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

```
import bokeh.io
import bokeh.plotting
bokeh.io.output_notebook()
```

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 represent data points on a plot. They can be used to create various types of visual elements such as circles, squares, lines, bars, and more. Glyphs are added to a Bokeh plot to visually represent the data in a meaningful way.

```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
b = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

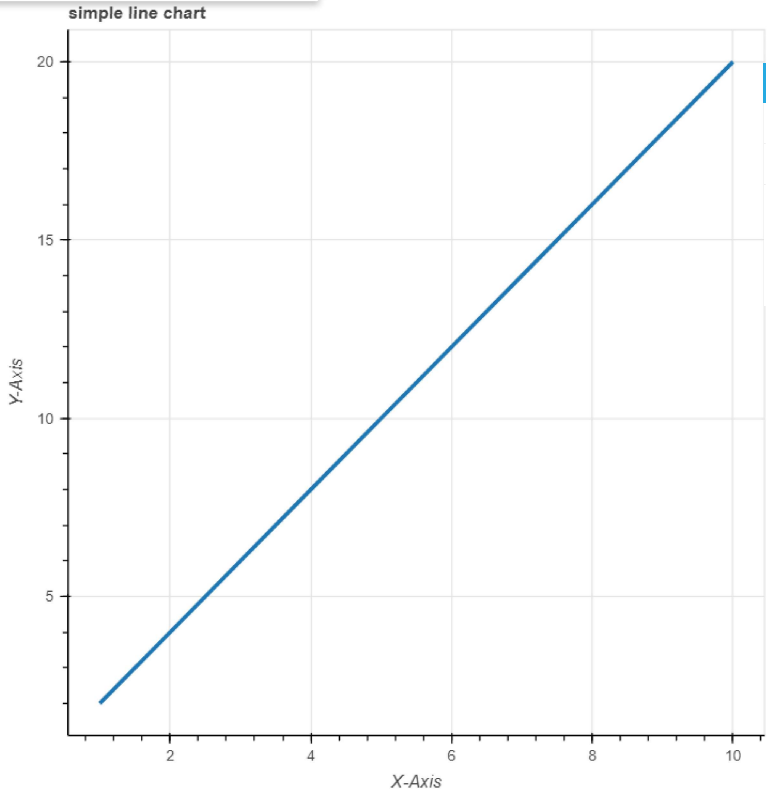
```
my_plot = figure(title="simple line chart", x_axis_label="X-Axis",
                 y_axis_label="Y-Axis")
```

```
my_plot.line(a, b, line_width=3)
```

```
output_file("line.html")
```

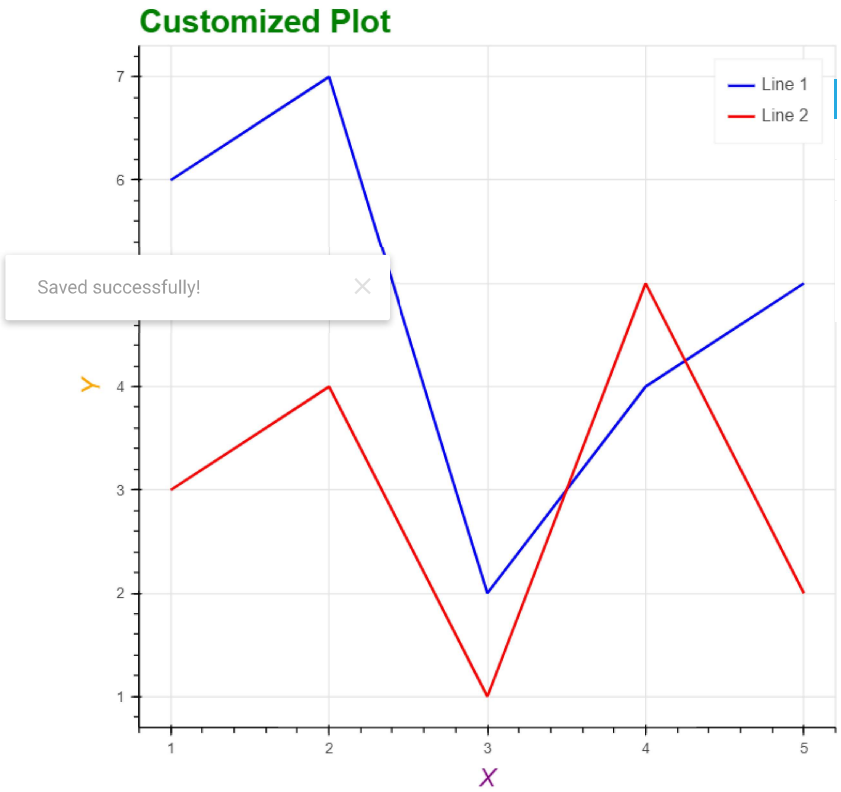
```
show(my_plot)
```

Saved successfully!



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

```
import bokeh.plotting as bplt
x = [1, 2, 3, 4, 5]
y1 = [6, 7, 2, 4, 5]
y2 = [3, 4, 1, 5, 2]

p = bplt.figure(title='Customized Plot', x_axis_label='X', y_axis_label='Y')

p.line(x, y1, color='blue', line_width=2, legend_label='Line 1')
p.line(x, y2, color='red', line_width=2, legend_label='Line 2')


p.title.text_color = 'green'
p.title.text_font = 'Arial'
p.title.text_font_size = '18pt'

p.xaxis.axis_label_text_color = 'purple'
p.xaxis.axis_label_text_font_size = '14pt'

p.yaxis.axis_label_text_color = 'orange'
p.yaxis.axis_label_text_font_size = '14pt'


bplt.show(p)
```



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 feature of the Bokeh library that allows you to create interactive plots that can be updated in real time. It enables you to host and serve your Bokeh plots, which can respond to user interactions and dynamically update their content without regenerating the entire plot.

To use the Bokeh server and create interactive plots that can be updated in real time, you follow these steps:

1. Create your Bokeh plot using the regular Bokeh API, defining the figure, adding glyphs, and customizing properties.
2. Define callback functions that specify how the plot should update in response to specific events or interactions. These callbacks are written in Python.

3. Start the Bokeh server by running the `bokeh serve` command in your terminal or command prompt, followed by the filename of your script or application.
4. Access the plot in your web browser by navigating to the appropriate URL, typically `http://localhost:5006`, where you can interact with the plot.
5. Interact with the plot, triggering events that call the defined callbacks. The plot updates dynamically based on the defined callback logic and sends the updated content to the browser, reflecting the changes in real time.

By utilizing the Bokeh server, you can create interactive plots and applications that respond to user input and update their content in real time. This allows for dynamic data exploration, live monitoring, and building interactive dashboards, among other applications.

Please note that this is a simplified explanation, and the Bokeh server can be used for more complex applications involving multiple plots, widgets, and layouts.

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

To embed a Bokeh plot into a web page or dashboard using Flask or Django, you can use the Bokeh components function, which generates the JavaScript and HTML code needed to display a Bokeh plot in a web page.

Here's an example of how to embed a Bokeh plot into a Flask web application:

```
from flask import Flask, render_template
from bokeh.plotting import figure
from bokeh.embed import components

app = Flask(__name__)

x = [1, 2, 3, 4, 5]
y = [6, 7, 2, 4, 5]
```

Saved successfully!

```
x_axis_label='x', y_axis_label='y')

@app.route('/')
def index():
    # generating the JavaScript and HTML components
    script, div = components(p)
    # rendering the template with the components
    return render_template('index.html', script=script, div=div)

if __name__ == '__main__':
    app.run(debug=True)
```

```
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
```