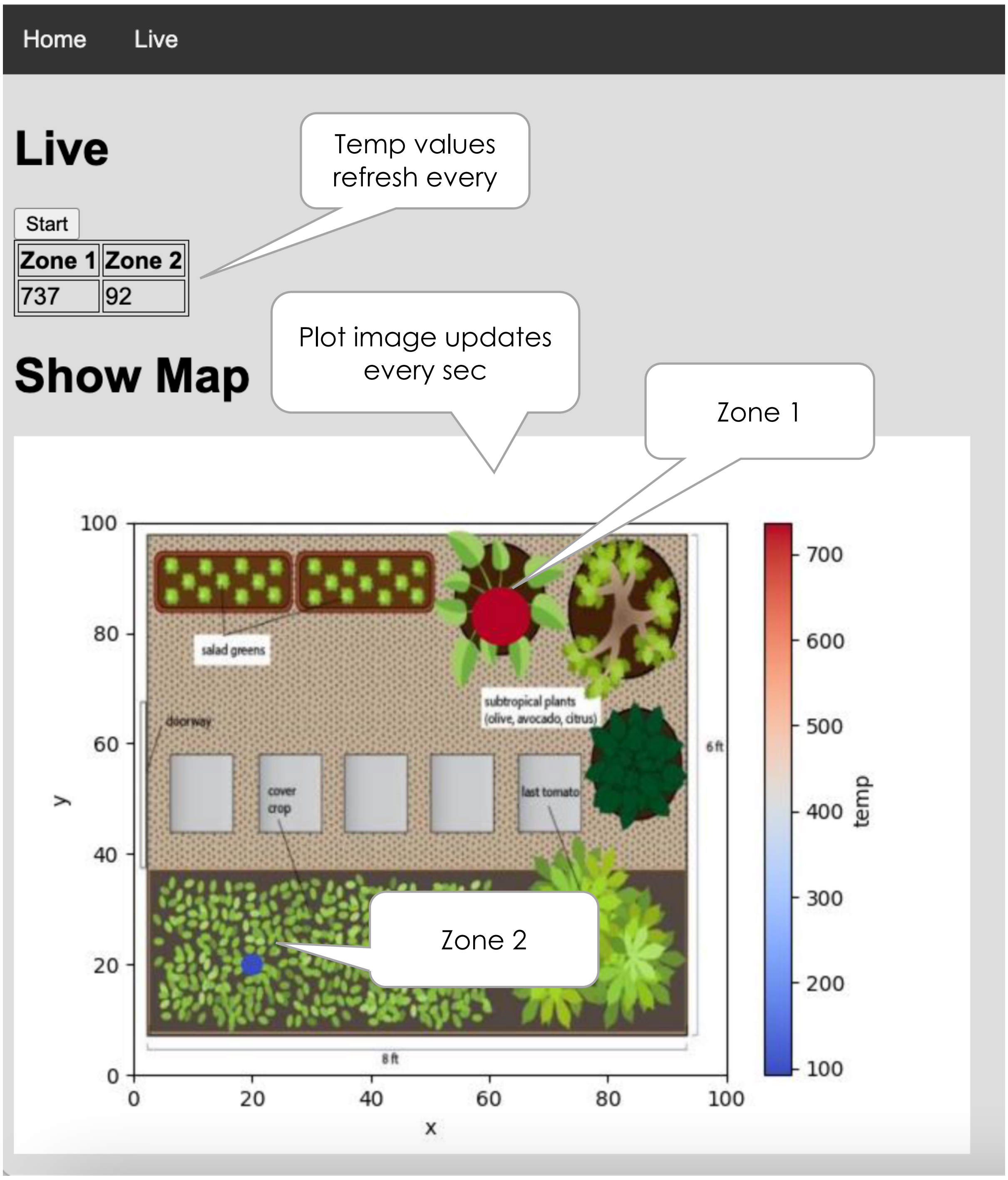
Greenhouse simulation – Temperature Heatmap

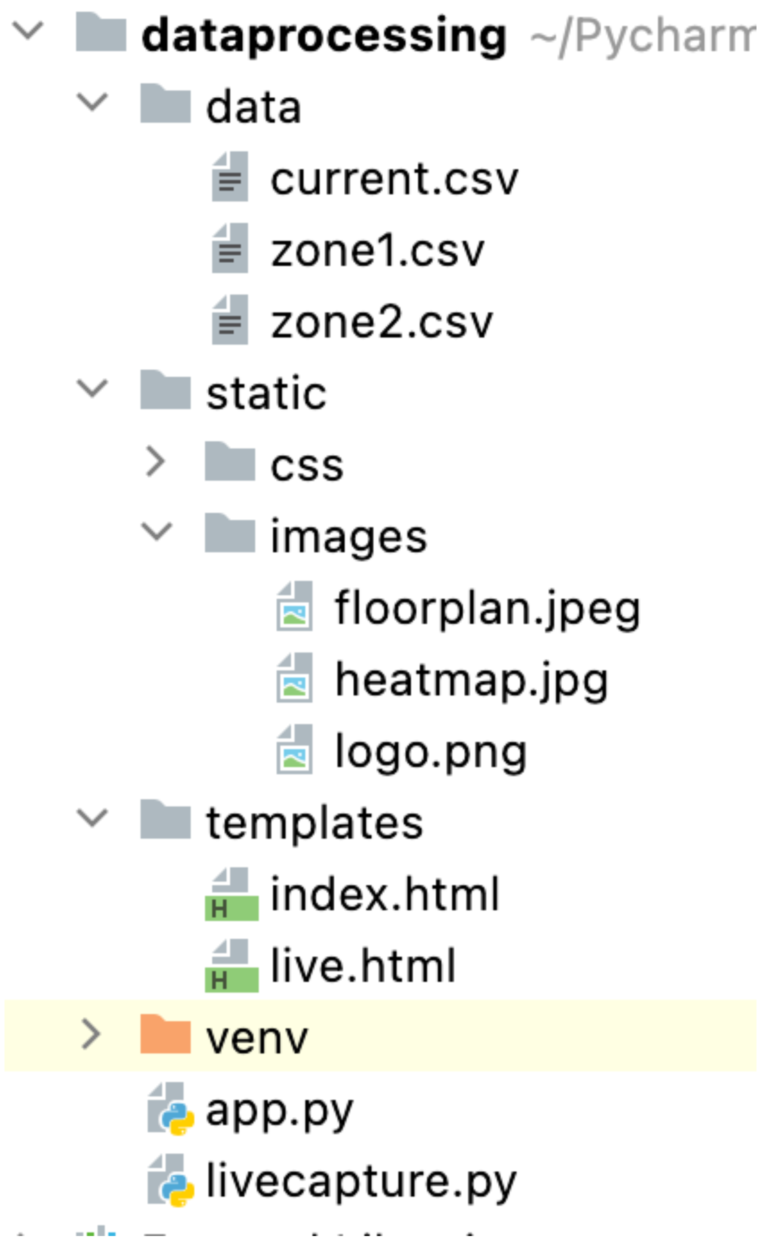

Home
Live
Start
Temp values refresh every
Plot image updates every sec
Zone 1
Zone 2
salad greens
doorway
subtropical plants (olive, avocado, citrus)
cover crop
last tomato
6 ft
8 ft
x
y
temp

Live

Zone 1	Zone 2
737	92

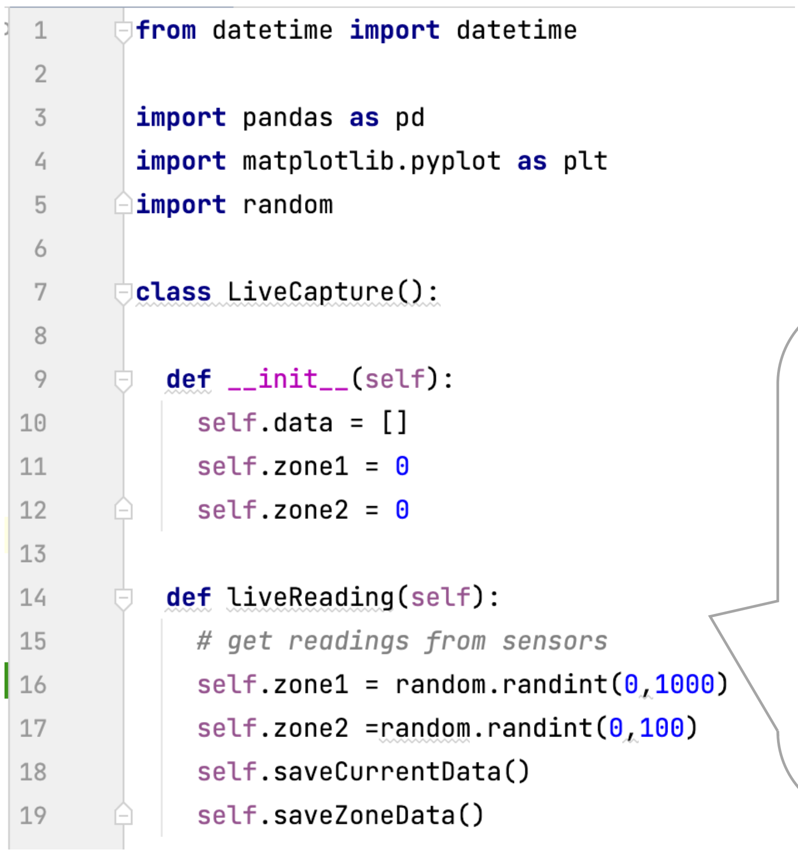
Show Map

Flask web application



```
dataprocessing ~/Pycharm
  data
    current.csv
    zone1.csv
    zone2.csv
  static
    css
    images
      floorplan.jpeg
      heatmap.jpg
      logo.png
  templates
    index.html
    live.html
  venv
  app.py
  livecapture.py
```

livecapture.py Class



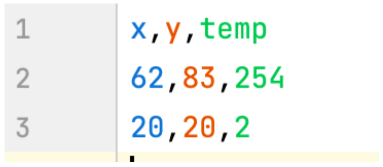
```python
from datetime import datetime

import pandas as pd
import matplotlib.pyplot as plt
import random

class LiveCapture():

    def __init__(self):
        self.data = []
        self.zone1 = 0
        self.zone2 = 0

    def liveReading(self):
        # get readings from sensors
        self.zone1 = random.randint(0,1000)
        self.zone2 =random.randint(0,100)
        self.saveCurrentData()
        self.saveZoneData()
```

This is simulating temperatures for zone 1 and zone 2 in the greenhouse.

It saves the current temperatures to the current data.csv file

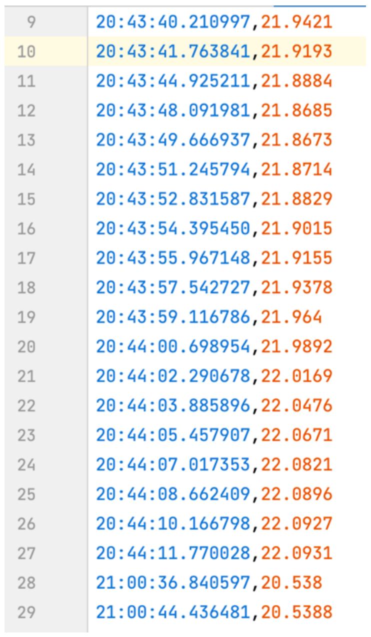
Appends the temperatures to zone1 and zone2 csv files

current.csv



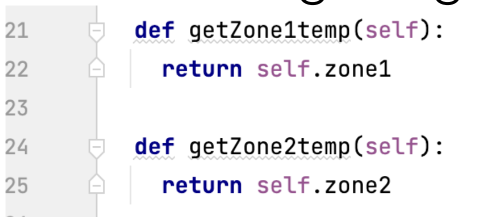
```
x,y,temp
62,83,254
20,20,2
```

zone1.csv



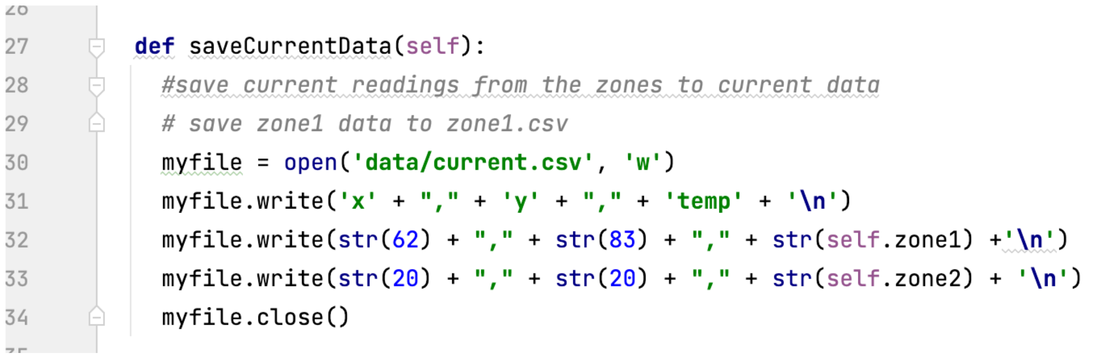
```
20:43:40.210997,21.9421
20:43:41.763841,21.9193
20:43:44.925211,21.8884
20:43:48.091981,21.8685
20:43:49.666937,21.8673
20:43:51.245794,21.8714
20:43:52.831587,21.8829
20:43:54.395450,21.9015
20:43:55.967148,21.9155
20:43:57.542727,21.9378
20:43:59.116786,21.964
20:44:00.698954,21.9892
20:44:02.290678,22.0169
20:44:03.885896,22.0476
20:44:05.457907,22.0671
20:44:07.017353,22.0821
20:44:08.662409,22.0896
20:44:10.166798,22.0927
20:44:11.770028,22.0931
21:00:36.840597,20.538
21:00:44.436481,20.5388
```

Methods for getting the current temps



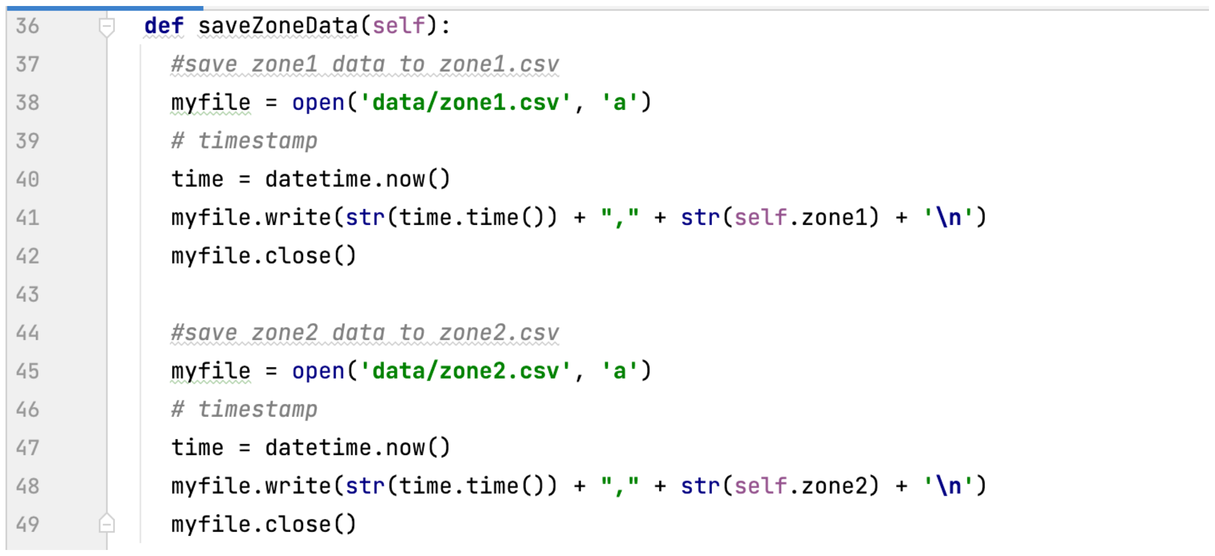
```python
def getZone1temp(self):
    return self.zone1

def getZone2temp(self):
    return self.zone2
```

Method for saving current temp data to the current.csv file



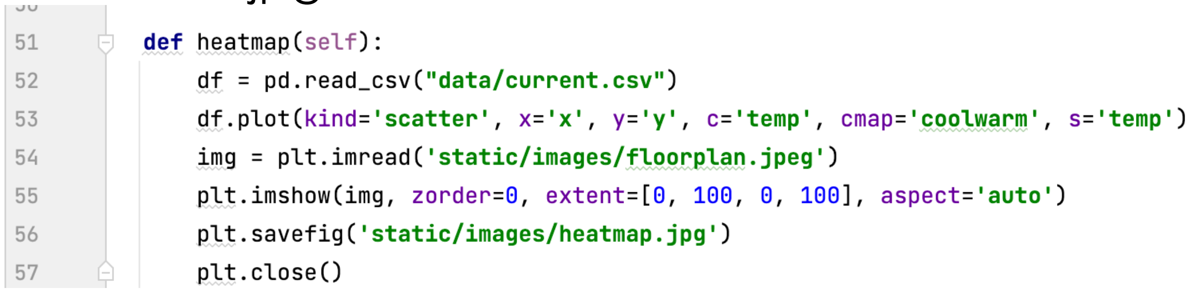
```python
def saveCurrentData(self):
    #save current readings from the zones to current data
    # save zone1 data to zone1.csv
    myfile = open('data/current.csv', 'w')
    myfile.write('x' + "," + 'y' + "," + 'temp' + '\n')
    myfile.write(str(62) + "," + str(83) + "," + str(self.zone1) +'\n')
    myfile.write(str(20) + "," + str(20) + "," + str(self.zone2) + '\n')
    myfile.close()
```

Method for saving the current temps to the zone1 and zone2 csv files.



```python
def saveZoneData(self):
    #save zone1 data to zone1.csv
    myfile = open('data/zone1.csv', 'a')
    # timestamp
    time = datetime.now()
    myfile.write(str(time.time()) + "," + str(self.zone1) + '\n')
    myfile.close()

    #save zone2 data to zone2.csv
    myfile = open('data/zone2.csv', 'a')
    # timestamp
    time = datetime.now()
    myfile.write(str(time.time()) + "," + str(self.zone2) + '\n')
    myfile.close()
```

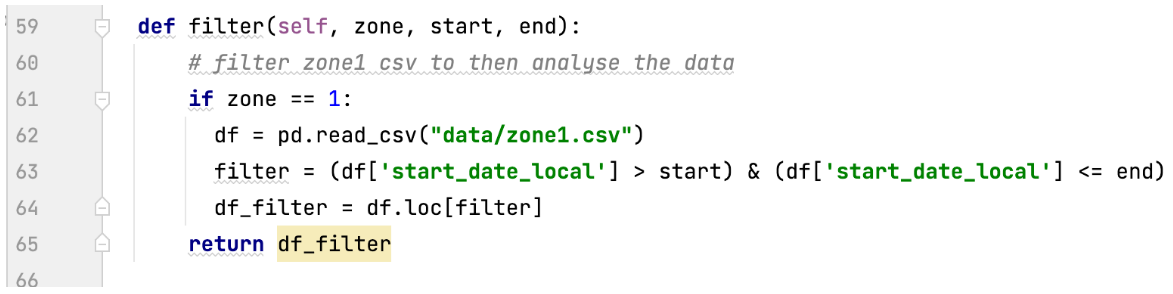
Method for generating a scatterplot and saving the chart to a jpg file



```python
def heatmap(self):
    df = pd.read_csv("data/current.csv")
    df.plot(kind='scatter', x='x', y='y', c='temp', cmap='coolwarm', s='temp')
    img = plt.imread('static/images/floorplan.jpeg')
    plt.imshow(img, zorder=0, extent=[0, 100, 0, 100], aspect='auto')
    plt.savefig('static/images/heatmap.jpg')
    plt.close()
```

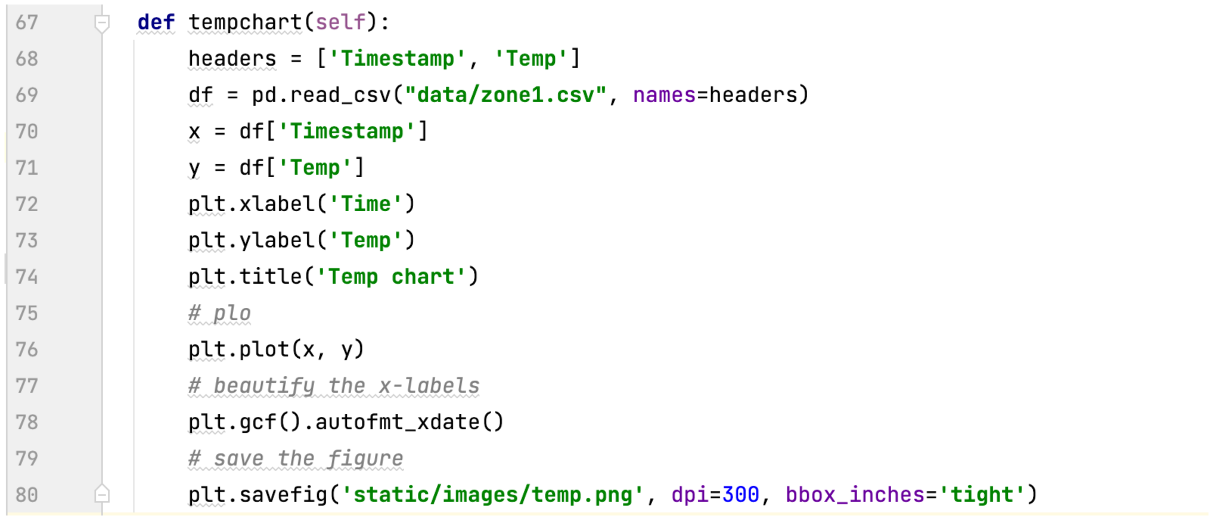
Useful methods:

Filter() will filter the df by start and end data, using data from a specified zone:



```python
def filter(self, zone, start, end):
    # filter zone1 csv to then analyse the data
    if zone == 1:
        df = pd.read_csv("data/zone1.csv")
        filter = (df['start_date_local'] > start) & (df['start_date_local'] <= end)
        df_filter = df.loc[filter]
    return df_filter
```

tempchart() creates a line chart using the data stored in zone1.csv file



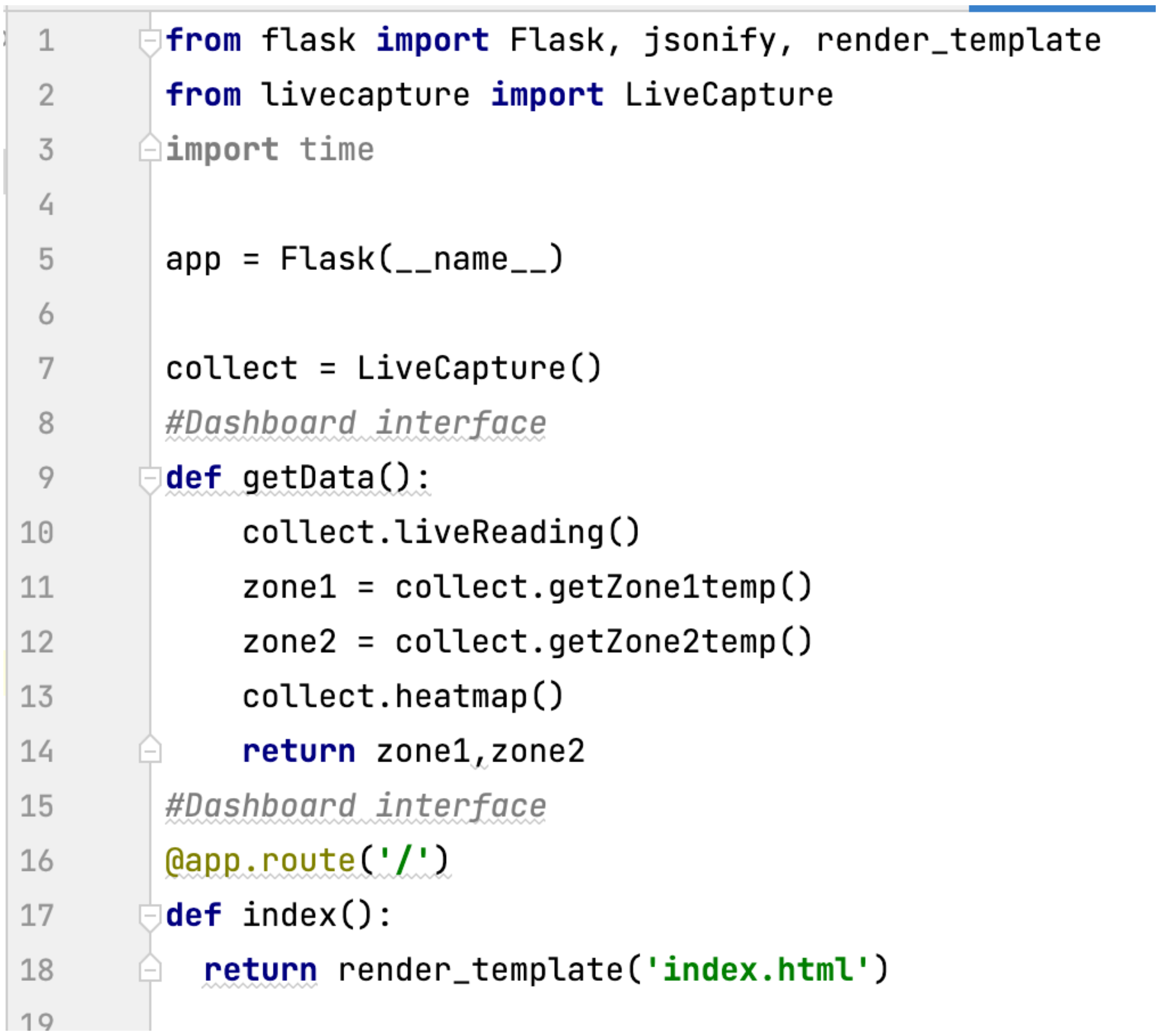
```python
def tempchart(self):
    headers = ['Timestamp', 'Temp']
    df = pd.read_csv("data/zone1.csv", names=headers)
    x = df['Timestamp']
    y = df['Temp']
    plt.xlabel('Time')
    plt.ylabel('Temp')
    plt.title('Temp chart')
    # plo
    plt.plot(x, y)
    # beautify the x-labels
    plt.gcf().autofmt_xdate()
    # save the figure
    plt.savefig('static/images/temp.png', dpi=300, bbox_inches='tight')
```

Live.html

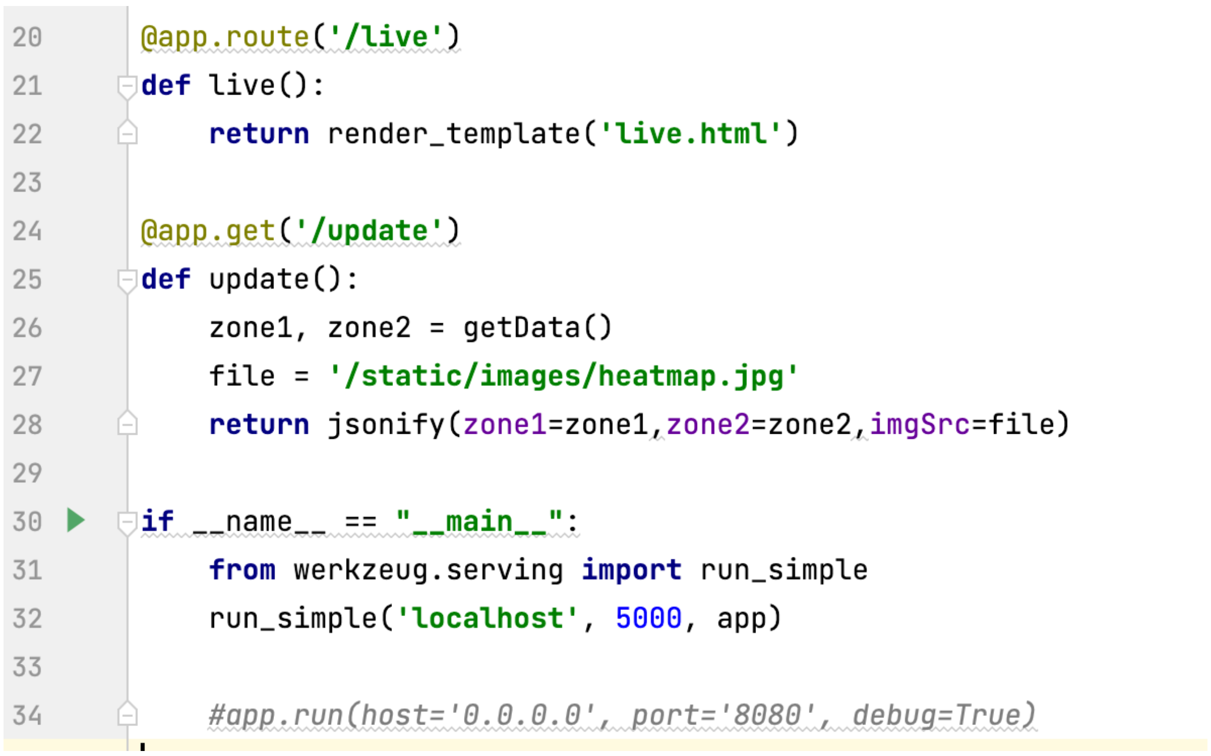
```
<!doctype html>
    <head>
    <!-- Custom styles for this template -->
        <link href="{{ url_for('static', filename='css/style.css') }}" rel="stylesheet">
        <title>Live</title>
    </head>
<body>
    <img src="{{ url_for('static', filename='images/logo.png') }}"/>

    <div class="topnav">
        <a class="nav-link active" href="{{ url_for('index')}}">Home</a>
        <a class="nav-link active" href="{{ url_for('live')}}">Live</a>
    </div>

    <div class="content">
        <h1> Live </h1>

        <button onclick="start();">Start</button>

        <table>
        <tr>
        <th> Zone 1</th>
        <th> Zone 2 </th>

        </tr>
        <tr>
        <td id="zone1" ></td>
        <td id="zone2"></td>
        </tr>
        </table>

        <h1> Show Heatmap </h1>
        <img id = 'image' src=""/>

  <script>
     function update(){
            //Fetch is a command to make a HTTP request
            fetch("/update").then(function(response){
                return response.json();

            }).then (function(data){
                    console.log(data);

                    document.getElementById("zone1").innerText = data['zone1'];
                    document.getElementById("zone2").innerText = data['zone2'];
                    document.getElementById("image").src = data['imgSrc']+'?t='+ new Date().getTime();
                }).catch(function(err){
                    console.log(err.message);
                });
        }
    function start(){
      setInterval(update, 1000);
    }

    </script>
    </div>

    <div class="footer">
    </div>
</body>
</html>
```

JavaScript is used to fetch data from the server every sec and update the zone1, zone2 and image elements.

app.py



```python
from flask import Flask, jsonify, render_template
from livecapture import LiveCapture
import time

app = Flask(__name__)

collect = LiveCapture()
#Dashboard interface
def getData():
    collect.liveReading()
    zone1 = collect.getZone1temp()
    zone2 = collect.getZone2temp()
    collect.heatmap()
    return zone1,zone2
#Dashboard interface
@app.route('/')
def index():
  return render_template('index.html')

```



```python
@app.route('/live')
def live():
    return render_template('live.html')

@app.get('/update')
def update():
    zone1, zone2 = getData()
    file = '/static/images/heatmap.jpg'
    return jsonify(zone1=zone1,zone2=zone2,imgSrc=file)

if __name__ == "__main__":
    from werkzeug.serving import run_simple
    run_simple('localhost', 5000, app)

    #app.run(host='0.0.0.0', port='8080', debug=True)
```