Show Data on Google Pie Chart using Python Flask

Last Updated : 30 Dec, 2022

In this article, we are going to build a mini project by using a Flask using Python and that will demonstrate to us the use of a google pie chart. A single web page on which input will be taken from users e.g an activity for how long they are performing. After clicking on submit it will generate a pie chart. It can be integrated into our application by putting javascript code into the page on which we are going to show the pie chart. Let's dive into the project. follow each and every step properly.

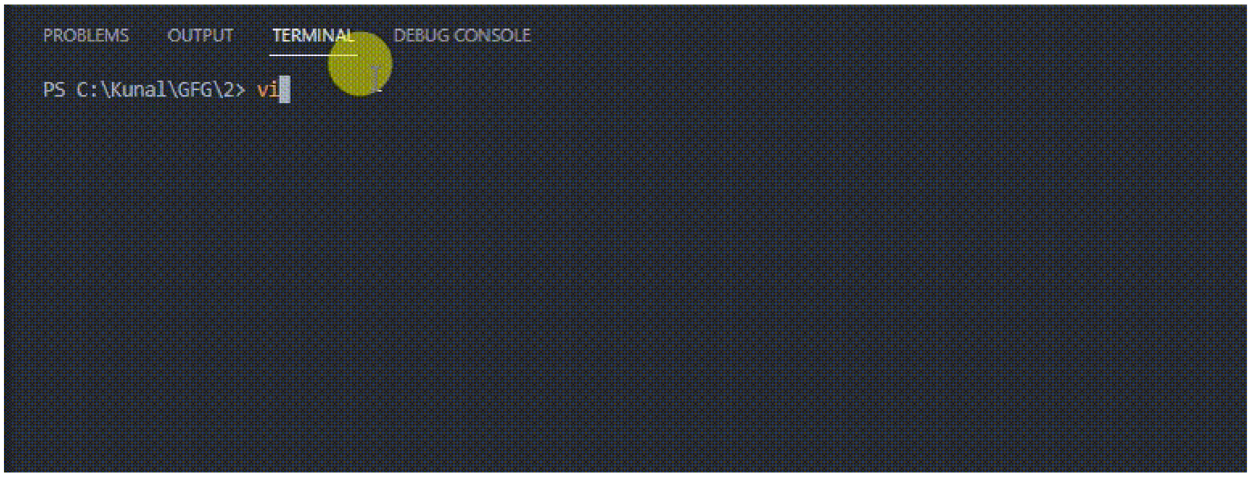
What is Google Pie Chart?

The pie chart is a great way to represent data in numerical proportions. It's a circular disk graphic on that data represented by different slices and colors. Google has provided these great tools for free, Not just only pie chart makers but it includes all other visualization ways. Mainly Our focus will be only on the pie chart. It has some types of 3D pie charts, Donut charts will see them while building the project.

Step to Implement Google Pie Chart using Python Flask

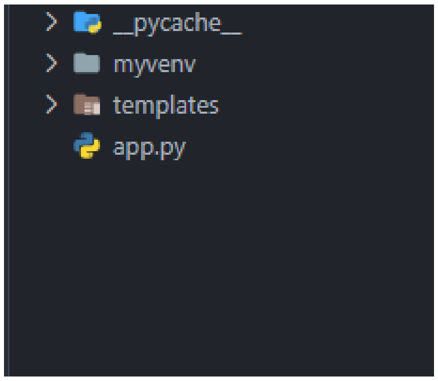
Step 1: Create a Virtual environment

Step 2: Create one project folder and open it in vs code. Enter the command in the terminal "virtual env". This will create a virtual environment for us and once it's done, activate the "env/Scripts/activate.ps1" command in the terminal.



```
PS C:\Kunal\GFG\2> vi
```

Step 3: The next step is to create a template folder. You should get a folder structure like this:



```
> __pycache__
> myvenv
> templates
  app.py
```

Step 4: Getting Input Data From Users

We required an HTML file for the form and a function to take the data. Create a form.html file inside the templates folder. Add the following codes to the respective files.

Python3

```
<!DOCTYPE html >
<html lang = "en" >
<head >
    <meta charset = "UTF-8" >
    <meta http-equiv = "X-UA-Compatible" content = "IE=edge" >
    <meta name = "viewport" content = "width=device-width, initial-scale=1.0" >
    <title > Document < /title >
</head >
<body >

    <form action = "{{ url_for("get_data__show_chart")}}" method = "post" >
    <div >
       <table >
        <tr >
           <td >
                <input type = "text" placeholder = "Title of pie chart" name = "
            </td >
            <td >
                <input type = "height" placeholder="height" name="ht" ></tr>
            </td >
            <td >
                <input type = "width" placeholder="width" name="wt" ></tr>
            </td >
        </tr >
        <tr >
            <th > Daily Activities</th>
            <th > Time in Hours</th>
        </tr >
        <tr >
            <td > <input type="text" name="act1" placeholder="Activity 1"></td>
            <td > <input type="number" name="t1" placeholder="Time 1"></td>
        </tr >
        <tr >
            <td > <input type="text"  name="act2" placeholder="Activity 2"></td>
            <td > <input type="number" name="t2" placeholder="Time 2"></td>
        </tr >
        <tr >
            <td > <input type="text" name="act3" placeholder="Activity 3"></td>
            <td > <input type="number" name="t3" placeholder="Time 3"></td>
        </tr >
    </table >
    </div >
    <button type = "submit" style="display:block;">Submit</button>
</form >
</body >
</html >
```

Step 5: The get_data__show_chart function will receive the post request that is the user data to proceed further and display the chart by rendering the index.html file else if it does not receive a post request then it will render the form.html file.

Python3

```
from flask import Flask, render_template, request
```

```
app = Flask(__name__)


@app.route("/", methods=['GET', 'POST'])
def get_data__show_chart():
    if request.method == 'POST':
        title = request.form.get('title')
        height = request.form.get('ht')
        width = request.form.get('wt')

        activity_1 = request.form.get('act1')
        time1 = request.form.get('t1')

        activity_2 = request.form.get('act2')
        time2 = request.form.get('t2')

        activity_3 = request.form.get('act3')
        time3 = request.form.get('t3')

        return render_template('chart.html',
                               act1=activity_1,
                               act2=activity_2,
                               act3=activity_3,
                               t1=time1,
                               t2=time2, t3=time3,
                               ht=height, wt=width,
                               title=title)

    return render_template('form.html', name='form')
```

Step 6: Data Representation using Pie Charts

We have to create chart.html which contains the javascript code to generate a pie chart from the input data that we got from the form.html file. Pie charts have 3 types:

1. Simple chart
2. 3D chart
3. Donut Chart

google.charts.load('current', {'packages':['corechart']}): google pie chart tool required a core chart package so it will load it.

drawcharts(): Includes the code to add the data, the no of columns and rows, and how should it be represented on the chart that information we have to provide it.

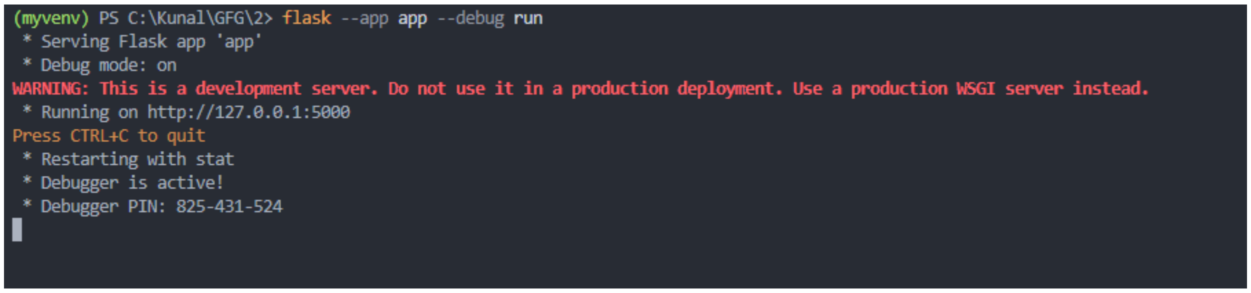
To create a 3D chart we just have to add **is3D: true** in the options. Similarly, all other data and options will remain the same. To create a Donut chart we have to add **pieHole: 0.4** inside the options object.

HTML

```
# templates/Chart.html

<html >
   <head >
        <h1 > Data Representation by using Pie charts < /h1 >
     </head >
     <body >
        <div style = "border: 1px solid black;">
            <div id = "simple_chart"></div>
            <div id = "3dchart"></div>
            <div id = "donut_chart"></div>
        </div >
        <script type = "text/javascript"  src="https://www.gstatic.com/charts/load
        <script type = "text/javascript">

            google.charts.load('current', {'packages': ['corechart']})

            google.charts.setOnLoadCallback(drawCharts)

            function drawCharts() {
                var data = new google.visualization.DataTable()
              data.addColumn('string', 'Daily Activity')
              data.addColumn('number', 'Hours');
              data.addRows([
                  ['{{act1}}', {{t1}}],
                  ['{{act2}}', {{t2}}],
                  ['{{act3}}', {{t3}}]
              ]);

              var options_1 = {
                  'title': 'Simple Chart',
                  'width': '{{wt}}',
                             'height': '{{ht}}'
              }

              var options_2 = {
                  'title': '3D Chart',
                            'width': '{{wt}}',
                            'height': '{{ht}}',
                            'is3D': true,
              }

              var options_3 = {
                  'title': 'Donut Chart',
                  'width': '{{wt}}',
                          'height': '{{ht}}',
                          'pieHole': 0.4,
              }

              var simple_chart = new google.visualization.PieChart(document.getEle
              simple_chart.draw(data, options_1);


              var _3dchart = new google.visualization.PieChart(document.getElement
              _3dchart.draw(data, options_2);

              var donut_chart = new google.visualization.PieChart(document.getElem
              donut_chart.draw(data, options_3);             }
        </script >
    </body >
</html >
```

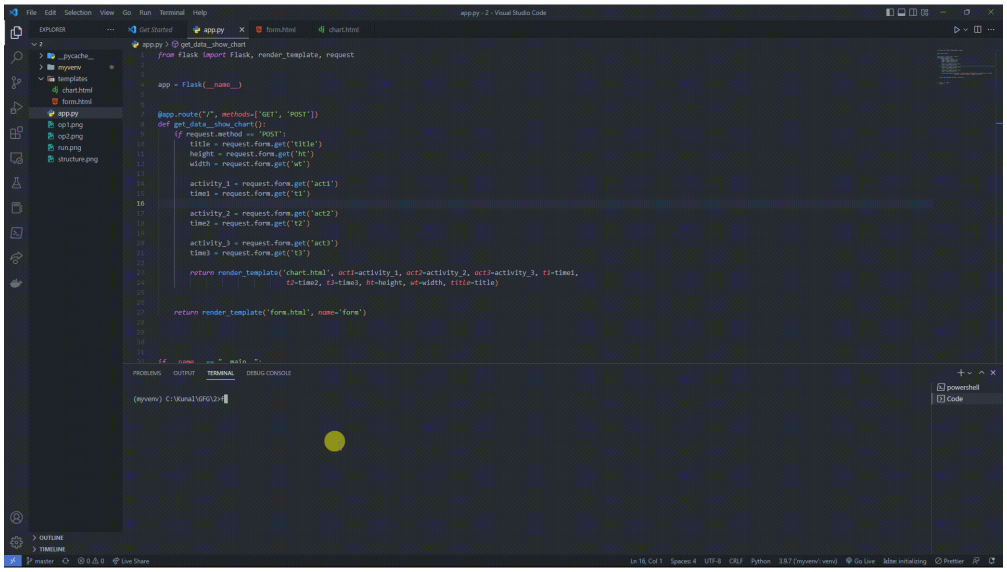
To run our project activate the virtual environment and enter the following command to run the application. Then visit the URL to check out the results

```
flask --app app_name --debug run
```



```
(myvenv) PS C:\Kunal\GFG\2> flask --app app --debug run
 * Serving Flask app 'app'
 * Debug mode: on
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
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 825-431-524
```

Output:



```
from flask import Flask, render_template, request


app = Flask(__name__)


@app.route("/", methods=['GET', 'POST'])
def get_data__show_chart():
    if request.method == 'POST':
        title = request.form.get('title')
        height = request.form.get('ht')
        width = request.form.get('wt')

        activity_1 = request.form.get('act1')
        time1 = request.form.get('t1')

        activity_2 = request.form.get('act2')
        time2 = request.form.get('t2')

        activity_3 = request.form.get('act3')
        time3 = request.form.get('t3')

        return render_template('chart.html', act1=activity_1, act2=activity_2, act3=activity_3, t1=time1,
                               t2=time2, t3=time3, ht=height, wt=width, title=title)


    return render_template('form.html', name='form')




if __name__ == "__main__":
```

```
(myvenv) C:\Kunal\GFG\2>f
```

kunal... 2

Previous Article **Next Article**

Python | Plotting charts in excel sheet using openpyxl module | Set 3

Similar Reads

Google Pie Chart using Flask

In this article, we will learn how to show data on a Google line chart using Python Flask. Python Flask is a popular web framework that allows us to create...

2 min read

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

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

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

Python Bokeh - Making a Pie Chart

Bokeh is a Python interactive data visualization. It renders its plots using HTML and JavaScript. It targets modern web browsers for presentation providing...

3 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

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

How to set border for wedges in Matplotlib pie chart?

Pie charts can be used for relative comparison of data. Python offers several data visualization libraries to work with. The Matplotlib library offers different types ...

3 min read

Multi-Series Pie Chart 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

Pie chart 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

Deep Learning

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

Inteview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
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