



| Topic | WEB DEVELOPMENT USING PYTHON | |
|--|--|--|
| Class Description | The student will learn about Flask and web development using Python. | |
| Class | PRO C116 | |
| Class time | 45 mins | |
| Goal | <ul style="list-style-type: none"> • Introduction to Flask. • Render HTML webpage using Flask. | |
| Resources Required | <ul style="list-style-type: none"> • Teacher Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone • Student Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen | |
| Class structure | Warm-Up Teacher-Led Activity 1 Student-Led Activity 1 Wrap-Up | 10 mins 10 mins 20 mins 05 mins |
| Credit & Permissions: | Flask by Armin Ronacher and contributors | |
| WARM-UP SESSION - 10 mins | | |
| <div>  </div> <p>Teacher Starts Slideshow Slide 1 to 4 Refer to speaker notes and follow the instructions on each slide.</p> | | |
| Teacher Action | | Student Action |

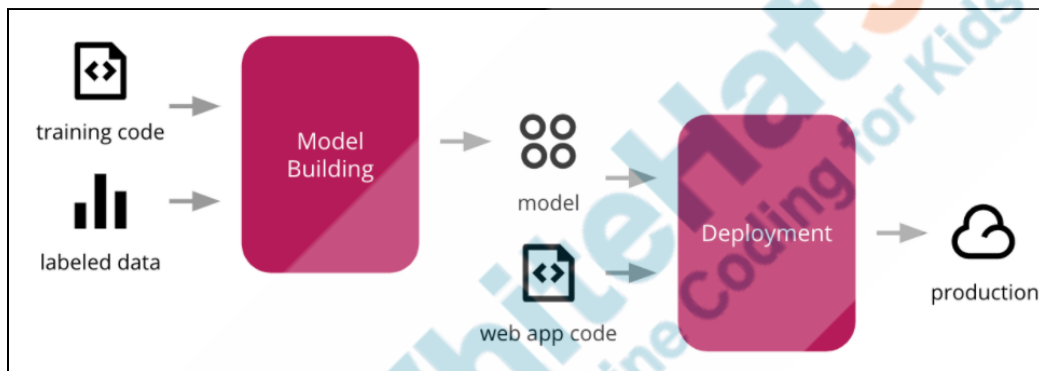
| <p>Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?</p> <p>Following are the WARM-UP session deliverables:</p> <ul style="list-style-type: none"> • Greet the student. • Revision of previous class activities. • Quizzes. | <p>ESR: Hi, thanks! Yes I am excited about it!</p> <p>Click on the slide show tab and present the slides</p> |
|---|---|
| <p style="text-align: center;">WARM-UP QUIZ Click on In-Class Quiz</p> | |
| <p style="text-align: center;">Continue WARM-UP Session Slide 5 to 12</p>  | |
| <p>Following are the session deliverables:</p> <ul style="list-style-type: none"> • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students. | |
| Teacher Action | Student Action |
| <p>We have defined the ML model and also tested it for images and text. Now, you may be surprised by how a real word system works on the ML model.</p> <p>Let me give you an example. When you watch any video on Youtube, it recommends the same category of videos. Similarly, when you buy a product or rather search for any product on any e-commerce website, it recommends the same set of products.</p> <p>A machine learning model is responsible for these suggestions. It is known as the Recommender system. It learns from user input and recommends the videos and products. There is a Machine learning model integrated with the app to perform this task.</p> <p>Similarly, we have created a model for sentiment analysis. Next, we'll be creating a webpage where the user input will</p> | |

be taken and based on our **Sentiment Analysis Model** the sentiments will be displayed on the webpage.

Deployment is the method to integrate a machine learning model into an existing application.

So we'll be **designing our webpage to integrate the Machine Learning model** that we have created.

Production is the final stage of every project. After deployment, the project goes live to be used by the target audience.



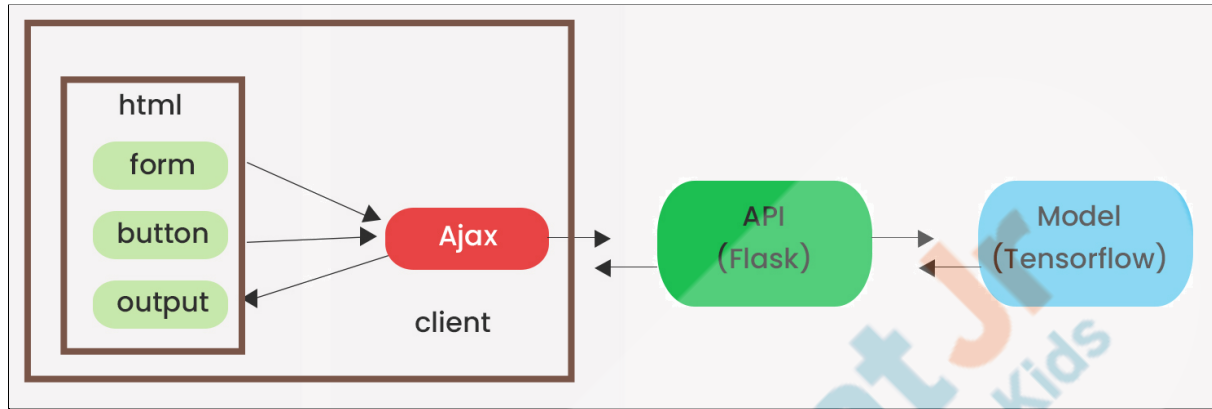
For any website or webpage, as you know, users send requests or data which should be processed and in return, a response is being sent. The **user request** is known as **the client-side** and the **response** is handled by the **server**.

HTML, CSS and **JS** are the **client-side** scripting languages whereas **Python** is the **server-side** scripting language.

To connect the client-side and server-side we'll be using **AJAX**.

AJAX stands for **Asynchronous JavaScript And XML**. It is used to communicate with servers. It can send and receive information in various formats, including **JSON, XML, HTML**, and **text files**.

In today's class we'll be creating HTML page and try to display it using Flask. We'll use AJAX in next class.



Note: Remember to mention the later use of AJAX they will be learning in the next class.

The very first step for this is to create a webpage. Do you know how to create a webpage?

ESR: Yes using **HTML**.

Great, today we are going to learn a method where Python will be used to display or render a web page.

Are you ready to learn about it?

ESR: Yes!



Teacher Ends Slideshow

TEACHER-LED ACTIVITY - 10 mins

Teacher Initiates Screen Share

ACTIVITY

- Introduction to the flask as a framework.
- How to render a web page using flask.

Teacher Action

Student Action

What is Flask?

Flask is a **Python** web framework, it's a Python module that lets you develop web applications easily.

A **framework**, or software framework, is a **platform for developing software applications**. It provides a foundation on which one can build programs for a specific platform.

A framework may include predefined classes and functions that can be used to process input, manage hardware devices, and interact with the system.

For example, a factory builds phones. The machines and workers are included within the factory to build a phone. A framework is like a factory and machines and workers are the predefined functions that serve a specific task. These are used to build phones in our case we are building an API.

Thus, different frameworks are used for different development purposes. In Python, **Django** and **Flask** are the two types of frameworks available for web development using Python.

Flask was originally developed by **Armin Ronacher**. He led a group of international Python practitioners called **Poocco**. It was founded in 2004.

[Teacher Activity 2: Flask documentation](#)

Create a new folder for the project. Create a virtual environment in (**Windows/Mac**) using the following command:

```
python -m venv <name_of_the_environment>
```

Activate the virtual environment using following command:

```
<name_of_the_environment>\Scripts\activate
```

So, to use **Flask** let's install flask using the pip command.

Open the command prompt.

pip install flask

Let's now write our first Flask program. Steps to write a simple flask program:

1. Create a new directory. Open it with VS Code and in it create a python file by name **app.py**.
2. First, we'll import the **Flask** class from the **flask** library. An instance/object of this class will be our application.
3. Next, we create an instance/object of this class by name variable name **app**. Its first argument is the name of the application's module or package. **__name__** is a shortcut that is appropriate for most cases. This is needed so that **Flask** knows where to look for resources such as templates and static files.

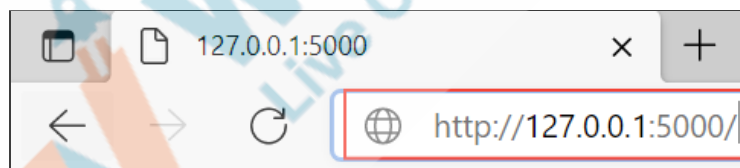
```
app.py > ...  
1  #Importing flask module in the project  
2  from flask import Flask  
3  
4  #Create an object of the Flask class  
5  app = Flask(__name__)  
6
```

4. We then use the **route()** method to tell Flask what URL should trigger our function. **'/'** is bound with the **first_flask()** function. It's the **endpoint** for your function.
5. The function returns the message we want to display in the user's browser. The default content type is **HTML**, so HTML in the string will be **rendered by the browser**.
6. Go to the **command prompt** and run **app.py** python file.

```
app.py > ...
1  #Importing flask module in the project
2  from flask import Flask
3
4  #Create an object of the Flask class
5  app = Flask(__name__)
6
7  #The route() function of the Flask class
8  #'/' URL is bound with first_flask function.
9  @app.route("/")
10 def first_flask():
11     return "This is my first flask program"
12
13 #run the application on local server
14 app.run()
```

```
(Flask_app) C:\Whitehat_jr\PRO C116>python app.py
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

7. Copy the URL (**http://127.0.0.1:5000/**) and open this in any browser. It's the **local host** to run the file on the server.



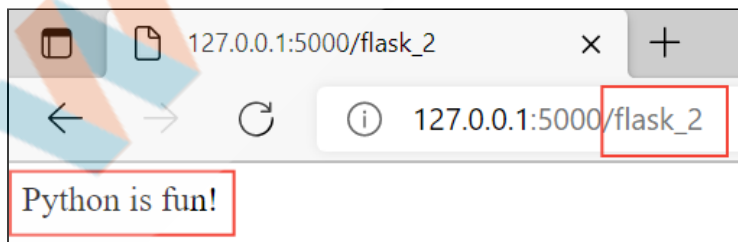
8. The text returned by this function can be seen on the webpage.



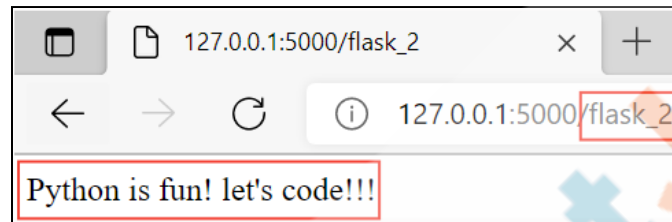
| | |
|--|--|
| <p>Now, do you remember what is meant by an API?</p> <p>Great!!</p> <p>And what is it used for?</p> <p>As the name suggests, Application Programming Interface (API) is used as an intermediary between two software or applications.</p> <p>You see here we have created an API that is used for communicating between Python and HTML files. Let's say I want to redirect the page when I am clicking a button on the webpage. Thus, we have to create another API to redirect it.</p> <p>Let's try to create another function with a different endpoint (/flask_2).</p> <p>In the app.run() function we set debug=True. This is an argument that can be given in the app.run() method. We don't want to restart our app every time we make changes so it is used to automatically restart the app.</p> <p>Save the file and go to the command prompt and run it.</p> | <p>ESR: Yes, Application Programming Interface.</p> <p>ESR: Varied</p> |
|--|--|


```
app.py > ...
1  #Importing flask module in the project
2  from flask import Flask
3
4  #Create an object of the Flask class
5  app = Flask(__name__)
6
7  #The route() function of the Flask class
8  # '/' URL is bound with first_flask function.
9  @app.route("/")
10 def first_flask():
11     return "This is my first flask program"
12
13 #run the application on local server
14 app.run()
15
16 #Define a function with different endpoint "/flask_2"
17 @app.route("/flask_2")
18 def second_flask():
19     return "Python is fun!"
20
21 #run using debug argument
22 app.run(debug=True)
23
```

It will give the same URL. But since we have changed our endpoint, write the name of your endpoint to display the result.



```
15 #Define a function with different endpoint
16
17 @app.route("/flask_2")
18 def second_flask():
19     return "Python is fun! let's code!!!"
20
21 #run using debug argument
22 app.run(debug=True)
```



When you visit websites, many elements are present such as images, videos buttons, text, tables, etc. you have already learned to create web pages using **HTML**.

How will it be if we can create an **HTML** page the way we used to and then let's display that page using **Flask**?

To do so we need to create two folders named **static** and **templates**.

Static folder:

This folder is public, clients can access it. The static folder on the other hand contains public content like images, CSS, javascript, and other files. These files can also be viewed using the **www. website-url/static** address.

Templates:

This folder is private, the client can't access it. All the sensitive data like code files and script files are kept in it. **Flask** uses its template folder for storing the raw templates which can be filled through the python program. By default, **Flask** makes the static folder public and the templates folder private. You can make these two folders where your python file is kept.

Our next step will be to create an **HTML** page. After creating our page we will render it using the **render_template()** method.

render_template():

The **render_template()** method is used to return a template(HTML page) through our python file instead of just returning any text.

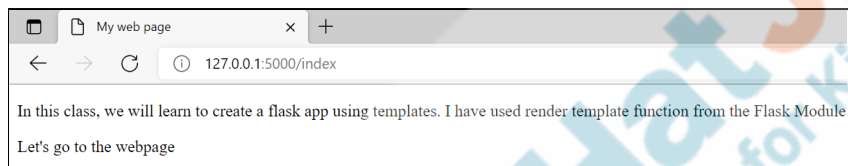
We have to create the HTML file in the templates folder. In VS code create a file **index.html**. Write the HTML heading and a paragraph and save it.

```
templates > <> index.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>My web page</title>
5  </head>
6  <body>
7
8  <p>In this class, we will learn to create a flask app using templates.
9  |   I have used render template function from the Flask Module</p>
10
11 <p>Let's go to the webpage</p>
12
13 </body>
14 </html>
```

The next step is to render this HTML page using the python file.

1. Create a python file with the name **flak_index.py**.
2. Import **render_template** class from **flask**.
3. Define the variable name '**app**' and create an instance or object of class **Flask**.
4. Write **@app.route("/index")** for creating the endpoint for your webpage.
5. Define a function **first_webpage()**. In this function return the **render_template()** method. In this method write the name of your HTML file.
6. Use the **app.run()** method to run the app.

```
flask_index.py > ...
1  from flask import Flask, render_template
2
3  app = Flask(__name__)
4
5  @app.route("/index")
6
7  #in the function return render_template('index.html')
8
9  def first_webpage():
10
11     return render_template('index.html')
12
13  app.run(debug=True)
```



My web page x +

127.0.0.1:5000/index

In this class, we will learn to create a flask app using templates. I have used render template function from the Flask Module

Let's go to the webpage

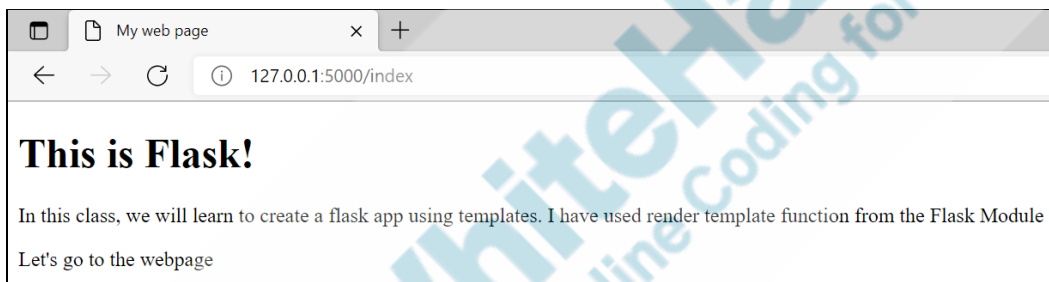
Now, we can pass variables to the HTML file using flask. This will help us to update the variable once in the python file and it'll be automatically updated at all the places wherever it is present on the HTML page.

We will first make a variable **name= 'Flask'**.

Then in the **render_template()** function pass it as an argument **index_variable=name**.

```
flask_index.py > ...
1  from flask import Flask, render_template
2  app = Flask(__name__)
3
4  #in the function return render_template('index.html')
5
6  @app.route("/index")
7  def first_webpage():
8      #Create a variable
9      name = 'Flask'
10     # Pass the variable in the template
11     return render_template('index.html', index_variable = name)
12  app.run(debug=True)
```

```
templates > <> index.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>My web page</title>
5  </head>
6  <body>
7
8  </passing variable in the webpage/>
9
10 <h1>This is {{ index_variable }}!</h1>
11
12 <p>In this class, we will learn to create a flask app using templates.
13 |   I have used render template function from the Flask Module</p>
14
15 <p>Let's go to the webpage</p>
16
17 </body>
18 </html>
```



As mentioned before, we have a static folder where we can store images that are to be displayed on the webpage. Add an image in the **static folder**.

Let's display the image on the same webpage.

Use an image tag to display the image. For the source of the image, we will be using the `url_for()` function. It is very useful for building a URL for a specific element. Here we are using it to display the image on a webpage.

In this function, we have to pass the folder and file name. Our image is stored in the **static** folder, by the name **'1.png'**.

```
templates > <> index.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>My web page</title>
5  </head>
6  <body>
7
8  </passing variable in the webpage/>
9
10 <h1>This is {{ My_name}}!</h1>
11
12 </Adding image to the webpage/>
13 
14
15 <p>In this class, we will learn to create a flask app using templates.
16 |   I have used render template function from the Flask Module</p>
17
18 <p>Let's go to the webpage</p>
19
20 </body>
21 </html>
```



Teacher Stops Screen Share

So now it's your turn.
Please share your screen with me.



Teacher Starts Slideshow

Slide 13 to 14

<Note: Only Applicable for Classes with VA>

Refer to speaker notes and follow the instructions on each slide.

We have one more class challenge for you.

For today's project, you will have to create an HTML webpage with a heading, a text input and a button!

Can you solve it?

Let's try. I will guide you through it.

ESR: Yes



Teacher Ends Slideshow

STUDENT-LED ACTIVITY - 20 mins

- Ask the student to press the ESC key to come back to the panel.
- Guide the student to start Screen Share.
- The teacher gets into Full Screen.

Student Initiates Screen Share

ACTIVITY

- Create a simple HTML page
- Create a Flask API to pass variable to HTML page
- Render HTML page using API

Teacher Action

Student Action

1. Open [Student activity1](#) for the boilerplate code.
2. Copy and paste the code in VS code.
3. Rename the file and save it in a folder.
4. Create a virtual environment in **(Windows/Mac)** using the following command:

```
python -m venv <name_of_the_environment>
```

5. Activate the virtual environment using following command:

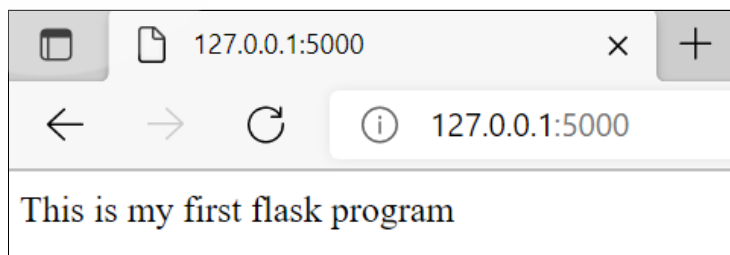
```
<name_of_the_environment>\Scripts\activate
```

6. Install flask using the following command:

```
pip install flask
```

7. Run the python file in the command prompt copy and paste the URL in any browser to run the file locally.
8. You can see the text displayed in the browser.

```
Boilerplate_code.py > ...
1  #Importing flask module in the project
2  from flask import Flask
3
4  #Create an object of the Flask class
5  app = Flask(__name__)
6
7  #The route() function of the Flask class
8  @app.route("/")
9
10 # '/' URL is bound with first_flask function.
11 def first_flask():
12
13     return "This is my first flask program"
14
15 #run the application on local server
16 app.run()
```

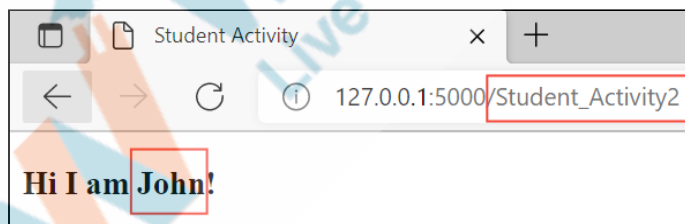


9. Now, create a **static** and **templates** folder in the same folder in which you have saved the Python file.
10. Now in your template folder, you can add an HTML file by any name.
11. Give a **heading** by passing a variable in the file.

```
templates > <> Student_Activity2.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>Student Activity</title>
5  </head>
6  <body>
7  |
8  |   <h3>Hi I am {{student_name}}!</h3>
9  |
10 </body>
11 </html>
```

Note: Refer to [Student Activity 2](#) for the sample solution.

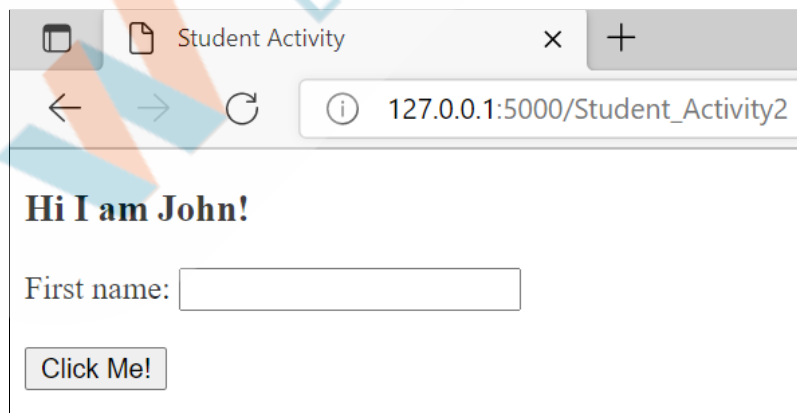
12. Now in your python file, create a variable to pass it to the HTML file. Also, change the endpoint.
13. Run the file in the command prompt and check the result on the localhost URL. **Do remember to change the endpoint.**



14. Create a **text input** using the **form** tag.
15. Now create a button as well. Use **button** tag for it.

```
templates > <> Student_Activity2.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>Student Activity</title>
5  </head>
6  <body>
7
8  |   <h3>Hi I am {{student_name}}!</h3>
9
10 |   <form>
11 |       <label for="fname">First name:</label>
12 |       <input type="text" id="fname" name="fname"><br><br>
13 |   </form>
14
15 |   <button type="button">Click Me!</button>
16
17 </body>
18 </html>
```

```
Student_Activity2.py > ...
1  from flask import Flask, render_template
2  app = Flask(__name__)
3
4  #in the function return render_template('index.html')
5
6  @app.route("/Student_Activity2")
7  def student_webpage():
8  |   #Create a variable
9  |   name = 'John'
10 |   # Pass the variable in the template
11 |   return render_template('Student_Activity2.html', student_name = name)
12 app.run(debug=True)
```



Student Activity

127.0.0.1:5000/Student_Activity2

Hi I am John!

First name:

Click Me!

Well done!!!

As you can see, the HTML page is rendered.

You can stop sharing your screen now.

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 mins

Teacher Starts Slideshow

Slide 15 to 20

<Note: Only Applicable for Classes with VA>



Activity details

You performed very well today.

Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ

Click on In-Class Quiz

Continue WRAP-UP Session

Slide 21 to 26

<Note: Only Applicable for Classes with VA>



Activity Details

Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK

- Appreciate and compliment the student for trying to learn a difficult concept.
- Get to know how they are feeling after the session.
- Review and check their understanding.

| Teacher Action | Student Action |
|--|---|
| <p>You get “hats-off” for your excellent work!</p> <p>In the next class, we’ll be learning about Ajax and jQuery.</p> | <p><i>Make sure you have given at least 2 hats-off during the class for:</i></p> <div> <div>Creatively Solved Activities +10</div> <div>Great Question +10</div> <div>Strong Concentration +10</div> </div> |
| <p align="center">PROJECT OVERVIEW DISCUSSION</p> <p align="center">Refer the document below in Activity Links Sections</p> | |
| <p>Teacher Clicks</p> | <p>✕ End Class</p> |

| ACTIVITY LINKS | | |
|---------------------|---------------------------|---|
| Activity Name | Description | Links |
| Teacher Activity 1 | Previous Class Code | https://colab.research.google.com/drive/1-kuwdZlqnczX1aSsFICzAt1BC2g7P6H2?usp=sharing |
| Teacher Activity 2 | Flask documentation | https://flask.palletsprojects.com/en/2.0.x/ |
| Teacher Activity 3 | Reference Code | https://github.com/procodingclass/PRO-C116-Reference-Code |
| Teacher Reference 1 | Project | https://s3-whjr-curriculum-uploads.whjr.online/c101ff29-f7af-488f-b590-a790366cbc7b.pdf |
| Teacher Reference 2 | Project Solution | https://github.com/procodingclass/-Project-Solution-C116.git |
| Teacher Reference 3 | Visual-Aid | https://s3-whjr-curriculum-uploads.whjr.online/47c89e17-34a6-444f-97d5-47947b44559e.html |
| Teacher Reference 4 | In-Class Quiz | https://s3-whjr-curriculum-uploads.whjr.online/96f9d4b0-3847-467a-845f-34d5add78f52.pdf |
| Student Activity 1 | Boilerplate Code | https://github.com/procodingclass/PRO-C116-Student-Boilerplate |
| Student Activity 2 | Student Activity Solution | https://github.com/procodingclass/PRO-C116-Student-Activity-2 |