How to Add Graphs to Flask apps

Last Updated : 19 Dec, 2022

Python. The Bokeh library will be used to create interactive bar graphs and we will visualize this graph through a simple frontend HTML page. For this, we will first write the endpoints in Flask which will help us to create Bokeh charts, and then we will create HTML templates that will utilize these Bokeh charts to display them to the user. Before moving let's understand the basic technologies we will use in this article.

Flask is a web framework that offers libraries for creating simple web applications in Python. It is built using the Jinja2 template engine and the WSGI tools. Flask is considered a micro-framework. Web server gateway interface, sometimes known as WSGI, is a standard for creating web applications in Python. It is regarded as the specification for the common interface between the web application and server. Jinja2 is a web template engine that renders dynamic web pages by fusing a template with a specific data source. You can install Flask using **pip**:

```
pip install flask==2.2.2
```

For building interactive visualizations for current web browsers, the Python library **Bokeh** is a good choice. It enables you to create stunning visualizations, from straightforward plots to intricate dashboards with streaming datasets. Without writing any JavaScript yourself, you may build visualizations that are powered by JavaScript using Bokeh. you can install Bokeh using pip:

```
pip install bokeh==3.0.1
```

Another way to add graphs to Flask apps is by using open-source JS charting libraries like **Chart.js**. We can pass the required data to create these charts

from the Flask app. To include Chart.js in your HTML website, you can use the following CDN –

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

Add Graphs to Flask using JavaScript Chart.js Library

Step 1: Here, we have defined two variables, namely, 'labels' and 'data'. These variables are passed to the HTML template, ***chartjs-example.html***. This HTML template as defined below should be placed in the ***templates*** directory of the root folder with the mentioned name.

Python3

```
# Importing required functions
from flask import Flask, render_template

# Flask constructor
app = Flask(__name__)

# Root endpoint
@app.route('/')
def homepage():

    # Define Plot Data
    labels = [
        'January',
        'February',
        'March',
        'April',
        'May',
        'June',
    ]

    data = [0, 10, 15, 8, 22, 18, 25]

    # Return the components to the HTML template
    return render_template(
        template_name_or_list='chartjs-example.html',
        data=data,
        labels=labels,
    )
```

```
# Main Driver Function
if __name__ == '__main__':
    # Run the application on the local development server ##
    app.run(debug=True)
```

Step 2: Within our HTML template, we use the Jinja2 notation, i.e., to mention the Python list variables we use the syntax **{{ variable_name | tojson }}**. This will pass the value from the Flask app to the HTML template and particularly our JS script. Therefore, any data that is prepared within the Flask app can be leveraged in the open-source JS charting libraries.

HTML

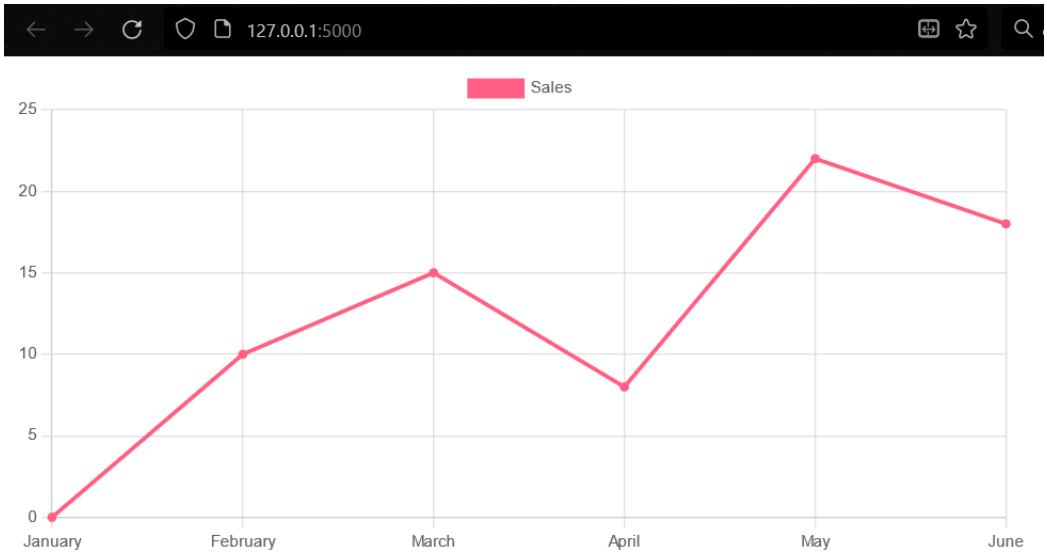
```
<!DOCTYPE html>
<html lang="en">

<head>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <title>Chartjs Charts</title>
</head>

<body>
    <div style="height: 50vh; width: 50%;">
        <canvas id="myChart"></canvas>
    </div>

    <script>
        const labels = {{ labels | tojson}};

        const data = {
            labels: labels,
            datasets: [{
                label: 'Sales',
                backgroundColor: 'rgb(255, 99, 132)',
                borderColor: 'rgb(255, 99, 132)',
                data: {{ data | tojson}},
            }]
        };

        const config = {
            type: 'line',
            data: data,
            options: { maintainAspectRatio: false }
        };
```

```
        const myChart = new Chart(
            document.getElementById('myChart'),
            config
        );

    </script>
</body>
</html>
```

Step 3: To run the above Flask code, we can use the following command (assuming your Flask filename is **app.*py***)

```
python app.py
```

Output:



Add Graphs to Flask using Bokeh

The Bokeh library has a ***components()*** method which returns HTML components to embed a Bokeh plot. The data for the plot is stored directly in the returned HTML. The returned components assume that BokehJS resources are already loaded. The HTML document or template in which they will be embedded needs to include script tags, either from a local URL or Bokeh's CDN.

```
<script src="https://cdn.bokeh.org/bokeh/release/bokeh-3.0.1.min.js">
</script>
```

> ***Syntax:*** *components(models: Model | Sequence[Model] | dict[str, Model], wrap_script: bool = True, wrap_plot_info: bool = True, theme: ThemeLike = None)*
>
> ***Parameters:***
>
> - ***models*** *(Model|list|dict|tuple: A single Model, a list/tuple of Models, or a dictionary of keys and Models.*
> - ***wrap_script*** *(boolean, optional): If True, the returned javascript is wrapped in a script tag. (default: True)*
> - ***wrap_plot_info*** *(boolean, optional) : If True, returns strings. Otherwise, return RenderRoot objects that can be used to build your own divs. (default: True)*
> - ***theme****: constitute the full set of roots of a document, applies the theme of that document to the components. Otherwise applies the default theme.*

After defining the Bokeh figure and the Bokeh scatter plot, we extract the 'script' and 'div' components from the figure to use in the HTML. The 'script' variable holds the JS code embedded in an HTML <script> tag. As for the 'div' variable, it contains a single HTML tag which holds the component. These two HTML codes can be utilized with the required Bokeh imports to add the graphs in our Flask apps.

Python3

```
# Importing required functions
import random

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

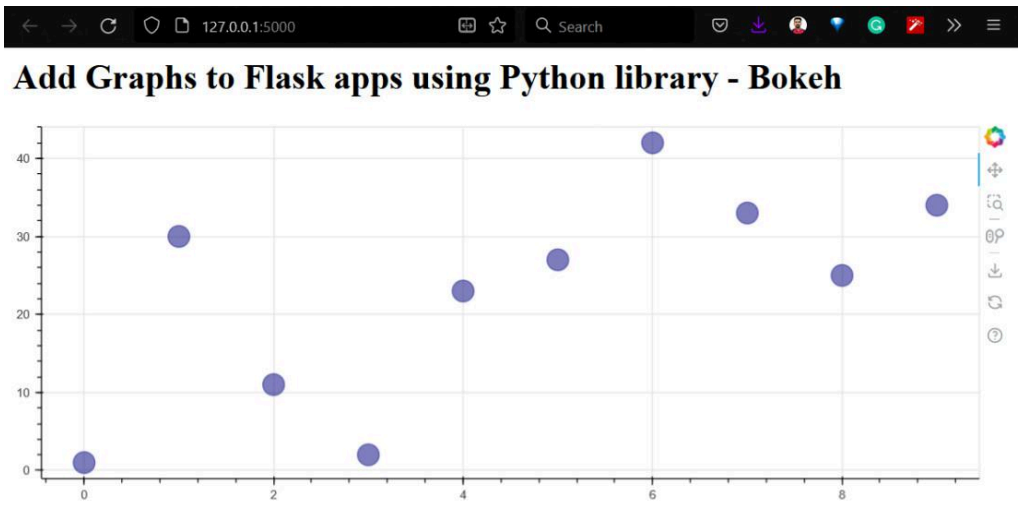
```
# Flask constructor
app = Flask(__name__)

# Root endpoint
@app.route('/')
def homepage():

    # Creating Plot Figure
    p = figure(height=350, sizing_mode="stretch_width")

    # Defining Plot to be a Scatter Plot
    p.circle(
        [i for i in range(10)],
        [random.randint(1, 50) for j in range(10)],
        size=20,
        color="navy",
        alpha=0.5
    )

    # Get Chart Components
    script, div = components(p)

    # Return the components to the HTML template
    return f'''
    <html lang="en">
        <head>
            <script src="https://cdn.bokeh.org/bokeh/release/bokeh-3.0.1.min.js"><
            <title>Bokeh Charts</title>
        </head>
        <body>
            <h1>Add Graphs to Flask apps using Python library - Bokeh</h1>
            { div }
            { script }
        </body>
    </html>
    '''


# Main Driver Function
if __name__ == '__main__':
    # Run the application on the local development server
    app.run(debug=True)
```

To run the above Flask code, we can use the following command (assuming your Flask filename is ***bokeh_charts.py***)

```
python bokeh_charts.py
```

Output:

The output when viewed on the browser through the link http://127.0.0.1:5000/ will look like this



Next Article

How to Run a Flask Application

Similar Reads

Documenting Flask Endpoint using Flask-Autodoc

Documentation of endpoints is an essential task in web development and being able to apply it in different frameworks is always a utility. This article discusses...

4 min read

How to use Flask-Session in Python Flask ?

Flask Session - Flask-Session is an extension for Flask that supports Server-side Session to your application.The Session is the time between the client logs in to...

4 min read

How to Integrate Flask-Admin and Flask-Login

In order to merge the admin and login pages, we can utilize a short form or any other login method that only requires the username and password. This is know...

8 min read

Minify HTML in Flask using Flask-Minify

Flask offers HTML rendering as output, it is usually desired that the output HTML should be concise and it serves the purpose as well. In this article, we would...

12 min read

Flask URL Helper Function - Flask url_for()

In this article, we are going to learn about the flask url_for() function of the flask URL helper in Python. Flask is a straightforward, speedy, scalable library, used f...

11 min read

Add User and Display Current Username in Flask

In this article, we'll talk about how to add a User and Display the Current Username on a Flask website. When we log in using our username and...

9 min read

How To Add Authentication to Your App with Flask-Login

Whether it is building a simple blog or a social media site, ensuring user sessions are working correctly can be tricky. Fortunately, Flask-Login provides a simplifie...

9 min read

Python | Visualize graphs generated in NetworkX using Matplotlib

Prerequisites: Generating Graph using Network X, Matplotlib IntroIn this article, we will be discussing how to plot a graph generated by NetworkX in Python...

3 min read

Visualize Graphs in Python

Prerequisites: Graph Data Structure And Algorithms A Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred...

2 min read

Styling Graphs in Pygal

Pygal is a Python module that is mainly used to build SVG (Scalar Vector Graphics) graphs and charts. SVG is a vector-based graphics in the XML format...

2 min read

Article Tags : Python Technical Scripter Technical Scripter 2022

Practice Tags : python

OOAD
System Design Bootcamp
Interview Questions

Puzzles

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects