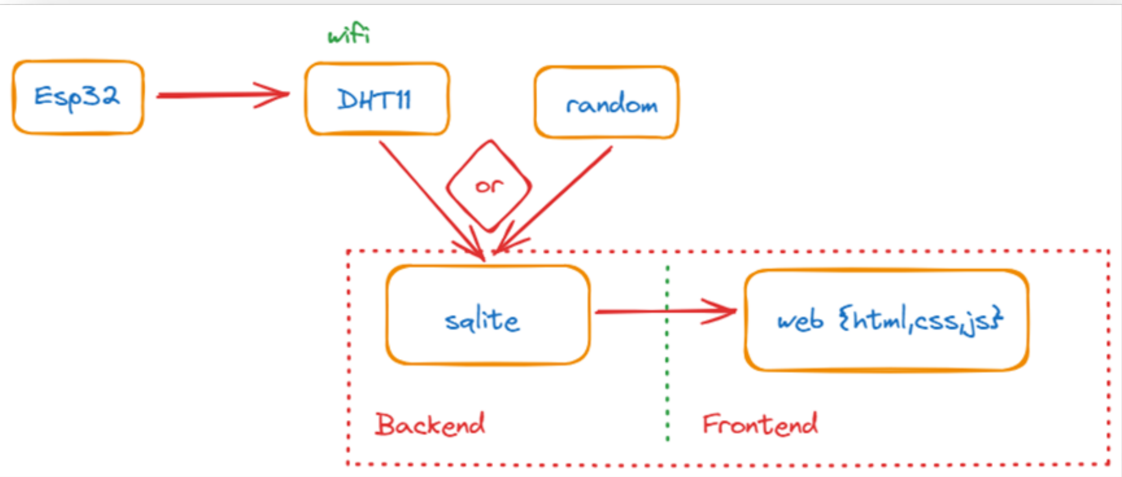
Hw3

資工三 4110056030

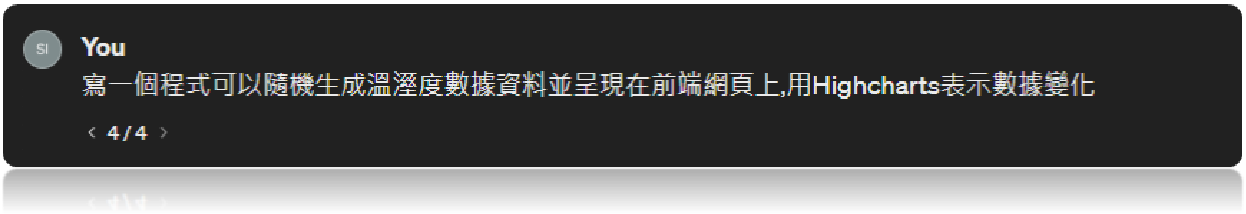
hw3流程圖示

> excalidraw



1. ChatGPT prompt and interaction

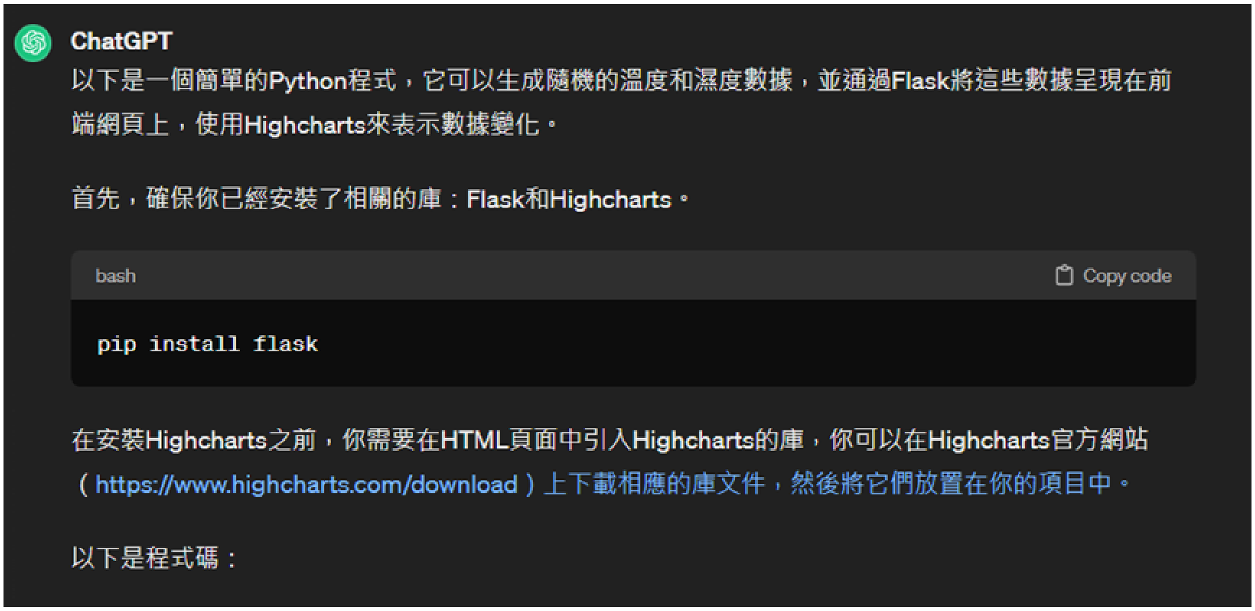
> my question



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

answer by chtgpt



ChatGPT

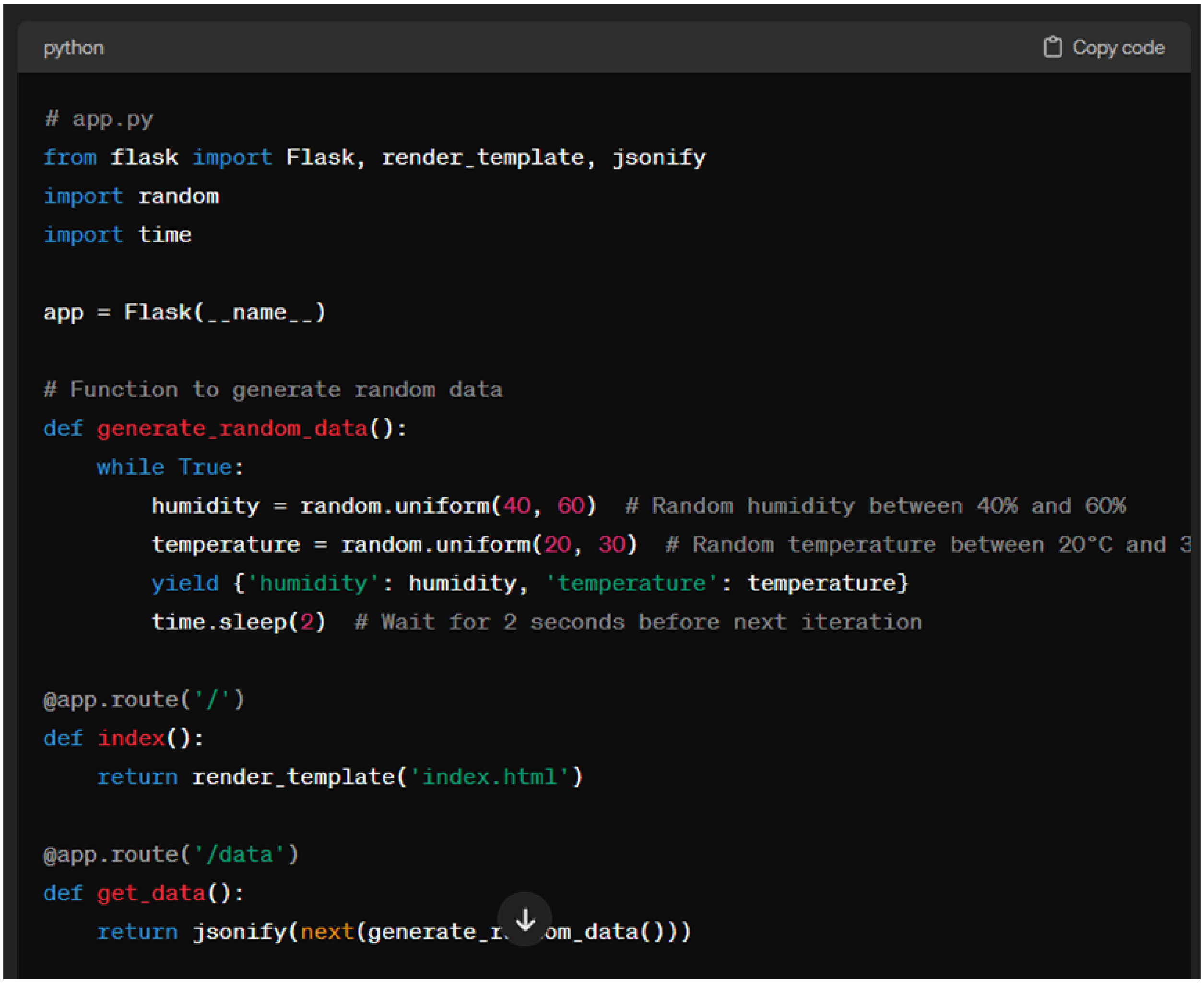
以下是一個簡單的Python程式，它可以生成隨機的溫度和濕度數據，並通過Flask將這些數據呈現在前端網頁上，使用Highcharts來表示數據變化。

首先，確保你已經安裝了相關的庫：Flask和Highcharts。

```bash
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：



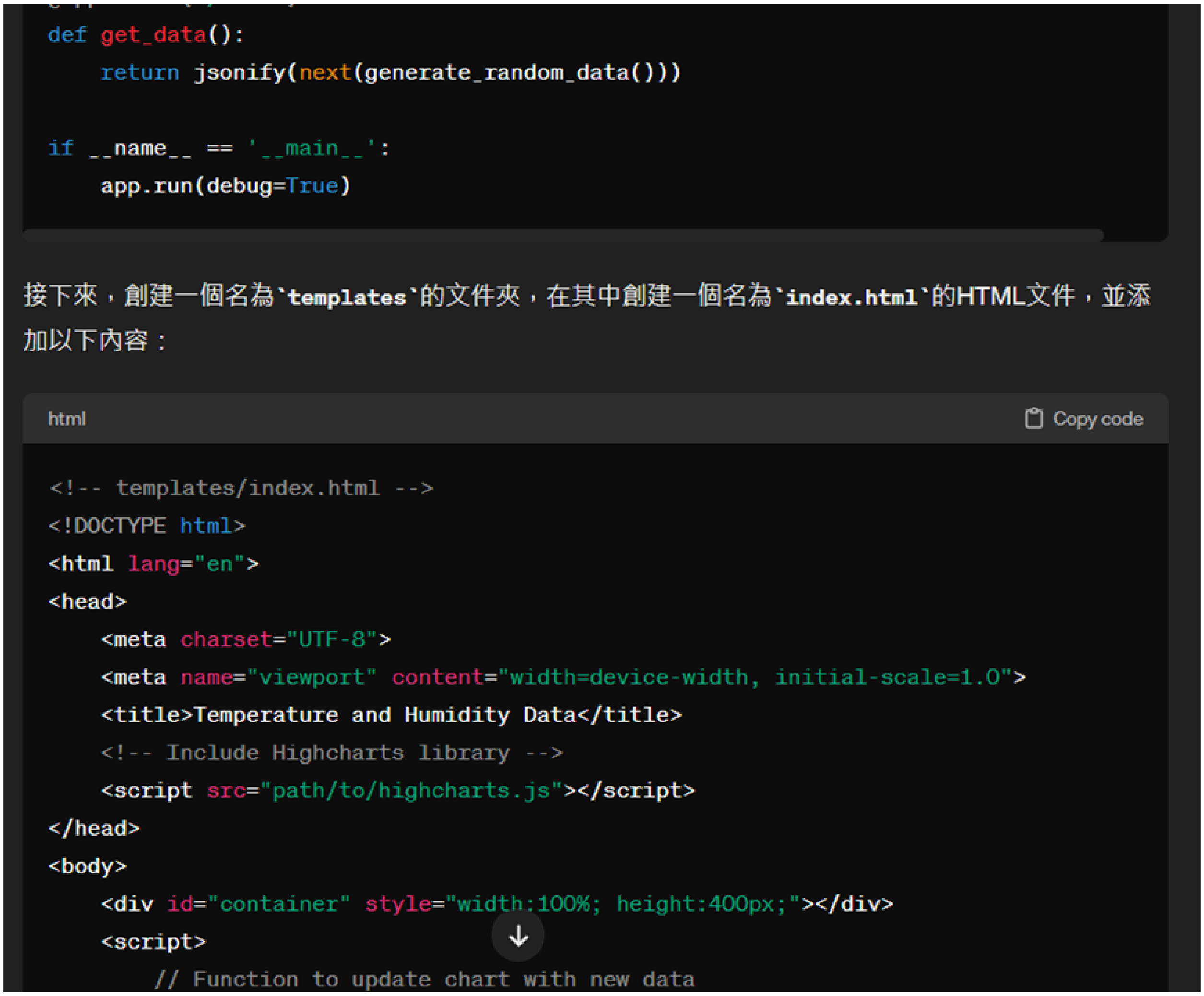
```python
# app.py
from flask import Flask, render_template, jsonify
import random
import time

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 3
        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_r  om_data()))
```

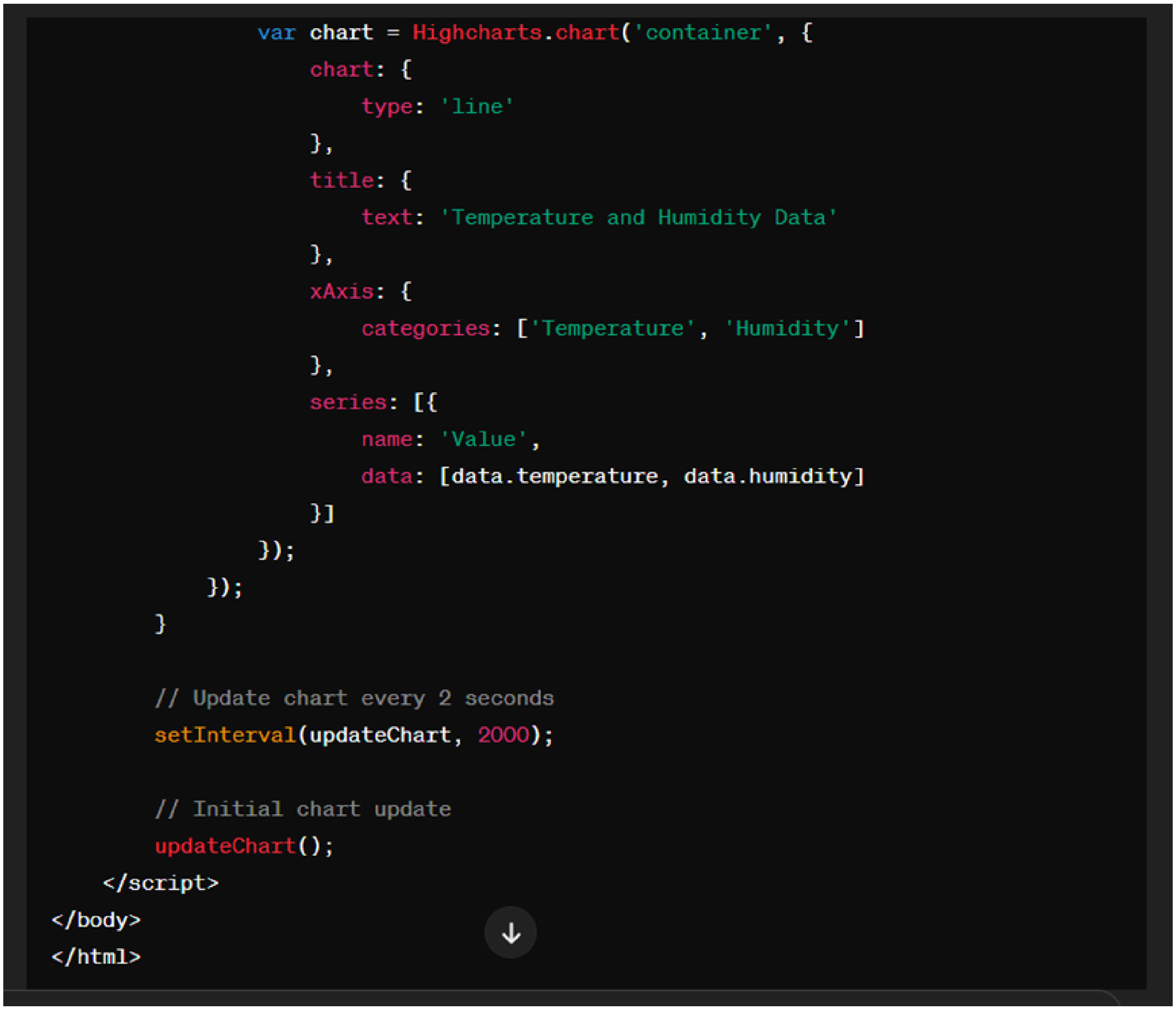


```
def get_data():
    return jsonify(next(generate_random_data()))

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：

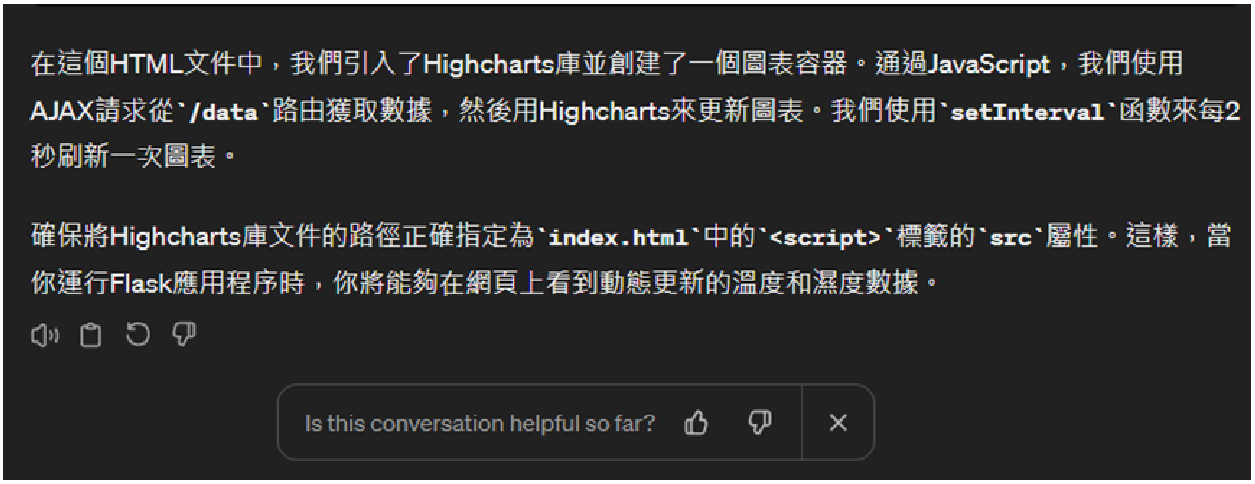
```html
<!-- templates/index.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Temperature and Humidity Data</title>
    <!-- Include Highcharts library -->
    <script src="path/to/highcharts.js"></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。

2. implementation code

> 兩種

▼ Random產生數值版本

- app.py

```
from flask import Flask, render_template, jsonify
import sqlite3
import random
import time

app = Flask(__name__)

# Function to generate random data and store it in the databa
def generate_and_store_data():
    # Connect to the SQLite database
    conn = sqlite3.connect('sensors.db')
    cursor = conn.cursor()
    while True:
        # Generate random humidity between 40% and 60%
        humidity = random.uniform(40, 60)
        # Generate random temperature between 20°C and 30°C
        temperature = random.uniform(20, 30)
        # Insert the generated data into the 'sensor_data' ta
        cursor.execute("INSERT INTO sensor_data (humidity, te
        conn.commit()
        # Wait for 2 seconds before next iteration
        time.sleep(2)

# Route to display the web page
@app.route('/')
def index():
    return render_template('index.html')
```

```
# Route to fetch data from the database
@app.route('/data')
def get_data():
    # Connect to the SQLite database
    conn = sqlite3.connect('sensors.db')
    cursor = conn.cursor()
    # Select the latest 30 entries of humidity and temperatu
    cursor.execute("SELECT humidity, temperature FROM sensor_
    # Fetch the selected data
    data = cursor.fetchall()
    return jsonify(data)

if __name__ == '__main__':
    # Create the database schema if it doesn't exist
    conn = sqlite3.connect('sensors.db')
    cursor = conn.cursor()
    # Create the 'sensor_data' table with columns id, humidit
    cursor.execute('''CREATE TABLE IF NOT EXISTS sensor_data
                      id INTEGER PRIMARY KEY AUTOINCREMENT,
                      humidity REAL,
                      temperature REAL)''')
    conn.commit()
    conn.close()

    # Start a new thread to generate and store data
    import threading
    threading.Thread(target=generate_and_store_data).start()

    # Run the Flask app
    app.run(debug=True)
```

- index.html

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Sensor Data Visualization</title>
    <script src="https://code.highcharts.com/highcharts.js">
</head>
<body>
    <div id="container" style="min-width: 310px; height: 400

    <script>
        var chart;

        function fetchDataAndPlot() {
            fetch('/data')
                .then(response => response.json())
                .then(data => {
                    var humidityData = [];
                    var temperatureData = [];
                    data.reverse(); // Reverse data to displa
                    data.forEach(entry => {
                        humidityData.push(entry[0]);
                        temperatureData.push(entry[1]);
                    });

                    if (!chart) {
                        chart = Highcharts.chart('container'
                            chart: {
                                type: 'line'
                            },
                            title: {
                                text: 'Humidity and Temperat
                            },
                            xAxis: {
                                categories: Array.from({ len
                            },
                            yAxis: {
                                title: {
                                    text: 'Value'
                                }
                            },
                            series: [{
                                name: 'Humidity',
                                data: humidityData,
                                color: '#33CEFF' // Change co
                            }, {
                                name: 'Temperature',
                                data: temperatureData,
                                color: 'yellow' // Change col
                            }]
                        });
                    } else {
                        chart.series[0].setData(humidityData
                        chart.series[1].setData(temperatureDa
                        chart.xAxis[0].setCategories(Array.fr
                    }
                });
        }

        setInterval(fetchDataAndPlot, 2000); // Fetch data ev
        fetchDataAndPlot(); // Fetch data initially
    </script>
</body>
</html>
```

▼ 實際數值版本

- DHT11.ino

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <DHT.h>

#define WIFI_SSID "網路名"
#define WIFI_PASSWORD "密碼"
```

```
#define SERVER_IP "http://192.168.227.14:5000/update_data"
#define SERVER_PORT 5000
#define SENSOR_PIN 2
#define DHTTYPE DHT11

DHT dht(SENSOR_PIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
  dht.begin();
}

void loop() {
  delay(2000);
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor");
    return;
  }
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");
  sendSensorData(temperature, humidity);
}

void sendSensorData(float temperature, float humidity) {
  WiFiClient client;
  HTTPClient http;

  String data = "{\"temperature\":" + String(temperature) + 

  if (http.begin(client,String(SERVER_IP))) {
    http.addHeader("Content-Type", "application/json");
    int httpResponseCode = http.POST(data);
    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println(response);
    } else {
      Serial.print("Error on sending POST: ");
      Serial.println(http.errorToString(httpResponseCode).c_
    }
    http.end();
  } else {
    Serial.println("Unable to connect to server");
  }
}
```

- app.py

```
# app.py

from flask import Flask, render_template, jsonify, request
import sqlite3
from datetime import datetime

app = Flask(__name__)
conn = sqlite3.connect('sensor_data.db', check_same_thread=Fa
c = conn.cursor()

c.execute('''CREATE TABLE IF NOT EXISTS sensor_data
             (id INTEGER PRIMARY KEY,
              temperature REAL,
              humidity REAL,
              timestamp TEXT)''')

@app.post('/update_data')
def update_data():
    data = request.json
    temperature = data.get('temperature')
    humidity = data.get('humidity')
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    print(f"Received data: temperature={temperature},humidit
    c.execute("INSERT INTO sensor_data (temperature, humidit
    conn.commit()
    return 'Data received and stored successfully'

@app.route('/api/data')
def get_sensor_data():
    c.execute("SELECT * FROM sensor_data ORDER BY timestamp 
    rows = c.fetchall()
    data = [{'timestamp': row[3], 'temperature': row[1], 'hu
    return jsonify(data)

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

if __name__ == '__main__':
    app.run(host = "0.0.0.0",port = 5000, debug = True)
```

- index.html

```
<!DOCTYPE html>
<html>
<head>
```

```
    <title>Sensor Data Visualization</title>
    <script src="https://code.highcharts.com/highcharts.js">
</head>
<body>
    <div id="container" style="min-width: 310px; height: 400

    <script>
        var chart;

        function fetchDataAndPlot() {
            fetch('/api/data') // Corrected path to fetch da
                .then(response => response.json())
                .then(data => {
                    var humidityData = [];
                    var temperatureData = [];
                    data.reverse(); // Reverse data to displa
                    data.forEach(entry => {
                        humidityData.push(entry.humidity); /
                        temperatureData.push(entry.temperatu
                    });

                    if (!chart) {
                        chart = Highcharts.chart('container',
                            chart: {
                                type: 'line'
                            },
                            title: {
                                text: 'Humidity and Temperatu
                            },
                            xAxis: {
                                categories: Array.from({ leng
                            },
                            yAxis: {
                                title: {
                                    text: 'Value'
                                }
                            },
                            series: [{
                                name: 'Humidity',
                                data: humidityData,
                                color: '#33CEFF' // Change c
                            }, {
                                name: 'Temperature',
                                data: temperatureData,
                                color: 'yellow' // Change co
                            }]
                        });
                    } else {
                        chart.series[0].setData(humidityData
                        chart.series[1].setData(temperatureD
                        chart.xAxis[0].setCategories(Array.f
                    }
                });
        }

        setInterval(fetchDataAndPlot, 2000); // Fetch data e
        fetchDataAndPlot(); // Fetch data initially
    </script>
</body>
</html>
```

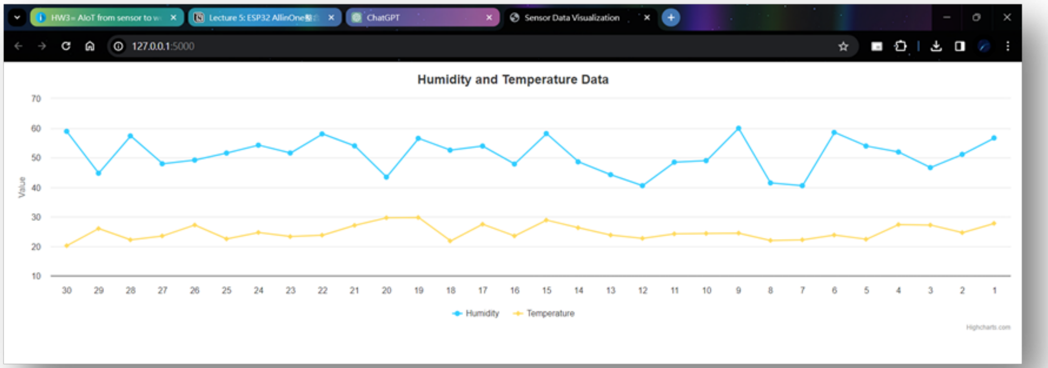
---

3. demo result

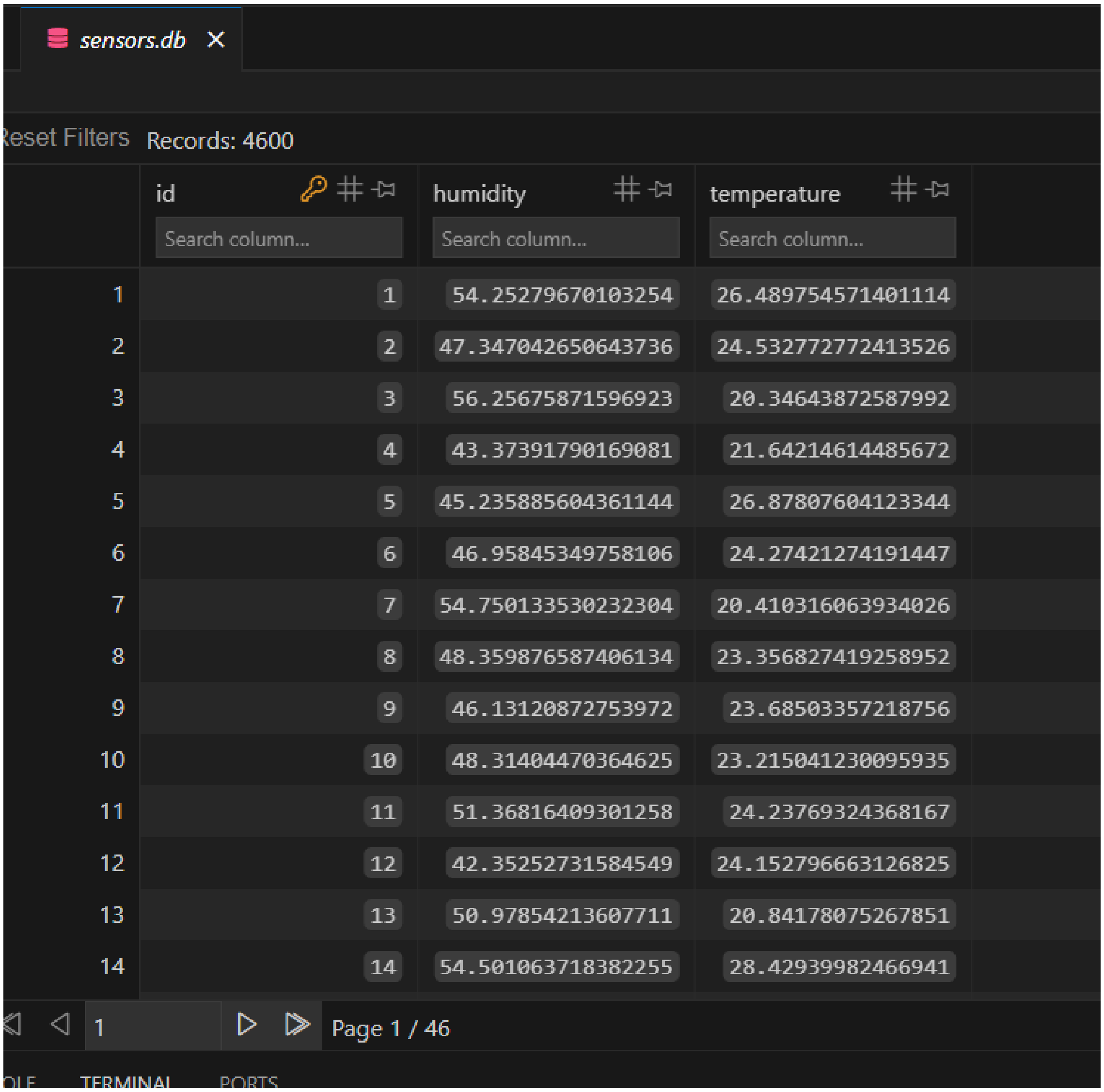
> 兩種

▼ Random產生數值版本

- Chart



- Sqlite viewer

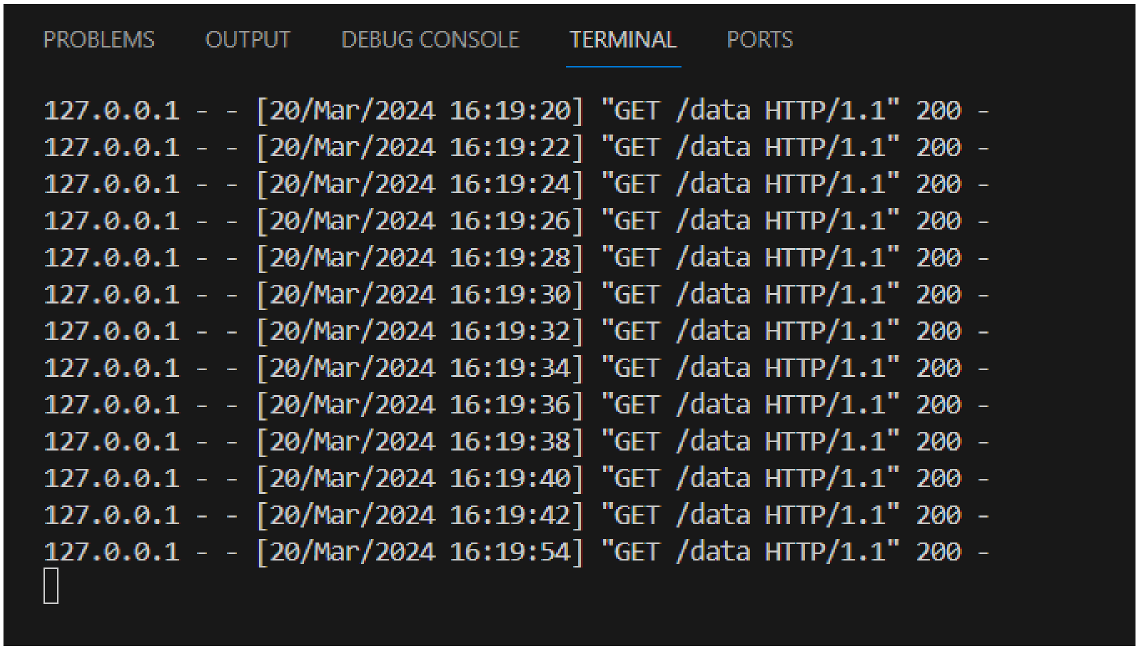


Records: 4600

	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

1 Page 1 / 46

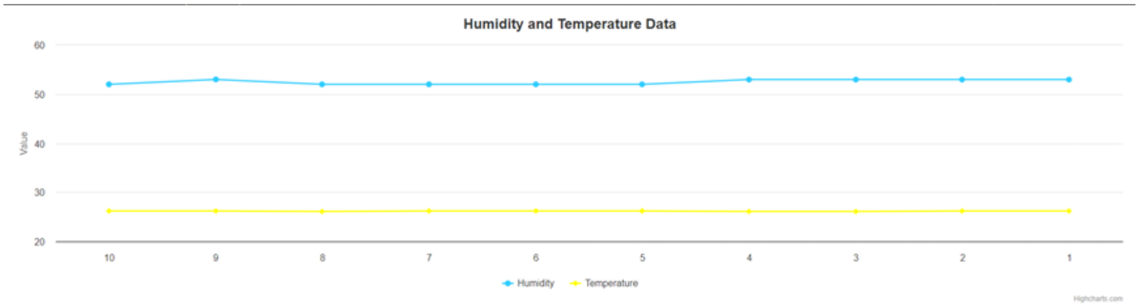
- terminal



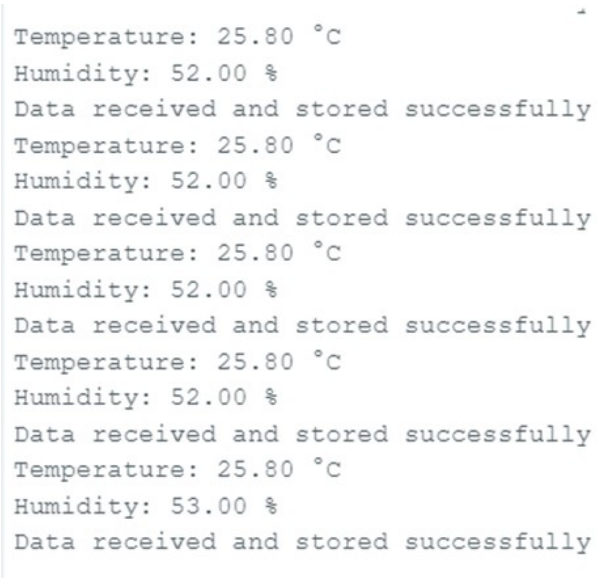
```
127.0.0.1 - - [20/Mar/2024 16:19:20] "GET /data HTTP/1.1" 200 -
127.0.0.1 - - [20/Mar/2024 16:19:22] "GET /data HTTP/1.1" 200 -
127.0.0.1 - - [20/Mar/2024 16:19:24] "GET /data HTTP/1.1" 200 -
127.0.0.1 - - [20/Mar/2024 16:19:26] "GET /data HTTP/1.1" 200 -
127.0.0.1 - - [20/Mar/2024 16:19:28] "GET /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 -
```

▼ 實際數值版本

- Chart

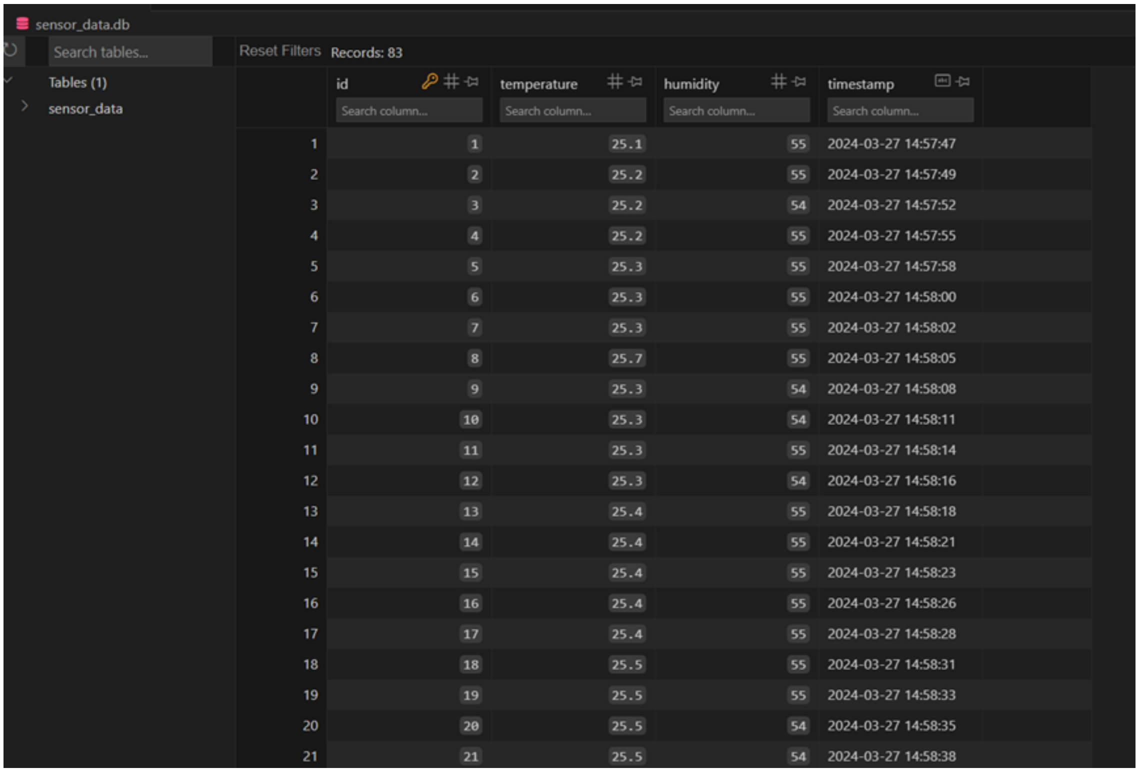


- 數值



```
Temperature: 25.80 °C
Humidity: 52.00 %
Data received and stored successfully
Temperature: 25.80 °C
Humidity: 52.00 %
Data received and stored successfully
Temperature: 25.80 °C
Humidity: 52.00 %
Data received and stored successfully
Temperature: 25.80 °C
Humidity: 52.00 %
Data received and stored successfully
Temperature: 25.80 °C
Humidity: 53.00 %
Data received and stored successfully
```

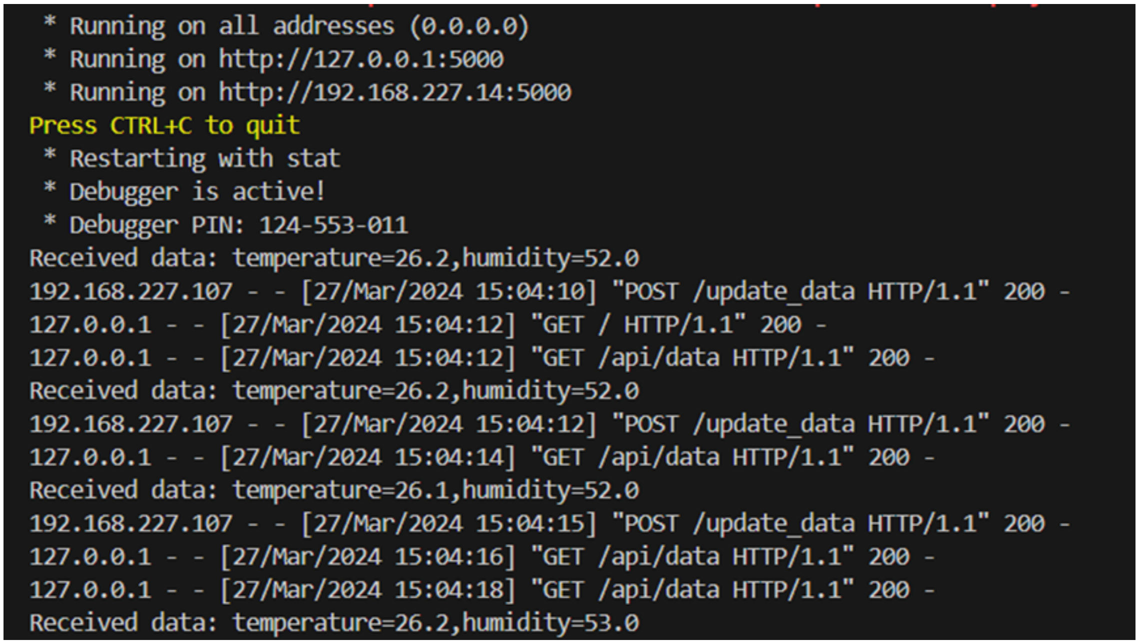
- Sqlite viewer



Records: 83

	id	temperature	humidity	timestamp
1	1	25.1	55	2024-03-27 14:57:47
2	2	25.2	55	2024-03-27 14:57:49
3	3	25.2	54	2024-03-27 14:57:52
4	4	25.2	55	2024-03-27 14:57:55
5	5	25.3	55	2024-03-27 14:57:58
6	6	25.3	55	2024-03-27 14:58:00
7	7	25.3	55	2024-03-27 14:58:02
8	8	25.7	55	2024-03-27 14:58:05
9	9	25.3	54	2024-03-27 14:58:08
10	10	25.3	54	2024-03-27 14:58:11
11	11	25.3	55	2024-03-27 14:58:14
12	12	25.3	54	2024-03-27 14:58:16
13	13	25.4	55	2024-03-27 14:58:18
14	14	25.4	55	2024-03-27 14:58:21
15	15	25.4	55	2024-03-27 14:58:23
16	16	25.4	55	2024-03-27 14:58:26
17	17	25.4	55	2024-03-27 14:58:28
18	18	25.5	55	2024-03-27 14:58:31
19	19	25.5	55	2024-03-27 14:58:33
20	20	25.5	54	2024-03-27 14:58:35
21	21	25.5	54	2024-03-27 14:58:38

- terminal



```
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.227.14:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 124-553-011
Received data: temperature=26.2,humidity=52.0
192.168.227.107 - - [27/Mar/2024 15:04:10] "POST /update_data HTTP/1.1" 200 -
127.0.0.1 - - [27/Mar/2024 15:04:12] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Mar/2024 15:04:12] "GET /api/data HTTP/1.1" 200 -
Received data: temperature=26.2,humidity=52.0
192.168.227.107 - - [27/Mar/2024 15:04:12] "POST /update_data HTTP/1.1" 200 -
127.0.0.1 - - [27/Mar/2024 15:04:14] "GET /api/data HTTP/1.1" 200 -
Received data: temperature=26.1,humidity=52.0
192.168.227.107 - - [27/Mar/2024 15:04:15] "POST /update_data HTTP/1.1" 200 -
127.0.0.1 - - [27/Mar/2024 15:04:16] "GET /api/data HTTP/1.1" 200 -
127.0.0.1 - - [27/Mar/2024 15:04:18] "GET /api/data HTTP/1.1" 200 -
Received data: temperature=26.2,humidity=53.0
```