N02_03_Jinja Expressions

In this chapter, students will learn and understand:

- State examples of delimiters used in Jinja templating language
- Use {# … #} as comments in templates
- Use {% if %} {% elif %} {% else %} {% endif %} conditional in templates
- Use {% for %} {% endfor %} control structure in templates
- Escape a delimiter using single quotes
- Explain the use of variable filters: length and safe

Section 1: Introduction of Jinja

1.1 Delimiters in Jinja

In general, these are the types of delimiters in Jinja that you will need to use.

No	Delimiter	Description
1	`{# … #}`	Used to denote comments in templates. Will not be included as output.
2	`{{ … }}`	Used to output the results of an expression or a variable to the end user. When this is rendered, it is replaced with a value and passed as a response to the browser.
3	`{% … %}`	Used to denote control structures that control the flow of a program, which include conditionals and loops. With the default syntax, control structures appear inside `{% ... %}` blocks. They include: • `{% if %} {% elif %} {% else %} {% endif %}` • `{% for %} {% endfor %}`

Exercise

Describe briefly what the following block of code does.

```
<ul>
{% for user in users %}
    <li><a href="{{ user.webpage }}">{{ user.username }}</a></li>
{% endfor %}
</ul>
```

……

1.2 Using Comments and Conditionals in Templates

Recall that in the previous task, you used `{{ ... }}` as variables to be passed in as arguments into functions and to output as display into a browser. Now you will try to create a comment in your template file.

1. In templates directory, open **about_us.html**. Add the following block of code to the `<body>` tag. What is the data type for the variable age in this case?

```
{# This block outputs a statement depending on the age #}
{% if age < 18 %}
<p>You are considered a teenager. </p>
{% elif 18 <= age <= 50 %}
<p>You are considered an adult. </p>
{% else %}
<p>You are considered a senior. </p>
{% endif %}
```

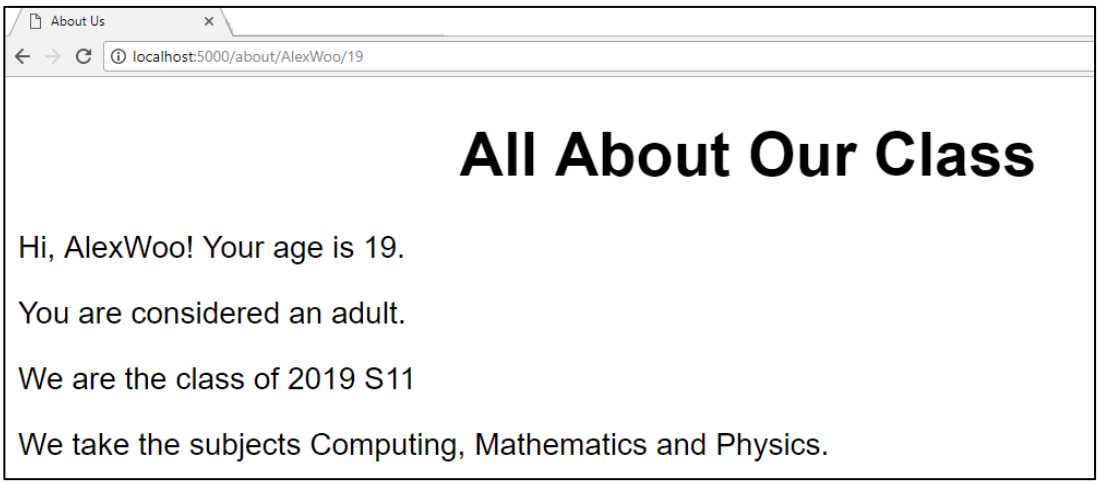
2. In **app.py**, modify your code such that the `about_us` function takes in a string argument (`name`) and an integer argument (`age`).

3. Run **app.py**.

4. Test the webpage `http://localhost:5000/<username>/<userage>`.

Replace `<username>` and `<userage>` with a name and age of your choice.

You should see the following if your application works.



What actually happens is that `about_us` function accepts the `<age>` variable as an argument when the user types it into the address bar of the browser. **about_us.html** is then rendered with `<age>` being passed into the template. Inside the template, the value of `<age>` is being compared in the conditional statement and the corresponding message is then displayed to the user.

1.3 Using For Loop In Templates

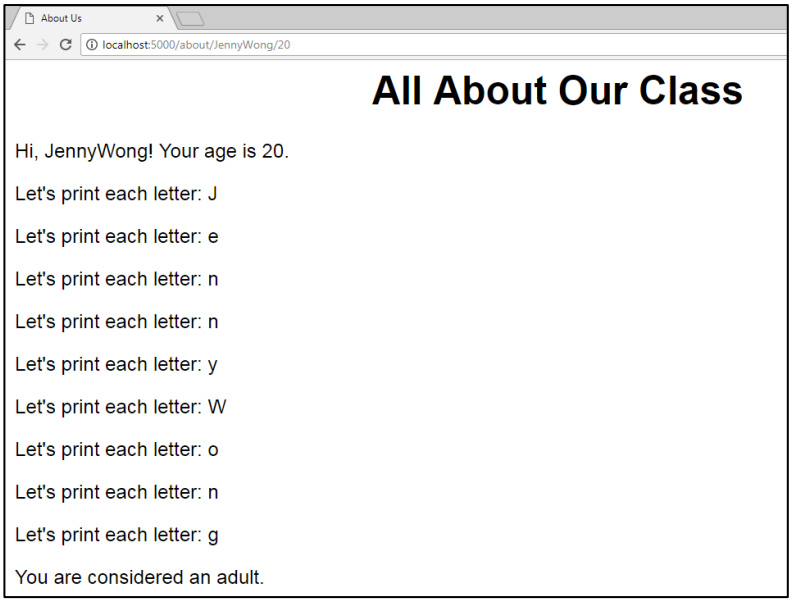
Note that Jinja2 supports `for` loops, but not `while` loops.

1. Open **about_us.html**.

2. Add a `for` loop to print each letter of the username on a new line.

3. Test the webpage `http://localhost:5000/<username>/<userage>`.

Replace `<username>` and `<userage>` with a name and age of your choice.
The output should look like this:



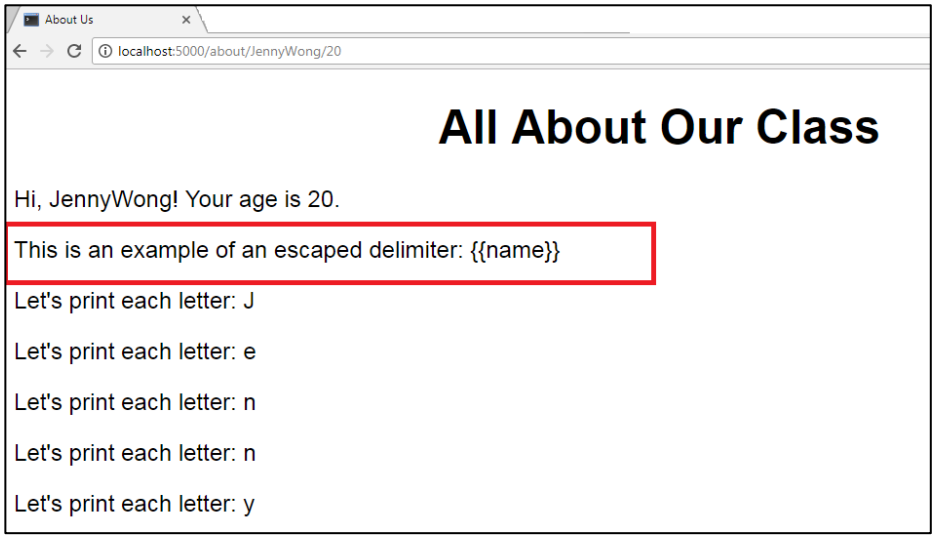
1.4 Escaping a Delimiter In Templates

If you want to use `{{ ... }}` as a string in a template but not as a variable, you have to use a special way to escape the delimiter. The easiest way is to output a variable delimiter by using a variable expression with single quotes as shown below.

```
{{ '{{' }}
```

1. Open **about_us.html**.

2. Modify the webpage such that it will display `{{ name }}` as a string on your webpage instead of the variable value. The output should look like this.

3. 

1.5 Using Variable Filters

Variables can be modified by filters. Filters are separated from the variable by a pipe symbol | and may have optional arguments in parentheses. For example, `{{ name | safe }}` will render the value as safe HTML without applying escape.

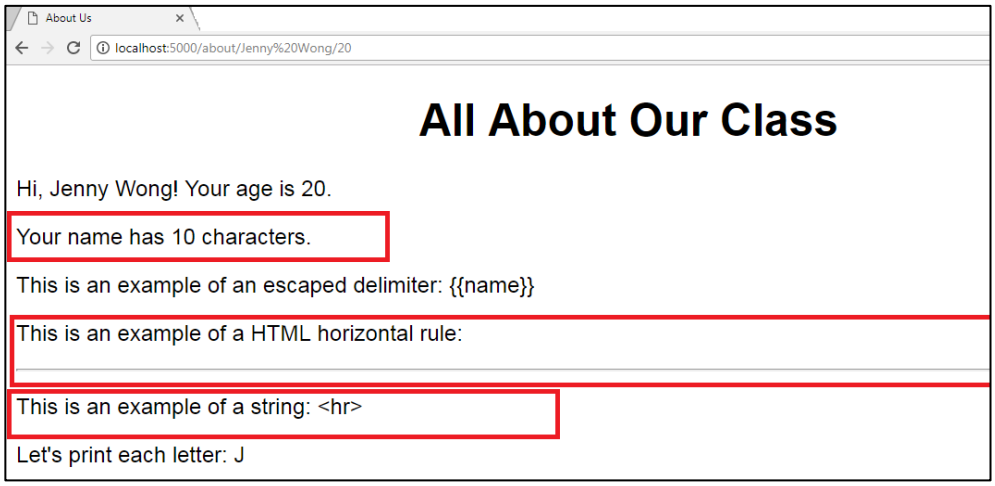
No	Variable Filter	Description
1	`{{ '<hr>' | safe }}`	Denotes it as safe HTML without applying escaping.
2	`{{ 'haha' | length }}`	Returns the length of the string.
3	`{{ range(1, 8) | random }}`	Returns a random value from the range.

1. In **about_us.html,** add code such that the length of the `{{ name }}` variable is printed.

2. Add code such that the `<hr>` tag will appear as a horizontal rule on the webpage.

3. Add code such that the `<hr>` tag will appear as a string on the webpage.

Your output should look like this.



References:

- MOE Python Flask Web Application Starter Kit
- MOE Teacher Training 2018 - HTML and CSS
- Udemy: Python and Flask Bootcamp: Create Websites using Flask!
- Udemy: Build Responsive Real World Websites with HTML5 and CSS3