Lab 22: Web programming

Apr 24, 2018

Main Event

1. Static HTML

Complete the following tasks using static-HTML. There is no requirement to use Python/Flask to complete this.

- Write a page containing five links to five of your favorite websites.
- Build a 3-by-3 table, where each cell contains a number from one through nine.

2. Dynamic HTML

The remaining exercises require that you are running a web server capable of executing Python code. One of the most straight forward frameworks is Flask:

```
$> conda install flask
```

Flask's quickstart is a good place to get a high-level understanding of the framework.

3. Current time

Build a page that displays the current time. The Python datetime module can help with this:

```
import datetime

i = datetime.datetime.now()

print('Current year', i.year)
```

```
print('Current month', i.month)
print('Current date (day)', i.day)
print('dd/mm/yyyy format', i.day, i.month, i.year)
print('Current hour', i.hour)
print('Current minute', i.minute)
print('Current second', i.second)
print('hh:mm:ss format', i.hour, i.month, i.second)
```

These are just examples so that you get the idea. If you would like to experiment with these commands, it's probably best if you do it in a Python interpreter first, then move the commands over to your web-enabled Python script. You are free to print the date in any manner you wish.

Solution:

```
import flask
import datetime

app = flask.Flask(__name__)

@app.route('/')
def f1():
    now = datetime.datetime.now()

    tm = [
        'Current year: {0}'.format(now.year),
        'Current month {0}'.format(now.month),
        'Current date (day) {0}'.format(now.day),
        'dd/mm/yyyy format {0}/{1}/{2}'.format(now.day, now.month, now
        'Current hour {0}'.format(now.hour),
        'Current minute {0}'.format(now.minute),
        'Current second {0}'.format(now.second),
        'hh:mm:ss format {0}'.format(datetime.datetime.strftime(now, '
    ]

    return '<br>'.join(tm)

app.run(host='0.0.0.0', threaded=True)
```

4. Current calendar

Build a page that displays the month as a table. The current day should be highlighted by making the cell a different color:

```
<td bgcolor="yellow">Yellow cell!</td>
```

A month can be thought of as a matrix with seven columns and a variable number of rows, depending on the month. Thus, building a calendar is essentially just a nested for-loop.

Along with datetime, the calendar module is useful here; as an example:

```
import datetime
import calendar

i = datetime.datetime.now()
c = calendar.monthrange(i.year, i.month)

print(i.month, 'starts on day', c[0])
print(i.month, 'has', c[1], 'days')
```

It is probably easiest to start your calendar from Monday. If you are up for the challenge, make a UAE calendar by making the first day Saturday.

Solution:

```
import flask
import datetime
import calendar

app = flask.Flask(__name__)

@app.route('/')
def f1():
    today = datetime.datetime.now()
    cal = calendar.Calendar(5)

    tbl = [ '<table cellpadding="10">' ]

    # name of the month
    month = today.strftime('%B')
```

```
    row = '<tr><th colspan="7">{0} {1}</th>'.format(month.upper(), toc
    tbl.append(row)

    # days of the week
    row = ''
    for i in cal.iterweekdays():
        row += '<th>{0}</th>'.format(calendar.day_name[i][:3])
    row = '<tr>{0}</tr>'.format(row)
    tbl.append(row)

    # the calendar!
    for week in cal.monthdatescalendar(today.year, today.month):
        tbl.append('<tr>')
        for day in week:
            color = ''
            if day.month == today.month:
                entry = day.day
                if entry == today.day:
                    color = 'bgcolor="yellow"'
            else:
                entry = ' ' # HTML space
            row = '<td {0} align="right">{1}</td>'.format(color, entry
            tbl.append(row)
        tbl.append('</tr>')

    tbl.append('</table>')

    return '\n'.join(tbl)

app.run(host='0.0.0.0', threaded=True)
```

5. Multiplication table

Build a 10-by-10 multiplication table. The upper-most row and left-most column should be the numbers to be multiplied. The corresponding cell should be the result of that multiplication.

Solution:

```
import flask

app = flask.Flask(__name__)
```

```
@app.route('/')
def times():
    rows = 10
    columns = rows

    table = [ '<table cellpadding="10">' ]
    for row in range(rows + 1):
        table.append('<tr>')
        for col in range(columns + 1):
            if not row:
                if not col:
                    entry = ' ' # HTML space
                else:
                    entry = '<b>{0}</b>'.format(col)
            elif not col:
                entry = '<b>{0}</b>'.format(row)
            else:
                entry = row * col
            table.append('<td>{0}</td>'.format(entry))
        table.append('</tr>')
    table.append('</table>')

    return '\n'.join(table)

app.run(host='0.0.0.0', threaded=True)
```

6. ASCII to HTML

Develop a web page that reads in the file list.txt and displays it in nicely formatted HTML. There are two things you must take into account:

1. HTML does not care about ASCII new lines. The `<br>` tag is what it looks for.
2. The ASCII file contains lists, with each bullet denoted with a space-“o”-space combination; so

   ```
   o Item 1
   o Item 2
   o Item 3
   ```

   would be the list in text format. HTML has its own version of bulleted lists, to which the text version needs to be converted.

Solution:

```
import flask

app = flask.Flask(__name__)

@app.route('/')
def ul():
    html = []
    dot = ' o '
    with open('list.txt') as fp:
        inside_list = False
        for i in fp:
            line = i.rstrip()
            if line[:len(dot)] == dot:
                if not inside_list:
                    html.append('<ul>')
                html.append('<li>' + line[len(dot):] + '</li>')
                inside_list = True
            else:
                if inside_list:
                    html.append('</ul>')
                    inside_list = False
                html.append(line)

    return '<br>\n'.join(html)

app.run(host='0.0.0.0', threaded=True)
```
