Assignment

Q1. How can you create a Bokeh plot using Python code?
Ans:To create a Bokeh plot using Python code, you need to follow these general steps:
Install Bokeh:

First, you need to install the Bokeh library. You can install it using the following pip command in your terminal or command prompt:

bash

Copy code

Import Bokeh Modules:

In your Python script or Jupyter Notebook, import the necessary modules from Bokeh. The core module is `bokeh.plotting`, which provides functions for creating different types of plots.

python

Copy code

```
from                import
```

Create a Figure:

Use the `figure()` function to create a figure object. This object represents the entire plot and allows you to customize various aspects of the plot.

python

Copy code

```
           "My Bokeh Plot"              "X-axis Label"
      "Y-axis Label"
```

Add Glyphs (Markers, Lines, etc.):

Use various glyph functions provided by Bokeh to add markers, lines, or other shapes to your plot. For example, `circle()`, `line()`, etc.

python

Copy code

```
    1  2  3  4  5     6  7  2  4  8       10       "navy"       0.5
```

Show the Plot:

Finally, use the `show()` function to display the plot. This will open the plot in a new browser window or inline if you're using a Jupyter Notebook.

python

Copy code

Putting it all together, here's a simple example:

python

Copy code

```
from                import

              "My Bokeh Plot"              "X-axis Label"
          "Y-axis Label"

    1  2  3  4  5     6  7  2  4  8       10       "navy"       0.5
```

This example creates a scatter plot with circles on the specified x and y coordinates. You can explore more advanced features and customization options offered by Bokeh for creating interactive and visually appealing plots.

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

Ans:In Bokeh, glyphs are the basic visual building blocks used to represent data on a plot. Glyphs define the visual properties of data points, such as markers, lines, bars, etc. Bokeh provides a variety of glyph functions for different types of plots, and you can customize their appearance and behavior.

Here are some common types of glyphs in Bokeh:

- `circle()`: Draws circles at specified x and y coordinates.
- `line()`: Draws lines connecting points at specified x and y coordinates.
- `rect()`: Draws rectangles with specified width and height at x and y coordinates.
- `square()`: Draws squares at specified x and y coordinates.
- `cross()`: Draws cross shapes at specified x and y coordinates.

Here's an example of how to add glyphs to a Bokeh plot:

python

Copy code

```
from                import

                    "Bokeh Plot with Glyphs"              "X-axis"
              "Y-axis"

            1  2  3  4  5     6  7  2  4  8       10       "navy"       0.5
              "Circles"

          1  2  3  4  5     6  7  2  4  8            2            "green"
            "Line"
```

```
2  4  6      9  5  7         0.9         1         "orange"         0.7
   "Rectangles"


            "top_left"
```

In this example:

- `circle()`, `line()`, and `rect()` functions are used to add different glyphs to the plot.
- Various parameters like `size`, `color`, `line_width`, `width`, `height`, etc., are used to customize the appearance of the glyphs.
- `legend_label` is used to provide labels for the legend.
- The `legend.location` attribute is set to determine the position of the legend on the plot.

This example demonstrates the use of multiple glyphs in a single Bokeh plot. You can explore the Bokeh documentation for a comprehensive list of glyph functions and their parameters: Bokeh Glyphs

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

Ans:Customizing the appearance of a Bokeh plot involves modifying various attributes of the plot, including the axes, title, legend, and other visual elements. Here's an overview of how you can customize these components:

1. Axes Customization:

Axis Labels and Ticks:

- Use `x_axis_label` and `y_axis_label` attributes to set the labels for the x and y axes.
- Use `x_axis_location` and `y_axis_location` to control the location of the axes.
- Customize ticks using `major_label_text_font_size`, `major_tick_line_color`, etc.

Axis Range:

- Set the range of the axes using `x_range` and `y_range`.

- Use `x_range.start`, `x_range.end`, `y_range.start`, and `y_range.end` to define specific ranges.

2. Title Customization:

Plot Title:

- Set the title of the plot using the `title` attribute.
- Customize title properties such as font size, color, etc.

3. Legend Customization:

- Use `legend` attribute to customize the legend.
- Set the `location` to specify where the legend should appear (e.g., "top_left", "bottom_right").
- Adjust `label_text_font_size`, `label_text_color`, and other properties.

Example:

python

Copy code

```
from                 import


                "Custom Bokeh Plot"                "X-axis"
             "Y-axis"
            600              400


            1  2  3  4  5      6  7  2  4  8        10        "navy"        0.5
             "Circles"


                                  "14pt"
                                  "14pt"
                                   "12pt"
                                   "12pt"


                       "16pt"
                   "purple"
```

```
                "12pt"
            "green"
"top_left"
```

In this example:

- Axes labels, title, and legend are customized using various attributes.
- `plot_width` and `plot_height` control the size of the plot.
- Different font sizes, colors, and other properties are adjusted for visual customization.

You can explore the Bokeh documentation for a comprehensive list of customization options:

Styling Visual Attributes

Q4. What is a Bokeh server, and how can you use it to create interactive plots that can be updated in
real time?
Ans:A Bokeh server is a feature of Bokeh that allows you to create interactive web applications with real-time updates. With Bokeh server, you can build dynamic and responsive web applications by adding interactivity to your Bokeh plots. It allows you to create applications where the state of the plot can change based on user interactions, events, or updates from external sources.

Bokeh server operates by running a Python script on a server, which serves a Bokeh plot or application to clients' web browsers. The server maintains the state of the application, and any changes made on the client side are communicated back to the server, triggering updates to the plot.

Here are the basic steps to create a Bokeh server application:

Import Necessary Modules:

Import the required Bokeh modules and classes.

python

Copy code

```
from                  import
from                 import
```

Create a Figure:

Create a Bokeh figure as you would in a regular Bokeh script.

python

Copy code

```
                    400              400        "Interactive Plot"
```

Create a ColumnDataSource:

Use `ColumnDataSource` to store the data that will be used in the plot. This allows for dynamic updates.

python

Copy code

```
                                'x'   1  2  3   'y'   4  5  6
```

Define Callbacks:

Define Python functions that will be called when certain events occur (e.g., button click, slider value change). These functions update the data or properties of the plot.

python

Copy code

```
def update_data
```

```
    pass
```

Connect Callbacks to Widgets:

Connect the defined callbacks to Bokeh widgets (e.g., buttons, sliders) to trigger updates.

python

Copy code

```
                0       10        5       1        "Slider"
            'value'
```

Add Widgets and Plots:

Add widgets and plots to the layout.

python

Copy code

Set Up Document:

Set up the Bokeh document using `curdoc()`.

python

Copy code

Run the Bokeh Server:

Save the script as `app.py` and run it using the Bokeh server command.

bash

Copy code

This is a basic example, and Bokeh server provides more advanced features for handling user interactions, handling session variables, and updating plots based on various events. It's a powerful tool for creating dynamic and interactive data visualizations.

Q5. How can you embed a Bokeh plot into a web page or dashboard using Flask or Django?
Ans:Embedding a Bokeh plot into a web page or dashboard using Flask or Django involves creating a Bokeh plot, saving it to an HTML file, and then rendering that HTML file within your Flask or Django application. Here are the general steps for both Flask and Django:

Embedding in Flask:

Install Flask and Bokeh:

If you haven't installed Flask and Bokeh, do so using:

bash

Copy code

Create a Flask App:

Create a Flask app (e.g., in a file named `app.py`):

python

Copy code

```
from       import
from                import
from             import
```

```
          '/'
def index

                      400              400        "Flask Embedding Example"
          1  2  3    4  5  6        10        "navy"        0.5




 return                 'index.html'

if             '__main__'
               True
```

Create HTML Template (`templates/index.html`):

Create an HTML template that includes the Bokeh script and div components:

html

Copy code

```
    html
lang "en"

 charset "UTF-8"
```

Run the Flask App:

Run the Flask app:

bash

Copy code

Visit `http://127.0.0.1:5000/` in your browser to see the embedded Bokeh plot.

Embedding in Django:

Install Django and Bokeh:

If you haven't installed Django and Bokeh, do so using:

bash

Copy code

Create a Django Project and App:

Create a Django project and app:

bash

Copy code

```
cd
```

Update `myapp/views.py`:

Update the `views.py` file in your app:

python

Copy code

```
from                  import
from                import
from             import

def index

                      400              400        "Django Embedding Example"
          1  2  3    4  5  6        10        "navy"        0.5
```

```
return                  'myapp/index.html'    'script'          'div'
```

Create HTML Template (`myapp/templates/myapp/index.html`):

Create an HTML template in the `myapp/templates/myapp` directory:

html

Copy code

```
    html
lang "en"

 charset "UTF-8"
```

Update `myproject/urls.py`:

Update the `urls.py` file in your project:

python

Copy code

```
from               import
from             import


    'admin/'
    ''          'myapp.urls'
```

Update `myapp/urls.py`:

Create a `urls.py` file in your app and define the URL pattern:

python

Copy code

```
from              import
from        import


      ''              'index'
```

Run the Django Development Server:

Run the Django development server:

bash

Copy code

Visit `http://127.0.0.1:8000/` in your browser to see the embedded Bokeh plot.

These examples demonstrate the basic setup for embedding Bokeh plots in Flask and Django applications. Adjust the code and structure according to the specific needs of your project.