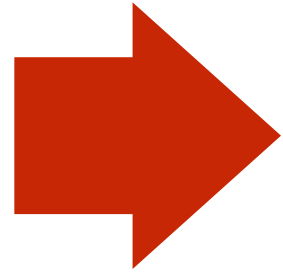


Web Programming

Server-side programming II.

Server-side programming

- Part I. handling requests
- Part II. templating
- Part III. handling data



Motivation

- We need to respond to incoming requests by returning HTML content
- Normally, it would mean a lot of print statements and hard-coded strings in Python code

```
HTML_FRAME_TOP = "<!DOCTYPE HTML>\n<html>\n<head>\n<title>My site</title>\n" \
                  "<link rel=\"stylesheet\" href=\"{css}\"/>\n</head>\n<body>\n"
HTML_FRAME_BOTTOM = "</body>\n</html>\n"

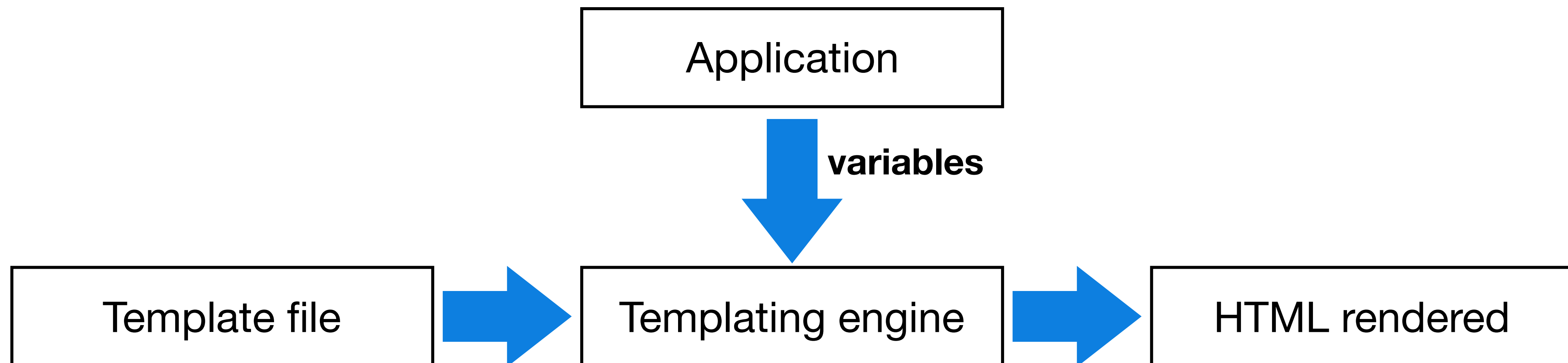
@app.route("/")
def index():
    html = HTML_FRAME_TOP.replace("{css}", url_for("static", filename="style.css"))
    html += "<h1>Hello world</h1>"
    html += HTML_FRAME_BOTTOM
    return html
```

Motivation (2)

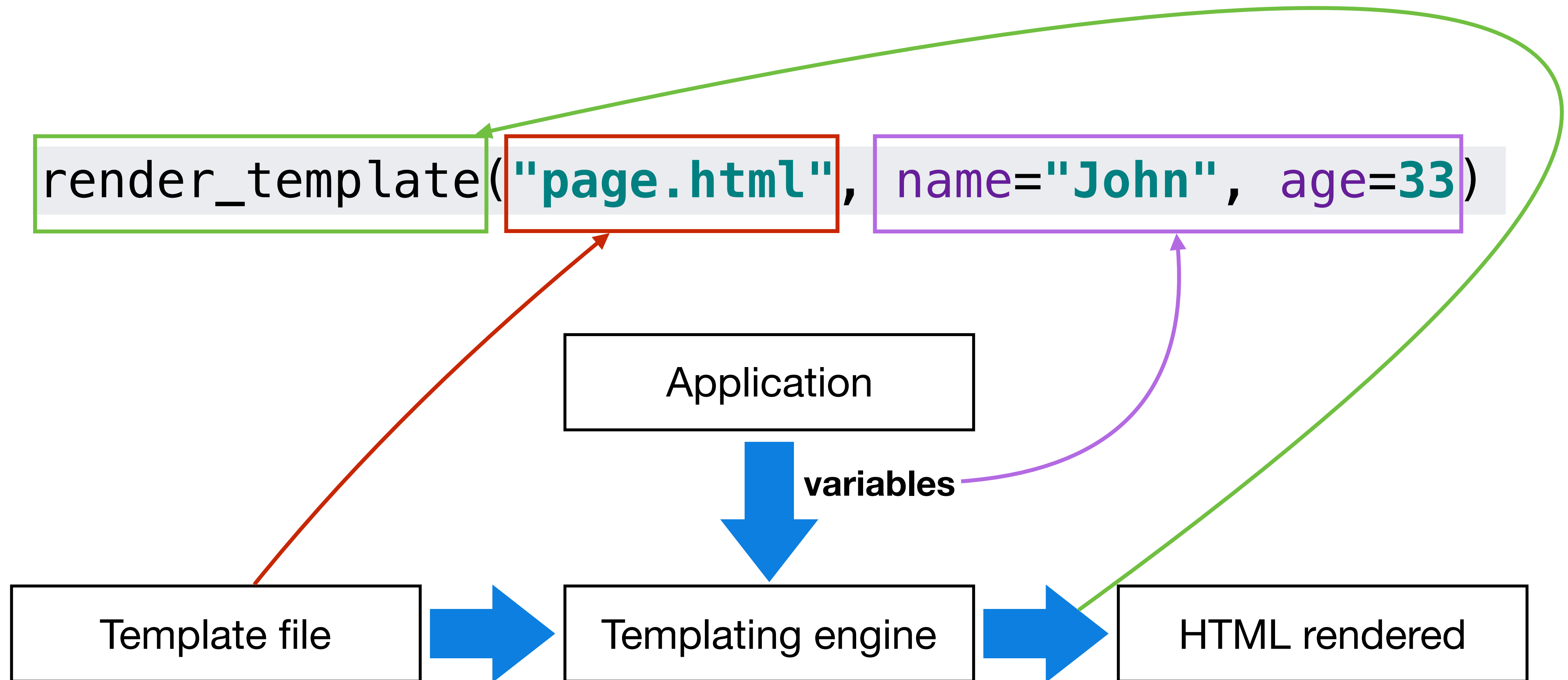
- Instead: separate business logic from presentation
 - Programmers and designers/site builders can work on the same page at once
 - The design can be changed without touching the code
- Idea: make HTML documents and add markup to identify areas that should be replaced

Templating

- A template is a HTML file that contains **variables** and **expressions**, which get replaced with values when the template is rendered



Templating



Example

app.py

```
@app.route("/test")
def test():
    return render_template("page.html", value="123")
```

templates/page.html

```
...
<p>Value given {{ value }}</p>
...
```

HTML rendered

<http://127.0.0.1:5000/test>

```
...
<p>Value given 123</p>
...
```

Templating

- A template is a HTML file that contains **variables** and **expressions**, which get replaced with values when the template is rendered
- `{{ ... }}` expressions (variables)
- `{% ... %}` statements (for, if, include, ...)
- `{# ... #}` comments

Expressions

- Variables

```
{{ value }}
```

- You can calculate with values

```
{{ value * 2 }}
```

- Use logic statements to combine multiple expressions

```
{{ "even" if value % 2 == 0 else "odd" }}
```

Filters

- Variables can be modified by **filters**
 - Filters are separated by the variable by a pipe symbol | and may have optional parameters **{{ var|filter }}**
 - Filters may be chained **{{ var|filter1|filter2(params) }}**

Filters (2)

- There is a risk that a variable will include characters that affect the resulting HTML. Content **escaping** is needed.
- Automatic escaping (default in Flask)
 - Everything is escaped automatically, except for values that are explicitly marked as safe

```
{{ my_variable|safe }}
```

- Manual escaping
 - Convert < > & “ etc. characters to HTML-safe sequences

```
{{ my_variable|e }}
```

- Remove HTML tags

```
{{ my_variable|striptags }}
```

Filters (3)

- Provide default value if the variable is undefined

```
{{ my_variable|default('my_variable is not defined') }}
```

- Convert the value to lower/uppercase

```
{{ my_variable|lower }}
```

```
{{ my_variable|upper }}
```

- Truncate string at a specified length

```
{{ my_variable|truncate(9) }}
```

- Turn URL string into a clickable link

```
{{ my_url|urlize }}
```

Filters (4)

- Replace occurrences of substrings

```
{{ my_variable|replace("john", "nicole") }}
```

- Round a number to a given precision

```
{{ my_variable|round }}
```

- Sum a sequence of numbers
 - also possible to sum only certain attributes

```
Total: {{ items|sum(attribute='price') }}
```

- See the full list here:

<http://jinja.pocoo.org/docs/2.9/templates/#builtin-filters>

Statements: for loop

app.py

```
@app.route("/test")
def test():
    return render_template("page.html", users=["john", "liza", "mary"])
```

templates/page.html

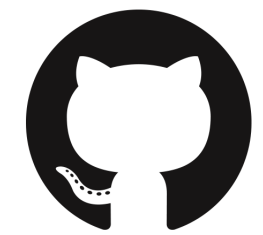
```
<h1>Members</h1>
<ul>
{% for user in users %}
    <li>{{ user }}</li>
{% endfor %}
</ul>
```

Statements: for loop

- Special variables that are accessible inside a for loop
 - **loop.index** — current iteration (indexed from 1)
 - **loop.index0** — current iteration (indexed from 0)
 - **loop.first** — true if first iteration
 - **loop.last** — true if last iteration
 - **loop.length** — number of items in the sequence
 - **loop.cycle** — helper function to cycle between a list of sequences

```
{% for row in rows %}  
    <li class="{{ loop.cycle('odd', 'even') }}">{{ row }}</li>  
{% endfor %}
```

Exercise #1



[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/exercises/python/flask2)
exercises/python/flask2

Template inheritance

- A **base** “skeleton” **template** contains all the common elements of the site and defines **blocks** that child templates can override

app.py

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

templates/index.html

```
{% extends "base.html" %}

{% block content %}
    <h1>Index</h1>
    <p class="important">
        Welcome to my awesome homepage.
    </p>
{% endblock %}
```

templates/base.html

```
<!DOCTYPE html>
<html>
<head> [...] </head>
<body>

    <div id="content">{% block content %}
        default content{% endblock %}</div>

</body>
</html>
```

Template inheritance

- A **base** “skeleton” **template** contains all the common elements of the site and defines **blocks** that child templates can override

app.py

```
@app.route("/")  
def
```

The **{% extends %}** tag tells the template engine that this template “extends” another template. This should be the first tag in the template!

tem

```
{% extends "base.html" %}
```

```
{% block content %}
```

```
<h1>Index</h1>
```

```
<p class="important">
```

```
Welcome to my awesome homepage.
```

```
</p>
```

```
{% endblock %}
```

templates/base.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<head> [...] </head>
```

```
<body>
```

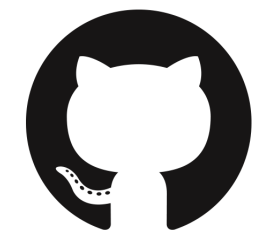
```
<div id="content">{% block content %}
```

```
default content{% endblock %}</div>
```

```
</body>
```

```
</html>
```

Exercise #2



[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/exercises/python/flask2)
exercises/python/flask2