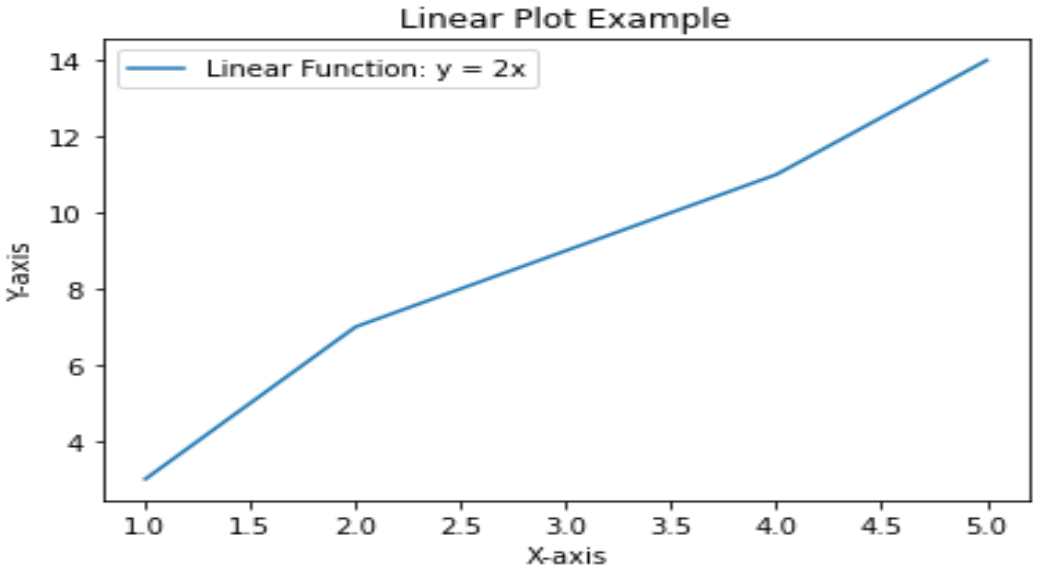
PROGRAM-6

6a) Write a Python program to illustrate Linear Plotting using Matplotlib

```
import matplotlib.pyplot as plt
def linear_plot():
# Sample data
    x = [1, 2, 3, 4, 5]
    y = [3, 7, 9, 11, 14]
    # Plotting the data
    plt.plot(x, y, label='Linear Function: y = 2x')
    # Adding labels and title
    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')
    plt.title('Linear Plot Example')
    plt.legend()
    plt.show()
    # Call the function to generate the plot
linear_plot()
```

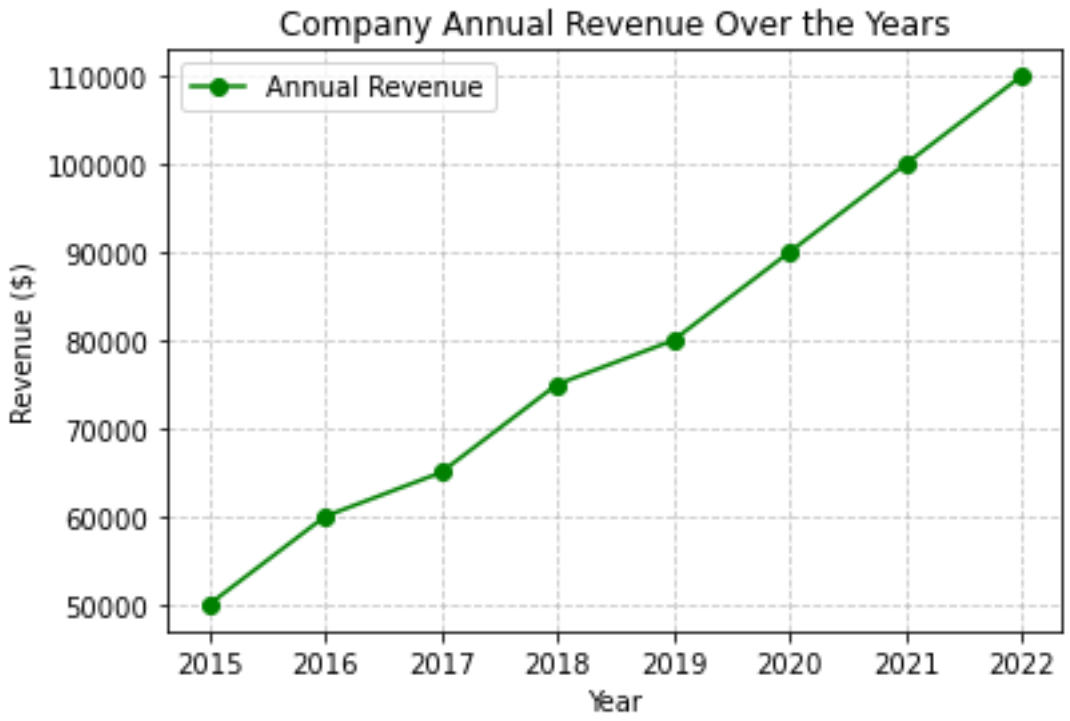
OUTPUT:



OR

```
import matplotlib.pyplot as plt
# Data for the linear plot
years = [2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022]
revenue = [50000, 60000, 65000, 75000, 80000, 90000, 100000, 110000]
# Create a linear plot
plt.plot(years,revenue, marker='o', color='green', linestyle='-', label='Annual Revenue')
# Add labels and a title
plt.xlabel('Year')
plt.ylabel('Revenue ($)')
plt.title('Company Annual Revenue Over the Years')
# Add a grid
plt.grid(True, linestyle='--', alpha=0.7)
# Add a legend
plt.legend()
# Display the plot
plt.show()
```

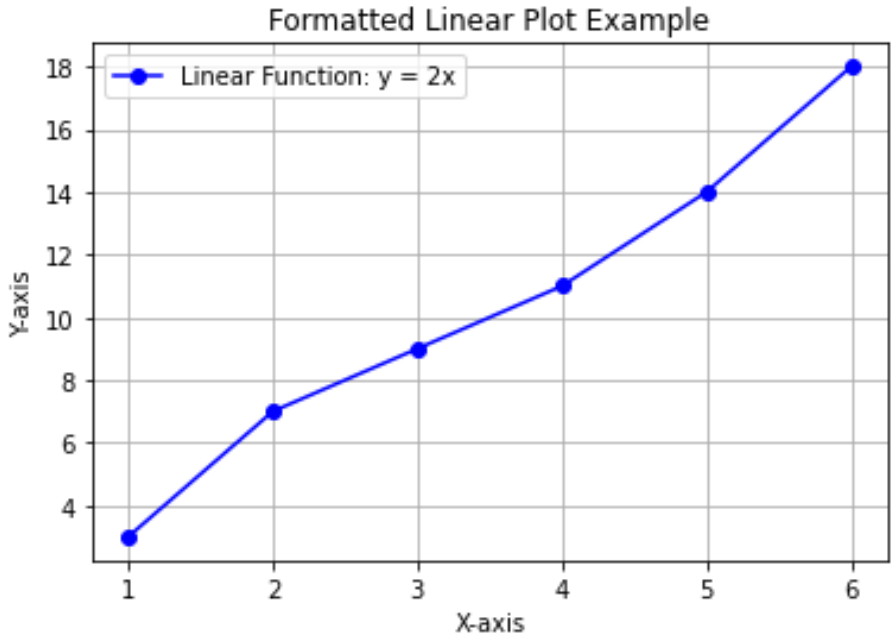
OUTPUT:



6b) Write a Python program to illustrate liner plotting with line formatting using Matplotlib

```
import matplotlib.pyplot as plt
def formatted_linear_plot():
# Sample data
    x = [1, 2, 3, 4, 5, 6]
    y = [3, 7, 9, 11, 14, 18]
    plt.plot(x, y, marker='o', linestyle='-', color='b', label='Linear Function: y = 2x')
    # Adding labels and title
    plt.xlabel('X-axis')
    plt.ylabel('Y-axis')
    plt.title('Formatted Linear Plot Example')
    plt.legend()
    plt.grid(True) # Add a grid for better readability
    plt.show()
# Call the function to generate the formatted linear plot
formatted_linear_plot()
```

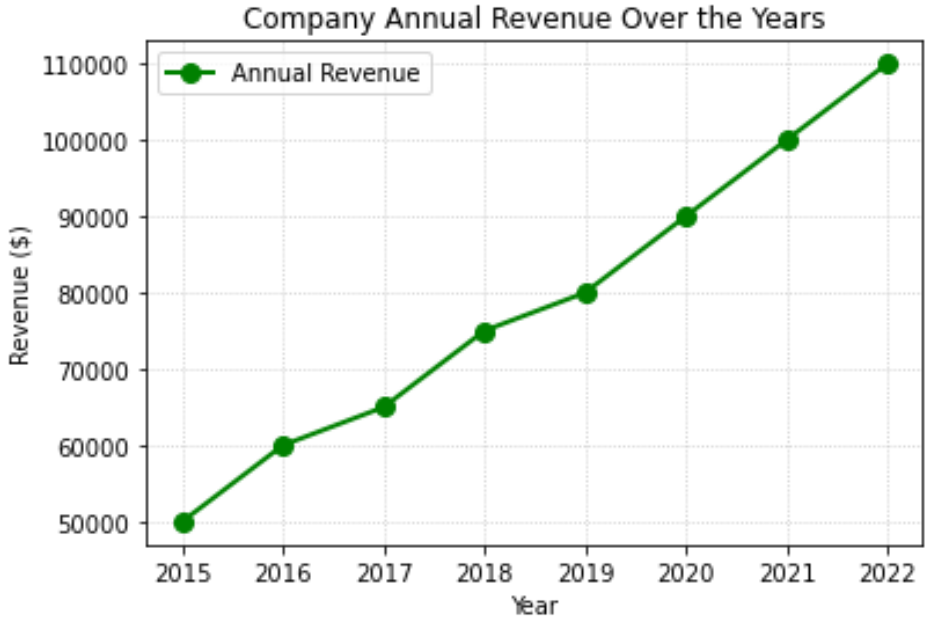
OUTPUT:



OR

```
import matplotlib.pyplot as plt
# Data for the linear plot
years = [2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022]
revenue = [50000, 60000, 65000, 75000, 80000, 90000, 100000, 110000]
# Create a linear plot with line formatting
plt.plot(years,revenue, marker='o', linestyle='-', color='green', label='Annual Revenue', linewidth=2, markersize=8)
# Add labels and a title
plt.xlabel('Year')
plt.ylabel('Revenue ($)')
plt.title('Company Annual Revenue Over the Years')
# Add a grid
plt.grid(True, linestyle=':', alpha=0.7)
# Add a legend
plt.legend()
# Display the plot
plt.show()
```

OUTPUT:

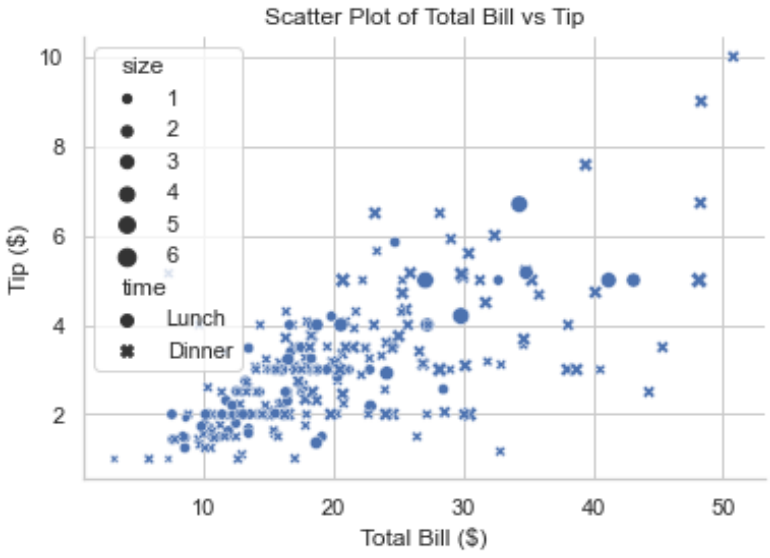


PROGRAM-7

7a) Write a Python program which explains uses of customizing seaborn plots with Aesthetic functions.

```
import seaborn as sns
import matplotlib.pyplot as plt
# Load a sample dataset
tips = sns.load_dataset("tips")
# Set the aesthetic style of the plot
sns.set(style="whitegrid")
# Create a scatter plot using Seaborn
sns.scatterplot(x="total_bill", y="tip", style="time", size="size", data=tips)
# Customize the plot further using Seaborn aesthetic functions
sns.despine() # Remove the top and right spines from the plot
# Set custom labels and title
plt.xlabel("Total Bill ($)")
plt.ylabel("Tip ($)")
plt.title("Scatter Plot of Total Bill vs Tip")
# Show the plot
plt.show()
```

OUTPUT:

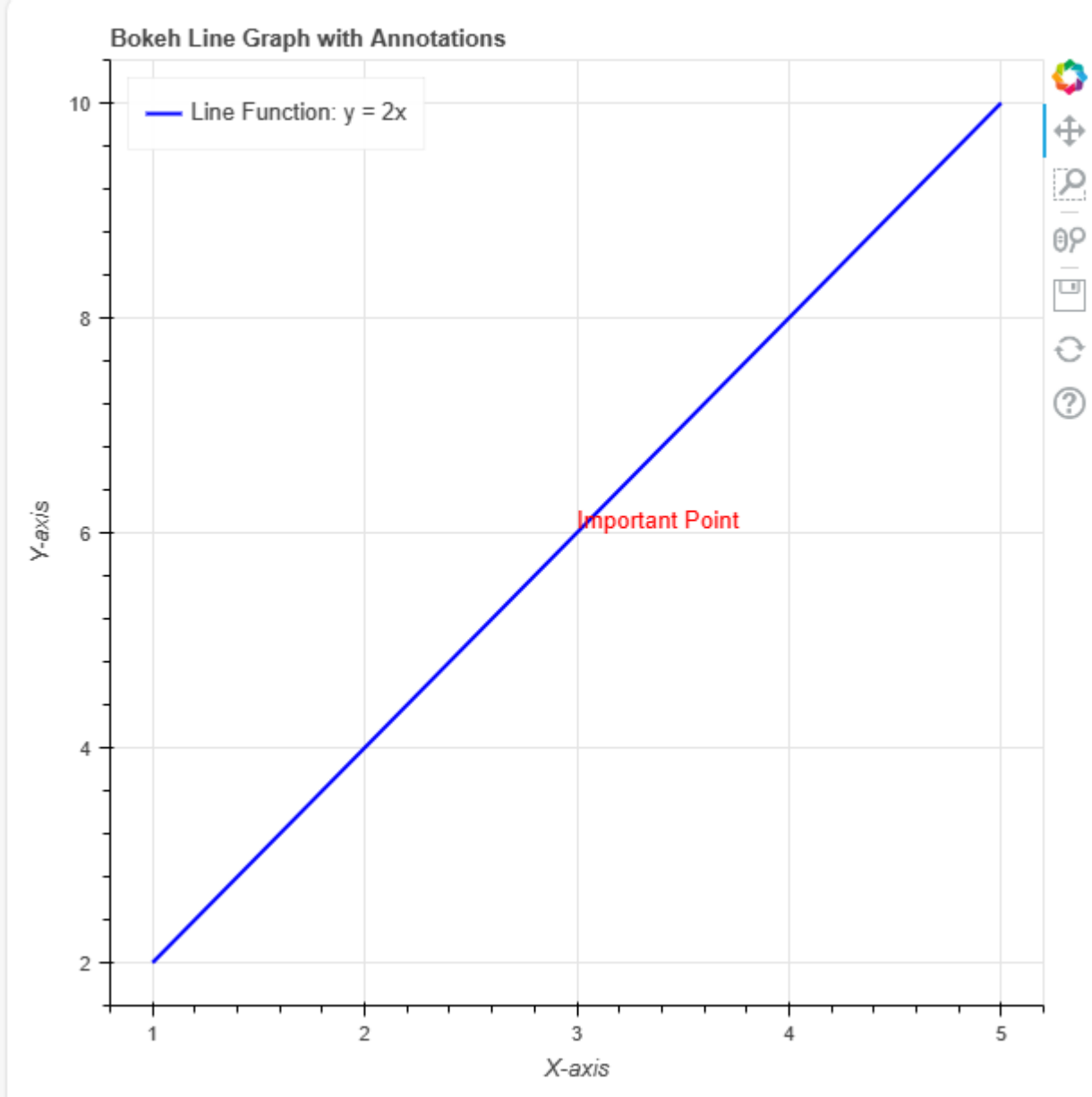


PROGRAM-8

8a) Write a Python program to explain working with bokeh line graph using Annotations and Legends.

```
from bokeh.plotting import figure, output_file, show
from bokeh.models import Label
# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
# Output to static HTML file
output_file("line_graph_with_annotations.html")
# Create a figure
p = figure(title="Bokeh Line Graph with Annotations", x_axis_label='X-axis', y_axis_label='Y-axis')
# Plot the line
p.line(x, y, line_width=2, line_color="blue", legend_label="Line Function: y = 2x")
# Add an annotation
annotation = Label(x=3, y=6, text="Important Point", text_font_size="10pt", text_color="red")
p.add_layout(annotation)
# Add legend
p.legend.location = "top_left"
p.legend.click_policy = "hide"
# Show the plot
show(p)
```

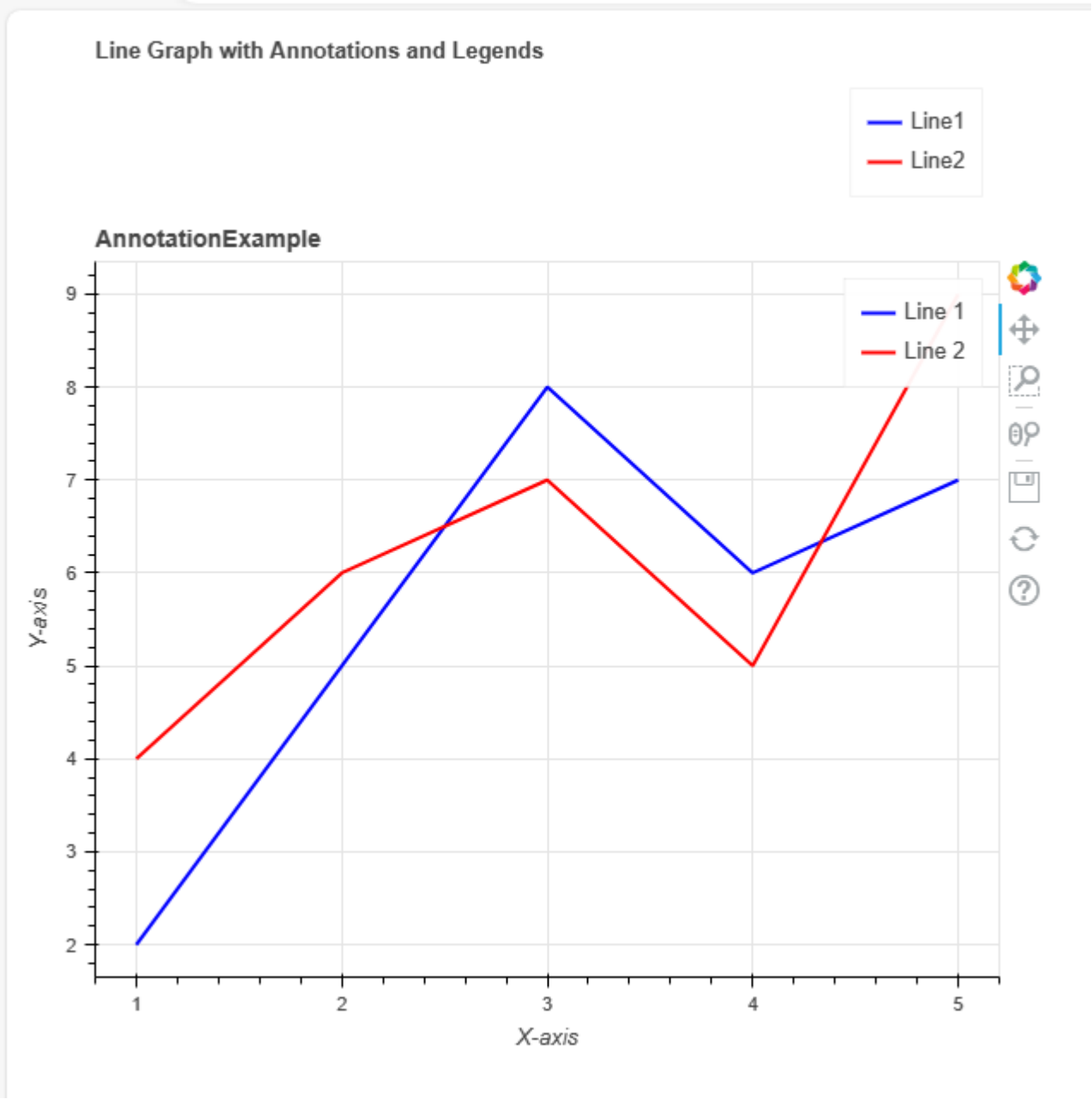
OUTPUT:


Bokeh Line Graph with Annotations
Line Function: y = 2x
Important Point
Y-axis
X-axis
10
8
6
4
2
1
2
3
4
5

OR

```
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource
from bokeh.models.annotations import Title, Legend, LegendItem
from bokeh.io import output_notebook
# Sample data
x = [1, 2, 3, 4, 5]
y1 = [2, 5, 8, 6, 7]
y2 = [4, 6, 7, 5, 9]
# Create a Bokeh figure
p = figure(title="Line Graph with Annotations and Legends", x_axis_label="X-axis", y_axis_label="Y-axis")
# Add data sources
source1=ColumnDataSource(data=dict(x=x,y=y1))
source2 = ColumnDataSource(data=dict(x=x, y=y2))
# Plot the first line
line1 = p.line('x', 'y', source=source1, line_width=2, line_color="blue", legend_label="Line 1")
# Plot the second line
line2 = p.line('x', 'y', source=source2, line_width=2, line_color="red", legend_label="Line 2")
# Add an annotation
annotation=Title(text="AnnotationExample",text_font_size="14px")
p.add_layout(annotation, 'above')
# Create a legend
legend=Legend(items=[LegendItem(label="Line1",renderers=[line1]),LegendItem(label="Line2",renderers=[line2])])
p.add_layout(legend, 'above')
# Show the plot
output_notebook()
show(p)
```

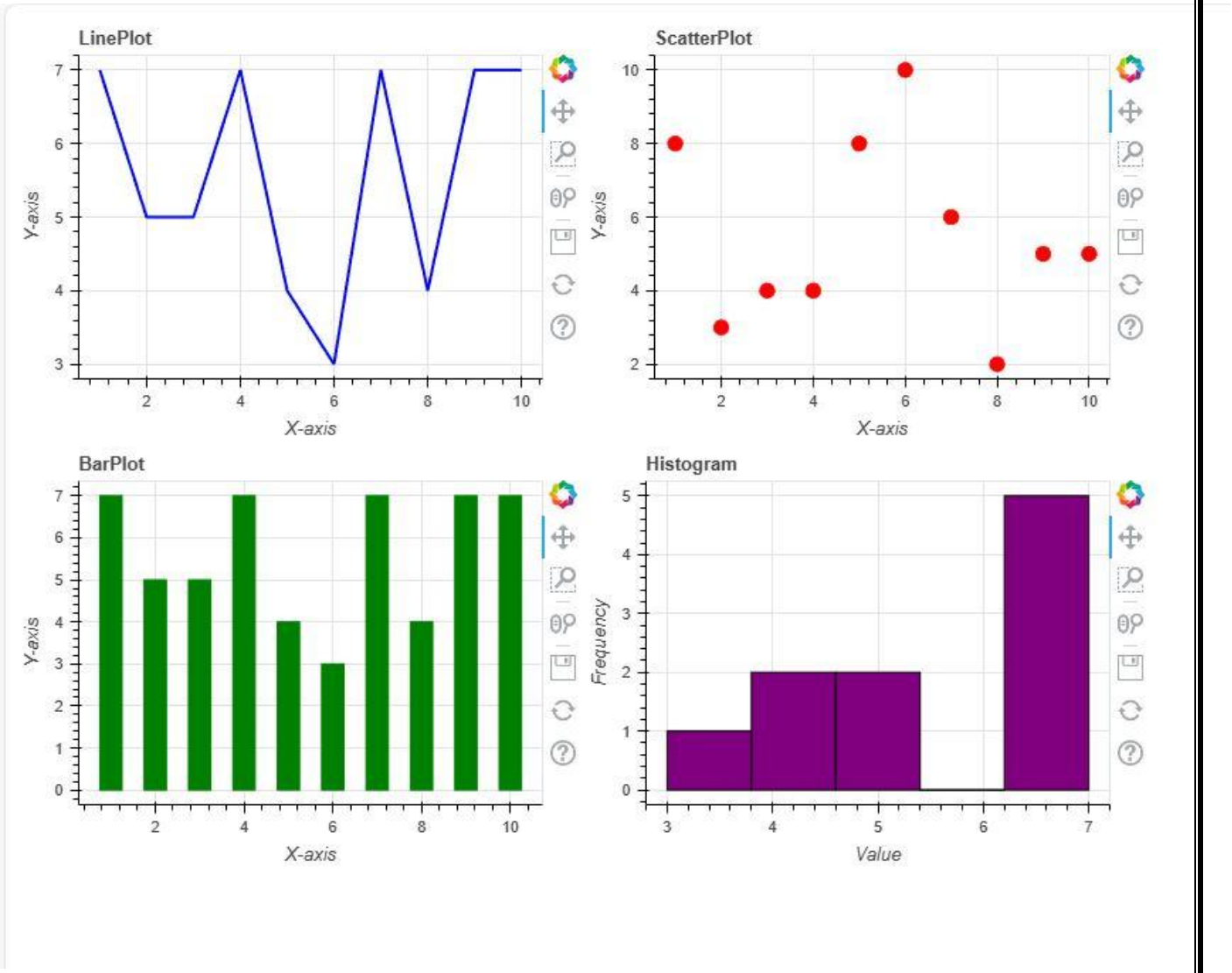
OUTPUT:


Line Graph with Annotations and Legends
Line1
Line2
AnnotationExample
Line 1
Line 2
Y-axis
X-axis
9
8
7
6
5
4
3
2
1
2
3
4
5

8b) Write a Python program for plotting different types of plots using Bokeh

```
from bokeh.plotting import figure, show, output_file
from bokeh.models import ColumnDataSource
from bokeh.layouts import layout
import random
import numpy as np

# Generate some sample data
x = list(range(1, 11))
y1 = [random.randint(1, 10) for inx in x]  # Corrected typo here
y2 = [random.randint(1, 10) for _ in x]

# Create a Bokeh figure with custom plot dimensions
p1 = figure(title="LinePlot", x_axis_label="X-axis", y_axis_label="Y-axis", plot_width=400, plot_height=300)
p1.line(x, y1, line_width=2, line_color="blue")

p2 = figure(title="ScatterPlot", x_axis_label="X-axis", y_axis_label="Y-axis", plot_width=400, plot_height=300)
p2.scatter(x, y2, size=10, color="red", marker="circle")

p3 = figure(title="BarPlot", x_axis_label="X-axis", y_axis_label="Y-axis", plot_width=400, plot_height=300)
p3.vbar(x=x, top=y1, width=0.5, color="green")

p4 = figure(title="Histogram", x_axis_label="Value", y_axis_label="Frequency", plot_width=400, plot_height=300)
hist, edges = np.histogram(y1, bins=5)
p4.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:], fill_color="purple", line_color="black")

# Create a layout with the plots
layout = layout([
    [p1, p2],
    [p3, p4]
])
# Output to an HTML file

output_file("bokeh_plots.html")
# Show the plots
show(layout)
```

OUTPUT:

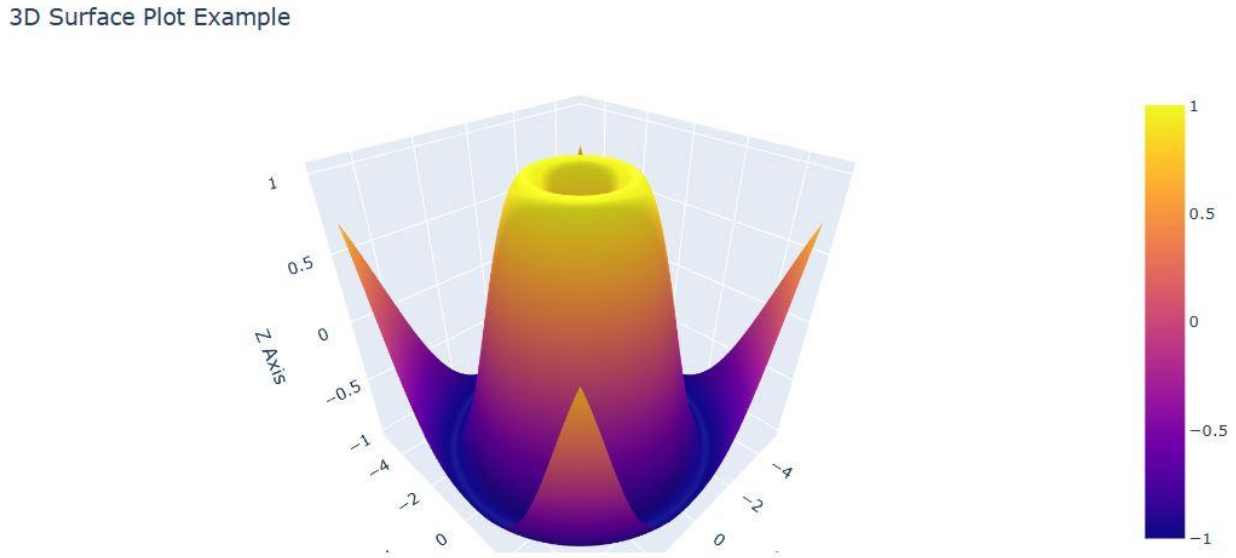

LinePlot
Y-axis
X-axis
ScatterPlot
Y-axis
X-axis
BarPlot
Y-axis
X-axis
Histogram
Frequency
Value

PROGRAM-9

9a) Write a Python program to draw 3D Plots using Plotly Libraries.

```
import plotly.graph_objects as go

import numpy as np

# Create data for the 3D surface plot

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 = go.Figure(data=[go.Surface(z=z, x=x, y=y)])

fig.update_layout(scene=dict(xaxis_title='XAxis',  yaxis_title='YAxis',  zaxis_title='Z Axis',) title='3D Surface Plot Example',)

fig.show()
```

OUTPUT:



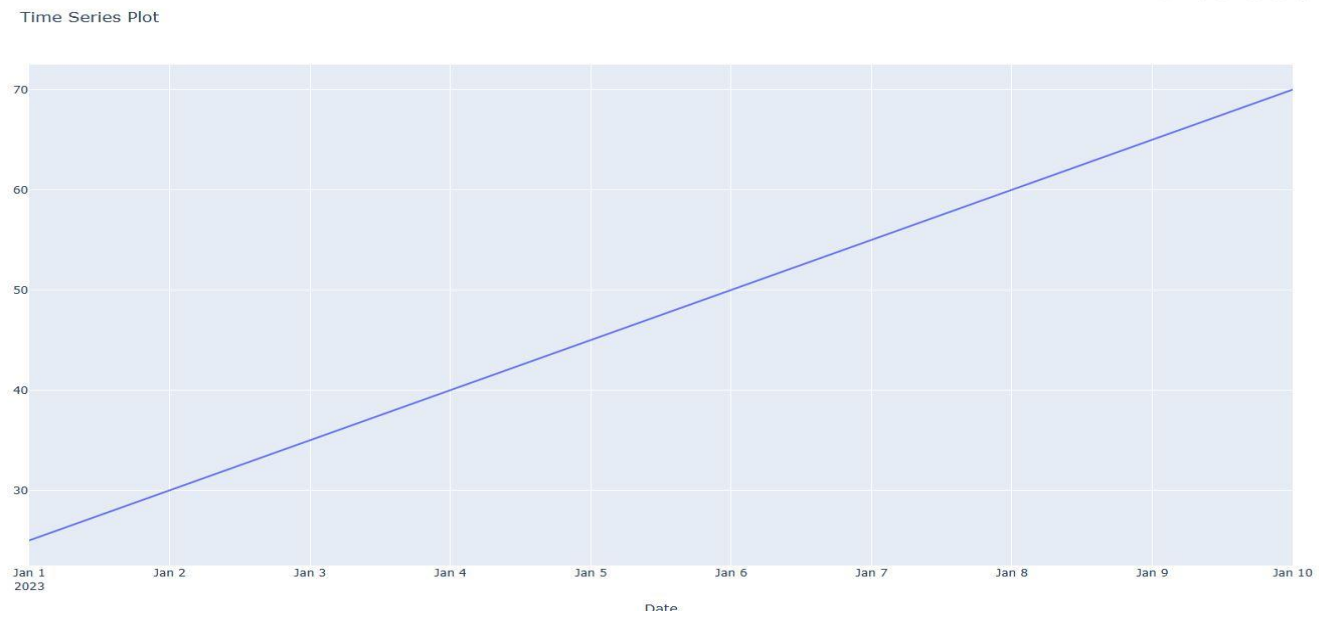
PROGRAM-10

10a). Write a Python program to draw Time Series using Plotly Libraries.

```
import plotly.graph_objs as go
import plotly.offline as pyo
import pandas as pd

# Sample time series data (you can load your own data)
data = {'Date': pd.date_range(start='2023-01-01', periods=10, freq='D'), 'Value': [25, 30, 35, 40, 45, 50, 55, 60, 65, 70]}
df = pd.DataFrame(data)

# Create a time series plot
trace = go.Scatter(x=df['Date'], y=df['Value'], mode='lines', name='Time Series')

layout = go.Layout(title='Time Series Plot', xaxis=dict(title='Date'), yaxis=dict(title='Value'))

# Include a line break or proper indentation
fig = go.Figure(data=[trace], layout=layout)

# Show the plot in a Jupyter Notebook or export it as an HTML file
pyo.plot(fig, filename='time_series_plot.html')
```

OUTPUT:

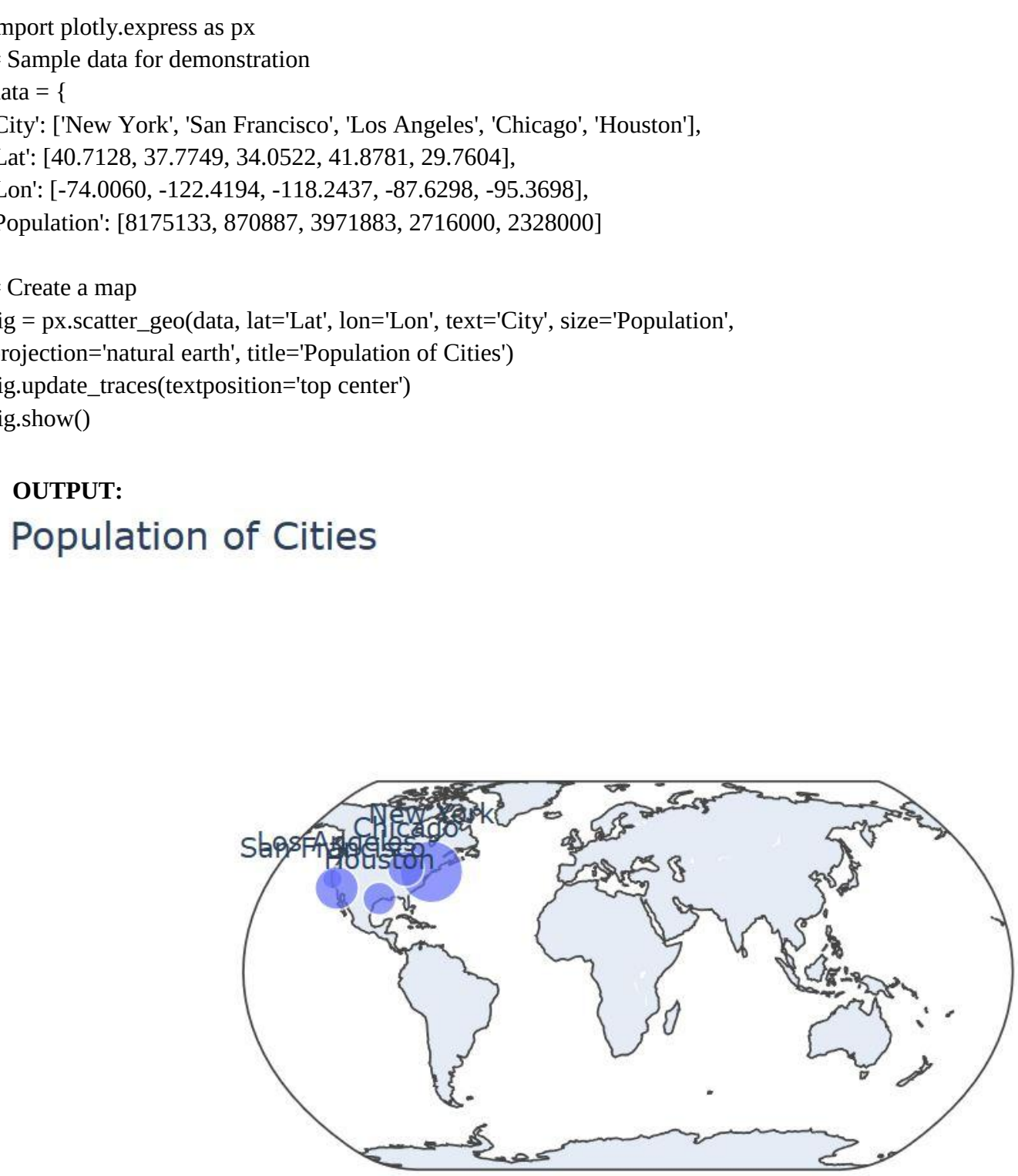
```
'time_series_plot.html'
```



10 b) Write a Python program for creating Maps using Plotly Libraries.

```
import plotly.express as px
# Sample data for demonstration
data = {
'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago', 'Houston'],
'Lat': [40.7128, 37.7749, 34.0522, 41.8781, 29.7604],
'Lon': [-74.0060, -122.4194, -118.2437, -87.6298, -95.3698],
'Population': [8175133, 870887, 3971883, 2716000, 2328000]
}
# Create a map
fig = px.scatter_geo(data, lat='Lat', lon='Lon', text='City', size='Population',
projection='natural earth', title='Population of Cities')
fig.update_traces(textposition='top center')
fig.show()
```

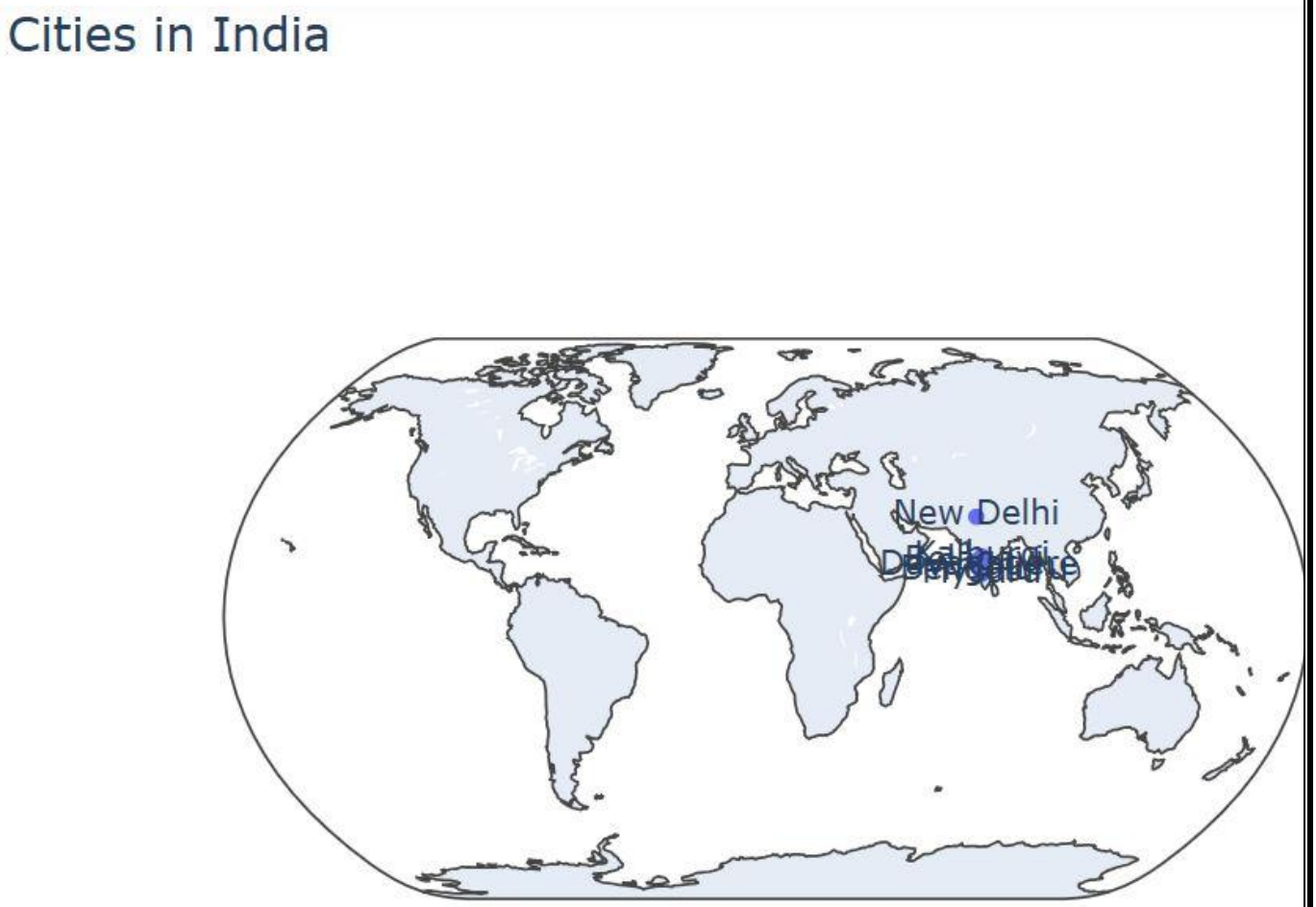
OUTPUT:



OR

```
import plotly.express as px
import pandas as pd
# Load your dataset (ensure it has 'lat' and 'lng' columns for latitude and longitude)
data = {
"City": ["Bengaluru", "Mysuru", "Belagavi", "Kalburgi", "Davanagere", "New Delhi"],
"Lat": [12.9716, 12.2958, 15.8497, 17.3291, 14.4644, 28.6139],
"Lon": [77.5946, 76.6394, 74.5048, 77.2471, 75.9224, 77.2090]
}
df = pd.DataFrame(data)
# Create a scatter map plot
fig = px.scatter_geo(df, lat="Lat", lon="Lon", text="City",
title="Cities in India", projection="natural earth")
# Show the map
fig.show()
```

OUTPUT:



VIVA QUESTIONS

What is Python?

1. Python is one of the most widely-used and popular programming languages, was developed by Guido van Rossum and released first on February 20, 1991.
2. Python is a free and open-source language with a very simple and clean syntax which makes it easyfor developers to learn Python.
3. It supports object-oriented programming and is most commonly used to perform general-purpose programming.
4. Python is used in several domains like Data Science, Machine Learning, Deep Learning, Artificial Intelligence, Scientific Computing Scripting, Networking, Game Development Web Development, Web Scraping, and various other domains, System Scripting, Software Development, and Complex Mathematics.

What are the benefits of using Python language as a tool in the present scenario?

The following are the benefits of using Python language:

Object-Oriented Language, High-Level Language, Dynamically Typed language, Extensive support Libraries, Presence of third-party modules, Open source and community development, Portable and Interactive, Portable across Operating systems.

Is Python a compiled language or an interpreted language?

Python is a partially compiled language and partially interpreted language. '#' is used to comment oneverything that comes after on the line.

Difference between a Mutable datatype and an Immutable data type?

Mutable data types can be edited i.e., they can change at runtime. Eg – List, Dictionary, etc. Immutable data types can not be edited i.e., they can not change at runtime. Eg – String, Tuple, etc.

What is a lambda function?

A lambda function is an anonymous function. This function can have any number of parameters but,can have just one statement.

Pass means performing no operation or in other words, it is a placeholder in the compound statement,where there should be a blank left and nothing has to be written there.

Python provides various web frameworks to develop web applications.

The popular python web frameworks are **Django, Pyramid, Flask.**

1. Python's standard library supports for E-mail processing, FTP, IMAP, and other Internet protocols.
2. Python's **SciPy and NumPy**help in scientific and computational application development.

3. Python's **Tkinter** library supports to create desktop-based GUI applications.

What is the difference between / and // in Python?

// represents floor division whereas / represents precise division. 5//2 = 2 5/2 = 2.5

Yes, **indentation is required in Python**. A Python interpreter can be informed that a group of statements belongs to a specific block of code by using Python indentation. Indentations make the code easy to read for developers in all programming languages but in Python, it is very important to indentthe code in a specific order.

What is Scope in Python?

The location where we can find a variable and also access it if required is called the scope of a variable. **Python Local variable:** Local variables are those that are initialized within a function and are uniqueto that function. It cannot be accessed outside of the function.

Python Global variables: Global variables are the ones that are defined and declared outside any function and are not specified to any function.

Module-level scope: It refers to the global objects of the current module accessible in the program. Outermost scope: It refers to any built-in names that the program can call. The name referenced islocated last among the objects in this scope.

Python documentation strings(or docstrings) provide a convenient way of associating documentation with Python modules, functions, classes, and methods.

Declaring Docstrings: The docstrings are declared using ”’triple single quotes”’ or “”””triple double quotes””””” just below the class, method, or function declaration. All functions should have a docstring. **Accessing Docstrings:** The docstrings can be accessed using the___doc method of the object or using the help function.

What is slicing in Python?

Python Slicing is a string operation for extracting a part of the string, or some part of a list. With this operator, one can specify where to start the slicing, where to end, and specify the step. List slicing returns a new list from the existing list.

PIP is an acronym for Python Installer Package which provides a seamless interface to install various Python modules. It is a command-line tool that can search for packages over the internet and install them without any user interaction.