

(1) ChatGPT prompt and interaction

You

寫一個程式可以隨機生成溫溼度數據資料並呈現在前端網頁上,用Highcharts表示數據變化

< 4/4 >



ChatGPT

以下是一個簡單的Python程式，它可以生成隨機的溫度和濕度數據，並通過Flask將這些數據呈現在前端網頁上，使用Highcharts來表示數據變化。

首先，確保你已經安裝了相關的庫：Flask和Highcharts。

bash

Copy code

```
pip install flask
```

在安裝Highcharts之前，你需要在HTML頁面中引入Highcharts的庫，你可以在Highcharts官方網站（<https://www.highcharts.com/download>）上下載相應的庫文件，然後將它們放置在你的項目中。

以下是程式碼：

```
python                                                                    Copy code

# app.py
from flask import Flask, render_template, jsonify
import random
import time

app = Flask(__name__)

# Function to generate random data
def generate_random_data():
    while True:
        humidity = random.uniform(40, 60) # Random humidity between 40% and 60%
        temperature = random.uniform(20, 30) # Random temperature between 20°C and 30°C
        yield {'humidity': humidity, 'temperature': temperature}
        time.sleep(2) # Wait for 2 seconds before next iteration

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/data')
def get_data():
    return jsonify(next(generate_random_data()))
```

```
def get_data():
    return jsonify(next(generate_random_data()))

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

接下來，創建一個名為`templates`的文件夾，在其中創建一個名為`index.html`的HTML文件，並添加以下內容：

```
html                                                                    Copy code

<!-- templates/index.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Temperature and Humidity Data</title>
  <!-- Include Highcharts library -->
  <script src="path/to/highcharts.js"></script>
</head>
<body>
  <div id="container" style="width:100%; height:400px;"></div>
  <script>
    // Function to update chart with new data
```

```

    var chart = Highcharts.chart('container', {
      chart: {
        type: 'line'
      },
      title: {
        text: 'Temperature and Humidity Data'
      },
      xAxis: {
        categories: ['Temperature', 'Humidity']
      },
      series: [{
        name: 'Value',
        data: [data.temperature, data.humidity]
      }]
    });
  });
}

// Update chart every 2 seconds
setInterval(updateChart, 2000);

// Initial chart update
updateChart();
</script>
</body>
</html>

```

在這個HTML文件中，我們引入了Highcharts庫並創建了一個圖表容器。通過JavaScript，我們使用AJAX請求從`/data`路由獲取數據，然後用Highcharts來更新圖表。我們使用`setInterval`函數來每2秒刷新一次圖表。

確保將Highcharts庫文件的路徑正確指定為`index.html`中的`<script>`標籤的`src`屬性。這樣，當你運行Flask應用程序時，你將能夠在網頁上看到動態更新的溫度和濕度數據。

🔊 📄 ↺ 🔒

Is this conversation helpful so far?



(2) implementation code

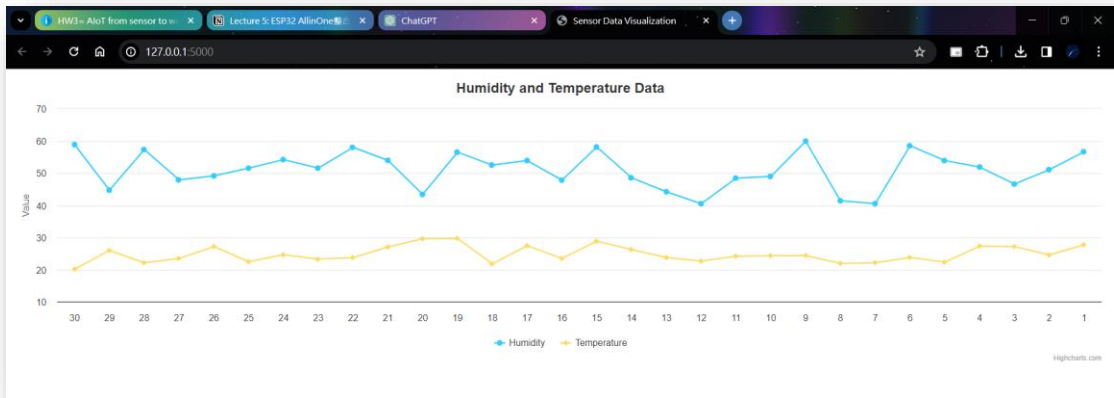
```
app.py x
app.py > ...
1  from flask import Flask, render_template, jsonify
2  import sqlite3
3  import random
4  import time
5
6  app = Flask(__name__)
7
8  # Function to generate random data and store it in the database
9  def generate_and_store_data():
10     # Connect to the SQLite database
11     conn = sqlite3.connect('sensors.db')
12     cursor = conn.cursor()
13     while True:
14         # Generate random humidity between 40% and 60%
15         humidity = random.uniform(40, 60)
16         # Generate random temperature between 20°C and 30°C
17         temperature = random.uniform(20, 30)
18         # Insert the generated data into the 'sensor_data' table
19         cursor.execute("INSERT INTO sensor_data (humidity, temperature) VALUES (?, ?)", (humidity, temperature))
20         conn.commit()
21         # Wait for 2 seconds before next iteration
22         time.sleep(2)
23
24 # Route to display the web page
25 @app.route('/')
26 def index():
27     return render_template('index.html')
28
29 # Route to fetch data from the database
30 @app.route('/data')
31 def get_data():
32     # Connect to the SQLite database
33     conn = sqlite3.connect('sensors.db')
34     cursor = conn.cursor()
35     # Select the latest 30 entries of humidity and temperature from 'sensor_data' table
36     cursor.execute("SELECT humidity, temperature FROM sensor_data ORDER BY id DESC LIMIT 30")
37     # Fetch the selected data
```

```
40
41 if __name__ == '__main__':
42     # Create the database schema if it doesn't exist
43     conn = sqlite3.connect('sensors.db')
44     cursor = conn.cursor()
45     # Create the 'sensor_data' table with columns id, humidity, and temperature
46     cursor.execute('''CREATE TABLE IF NOT EXISTS sensor_data (
47         id INTEGER PRIMARY KEY AUTOINCREMENT,
48         humidity REAL,
49         temperature REAL)''')
50     conn.commit()
51     conn.close()
52
53     # Start a new thread to generate and store data
54     import threading
55     threading.Thread(target=generate_and_store_data).start()
56
57     # Run the Flask app
58     app.run(debug=True)
59
```

```
app.py x index.html x
templates > index.html > html > body > script > fetchDataAndPlot > then() callback > series > color
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Sensor Data Visualization</title>
5   <script src="https://code.highcharts.com/highcharts.js"></script>
6 </head>
7 <body>
8   <div id="container" style="min-width: 310px; height: 400px; margin: 0 auto"></div>
9
10  <script>
11    var chart;
12
13    function fetchDataAndPlot() {
14      fetch('/data')
15        .then(response => response.json())
16        .then(data => {
17          var humidityData = [];
18          var temperatureData = [];
19          data.reverse(); // Reverse data to display in chronological order
20          data.forEach(entry => {
21            humidityData.push(entry[0]);
22            temperatureData.push(entry[1]);
23          });
24
25          if (!chart) {
26            chart = Highcharts.chart('container', {
27              chart: {
28                type: 'line'
29              },
30              title: {
31                text: 'Humidity and Temperature Data'
32              },
33              xAxis: {
34                categories: Array.from({ length: data.length }, (_, i) => i + 1).reverse()
35              },
36              yAxis: {
37                title: {
```

```
36              title: {
37                text: 'Value'
38              }
39            },
40            series: [{
41              name: 'Humidity',
42              data: humidityData,
43              color: '#33CEFF' // Change color here
44            }, {
45              name: 'Temperature',
46              data: temperatureData,
47              color: '#FFDB5F' // Change color here
48            }]
49          });
50        } else {
51          chart.series[0].setData(humidityData);
52          chart.series[1].setData(temperatureData);
53          chart.xAxis[0].setCategories(Array.from({ length: data.length }, (_, i) => i + 1).reverse());
54        }
55      }
56    });
57  }
58
59  setInterval(fetchDataAndPlot, 2000); // Fetch data every two seconds
60  fetchDataAndPlot(); // Fetch data initially
61 </script>
62 </body>
63 </html>
64
```

(3) demo result



Sqlite viewer

sensors.db

Reset Filters Records: 4600

	id	humidity	temperature
1	1	54.25279670103254	26.489754571401114
2	2	47.347042650643736	24.532772772413526
3	3	56.25675871596923	20.34643872587992
4	4	43.37391790169081	21.64214614485672
5	5	45.235885604361144	26.87807604123344
6	6	46.95845349758106	24.27421274191447
7	7	54.750133530232304	20.410316063934026
8	8	48.359876587406134	23.356827419258952
9	9	46.13120872753972	23.68503357218756
10	10	48.31404470364625	23.215041230095935
11	11	51.36816409301258	24.23769324368167
12	12	42.35252731584549	24.152796663126825
13	13	50.97854213607711	20.84178075267851
14	14	54.501063718382255	28.42939982466941

Page 1 / 46

terminal

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
127.0.0.1	-	-	[20/Mar/2024 16:19:20] "GET /data HTTP/1.1" 200 -	
127.0.0.1	-	-	[20/Mar/2024 16:19:22] "GET /data HTTP/1.1" 200 -	
127.0.0.1	-	-	[20/Mar/2024 16:19:24] "GET /data HTTP/1.1" 200 -	
127.0.0.1	-	-	[20/Mar/2024 16:19:26] "GET /data HTTP/1.1" 200 -	
127.0.0.1	-	-	[20/Mar/2024 16:19:28] "GET /data HTTP/1.1" 200 -	
127.0.0.1	-	-	[20/Mar/2024 16:19:30] "GET /data HTTP/1.1" 200 -	
127.0.0.1	-	-	[20/Mar/2024 16:19:32] "GET /data HTTP/1.1" 200 -	
127.0.0.1	-	-	[20/Mar/2024 16:19:34] "GET /data HTTP/1.1" 200 -	
127.0.0.1	-	-	[20/Mar/2024 16:19:36] "GET /data HTTP/1.1" 200 -	
127.0.0.1	-	-	[20/Mar/2024 16:19:38] "GET /data HTTP/1.1" 200 -	
127.0.0.1	-	-	[20/Mar/2024 16:19:40] "GET /data HTTP/1.1" 200 -	
127.0.0.1	-	-	[20/Mar/2024 16:19:42] "GET /data HTTP/1.1" 200 -	
127.0.0.1	-	-	[20/Mar/2024 16:19:54] "GET /data HTTP/1.1" 200 -	