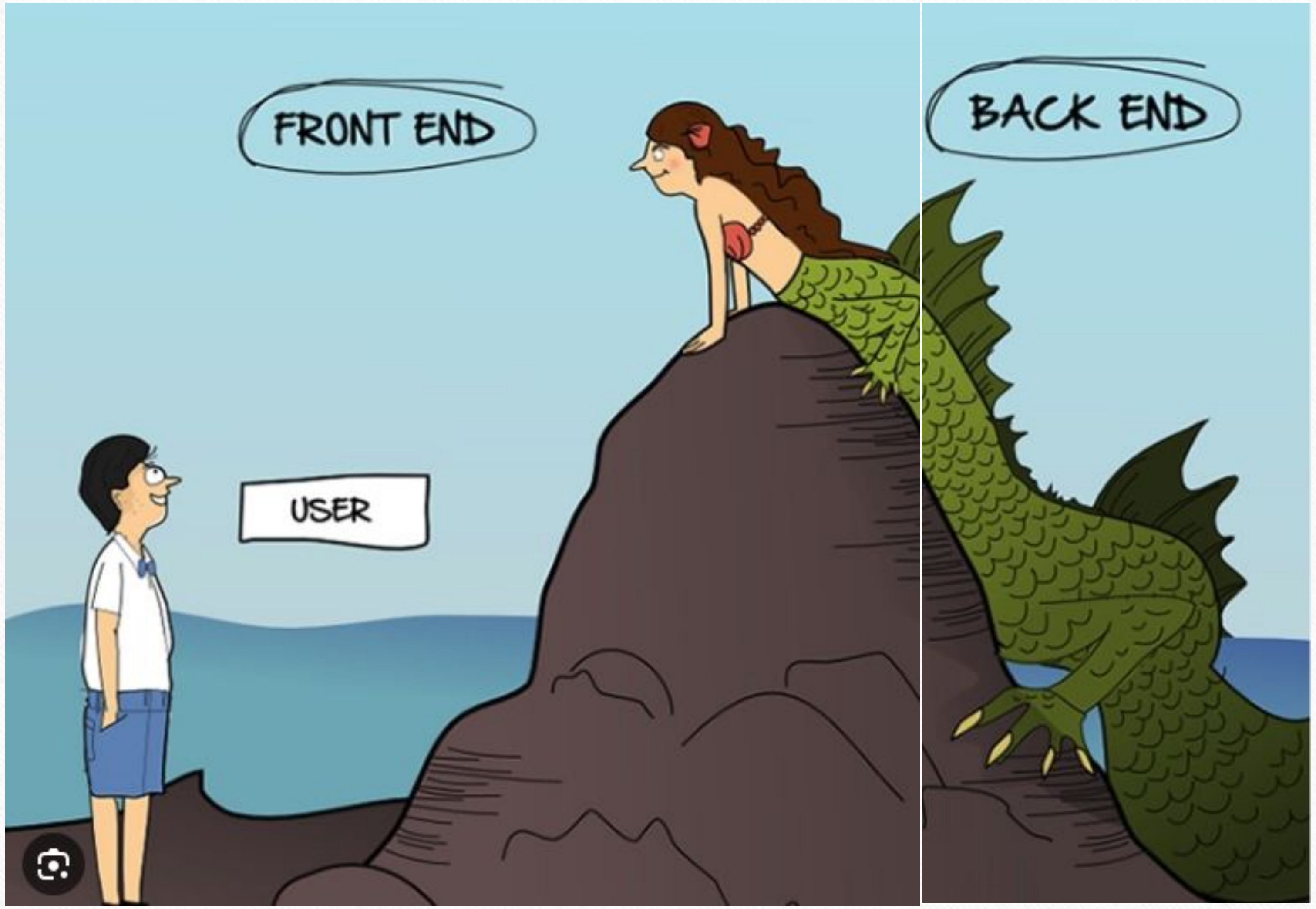
Web Development with Python

-MSE800


FRONT END
BACK END
USER

Full Stack = Front-End + Back-End

Languages – Front-End

HTML 5

CSS 3

JS 5

Languages – Back-End

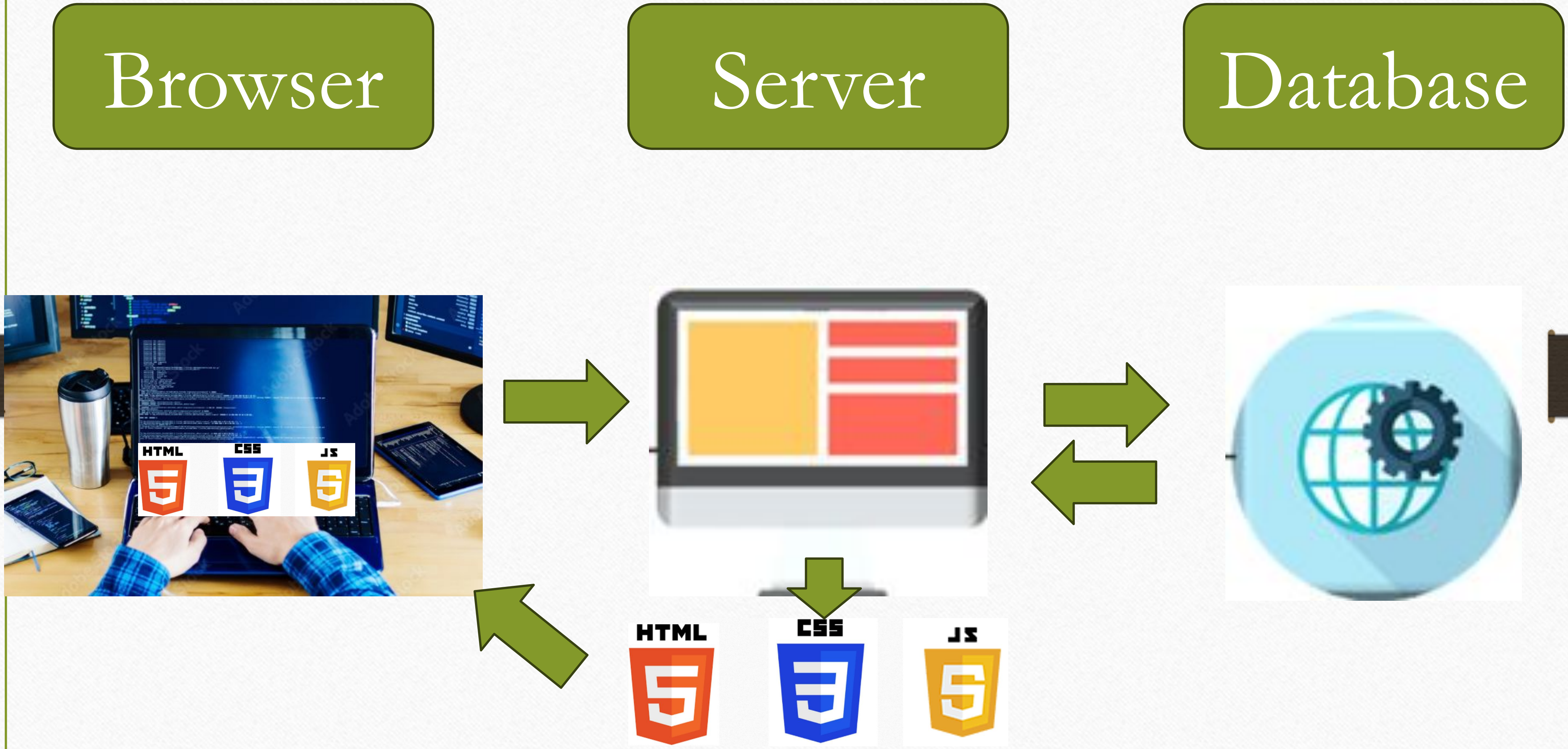
- JavaScript
- Java
- Python
- Ruby…..

Top popular frameworks

Django

Flask

What is the Backend?


Browser
Server
Database
HTML
CSS
JS
HTML
CSS
JS

Develop websites using Flask

One of the most popular web development framework.

Create a .py file and name it as “hello_flask.py”

A minimal Flask application looks like:

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


@app.route("/")
def hello_flask():
    return "<p>Hello, Flask!</p>"
```

https://flask.palletsprojects.com/en/3.0.x/quickstart/

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

@app.route("/")
def hello_flask():
    return "<p>Hello, Flask!</p>"
```

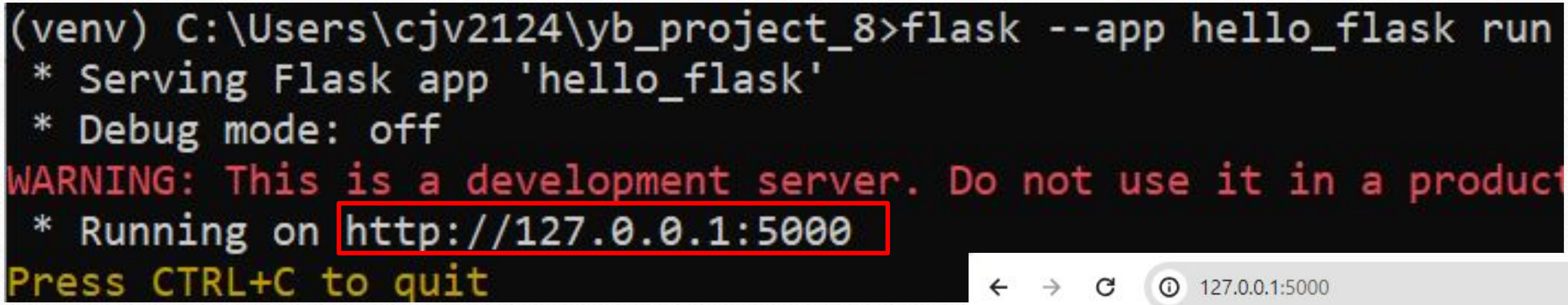
A web framework for Python that provides tools, libraries, and technologies to build a web application.

Uses to determine the root path of the application

Tells Flask what URL should trigger the following function.

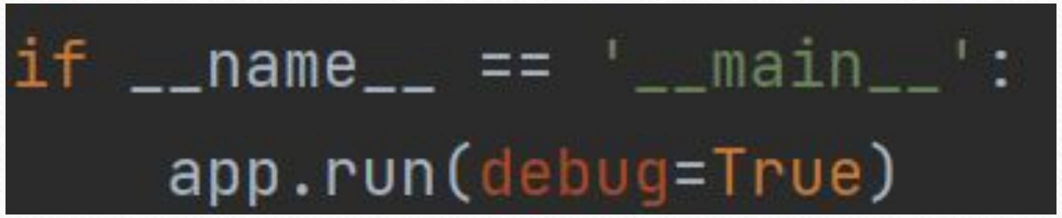
RUN a Flask Application:

1. Command Prompt: flask --app hello_flask run

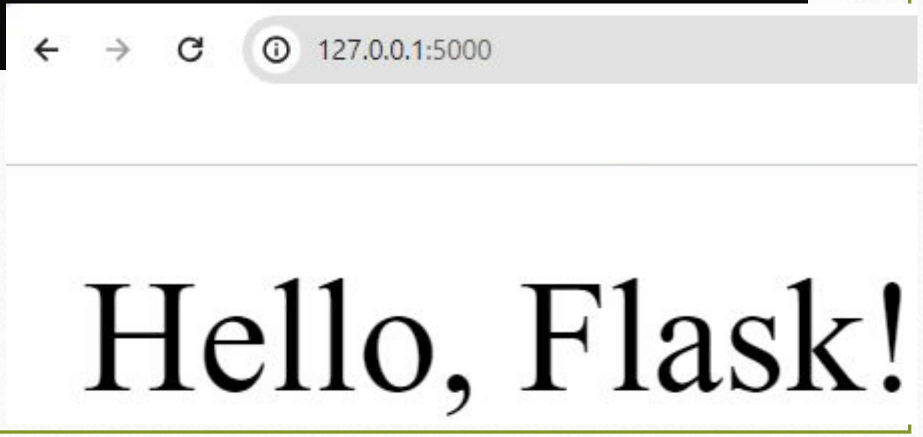


```
(venv) C:\Users\cjv2124\yb_project_8>flask --app hello_flask run
 * Serving Flask app 'hello_flask'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a product
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

2.

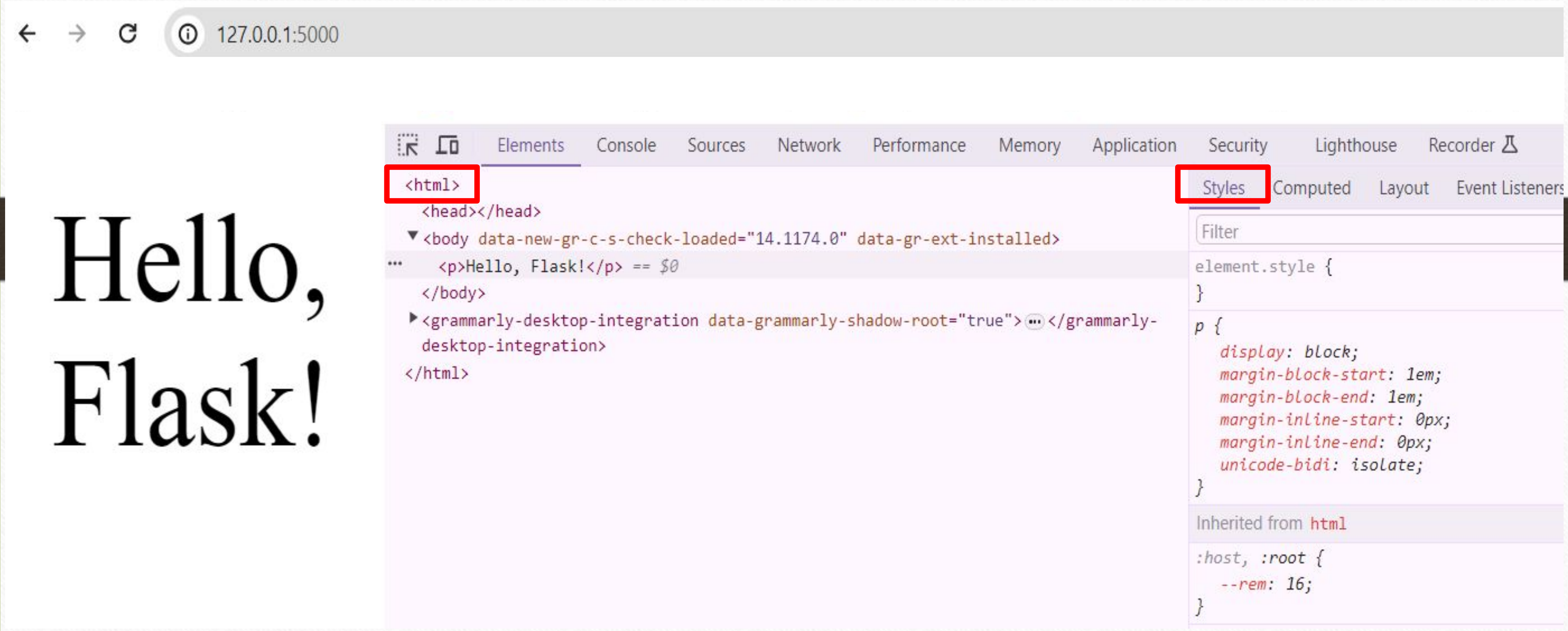


```
if __name__ == '__main__':
    app.run(debug=True)
```



Hello, Flask!

Chrome developer tool



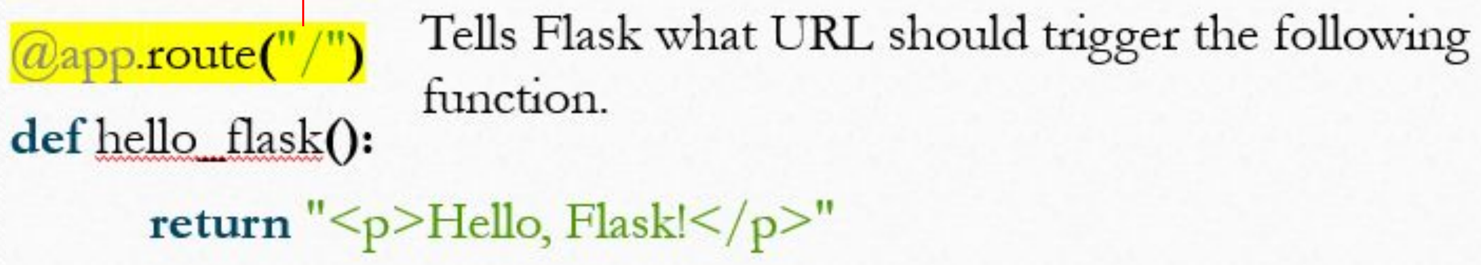
```
<html>
  <head></head>
  ▼<body data-new-gr-c-s-check-loaded="14.1174.0" data-gr-ext-installed>
    <p>Hello, Flask!</p> == $0
  </body>
  ▶<grammarly-desktop-integration data-grammarly-shadow-root="true">…</grammarly-
  desktop-integration>
</html>
```

```
element.style {
}
p {
  display: block;
  margin-block-start: 1em;
  margin-block-end: 1em;
  margin-inline-start: 0px;
  margin-inline-end: 0px;
  unicode-bidi: isolate;
}
Inherited from html
:host, :root {
  --rem: 16;
}
```

Flask URL Path and Flask Debugging

What is URL path in Flask?

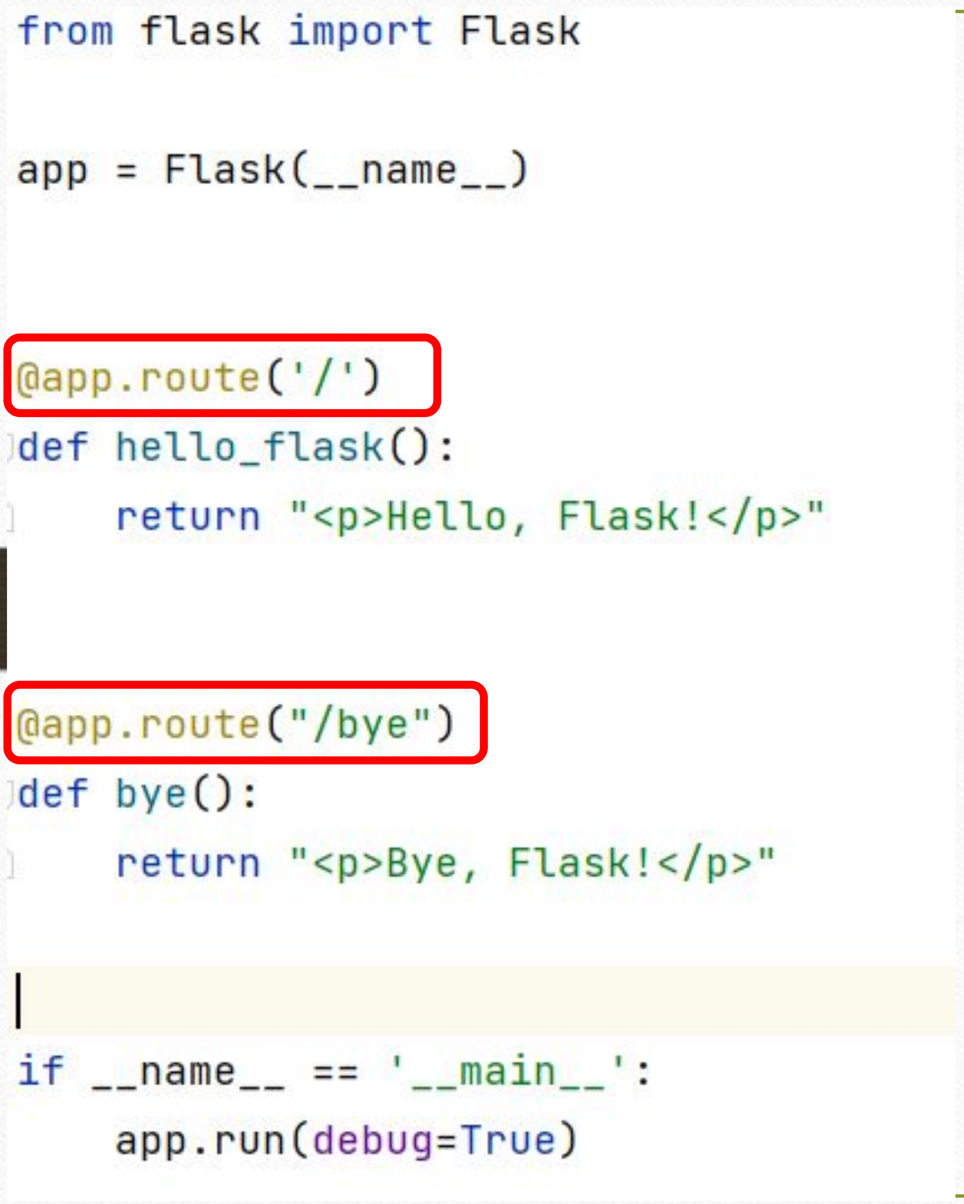
Home route



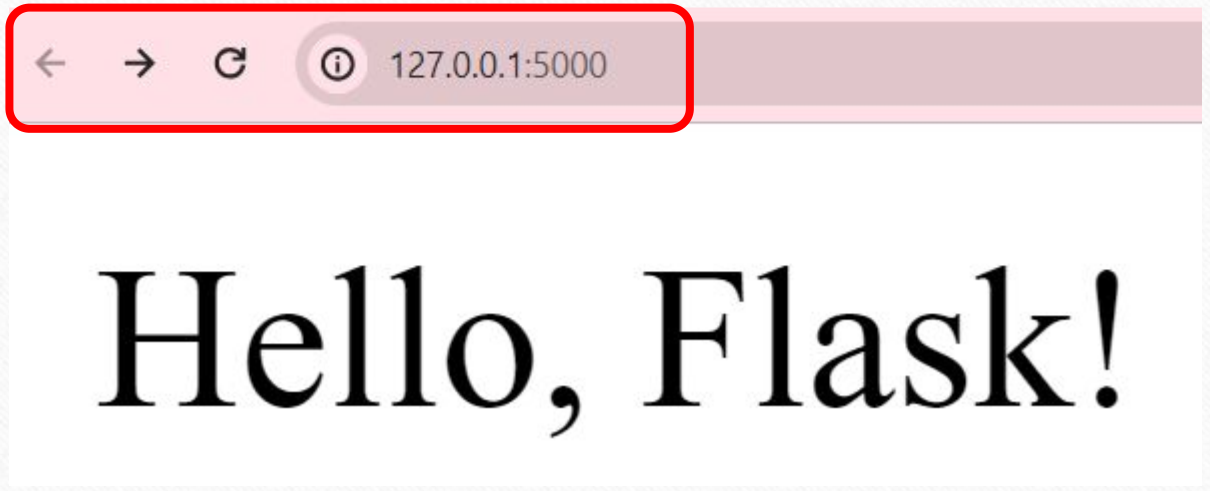
```
@app.route("/")
def hello_flask():
    return "<p>Hello, Flask!</p>"
```

Tells Flask what URL should trigger the following function.

- In Flask, a **URL path** refers to the part of the URL that determines a specific route within your Flask application.
- Each route is associated with a Python function, and when a user accesses a URL that matches a route, Flask executes the associated function and returns a response.

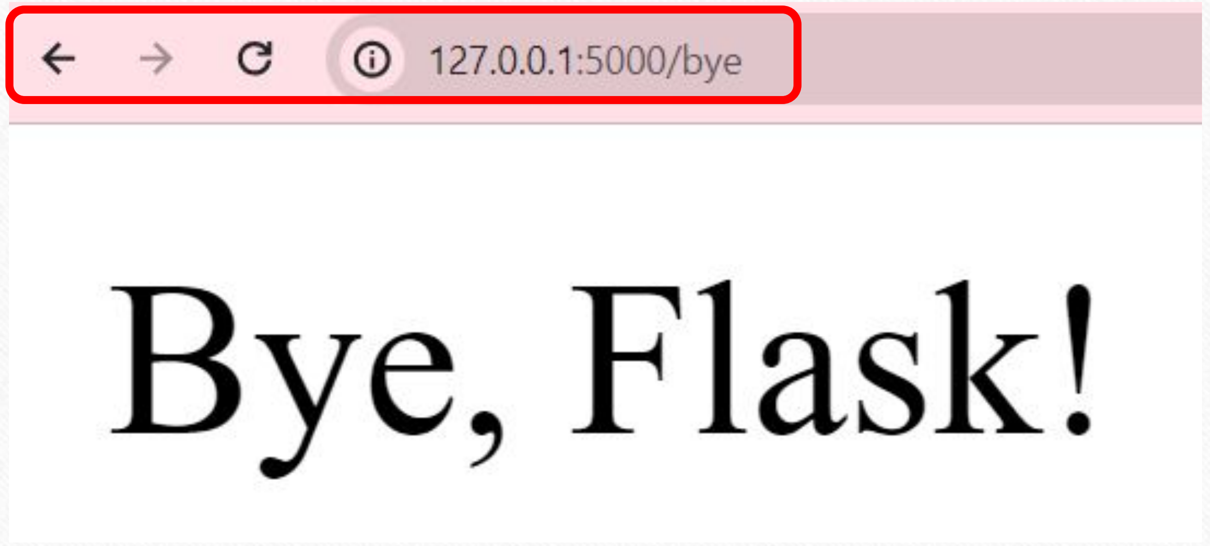


```
from flask import Flask

app = Flask(__name__)


@app.route('/')
def hello_flask():
    return "<p>Hello, Flask!</p>"


@app.route("/bye")
def bye():
    return "<p>Bye, Flask!</p>"


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



127.0.0.1:5000

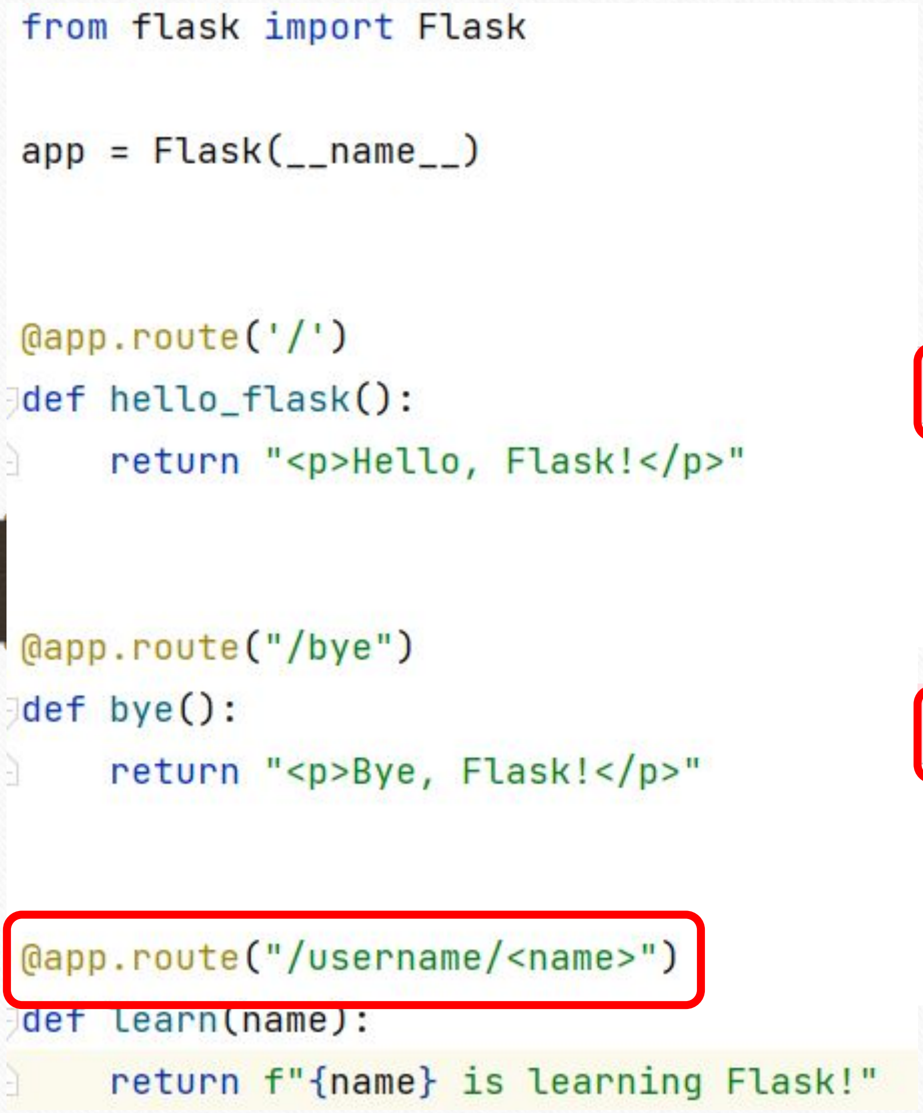
Hello, Flask!



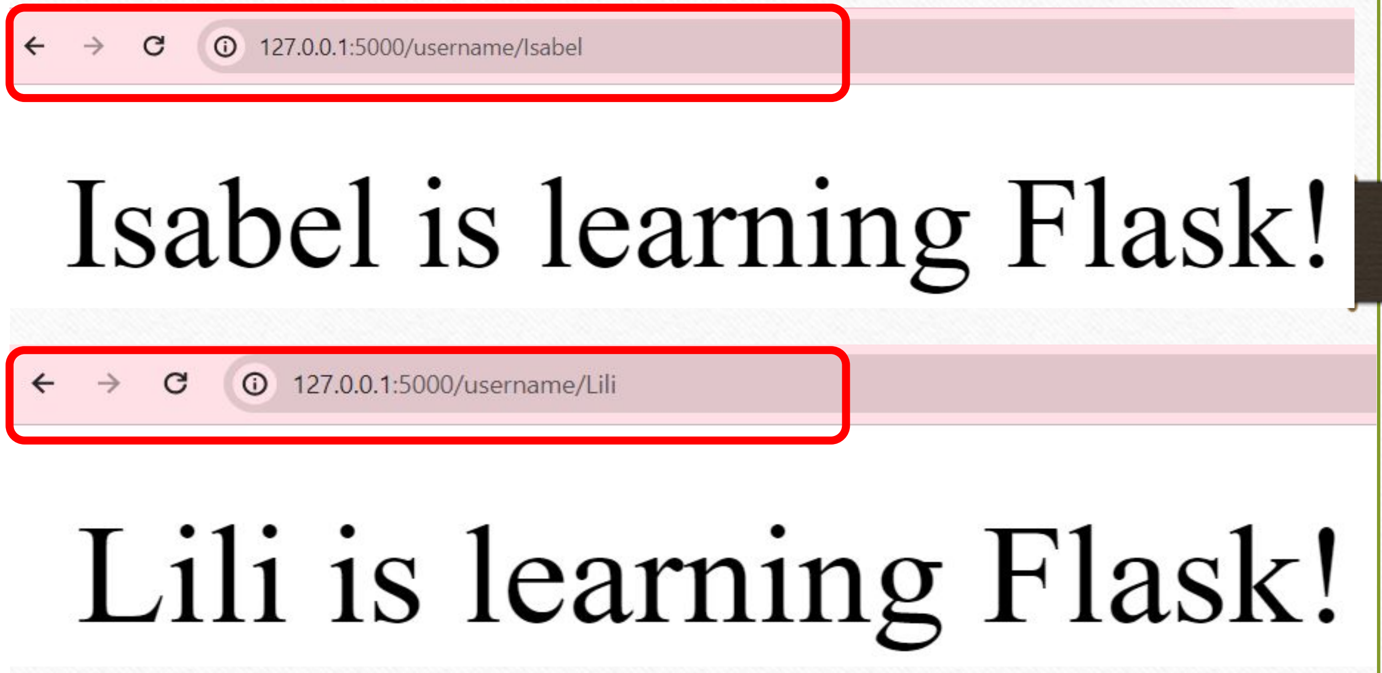
127.0.0.1:5000/bye

Bye, Flask!

Variable Path



```
from flask import Flask

app = Flask(__name__)


@app.route('/')
def hello_flask():
    return "<p>Hello, Flask!</p>"


@app.route("/bye")
def bye():
    return "<p>Bye, Flask!</p>"


@app.route("/username/<name>")
def learn(name):
    return f"{name} is learning Flask!"
```



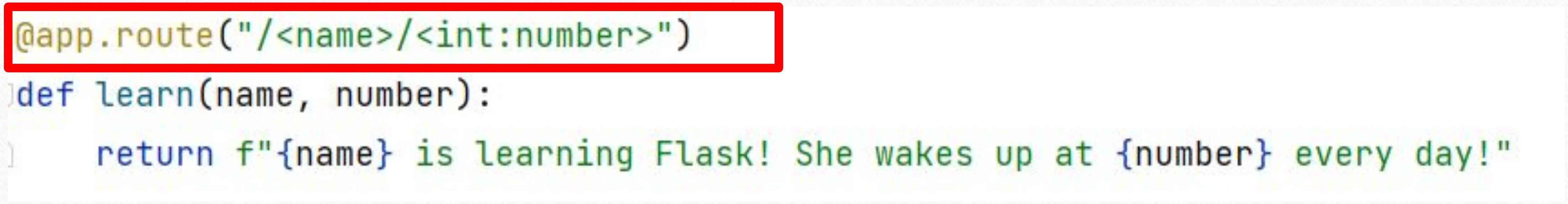
127.0.0.1:5000/username/Isabel

Isabel is learning Flask!

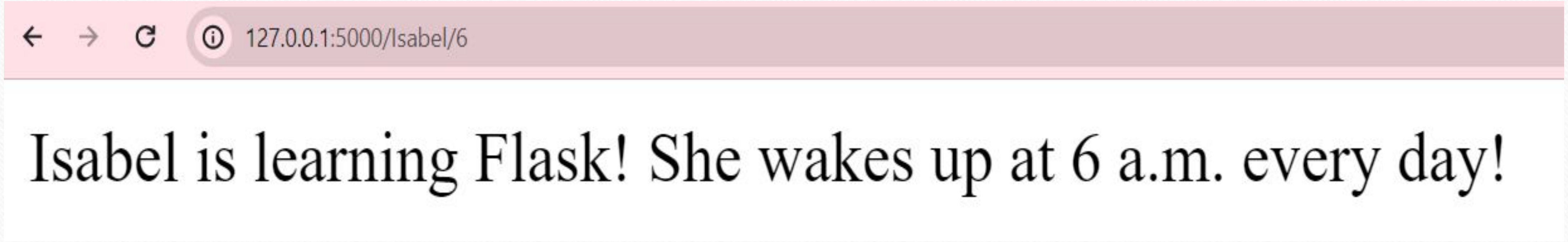
127.0.0.1:5000/username/Lili

Lili is learning Flask!

Variable Path (cont.)



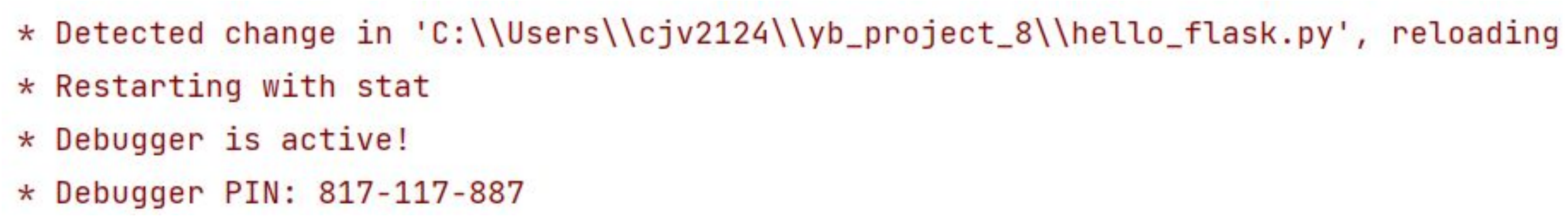
```
@app.route("/<name>/<int:number>")
def learn(name, number):
    return f"{name} is learning Flask! She wakes up at {number} every day!"
```



```
127.0.0.1:5000/Isabel/6
```

Isabel is learning Flask! She wakes up at 6 a.m. every day!

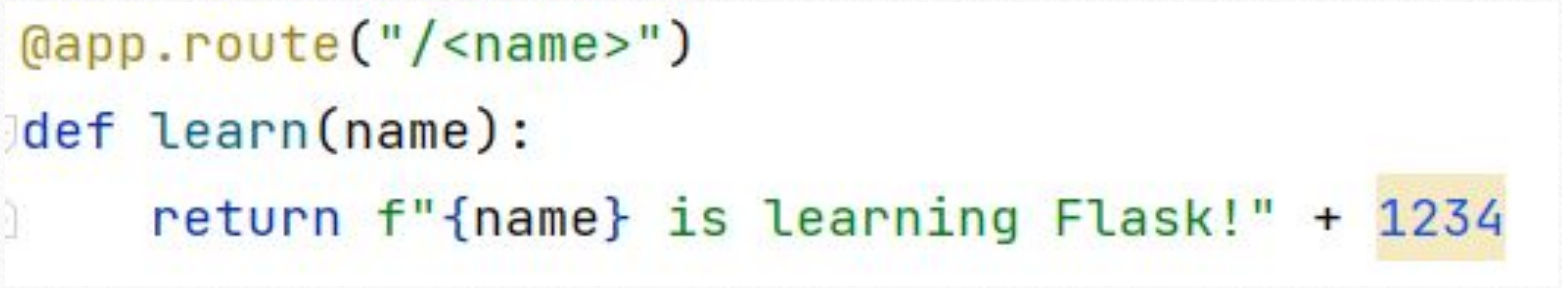
Different between app.run(debug=True) and app.run()



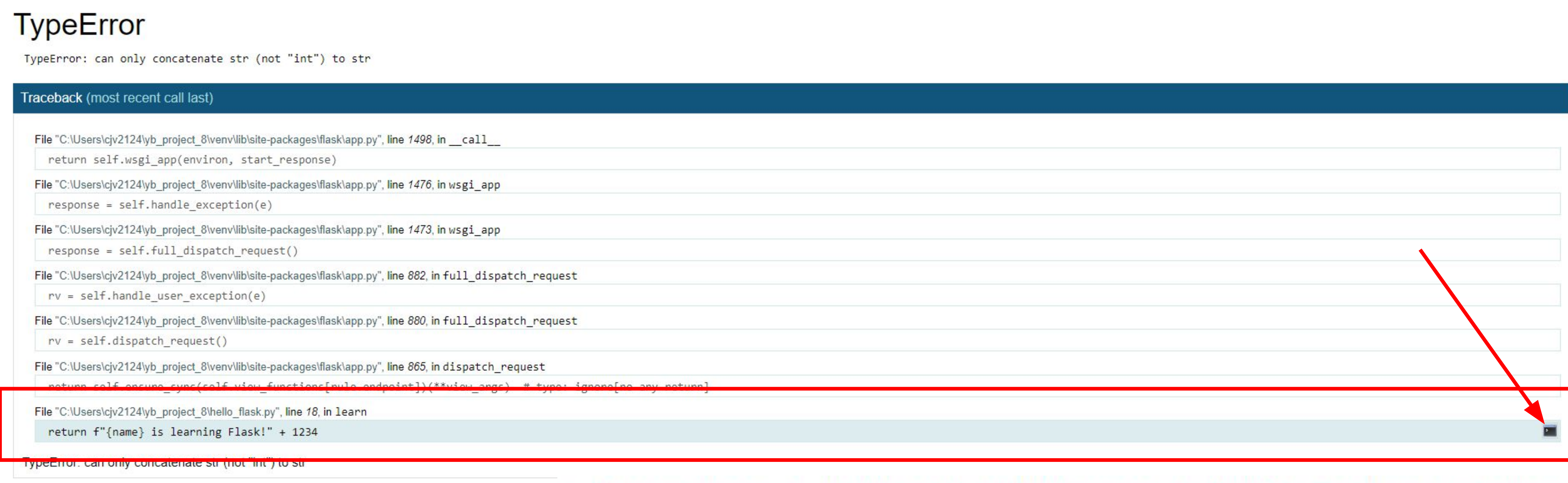
```
* Detected change in 'C:\\Users\\cjv2124\\yb_project_8\\hello_flask.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 817-117-887
```

Otherwise, stop your server and rerun

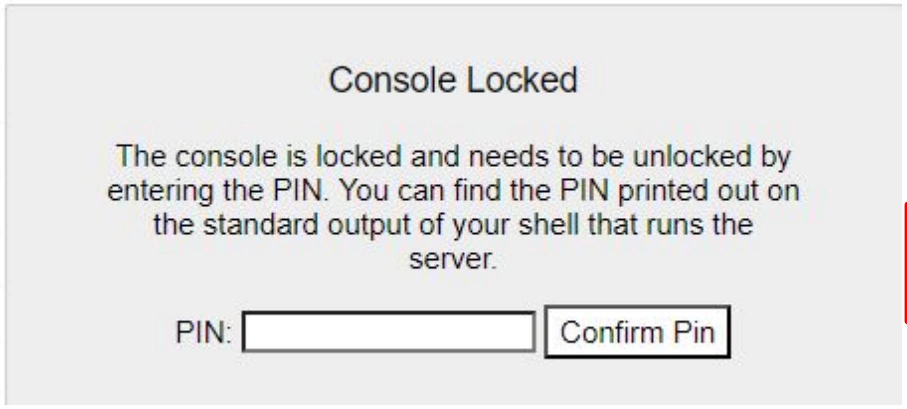
Debug-Flask debug view



```
@app.route("/<name>")
def learn(name):
    return f"{name} is learning Flask!" + 1234
```

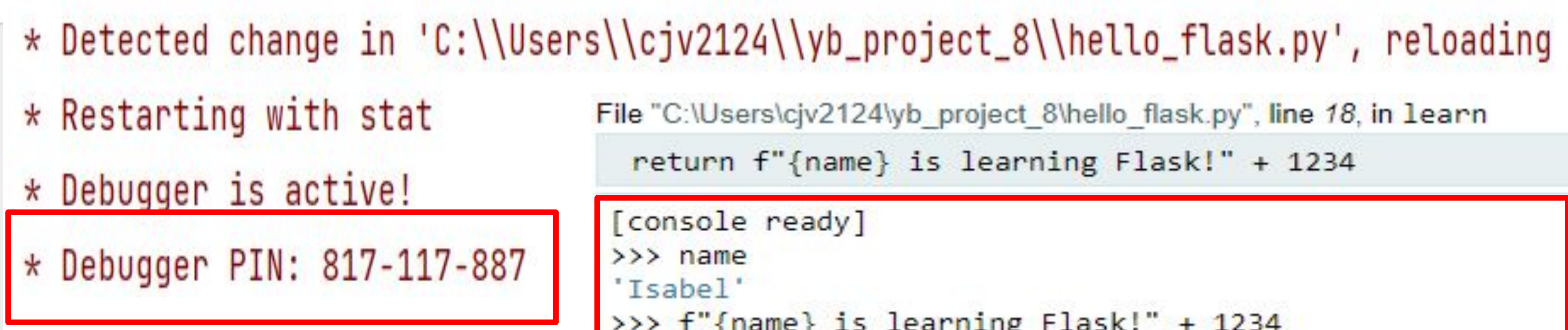


```
TypeError

TypeError: can only concatenate str (not "int") to str

Traceback (most recent call last)

File "C:\Users\cjv2124\yb_project_8\venv\lib\site-packages\flask\app.py", line 1498, in __call__
    return self.wsgi_app(environ, start_response)
File "C:\Users\cjv2124\yb_project_8\venv\lib\site-packages\flask\app.py", line 1476, in wsgi_app
    response = self.handle_exception(e)
File "C:\Users\cjv2124\yb_project_8\venv\lib\site-packages\flask\app.py", line 1473, in wsgi_app
    response = self.full_dispatch_request()
File "C:\Users\cjv2124\yb_project_8\venv\lib\site-packages\flask\app.py", line 882, in full_dispatch_request
    rv = self.handle_user_exception(e)
File "C:\Users\cjv2124\yb_project_8\venv\lib\site-packages\flask\app.py", line 880, in full_dispatch_request
    rv = self.dispatch_request()
File "C:\Users\cjv2124\yb_project_8\venv\lib\site-packages\flask\app.py", line 865, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)  # type: ignore[no-any-return]
File "C:\Users\cjv2124\yb_project_8\hello_flask.py", line 18, in learn
    return f"{name} is learning Flask!" + 1234

TypeError: can only concatenate str (not "int") to str
```

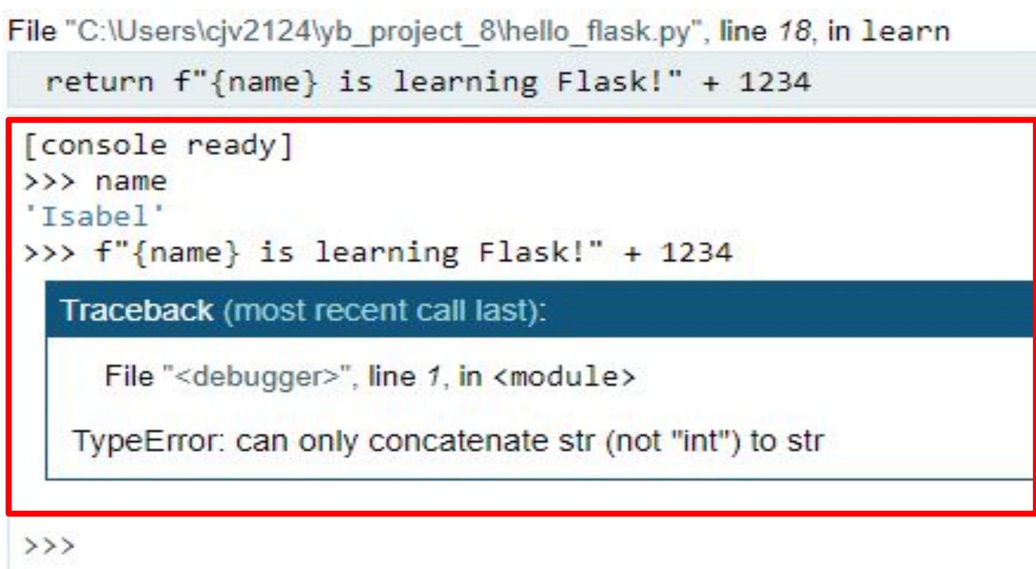


Console Locked

The console is locked and needs to be unlocked by entering the PIN. You can find the PIN printed out on the standard output of your shell that runs the server.



```
* Detected change in 'C:\\Users\\cjv2124\\yb_project_8\\hello_flask.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 817-117-887
```



```
File "C:\Users\cjv2124\yb_project_8\hello_flask.py", line 18, in learn
    return f"{name} is learning Flask!" + 1234

[console ready]
>>> name
'Isabel'
>>> f"{name} is learning Flask!" + 1234
Traceback (most recent call last):
    File "<debugger>", line 1, in <module>
  TypeError: can only concatenate str (not "int") to str
>>>
```

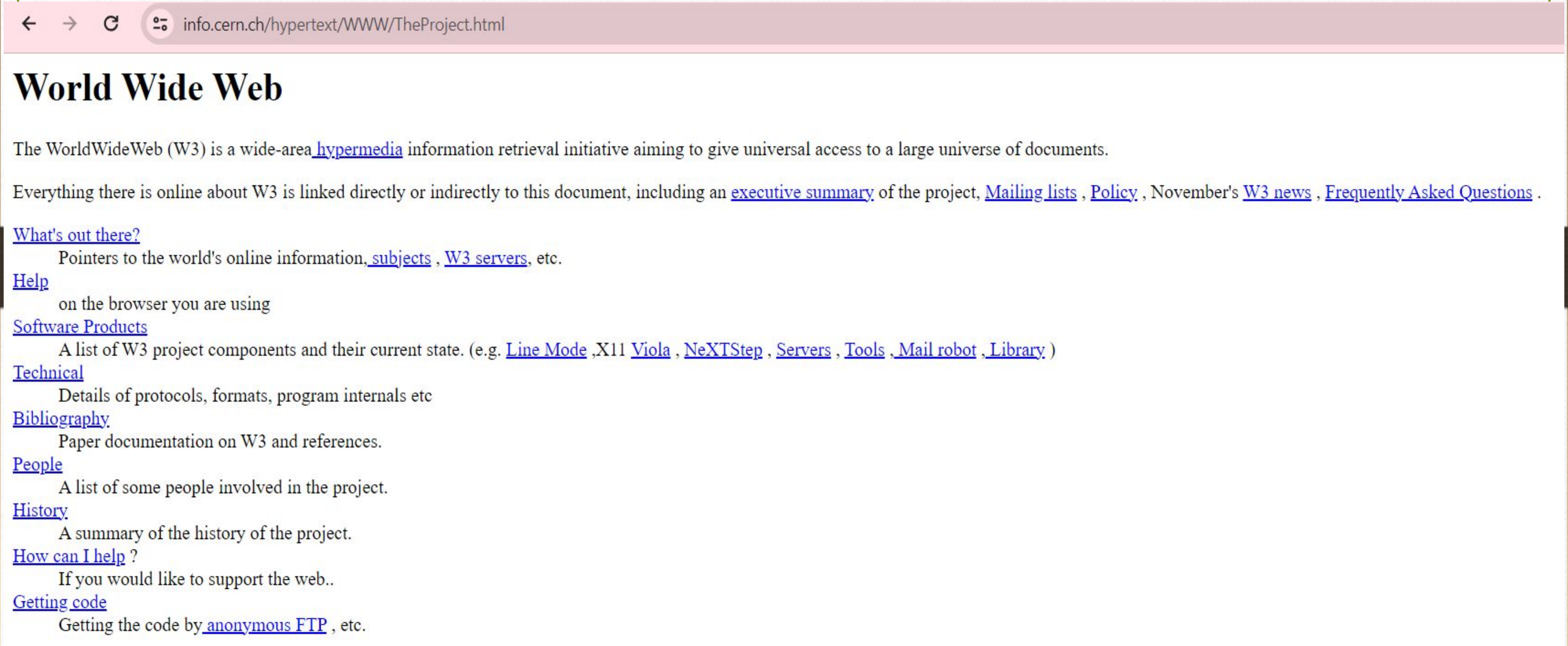
Rendering HTML Elements with Flask

HTML
CSS

What is HTML?

- HyperText Markup Language
- HTML defines the content and structure of the website.

World’s First Website : https://info.cern.ch/hypertext/WWW/TheProject.html


info.cern.ch/hypertext/WWW/TheProject.html

World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
: Pointers to the world's online information, subjects , W3 servers, etc.

Help
: on the browser you are using

Software Products
: A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)

Technical
: Details of protocols, formats, program internals etc

Bibliography
: Paper documentation on W3 and references.

People
: A list of some people involved in the project.

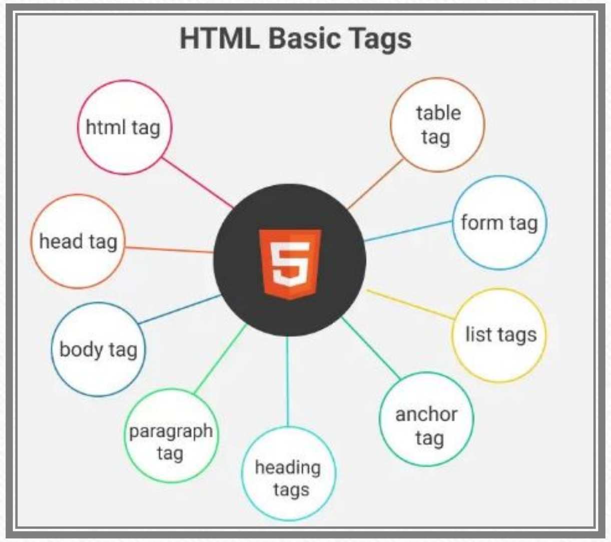
History
: A summary of the history of the project.

How can I help ?
: If you would like to support the web..

Getting code
: Getting the code by anonymous FTP , etc.

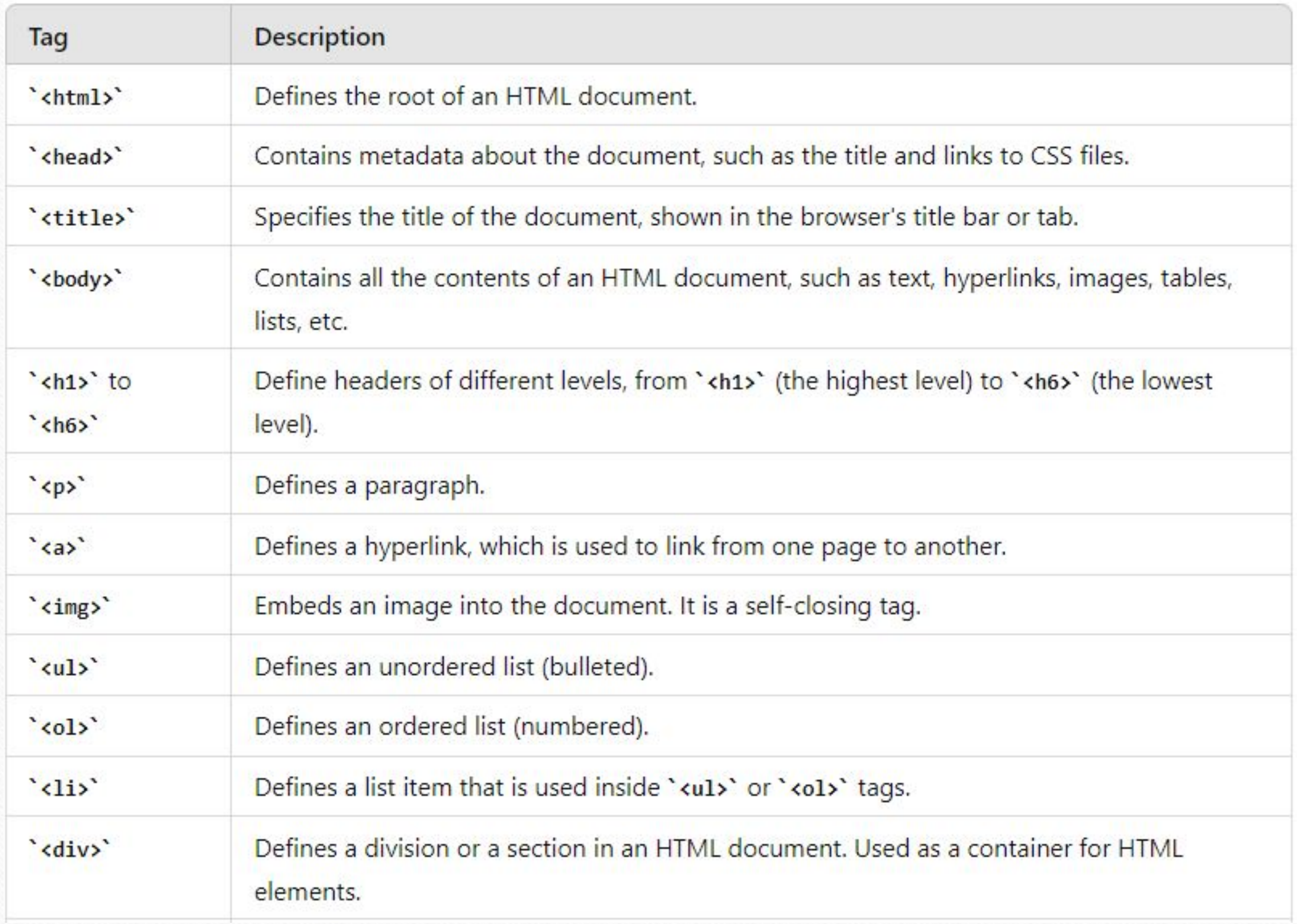
HTML Tags

- HTML tags are used to create elements within an HTML document.
- Most tags have an opening tag and a closing tag with content in between.
    - For example, <p>This is a paragraph.</p>, uses the <p> tag to denote a paragraph.
    - Some tags are self-closing and do not need a closing tag, such as the <img> tag for images.



Example of Basic HTML Tags

----more details: https://www.w3schools.com/tags/



Tag	Description
`<html>`	Defines the root of an HTML document.
`<head>`	Contains metadata about the document, such as the title and links to CSS files.
`<title>`	Specifies the title of the document, shown in the browser's title bar or tab.
`<body>`	Contains all the contents of an HTML document, such as text, hyperlinks, images, tables, lists, etc.
`<h1>` to `<h6>`	Define headers of different levels, from `<h1>` (the highest level) to `<h6>` (the lowest level).
`<p>`	Defines a paragraph.
`<a>`	Defines a hyperlink, which is used to link from one page to another.
`<img>`	Embeds an image into the document. It is a self-closing tag.
`<ul>`	Defines an unordered list (bulleted).
`<ol>`	Defines an ordered list (numbered).
`<li>`	Defines a list item that is used inside `<ul>` or `<ol>` tags.
`<div>`	Defines a division or a section in an HTML document. Used as a container for HTML elements.

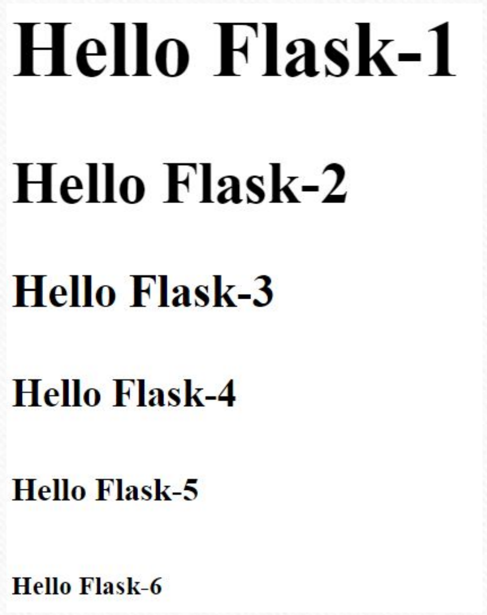
Usage of HTML

1. Heading Elements : <h1>Hello Flask</h1>

Element

Opening and closing tag

```html
<h1>Hello Flask-1</h1>
<h2>Hello Flask-2</h2>
<h3>Hello Flask-3</h3>
<h4>Hello Flask-4</h4>
<h5>Hello Flask-5</h5>
<h6>Hello Flask-6</h6>
```



Hello Flask-1

Hello Flask-2

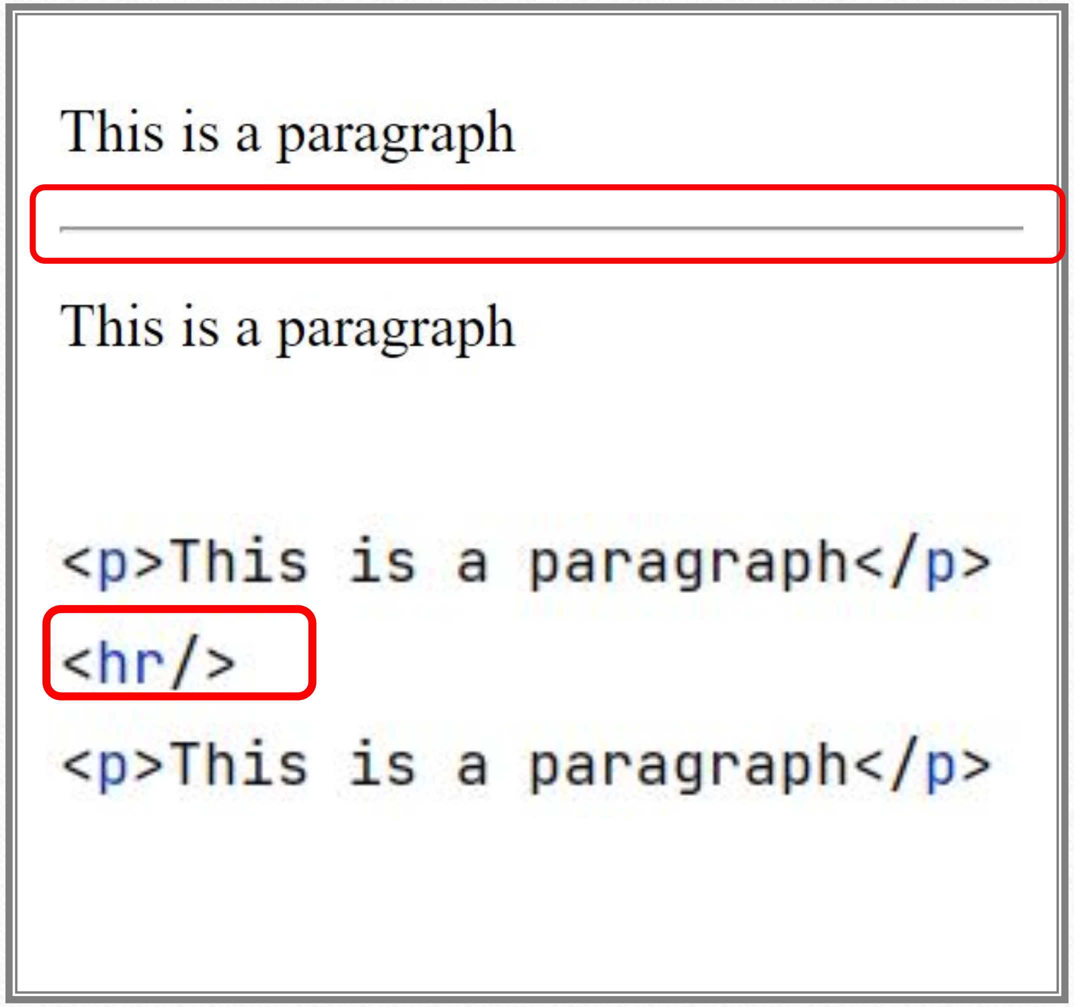
Hello Flask-3

Hello Flask-4

Hello Flask-5

Hello Flask-6

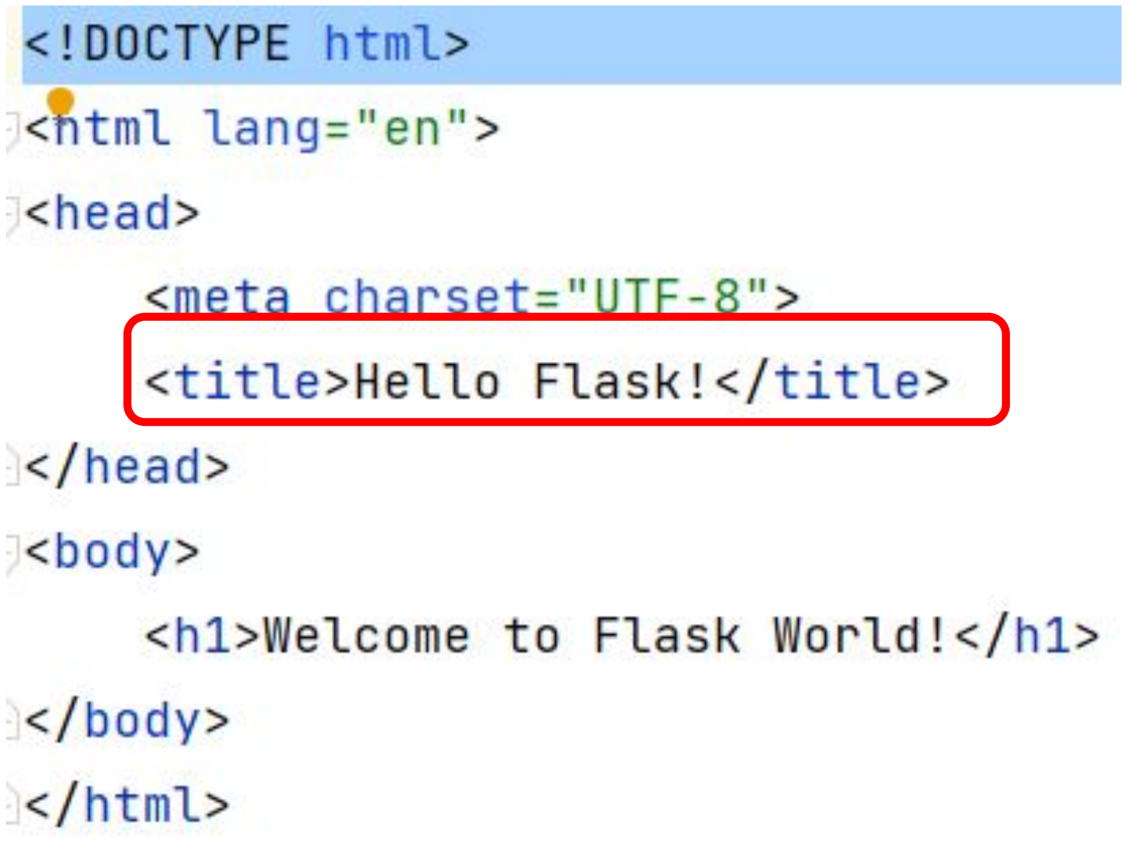
2. Self Closing Tags


This is a paragraph

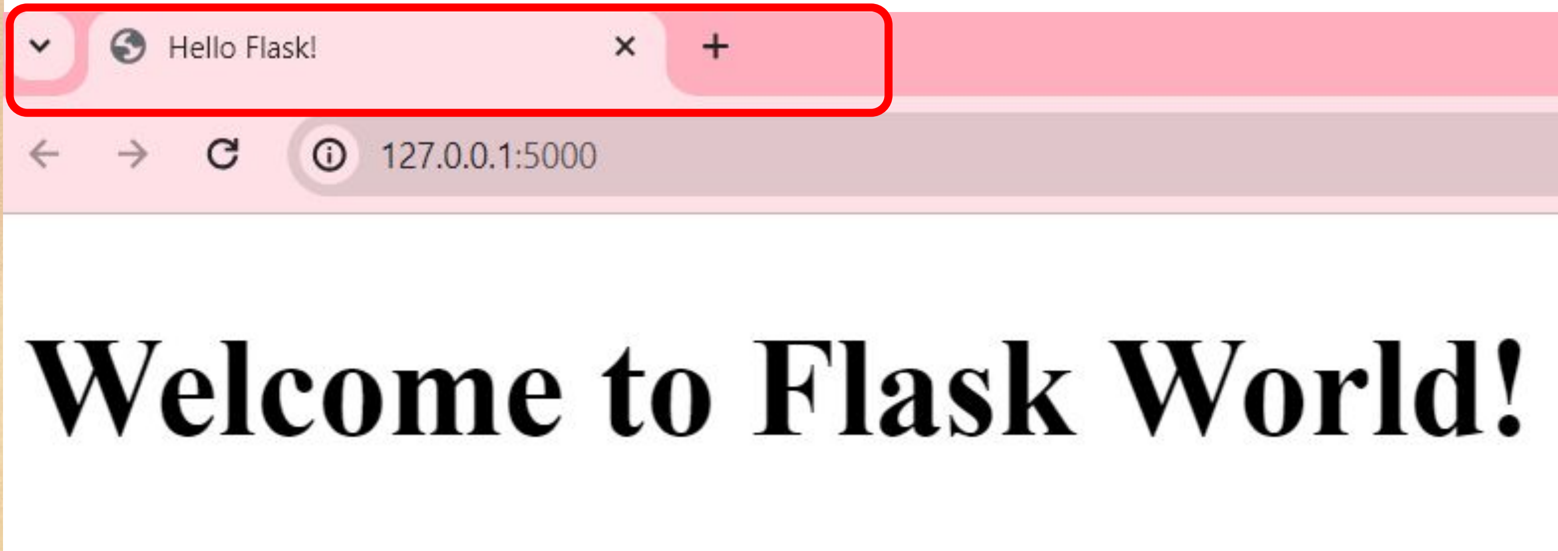
This is a paragraph

```
<p>This is a paragraph</p>
<hr/>
<p>This is a paragraph</p>
```

The HTML Boilerplate



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello Flask!</title>
</head>
<body>
    <h1>Welcome to Flask World!</h1>
</body>
</html>
```



Hello Flask!

127.0.0.1:5000

Welcome to Flask World!

What is CSS?

- It stands for Cascading Style Sheets.
- Is a stylesheet language used to describe the presentation of a document written in HTML.
- CSS defines how elements should be displayed on screen, on paper, or in other media

HTML?? CSS??

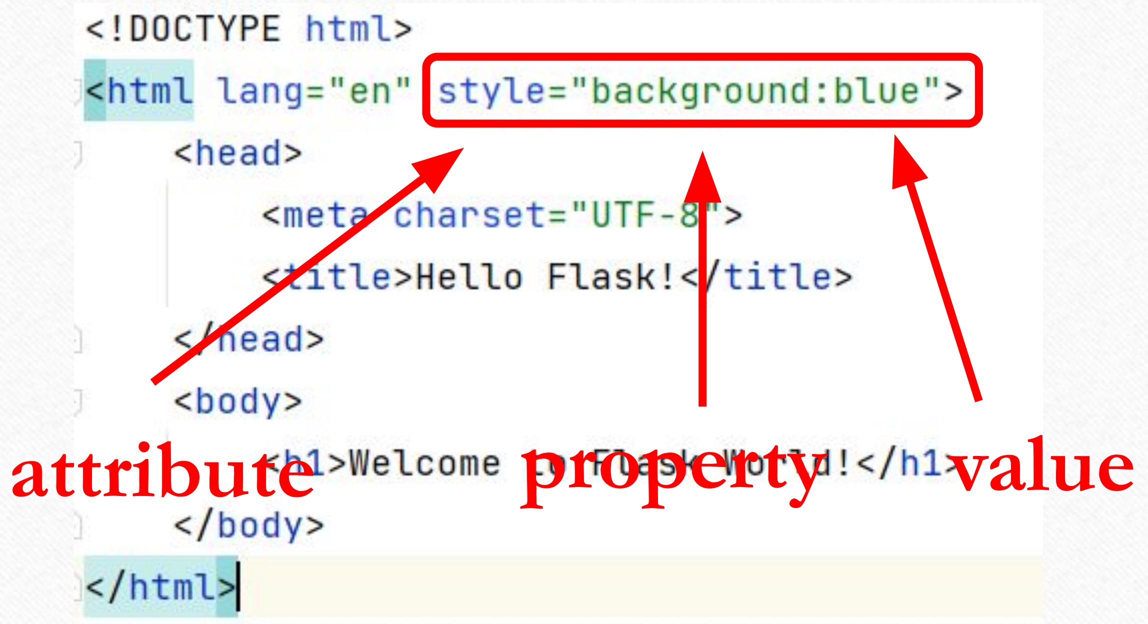
- HTML and CSS are both crucial technologies for creating web pages, but they have different purposes.
- HTML is used to structure and organize content on the web. It tells the browser what content to display (like text, images, videos, forms) and how to structure it (using elements like headings, paragraphs, lists).
- CSS is used to control the presentation of web pages, including layout, colors, fonts, and spacing. It enhances the appearance of HTML content and is essential for creating visually engaging and consistent designs across a website.

How to add CSS?

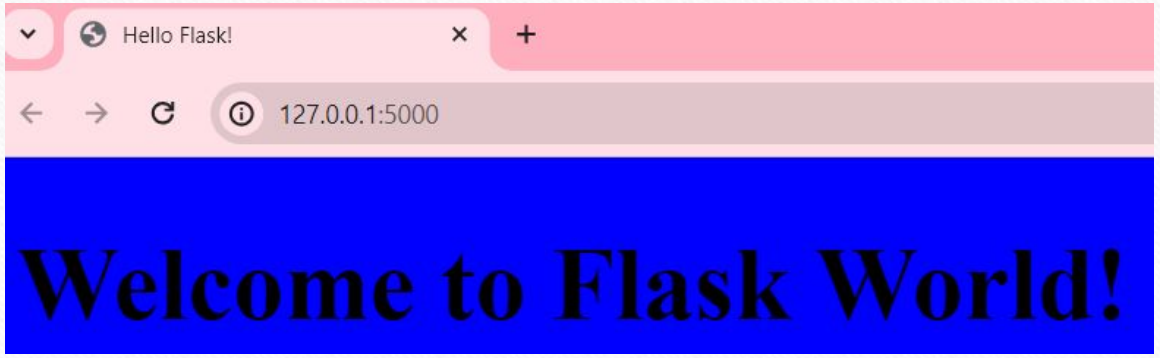
- Three ways to add:
  - Inline: <tag style="css"/>
  - Internal: <style>css</style>
  - External: <link href="style.css"/>

Inline Style to add CSS

The inline CSS goes into the opening tag



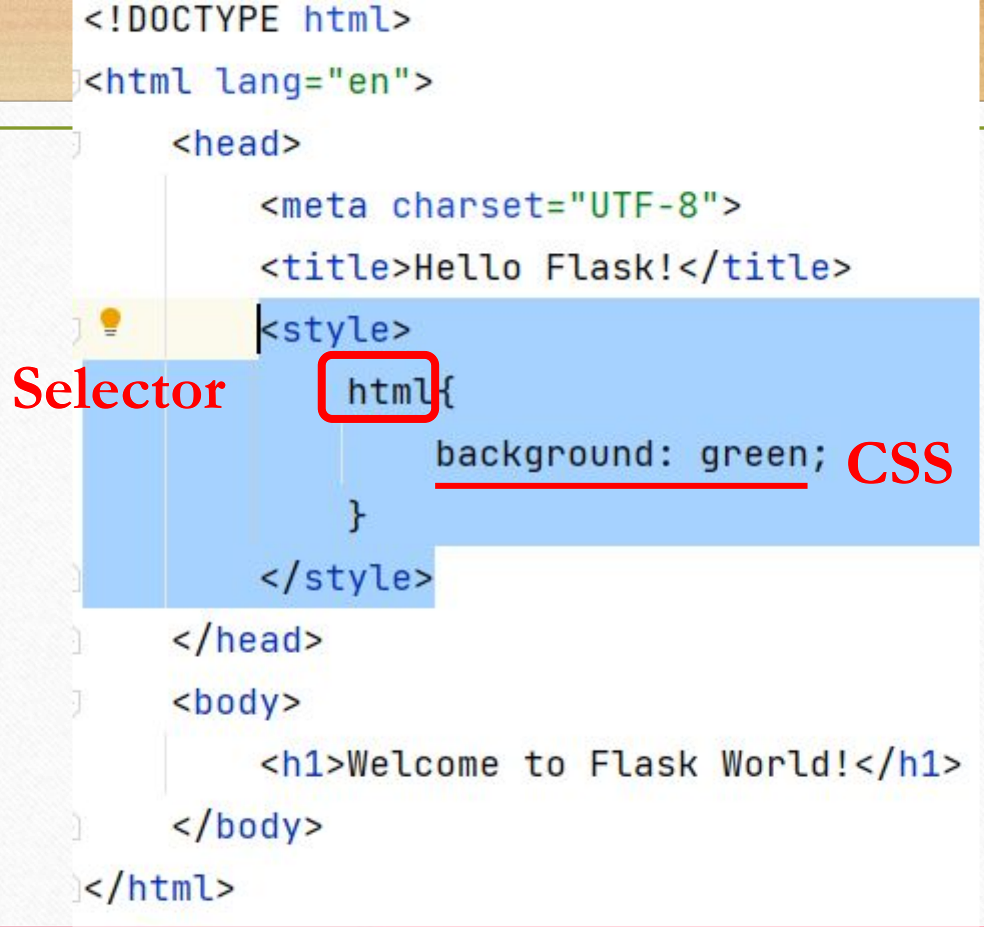
```
<!DOCTYPE html>
<html lang="en" style="background:blue">
    <head>
        <meta charset="UTF-8">
        <title>Hello Flask!</title>
    </head>
    <body>
        <h1>Welcome to Flask World!</h1>
    </body>
</html>
```



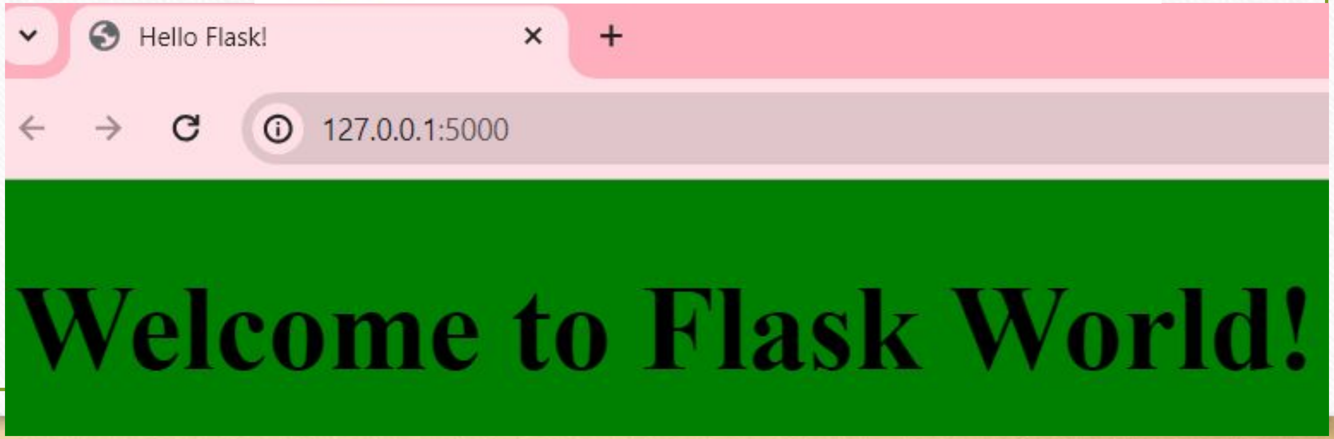
Welcome to Flask World!

Internal Style to add CSS

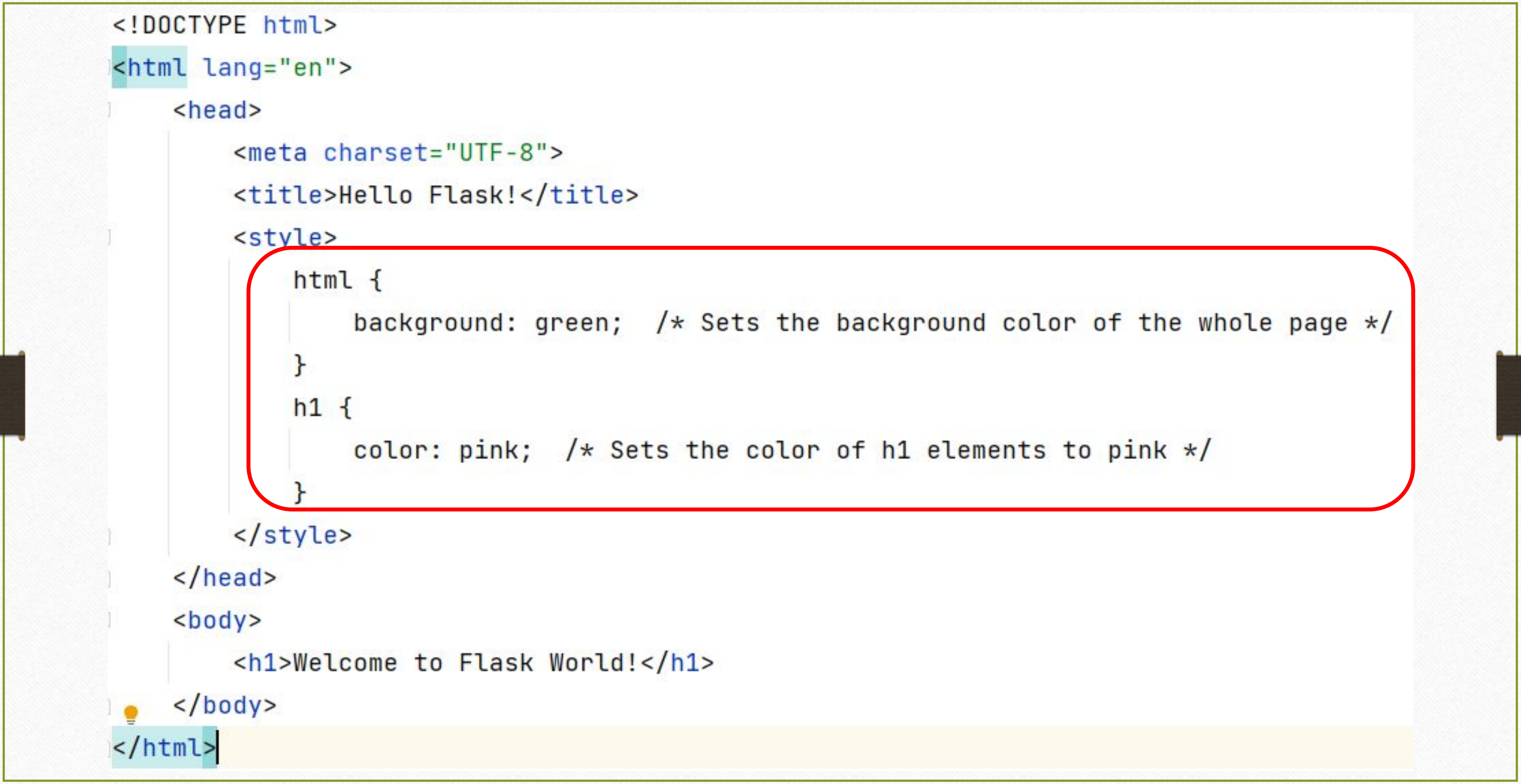
- This is done through a special HTML tag called the “style element.”
- The internal style can apply to anywhere within the same html document.



```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Hello Flask!</title>
        <style>
            html{
                background: green;
            }
        </style>
    </head>
    <body>
        <h1>Welcome to Flask World!</h1>
    </body>
</html>
```

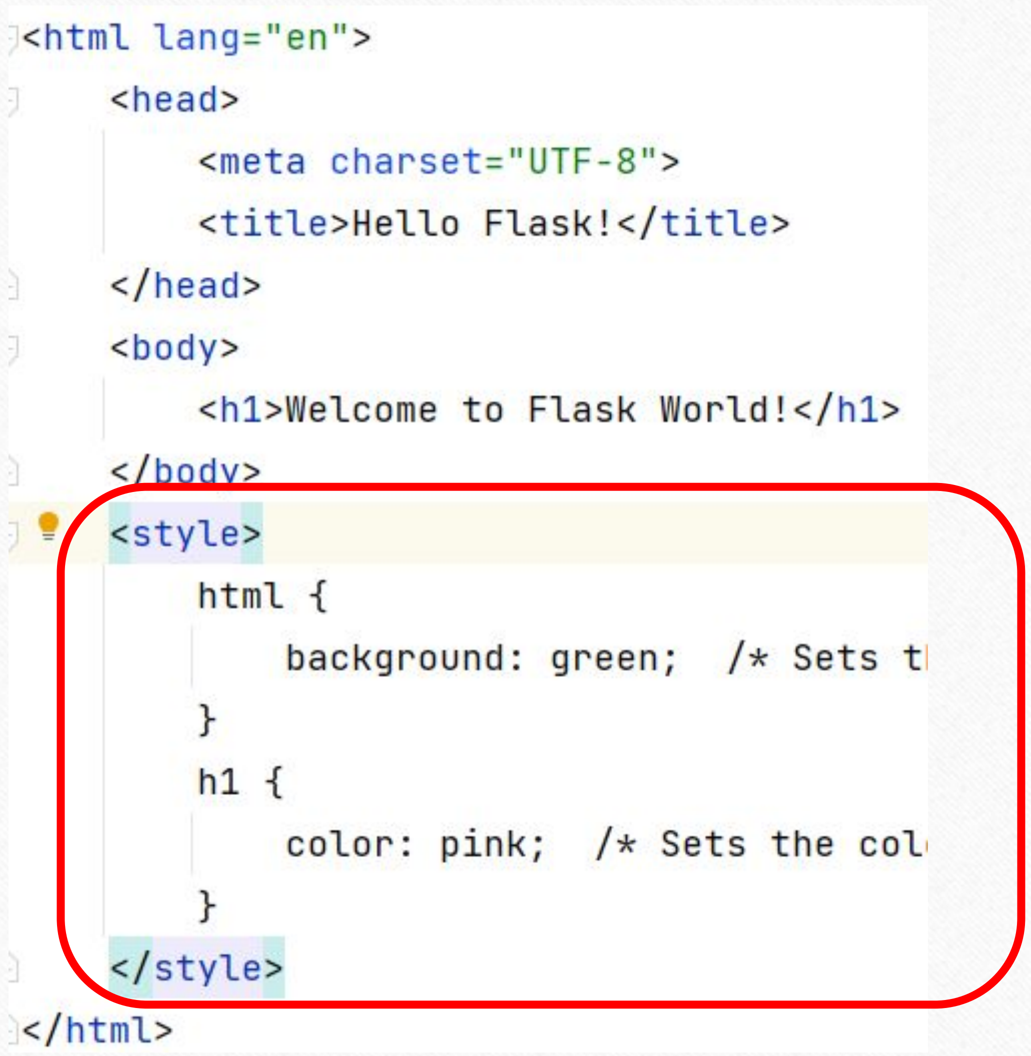


Welcome to Flask World!



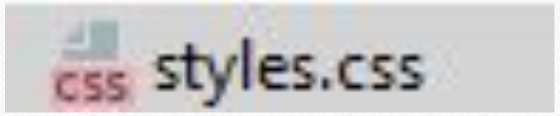
```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Hello Flask!</title>
        <style>
            html {
                background: green;  /* Sets the background color of the whole page */
            }
            h1 {
                color: pink;  /* Sets the color of h1 elements to pink */
            }
        </style>
    </head>
    <body>
        <h1>Welcome to Flask World!</h1>
    </body>
</html>
```

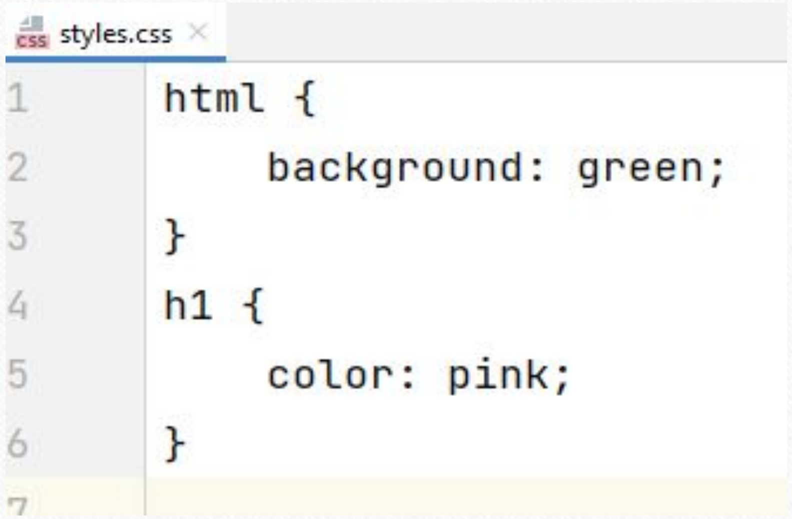
Clearer:



```
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Hello Flask!</title>
    </head>
    <body>
        <h1>Welcome to Flask World!</h1>
    </body>
    <style>
        html {
            background: green;  /* Sets t
        }
        h1 {
            color: pink;  /* Sets the col
        }
    </style>
</html>
```

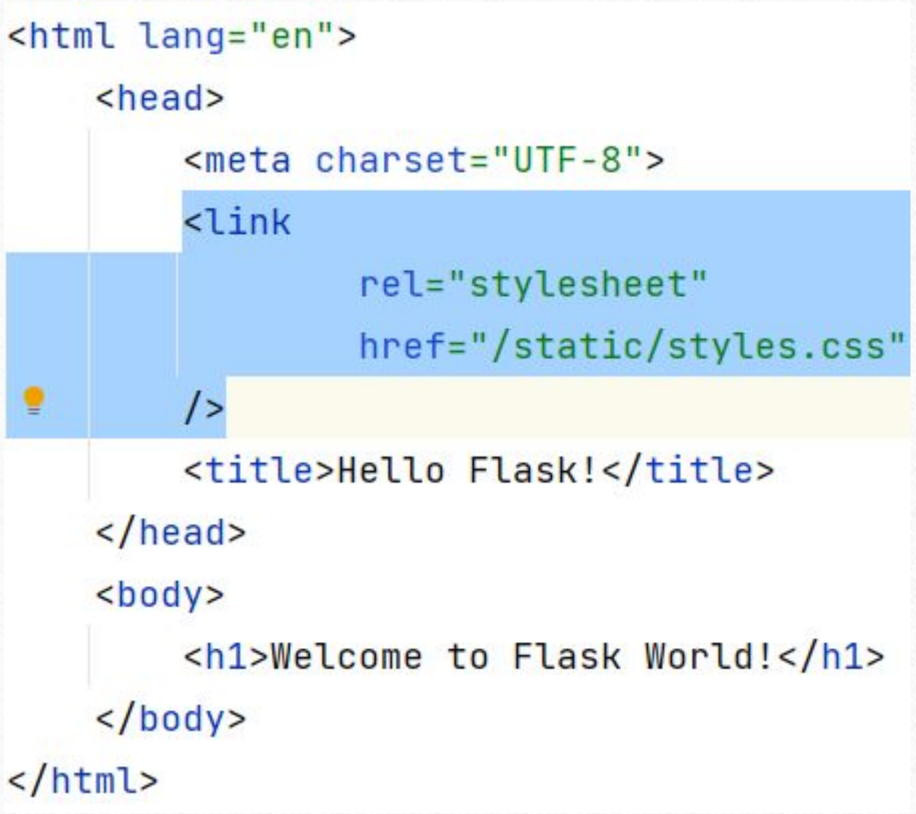
External Style to add CSS


styles.css


styles.css

```css
html {
    background: green;
}
h1 {
    color: pink;
}
```

Link up the style sheet file with .html file?



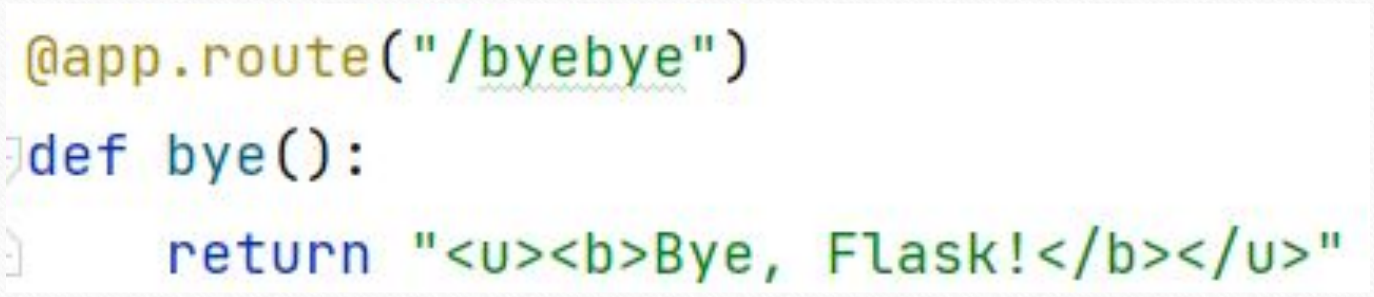
```html
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <link
                rel="stylesheet"
                href="/static/styles.css"
        />
        <title>Hello Flask!</title>
    </head>
    <body>
        <h1>Welcome to Flask World!</h1>
    </body>
</html>
```

Summarization

Inline	Internal	External
Only want to target a single element.	Only want to target a single web page.	Only want to target a multi-page website.

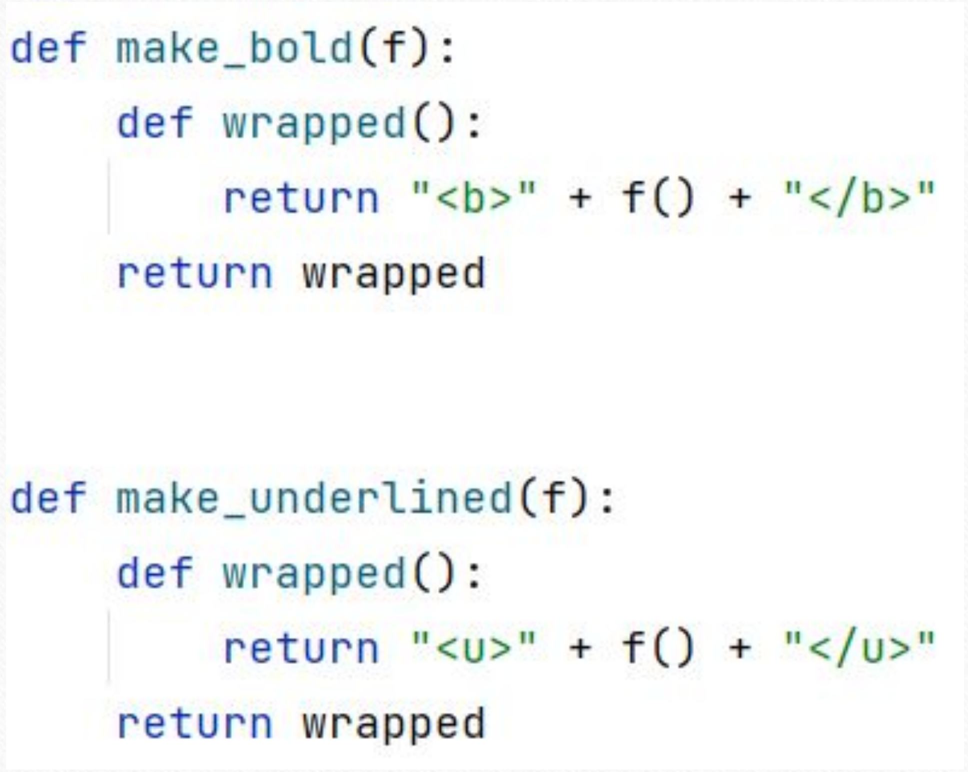
Rendering HTML Elements with Flask

1.

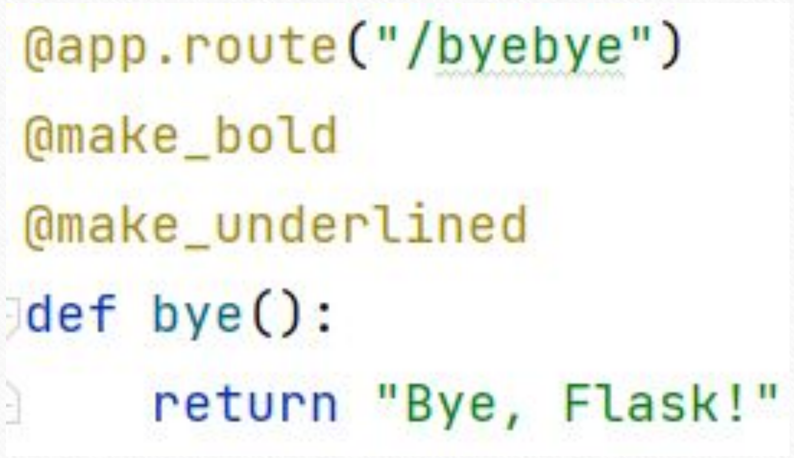


```python
@app.route("/byebye")
def bye():
    return "<u><b>Bye, Flask!</b></u>"
```

2.



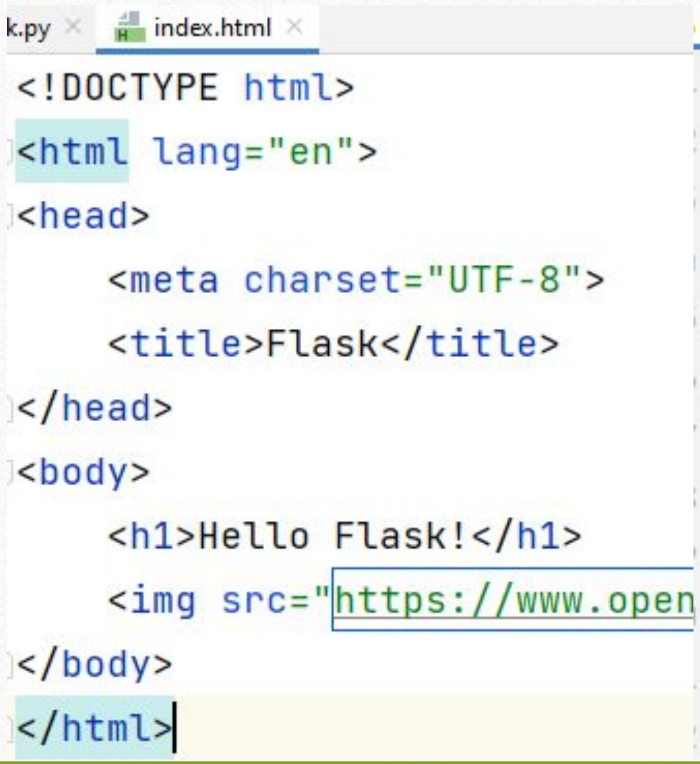
```python
def make_bold(f):
    def wrapped():
        return "<b>" + f() + "</b>"
    return wrapped


def make_underlined(f):
    def wrapped():
        return "<u>" + f() + "</u>"
    return wrapped
```



```python
@app.route("/byebye")
@make_bold
@make_underlined
def bye():
    return "Bye, Flask!"
```

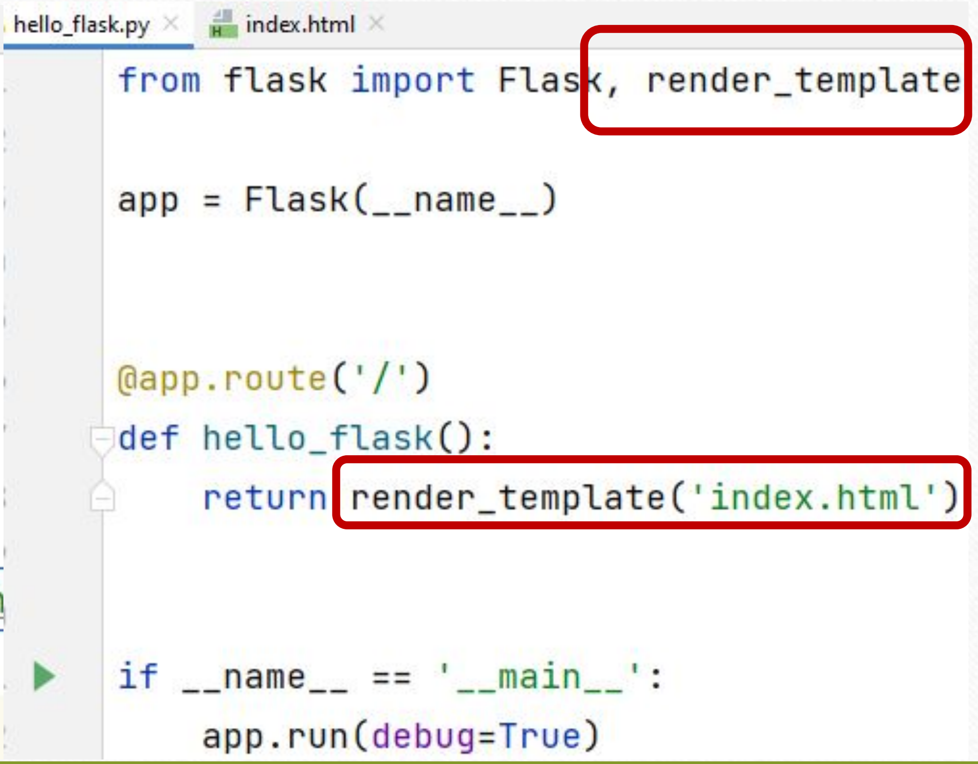
3. Render a template, how??

1. Create a folder called 'templates' and create a HTML file inside this folder.
2. Complete the HTML code inside this HTML file.
3. Import render_template from flask module
4. Return the render_template method and pass the HTML file name.



k.py × index.html ×

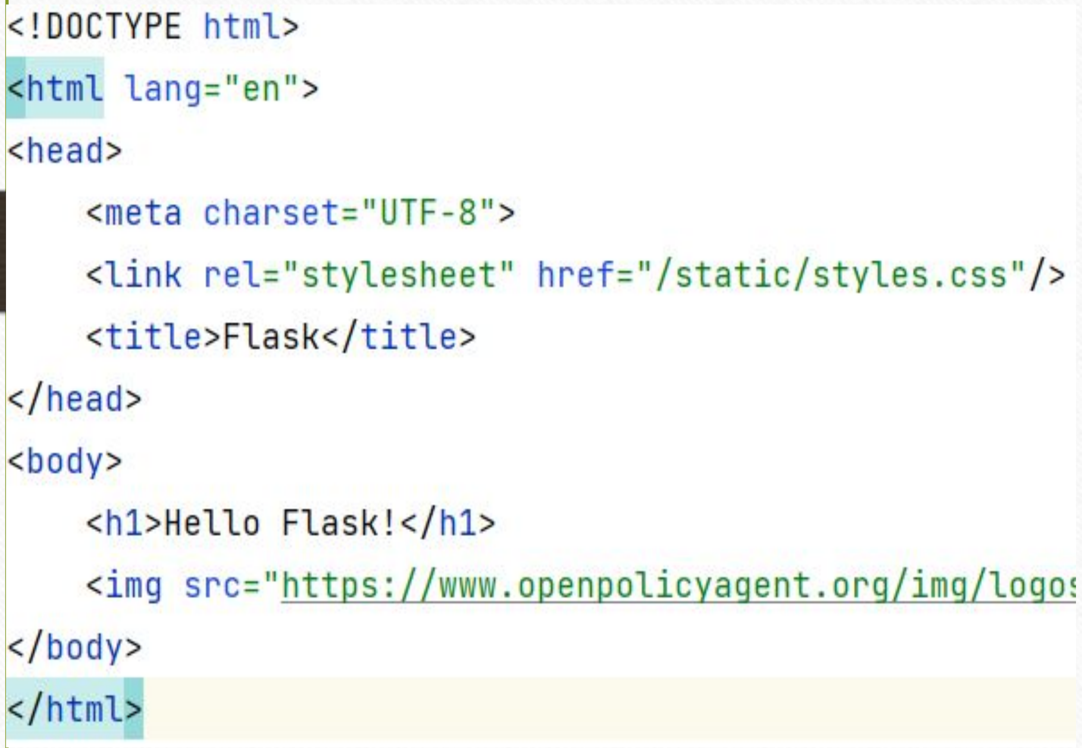
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Flask</title>
</head>
<body>
    <h1>Hello Flask!</h1>
    <img src="https://www.open
</body>
</html>
```



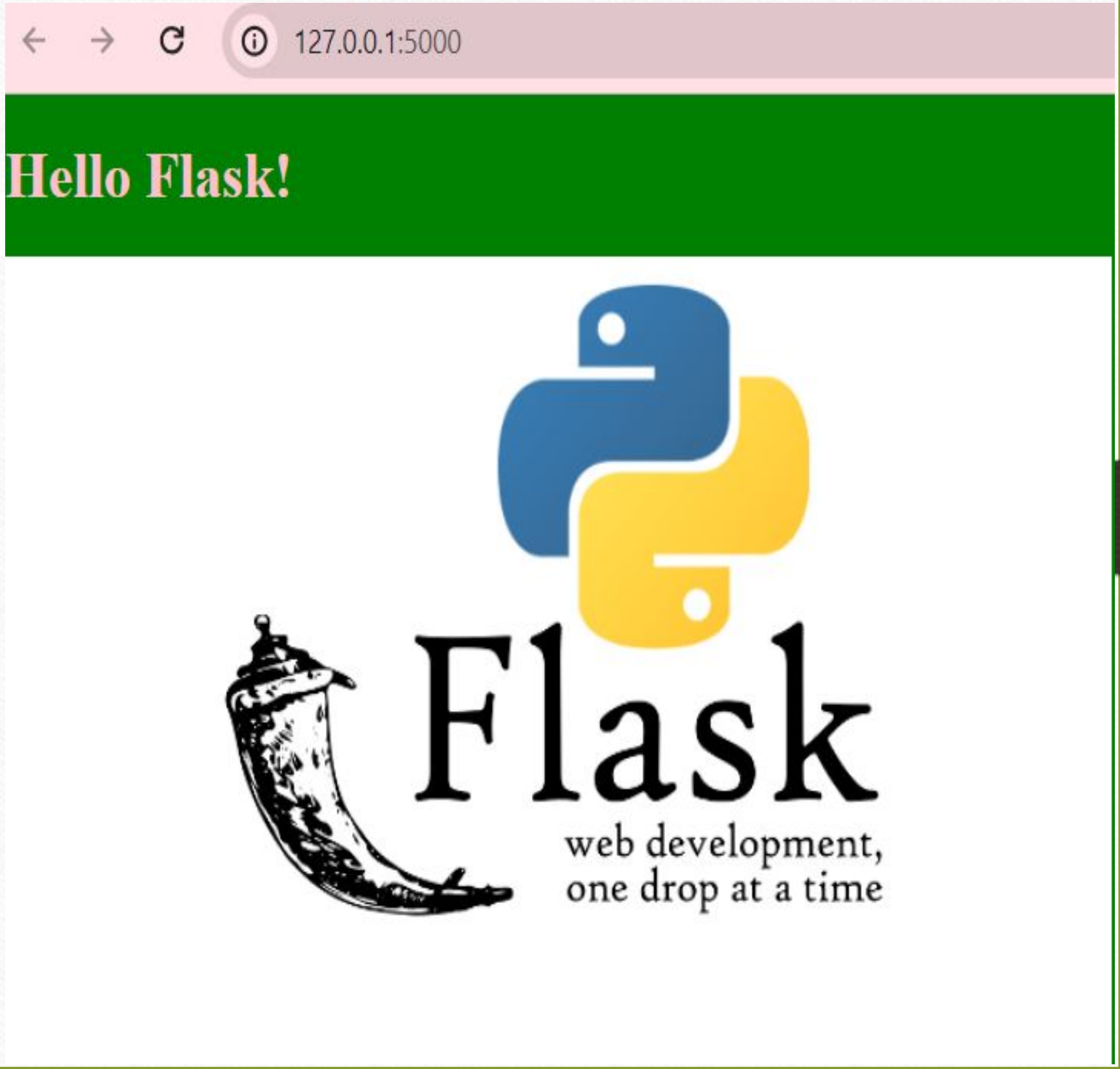
hello_flask.py × index.html ×

```python
from flask import Flask, render_template

app = Flask(__name__)


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


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

Output:



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="/static/styles.css"/>
    <title>Flask</title>
</head>
<body>
    <h1>Hello Flask!</h1>
    <img src="https://www.openpolicyagent.org/img/logos
</body>
</html>
```



127.0.0.1:5000

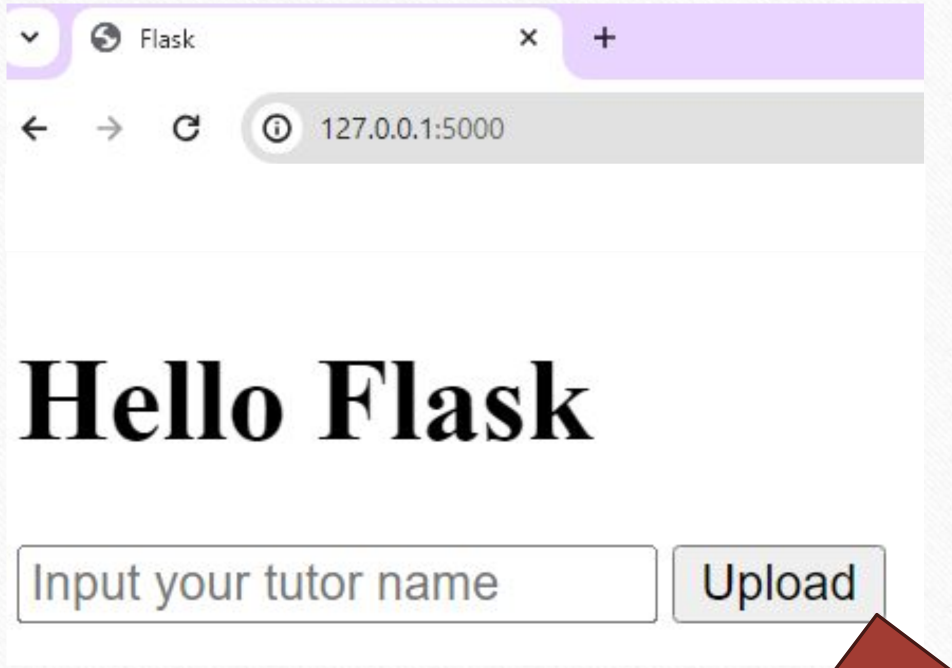
Hello Flask!

POST Requests with Flask Servers

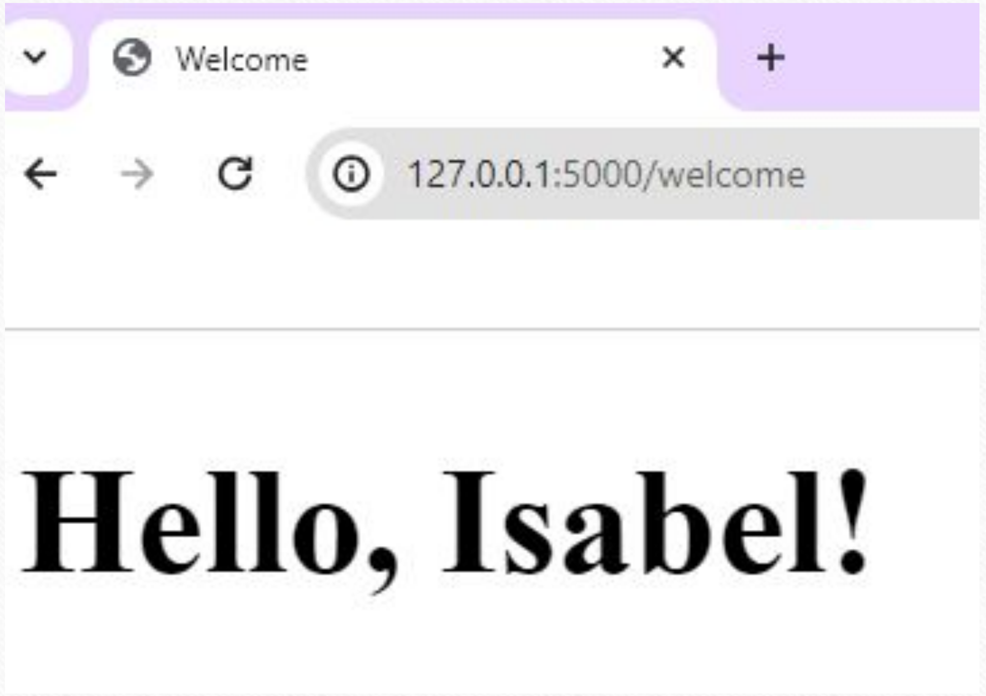
What is post request?

- In web development, a **POST request** is one of the common types of HTTP methods used to send data to a server, typically when submitting form data or uploading a file.

- POST requests include the data in the body of the request.

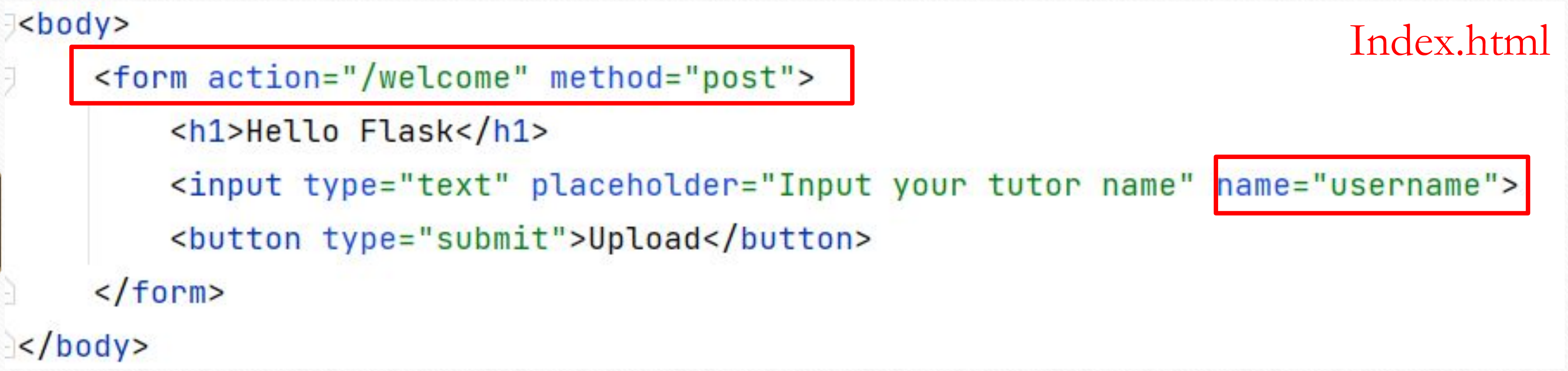


Hello Flask



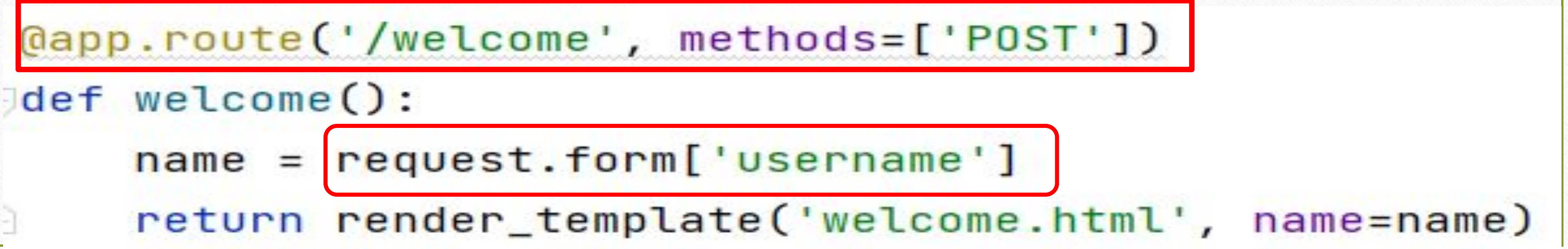
Hello, Isabel!

Need Flask server to be able to receive the data entered by the user. ----Need action and method

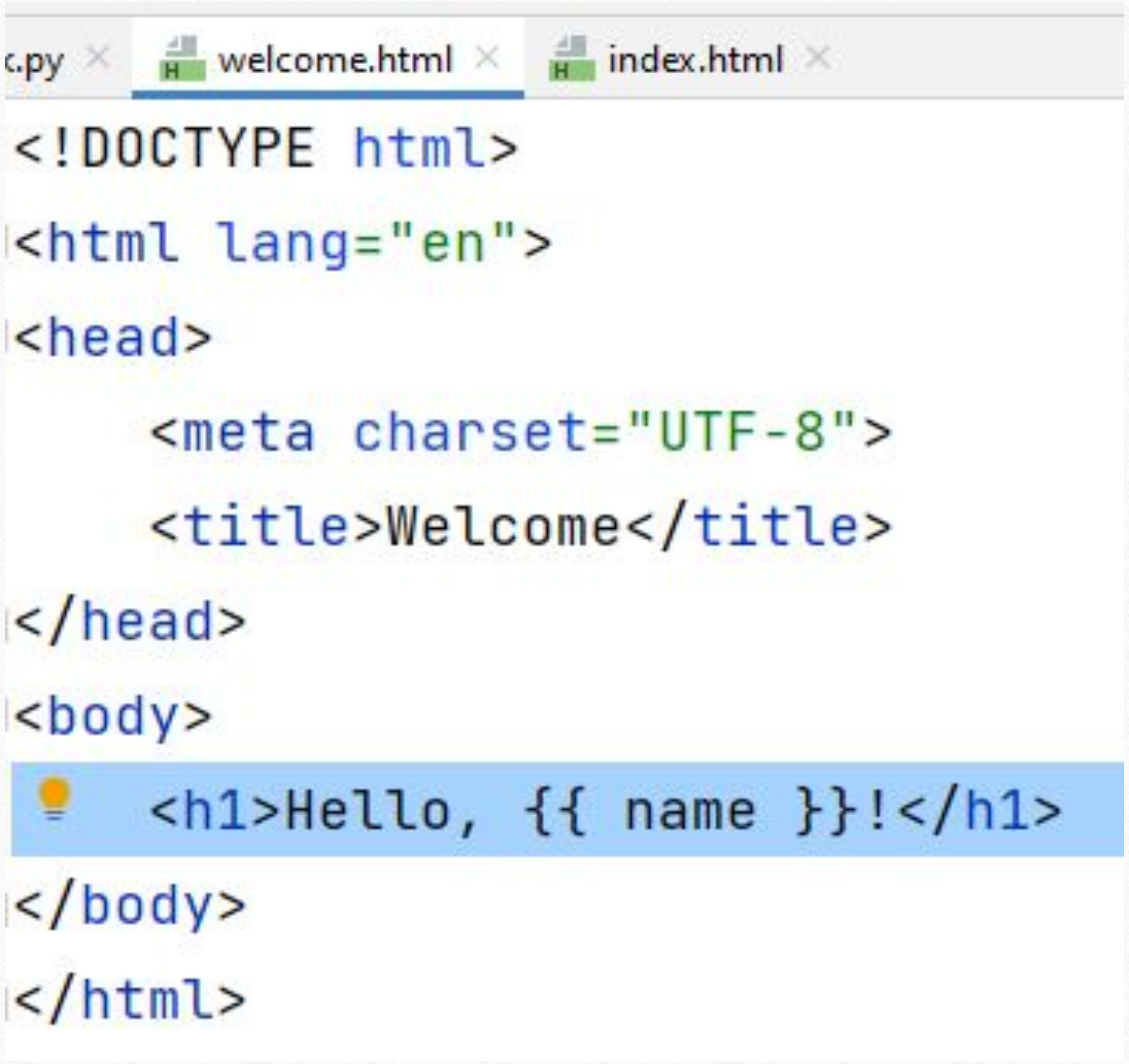


Index.html

```
<body>
    <form action="/welcome" method="post">
        <h1>Hello Flask</h1>
        <input type="text" placeholder="Input your tutor name" name="username">
        <button type="submit">Upload</button>
    </form>
</body>
```



```
@app.route('/welcome', methods=['POST'])
def welcome():
    name = request.form['username']
    return render_template('welcome.html', name=name)
```



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Welcome</title>
</head>
<body>
    <h1>Hello, {{ name }}!</h1>
</body>
</html>
```

Exercise in Class

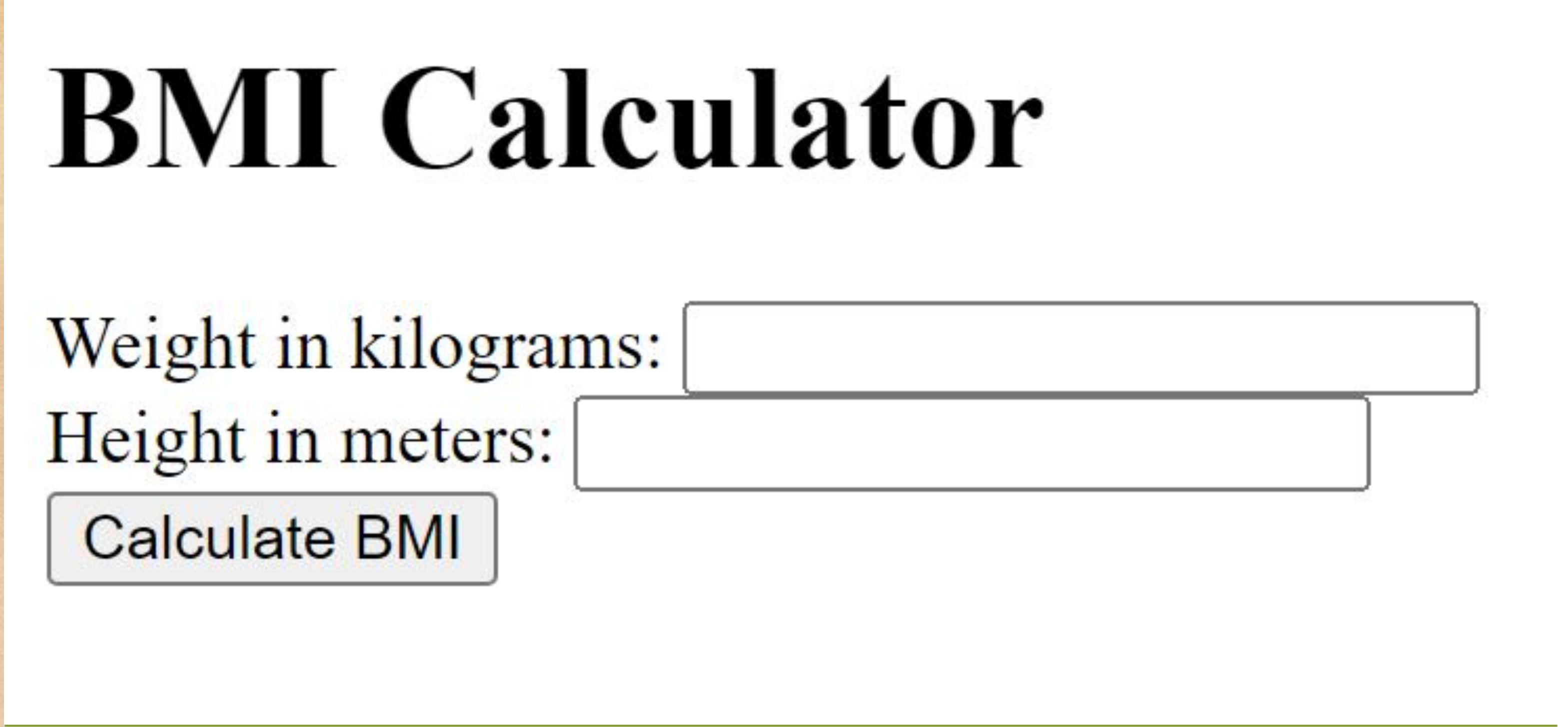
Building a BMI (Body Mass Index) Calculator using Flask

Requirements:

- Build a BMI (Body Mass Index) calculator that computes the BMI score based on a person's weight and height.
- Use conditional statements to interpret the BMI score into categories such as Underweight, Normal weight, Overweight, and Obese.
- Set the BMI classification thresholds as follows:

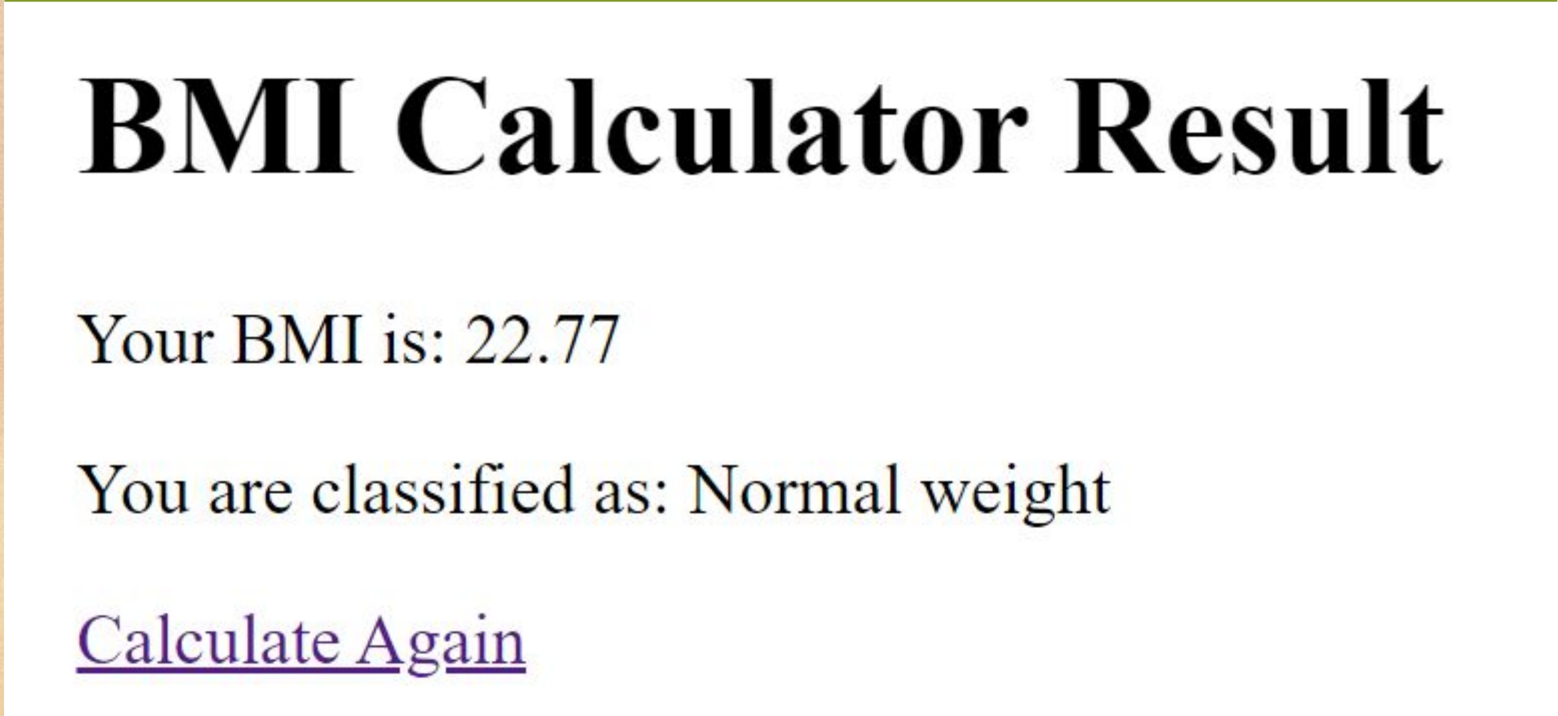
  - Underweight: less than 18.5
  - Normal weight: 18.5 to 24.9
  - Overweight: 25 to 29.9
  - Obese: 30 or more
- Print out the person's BMI score and interpretation.

Layout


BMI Calculator

Weight in kilograms:

Height in meters:


BMI Calculator Result

Your BMI is: 22.77

You are classified as: Normal weight

Calculate Again

Thank you