**Bokeh**

Q1. How can you create a Bokeh plot using Python code?

from bokeh.plotting import figure, show

# Create a new plot with specified title and axis labels

p = figure(title="Bokeh Plot", x\_axis\_label="X-axis", y\_axis\_label="Y-axis")

# Add data points to the plot

x = [1, 2, 3, 4, 5]

y = [6, 7, 2, 4, 5]

p.circle(x, y, size=10, color='blue')

# Show the plot

show(p)

his code creates a scatter plot with blue circles as data points. The x-coordinates of the data points are provided in the x list, and the y-coordinates are provided in the y list. The size parameter controls the size of the circles.

Q2. What are glyphs in Bokeh, and how can you add them to a Bokeh plot? Explain with an example.

In Bokeh, glyphs are visual markers that can be added to a plot to represent and visualize data points. They are essentially the building blocks of Bokeh plots, used to create various types of visualizations such as scatter plots, line plots, bar plots, and more.

To add glyphs to a Bokeh plot, we can typically use the figure class provided by the Bokeh library. The figure class represents a plot and provides methods to add different types of glyphs to it. Here's an example that demonstrates how to add glyphs to a Bokeh plot:

from bokeh.plotting import figure, show

from bokeh.io import output\_notebook

# Enable Bokeh to display plots in Jupyter Notebook

output\_notebook()

# Create a new plot with specified width, height, and title

p = figure(width=400, height=300, title='Scatter Plot Example')

# Define some sample data

x = [1, 2, 3, 4, 5]

y = [6, 7, 2, 4, 5]

# Add a circle glyph to the plot

p.circle(x, y, size=10, color='red', alpha=0.5)

# Add a line glyph to the plot

p.line(x, y, line\_width=2, color='blue')

# Show the plot

show(p)

Q3. How can you customize the appearance of a Bokeh plot, including the axes, title, and legend?

To customize the appearance of a Bokeh plot, we can modify various attributes of the axes, title, and legend. Bokeh provides a wide range of options to customize the appearance of your plots. Here's a guide on how to customize these elements:

1. Axes:

To modify the axis labels, you can use the xaxis\_label and yaxis\_label properties of the figure object.

p.xaxis.axis\_label = "X-axis label"

p.yaxis.axis\_label = "Y-axis label"

Properties using the xaxis and yaxis attributes.

p.xaxis.ticker.num\_minor\_ticks = 5 # Number of minor ticks between major ticks

p.xaxis.major\_label\_text\_color = "red" # Color of the major tick labels

The x\_range and y\_range properties.

p.x\_range = Range1d(start=0, end=10) # Sets the x-axis range from 0 to 10

1. Title:

The title attribute of the figure object.

p.title.text = "Plot Title"

p.title.text\_color = "blue" # Color of the title

p.title.text\_font\_size = "14pt" # Font size of the title

1. Legend:

To add a legend to plot, we can use the legend attribute of the figure object.

p.legend.title = "Legend Title" # Title of the legend

p.legend.location = "top\_left" # Position of the legend

p.legend.label\_text\_font\_size = "12pt" # Font size of the legend labels

The legend.items property.

p.legend.items[0].label = "Series 1" # Modify the label of the first legend item

p.legend.items[0].renderers = [renderer1] # Assign a specific renderer to the first legend item

Q4. What is a Bokeh server, and how can you use it to create interactive plots that can be updated in real time?

The Bokeh server is a key component of the Bokeh library, which is a Python library used for creating interactive visualizations and data applications in web browsers. The server allowsas to create and deploy interactive plots and applications that can be updated in real time.

To use the Bokeh server to create interactive plots, we need to follow these general steps:

1. Import the necessary modules: Import the required modules from the Bokeh library, such as bokeh.plotting for creating plots and bokeh.server.server for running the Bokeh server.
2. Create a plot: Use the Bokeh plotting module to create a plot by defining the data, setting up the plot properties, and adding glyphs (such as circles, lines, or bars) to represent the data.
3. Add interactive features: Enhance the plot by adding interactive features like tooltips, hover tools, selection tools, or widgets.
4. Define callbacks: Define callbacks that specify the actions to be performed when certain events occur. For example, you can define a callback that updates the plot based on user input or changes in the data.
5. Create a Bokeh application: Define a Bokeh application that encapsulates the plot and its associated callbacks. This application represents the interactive plot that will be served by the Bokeh server.
6. Start the Bokeh server: Use the bokeh.server.server module to start the Bokeh server and specify the applications to be served.
7. View and interact with the plot: Open a web browser and navigate to the URL where the Bokeh server is running. You can then view and interact with the plot in real time, and any updates triggered by the defined callbacks will be reflected in the plot.

Q5. How can you embed a Bokeh plot into a web page or dashboard using Flask or Django?

1. A simple scatter plot using Bokeh:

from bokeh.plotting import figure

from bokeh.embed import components

def create\_plot():

p = figure(title='My Bokeh Plot', x\_axis\_label='X', y\_axis\_label='Y')

p.circle([1, 2, 3, 4, 5], [6, 7, 2, 4, 5])

return p

2. The components function provided by Bokeh.

def generate\_components():

plot = create\_plot()

script, div = components(plot)

return script, div