PHINMA EDUCATION
MAKING LIVES BETTER THROUGH EDUCATION

ITE 306: Integrative Programming and Technologies
Module #20

Name: ______________________________ Class number: _______
Section: ____________ Schedule: ______________________ Date: ______________

Lesson title: Flask – Templates and Static Files
Lesson Objectives:
At the end of this module, I should be able to:
1.) Know how to integrate Python, HTML, CSS, Javascript and Flask
2.) Create a RESTful application using flash

Materials:
Activity Sheets, computer
References:
- https://www.tutorialspoint.com/
- https://www.javatpoint.com/

Productivity Tip:
"Brain-dump your worries before you study."

A. LESSON PREVIEW/REVIEW

1) Introduction

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program. Applications that use the Flask framework include Pinterest and LinkedIn.

History

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application.

When Ronacher and Georg Brandl created a bulletin board system written in Python, the Pocoo projects Werkzeug and Jinja were developed.

Flask has become popular among Python enthusiasts. As of January 2020, it has more stars on GitHub than any other Python web-development framework, and was voted the most popular web framework in the Python Developers Survey 2018.

2) Activity 1: What I Know Chart, part 1

Try answering the questions below by writing your ideas under the first column *What I Know*. It's okay if you write key words or phrases that you think are related to the questions. Leave the third column blank for Activity 4.

Name: ____ Class number: _______
Section: ____________ Schedule: ______________________________ Date: ________________

What I Know	Questions:	What I Learned (Activity 4)
	1.) What do you think are the advantages of having a template?	
	2.) What do you think is the advantage of using Flask?	

B.MAIN LESSON

1) Activity 2: Content Notes

Flask – Templates

It is possible to return the output of a function bound to a certain URL in the form of HTML. For instance, in the following script, **index()** function will render **'Hello World'** with **<h1>** tag attached to it.

Example #1 without Template

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
   return '<html><body><h1>Hello World</h1></body></html>'

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

However, generating HTML content from Python code is bulky, especially when variable data and Python language elements like conditionals or loops need to be put. This would require frequent escaping from HTML.

This is where one can take advantage of **Jinja2** template engine, on which Flask is based. Instead of returning hardcode HTML from the function, a HTML file can be rendered by the **render_template()** function.

Example #2 with Template

```
from flask import Flask
app = Flask(__name__)

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

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

Name: ____ Class number: _______
Section: ____________ Schedule: ____________________________ Date: ______________

Flask will try to find the HTML file in the templates folder, in the same folder in which this script is present.

- **Application folder**
  - **Hello.py**
  - **templates**
    - **hello.html**

Use these directories for Example #2

The term **'web templating system'** refers to designing an HTML script in which the variable data can be inserted dynamically. A web template system comprises of a template engine, some kind of data source and a template processor.

Flask uses **jinja2** template engine. A web template contains HTML syntax interspersed placeholders for variables and expressions (in these case Python expressions) which are replaced values when the template is rendered.

The following code is saved as **hello.html** in the templates folder.

Example #2 with Template

```
<!doctype html>
<html>
   <body>

      <h1>Hello {{ name }}!</h1>

   </body>
</html>
```

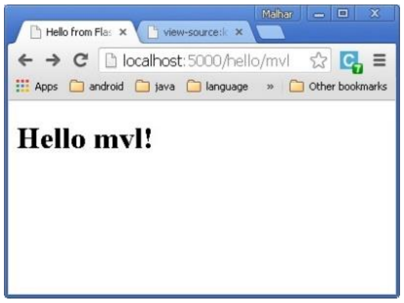
Next, run the following script from Python shell.

Modify Example #2 (cont.)

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/hello/<user>')
def hello_name(user):
   return render_template('hello.html', name = user)

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

As the development server starts running, open the browser and enter URL as − **http://localhost:5000/hello/mvl**

The **variable** part of URL is inserted at **{{ name }}** place holder.



Hello mvl!

Name: ______________________________ Class number: ______
Section: __________ Schedule: ______________________ Date: __________

The **jinja2** template engine uses the following delimiters for escaping from HTML.

- {% ... %} for Statements
- {{ ... }} for Expressions to print to the template output
- {# ... #} for Comments not included in the template output
- # ... ## for Line Statements

In the following example, use of conditional statement in the template is demonstrated. The URL rule to the **hello()** function accepts the integer parameter. It is passed to the **hello.html** template. Inside it, the value of number received (marks) is compared (greater or less than 50) and accordingly HTML is conditionally rendered.

Example 3:

The Python Script is as follows:

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/hello/<int:score>')
def hello_name(score):
   return render_template('hello.html', marks = score)

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

HTML template script of **hello.html** is as follows:

```
<!doctype html>
<html>
   <body>
      {% if marks>50 %}
         <h1> Your result is pass!</h1>
      {% else %}
         <h1>Your result is fail</h1>
      {% endif %}
   </body>
</html>
```

Note that the conditional statements **if-else** and **endif** are enclosed in delimiter **{%..%}**.

Run the Python script and visit URL **http://localhost:5000/hello/60** and then **http://localhost:5000/hello/30** to see the output of HTML changing conditionally.

Name: ______________________________ Class number: _______
Section: ____________ Schedule: ________________________ Date: ________________

The Python loop constructs can also be employed inside the template. In the following script, the **result()** function sends a dictionary object to template **results.html** when URL **http://localhost:5000/result** is opened in the browser.

The Template part of **result.html** employs a **for loop** to render key and value pairs of dictionary object **result{}** as cells of an HTML table.

Example 4:
Run the following code from Python shell.

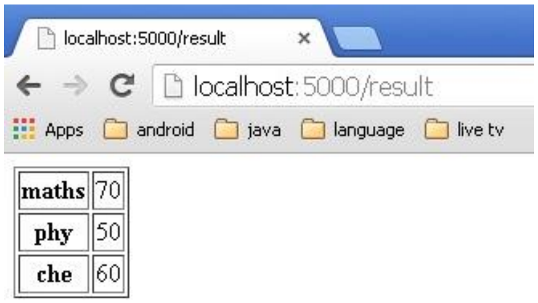
```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/result')
def result():
        dict = {'phy':50,'che':60,'maths':70}
        return render_template('result.html', result = dict)

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

Save the following HTML script as **result.html** in the templates folder.

```
<!doctype html>
<html>
   <body>
      <table border = 1>
         {% for key, value in result.items() %}
            <tr>
               <th> {{ key }} </th>
               <td> {{ value }} </td>
            </tr>
         {% endfor %}
      </table>
   </body>
</html>
```

Here, again the Python statements corresponding to the **For** loop are enclosed in {%..%} whereas, the expressions **key and value** are put inside **{{ }}**.

After the development starts running, open **http://localhost:5000/result** in the browser to get the following output.



maths	70
phy	50
che	60

Name: ______________________________ Class number: ______
Section: __________ Schedule: ____________________ Date: __________

Flask – Static Files

A web application often requires a static file such as a **javascript** file or a **CSS** file supporting the display of a web page. Usually, the web server is configured to serve them for you, but during the development, these files are served from *static* folder in your package or next to your module and it will be available at ***/static*** on the application.

A special endpoint 'static' is used to generate URL for static files.

In the following example, a **javascript** function defined in **hello.js** is called on **OnClick** event of HTML button in **index.html**, which is rendered on **'/'** URL of the Flask application.

- **Application folder**
  - **runme.py**
  - **templates**
    - **index.html**
  - **static**
    - **hello.js**

Create the following directories for Example #5

Example 5:

```
from flask import Flask, render_template
app = Flask(__name__)

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

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

The HTML script of **index.html** is given below.

```
<html>
   <head>
      <script type = "text/javascript"
         src = "{{ url_for('static', filename = 'hello.js') }}" ></script>
   </head>

   <body>
      <input type = "button" onclick = "sayHello()" value = "Say Hello" />
   </body>
</html>
```

hello.js contains **sayHello()** function.

```
function sayHello() {
   alert("Hello World")
}
```

Name: ______________________________________ Class number: ______
Section: __________ Schedule: ________________________ Date: ____________

2) Activity 3: Skill-building Activities
Coding Exercise: Open your code editor and create the following Flask applications.

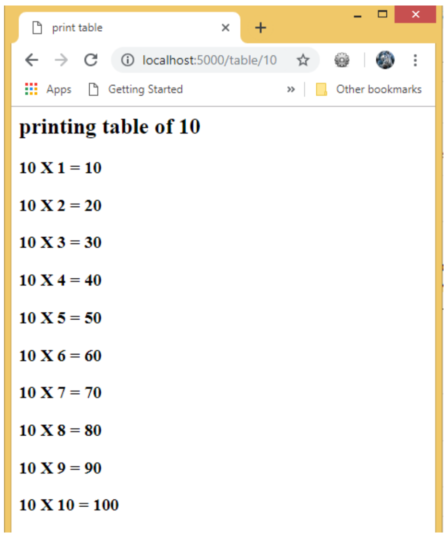
1.) Create a Multiplication Table Flask application.
Print the multiplication table of a number specified in the URL.
Example: the URL *http://localhost:5000/table/10* will print the table of 10 on the browser's window.

script.py

```
from flask import *
app = Flask(__name__)

@app.route('/table/<int:num>')
def table(num):
      return render_template('print-table.html',n=num)
if __name__ == '__main__':
   app.run(debug = True)
```

print-table.html

```
<html>
<head>
<title>print table</title>
</head>
<body>
<h2> printing table of {{n}}</h2>
{% for i  in range(1,11): %}
    <h3>{{n}} X {{i}} = {{n * i}} </h3>
{% endfor %}
</body>
</html>
```



```
printing table of 10
10 X 1 = 10
10 X 2 = 20
10 X 3 = 30
10 X 4 = 40
10 X 5 = 50
10 X 6 = 60
10 X 7 = 70
10 X 8 = 80
10 X 9 = 90
10 X 10 = 100
```

2.) Create a flask script that returns the HTML file, **message.html** which is styled using the stylesheet **style.css**. The flask template system finds the static CSS file under **static/css** directory. Hence the style.css is saved at the specified path.

script.py

```
from flask import *
app = Flask(__name__)

@app.route('/')
def message():
      return render_template('message.html')
if __name__ == '__main__':
      app.run(debug = True)
```

Name: ____ Class number: ______
Section: __________ Schedule: ____________________________ Date: ____________

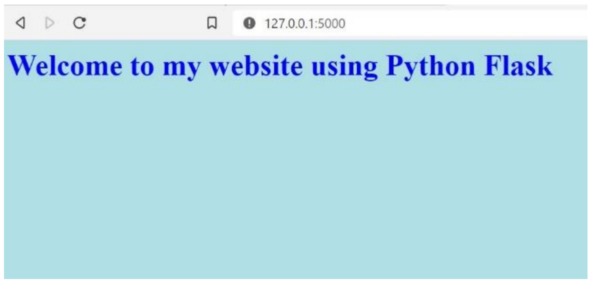
message.html

```
<html>
<head>
    <title>Message</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>

<body>
    <h1>Welcome to my website using Python Flask</h1>
</body>
</html>
```

style.css

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

Output:



Welcome to my website using Python Flask

3) Activity 4: What I Know Chart, part 2
It's time to answer the questions in the *What I Know* chart in Activity 1. Log in your answers in the third column of the table in Activity 1.

4) Activity 5: Check for Understanding
1. What is the jinja2 delimiters for Line Statements in escaping from HTML?
   Answer:____________
2. What is the jinja2 delimiters for Statements in escaping from HTML?
   Answer:____________
3. What is the jinja2 delimiters for Expressions to print to the template output?
   Answer:____________
4. What is the jinja2 delimiters for Comments not included in the template output?
   Answer:____________
5. Flask class runs the application on the local development server, what is the default port number of this server? **Answer:** ______________
6. In Jinja2 a HTML file can be rendered by this function.
   Answer: _________________
7. What delimiter should the conditional statements **if-else** and **endif** be enclosed?
   Answer:____________

Name: ____ Class number: ______
Section: ___________ Schedule: _____________________________ Date: ______________

8. The Python statements corresponding to the **For loop** are enclosed in ______ delimiter.
   Answer:____________
9. It is a popular templating engine for Python. **Answer:**____________
10. Flask class runs the application on the local development server, what is the default host number?
    Answer: _________________

C. LESSON WRAP-UP

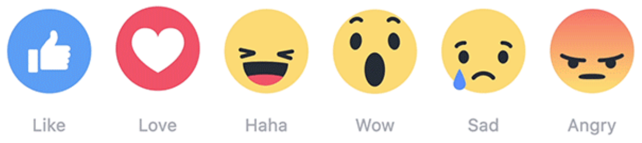
Activity 6: Thinking about Learning

Congratulations for finishing this module! **Shade** the number of the module that you finished to track how much work you have accomplished and how much work there is left to do.

You are done with the session! Let's track your progress.

Period 1											Period 2											Period 3								
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Rate the session for today by encircling the emoji that best captures your experience and write your reason for choosing that emoji.



Reason/s: ____

FAQs

1.) Is flask a Web server?

Answer: *Flask is a micro web application framework. That means it is basically a set of tools and libraries that make it easier to build web applications in Python. Flask does however include a web server that can be used for testing and development.*

2.) Who uses flask Python?

Answer: *Below are the list of some big companies using Flask*

- *Red Hat. Fedora on pagure infrastructure, Openstack, Beaker, Bodhi, Openshift.io, Ansible (Jinja), fedoralovespython.org and many internal tools and APIs.*
- *Rackspace.*
- *Airbnb. Airflow.*
- *Netflix.*
- *PythonAnywhere.*
- *Lyft.*
- *Reddit.*
- *Mailgun.*
- *MIT*
- *Mozilla*
- *Balrog (Application Update Service), Release Engineering Services and others tools and APIs*
- *Hotjar*
- *Patreon*
- *Teradata*
- *Uber*
- *Samsung*
- *Nginx*
- *1.5k more companies in https://stackshare.io/flask/*

Name: ______________________________ Class number: ______
Section: __________ Schedule: ______________________ Date: ____________

KEY TO CORRECTIONS

Answers to Checking for Understanding:

1. What is the jinja2 delimiters for Line Statements in escaping from HTML?
   Answer: # ... ##

2. What is the jinja2 delimiters for Statements in escaping from HTML?
   Answer: *{% ... %}*

3. What is the jinja2 delimiters for Expressions to print to the template output?
   Answer: *{{ ... }}*

4. What is the jinja2 delimiters for Comments not included in the template output?
   Answer: *{# ... #}*

5. Flask class runs the application on the local development server, what is the default port number of this server?
   Answer: *5000*

6. In Jinja2 a HTML file can be rendered by this function.
   Answer: *render_template()*

7. What delimiter should the conditional statements **if-else** and **endif** be enclosed?
   Answer: *{%..%}*

8. The Python statements corresponding to the **For loop** are enclosed in ________ delimiter.
   Answer: *{%..%}*

9. It is a popular templating engine for Python.
   Answer: *Jinja2*

10. Flask class runs the application on the local development server, what is the default host number?
    Answer: *127.0.0.1 (localhost)*