5b. Write a Python Program for plotting different types of plots using Bokeh

1. Introduction.

- The provided Python program uses the Bokeh library to create and display two different types of plots (line plot and scatter plot) in a single visualization. The primary objective is to demonstrate the usage of Bokeh to generate interactive plots with hover tooltips. The outcome of the program is a visual representation of two plots with the ability to view data points' coordinates upon hovering over them.
- The code is structured to first create each plot separately and then display them together in a single row layout.
- The program utilizes Bokeh's HoverTool to enable interactive hovering over data points in two plots. Hovering triggers tooltips displaying X and Y coordinates, enhancing the user's ability to explore and understand the plotted data dynamically. This feature provides an interactive and informative experience for visual data analysis.

2. Program code.

```
from bokeh.io import output_file as OF
from bokeh.io import show
from bokeh.layouts import row
from bokeh.plotting import figure as figs
from bokeh.models import HoverTool

# Now we will create a new plot
fig1 = figs(width=400, height=400, title="Plot 1")

# Data for the first plot
x1 = [2,3,5,6]
y1 = [1,4,4,7]
fig1.line(x1, y1, line_width=4)

# Add a HoverTool to the first plot
hover1 = HoverTool(tooltips=[("X", "@x"), ("Y", "@y")])
fig1.add_tools(hover1)
```

```
# Then, we will create another plot
x2 = y2 = list(range(10))
fig2 = figs(width=400, height=400, title="Plot 2")
fig2.circle(x2, y2, size=5)

# Add a HoverTool to the second plot
hover2 = HoverTool(tooltips=[("X", "@x"), ("Y", "@y")])
fig2.add_tools(hover2)

# depict visualization
show(row(fig1, fig2))
```

2. Explanation of the code

2a. Importing Functions and Classes

```
from bokeh.io import output_file as OF
from bokeh.io import show
from bokeh.layouts import row
from bokeh.plotting import figure as figs
from bokeh.models import HoverTool
```

The above lines of code sets up the environment for creating interactive plots using Bokeh. It imports functions and classes necessary for handling output files, displaying plots, arranging layouts, creating plotting figures, and adding hover interactions to enhance the visual representation of data.

from bokeh.io import output_file as OF:

- output_file is a function from Bokeh used for specifying the output file where the plot will be saved or displayed. The as OF alias allows you to use the abbreviation OF instead of typing out output_file each time.

from bokeh.io import show:

- show is a function from Bokeh that is used to display the generated plots in a browser or notebook environment. It is a crucial step in visualizing the plots created using Bokeh.

from bokeh.layouts import row:

- row is a layout function from Bokeh that arranges plots or widgets in a horizontal row. It is used here to display multiple plots side by side.

from bokeh.plotting import figure as figs:

- figure is a class in Bokeh that represents a plotting figure. It is the canvas on which you create various types of plots. The as figs alias is used to refer to the figure class using the shorter name figs.

from bokeh.models import HoverTool:

- HoverTool is a class in Bokeh that provides the ability to add hover interactions to plots. It allows you to define tooltips that appear when the cursor hovers over specific plot elements, providing additional information about those elements.

2b. Initialize a new Bokeh plotting figure (fig1)

fig1 = figs(width=400, height=400, title="Plot 1"):

Initializes a new Bokeh plotting figure (fig1) with a canvas size of 400x400 pixels and a title of "Plot 1". This figure serves as the basis for creating visualizations such as line plots, scatter plots, or other types of charts.

- fig1: This is a variable name used to refer to the newly created figure. You can choose any valid variable name, but here it's named fig1 to represent the first figure.
- figs: This is an alias for the figure class from Bokeh, as defined in the import statement from bokeh.plotting import figure as figs. Using figs instead of figure is a matter of convenience and shorter code.
- width=400, height=400: These parameters set the width and height of the plotting figure canvas. In this case, the canvas will have a width of 400 pixels and a height of 400 pixels.
- title="Plot 1": This parameter sets the title of the plot. Here, the title is set to "Plot 1". The title is typically displayed at the top of the plot, providing a brief description of the content.

2c. Data for the first plot

x1 = [2,3,5,6]

y1 = [1,4,4,7]

- x1 and y1 represent the X and Y coordinates of the data points for the first plot. In this example, there are four data points: (2, 1), (3, 4), (5, 4), and (6, 7).

fig1.line(x1, y1, line_width=4)

- fig1.line(): This is a method provided by the Bokeh figure class (figs in this case) to draw a line on the plot.
- x1, y1: These are the data points specified earlier. The line() method will connect these points with a line.
- line_width=4: This parameter sets the width of the line to 4 units, making the line more prominent on the plot.

2d. Incorporating Hover

hover1 = HoverTool(tooltips=[("X", "@x"), ("Y", "@y")])

- **HoverTool()** is a class provided by Bokeh for creating interactive hover tools that display additional information when the mouse cursor hovers over data points.
- **tooltips:** This parameter specifies the content of the tooltip that appears upon hovering. Here, it's set to a list of tuples, where each tuple contains a label and a special syntax starting with "@". The "@x" and "@y" are placeholders that will be replaced with the actual X and Y coordinates of the hovered data point.

2e. Adding HoverTool to the first plot (fig1):

fig1.add_tools(hover1)

- add_tools(): This method is used to attach tools to a Bokeh figure. Here, it adds the HoverTool (hover1) to the first figure (fig1), enabling hover functionality for that particular plot.

This portion of the code is responsible for creating a second plot (fig2) and adding interactivity to it through a HoverTool. Let's break down each part of the code:

2f. Creating data for the second plot:

The below code creates a second plot (fig2) representing a scatter plot of circles with specified data points. It also introduces a HoverTool (hover2) for this plot, enabling interactive tooltips displaying the X and Y coordinates upon hovering over the circles.

x2 = y2 = list(range(10))

- x2 and y2 are both assigned the same list of values ranging from 0 to 9. This creates a set of data points for the second plot. Since the lists are identical, it creates a simple linear set of data points.

Creating the second plot (fig2):

fig2 = figs(width=400, height=400, title="Plot 2")

- This line initializes a new Bokeh plotting figure (fig2) with a canvas size of 400x400 pixels and a title of "Plot 2". It is similar to the creation of the first plot (fig1).

Plotting circles in the second plot (fig2):

fig2.circle(x2, y2, size=5)

- The circle() method is used to create a scatter plot of circles in the second figure (fig2). It takes the X and Y coordinates from x2 and y2, respectively, and sets the size of each circle to 5 units.

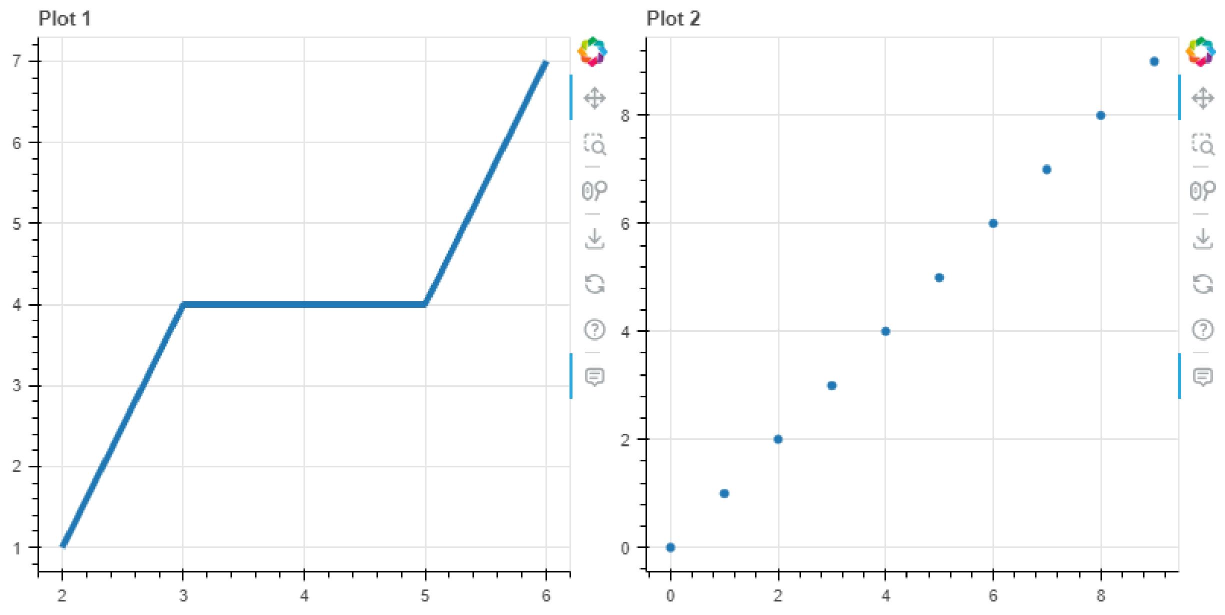
Adding HoverTool to the second plot (fig2):

hover2 = HoverTool(tooltips=[("X", "@x"), ("Y", "@y")])

fig2.add_tools(hover2)

- Similar to the first plot, a new HoverTool (hover2) is created with tooltips displaying the X and Y coordinates. This hover tool is then added to the second figure (fig2) using the add_tools() method, enabling hover functionality for the scatter plot.

3. Output



Analysis: The output of the program reveals an interactive visualization displayed in a browser with two plots displayed side by side. The first plot is a line plot showcasing a seriesof connected data points, while the second plot is a scatter plot with circles representing a set of points. Both plots are equipped with HoverTools, allowing users to obtain detailed information about individual data points by simply hovering over them. The tooltips display the X and Y coordinates of the respective points, enhancing the user's ability to interactively explore and analyze the visualized data. The side-by-side arrangement of the plots providesa convenient means for users to compare and contrast the two visual representations in a single view.