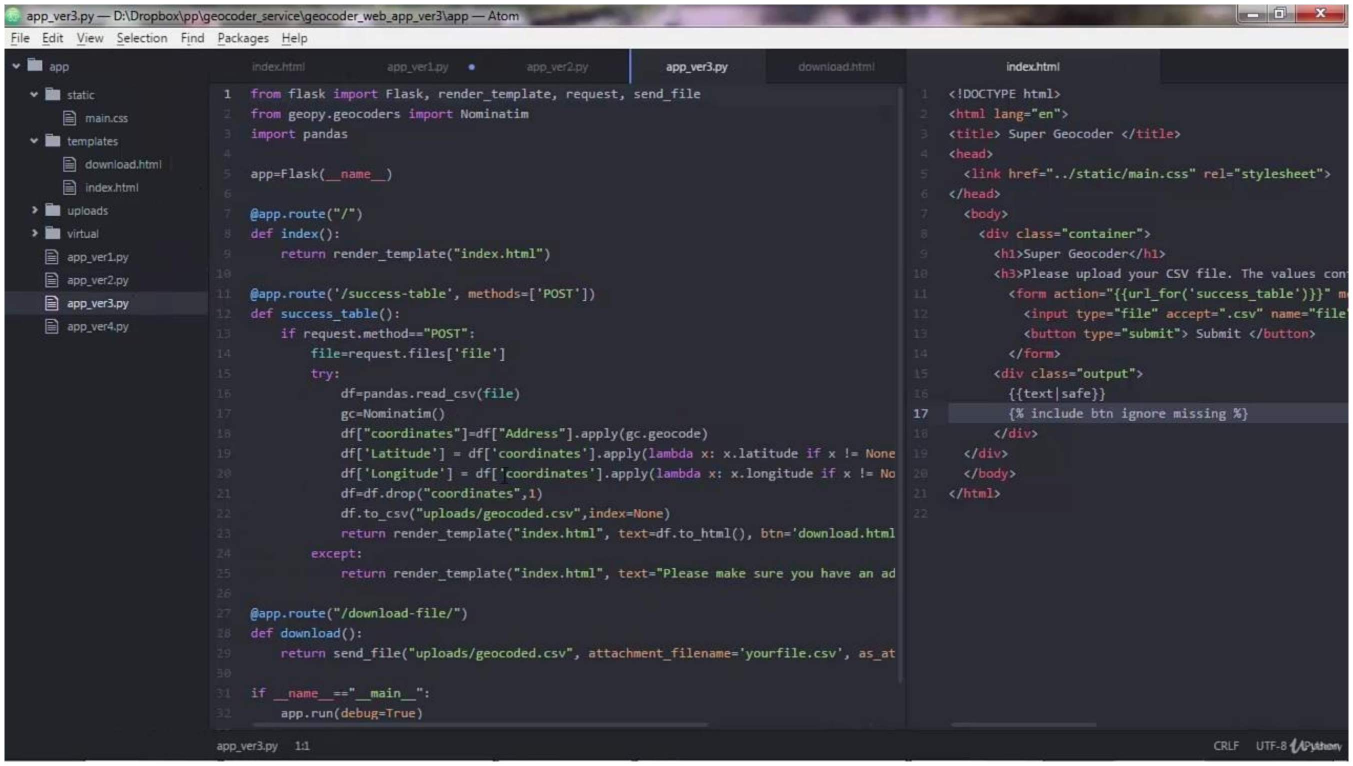
Date:	**06/06/2020**	**Name:**	**Soundarya NA**
Course:	**Udemy**	**USN:**	**4AL16EC077**
Topic:	**Python**	**Semester & Section:**	**8th B**

AFTERNOON SESSION DETAILS

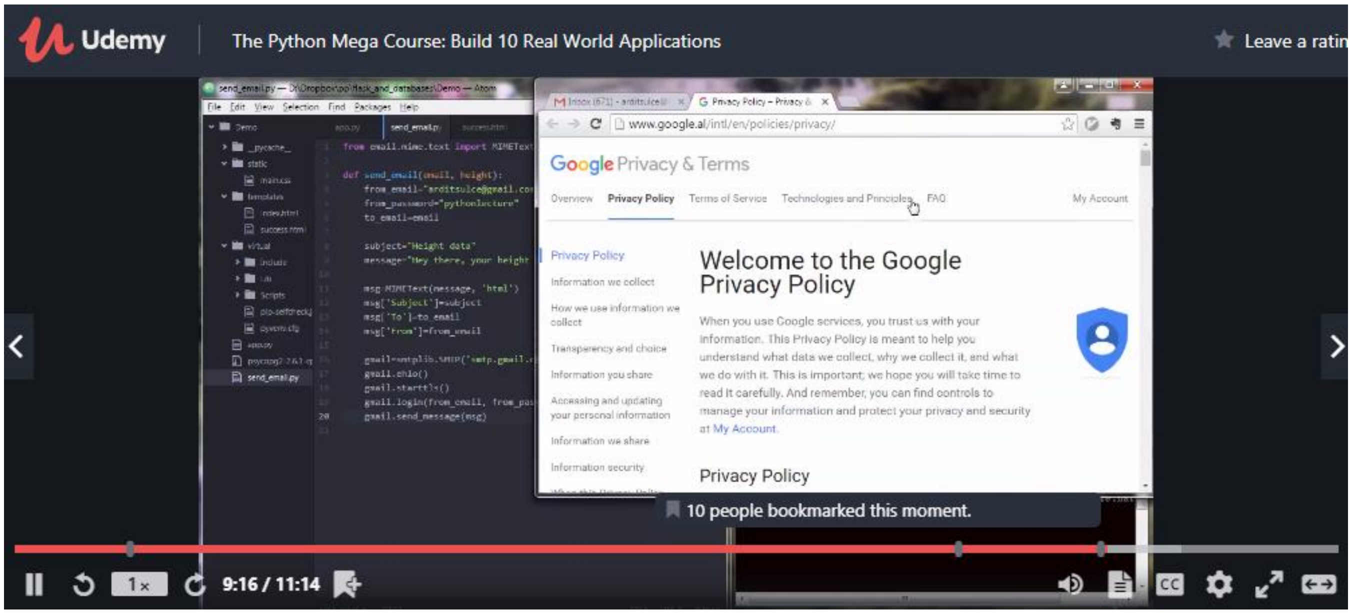


app_ver3.py

```python
from flask import Flask, render_template, request, send_file
from geopy.geocoders import Nominatim
import pandas

app=Flask(__name__)

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

@app.route('/success-table', methods=['POST'])
def success_table():
    if request.method=="POST":
        file=request.files['file']
        try:
            df=pandas.read_csv(file)
            gc=Nominatim()
            df["coordinates"]=df["Address"].apply(gc.geocode)
            df['Latitude'] = df['coordinates'].apply(lambda x: x.latitude if x != None
            df['Longitude'] = df['coordinates'].apply(lambda x: x.longitude if x != No
            df=df.drop("coordinates",1)
            df.to_csv("uploads/geocoded.csv",index=None)
            return render_template("index.html", text=df.to_html(), btn='download.html
        except:
            return render_template("index.html", text="Please make sure you have an ad

@app.route("/download-file/")
def download():
    return send_file("uploads/geocoded.csv", attachment_filename='yourfile.csv', as_at

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

index.html

```html
<!DOCTYPE html>
<html lang="en">
<title> Super Geocoder </title>
<head>
  <link href="../static/main.css" rel="stylesheet">
</head>
  <body>
    <div class="container">
      <h1>Super Geocoder</h1>
      <h3>Please upload your CSV file. The values con
        <form action="{{url_for('success_table')}}" m
          <input type="file" accept=".csv" name="file
          <button type="submit"> Submit </button>
        </form>
      <div class="output">
        {{text|safe}}
        {% include btn ignore missing %}
      </div>
    </div>
  </body>
</html>
```



Udemy | The Python Mega Course: Build 10 Real World Applications

send_email.py

```python
from email.mime.text import MIMEText

def send_email(email, height):
    from_email="arditsulce@gmail.co
    from_password="pythonlecture"
    to_email=email

    subject="Height data"
    message="Hey there, your height

    msg=MIMEText(message, 'html')
    msg['Subject']=subject
    msg['To']=to_email
    msg['From']=from_email

    gmail=smtplib.SMTP('smtp.gmail.
    gmail.ehlo()
    gmail.starttls()
    gmail.login(from_email, from_pa
    gmail.send_message(msg)
```

Google Privacy & Terms

Welcome to the Google Privacy Policy

When you use Google services, you trust us with your information. This Privacy Policy is meant to help you understand what data we collect, why we collect it, and what we do with it. This is important; we hope you will take time to read it carefully. And remember, you can find controls to manage your information and protect your privacy and security at My Account.

Privacy Policy

10 people bookmarked this moment.

9:16 / 11:14

Report:

The GeocoderRequest object literal contains the following fields:

```
{
 address: string,
 location: LatLng,
 placeId: string,
 bounds: LatLngBounds,
 componentRestrictions: GeocoderComponentRestrictions,
 region: string
}
```

Required parameters: You must supply one, and only one, of the following fields:

• address — The address which you want to geocode.

or

location — The LatLng (or LatLngLiteral) for which you wish to obtain the closest, human-readable address. The geocoder performs a reverse geocode. See Reverse Geocoding for more information.

or

placeId — The place ID of the place for which you wish to obtain the closest, human-readable address. See more about retrieving an address for a place ID.

Optional parameters:

• bounds — The LatLngBounds within which to bias geocode results more prominently. The bounds parameter will only influence, not fully restrict, results from the geocoder. See more information about viewport biasing below.

• componentRestrictions — Used to restrict results to a specific area. See more information about component filtering below.

• region — The region code, specified as a IANA language region subtag. In most cases, these tags map directly to familiar ccTLD ("top-level domain") two-character values. The region parameter will only influence, not fully restrict, results from the geocoder. See more information about region code biasing below.

The GeocoderResult object represents a single geocoding result. A geocode request may return multiple result objects:

results[]: {

```
 types[]: string,
 formatted_address: string,
 address_components[]: {
   short_name: string,
   long_name: string,
   postcode_localities[]: string,
   types[]: string
 },
 partial_match: boolean,
 place_id: string,
 postcode_localities[]: string,
 geometry: {
   location: LatLng,
   location_type: GeocoderLocationType
   viewport: LatLngBounds,
   bounds: LatLngBounds
 }
}
```

Code:

```
<html>
  <head>
  <meta name="viewport" content="initial-scale=1.0, width=device-width" />
  <script     src="https://js.api.here.com/v3/3.1/mapsjs-core.js"type="text/javascript"     charset="utf-
8"></script>
  <script    src="https://js.api.here.com/v3/3.1/mapsjs-service.js"type="text/javascript"    charset="utf-
8"></script>
  </head>
  <body style='margin: 0'>
  <div style="width: 100vw; height: 100vh" id="mapContainer"></div>
  <script>
```

```
   // Initialize the platform object:
   var platform = new H.service.Platform({
   'apikey': '{{apikey}}'
   });

          const lng = {{longitude}};
          const lat = {{latitude}};

// Obtain the default map types from the platform object
          var defaultLayers = platform.createDefaultLayers();

// Instantiate (and display) a map object:
var map = new H.Map(
   document.getElementById('mapContainer'),
   defaultLayers.vector.normal.map,
   {
     zoom: 10,
     center: { lat: lat, lng: lng }
   });

          const marker = new H.map.Marker({lat: lat, lng: lng});
          map.addObject(marker);
  </script>
  </body>
</html>
```