```
#How can you create a Bokeh plot using Python code?

#To create a Bokeh plot using Python code, you can follow these general steps:
#Import the necessary Bokeh modules and classes.
#Prepare the data to be plotted.
#Create a Figure object that defines the plot properties.
#Add the necessary glyphs (markers, lines, etc.) to the figure to display the data.
#Define the formatting and layout of the plot (e.g., axis labels, legend, gridlines).
#Show or save the resulting plot.
from bokeh.plotting import figure, output_file, show
import numpy as np

# Prepare the data
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Create a Figure object
fig = figure(title="Sine Curve", x_axis_label="X", y_axis_label="Y")

# Add a circle glyph to the figure
fig.circle(x, y, size=5, color="blue")

# Show the resulting plot in the default browser
show(fig)
#In this example, we first imported the necessary Bokeh modules and classes (figure, output_file, and show). Then, we prepared the data f


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

#Glyphs in Bokeh are visual marks, such as markers or lines, used to represent data points or lines in a plot. Bokeh provides a wide rang

#To add glyphs to a Bokeh plot, you can use one of the available glyph methods, such as circle, line, or rect, which take one or more arr
from bokeh.plotting import figure, output_file, show
import numpy as np

# Prepare the data
x = np.random.rand(50)
y = np.random.rand(50)

# Create a Figure object
fig = figure(title="Scatter Plot", x_axis_label="X", y_axis_label="Y")

# Add a circle glyph to the figure
fig.circle(x, y, size=10, color="blue", alpha=0.5)

# Show the resulting plot in the default browser
show(fig)



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

'''#Bokeh provides a wide range of options for customizing the appearance of a plot, including the axes, title, legend, and more. Here ar
# Changing axis properties: You can change the properties of the x-axis and y-axis using the xaxis and yaxis attributes of the Figure obj
fig.xaxis.axis_label = "X-axis Label"
fig.xaxis.axis_label_text_font_size = "16pt"
fig.xaxis.major_label_text_font_size = "14pt"
#Adding a plot title: You can set the title of the plot using the title attribute of the Figure object:
fig.title.text = "My Plot Title"
fig.title.text_font_size = "20pt"
#Adding a legend: If you have multiple glyphs in your plot, you can add a legend using the legend attribute of each glyph method. For exa
#fig.circle(x, y1, size=10, color="blue", alpha=0.5, legend_label="Series 1")
#fig.line(x, y2, line_width=2, color="red", alpha=0.8, legend_label="Series 2")
#fig.legend.location = "top_left"
fig.legend.label_text_font_size = "14pt"
#Changing plot background and border: You can set the background color and border properties of the plot using the background_fill_color,
fig.background_fill_color = "#f2f2f2"
fig.border_fill_color = "white"
fig.border_line_width = 2
#Adjusting plot layout: You can adjust the layout of the plot using the sizing_mode attribute of the Figure object, which controls how th
fig.sizing_mode = "scale_width"  # adjust plot width, keep height constant
fig.sizing_mode = "stretch_both"  # stretch plot to fill entire available space
#These are just a few examples of how you can customize the appearance of a Bokeh plot. Bokeh provides many more options and features for
```

```
'#Bokeh provides a wide range of options for customizing the appearance of a plot, including the axes, title, legend, and
more. Here are some ways to customize the appearance of a Bokeh plot\n# Changing axis properties: You can change the prope
rties of the x-axis and y-axis using the xaxis and yaxis attributes of the Figure object. For example, you can set the lab
el, font size, and tick labels as follows:\nfig.xaxis.axis_label = "X-axis Label"\nfig.xaxis.axis_label_text_font_size =
"16pt"\nfig.xaxis.major_label_text_font_size = "14pt"\n#Adding a plot title: You can set the title of the plot using the t
itle attribute of the Figure object:\nfig.title.text = "My Plot Title"\nfig.title.text_font_size = "20pt"\n#Adding a legen
d: If you have multiple glyphs in your plot, you can add a legend using the legend attribute of each glyph method. For exa
```

```
#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 Python application that allows you to create interactive Bokeh plots that can be updated in real-time. With the Bo

Define the layout: Define the layout of your Bokeh plot using the various layout and widget objects provided by Bokeh. You can create plo

Define the callback functions: Define the callback functions that will handle the user input or the data changes. Bokeh provides several 

Create the Bokeh server app: Create the Bokeh server app by defining a function that returns the layout and the callback functions. The f

Run the Bokeh server: Run the Bokeh server using the bokeh serve command. The command takes the name of the Python file containing the ap

Here is a simple example that demonstrates how to create a Bokeh server app that updates a plot in real-time based on a slider widget:'''
from bokeh.io import curdoc
from bokeh.layouts import column
from bokeh.models import ColumnDataSource, Slider
from bokeh.plotting import figure

# Create a ColumnDataSource with some initial data
source = ColumnDataSource(data=dict(x=[1, 2, 3], y=[1, 2, 3]))

# Create a plot with the initial data
plot = figure(plot_height=300, plot_width=600)
plot.line('x', 'y', source=source)

# Define the callback function for the slider widget
def callback(attr, old, new):
    # Update the data based on the new slider value
    source.data = dict(x=[1, 2, 3], y=[new**2, new**3, new**4])

# Create a slider widget and attach the callback function
slider = Slider(start=1, end=10, value=1, step=1, title="Slider")
slider.on_change('value', callback)

# Add the plot and the slider widget to the document
curdoc().add_root(column(slider, plot))
#When you run the Bokeh server using bokeh serve myapp.py, a local server is started that listens for incoming connections. You can open 




#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 follow the following steps:

Create the Bokeh plot: Create the Bokeh plot using the Bokeh library as you would normally.

Convert the Bokeh plot to HTML: Convert the Bokeh plot to an HTML file using the bokeh.embed module. This module provides functions for g

Create a Flask or Django view: Create a Flask or Django view that renders the HTML file generated by Bokeh. You can use the render_templa

Add the view to your application: Add the view to your Flask or Django application by defining a URL route that maps to the view. You can
from flask import Flask, render_template
from bokeh.plotting import figure
from bokeh.embed import components

app = Flask(__name__)

@app.route("/")
def index():
    # Create the Bokeh plot
    plot = figure(plot_width=400, plot_height=400)
    plot.circle([1, 2, 3], [4, 5, 6])

    # Convert the plot to HTML components
    script, div = components(plot)

    # Render the HTML template with the plot components
    return render_template("index.html", script=script, div=div)

if __name__ == "__main__":
    app.run(debug=True)
'''In this example, we create a Flask application with a single view mapped to the root URL /. In the view function, we create a Bokeh pl
from django.views.generic import TemplateView
from bokeh.plotting import figure
from bokeh.embed import components

class IndexView(TemplateView):
    template_name = "index.html"

    def get_context_data(self, **kwargs):
        # Create the Bokeh plot
        plot = figure(plot_width=400, plot_height=400)
        plot.circle([1, 2, 3], [4, 5, 6])
        # Convert the plot to HTML components
        script, div = components(plot)

        # Add the plot components to the context
        context = super().get_context_data(**kwargs)
        context["script"] = script
        context["div"] = div
        return context
'''In this example, we create a Django view called IndexView that extends the TemplateView class. We specify the name of the HTML templat
```

```
 * 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
```

Colab paid products - Cancel contracts here

✓ 3m 23s completed at 9:20 PM