Presenting data on the web – HTML & CSS

INFO20002: Foundations of Informatics
05/04/2016

Outcomes

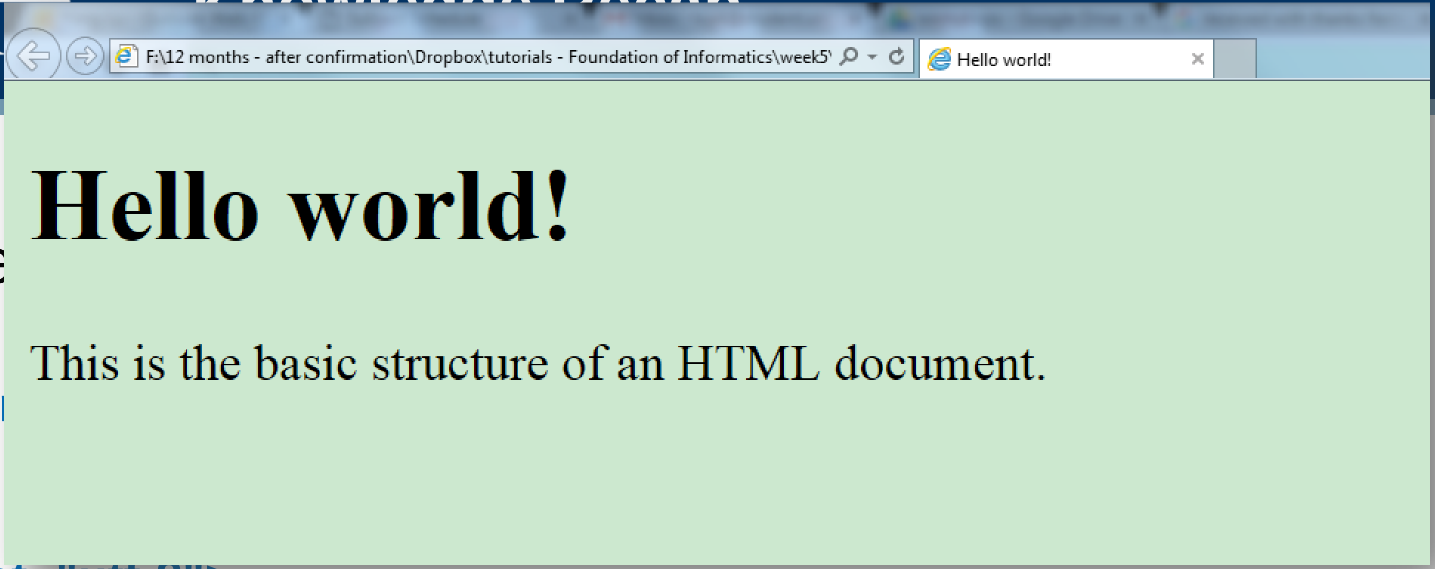
- Learn the format, relation and function of HTML and CSS
- Use FLASK framework to build dynamic web page

THE UNIVERSITY OF MELBOUR

Knowledge Recap

HTML (hype

```
<!DOCTYPE ht
<html>
 <head>
   <meta charset="utf-8">
   <title>Hello world!</title>
 </head>
 <body>
   <h1>Hello world!</h1>
   <p>This is the basic structure of an HTML document.</p>
 </body>
</html>
```


F:\12 months - after confirmation\Dropbox\tutorials - Foundation of Informatics\week5'
Hello world!

Hello world!

This is the basic structure of an HTML document.

Knowledge Recap

HTML tags – markup the format/meaning of the web elements.

- Structural markup
    - Heading <h1>,<h2>,<h3>
    - Paragraphs: <p>
- Stylistic markup
    - Bold: <b>
    - Italic: <i>
- Semantic markup
    - Links: <a>
    - Tables: <table> (<tr/><th/><tb/>)
    - Forms: <form> (<input type="text"/>…)

http://www.w3schools.com/html/html_lists.asp

Difference between block and Inline elements

Block-level elements – represented from the newlines

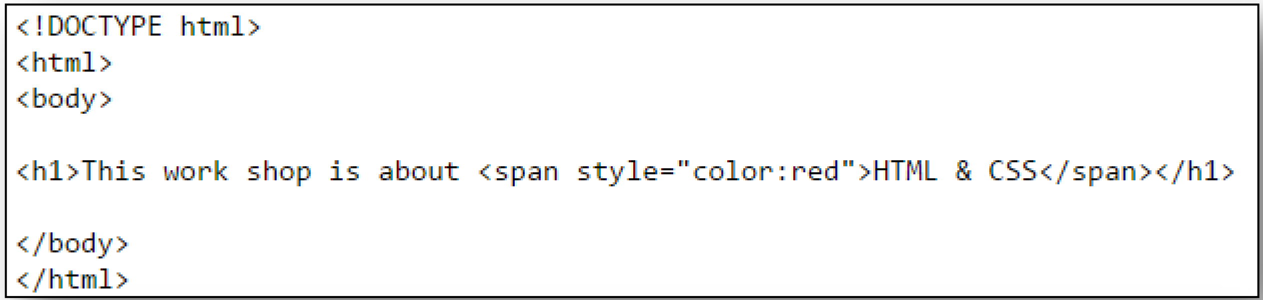
Examples :

- <div>
- <h1> - <h6>
- <p>

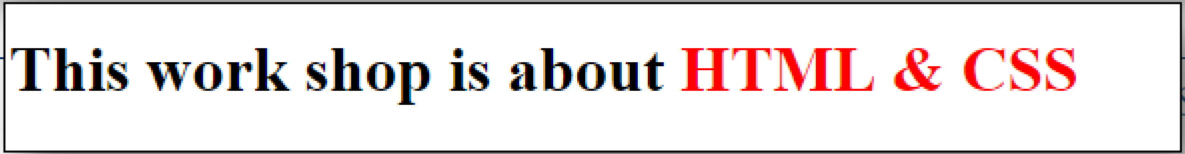
Inline elements – only takes up as much width as necessary

Examples:

- <span>
- <a>
- <img>



```
<!DOCTYPE html>
<html>
<body>

<h1>This work shop is about <span style="color:red">HTML & CSS</span></h1>

</body>
</html>
```



This work shop is about HTML & CSS

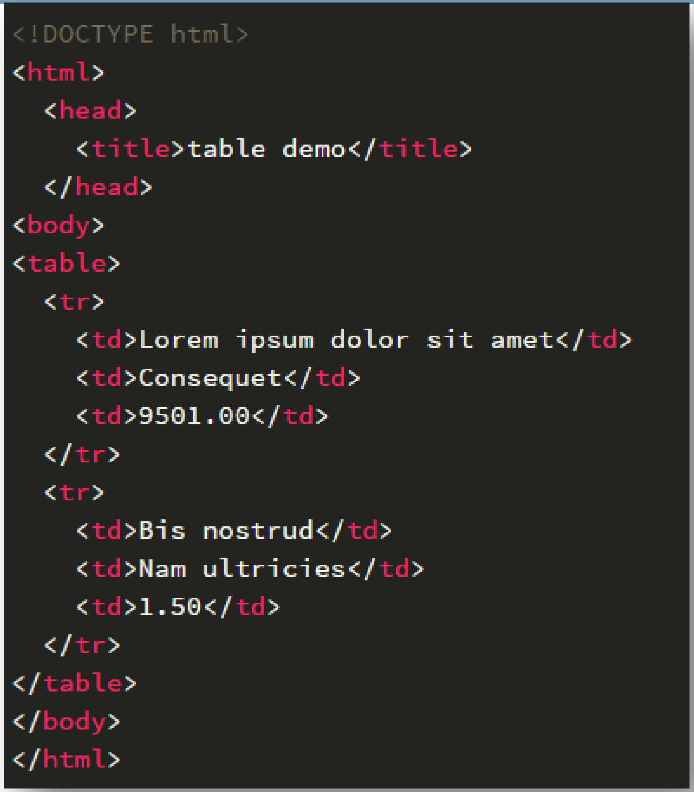
CSS

CSS – Cascading style sheets

- Separated contents from HTML
- Reduces the amount of effort required to stylise a document or web site

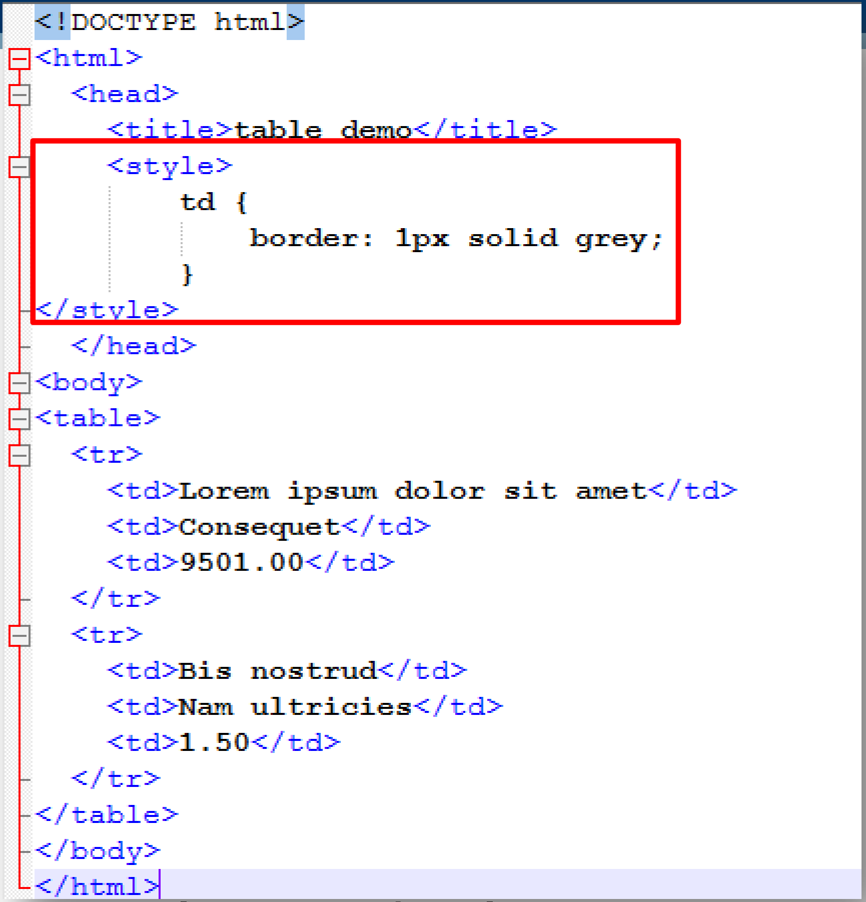
  - Embedded css: declared in a css style block inside the document head (<head>)
  - External css: using selector (e.g. h1{}, table{}, td{}) –to point the part to stylize

Exercise 1


```
<!DOCTYPE html>
<html>
  <head>
    <title>table demo</title>
  </head>
<body>
<table>
  <tr>
    <td>Lorem ipsum dolor sit amet</td>
    <td>Consequet</td>
    <td>9501.00</td>
  </tr>
  <tr>
    <td>Bis nostrud</td>
    <td>Nam ultricies</td>
    <td>1.50</td>
  </tr>
</table>
</body>
</html>
```

1. Save it as ***table.html*** and serve it in the browser. Observe the output
2. Embed the css rules to stylise the web. Observe the output

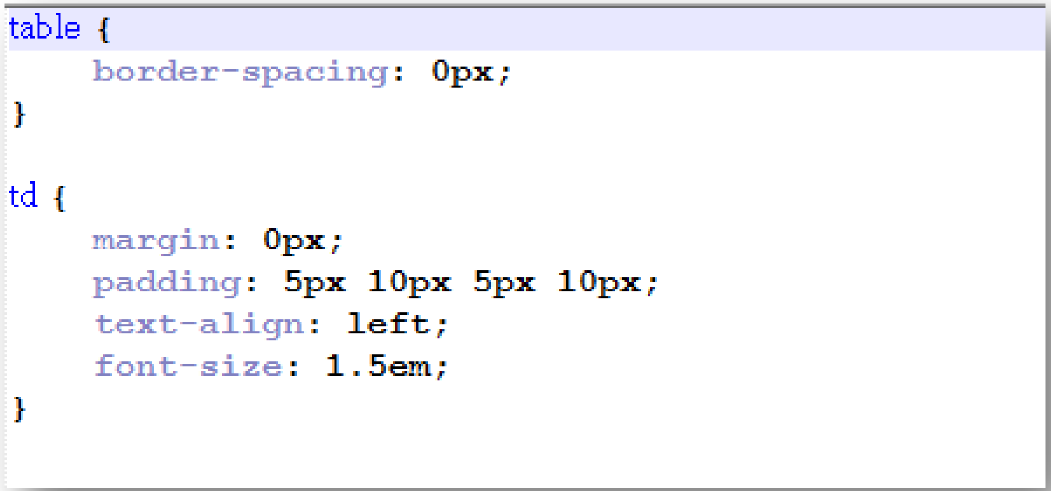
Exercise 2



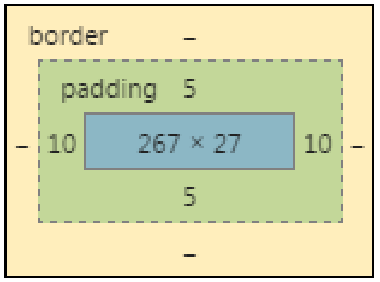
```
<!DOCTYPE html>
<html>
  <head>
    <title>table demo</title>
    <style>
        td {
            border: 1px solid grey;
        }
</style>
  </head>
<body>
<table>
  <tr>
    <td>Lorem ipsum dolor sit amet</td>
    <td>Consequet</td>
    <td>9501.00</td>
  </tr>
  <tr>
    <td>Bis nostrud</td>
    <td>Nam ultricies</td>
    <td>1.50</td>
  </tr>
</table>
</body>
</html>
```

1. Remove the "<style>… </style>"
2. Create a separate ***table.css*** with the given rule.
3. Use **link** method to reference ***table.css*** file
4. Add more stylistic rules (as the hands-on required)

Try the inspector – F12

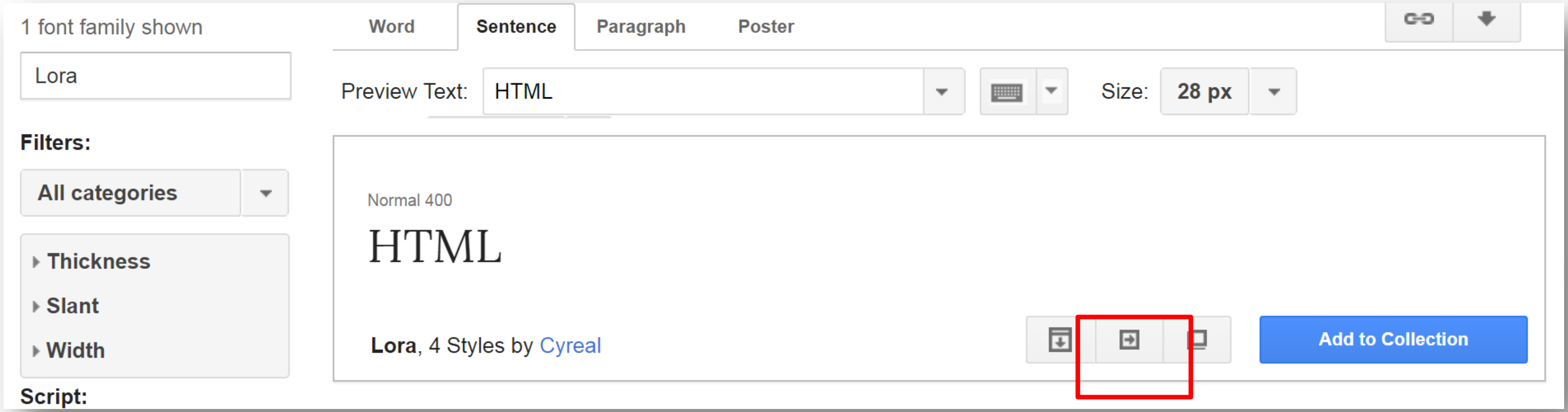


```
table {
    border-spacing: 0px;
}

td {
    margin: 0px;
    padding: 5px 10px 5px 10px;
    text-align: left;
    font-size: 1.5em;
}
```



Take a look at google fonts – Exercise 3

Google fonts is a set of online collection of typefaces


1 font family shown

Lora

Filters:

All categories

Thickness

Slant

Width

Script:

Lora, 4 Styles by Cyreal

```
@import url(http://fonts.googleapis.com/css?family=Lora);
```

```
font-family: "Lora", serif;
```

1. Use the ***Lora*** font by adding rules in ***table.css***
2. Stylize the contents of **table cells**

In case of the browser failure, you may have an alternative choice, e.g. *serif* – fallback system

THE UNIVERSITY OF MELBOURNE

Exercise 4 – play with colour

Adobe color is a online palette with different colors…

1. Select one palette and choose three colors from it
2. Using CSS2's first-child pseudo and + (sibling operator) as the selector to achieve following effects:

Column 1 of the table

Font color: black
Background color: color-palette-1
Text align: left

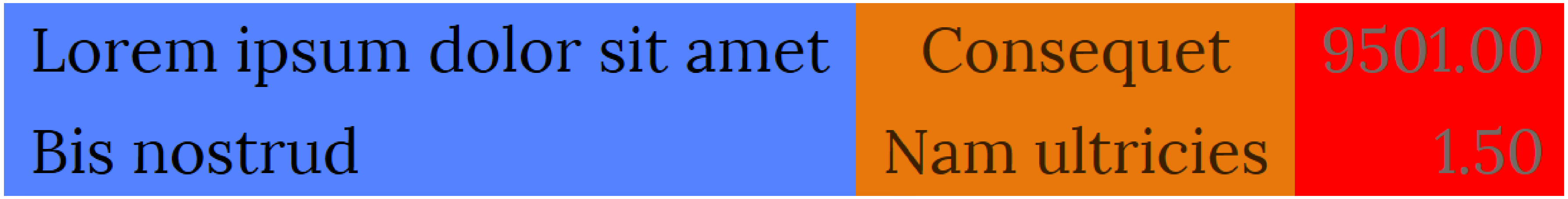
Column 2 of the table

Font color: rgb(60,30,0)
Background color: color-palette-2
Text align: center

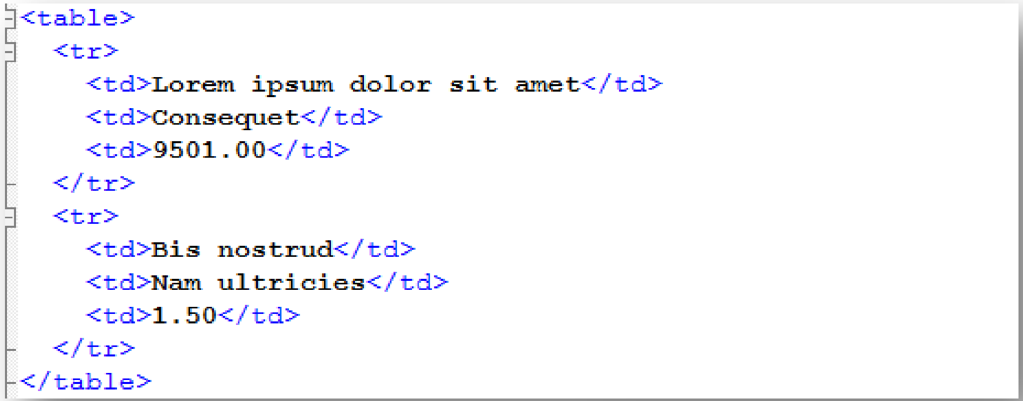
Column 3 of the table

Font color: #736A65
Background color: color-palette-3
Text align: right

3. Once you finish it, try the css3's pseudo class nth-child() to achieve the same effect



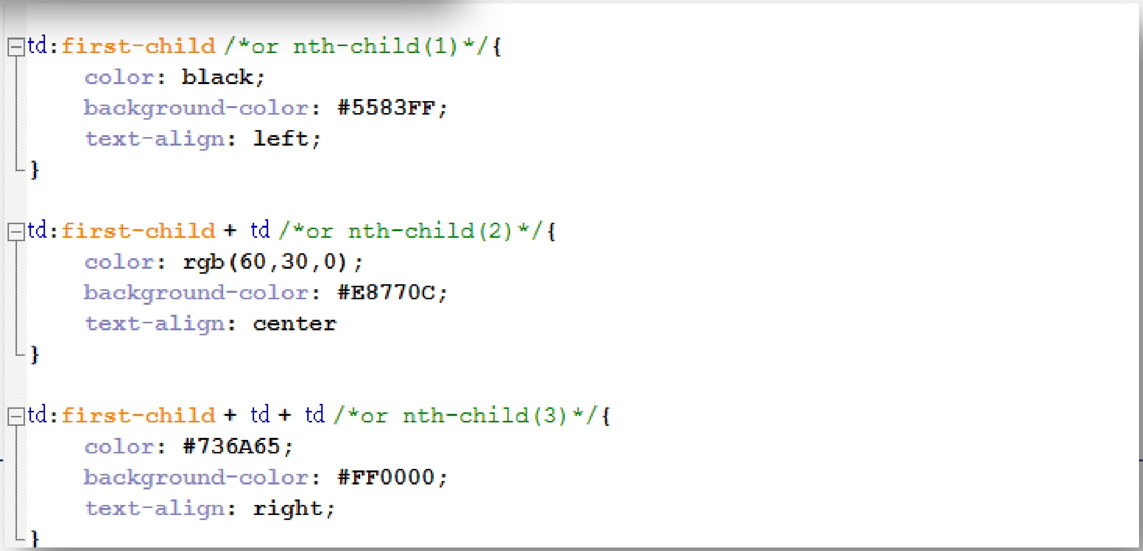
Lorem ipsum dolor sit amet	Consequet	9501.00
Bis nostrud	Nam ultricies	1.50



```
<table>
  <tr>
    <td>Lorem ipsum dolor sit amet</td>
    <td>Consequet</td>
    <td>9501.00</td>
  </tr>
  <tr>
    <td>Bis nostrud</td>
    <td>Nam ultricies</td>
    <td>1.50</td>
  </tr>
</table>
```

HTML

CSS

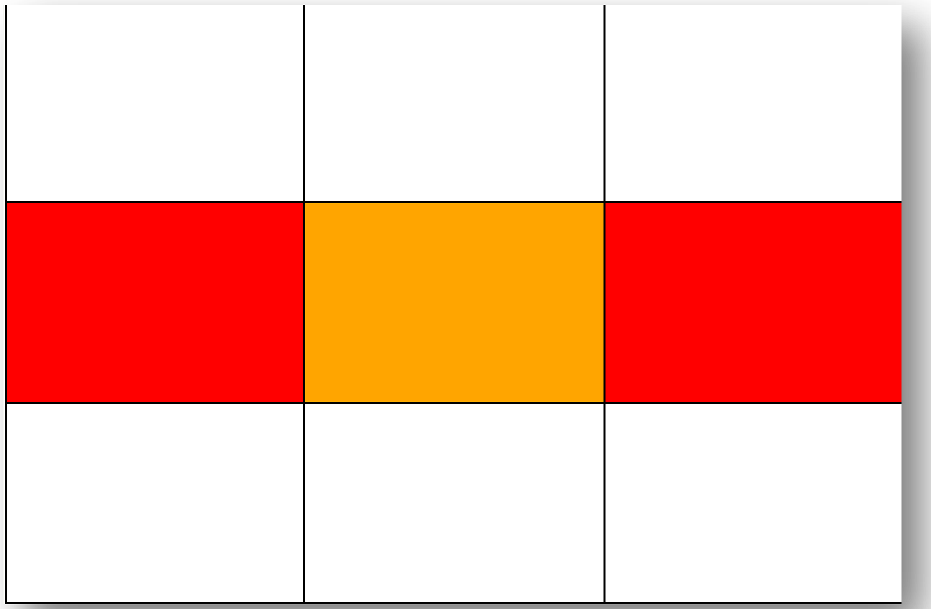


```
td:first-child /*or nth-child(1)*/{
    color: black;
    background-color: #5583FF;
    text-align: left;
}

td:first-child + td /*or nth-child(2)*/{
    color: rgb(60,30,0);
    background-color: #E8770C;
    text-align: center
}

td:first-child + td + td /*or nth-child(3)*/{
    color: #736A65;
    background-color: #FF0000;
    text-align: right;
}
```

Exercise 5: HTML & CSS table

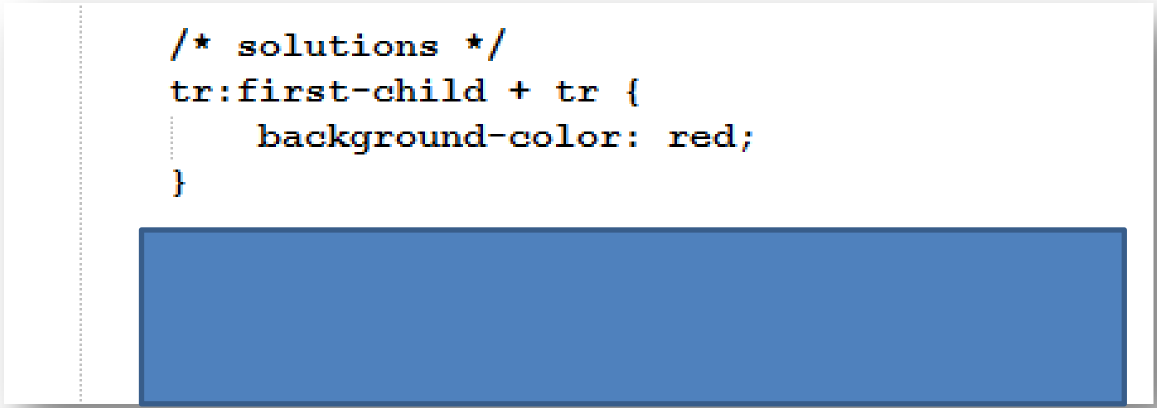
1. Save a new file as ***table2.html***
2. Achieve this effect by adding css rules (*embedded* css)

Both css2 and css3 are good to use

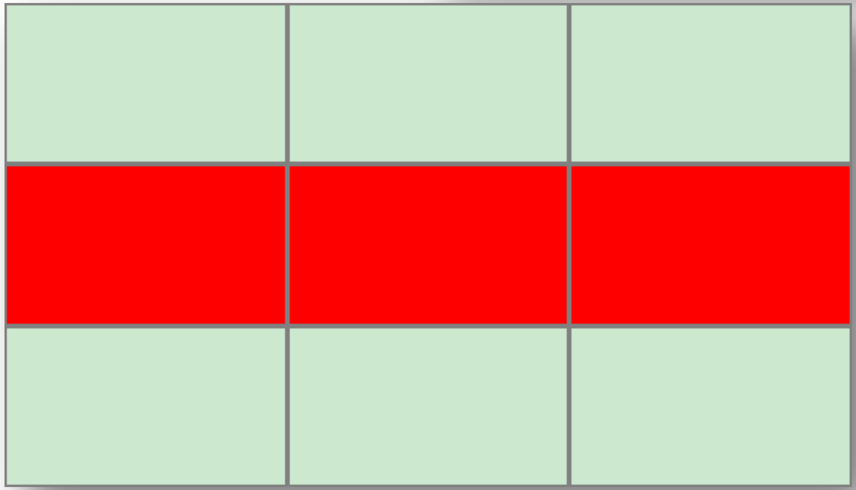


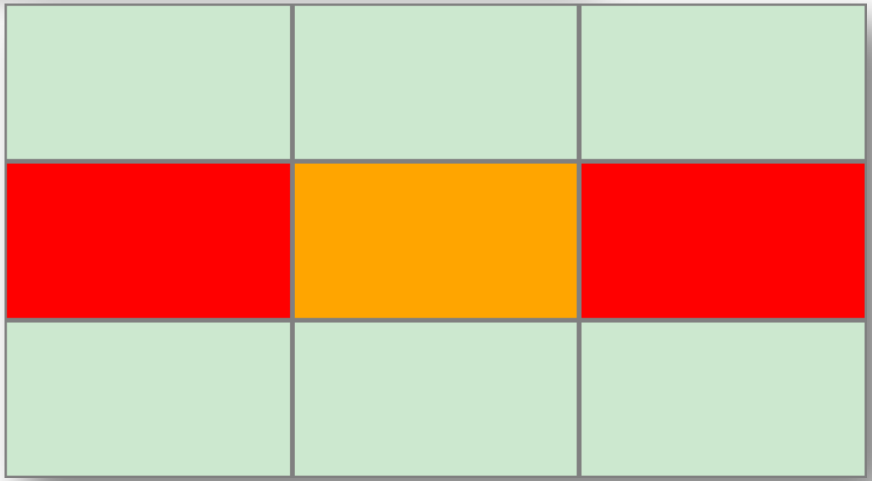
🟥	🟧	🟥

Solution



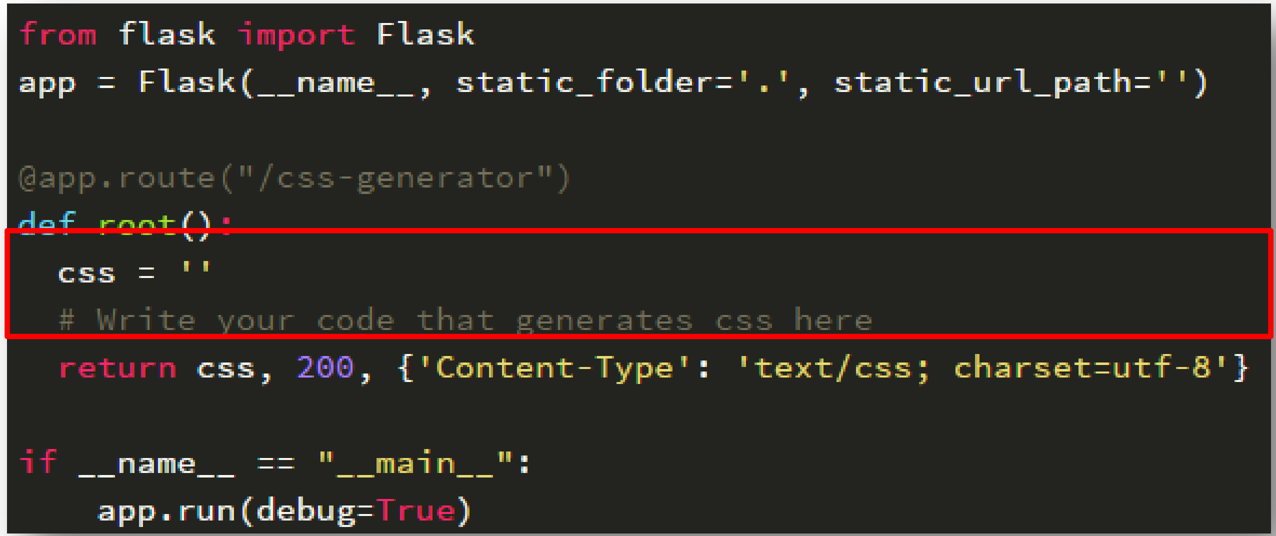
```
/* solutions */
tr:first-child + tr {
    background-color: red;
}
```





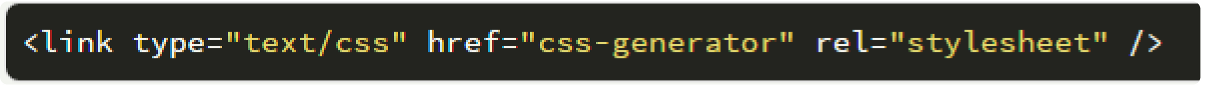
"Flask"

Recall how to set up web application by using Flask class



```python
from flask import Flask
app = Flask(__name__, static_folder='.', static_url_path='')

@app.route("/css-generator")
def root():
  css = ''
  # Write your code that generates css here
  return css, 200, {'Content-Type': 'text/css; charset=utf-8'}

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

1. In the same directory, create an empty Python file called **css.py**
2. Remove the embedded css and add the linking command:



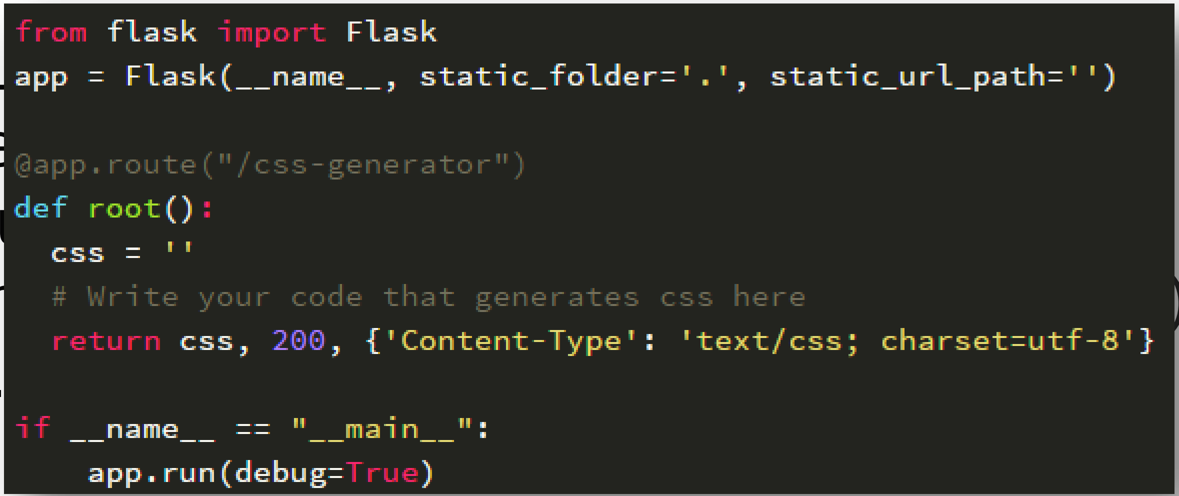
```html
<link type="text/css" href="css-generator" rel="stylesheet" />
```

3. Run the app from **http://localhost/table2.html**

THE UNIVERSITY OF MELBOURNE

Exercise 6 – randomise colours

HSL
Hue
Sat
Ligh)
e.g.



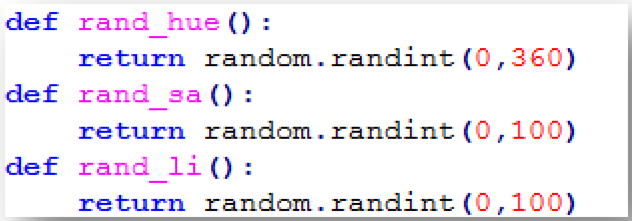
```python
from flask import Flask
app = Flask(__name__, static_folder='.', static_url_path='')

@app.route("/css-generator")
def root():
    css = ''
    # Write your code that generates css here
    return css, 200, {'Content-Type': 'text/css; charset=utf-8'}

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

Modify the ***css.py*** so that it can generate random colors to every cell once reloading the page

–Search the module “random” and function “random.randint”
–Concatenate the ‘css’ string

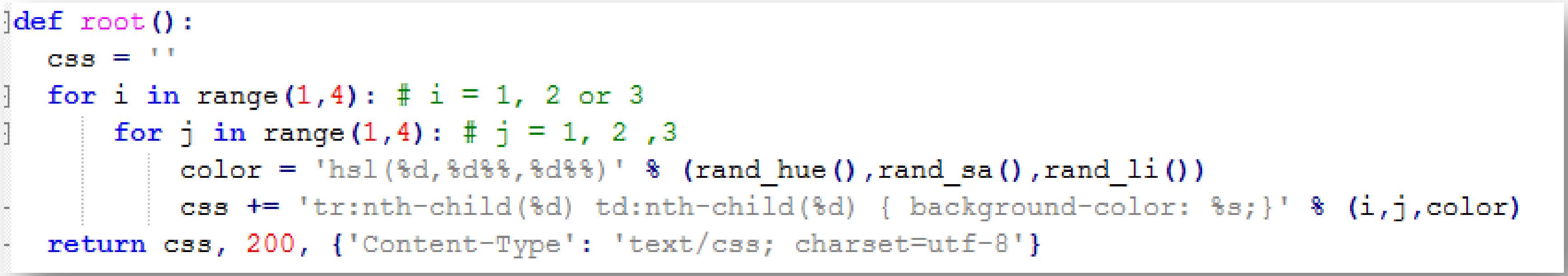
https://en.wikipedia.org/wiki/HSL_and_HSV

Answer: 1. define the function of generating random colours



```python
def rand_hue():
    return random.randint(0,360)
def rand_sa():
    return random.randint(0,100)
def rand_li():
    return random.randint(0,100)
```

2. Iterate on every cell of the 3*3 table (for-loop)

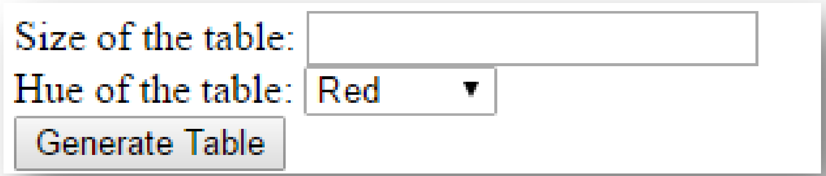


```python
def root():
  css = ''
  for i in range(1,4): # i = 1, 2 or 3
      for j in range(1,4): # j = 1, 2 ,3
          color = 'hsl(%d,%d%%,%d%%)' % (rand_hue(),rand_sa(),rand_li())
          css += 'tr:nth-child(%d) td:nth-child(%d) { background-color: %s;}' % (i,j,color)
  return css, 200, {'Content-Type': 'text/css; charset=utf-8'}
```

THE UNIVERSITY OF MELBOURNE

Exercise 7 – random table

- This time we need to generate html & css contents here



Size of the table:

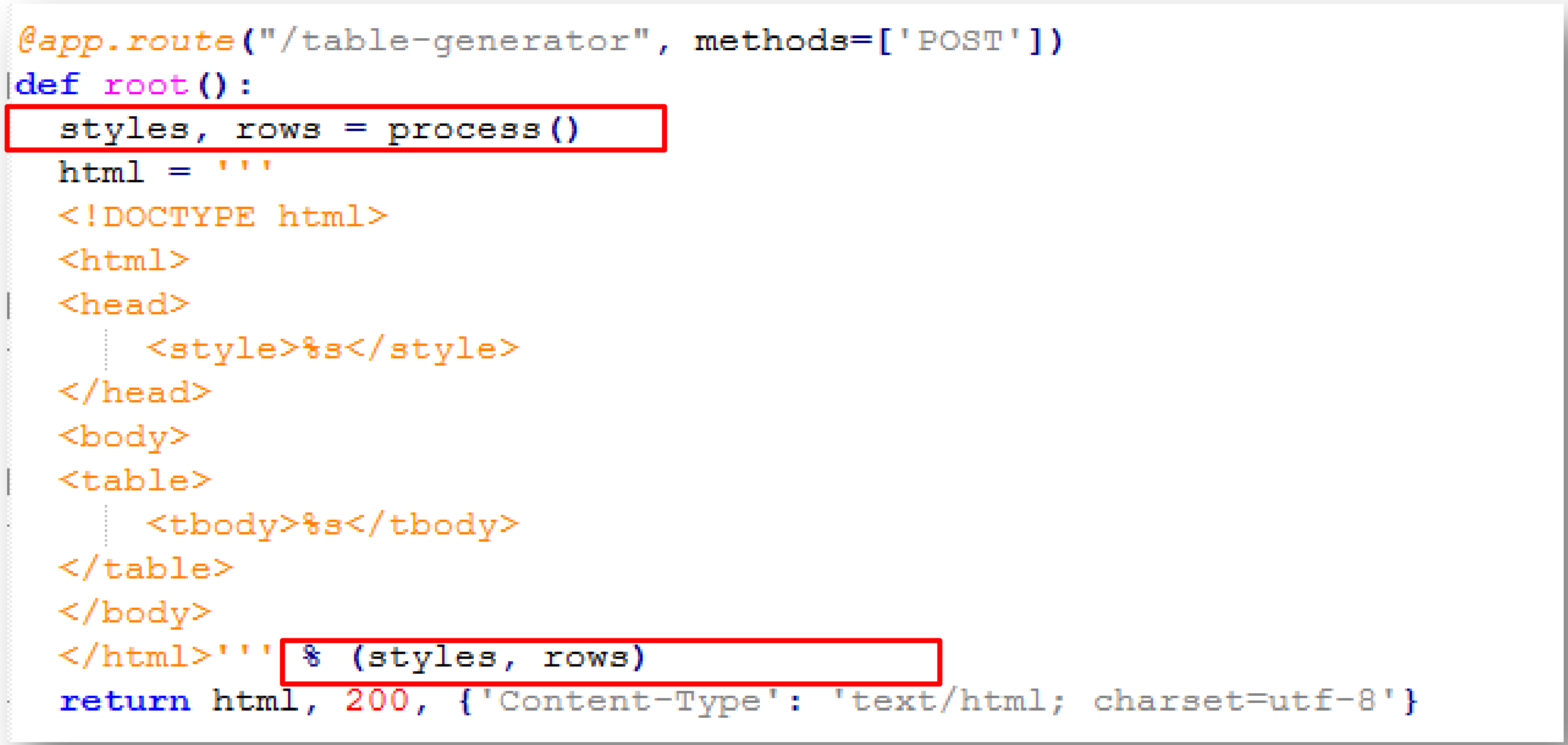
Hue of the table: Red

- Try to add the following functions in the py file.
  - *Size* is specified by 'n' , i.e. n*n table
  - *Hue* is given by the 'drop-down' list (s and l are '100%' by default)
  - Hints:
    - Using embedded CSS
    - You need to "import request from flask" to get contents from 'request' – search 'how to use flask request')
    - You need to learn the "element name" in form.html to transmit the values given by users.

THE UNIVERSITY OF MELBOURNE

Tip

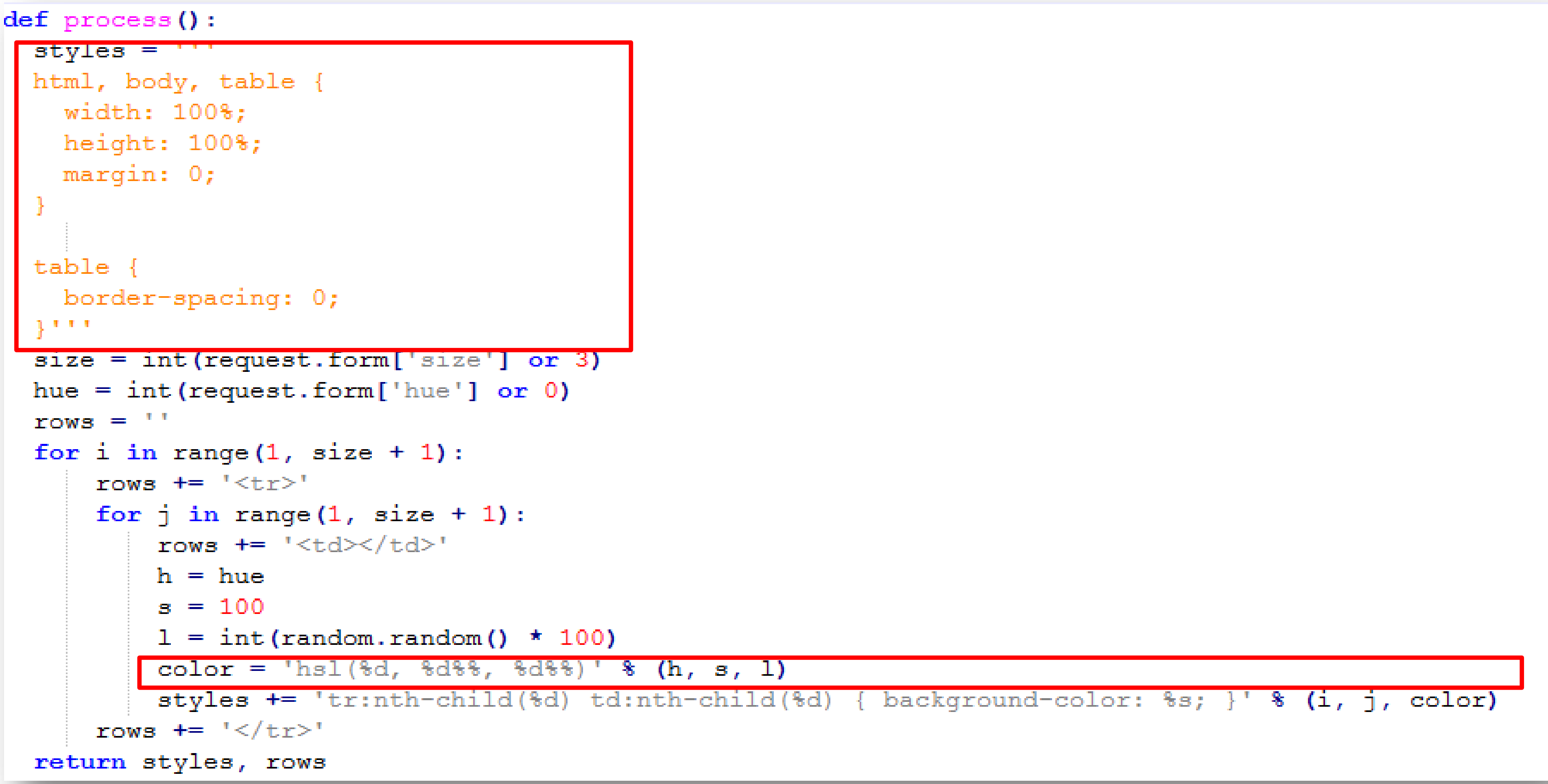
1. HTML skeleton (css and table body are dynamically specified)
2. How to design the process() to achieve the dynamic values?



```
@app.route("/table-generator", methods=['POST'])
def root():
    styles, rows = process()
    html = '''
    <!DOCTYPE html>
    <html>
    <head>
        <style>%s</style>
    </head>
    <body>
    <table>
        <tbody>%s</tbody>
    </table>
    </body>
    </html>''' % (styles, rows)
    return html, 200, {'Content-Type': 'text/html; charset=utf-8'}
```

Solution

Generate ‘rows’ and ‘styles’ for HTML and CSS parts (embedded)



```
def process():
    styles = '''
    html, body, table {
      width: 100%;
      height: 100%;
      margin: 0;
    }

    table {
      border-spacing: 0;
    }'''
    size = int(request.form['size'] or 3)
    hue = int(request.form['hue'] or 0)
    rows = ''
    for i in range(1, size + 1):
        rows += '<tr>'
        for j in range(1, size + 1):
            rows += '<td></td>'
            h = hue
            s = 100
            l = int(random.random() * 100)
            color = 'hsl(%d, %d%%, %d%%)' % (h, s, l)
            styles += 'tr:nth-child(%d) td:nth-child(%d) { background-color: %s; }' % (i, j, color)
        rows += '</tr>'
    return styles, rows
```