The Python Micro Book

A cheat sheet to serve as a suplement of "The Python Mega Course"

Please note that this is not a normal book that will train you in Python. The book comes as a supplement of "The Python Mega Course" and it can be used as a Python code cheat sheet for quick look up to Python code snippets and commands.

The book contains the code for building 10 different Python applications which are explained in "The Python Mega Course".

Ardit Sulce

Application #1: Name Generator

```
import random
vowels='aeiouy'
consonants='bcdfghjklmnpqrstvwxz'
letters=vowels+consonants
letter_input_1=input("What letter do you want? Enter 'v' for vowels, 'c'
for consonants, 'l' for any letter: ")
letter_input_2=input("What letter do you want? Enter 'v' for vowels, 'c'
for consonants, 'l' for any letter: ")
letter_input_3=input("What letter do you want? Enter 'v' for vowels, 'c'
for consonants, 'l' for any letter: ")
def plot():
    if letter_input_1 == 'v':
        l1=random.choice(vowels)
    elif letter_input_1 == 'c':
        l1=random.choice(consonants)
    elif letter_input_1 == 'l':
        l1=random.choice(letters)
    else:
        l1=letter_input_1
    if letter_input_2 == 'v':
        l2=random.choice(vowels)
    elif letter_input_2 == 'c':
        l2=random.choice(consonants)
    elif letter_input_2 == 'l':
        l2=random.choice(letters)
    else:
        l2=letter_input_2
    if letter_input_3 == 'v':
        l3=random.choice(vowels)
    elif letter_input_3 == 'c':
        l3=random.choice(consonants)
    elif letter_input_3 == 'l':
        l3=random.choice(letters)
    else:
        l3=letter_input_3
    name = l1+l2+l3
    return name
for i in range(10):
    print(plot())
```

Application #2: Producing Web Maps With Folium

```
import folium
import pandas
df=pandas.read_csv("Volcanoes_USA.txt")
map=folium.Map(location=[df['LAT'].mean(),df['LON'].mean()],zoom_start=6,ti
les='Stamen Terrain')
def color(elev):
    minimum=int(min(df['ELEV']))
    step=int((max(df['ELEV'])-min(df['ELEV']))/3)
    if elev in range(minimum,minimum+step):
        col='green'
    elif elev in range(minimum+step,minimum+step*2):
        col='orange'
    else:
        col='red'
    return col
for lat,lon,name,elev in zip(df['LAT'],df['LON'],df['NAME'],df['ELEV']):

map.simple_marker(location=[lat,lon],popup=name,marker_color=color(elev))
map.create_map(path='test.html')
```

Application #3: Building a Website Blocker

```
import time
from datetime import datetime as dt
hosts_temp=r"D:\Dropbox\pp\block_websites\Demo\hosts"
hosts_path=r"C:\Windows\System32\drivers\etc\hosts"
redirect="127.0.0.1"
website_list=["www.facebook.com","facebook.com","dub119.mail.live.com","www
.dub119.mail.live.com"]
while True:
    if dt(dt.now().year,dt.now().month,dt.now().day,8) < dt.now() <
dt(dt.now().year,dt.now().month,dt.now().day,16):
        print("Working hours...")
        with open(hosts_path,'r+') as file:
            content=file.read()
            for website in website_list:
                if website in content:
                    pass
                else:
                    file.write(redirect+" "+ website+"\n")
    else:
        with open(hosts_path,'r+') as file:
            content=file.readlines()
            file.seek(0)
            for line in content:
                if not any(website in line for website in website_list):
                    file.write(line)
            file.truncate()
        print("Fun hours...")
    time.sleep(5)
```

Application #4: Building a Website with Flask

```
#script1.py
from flask import Flask, render_template
app=Flask(__name__)
@app.route('/')
def home():
    return render_template("home.html")
@app.route('/about/')
def about():
    return render_template("about.html")
if __name__=="__main__":
    app.run(debug=True)
```

#templates/home.html

```
<!DOCTYPE html>
<html>
  <head>
      <title>Flask App</title>
      <link rel="stylesheet"
href="{{url_for('static',filename='css/main.css')}}"
  </head>
  <body>
    <header>
      <div class="container">
        <h1 class="logo">Ardit's web app</h1>
        <strong><nav>
          <ul class="menu">
            <li><a href="{{ url_for('home') }}">Home</a></li>
            <li><a href="{{ url_for('about') }}">About</a></li>
          </ul>
        </nav></strong>
      </div>
    </header>
    <div class="container">
        {%block content%}
        {%endblock%}
    </div>
  </body>
</html>
```

#templates/home.html

```
{%extends "layout.html"%}
{%block content%}
<div class="home">
    <h1>My homepage</h1>
    <p>This is a test website</p>
</div>
{%endblock%}
```

#templates/about.html

```
{%extends "layout.html"%}
{%block content%}
<div class="about">
    <h1>My about page</h1>
    <p>This is a test website again</p>
</div>
{%endblock%}
```

#/static/main.css

```
body {
  margin: 0;
  padding: 0;
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  color: #444;
}
header {
  background-color: #DFB887;
  height: 35px;
  width: 100%;
  opacity: .9;
  margin-bottom: 10px;
}
header h1.logo {
  margin: 0;
  font-size: 1.7em;
  color: #fff;
  text-transform: uppercase;
  float: left;
}
header h1.logo:hover {
  color: #fff;
  text-decoration: none;
}
```

```
.container {
  width: 1200px;
  margin: 0 auto;
}
div.home {
  padding: 10px 0 30px 0;
  background-color: #E6E6FA;
  -webkit-border-radius: 6px;
     -moz-border-radius: 6px;
          border-radius: 6px;
}
div.about {
  padding: 10px 0 30px 0;
  background-color: #E6E6FA;
  -webkit-border-radius: 6px;
     -moz-border-radius: 6px;
          border-radius: 6px;
}
h2 {
  font-size: 3em;
  margin-top: 40px;
  text-align: center;
  letter-spacing: -2px;
}
h3 {
  font-size: 1.7em;
  font-weight: 100;
  margin-top: 30px;
  text-align: center;
  letter-spacing: -1px;
  color: #999;
}
.menu {
  float: right;
  margin-top: 8px;
}
.menu li {
  display: inline;
}
.menu li + li {
  margin-left: 35px;
}
.menu li a {
  color: #444;
  text-decoration: none;
}
```

Application #5: Creating a Book Store Application

#frontend.py

```
from tkinter import *
import backend
def get_selected_row(event):
    global selected_tuple
    index=list1.curselection()[0]
    selected_tuple=list1.get(index)
    e1.delete(0,END)
    e1.insert(END,selected_tuple[1])
    e2.delete(0,END)
    e2.insert(END,selected_tuple[2])
    e3.delete(0,END)
    e3.insert(END,selected_tuple[3])
    e4.delete(0,END)
    e4.insert(END,selected_tuple[4])
def view_command():
    list1.delete(0,END)
    for row in backend.view():
        list1.insert(END,row)
def search_command():
    list1.delete(0,END)
    for row in
backend.search(title_text.get(),author_text.get(),year_text.get(),isbn_text
.get()):
        list1.insert(END,row)
def add_command():

backend.insert(title_text.get(),author_text.get(),year_text.get(),isbn_text
.get())
    list1.delete(0,END)

list1.insert(END,(title_text.get(),author_text.get(),year_text.get(),isbn_t
ext.get()))
def delete_command():
    backend.delete(selected_tuple[0])
def update_command():

backend.update(selected_tuple[0],title_text.get(),author_text.get(),year_te
xt.get(),isbn_text.get())
window=Tk()
window.wm_title("BookStore")
l1=Label(window,text="Title")
l1.grid(row=0,column=0)
l2=Label(window,text="Author")
l2.grid(row=0,column=2)
l3=Label(window,text="Year")
l3.grid(row=1,column=0)
l4=Label(window,text="ISBN")
l4.grid(row=1,column=2)
title_text=StringVar()
e1=Entry(window,textvariable=title_text)
e1.grid(row=0,column=1)
author_text=StringVar()
e2=Entry(window,textvariable=author_text)
e2.grid(row=0,column=3)
year_text=StringVar()
e3=Entry(window,textvariable=year_text)
```

```
e3.grid(row=1,column=1)
isbn_text=StringVar()
e4=Entry(window,textvariable=isbn_text)
e4.grid(row=1,column=3)
list1=Listbox(window, height=6,width=35)
list1.grid(row=2,column=0,rowspan=6,columnspan=2)
sb1=Scrollbar(window)
sb1.grid(row=2,column=2,rowspan=6)
list1.configure(yscrollcommand=sb1.set)
sb1.configure(command=list1.yview)
list1.bind('<<ListboxSelect>>',get_selected_row)
b1=Button(window,text="View all", width=12,command=view_command)
b1.grid(row=2,column=3)
b2=Button(window,text="Search entry", width=12,command=search_command)
b2.grid(row=3,column=3)
b3=Button(window,text="Add entry", width=12,command=add_command)
b3.grid(row=4,column=3)
b4=Button(window,text="Update selected", width=12,command=update_command)
b4.grid(row=5,column=3)
b5=Button(window,text="Delete selected", width=12,command=delete_command)
b5.grid(row=6,column=3)
b6=Button(window,text="Close", width=12,command=window.destroy)
b6.grid(row=7,column=3)
window.mainloop()
```

#backend.py

```
import sqlite3
def connect():
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS book (id INTEGER PRIMARY KEY,
title text, author text, year integer, isbn integer)")
    conn.commit()
    conn.close()
def insert(title,author,year,isbn):
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("INSERT INTO book VALUES
(NULL,?,?,?,?)",(title,author,year,isbn))
    conn.commit()
    conn.close()
def view():
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("SELECT * FROM book")
    rows=cur.fetchall()
    conn.close()
    return rows
def search(title="",author="",year="",isbn=""):
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("SELECT * FROM book WHERE title=? OR author=? OR year=? OR
isbn=?", (title,author,year,isbn))
    rows=cur.fetchall()
    conn.close()
    return rows
def delete(id):
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("DELETE FROM book WHERE id=?",(id,))
    conn.commit()
    conn.close()
def update(id,title,author,year,isbn):
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("UPDATE book SET title=?, author=?, year=?, isbn=? WHERE
id=?",(title,author,year,isbn,id))
    conn.commit()
    conn.close()
connect()
```

Application #6: Building a Motion Detector

#plotting.py

```
from motion_detector import df
from bokeh.plotting import figure, show, output_file
from bokeh.models import HoverTool, ColumnDataSource
df["Start_string"]=df["Start"].dt.strftime("%Y-%m-%d %H:%M:%S")
df["End_string"]=df["End"].dt.strftime("%Y-%m-%d %H:%M:%S")
cds=ColumnDataSource(df)
p=figure(x_axis_type='datetime',height=100, width=500,
responsive=True,title="Motion Graph")
p.yaxis.minor_tick_line_color=None
p.ygrid[0].ticker.desired_num_ticks=1
hover=HoverTool(tooltips=[("Start","@Start_string"),("End","@End_string")])
p.add_tools(hover)
q=p.quad(left="Start",right="End",bottom=0,top=1,color="green",source=cds)
output_file("Graph1.html")
show(p)
```

#motion_detector.py

```
import cv2, time, pandas
from datetime import datetime
first_frame=None
status_list=[None,None]
times=[]
df=pandas.DataFrame(columns=["Start","End"])
video=cv2.VideoCapture(0)
while True:
    check, frame = video.read()
    status=0
    gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    gray=cv2.GaussianBlur(gray,(21,21),0)
    if first_frame is None:
        first_frame=gray
        continue
    delta_frame=cv2.absdiff(first_frame,gray)
    thresh_frame=cv2.threshold(delta_frame, 30, 255, cv2.THRESH_BINARY)[1]
    thresh_frame=cv2.dilate(thresh_frame, None, iterations=2)
    (_,cnts,_)=cv2.findContours(thresh_frame.copy(),cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    for contour in cnts:
        if cv2.contourArea(contour) < 10000:
            continue
        status=1
        (x, y, w, h)=cv2.boundingRect(contour)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0,255,0), 3)
    status_list.append(status)
    status_list=status_list[-2:]
    if status_list[-1]==1 and status_list[-2]==0:
        times.append(datetime.now())
    if status_list[-1]==0 and status_list[-2]==1:
        times.append(datetime.now())
    cv2.imshow("Gray Frame",gray)
    cv2.imshow("Delta Frame",delta_frame)
    cv2.imshow("Threshold Frame",thresh_frame)
    cv2.imshow("Color Frame",frame)
    key=cv2.waitKey(1)
    if key==ord('q'):
        if status==1:
            times.append(datetime.now())
        break
print(status_list)
print(times)
for i in range(0,len(times),2):
    df=df.append({"Start":times[i],"End":times[i+1]},ignore_index=True)
df.to_csv("Times.csv")
video.release()
cv2.destroyAllWindows
```

Application #7: Building a a Property Listing Scrapper

```
import requests
from bs4 import BeautifulSoup
r=requests.get("http://www.century21.com/real-estate/rock-springs-
wy/LCWYROCKSPRINGS/")
c=r.content
soup=BeautifulSoup(c,"html.parser")
all=soup.find_all("div",{"class":"propertyRow"})
all[0].find("h4",{"class":"propPrice"}).text.replace("\n","").replace("
","")
page_nr=soup.find_all("a",{"class":"Page"})[-1].text
l=[]
for page in range(0,int(page_nr)*10,10):

print("http://www.century21.com/search.c21?lid=CWYROCKSPRINGS&t=0&s="+str(p
age)+"&subView=searchView.Paginate")

r=requests.get("http://www.century21.com/search.c21?lid=CWYROCKSPRINGS&t=0&
s="+str(page)+"&subView=searchView.Paginate")
    c=r.json()["list"]
    soup=BeautifulSoup(c,"html.parser")
    all=soup.find_all("div",{"class":"propertyRow"})
    for item in all:
        d={}

d["Address"]=item.find_all("span",{"class","propAddressCollapse"})[0].text
        try:

d["Locality"]=item.find_all("span",{"class","propAddressCollapse"})[1].text
        except:
            d["Locality"]=None

d["Price"]=item.find("h4",{"class","propPrice"}).text.replace("\n","").repl
ace(" ","")
        try:
            d["Beds"]=item.find("span",{"class","infoBed"}).find("b").text
        except:
            d["Beds"]=None

        try:
            d["Area"]=item.find("span",{"class","infoSqFt"}).find("b").text
        except:
            d["Area"]=None

        try:
            d["Full
Baths"]=item.find("span",{"class","infoValueFullBath"}).find("b").text
        except:
            d["Full Baths"]=None
        try:
            d["Half
Baths"]=item.find("span",{"class","infoValueHalfBath"}).find("b").text
        except:
            d["Half Baths"]=None
        for column_group in item.find_all("div",{"class":"columnGroup"}):
            for feature_group, feature_name in
zip(column_group.find_all("span",{"class":"featureGroup"}),column_group.fin
```

```
d_all("span",{"class":"featureName"})):
                if "Lot Size" in feature_group.text:
                    d["Lot Size"]=feature_name.text
        l.append(d)

import pandas
df=pandas.DataFrame(l)
df.to_csv("Output.csv")
```

Application #8: Building a Web Graph of Stock Market Data

#script1.py

```
from flask import Flask, render_template
app=Flask(__name__)
@app.route('/plot/')
def plot():
    from pandas_datareader import data
    import datetime
    from bokeh.plotting import figure, show, output_file
    from bokeh.embed import components
    from bokeh.resources import CDN
    start=datetime.datetime(2015,11,1)
    end=datetime.datetime(2016,3,10)

df=data.DataReader(name="GOOG",data_source="yahoo",start=start,end=end)
    def inc_dec(c, o):
        if c > o:
            value="Increase"
        elif c < o:
            value="Decrease"
        else:
            value="Equal"
        return value
    df["Status"]=[inc_dec(c,o) for c, o in zip(df.Close,df.Open)]
    df["Middle"]=(df.Open+df.Close)/2
    df["Height"]=abs(df.Close-df.Open)
    p=figure(x_axis_type='datetime', width=1000, height=300,
responsive=True)
    p.title="Candlestick Chart"
    p.grid.grid_line_alpha=0.3
    hours_12=12*60*60*1000
    p.segment(df.index, df.High, df.index, df.Low, color="Black")

p.rect(df.index[df.Status=="Increase"],df.Middle[df.Status=="Increase"],
           hours_12,
df.Height[df.Status=="Increase"],fill_color="#CCFFFF",line_color="black")

p.rect(df.index[df.Status=="Decrease"],df.Middle[df.Status=="Decrease"],
           hours_12,
df.Height[df.Status=="Decrease"],fill_color="#FF3333",line_color="black")
    script1, div1 = components(p)
    cdn_js=CDN.js_files[0]
    cdn_css=CDN.css_files[0]
    return render_template("plot.html",
    script1=script1,
    div1=div1,
    cdn_css=cdn_css,
    cdn_js=cdn_js )
@app.route('/')
def home():
    return render_template("home.html")
@app.route('/about/')
def about():
    return render_template("about.html")
if __name__=="__main__":
    app.run(debug=True)
```

#templates/layout.html

```
<!DOCTYPE html>
<html>
  <head>
      <title>Flask App</title>
      <link rel="stylesheet"
href="{{url_for('static',filename='css/main.css')}}"
  </head>
  <body>
    <header>
      <div class="container">
        <h1 class="logo">Ardit's web app</h1>
        <strong><nav>
          <ul class="menu">
            <li><a href="{{ url_for('home') }}">Home</a></li>
            <li><a href="{{ url_for('about') }}">About</a></li>
          </ul>
        </nav></strong>
      </div>
    </header>
    <div class="container">
        {%block content%}
        {%endblock%}
    </div>
  </body>
</html>
```

#templates/home.html

```
{%extends "layout.html"%}
{%block content%}
<div class="home">
    <h1>My homepage</h1>
    <p>This is a test website</p>
</div>
{%endblock%}
```

#templates/about.html

```
{%extends "layout.html"%}
{%block content%}
<div class="about">
    <h1>My about page</h1>
    <p>This is a test website again</p>
</div>
{%endblock%}
```

#templates/plot.html

```
{%extends "layout.html"%}
{%block content%}
<link rel="stylesheet" href={{cdn_css | safe}} type="text/css" />
<script type="text/javascript" src={{cdn_js | safe}}></script>
<div class="about">
    <h1>My about page</h1>
    <p>This is a test website again</p>
</div>
{{script1 | safe}}
{{div1 | safe}}
{%endblock%}
```

#static/main.css

```
body {
  margin: 0;
  padding: 0;
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  color: #444;
}
/*
 * Formatting the header area
 */
header {
  background-color: #DFB887;
  height: 35px;
  width: 100%;
  opacity: .9;
  margin-bottom: 10px;
}
header h1.logo {
  margin: 0;
  font-size: 1.7em;
  color: #fff;
  text-transform: uppercase;
  float: left;
}
header h1.logo:hover {
  color: #fff;
  text-decoration: none;
}
/*
 * Center the body content
 */
.container {
  width: 1200px;
  margin: 0 auto;
}
div.home {
  padding: 10px 0 30px 0;
  background-color: #E6E6FA;
  -webkit-border-radius: 6px;
     -moz-border-radius: 6px;
```

```
            border-radius: 6px;
}
div.about {
  padding: 10px 0 30px 0;
  background-color: #E6E6FA;
  -webkit-border-radius: 6px;
     -moz-border-radius: 6px;
          border-radius: 6px;
}
h2 {
  font-size: 3em;
  margin-top: 40px;
  text-align: center;
  letter-spacing: -2px;
}
h3 {
  font-size: 1.7em;
  font-weight: 100;
  margin-top: 30px;
  text-align: center;
  letter-spacing: -1px;
  color: #999;
}
.menu {
  float: right;
  margin-top: 8px;
}
.menu li {
  display: inline;
}
.menu li + li {
  margin-left: 35px;
}
.menu li a {
  color: #444;
  text-decoration: none;
}
```

Application #9: Building a Data Collector Web App

#app.py

```
from flask import Flask, render_template, request
from flask.ext.sqlalchemy import SQLAlchemy
from send_email import send_email
from sqlalchemy.sql import func
app=Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI']='postgresql://postgres:postgres123@lo
calhost/height_collector'
db=SQLAlchemy(app)
class Data(db.Model):
    __tablename__="data"
    id=db.Column(db.Integer, primary_key=True)
    email_=db.Column(db.String(120), unique=True)
    height_=db.Column(db.Integer)
    def __init__(self, email_, height_):
        self.email_=email_
        self.height_=height_
@app.route("/")
def index():
    return render_template("index.html")
@app.route("/success", methods=['POST'])
def success():
    if request.method=='POST':
        email=request.form["email_name"]
        height=request.form["height_name"]
        if db.session.query(Data).filter(Data.email_==email).count() == 0:
            data=Data(email,height)
            db.session.add(data)
            db.session.commit()

average_height=db.session.query(func.avg(Data.height_)).scalar()
            average_height=round(average_height,1)
            count=db.session.query(Data.height_).count()
            send_email(email, height, average_height, count)
            return render_template("success.html")
    return render_template('index.html',
    text="Seems like we've got something from that email address already!")
if __name__ == '__main__':
    app.debug=True
    app.run()
```

#send_email.py

```
from email.mime.text import MIMEText
import smtplib
def send_email(email, height, average_height, count):
    from_email="arditsulce@gmail.com"
    from_password="Top1Gear2"
    to_email=email
    subject="Height data"
    message="Hey there, your height is <strong>%s</strong>. <br> Average
height of all is <strong>%s</strong> and that is calculated out of
<strong>%s</strong> people. <br> Thanks!" % (height, average_height,
count)
    msg=MIMEText(message, 'html')
    msg['Subject']=subject
    msg['To']=to_email
    msg['From']=from_email
    gmail=smtplib.SMTP('smtp.gmail.com',587)
    gmail.ehlo()
    gmail.starttls()
    gmail.login(from_email, from_password)
    gmail.send_message(msg)
```

#index.html

```
<!DOCTYPE html>
<html lang="en">
  <title> Data Collector App</title>
  <head>
    <link href="../static/main.css" rel="stylesheet">
  </head>
    <body>
      <div class="container">
        <h1>Collecting height</h1>
        <h3>Please fill the entires to get population statistics on
height</h3>
        <div class="message">
          {{text | safe}}
        </div>
        <form action="{{url_for('success')}}" method="POST">
          <input title="Your email will be safe with us"
placeholder="Enter your email address" type="email" name="email_name"
required> <br>
          <input title="Your data will be safe with us" placeholder="Enter
your height in cm" type="number" min="50", max="300" name="height_name"
required> <br>
          <button type="submit"> Submit </button>
        </form>
      </div>
    </body>
</html>
```

#success.html

```
<!DOCTYPE html>
<html lang="en">
  <title> Data Collector App</title>
  <head>
    <link href="../static/main.css" rel="stylesheet">
  </head>
    <body>
      <div class="container">
        <p> Thank you for your submission! <br>
          You will receive an email with the survey results shortly.
        </p>
      </div>
    </body>
</html>
```

#static/main.css

```
html, body {
  height: 100%;
  margin: 0;
}
.container {
  margin: 0 auto;
  width: 100%;
  height: 100%;
  background-color: #006666;
  color: #e6ffff;
  overflow:hidden;
  text-align: center;
}
.container h1 {
  font-family: Arial, sans-serif;
  font-size: 30px;
  color: #DDCCEE;
  margin-top: 80px;
}
.container form {
  margin: 20px;
}
.container input {
  width: 350px;
  height: 15px;
  font-size: 15px;
  margin: 2px;
  padding: 20px;
  transition: all 0.2s ease-in-out;
}
.container button {
  width:70px;
  height: 30px;
  background-color: steelblue;
  margin: 3px;
}
.container p {
  margin: 100px;
}
.message {
  font-family: Arial, sans-serif;
  font-size: 15px;
  color: #ff9999
}
```

APPLICATION #10: Building a Geocoder Web Service

#app.py

```
from flask import Flask, render_template, request, send_file
from geopy.geocoders import Nominatim
import pandas
import datetime
app=Flask(__name__)
@app.route("/")
def index():
    return render_template("index.html")
@app.route('/success-table', methods=['POST'])
def success_table():
    global filename
    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 else None)
            df['Longitude'] = df['coordinates'].apply(lambda x:
x.longitude if x != None else None)
            df=df.drop("coordinates",1)
            filename=datetime.datetime.now().strftime("uploads/%Y-%m-%d-
%H-%M-%S-%f"+".csv")
            df.to_csv(filename,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 address column in your CSV file!")
@app.route("/download-file/")
def download():
    return send_file(filename, attachment_filename='yourfile.csv',
as_attachment=True)
if __name__=="__main__":
    app.run(debug=True)
```

#index.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 containing addresses
should be in a column named <em>address</em> or <em>Address</em></h3>
        <form action="{{url_for('success_table')}}" method="POST"
enctype="multipart/form-data">
          <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>
```

#download.hmtl

```
<!DOCTYPE html>
<html lang="en">
<div class="download">
<a href={{url_for('download')}} target="blank"> <button class="btn">
Download </button></a>
</div>
</html>
```

#main.css

```css
html, body {
  height: 100%;
  margin: 0;
}
.container {
  margin: 0 auto;
  width: 100%;
  height: 100%;
  background-color: #006666;
  color: #e6ffff;
  overflow:hidden;
  text-align: center;
}
.container form {
  margin: 20px;
}
.container h1 {
  font-family: Arial, sans-serif;
  font-size: 30px;
  color: #DDCCEE;
  margin-top: 80px;
}
.container button {
  width:70px;
  height: 30px;
  background-color: steelblue;
  margin: 3px;
}
.container input {
  width: 200px;
  height: 15px;
  font-size: 15px;
  margin: 2px;
  padding: 5px;
  transition: all 0.2s ease-in-out;
}
.output {
  display: inline-block;
}
```