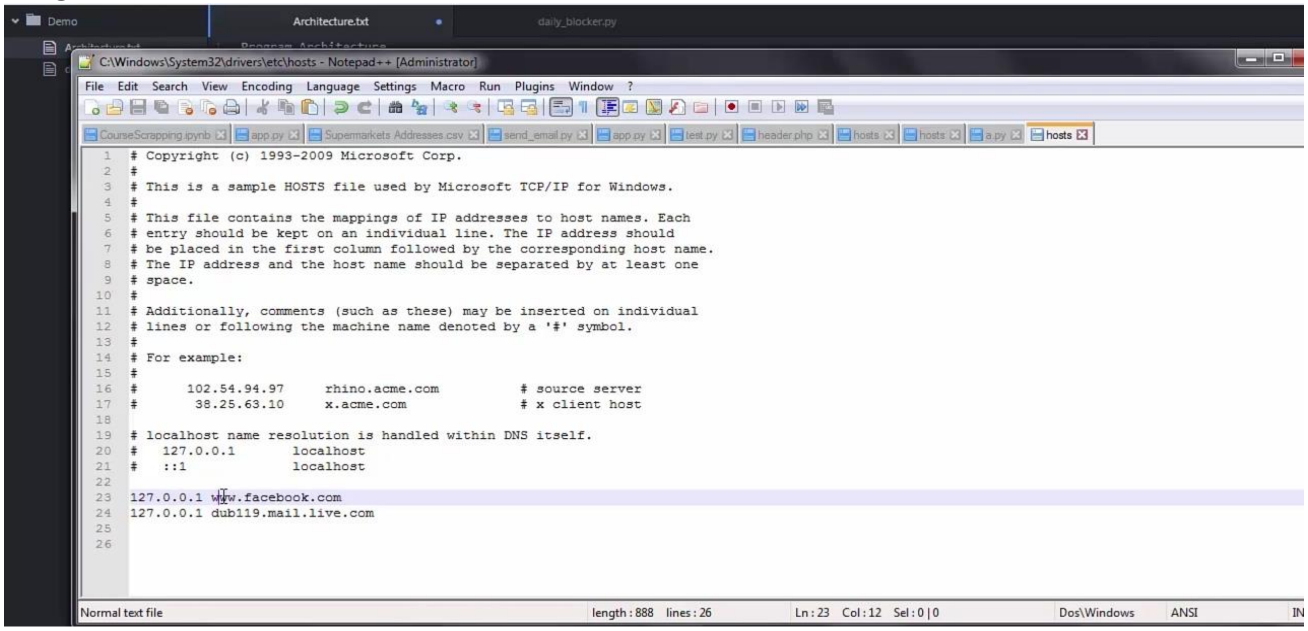
DAILY ASSESSMENT FORMAT

Date:	24th May 2020	**Name:**	Soundarya NA
Course:	UDEMY	**USN:**	4AL16EC077
Topic:	PYTHON: Application 4: Build a personal Website with Python and Flask	**Semester & Section:**	8th - B

AFTERNOON SESSION DETAILS

Image of session

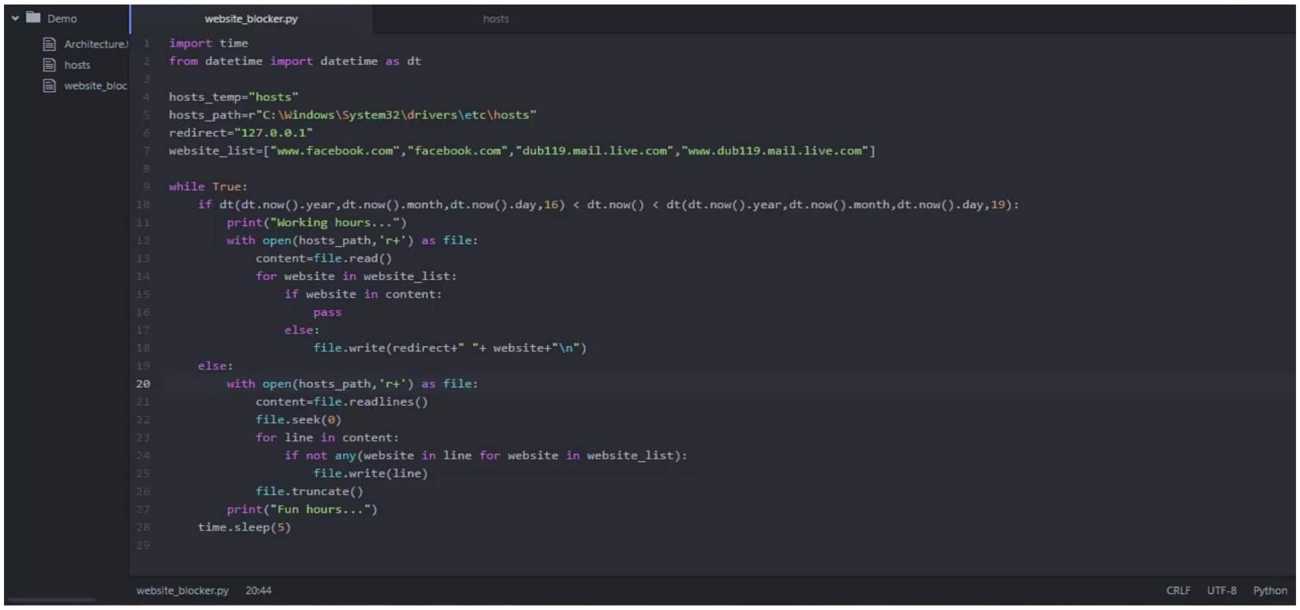


```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#   127.0.0.1       localhost
#   ::1             localhost

127.0.0.1 www.facebook.com
127.0.0.1 dub119.mail.live.com


```



```python
import time
from datetime import datetime as dt

hosts_temp="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,16) < dt.now() < dt(dt.now().year,dt.now().month,dt.now().day,19):
        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)

```

Report:

Code1:

```
from flask import Flask


app = Flask(__name__)


@app.route("/")
def home():
    return "Hello, World!"


@app.route("/salvador")
def salvador():
    return "Hello, Salvador"


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

We will write code that will take care of the server-side processing. Our code will receive requests. It will figure out what those requests are dealing with and what they are asking. It will also figure out what response to send to the user. It makes the process of designing a web application simpler. Flask lets us focus on what the users are requesting and what sort of response to give back.

HTTP is the protocol for websites. The internet uses it to interact and communicate with computers and servers. Let me give you an example of how you use it every day.
When you type the name of a website in the address bar of your browser and you hit enter. What happens is that an HTTP request has been sent to a server.
For example, when I go to my address bar and type google.com, then hit enter, an HTTP request is sent to a Google Server. The Google Server receives the request and needs to figure how to interpret that request. The Google Server sends back an HTTP response that contains the information that my web browser receives. Then it displays what you asked for on a page in the browser.

Code2:

```
<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <title>About Flask</title>

  </head>

  <body>

    {% extends "template.html" %}

    {% block content %}


    <h1> About Flask </h1>

    <p> Flask is a micro web framework written in Python.</p>

    <p> Applications that use the Flask framework include Pinterest,

      LinkedIn, and the community web page for Flask itself.</p>


    {% endblock %}

  </body>

</html>
```

Code3:

Linking our CSS with our HTML file

```
<!DOCTYPE html>

<html lang="en" dir="ltr">

  <head>

    <meta charset="utf-8">

    <title>Flask Parent Template</title>


    <link rel="stylesheet" href="{{ url_for('static',    filename='css/template.css') }}">


</head>
```

```
 <body>
  <header>
   <div class="container">
    <h1 class="logo">First 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>

{% block content %}
{% endblock %}

 </body>
</html>
```

Why use virtualenv?

You may use Python for others projects besides web-development. Your projects might have different versions of Python installed, different dependencies and packages. We use virtualenv to create an isolated environment for your Python project. This means that each project can have its own dependencies regardless of what dependencies every other project has.

Deploy Your Web Application to the Cloud:

To deploy our web application to the cloud, we will use Google App Engine (Standard Environment). This is an example of a Platform as a Service (PaaS). PaaS refers to the delivery of operating systems and associated services over the internet without downloads or installation. The approach lets customers create and deploy applications without having to invest in the underlying infrastructure (More info on PaaS check out TechTarget).

Activating the virtual environment:

Now go to your terminal or command prompt. Go to the directory that contains the file called activate. The file called activate is found inside a folder called Scripts for Windows and bin for OS X and Linux.

The Application:

Now check the URL of your application. The application will be store in the following way:

"your project id".appspot.com

Conclusion:

- Use the framework called Flask to use Python as a Server Side Language.
- Learned how to use HTML, CSS, and Flask to make a website.
- Learned how to create Virtual Environments using virtualenv.
- Use Google App Engine Standard Environment to deploy an application to the cloud.