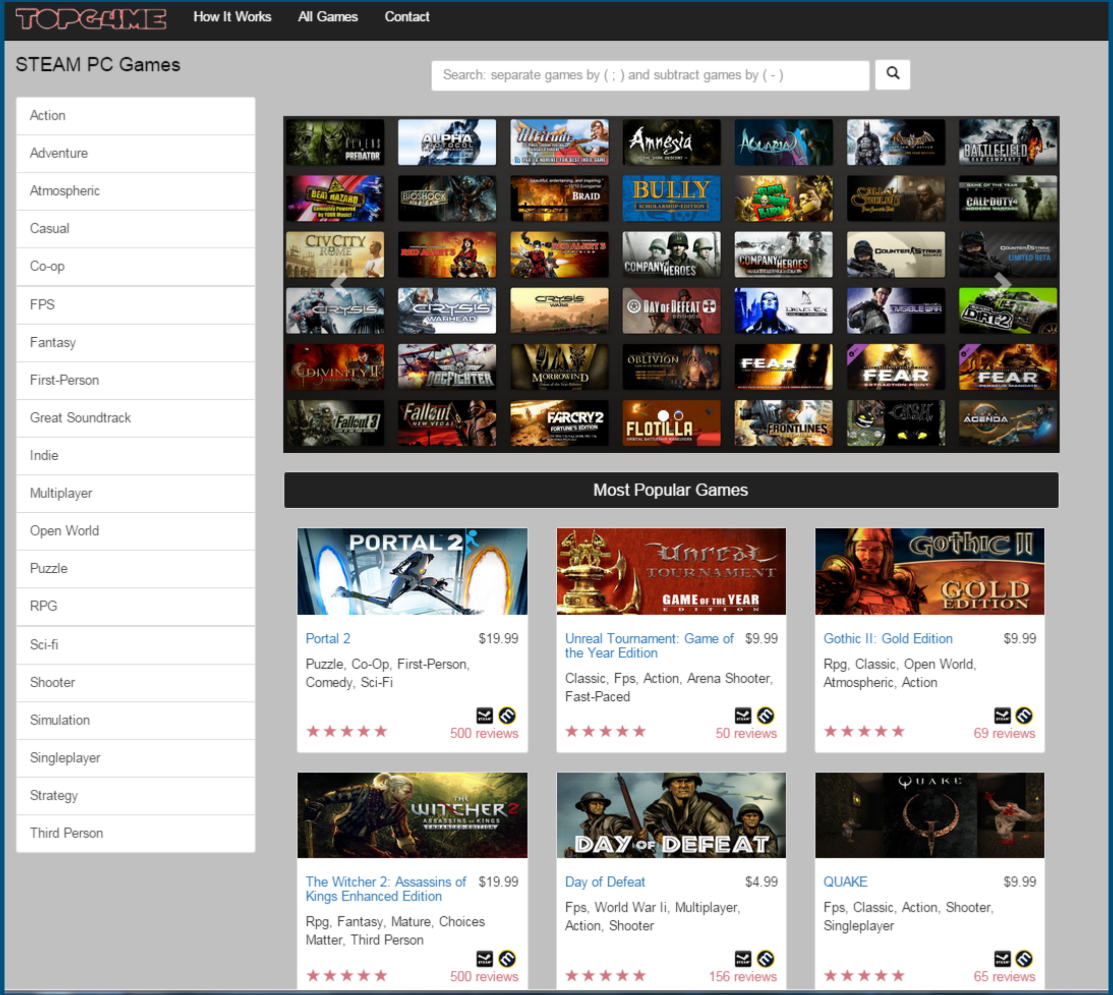
Flask - Web Framework

By: Jeferson Bisconde

Flask Example



STEAM PC Games

- Action
- Adventure
- Atmospheric
- Casual
- Co-op
- FPS
- Fantasy
- First-Person
- Great Soundtrack
- Indie
- Multiplayer
- Open World
- Puzzle
- RPG
- Sci-fi
- Shooter
- Simulation
- Singleplayer
- Strategy
- Third Person

Most Popular Games

Portal 2 $19.99
Puzzle, Co-Op, First-Person, Comedy, Sci-Fi
★★★★★ 500 reviews

Unreal Tournament: Game of the Year Edition $9.99
Classic, Fps, Action, Arena Shooter, Fast-Paced
★★★★★ 50 reviews

Gothic II: Gold Edition $9.99
Rpg, Classic, Open World, Atmospheric, Action
★★★★★ 69 reviews

The Witcher 2: Assassins of Kings Enhanced Edition $19.99
Rpg, Fantasy, Mature, Choices Matter, Third Person
★★★★★ 500 reviews

Day of Defeat $4.99
Fps, World War Ii, Multiplayer, Action, Shooter
★★★★★ 156 reviews

QUAKE $9.99
Fps, Classic, Action, Shooter, Singleplayer
★★★★★ 65 reviews

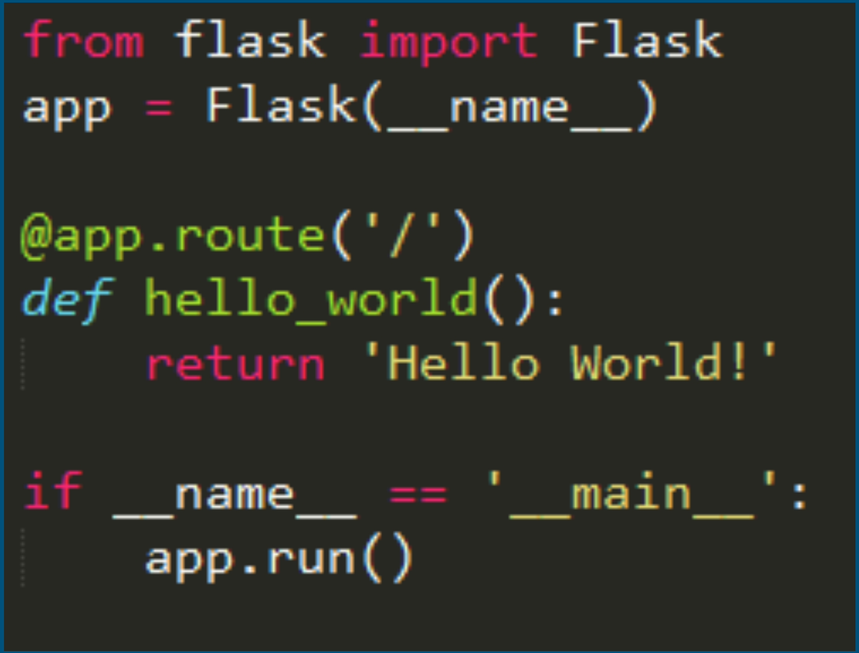
Ingredients

- Flask + other libraries
- Python app (core of every Flask project)
- Templates
    - html
        - Jinja2
- Static
    - CSS
    - Javascript
    - any additional details (fonts etc.)

Flask

- Simple to start out
- Easy to understand
- Well-documented
- Great for self-contained projects
- Tutorial:
    - http://flask.pocoo.org/docs/0.10/quickstart/

Flask - Hello World Example

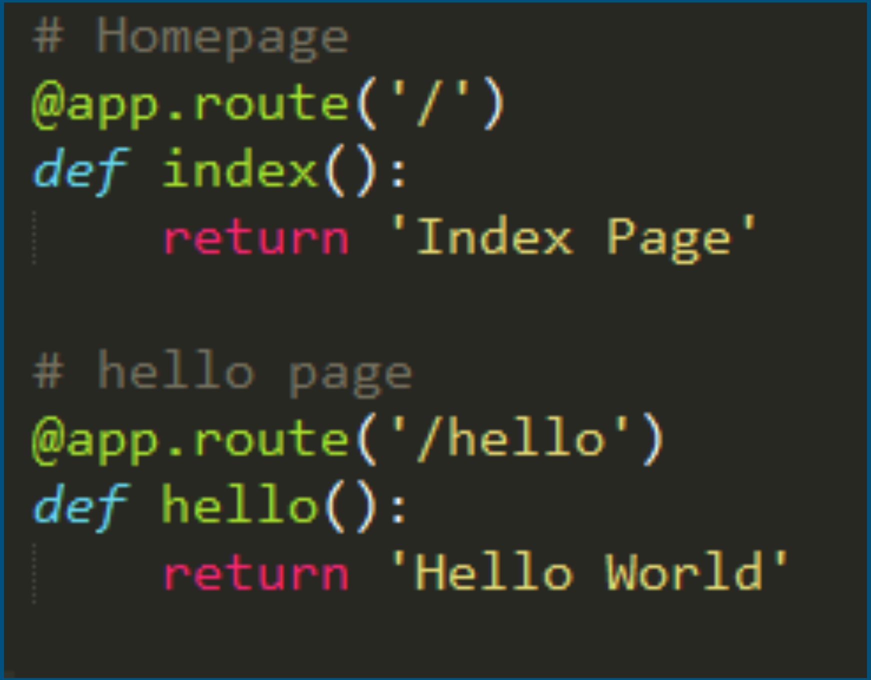


```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

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

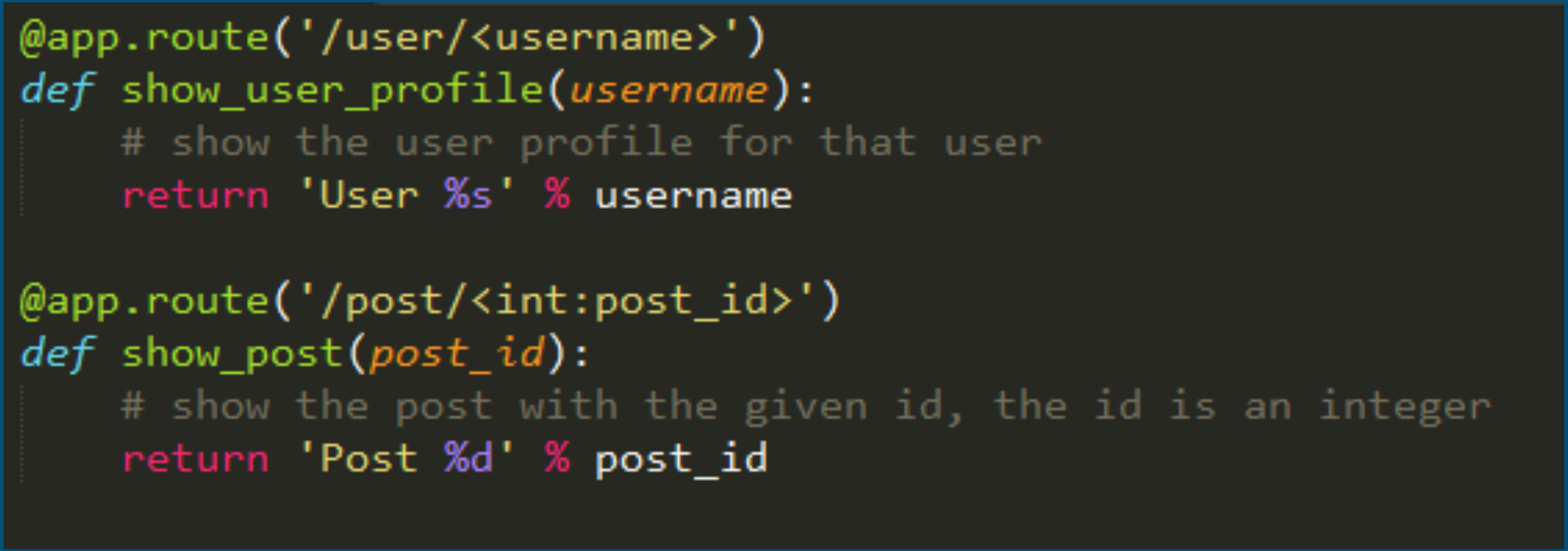
Flask - Routing (decorator)



```
# Homepage
@app.route('/')
def index():
    return 'Index Page'

# hello page
@app.route('/hello')
def hello():
    return 'Hello World'
```

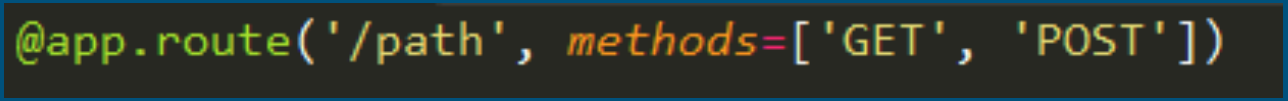
Flask - Variable Rules



```python
@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return 'User %s' % username

@app.route('/post/<int:post_id>')
def show_post(post_id):
    # show the post with the given id, the id is an integer
    return 'Post %d' % post_id
```

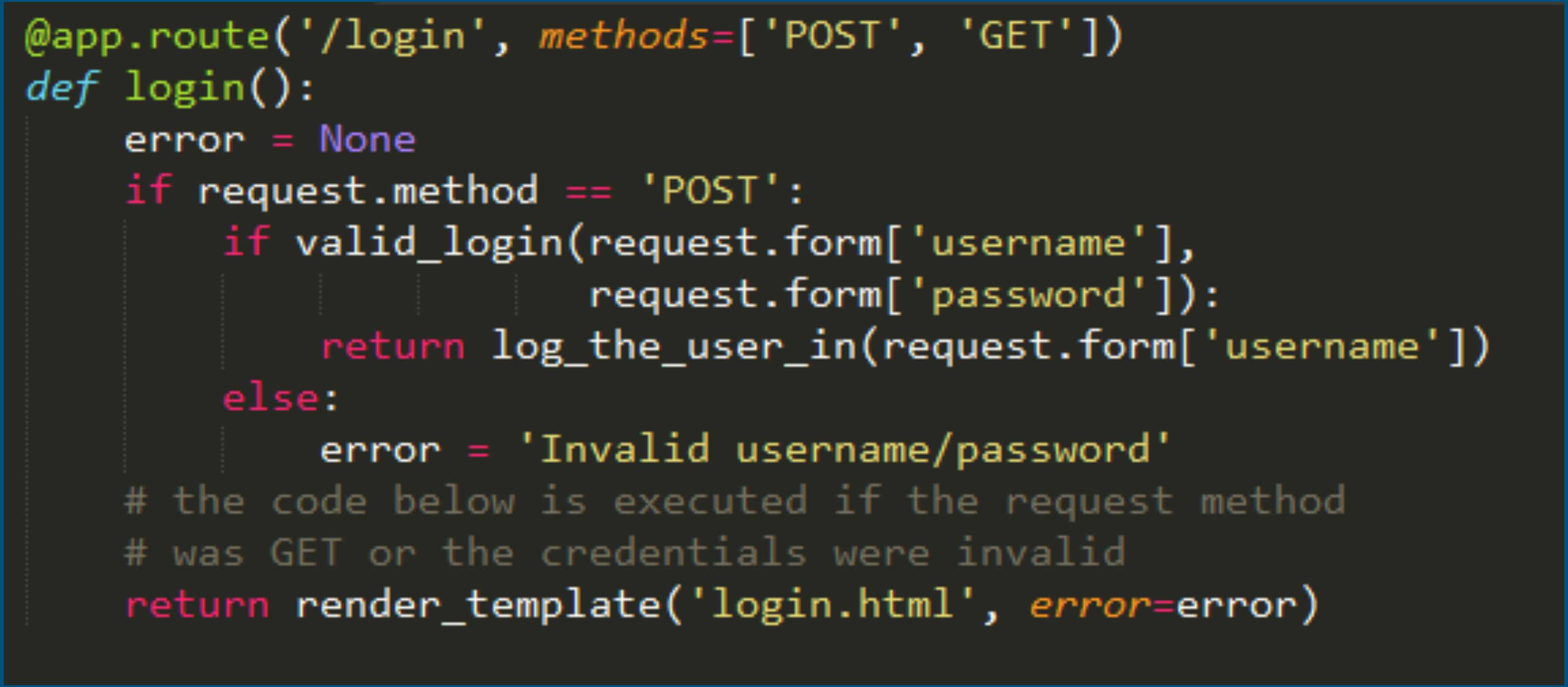
Flask - HTTP Methods



```
@app.route('/path', methods=['GET', 'POST'])
```

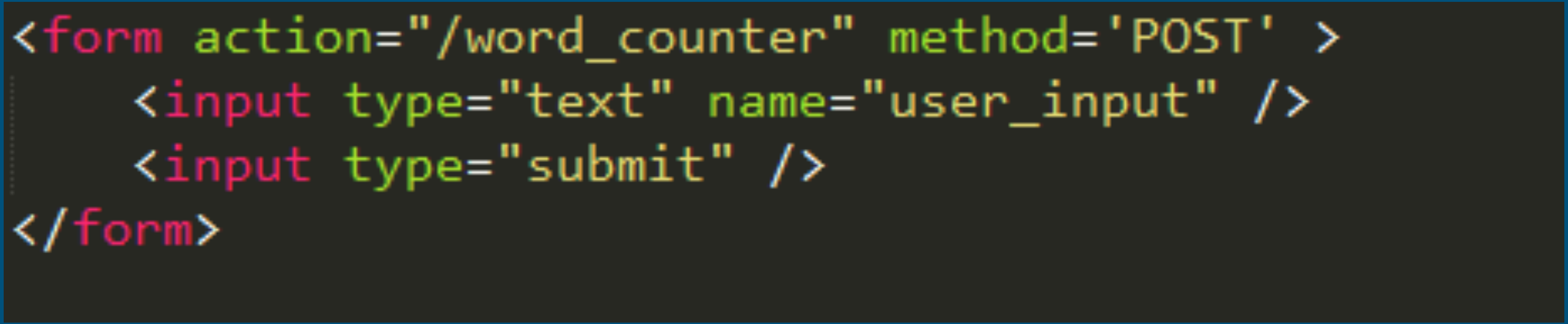
- GET method
  - the browser tells the server to just get the information stored and send it
  - most common method (default)
- POST method
  - the browser tells the server that it wants to post new information
  - forms usually transmit data this way

Flask - Request Data



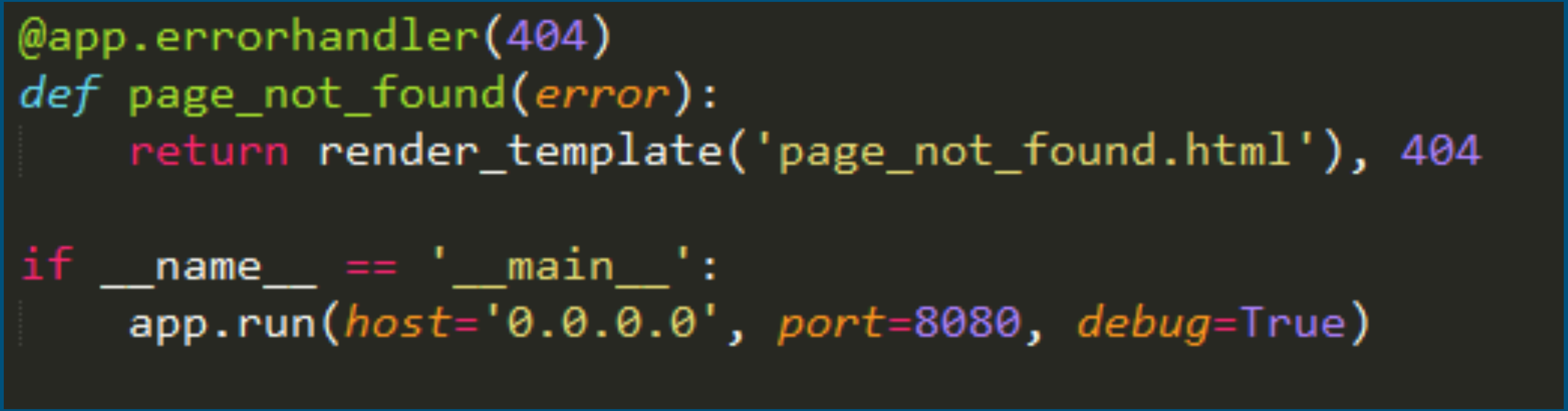
```python
@app.route('/login', methods=['POST', 'GET'])
def login():
    error = None
    if request.method == 'POST':
        if valid_login(request.form['username'],
                       request.form['password']):
            return log_the_user_in(request.form['username'])
        else:
            error = 'Invalid username/password'
    # the code below is executed if the request method
    # was GET or the credentials were invalid
    return render_template('login.html', error=error)
```

Flask - Form page



```
<form action="/word_counter" method='POST' >
    <input type="text" name="user_input" />
    <input type="submit" />
</form>
```

Flask - Debug Mode and Error



```python
@app.errorhandler(404)
def page_not_found(error):
    return render_template('page_not_found.html'), 404

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080, debug=True)
```

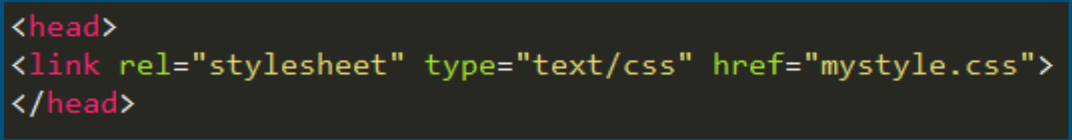
Flask Summary

- The decorator @app.route('/') is what indicates the home page.
  - name of the function is irrelevant
- Here's an explanation of the parameters in app.run:
  - host: Setting the host to 0.0.0.0 means we're running locally
  - port: This is which port to run the app on
  - debug: enables you to see what errors occur
    - You should turn this off in a final live version.

You can see your live app at: http://0.0.0.0:8080/

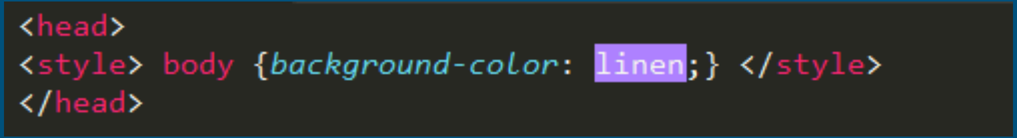
CSS (Cascading Style Sheets)

External



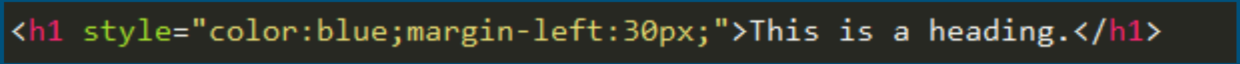
```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Internal



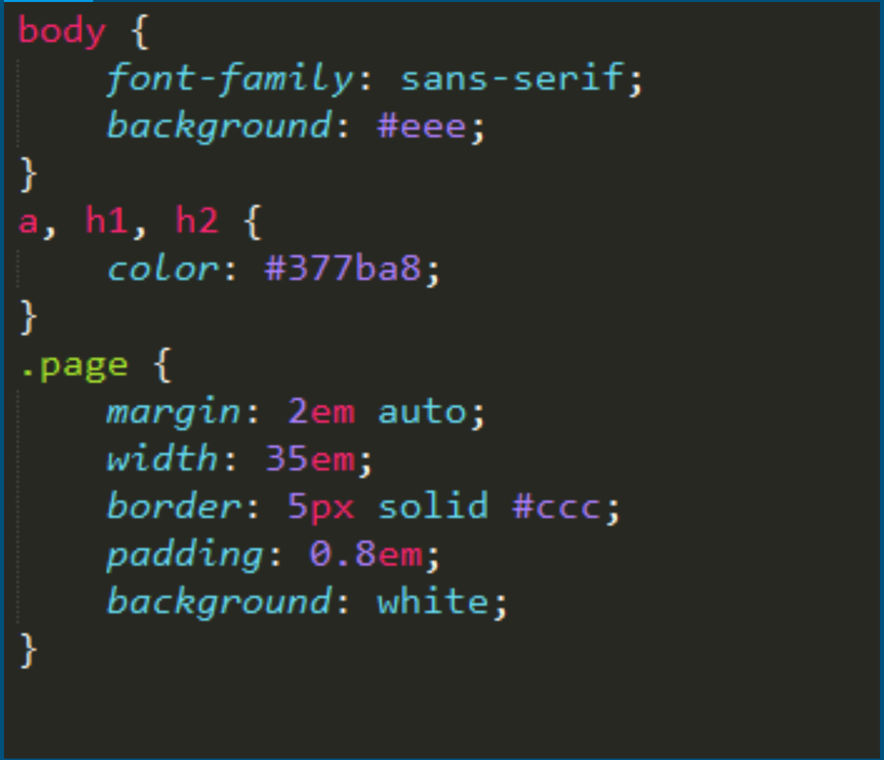
```
<head>
<style> body {background-color: linen;} </style>
</head>
```

Inline



```
<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>
```

CSS - Example



```css
body {
    font-family: sans-serif;
    background: #eee;
}
a, h1, h2 {
    color: #377ba8;
}
.page {
    margin: 2em auto;
    width: 35em;
    border: 5px solid #ccc;
    padding: 0.8em;
    background: white;
}
```

CSS - Summary

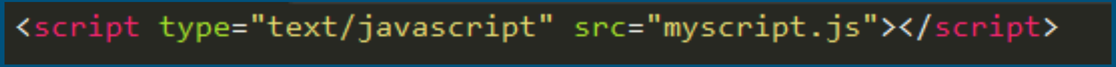
- Can be specified in:
  - external CSS file
  - inside the <head> section
  - inside an HTML element
- Cascading order (from lowest priority to the highest)
  - Browser default
  - External and internal style sheets
  - Inline style
- Tutorial:
  - http://www.tutorialspoint.com/css/

Jinja2

- modern and designer-friendly templating language
- Tutorial - https://realpython.com/blog/python/primer-on-jinja-templating/
- Example:
    - {% set variable_name = value %}
    - {{ variable_name }}
    - {{ variable_name | int }}
    - {% for n in my_list %}
    - {% endfor %}
    - {% if cond %}
    -         do something
    - {% endif %}

Javascript

- don’t use if it’s not needed
  - use forms if you can
- adds interaction/dynamic behavior to your web page



```
<script type="text/javascript" src="myscript.js"></script>
```

- Tutorial:
  - http://javascript.didacto.net/

Things you'll see out there

- Ajax
- Django
- virtualenv
- requirements.txt