# CSPB-3308 Lab 6 : Flask Tutorial - Custom Routes

## Exploring HTML

We will be exploring HTML in a future lab, but here are a few things you need to know to complete the Flask lab.

HTML is a tag language. There are start and end tags that you use to give the browser instructions about how to present your information. I consider writing HTML to be the same as other programming languages. Instead of giving instructions to the Python command interpreter or the C compiler, you are instructing the browser.

To make a section of text be displayed in a bold font, place the `<b>` and `</b>` tags around the text.

```
Normal text can be <b> bolded text </b> and then return to normal.
```

You can also specify text to be displayed as italic.

```
Normal text can be <b> bolded text </b> and then return to normal.
It can also be <i> italic text </i>.

<b><i>OR both!</i></b>
```

You can overlap the instructions (although it is not the best practice because it does not work in every rendering application).

```
This is normal, <b>bolded text, <i>bold and italic,</b>
just italic</i>.
```

## Creating Interesting Routes for Flask Server

In the description of the lab, you are asked to create your own routes that display pages that have specific characteristics. In this document, we will attempt to give examples of the HTML display that each one of the route ideas would produce.

## Static Pages

Each of the static pages can be stored within the Python server code as a single string. You will have already created some of the when implementing the simple routes that return strings like "Hello".

You can create multiple line strings in Python to store long strings.

```
@app.route('/about')
def about():
   long_string = ''' This is text on multiple
                     lines of the Python file,
                     but are all part of one string!
                 '''

   return long_string

@app.route('/table')
def table():
   long_string = ''' This is text outside of table.
                     <table>

                          --- place HTML for rows/columns here --

                     </table>
                 '''
```

```
    return long_string
```

The results from a call to `about()` will have all the indentation and line endings in the string.

```
' This is text on multiple\n                        lines of the Python file,\n
but are all part of one string!\n                    '
```

You can add HTML tags into those static strings. For example, the HTML above for bold and italic could be stored as a string and returned as the response from a route. The different types of HTML displays for lists and tables can also be stored statically. Below is a simple table that has two rows and three columns filled with data.

```
<table>
 <tr>
     <td>one A</td>
     <td>one B</td>
     <td>one C</td>
 </tr>
 <tr>
     <td>two A</td>
     <td>two B</td>
     <td>two C</td>
 </tr>
</table>
```

Each row is delimited by the `<tr> ... </tr>` tag. Similarly, each column is delimited by the `<td> ... </td>` tag. The whole table is delimited with the `<table> ... </table>` tag. Here is what a string with the table tag listed above will look like when it is returned from a route and rendered in the display by a browser.

| | | |
|---|---|---|
| one A | one B | one C |
| two A | two B | two C |

W3Schools is a great place to learn how to use different HTML tags. See the instructions on how to use different tags in HTML at the W3Schools.com website.

| Construct | Link |
|---|---|
| Bold | https://www.w3schools.com/tags/tag_b.asp |
| Italic | https://www.w3schools.com/tags/tag_i.asp |
| Headers | https://www.w3schools.com/tags/tag_hn.asp |
| Ordered Lists | https://www.w3schools.com/tags/tag_ol.asp |
| Unordered Lists | https://www.w3schools.com/tags/tag_ul.asp |
| Tables | https://www.w3schools.com/tags/tag_table.asp |

## Dynamic Creation of HTML string

The string can be stored statically like we have seen above, or dynamically created.
Using concatenation we can add more text to a string.
For example, we can place all the values from 1 to 10 on separate lines.

```
response = "<b>"   # Make everything bold

for i in range(10):
    response += str(i+1)
    response += "<br>"   # cause a line break
response += "</b>"   # close bold tag
```

We could get the data for each element from a list, a dictionary, read them from a file, or even get the data from a database. You dynamically build the HTML string to be returned to the browser.

## Reading Data from a File

You can read and parse lines of data from a file as needed. Once you have the data, you would dynamically generate a information to be placed on the page.

See Example of using Python WITH statement to view examples of reading file data into a string.