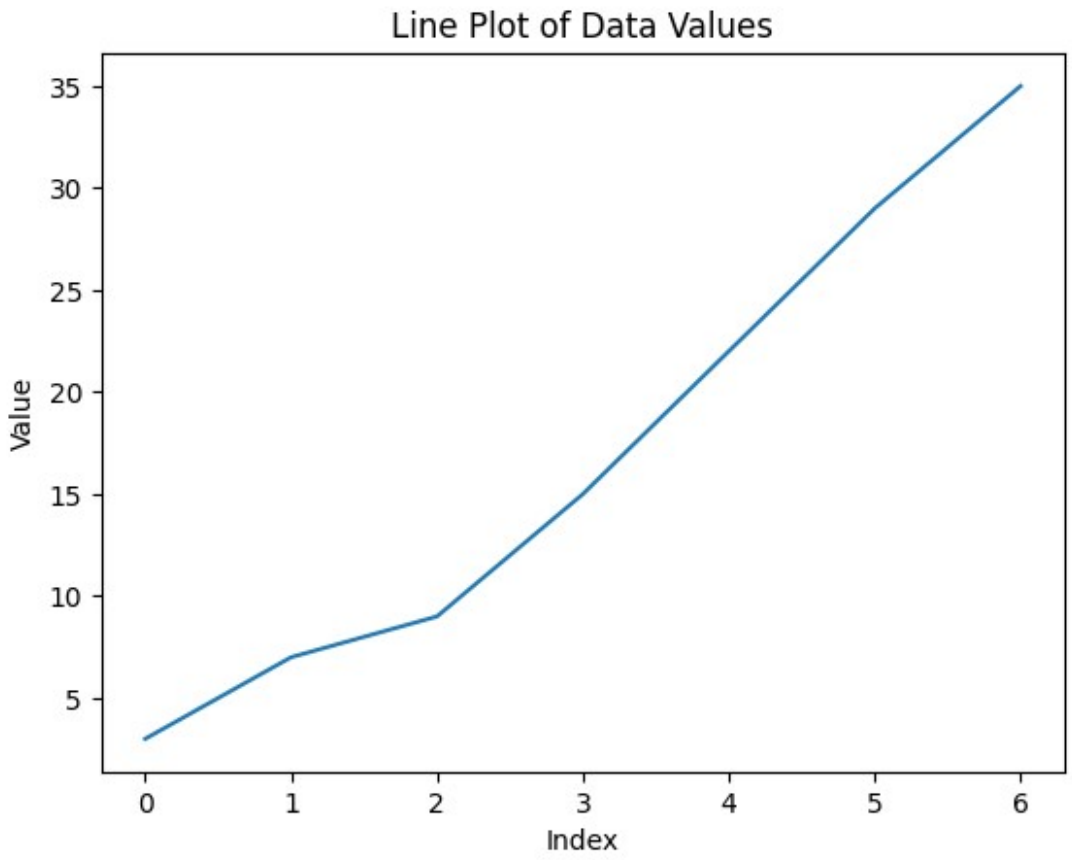
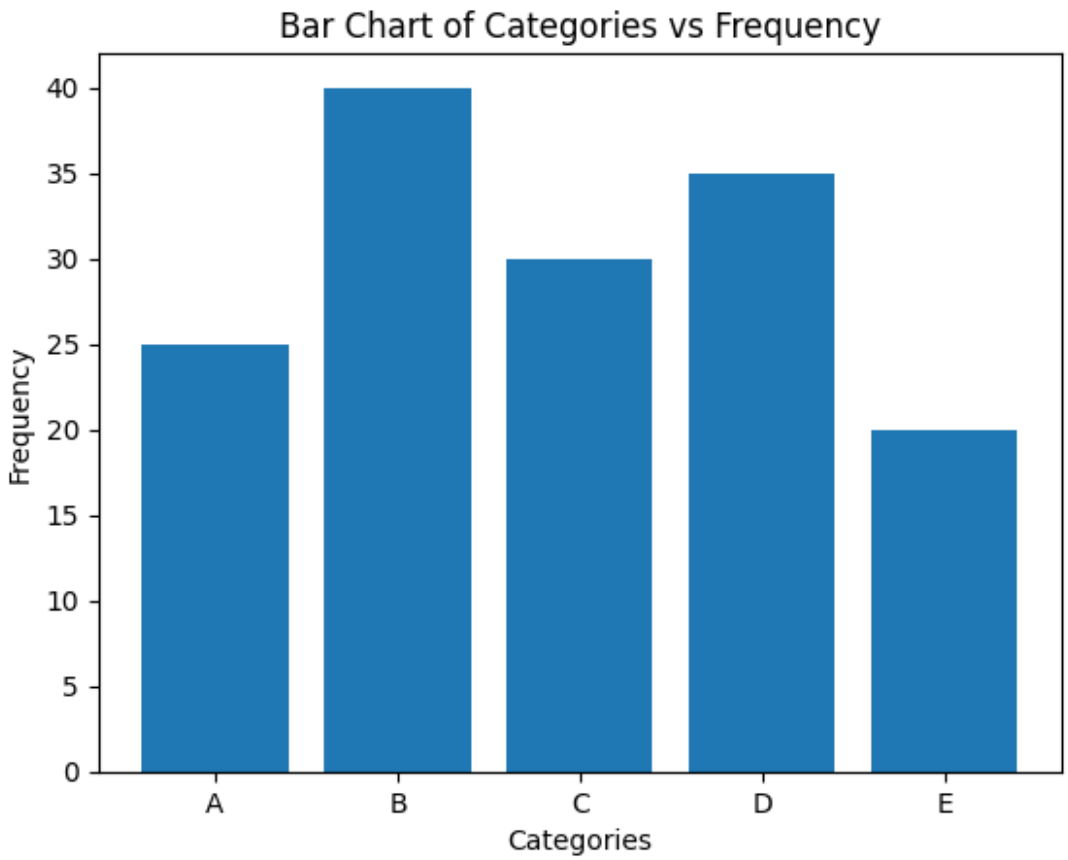
```python
# MATPLOTLIB ASSIGNMENT:-

# 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]

 # Ans:-
import matplotlib.pyplot as plt

# Data
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [2, 4, 5, 7, 6, 8, 9, 10, 12, 13]

# Create a scatter plot
plt.scatter(x, y)

# Add labels and title
plt.xlabel('X values')
plt.ylabel('Y values')
plt.title('Scatter Plot of X vs Y')

# Show plot
plt.show()
```



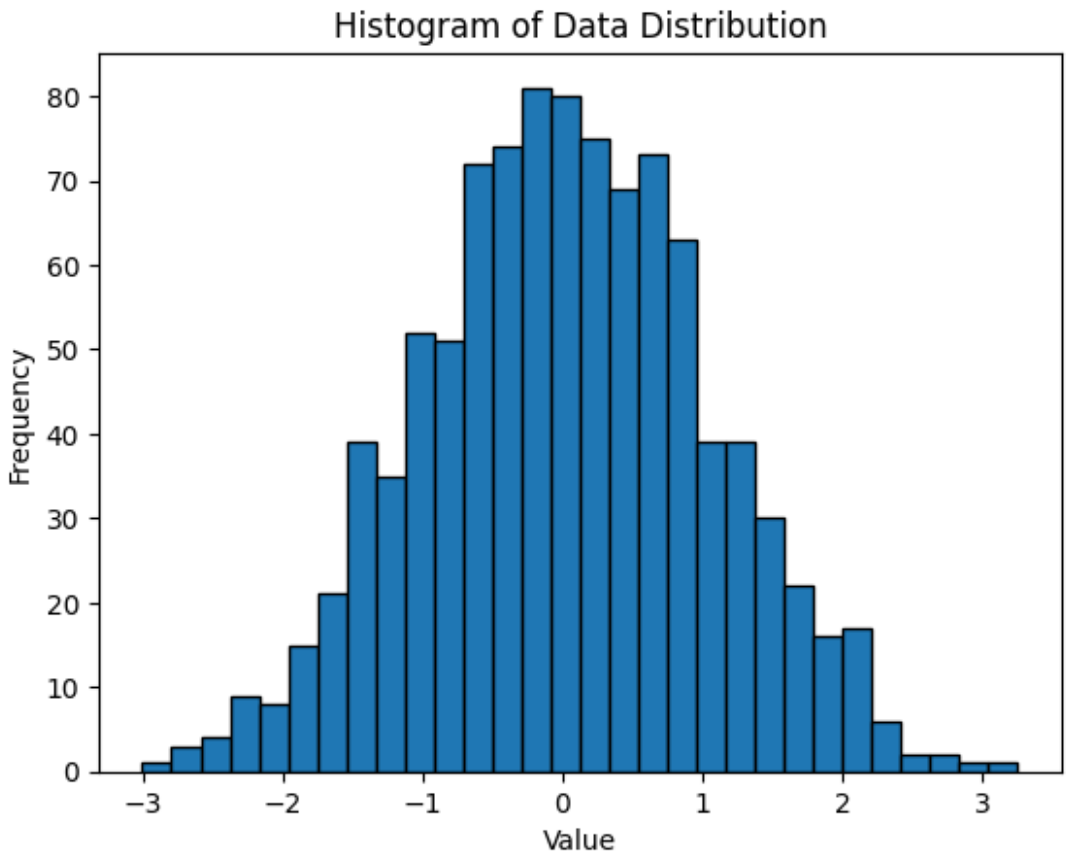

```python
# 2.Generate a line plot to visualize the trend of values for the
given data .
# data = np.array([3, 7, 9, 15, 22, 29, 35])

#Ans:-
import matplotlib.pyplot as plt
import numpy as np

# Data
data = np.array([3, 7, 9, 15, 22, 29, 35])

# Create a line plot
plt.plot(data)

# Add labels and title
plt.xlabel('Index')
plt.ylabel('Value')
plt.title('Line Plot of Data Values')

# Show plot
plt.show()
```



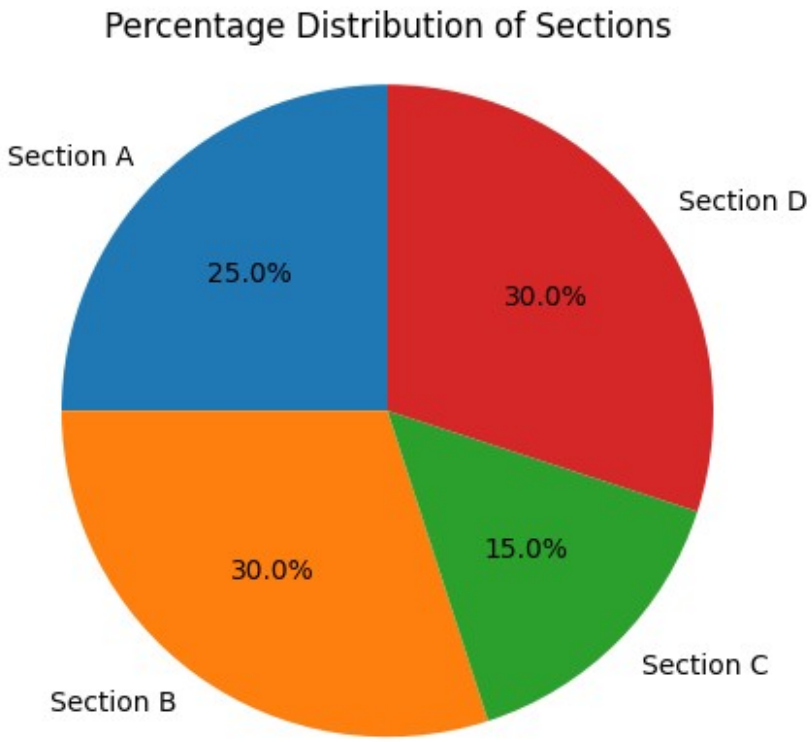

```python
# 3.Display a bar chart to represent the frequency of each item in the given array categories.
# categories = ['A', 'B', 'C', 'D', 'E']
#  values = [25, 40, 30, 35, 20]


# Ans:-
import matplotlib.pyplot as plt

# Data
categories = ['A', 'B', 'C', 'D', 'E']
values = [25, 40, 30, 35, 20]

# Create a bar chart
plt.bar(categories, values)

# Add labels and title
plt.xlabel('Categories')
plt.ylabel('Frequency')
plt.title('Bar Chart of Categories vs Frequency')

# Show plot
plt.show()
```




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

#Ans:-
import numpy as np
import matplotlib.pyplot as plt

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

# Create a histogram
plt.hist(data, bins=30, edgecolor='black')

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

# Show plot
plt.show()
```

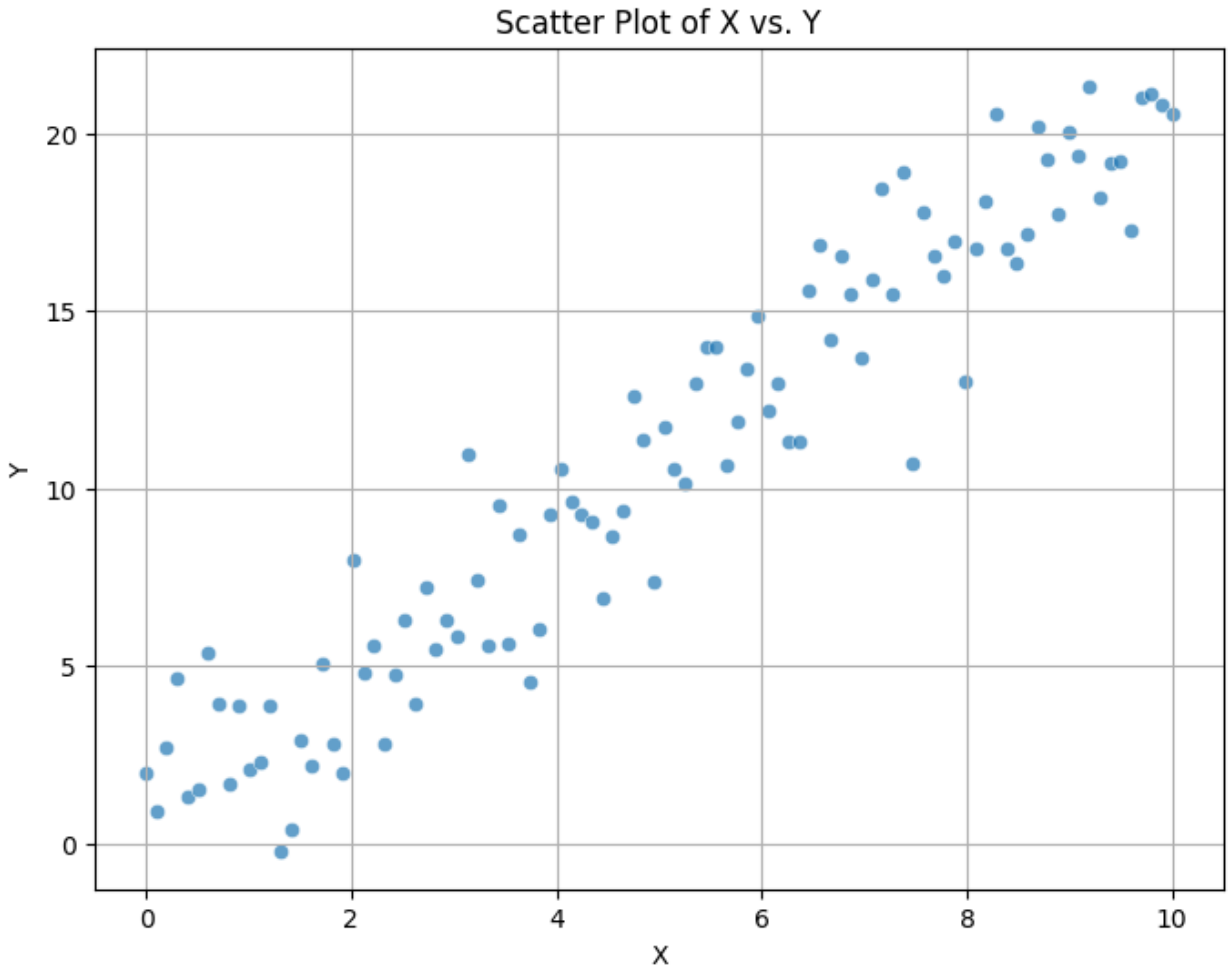



```python
# 5.Show a pie chart to represent the percentage distribution of
different sections in the array `sections.
# sections = ['Section A', 'Section B', 'Section C', 'Section D']
# sizes = [25, 30, 15, 30]

#Ans:-
import matplotlib.pyplot as plt

# Data
sections = ['Section A', 'Section B', 'Section C', 'Section D']
sizes = [25, 30, 15, 30]

# Create a pie chart
plt.pie(sizes, labels=sections, autopct='%1.1f%%', startangle=90)

# Equal aspect ratio ensures that pie is drawn as a circle.
plt.axis('equal')

# Add a title
plt.title('Percentage Distribution of Sections')

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



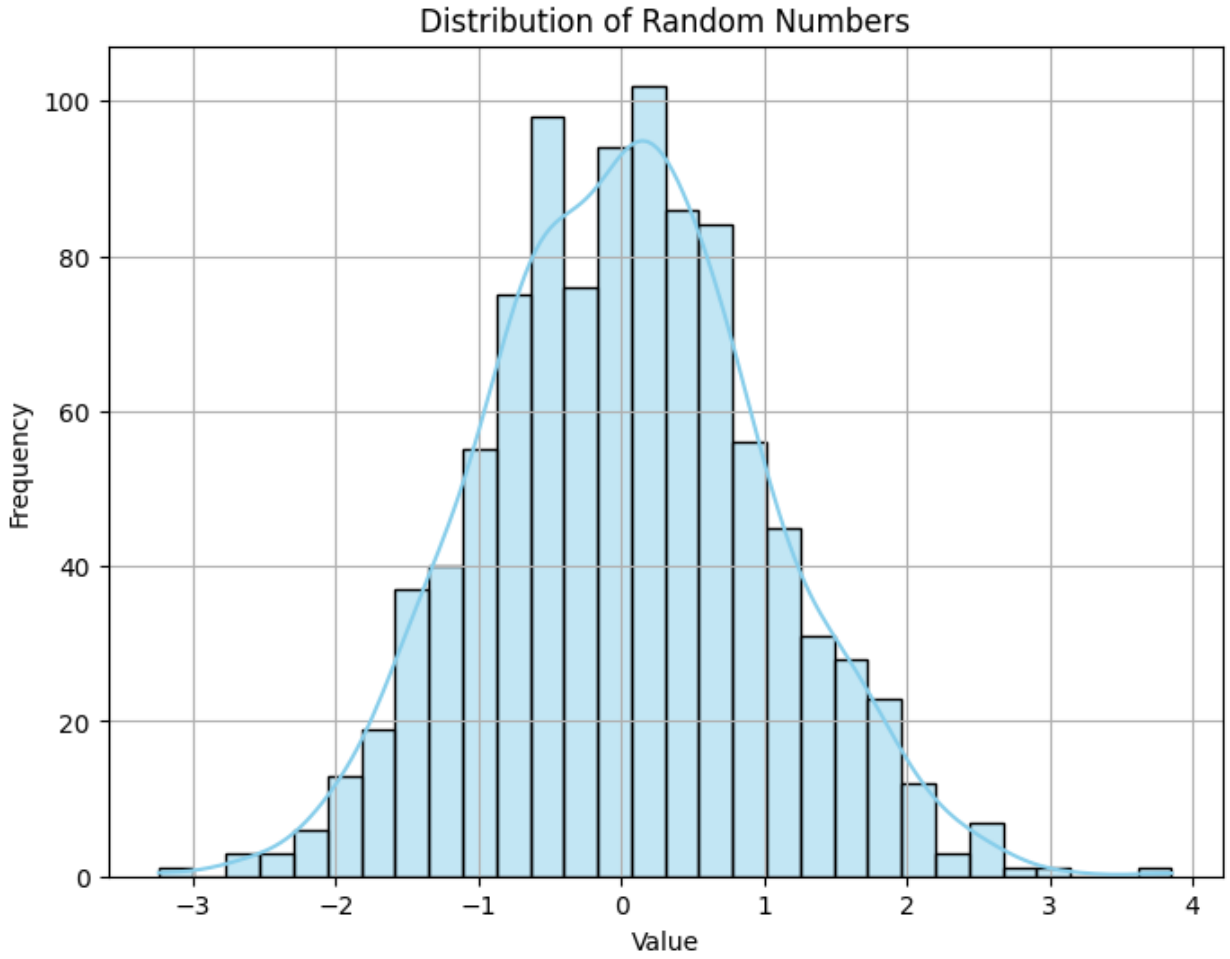

```
# SEABORN ASSIGNMENT:-
# 1. Create a scatter plot to visualize the relationship between two
variables, by generating a synthetic dataset.

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

# Set seed for reproducibility
np.random.seed(42)

# Number of samples
n_samples = 100

# Generate synthetic data
x = np.linspace(0, 10, n_samples)
y = 2 * x + 1 + np.random.normal(scale=2, size=n_samples)  # Linear
relationship with noise

# Create a DataFrame
df = pd.DataFrame({'X': x, 'Y': y})
# Create a scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='X', y='Y', alpha=0.7)
```

```
plt.title('Scatter Plot of X vs. Y')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)
plt.show()
```




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


#Ans:-
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Set seed for reproducibility
np.random.seed(42)
```

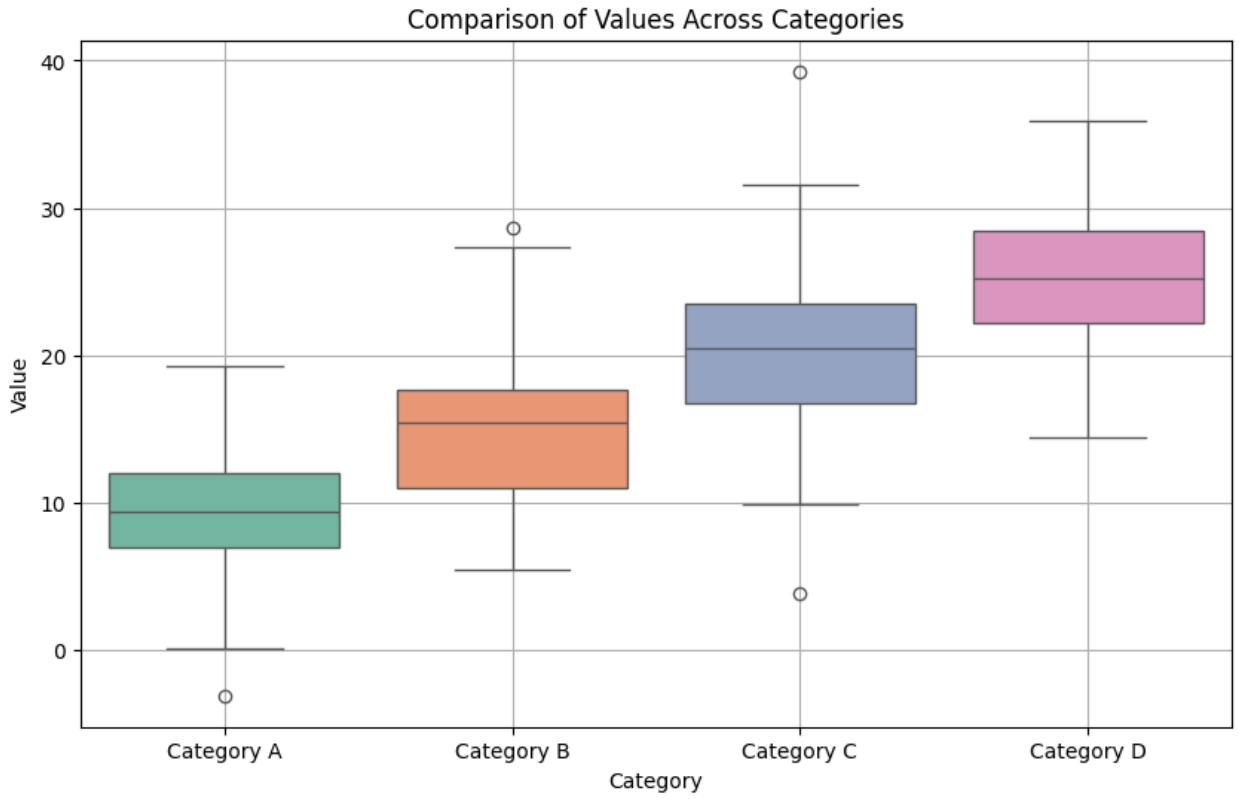
```
# Generate random numbers
data = np.random.normal(loc=0, scale=1, size=1000)  # Mean=0, Std=1

# Create a DataFrame
import pandas as pd
df = pd.DataFrame({'Value': data})
# Create a distribution plot
plt.figure(figsize=(8, 6))
sns.histplot(df['Value'], kde=True, bins=30, color='skyblue')
plt.title('Distribution of Random Numbers')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```




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

```python
#Ans:-
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Set seed for reproducibility
np.random.seed(42)

# Generate synthetic data
categories = ['Category A', 'Category B', 'Category C', 'Category D']
n_samples_per_category = 100

data = {
    'Category': np.repeat(categories, n_samples_per_category),
    'Value': np.concatenate([
        np.random.normal(loc=10 + i * 5, scale=5,
size=n_samples_per_category)
        for i in range(len(categories))
    ])
}

# Create a DataFrame
df = pd.DataFrame(data)
# Create a box plot
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Category', y='Value', palette='Set2')
plt.title('Comparison of Values Across Categories')
plt.xlabel('Category')
plt.ylabel('Value')
plt.grid(True)
plt.show()
```

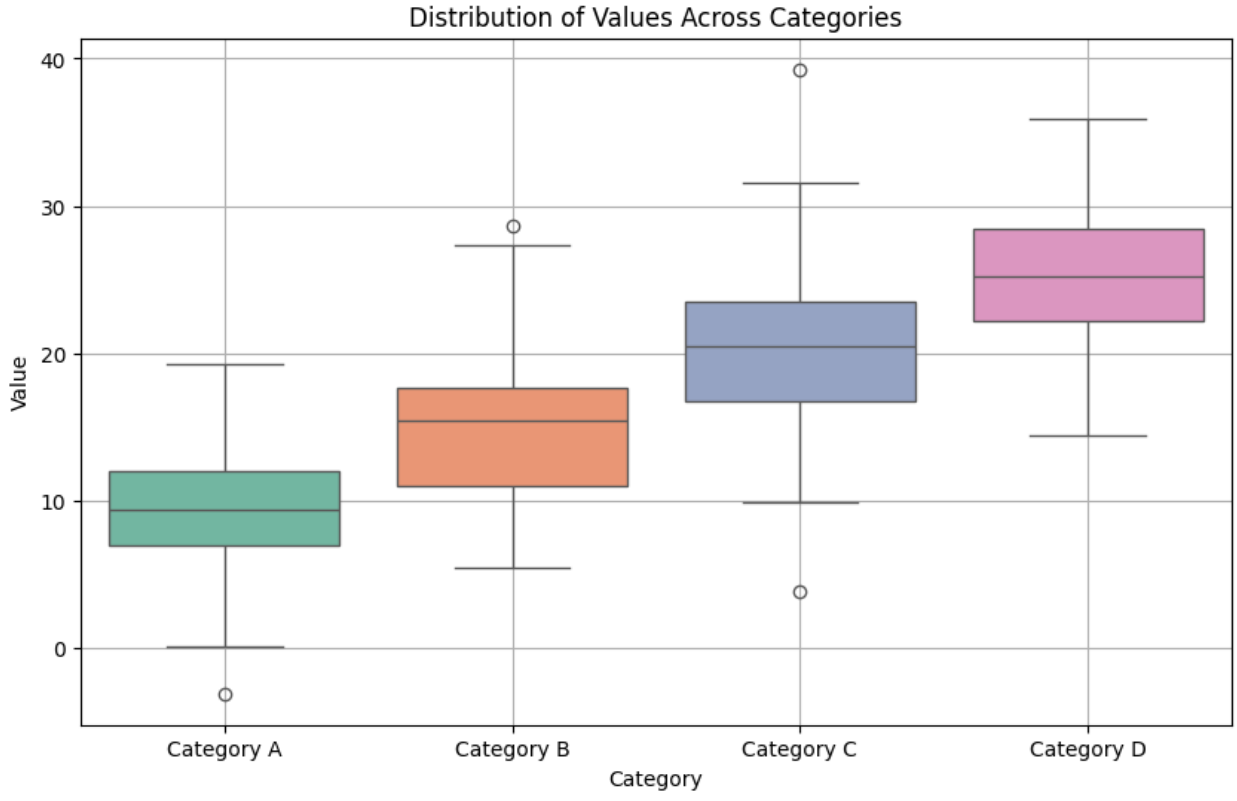
```
<ipython-input-17-d1c36ce88446>:28: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(data=df, x='Category', y='Value', palette='Set2')
```




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


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

# Set seed for reproducibility
np.random.seed(42)

# Generate synthetic data
categories = ['Category A', 'Category B', 'Category C', 'Category D']
n_samples_per_category = 100

# Generate random values for each category
data = {
    'Category': np.repeat(categories, n_samples_per_category),
    'Value': np.concatenate([
        np.random.normal(loc=10 + i * 5, scale=5,
size=n_samples_per_category)
        for i in range(len(categories))
```

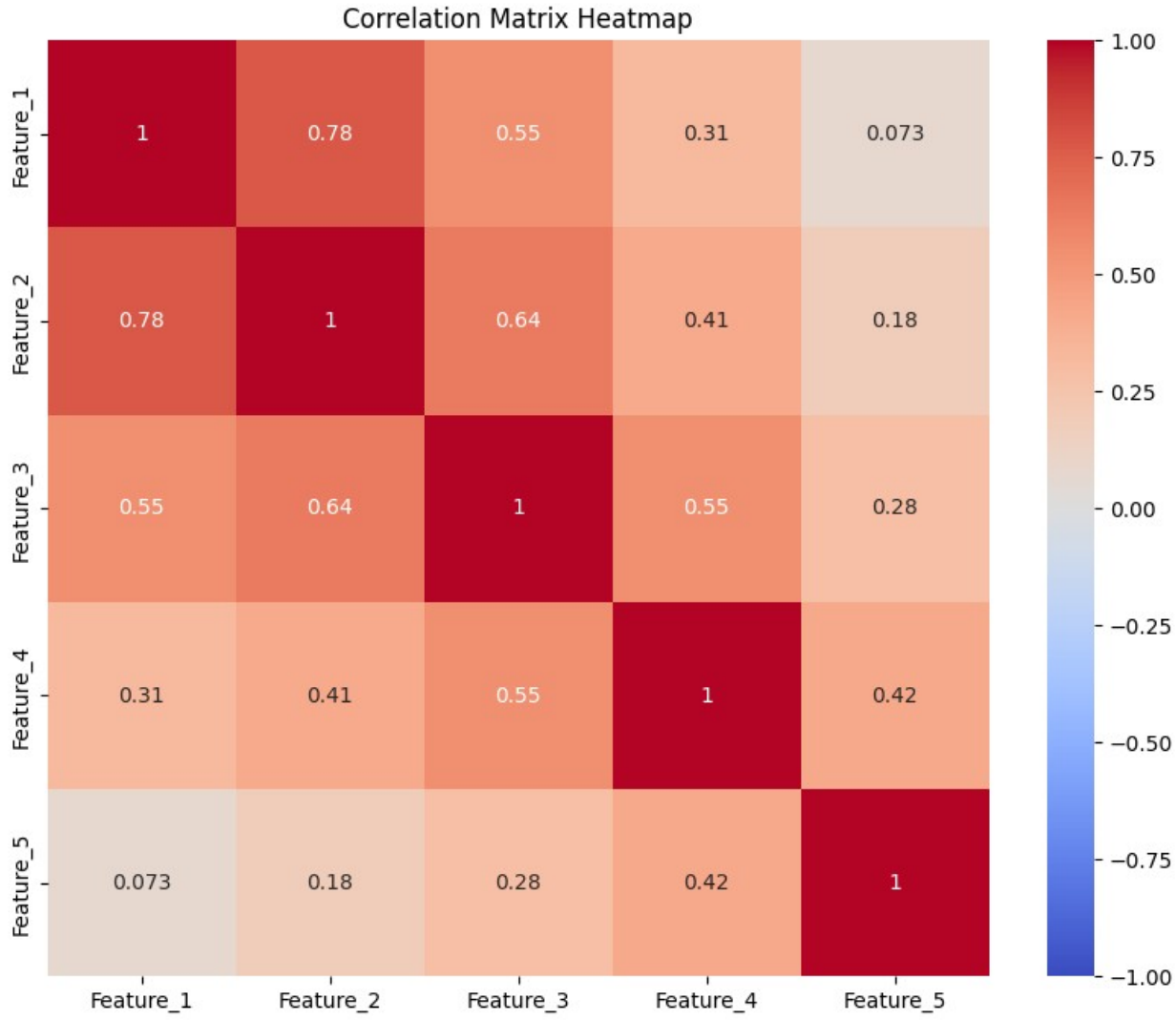
```
    ])
}

# Create a DataFrame
df = pd.DataFrame(data)
# Create a box plot
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Category', y='Value', palette='Set2')
plt.title('Distribution of Values Across Categories')
plt.xlabel('Category')
plt.ylabel('Value')
plt.grid(True)
plt.show()
```

```
<ipython-input-18-f46c40fc3061>:30: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(data=df, x='Category', y='Value', palette='Set2')
```




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

```python
# Ans:-
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Set seed for reproducibility
np.random.seed(42)

# Number of samples and features
n_samples = 100
n_features = 5

# Generate a random covariance matrix
cov_matrix = np.array([[1, 0.8, 0.6, 0.4, 0.2],
                       [0.8, 1, 0.7, 0.5, 0.3],
                       [0.6, 0.7, 1, 0.6, 0.4],
                       [0.4, 0.5, 0.6, 1, 0.5],
                       [0.2, 0.3, 0.4, 0.5, 1]])

# Generate data with the specified covariance matrix
mean = np.zeros(n_features)
data = np.random.multivariate_normal(mean, cov_matrix, size=n_samples)

# Create a DataFrame
df = pd.DataFrame(data, columns=[f'Feature_{i+1}' for i in
range(n_features)])
# Compute the correlation matrix
corr_matrix = df.corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1,
center=0)
plt.title('Correlation Matrix Heatmap')
plt.show()
```




```
# PLOTLY ASSIGNMENT:-

# 1.Using the given dataset, to generate a 3D scatter plot to
visualize the distribution of data points in a threedimensional 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)


#Ans:-
import numpy as np
import pandas as pd
```

```
import plotly.express as px

# Set seed for reproducibility
np.random.seed(30)

# Generate synthetic data
data = {
    'X': np.random.uniform(-10, 10, 300),
    'Y': np.random.uniform(-10, 10, 300),
    'Z': np.random.uniform(-10, 10, 300)
}

# Create a DataFrame
df = pd.DataFrame(data)

# Create a 3D scatter plot
fig = px.scatter_3d(df, x='X', y='Y', z='Z', title='3D Scatter Plot of
Data Points')

# Show the plot
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❯
# 1.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)


#Ans:-
import numpy as np
import pandas as pd
import plotly.express as px
```

```
# Set seed for reproducibility
np.random.seed(15)

# Generate synthetic student grades data
data_grades = {
    'Grade': np.random.choice(['A', 'B', 'C', 'D', 'F'], 200),
    'Score': np.random.randint(50, 100, 200)
}

# Create a DataFrame
df_grades = pd.DataFrame(data_grades)

# Create a violin plot
fig_grades = px.violin(df_grades, y='Score', x='Grade', box=True,
points='all', title='Distribution of Scores Across Different Grades')

# Show the plot
fig_grades.show()
import numpy as np
import pandas as pd
import plotly.express as px

# Set seed for reproducibility
np.random.seed(20)

# Generate synthetic sales data
data_sales = {
    '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)
}

# Create a DataFrame
df_sales = pd.DataFrame(data_sales)

# Pivot the DataFrame to get a matrix format for heatmap
heatmap_data = df_sales.pivot_table(index='Day', columns='Month',
values='Sales', aggfunc='mean')

# Create a heatmap
fig_sales = px.imshow(heatmap_data, title='Sales Heatmap Across
Different Months and Days', color_continuous_scale='Viridis')

# Show the plot
fig_sales.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)


#Ans:-
import numpy as np
import pandas as pd
import plotly.express as px

# Set seed for reproducibility
np.random.seed(20)

# Generate synthetic sales data
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)
}

# Create a DataFrame
df = pd.DataFrame(data)
# Pivot the DataFrame to get a matrix format for heatmap
heatmap_data = df.pivot_table(index='Day', columns='Month',
values='Sales', aggfunc='mean')
# Create a heatmap
fig = px.imshow(
    heatmap_data,
    title='Sales Heatmap Across Different Months and Days',
    color_continuous_scale='Viridis',
    labels={'color': 'Sales'},
    aspect='auto'
)

# Show the plot
fig.show()


# 4. Using the given x and y data, generate a 3D surface plot to
visualize the function Z = sin(sqrt(x^2+y^2)).
#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)


#Ans:-
import numpy as np
import pandas as pd
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))

data = {
    'X': x.flatten(),
    'Y': y.flatten(),
    'Z': z.flatten()
}
df = pd.DataFrame(data)
# Create the 3D surface plot
fig = go.Figure(data=[go.Surface(z=z, x=x, y=y)])

# Update the layout
fig.update_layout(
    title='3D Surface Plot of Z = sin(sqrt(x^2 + y^2))',
    scene=dict(
        xaxis_title='X',
        yaxis_title='Y',
        zaxis_title='Z'
    )
)

# Show the plot
fig.show()


# 5.Using the given dataset, create a bubble chart to represent each
country's population (y-axis), GDP (xaxis), 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)


#Ans:-
import numpy as np
import pandas as pd
import plotly.express as px

# Set seed for reproducibility
np.random.seed(25)

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

# Create a DataFrame
df = pd.DataFrame(data)
# Create the bubble chart
fig = px.scatter(
    df,
    x='GDP',
    y='Population',
    size='Population',
    color='Country',
    hover_name='Country',
    title='Bubble Chart of Country Population vs GDP',
    size_max=60  # Adjust the maximum bubble size for better
visualization
)

# Show the plot
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.


#Ans:-
import numpy as np
from bokeh.plotting import figure, show, output_notebook
```

```
# Display plots in the notebook
output_notebook()

# Generate data
x = np.linspace(0, 10, 500)
y = np.sin(x)

# Create a Bokeh figure
p = figure(title="Sine Wave", x_axis_label='x', y_axis_label='sin(x)')

# Add a line renderer with the data
p.line(x, y, legend_label='sin(x)', line_width=2)

# Show the plot
show(p)
```

```
"'use strict';\n(function(root) {\n  function now() {\n    return new
Date();\n  }\n\n  const force = true;\n\n  if (typeof
root._bokeh_onload_callbacks === \"undefined\" || force === true) {\n
root._bokeh_onload_callbacks = [];\n    root._bokeh_is_loading =
undefined;\n  }\n\n\n  if (typeof (root._bokeh_timeout) ===
\"undefined\" || force === true) {\n    root._bokeh_timeout =
Date.now() + 5000;\n    root._bokeh_failed_load = false;\n  }\n\n
const NB_LOAD_WARNING = {'data': {'text/html':\n     \"<div
style='background-color: #fdd'>\\n\"+\n     \"<p>\\n\"+\n
\"BokehJS does not appear to have successfully loaded. If loading
BokehJS from CDN, this \\n\"+\n     \"may be due to a slow or bad
network connection. Possible fixes:\\n\"+\n     \"</p>\\n\"+\n
\"<ul>\\n\"+\n     \"<li>re-rerun `output_notebook()` to attempt to
load from CDN again, or</li>\\n\"+\n     \"<li>use INLINE resources
instead, as so:</li>\\n\"+\n     \"</ul>\\n\"+\n     \"<code>\\n\"+\n
\"from bokeh.resources import INLINE\\n\"+\n
\"output_notebook(resources=INLINE)\\n\"+\n     \"</code>\\n\"+\n
\"</div>\"}};\n\n  function display_loaded(error = null) {\n    const
el = document.getElementById(null);\n    if (el != null) {\n
const html = (() => {\n        if (typeof root.Bokeh ===
\"undefined\") {\n          if (error == null) {\n            return
\"BokehJS is loading ...\";\n          } else {\n            return
\"BokehJS failed to load.\";\n          }\n        } else {\n
const prefix = `BokehJS ${root.Bokeh.version}`;\n          if (error
== null) {\n            return `${prefix} successfully loaded.`;\n
} else {\n            return `${prefix} <b>encountered errors</b>
while loading and may not function as expected.`;\n          }\n
}\n      })();\n      el.innerHTML = html;\n\n      if (error != null)
{\n        const wrapper = document.createElement(\"div\");\n
wrapper.style.overflow = \"auto\";\n        wrapper.style.height =
\"5em\";\n        wrapper.style.resize = \"vertical\";\n        const
content = document.createElement(\"div\");\n
content.style.fontFamily = \"monospace\";\n
content.style.whiteSpace = \"pre-wrap\";\n
```

```
content.style.backgroundColor = \"rgb(255, 221, 221)\";\n
content.textContent = error.stack ?? error.toString();\n
wrapper.append(content);\n        el.append(wrapper);\n      }\n    }
else if (Date.now() < root._bokeh_timeout) {\n      setTimeout(() =>
display_loaded(error), 100);\n    }\n  }\n\n  function run_callbacks()
{\n    try {\n
root._bokeh_onload_callbacks.forEach(function(callback) {\n        if
(callback != null)\n          callback();\n      });\n    } finally {\
n      delete root._bokeh_onload_callbacks\n    }\n
console.debug(\"Bokeh: all callbacks have finished\");\n  }\n\n
function load_libs(css_urls, js_urls, callback) {\n    if (css_urls ==
null) css_urls = [];\n    if (js_urls == null) js_urls = [];\n\n
root._bokeh_onload_callbacks.push(callback);\n    if
(root._bokeh_is_loading > 0) {\n      console.debug(\"Bokeh: BokehJS
is being loaded, scheduling callback at\", now());\n      return
null;\n    }\n    if (js_urls == null || js_urls.length === 0) {\n
run_callbacks();\n      return null;\n    }\n
console.debug(\"Bokeh: BokehJS not loaded, scheduling load and
callback at\", now());\n    root._bokeh_is_loading = css_urls.length +
js_urls.length;\n\n    function on_load() {\n
root._bokeh_is_loading--;\n      if (root._bokeh_is_loading === 0) {\n
console.debug(\"Bokeh: all BokehJS libraries/stylesheets loaded\");\n
run_callbacks()\n      }\n    }\n\n    function on_error(url) {\n
console.error(\"failed to load \" + url);\n    }\n\n    for (let i =
0; i < css_urls.length; i++) {\n      const url = css_urls[i];\n
const element = document.createElement(\"link\");\n
element.onload = on_load;\n      element.onerror = on_error.bind(null,
url);\n      element.rel = \"stylesheet\";\n      element.type =
\"text/css\";\n      element.href = url;\n      console.debug(\"Bokeh:
injecting link tag for BokehJS stylesheet: \", url);\n
document.body.appendChild(element);\n    }\n\n    for (let i = 0; i <
js_urls.length; i++) {\n      const url = js_urls[i];\n      const
element = document.createElement('script');\n      element.onload =
on_load;\n      element.onerror = on_error.bind(null, url);\n
element.async = false;\n      element.src = url;\n
console.debug(\"Bokeh: injecting script tag for BokehJS library: \",
url);\n      document.head.appendChild(element);\n    }\n  };\n\n
function inject_raw_css(css) {\n    const element =
document.createElement(\"style\");\n
element.appendChild(document.createTextNode(css));\n
document.body.appendChild(element);\n  }\n\n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-3.4.3.min.js\"];\n
const css_urls = [];\n\n  const inline_js = [    function(Bokeh) {\n
Bokeh.set_log_level(\"info\");\n    },\nfunction(Bokeh) {\n    }\
n  ];\n\n  function run_inline_js() {\n    if (root.Bokeh !==
```

```
undefined || force === true) {\n      try {\n            for (let i =
0; i < inline_js.length; i++) {\n      inline_js[i].call(root,
root.Bokeh);\n    }\n\n      } catch (error) {throw error;\n      }}
else if (Date.now() < root._bokeh_timeout) {\n
setTimeout(run_inline_js, 100);\n    } else if (!
root._bokeh_failed_load) {\n      console.log(\"Bokeh: BokehJS failed
to load within specified timeout.\");\n      root._bokeh_failed_load =
true;\n    } else if (force !== true) {\n      const cell = $
(document.getElementById(null)).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\n  }\n\
n  if (root._bokeh_is_loading === 0) {\n    console.debug(\"Bokeh:
BokehJS loaded, going straight to plotting\");\n    run_inline_js();\n
} else {\n    load_libs(css_urls, js_urls, function() {\n
console.debug(\"Bokeh: BokehJS plotting callback run at\", now());\n
run_inline_js();\n    });\n  }\n}(window));"

""

# 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.

#Ans:-
import numpy as np
import pandas as pd
from bokeh.plotting import figure, show, output_notebook
from bokeh.io import curdoc
from bokeh.models import ColumnDataSource

# Display plots in the notebook
output_notebook()

# Generate random data
np.random.seed(42)
n = 100
x = np.random.rand(n) * 10
y = np.random.rand(n) * 10
sizes = np.random.randint(10, 100, n)  # Marker sizes
colors = np.random.choice(['red', 'green', 'blue', 'orange',
'purple'], n)  # Marker colors

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

# Convert DataFrame to ColumnDataSource
source = ColumnDataSource(df)

# Create a Bokeh figure
p = figure(title="Scatter Plot with Variable Marker Sizes and Colors",
x_axis_label='x', y_axis_label='y')
```

```
# Add a scatter renderer with the data
p.scatter(x='x', y='y', size='sizes', color='colors',
legend_field='colors', source=source, fill_alpha=0.6)

# Show the plot
show(p)
```

```
"'use strict';\n(function(root) {\n  function now() {\n    return new
Date();\n  }\n\n  const force = true;\n\n  if (typeof
root._bokeh_onload_callbacks === \"undefined\" || force === true) {\n
root._bokeh_onload_callbacks = [];\n    root._bokeh_is_loading =
undefined;\n  }\n\n\n  if (typeof (root._bokeh_timeout) ===
\"undefined\" || force === true) {\n    root._bokeh_timeout =
Date.now() + 5000;\n    root._bokeh_failed_load = false;\n  }\n\n
const NB_LOAD_WARNING = {'data': {'text/html':\n     \"<div
style='background-color: #fdd'>\\n\"+\n     \"<p>\\n\"+\n
\"BokehJS does not appear to have successfully loaded. If loading
BokehJS from CDN, this \\n\"+\n     \"may be due to a slow or bad
network connection. Possible fixes:\\n\"+\n     \"</p>\\n\"+\n
\"<ul>\\n\"+\n     \"<li>re-rerun `output_notebook()` to attempt to
load from CDN again, or</li>\\n\"+\n     \"<li>use INLINE resources
instead, as so:</li>\\n\"+\n     \"</ul>\\n\"+\n     \"<code>\\n\"+\n
\"from bokeh.resources import INLINE\\n\"+\n
\"output_notebook(resources=INLINE)\\n\"+\n     \"</code>\\n\"+\n
\"</div>\"}};\n\n  function display_loaded(error = null) {\n    const
el = document.getElementById(null);\n    if (el != null) {\n
const html = (() => {\n        if (typeof root.Bokeh ===
\"undefined\") {\n          if (error == null) {\n            return
\"BokehJS is loading ...\";\n          } else {\n            return
\"BokehJS failed to load.\";\n          }\n        } else {\n
const prefix = `BokehJS ${root.Bokeh.version}`;\n          if (error
== null) {\n            return `${prefix} successfully loaded.`;\n
} else {\n            return `${prefix} <b>encountered errors</b>
while loading and may not function as expected.`;\n          }\n
}\n      })();\n      el.innerHTML = html;\n\n      if (error != null)
{\n        const wrapper = document.createElement(\"div\");\n
wrapper.style.overflow = \"auto\";\n        wrapper.style.height =
\"5em\";\n        wrapper.style.resize = \"vertical\";\n        const
content = document.createElement(\"div\");\n
content.style.fontFamily = \"monospace\";\n
content.style.whiteSpace = \"pre-wrap\";\n
content.style.backgroundColor = \"rgb(255, 221, 221)\";\n
content.textContent = error.stack ?? error.toString();\n
wrapper.append(content);\n        el.append(wrapper);\n      }\n    }
else if (Date.now() < root._bokeh_timeout) {\n      setTimeout(() =>
display_loaded(error), 100);\n    }\n  }\n\n  function run_callbacks()
{\n    try {\n
```

```
root._bokeh_onload_callbacks.forEach(function(callback) {\n        if
(callback != null)\n          callback();\n      });\n    } finally {\
n      delete root._bokeh_onload_callbacks\n    }\n
console.debug(\"Bokeh: all callbacks have finished\");\n  }\n\n
function load_libs(css_urls, js_urls, callback) {\n    if (css_urls ==
null) css_urls = [];\n    if (js_urls == null) js_urls = [];\n\n
root._bokeh_onload_callbacks.push(callback);\n    if
(root._bokeh_is_loading > 0) {\n      console.debug(\"Bokeh: BokehJS
is being loaded, scheduling callback at\", now());\n      return
null;\n    }\n    if (js_urls == null || js_urls.length === 0) {\n
run_callbacks();\n      return null;\n    }\n
console.debug(\"Bokeh: BokehJS not loaded, scheduling load and
callback at\", now());\n    root._bokeh_is_loading = css_urls.length +
js_urls.length;\n\n    function on_load() {\n
root._bokeh_is_loading--;\n      if (root._bokeh_is_loading === 0) {\n
console.debug(\"Bokeh: all BokehJS libraries/stylesheets loaded\");\n
run_callbacks()\n      }\n    }\n\n    function on_error(url) {\n
console.error(\"failed to load \" + url);\n    }\n\n    for (let i =
0; i < css_urls.length; i++) {\n      const url = css_urls[i];\n
const element = document.createElement(\"link\");\n
element.onload = on_load;\n      element.onerror = on_error.bind(null,
url);\n      element.rel = \"stylesheet\";\n      element.type =
\"text/css\";\n      element.href = url;\n      console.debug(\"Bokeh:
injecting link tag for BokehJS stylesheet: \", url);\n
document.body.appendChild(element);\n    }\n\n    for (let i = 0; i <
js_urls.length; i++) {\n      const url = js_urls[i];\n      const
element = document.createElement('script');\n      element.onload =
on_load;\n      element.onerror = on_error.bind(null, url);\n
element.async = false;\n      element.src = url;\n
console.debug(\"Bokeh: injecting script tag for BokehJS library: \",
url);\n      document.head.appendChild(element);\n    }\n  };\n\n
function inject_raw_css(css) {\n    const element =
document.createElement(\"style\");\n
element.appendChild(document.createTextNode(css));\n
document.body.appendChild(element);\n  }\n\n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-3.4.3.min.js\"];\n
const css_urls = [];\n\n  const inline_js = [    function(Bokeh) {\n
Bokeh.set_log_level(\"info\");\n    },\nfunction(Bokeh) {\n    }\
n  ];\n\n  function run_inline_js() {\n    if (root.Bokeh !==
undefined || force === true) {\n      try {\n            for (let i =
0; i < inline_js.length; i++) {\n      inline_js[i].call(root,
root.Bokeh);\n    }\n\n      } catch (error) {throw error;\n      }}
else if (Date.now() < root._bokeh_timeout) {\n
setTimeout(run_inline_js, 100);\n    } else if (!
root._bokeh_failed_load) {\n      console.log(\"Bokeh: BokehJS failed
```

```
to load within specified timeout.\");\n      root._bokeh_failed_load =
true;\n    } else if (force !== true) {\n      const cell = $
(document.getElementById(null)).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\n  }\n\
n  if (root._bokeh_is_loading === 0) {\n    console.debug(\"Bokeh:
BokehJS loaded, going straight to plotting\");\n    run_inline_js();\n
} else {\n    load_libs(css_urls, js_urls, function() {\n
console.debug(\"Bokeh: BokehJS plotting callback run at\", now());\n
run_inline_js();\n    });\n  }\n}(window));"

""

# 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]

#Ans:-
from bokeh.plotting import figure, show
from bokeh.io import output_notebook

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

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

# Create a figure
p = figure(x_range=fruits, height=350, title="Fruit Counts",
           toolbar_location=None, tools="", x_axis_label="Fruits",
y_axis_label="Count")

# Add bars to the figure
p.vbar(x=fruits, top=counts, width=0.9)

# Add labels
p.xgrid.grid_line_color = None
p.y_range.start = 0
p.axis.minor_tick_line_color = None
p.outline_line_color = None

# Show the plot
show(p)

"'use strict';\n(function(root) {\n  function now() {\n    return new
Date();\n  }\n\n  const force = true;\n\n  if (typeof
root._bokeh_onload_callbacks === \"undefined\" || force === true) {\n
root._bokeh_onload_callbacks = [];\n    root._bokeh_is_loading =
undefined;\n  }\n\n\n  if (typeof (root._bokeh_timeout) ===
```

```
\"undefined\" || force === true) {\n    root._bokeh_timeout =
Date.now() + 5000;\n    root._bokeh_failed_load = false;\n  }\n\n
const NB_LOAD_WARNING = {'data': {'text/html':\n     \"<div
style='background-color: #fdd'>\\n\"+\n     \"<p>\\n\"+\n
\"BokehJS does not appear to have successfully loaded. If loading
BokehJS from CDN, this \\n\"+\n     \"may be due to a slow or bad
network connection. Possible fixes:\\n\"+\n     \"</p>\\n\"+\n
\"<ul>\\n\"+\n     \"<li>re-rerun `output_notebook()` to attempt to
load from CDN again, or</li>\\n\"+\n     \"<li>use INLINE resources
instead, as so:</li>\\n\"+\n     \"</ul>\\n\"+\n     \"<code>\\n\"+\n
\"from bokeh.resources import INLINE\\n\"+\n
\"output_notebook(resources=INLINE)\\n\"+\n     \"</code>\\n\"+\n
\"</div>\"}};\n\n  function display_loaded(error = null) {\n    const
el = document.getElementById(null);\n    if (el != null) {\n
const html = (() => {\n        if (typeof root.Bokeh ===
\"undefined\") {\n          if (error == null) {\n            return
\"BokehJS is loading ...\";\n          } else {\n            return
\"BokehJS failed to load.\";\n          }\n        } else {\n
const prefix = `BokehJS ${root.Bokeh.version}`;\n          if (error
== null) {\n            return `${prefix} successfully loaded.`;\n
} else {\n            return `${prefix} <b>encountered errors</b>
while loading and may not function as expected.`;\n          }\n
}\n      })();\n      el.innerHTML = html;\n\n      if (error != null)
{\n        const wrapper = document.createElement(\"div\");\n
wrapper.style.overflow = \"auto\";\n        wrapper.style.height =
\"5em\";\n        wrapper.style.resize = \"vertical\";\n        const
content = document.createElement(\"div\");\n
content.style.fontFamily = \"monospace\";\n
content.style.whiteSpace = \"pre-wrap\";\n
content.style.backgroundColor = \"rgb(255, 221, 221)\";\n
content.textContent = error.stack ?? error.toString();\n
wrapper.append(content);\n        el.append(wrapper);\n      }\n    }
else if (Date.now() < root._bokeh_timeout) {\n      setTimeout(() =>
display_loaded(error), 100);\n    }\n  }\n\n  function run_callbacks()
{\n    try {\n
root._bokeh_onload_callbacks.forEach(function(callback) {\n        if
(callback != null)\n          callback();\n      });\n    } finally {\
n      delete root._bokeh_onload_callbacks\n    }\n
console.debug(\"Bokeh: all callbacks have finished\");\n  }\n\n
function load_libs(css_urls, js_urls, callback) {\n    if (css_urls ==
null) css_urls = [];\n    if (js_urls == null) js_urls = [];\n\n
root._bokeh_onload_callbacks.push(callback);\n    if
(root._bokeh_is_loading > 0) {\n      console.debug(\"Bokeh: BokehJS
is being loaded, scheduling callback at\", now());\n      return
null;\n    }\n    if (js_urls == null || js_urls.length === 0) {\n
run_callbacks();\n      return null;\n    }\n
console.debug(\"Bokeh: BokehJS not loaded, scheduling load and
callback at\", now());\n    root._bokeh_is_loading = css_urls.length +
js_urls.length;\n\n    function on_load() {\n
```

```
root._bokeh_is_loading--;\n      if (root._bokeh_is_loading === 0) {\n
console.debug(\"Bokeh: all BokehJS libraries/stylesheets loaded\");\n
run_callbacks()\n      }\n    }\n\n    function on_error(url) {\n
console.error(\"failed to load \" + url);\n    }\n\n    for (let i =
0; i < css_urls.length; i++) {\n      const url = css_urls[i];\n
const element = document.createElement(\"link\");\n
element.onload = on_load;\n      element.onerror = on_error.bind(null,
url);\n      element.rel = \"stylesheet\";\n      element.type =
\"text/css\";\n      element.href = url;\n      console.debug(\"Bokeh:
injecting link tag for BokehJS stylesheet: \", url);\n
document.body.appendChild(element);\n    }\n\n    for (let i = 0; i <
js_urls.length; i++) {\n      const url = js_urls[i];\n      const
element = document.createElement('script');\n      element.onload =
on_load;\n      element.onerror = on_error.bind(null, url);\n
element.async = false;\n      element.src = url;\n
console.debug(\"Bokeh: injecting script tag for BokehJS library: \",
url);\n      document.head.appendChild(element);\n    }\n  };\n\n
function inject_raw_css(css) {\n    const element =
document.createElement(\"style\");\n
element.appendChild(document.createTextNode(css));\n
document.body.appendChild(element);\n  }\n\n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-3.4.3.min.js\"];\n
const css_urls = [];\n\n  const inline_js = [    function(Bokeh) {\n
Bokeh.set_log_level(\"info\");\n    },\nfunction(Bokeh) {\n    }\
n  ];\n\n  function run_inline_js() {\n    if (root.Bokeh !==
undefined || force === true) {\n      try {\n            for (let i =
0; i < inline_js.length; i++) {\n      inline_js[i].call(root,
root.Bokeh);\n    }\n\n      } catch (error) {throw error;\n      }}
else if (Date.now() < root._bokeh_timeout) {\n
setTimeout(run_inline_js, 100);\n    } else if (!
root._bokeh_failed_load) {\n      console.log(\"Bokeh: BokehJS failed
to load within specified timeout.\");\n      root._bokeh_failed_load =
true;\n    } else if (force !== true) {\n      const cell = $
(document.getElementById(null)).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\n  }\n\
n  if (root._bokeh_is_loading === 0) {\n    console.debug(\"Bokeh:
BokehJS loaded, going straight to plotting\");\n    run_inline_js();\n
} else {\n    load_libs(css_urls, js_urls, function() {\n
console.debug(\"Bokeh: BokehJS plotting callback run at\", now());\n
run_inline_js();\n    });\n  }\n}(window));"

""

# 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)

#Ans:-
import numpy as np
from bokeh.plotting import figure, show
from bokeh.io import output_notebook

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

# Generate random data
data_hist = np.random.randn(1000)

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

# Create a figure
p = figure(title="Histogram of Random Data", x_axis_label='Value',
y_axis_label='Frequency',
           tools="save", background_fill_color="#fafafa")

# Add a quad glyph for the histogram
p.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:],
fill_color="navy", line_color="white", alpha=0.5)

# Show the plot
show(p)

"'use strict';\n(function(root) {\n  function now() {\n    return new
Date();\n  }\n\n  const force = true;\n\n  if (typeof
root._bokeh_onload_callbacks === \"undefined\" || force === true) {\n
root._bokeh_onload_callbacks = [];\n    root._bokeh_is_loading =
undefined;\n  }\n\n\n  if (typeof (root._bokeh_timeout) ===
\"undefined\" || force === true) {\n    root._bokeh_timeout =
Date.now() + 5000;\n    root._bokeh_failed_load = false;\n  }\n\n
const NB_LOAD_WARNING = {'data': {'text/html':\n     \"<div
style='background-color: #fdd'>\\n\"+\n     \"<p>\\n\"+\n
\"BokehJS does not appear to have successfully loaded. If loading
BokehJS from CDN, this \\n\"+\n     \"may be due to a slow or bad
network connection. Possible fixes:\\n\"+\n     \"</p>\\n\"+\n
\"<ul>\\n\"+\n     \"<li>re-rerun `output_notebook()` to attempt to
load from CDN again, or</li>\\n\"+\n     \"<li>use INLINE resources
instead, as so:</li>\\n\"+\n     \"</ul>\\n\"+\n     \"<code>\\n\"+\n
\"from bokeh.resources import INLINE\\n\"+\n
\"output_notebook(resources=INLINE)\\n\"+\n     \"</code>\\n\"+\n
\"</div>\"}};\n\n  function display_loaded(error = null) {\n    const
el = document.getElementById(null);\n    if (el != null) {\n
const html = (() => {\n        if (typeof root.Bokeh ===
\"undefined\") {\n          if (error == null) {\n            return
```

```
\"BokehJS is loading ...\";\n          } else {\n            return
\"BokehJS failed to load.\";\n          }\n        } else {\n
const prefix = `BokehJS ${root.Bokeh.version}`;\n          if (error
== null) {\n            return `${prefix} successfully loaded.`;\n
} else {\n            return `${prefix} <b>encountered errors</b>
while loading and may not function as expected.`;\n          }\n
}\n      })();\n      el.innerHTML = html;\n\n      if (error != null)
{\n        const wrapper = document.createElement(\"div\");\n
wrapper.style.overflow = \"auto\";\n        wrapper.style.height =
\"5em\";\n        wrapper.style.resize = \"vertical\";\n        const
content = document.createElement(\"div\");\n
content.style.fontFamily = \"monospace\";\n
content.style.whiteSpace = \"pre-wrap\";\n
content.style.backgroundColor = \"rgb(255, 221, 221)\";\n
content.textContent = error.stack ?? error.toString();\n
wrapper.append(content);\n        el.append(wrapper);\n      }\n    }
else if (Date.now() < root._bokeh_timeout) {\n      setTimeout(() =>
display_loaded(error), 100);\n    }\n  }\n\n  function run_callbacks()
{\n    try {\n
root._bokeh_onload_callbacks.forEach(function(callback) {\n        if
(callback != null)\n          callback();\n      });\n    } finally {\
n      delete root._bokeh_onload_callbacks\n    }\n
console.debug(\"Bokeh: all callbacks have finished\");\n  }\n\n
function load_libs(css_urls, js_urls, callback) {\n    if (css_urls ==
null) css_urls = [];\n    if (js_urls == null) js_urls = [];\n\n
root._bokeh_onload_callbacks.push(callback);\n    if
(root._bokeh_is_loading > 0) {\n      console.debug(\"Bokeh: BokehJS
is being loaded, scheduling callback at\", now());\n      return
null;\n    }\n    if (js_urls == null || js_urls.length === 0) {\n
run_callbacks();\n      return null;\n    }\n
console.debug(\"Bokeh: BokehJS not loaded, scheduling load and
callback at\", now());\n    root._bokeh_is_loading = css_urls.length +
js_urls.length;\n\n    function on_load() {\n
root._bokeh_is_loading--;\n      if (root._bokeh_is_loading === 0) {\n
console.debug(\"Bokeh: all BokehJS libraries/stylesheets loaded\");\n
run_callbacks()\n      }\n    }\n\n    function on_error(url) {\n
console.error(\"failed to load \" + url);\n    }\n\n    for (let i =
0; i < css_urls.length; i++) {\n      const url = css_urls[i];\n
const element = document.createElement(\"link\");\n
element.onload = on_load;\n      element.onerror = on_error.bind(null,
url);\n      element.rel = \"stylesheet\";\n      element.type =
\"text/css\";\n      element.href = url;\n      console.debug(\"Bokeh:
injecting link tag for BokehJS stylesheet: \", url);\n
document.body.appendChild(element);\n    }\n\n    for (let i = 0; i <
js_urls.length; i++) {\n      const url = js_urls[i];\n      const
element = document.createElement('script');\n      element.onload =
on_load;\n      element.onerror = on_error.bind(null, url);\n
element.async = false;\n      element.src = url;\n
console.debug(\"Bokeh: injecting script tag for BokehJS library: \",
```

```
url);\n      document.head.appendChild(element);\n    }\n  };\n\n
function inject_raw_css(css) {\n    const element =
document.createElement(\"style\");\n
element.appendChild(document.createTextNode(css));\n
document.body.appendChild(element);\n  }\n\n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-3.4.3.min.js\"];\n
const css_urls = [];\n\n  const inline_js = [    function(Bokeh) {\n
Bokeh.set_log_level(\"info\");\n    },\nfunction(Bokeh) {\n    }\
n  ];\n\n  function run_inline_js() {\n    if (root.Bokeh !==
undefined || force === true) {\n      try {\n            for (let i =
0; i < inline_js.length; i++) {\n      inline_js[i].call(root,
root.Bokeh);\n    }\n\n      } catch (error) {throw error;\n      }}
else if (Date.now() < root._bokeh_timeout) {\n
setTimeout(run_inline_js, 100);\n    } else if (!
root._bokeh_failed_load) {\n      console.log(\"Bokeh: BokehJS failed
to load within specified timeout.\");\n      root._bokeh_failed_load =
true;\n    } else if (force !== true) {\n      const cell = $
(document.getElementById(null)).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\n  }\n\
n  if (root._bokeh_is_loading === 0) {\n    console.debug(\"Bokeh:
BokehJS loaded, going straight to plotting\");\n    run_inline_js();\n
} else {\n    load_libs(css_urls, js_urls, function() {\n
console.debug(\"Bokeh: BokehJS plotting callback run at\", now());\n
run_inline_js();\n    });\n  }\n}(window));"

""

# 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)

#Ans:-
import numpy as np
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.palettes import Viridis256

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

# Generate data
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)

# Create a figure
p = figure(title="Heatmap", x_axis_label='X', y_axis_label='Y',
           x_range=(0, 1), y_range=(0, 1),
           toolbar_location=None, tools="",
           width=400, height=400)

# Add image glyph
p.image(image=[data_heatmap], x=0, y=0, dw=1, dh=1,
palette=Viridis256)

# Customize plot appearance
p.xgrid.grid_line_color = None
p.ygrid.grid_line_color = None
p.axis.axis_line_color = None
p.axis.major_label_orientation = "horizontal"

# Show the plot
show(p)
```

```
"'use strict';\n(function(root) {\n  function now() {\n    return new
Date();\n  }\n\n  const force = true;\n\n  if (typeof
root._bokeh_onload_callbacks === \"undefined\" || force === true) {\n
root._bokeh_onload_callbacks = [];\n    root._bokeh_is_loading =
undefined;\n  }\n\n\n  if (typeof (root._bokeh_timeout) ===
\"undefined\" || force === true) {\n    root._bokeh_timeout =
Date.now() + 5000;\n    root._bokeh_failed_load = false;\n  }\n\n
const NB_LOAD_WARNING = {'data': {'text/html':\n     \"<div
style='background-color: #fdd'>\\n\"+\n     \"<p>\\n\"+\n
\"BokehJS does not appear to have successfully loaded. If loading
BokehJS from CDN, this \\n\"+\n     \"may be due to a slow or bad
network connection. Possible fixes:\\n\"+\n     \"</p>\\n\"+\n
\"<ul>\\n\"+\n     \"<li>re-rerun `output_notebook()` to attempt to
load from CDN again, or</li>\\n\"+\n     \"<li>use INLINE resources
instead, as so:</li>\\n\"+\n     \"</ul>\\n\"+\n     \"<code>\\n\"+\n
\"from bokeh.resources import INLINE\\n\"+\n
\"output_notebook(resources=INLINE)\\n\"+\n     \"</code>\\n\"+\n
\"</div>\"}};\n\n  function display_loaded(error = null) {\n    const
el = document.getElementById(null);\n    if (el != null) {\n
const html = (() => {\n        if (typeof root.Bokeh ===
\"undefined\") {\n          if (error == null) {\n            return
\"BokehJS is loading ...\";\n          } else {\n            return
\"BokehJS failed to load.\";\n          }\n        } else {\n
const prefix = `BokehJS ${root.Bokeh.version}`;\n          if (error
== null) {\n            return `${prefix} successfully loaded.`;\n
} else {\n            return `${prefix} <b>encountered errors</b>
while loading and may not function as expected.`;\n          }\n
}\n      })();\n      el.innerHTML = html;\n\n      if (error != null)
```

```
{\n        const wrapper = document.createElement(\"div\");\n
wrapper.style.overflow = \"auto\";\n        wrapper.style.height =
\"5em\";\n        wrapper.style.resize = \"vertical\";\n        const
content = document.createElement(\"div\");\n
content.style.fontFamily = \"monospace\";\n
content.style.whiteSpace = \"pre-wrap\";\n
content.style.backgroundColor = \"rgb(255, 221, 221)\";\n
content.textContent = error.stack ?? error.toString();\n
wrapper.append(content);\n        el.append(wrapper);\n      }\n    }
else if (Date.now() < root._bokeh_timeout) {\n      setTimeout(() =>
display_loaded(error), 100);\n    }\n  }\n\n  function run_callbacks()
{\n    try {\n
root._bokeh_onload_callbacks.forEach(function(callback) {\n        if
(callback != null)\n          callback();\n      });\n    } finally {\
n      delete root._bokeh_onload_callbacks\n    }\n
console.debug(\"Bokeh: all callbacks have finished\");\n  }\n\n
function load_libs(css_urls, js_urls, callback) {\n    if (css_urls ==
null) css_urls = [];\n    if (js_urls == null) js_urls = [];\n\n
root._bokeh_onload_callbacks.push(callback);\n    if
(root._bokeh_is_loading > 0) {\n      console.debug(\"Bokeh: BokehJS
is being loaded, scheduling callback at\", now());\n      return
null;\n    }\n    if (js_urls == null || js_urls.length === 0) {\n
run_callbacks();\n      return null;\n    }\n
console.debug(\"Bokeh: BokehJS not loaded, scheduling load and
callback at\", now());\n    root._bokeh_is_loading = css_urls.length +
js_urls.length;\n\n    function on_load() {\n
root._bokeh_is_loading--;\n      if (root._bokeh_is_loading === 0) {\n
console.debug(\"Bokeh: all BokehJS libraries/stylesheets loaded\");\n
run_callbacks()\n      }\n    }\n\n    function on_error(url) {\n
console.error(\"failed to load \" + url);\n    }\n\n    for (let i =
0; i < css_urls.length; i++) {\n      const url = css_urls[i];\n
const element = document.createElement(\"link\");\n
element.onload = on_load;\n      element.onerror = on_error.bind(null,
url);\n      element.rel = \"stylesheet\";\n      element.type =
\"text/css\";\n      element.href = url;\n      console.debug(\"Bokeh:
injecting link tag for BokehJS stylesheet: \", url);\n
document.body.appendChild(element);\n    }\n\n    for (let i = 0; i <
js_urls.length; i++) {\n      const url = js_urls[i];\n      const
element = document.createElement('script');\n      element.onload =
on_load;\n      element.onerror = on_error.bind(null, url);\n
element.async = false;\n      element.src = url;\n
console.debug(\"Bokeh: injecting script tag for BokehJS library: \",
url);\n      document.head.appendChild(element);\n    }\n  };\n\n
function inject_raw_css(css) {\n    const element =
document.createElement(\"style\");\n
element.appendChild(document.createTextNode(css));\n
document.body.appendChild(element);\n  }\n\n  const js_urls =
[\"https://cdn.bokeh.org/bokeh/release/bokeh-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-gl-3.4.3.min.js\",
```

```
\"https://cdn.bokeh.org/bokeh/release/bokeh-widgets-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-tables-3.4.3.min.js\",
\"https://cdn.bokeh.org/bokeh/release/bokeh-mathjax-3.4.3.min.js\"];\n
const css_urls = [];\n\n  const inline_js = [    function(Bokeh) {\n
Bokeh.set_log_level(\"info\");\n    },\nfunction(Bokeh) {\n    }\
n  ];\n\n  function run_inline_js() {\n    if (root.Bokeh !==
undefined || force === true) {\n      try {\n            for (let i =
0; i < inline_js.length; i++) {\n      inline_js[i].call(root,
root.Bokeh);\n    }\n\n      } catch (error) {throw error;\n      }}
else if (Date.now() < root._bokeh_timeout) {\n
setTimeout(run_inline_js, 100);\n    } else if (!
root._bokeh_failed_load) {\n      console.log(\"Bokeh: BokehJS failed
to load within specified timeout.\");\n      root._bokeh_failed_load =
true;\n    } else if (force !== true) {\n      const cell = $
(document.getElementById(null)).parents('.cell').data().cell;\n
cell.output_area.append_execute_result(NB_LOAD_WARNING)\n    }\n  }\n\
n  if (root._bokeh_is_loading === 0) {\n    console.debug(\"Bokeh:
BokehJS loaded, going straight to plotting\");\n    run_inline_js();\n
} else {\n    load_libs(css_urls, js_urls, function() {\n
console.debug(\"Bokeh: BokehJS plotting callback run at\", now());\n
run_inline_js();\n    });\n  }\n}(window));"

""
```