PHISHING-1

What is our GOAL for this MODULE?

In this class, we learned about phishing and we created flask server to store user's information

What did we ACHIEVE in the class TODAY?

- Flask server
- Get command used in flask server

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Flask
- App route
- Get command to retrieve user data
- Csv format

How did we DO the activities?

1. Import required libraries

   - import Flask and request
   - import url_for : **'url_for**" generates a URL to an endpoint according to the method provided.
   - import render_template : This function generates output from a template
   - import jsonify : Handles "JSON" data properly using Flask's ***jsonify()*** method
   - import csv : reads and writes tabular data in CSV format.

```
from flask import Flask, redirect, url_for, request, render_template, jsonify
import csv
```

2. Call the main function

   - The debug parameter is set to true. This will help track down possible Python errors on the web page

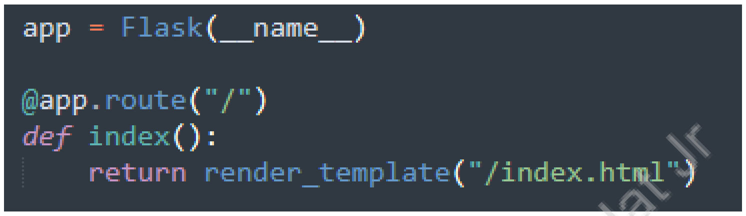
```
if __name__ == "__main__":
    app.run(debug = True)
```

3. Create the Flask class, and create a new instance of it. And pass the argument, the "***_name_*** **"**.Flask needs this information so it knows where to look for resources such as templates and static files.

   - With "***route()"***, we tell Flask which URL should run our function.
   - Next, we are creating a function "***index"*** that returns the (index.html) that will be created later to design our webpage The function is mapped to the home using **'/' URL.** This means when the user navigates to ***"localhost:5000",*** the

home function will run and the output will be displayed on the webpage.

- Using the ***"render_template"*** method from the flask framework, we passed an HTML file to the method and it returned to the browser when the user visits the "URL" associated with that template.

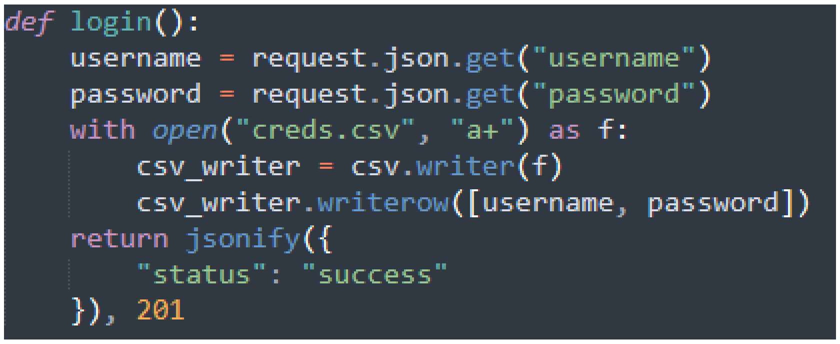


```
app = Flask(__name__)

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

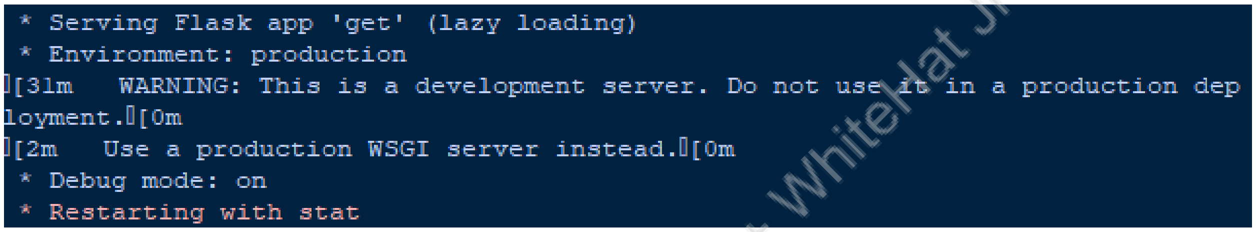
4. In the next step, we will create the login function, whose primary function is to retrieve the username and password and to store the same in the csv format

   - Initialize variable ***"username"*** which will request for json data using **get ()** *method.* ***get()*** *is used to request data from a specified resource.*
   - Initialize variable ***"password"*** which will request for json data using **get ()** *method*
   - By using open() we can access csv files, and by using "a+" we can append usernames and passwords inside csv files.
   - "CSV" or (Comma-separated values) files are text files that contain a list of values (or fields) separated by commas. CSV is a common data exchange format used by many applications., HTML, JSON and others are also common data exchange formats.
   - For writing data inside csv files will use ***"writer()"***
   - CSV represents data in tabular form and we want to write data in row format by using **writerow()** method
   - ***"writerow()"*** will write username and password inside csv file
   - Return json objects
   - If data was successfully inserted in csv files, so show status success



```python
def login():
    username = request.json.get("username")
    password = request.json.get("password")
    with open("creds.csv", "a+") as f:
        csv_writer = csv.writer(f)
        csv_writer.writerow([username, password])
    return jsonify({
        "status": "success"
    }), 201
```

5. Now it's time to run the program



```
 * Serving Flask app 'get' (lazy loading)
 * Environment: production
[31m   WARNING: This is a development server. Do not use it in a production dep
loyment.[0m
[2m   Use a production WSGI server instead.[0m
 * Debug mode: on
 * Restarting with stat
```

What's next?

In the next class, you will be learning more about phishing

EXTEND YOUR KNOWLEDGE:

To learn more about phishing click here