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

Import the necessary modules from Bokeh. Prepare your data, typically as Python lists or Pandas DataFrames. Create a figure object using figure() to specify the plot's attributes (e.g., width, height, title). Add glyphs (visual elements like lines, circles, or bars) to the figure using Bokeh's glyph methods. Customize the plot's appearance, such as axes, titles, and legends. Show or save the plot.

In [ ]:
```
from bokeh.plotting import figure, show

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Create a figure
p = figure(title="Simple Line Plot", x_axis_label="X-axis", y_axis_label="Y-axis

# Add a line glyph to the figure
p.line(x, y, line_width=2, line_color="blue", legend_label="Line")

# Customize the appearance (e.g., add grid lines)
p.grid.grid_line_alpha = 0.5

# Show the plot
show(p
```

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 shapes or markers used to represent data on a plot. Common glyphs include lines, circles, squares, and bars. You can add glyphs to a Bokeh plot using specific glyph functions corresponding to the type of visualization you want. For example:

line() adds a line glyph. circle() adds a circle glyph. square() adds a square glyph. rect() adds a rectangle glyph. vbar() and hbar() add vertical and horizontal bar glyphs, respectively.

In [3]:
```
from bokeh.plotting import figure, show

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Create a figure
p = figure(title="Circle Glyph Example", x_axis_label="X-axis", y_axis_label="Y-

# Add circle glyphs to the figure
p.circle(x, y, size=10, color="blue", legend_label="Circles")

# Show the plot
show(p)
```

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

Title and Labels: You can set the title and axis labels using the title, x_axis_label, and y_axis_label attributes of the figure.

Legends: To add legends to your plot, assign a legend_label to the glyphs you want to include in the legend, and then call add_legend() on the figure.

Axes: Customize axes using properties like xaxis, yaxis, axis_label_standoff, axis_label_text_font_size, and more.

Grid Lines: You can control grid lines with properties like grid.grid_line_color, grid.grid_line_alpha, and grid.grid_line_dash.

In [4]:
```
from bokeh.plotting import figure, show

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Create a figure with customizations
p = figure(
    title="Customized Bokeh Plot",
    x_axis_label="X-axis",
    y_axis_label="Y-axis",
    width=400,
    height=300,
)
p.line(x, y, line_width=2, line_color="blue", legend_label="Line")
p.circle(x, y, size=10, color="red", legend_label="Circles")

# Customize the appearance further
p.legend.location = "top_left"
p.grid.grid_line_color = "gray"
p.grid.grid_line_alpha = 0.3

# Show the plot
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?

A Bokeh server is a component of Bokeh that allows you to create interactive, real-time data applications and dashboards. With a Bokeh server, you can build web applications that update and respond to user interactions or external data changes dynamically. To use a Bokeh server:

Define a Bokeh application by creating a Python function that sets up the plot and defines any interactions or updates.

1.Create a curdoc() instance to access the current document.

2.Add your plot and widgets to the document.

3.Define callbacks to handle user interactions, data updates, or events.

4.Run the Bokeh server to serve your application locally or deploy it to a web server.

In [6]:
```python
from bokeh.plotting import curdoc, figure
from bokeh.models import Button
import numpy as np

# Create initial data
x = np.linspace(0, 4 * np.pi, 100)
y = np.sin(x)

# Create a figure
plot = figure(title="Real-time Sine Wave")

# Add a line glyph
line = plot.line(x, y, line_width=2)

# Create a button widget
button = Button(label="Update Data")

# Define a callback function
def update_data():
    new_y = np.sin(x + np.random.rand() * np.pi)  # Generate random data
    line.data_source.data["y"] = new_y  # Update the y-data

# Attach the callback to the button
button.on_click(update_data)

# Add the plot and button to the current document
curdoc().add_root(plot)
curdoc().add_root(button)
```

In [ ]:
```
Q5. How can you embed a Bokeh plot into a web page or dashboard using Flask or D
```

Create your Flask or Django web application.

1.Import the necessary Bokeh modules (bokeh.embed for Flask, and bokeh.embed.server_document for Django).

2.Create a Bokeh plot and add it to a Bokeh document.

3.Use the bokeh.embed functions to generate the HTML or script components for the plot.

4.In your Flask