Python for Automatic Web Page Generation

Presentation to the Southern California OS/2 User Group
Programming SIG
on
February 19, 2005

by
Greg Smith
gsmith@well.com

Introduction

I maintain the web site for the local section of the American Institute of Chemical Engineers at http://www.geocities.com/aiche_sc/. This web site has information on our upcoming meetings. Before 2004, however, it did not have a calendar showing the meeting schedule for the year. This article will describe a Python program that I use on my PC to generate a schedule page to upload to the web site.

The local section of the professional society has two meetings each month. The meeting of the executive committee always occurs on the first Tuesday of the month. The executive committee has met for several years at the same restaurant in Montebello. The general meeting for the members usually takes place on the third Tuesday of the month. However, the Los Angeles local section has a number of joint meetings throughout the year with the Orange County local section. The joint meetings usually take place on the fourth Tuesday of the month; however, other meeting dates are possible if the joint meeting is a plant or university tour.

A Useful Command Line Utility

The open source community has ported a number of traditional Unix utilities to a number of different operating systems. The EMX port of the **cal** command provides a convenient way to begin generation of an HTML web page. The **cal** command with no parameters displays a simple calendar for the current month. For example:

```
$ cal
    December 2004
Su Mo Tu We Th Fr Sa
             1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

With one parameter, the **cal** command will generate a calendar for the year specified. Note that common abbreviations for the year will not produce the expected result – **cal 99**

will produce a calendar for 99 A.D. If you want the calendar for 1999, you must use the command **cal 1999**. For example:

```
$ cal 1999
                                1999

       January               February                March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                1  2       1  2  3  4  5  6       1  2  3  4  5  6
 3  4  5  6  7  8  9    7  8  9 10 11 12 13    7  8  9 10 11 12 13
10 11 12 13 14 15 16   14 15 16 17 18 19 20   14 15 16 17 18 19 20
17 18 19 20 21 22 23   21 22 23 24 25 26 27   21 22 23 24 25 26 27
24 25 26 27 28 29 30   28                     28 29 30 31
31
        April                  May                   June
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                1  2  3                   1            1  2  3  4  5
 4  5  6  7  8  9 10    2  3  4  5  6  7  8    6  7  8  9 10 11 12
11 12 13 14 15 16 17    9 10 11 12 13 14 15   13 14 15 16 17 18 19
18 19 20 21 22 23 24   16 17 18 19 20 21 22   20 21 22 23 24 25 26
25 26 27 28 29 30      23 24 25 26 27 28 29   27 28 29 30
                       30 31
        July                  August                September
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                1  2  3    1  2  3  4  5  6  7               1  2  3  4
 4  5  6  7  8  9 10    8  9 10 11 12 13 14    5  6  7  8  9 10 11
11 12 13 14 15 16 17   15 16 17 18 19 20 21   12 13 14 15 16 17 18
18 19 20 21 22 23 24   22 23 24 25 26 27 28   19 20 21 22 23 24 25
25 26 27 28 29 30 31   29 30 31              26 27 28 29 30

       October               November              December
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
                1  2       1  2  3  4  5  6               1  2  3  4
 3  4  5  6  7  8  9    7  8  9 10 11 12 13    5  6  7  8  9 10 11
10 11 12 13 14 15 16   14 15 16 17 18 19 20   12 13 14 15 16 17 18
17 18 19 20 21 22 23   21 22 23 24 25 26 27   19 20 21 22 23 24 25
24 25 26 27 28 29 30   28 29 30              26 27 28 29 30 31
31
```

With two parameters, the command will generate a calendar for the specified month and year. For example:

```
$ cal 7 2005
      July 2005
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

The simple text calendar produced by the **cal** command served as a basis for the initial schedule page on the web site. One approach would paste the calendar directly into the

web page using "<pre>" and "</pre>" tags to protect the formatting. The appearance of the table can be improved by taking the text calendar and converting it to an HTML table. A lot of cut and paste with a text editor converted the initial **cal** output to HTML for the original version of the web page. However, creating HTML tables can be tedious, boring, painful, and error prone. Consequently, a Python program can be used to automate the process of converting the **cal** command output to a web page with the HTML tables.

Python Hooks

The **os** module in Python provides many functions and data values used to directly access the operating system. The **popen** function can be used to execute an operating system command and open a pipe between the command and the Python program. In this case, the return value of the **popen** function is a file object connected to the pipe. You can read the output of the command using the file handle. For example:

```
>>> import os
>>> filehandle = os.popen("dir c:\\work")
>>> results = filehandle.readlines()
>>> print results
[' Volume in drive C has no label.\n', ' Volume Serial Number is 6441-
0393\n', '\n', ' Directory of c:\\work\n', '\n', '12/28/2004  04:15 PM
<DIR>          .\n', '12/28/2004  04:15 PM    <DIR>          ..\n',
'12/28/2004  04:15 PM             8,178 GenerateSchedule_1.3a.py\n', '
1 File(s)          8,178 bytes\n', '               2 Dir(s)
46,524,391,424 bytes free\n']
>>> print "".join(results)
 Volume in drive C has no label.
 Volume Serial Number is 6441-0393

 Directory of c:\work

12/28/2004  04:15 PM    <DIR>          .
12/28/2004  04:15 PM    <DIR>          ..
12/28/2004  04:15 PM             8,178 GenerateSchedule_1.3a.py
               1 File(s)          8,178 bytes
               2 Dir(s)  46,524,391,424 bytes free

>>>
```

Note that each string returned in the list `results` has a newline appended at the end. When these strings are concatenated with the **join** method, the resulting string can be printed to display the results of the command. In addition, the file object can also be used as an iterator in a for loop or a list comprehension. For example, if we use the file object in a list comprehension we get:

```
>>> import os
>>> filehandle = os.popen("dir c:\\work")
>>> results = [ x for x in filehandle ]
>>> print results
[' Volume in drive C has no label.\n', ' Volume Serial Number is 6441-
```

```
0393\n', '\n', ' Directory of c:\\work\n', '\n', '12/28/2004  04:15 PM
<DIR>          .\n', '12/28/2004  04:15 PM    <DIR>          ..\n',
'12/28/2004  04:15 PM             8,178 GenerateSchedule_1.3a.py\n', '
1 File(s)          8,178 bytes\n', '               2 Dir(s)
46,524,383,232 bytes free\n']
>>>
```

Alternately, the file object can be used in a regular for loop to get:

```
>>> import os
>>> filehandle = os.popen("dir c:\\work")
>>> for x in filehandle:
      print [x]


[' Volume in drive C has no label.\n']
[' Volume Serial Number is 6441-0393\n']
['\n']
[' Directory of c:\\work\n']
['\n']
['12/28/2004  04:15 PM    <DIR>          .\n']
['12/28/2004  04:15 PM    <DIR>          ..\n']
['12/28/2004  04:15 PM             8,178 GenerateSchedule_1.3a.py\n']
['               1 File(s)          8,178 bytes\n']
['               2 Dir(s)  46,524,395,520 bytes free\n']
>>>
```

Processing the Calendar

The **cal** command output is accessed in the Python program with the following series of commands:

```
>>> import os, string
>>> filehandle = os.popen("cal 2 2009")
>>> cal_lines = filehandle.readlines()
>>> print cal_lines
['   February 2009\n', ' S  M Tu  W Th  F  S\n', ' 1  2  3  4  5  6
7\n', ' 8  9 10 11 12 13 14\n', '15 16 17 18 19 20 21\n', '22 23 24 25
26 27 28\n', '\n', '\n']
>>>
```

The first item in the list, `cal_lines[0]`, is the month and year heading for the calendar. The second item in the list, `cal_lines[1]`, has the day of the week abbreviations for the calendar. The remaining items in the list have the data for the month. We can generate the HTML from this list using following code:

```
# First line is name of the month
aline = string.rstrip(cal_lines[0],'\n')
print '<table border="5" summary="Month of ' + aline + '">'
print ' '*2 + '<tr><th colspan="7">' + aline + '</th></tr>'

# Followed by days of the week
print ' '*2 + '<tr>'
```

```
for idx in range(7):
    print ' '*4 + '<td align="right">' +\
          cal_lines[1][idx*3:idx*3+2] + '</td>'
print ' '*2 + '</tr>'

# Followed by four to six weeks worth of data
first_tues = 0
for date_line in cal_lines[2:]:
    if date_line == "\n":  continue   # Nothing in last week--skip it
    print ' '*2 + '<tr>'
    for idx in range(7):
        col = date_line[idx*3:idx*3+2]
        if col == "  " or col == "":
            print ' '*4 + '<td align="right"> </td>'
        else:
            print ' '*4 + '<td align="right">' + col + '</td>'
            if idx == 2 and first_tues == 0:
                first_tues = col
    print ' '*2 + '</tr>'
print '</table>'
```

Executing the above code generates the following HTML:

```
<table border="5" summary="Month of    February 2009">
  <tr><th colspan="7">   February 2009</th></tr>
  <tr>
    <td align="right"> S</td>
    <td align="right"> M</td>
    <td align="right">Tu</td>
    <td align="right"> W</td>
    <td align="right">Th</td>
    <td align="right"> F</td>
    <td align="right"> S</td>
  </tr>
  <tr>
    <td align="right"> 1</td>
    <td align="right"> 2</td>
    <td align="right"> 3</td>
    <td align="right"> 4</td>
    <td align="right"> 5</td>
    <td align="right"> 6</td>
    <td align="right"> 7</td>
  </tr>
  <tr>
    <td align="right"> 8</td>
    <td align="right"> 9</td>
    <td align="right">10</td>
    <td align="right">11</td>
    <td align="right">12</td>
    <td align="right">13</td>
    <td align="right">14</td>
  </tr>
  <tr>
    <td align="right">15</td>
    <td align="right">16</td>
    <td align="right">17</td>
```

```
    <td align="right">18</td>
    <td align="right">19</td>
    <td align="right">20</td>
    <td align="right">21</td>
  </tr>
  <tr>
    <td align="right">22</td>
    <td align="right">23</td>
    <td align="right">24</td>
    <td align="right">25</td>
    <td align="right">26</td>
    <td align="right">27</td>
    <td align="right">28</td>
  </tr>
</table>
```

Making a Complete Web Page

The code above provides a basis for producing a complete web page. If we make the above example code into a function, we can loop through all twelve months of the year. (The function `eachmonth` is defined in the program listing in Appendix A.) As the code loops through each month, the HTML output generated by the function can be embedded in a larger table to control the formatting of the web page as shown in the following figure:

HTML Table for Month 1 (January)	Callouts for the month of January
HTML Table for Month 2 (February)	Callouts for the month of February
...	...
HTML Table for Month 12 (December)	Callouts for the month of December

The HTML wrapper for this table takes the general form:

```
<table summary="Overall wrapper">
  <tr>  <!-- month 1 -->
    <td>  <!-- month in left column -->
        HTML Table for Month 1 (January)
    </td>
    <td>  <!-- call outs in right column -->
        Callouts for the month of January
    </td>
  </tr>
  <tr>  <!-- month 2 -->
    <td>  <!-- month in left column -->
```

```
        HTML Table for Month 2 (February)
    </td>
    <td>  <!-- call outs in right column -->
        Callouts for the month of February
    </td>
  </tr>

          .   .   .

  <tr>  <!-- month 12 -->
    <td>  <!-- month in left column -->
        HTML Table for Month 12 (December)
    </td>
    <td>  <!-- call outs in right column -->
        Callouts for the month of December
    </td>
  </tr>
</table>
```

Processing the callouts for the above table makes use of the `topic` data structure in the program (see Appendix A). This list includes an ordered pair for each month consisting of an offset from the first Tuesday of the month and the text describing the meeting event. (Note: As pointed out in the introduction, the executive committee meeting always occurs on the first Tuesday of the month.) The offset, when added to the date occurring on the first Tuesday of the month, gives the date of the general meeting. In most cases, the regular meeting on the third Tuesday of the month takes place fourteen days after the executive committee meeting. However special events for the regular meeting can take place at other dates requiring the use of the different offsets in the `topic` data structure.

The remaining steps in producing the complete web page include: a) generation of the required HTML at the beginning of the page, and b) generation of the required HTML at he end of the page. This HTML contains constant data defined in the variables `head` and `foot` in the program listing in the Appendix. A print statement before the main loop writes out the constant information in `head` to start the web page. The data in `foot` completes the web page.

There Is More Than One Way to Do It

Developing a program is an evolutionary process. As noted above, the first cut at a schedule page used the **cal** command output and a text editor to generate the HTML. The next cut used throw-away python scripts similar to the code on pages 4-5 with individual months generated by command line execution of the **cal** command piped through the scripts. The tables for the separate months were then combined in a text editor to complete the web page. In the next step of the program evolution, the throw-away script was converted to a subroutine and the piped file using **popen** was then introduced. Finally, the code needed to complete the web page was added.

The evolutionary process, however, sometimes takes you past an obvious programming shortcut. In this case, the standard Python library offers the **calendar** module with routines that can be substituted for the external **cal** command. The **month** function in this module returns a string that is *almost* the same as the output of the **cal** command. The following code illustrates the default behavior of the **month** function:

```
>>> import calendar
>>> cal_lines = calendar.month(2009,2)
>>> cal_lines
'   February 2009\nMo Tu We Th Fr Sa Su\n                   1\n 2  3  4
5  6  7  8\n 9 10 11 12 13 14 15\n16 17 18 19 20 21 22\n23 24 25 26 27
28\n'
>>> print cal_lines
   February 2009
Mo Tu We Th Fr Sa Su
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28

>>>
```

As can be seen here, the **month** function returns a single string instead of the list we obtain with the **popen** function that runs the external **cal** command. Also, the default format for the calendar follows the European convention where Monday starts the week. The following code can be used to create a `cal_lines` list suitable for use use in the `eachmonth` function in the complete program:

```
>>> import calendar, string
>>> calendar.setfirstweekday(calendar.SUNDAY)
>>> cal_lines = string.split(calendar.month(2009,2),'\n')
>>> cal_lines
['   February 2009', 'Su Mo Tu We Th Fr Sa', ' 1  2  3  4  5  6  7', '
8  9 10 11 12 13 14', '15 16 17 18 19 20 21', '22 23 24 25 26 27 28',
'']
>>> cal_lines.pop()
''
>>> cal_lines
['   February 2009', 'Su Mo Tu We Th Fr Sa', ' 1  2  3  4  5  6  7', '
8  9 10 11 12 13 14', '15 16 17 18 19 20 21', '22 23 24 25 26 27 28']
>>>
```

Since the newline character, `'\n'`, was used as the delimiter for **string.split**, the `cal_lines` list returned above differs from the `cal_lines` list returned by the **popen** function. The code above produces strings without the newline at the end; the code discussed earlier in the article produces strings with a terminating newline. Consequently, the tests for the newlines can be removed from the `eachmonth` function. The revised program listing using the **calendar** module is listed in Appendix B.

Conclusion

Python provides a convenient method for automatically generating the schedule page for the professional society web site. The program generates valid HTML and does away with the tedious, boring, painful, and error prone editing of HTML tables. The `topic` data structure at the beginning of the program allows flexibility in scheduling the general meeting. In addition, the description of the topic for the general meeting can be easily updated with a simple text editor.

Since Python is open source, this method of web page generation is independent of the operating system, the web server, and the web browser. This Python program can serve as the basis for generating other web pages containing complex HTML. The generated pages can be easily updated without introducing errors in the HTML.

References

Jenkins, Stephen B. "Offline Programmatic Generation of Web Pages," *;login:*, **20**(5) (October 2004), pp. 6-11.

Pilgrim, Mark. *Dive Into Python.* Apress, Berkeley, CA, 2004. Also available online at http://www.diveintopython.com/.

Appendix A

Program Listing for Generate_Schedule_1.3a.py

```
#!/usr/bin/python
#
#  Generate the schedule page for the Southern California
#  AIChE web site.  This script calls on the Unix 'cal'
#  command to determine the months for the specified year
#  and generates the HTML from the 'cal' output.

import sys, string, os, time

#  Edit the topics with the information as it becomes available.  The
#  entry for each month is a tuple with two values.  The first item in
#  the tuple is the number of days after the first Tuesday for the
#  date of the meeting.  The second entry is the text describing the
#  meeting topic.
year = " 2005"
pad = '\n' + ' '*10
topic = ( ( 21,    # January topic ... third Tuesday
            pad + 'Joint meeting with Orange County Section at Khoury\'s' +
            pad + 'Restaurant in Long Beach, 5:30 P.M.' +
            pad + 'Topic: Dredging in Lake Tahoe and Escaping to Tell' +
            pad + 'the Tale--Environmental and Regulatory Hurdles.'),
          ( 14,    # February topic ... third Tuesday
            pad + 'Local Section General Meeting at the Quiet Cannon' +
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Wasewater Treatment in the San Gabriel Valley.'),
          ( 14,    # March topic ... third Tuesday
            pad + 'Local Section General Meeting at the Quiet Cannon' +
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 21,    # April topic ... third Tuesday
            pad + 'Joint meeting with Orange County Section at Khoury\'s' +
            pad + 'Restaurant in Long Beach, 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 14,    # May topic ... third Tuesday
            pad + 'Local Section General Meeting at the Quiet Cannon' +
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 14,    # June topic ... third Tuesday
            pad + 'Local Section General Meeting at the Quiet Cannon' +
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 21,    # July topic ... third Tuesday
            pad + 'Joint meeting with Orange County Section at Khoury\'s' +
            pad + 'Restaurant in Long Beach, 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 14,    # August topic ... third Tuesday
            pad + 'Local Section General Meeting at the Quiet Cannon' +
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 14,    # September topic ... third Tuesday
            pad + 'Local Section General Meeting at the Quiet Cannon' +
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 21,    # October topic ... third Tuesday
            pad + 'Joint meeting with Orange County Section at Khoury\'s' +
            pad + 'Restaurant in Long Beach, 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 14,    # November topic ... third Tuesday
            pad + 'Local Section General Meeting at the Quiet Cannon' +
```

```
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 14,    # December topic ... third Tuesday
            pad + 'No Local Section General Meeting is currently' +
            pad + 'scheduled for December.') )

#  HTML constant text at the top of the page

head = """<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
  <title>Southern California AIChE Section Schedule</title>
  <meta http-equiv="Content-Type" content="text/html; charset=us-ascii">
  <link rel="stylesheet" type="text/css" href="sitestyle.css">
</head>

<body background="backgd.gif">
  <div id="Menu">
    <img src="aiche.gif" width="139" height="142" alt="(AIChE Logo)">

    <p><a href="index.htm">Home</a></p>

    <p><a href="about.htm">About the Section</a></p>

    <p><a href="message.htm">Chair's Message</a></p>

    <p><a href="meeting.htm">Next Meeting</a></p>

    <p><a href="schedule.htm">Schedule</a></p>

    <p><a href="execcomm.htm">Executive Committee</a></p>

    <p><a href="links.htm">Other Resources</a></p>

    <p><a href="past.htm">&bull; Past Meetings</a></p>

    <p><a href="student.htm">&bull; Student Chapters</a></p>

    <p><a href="society.htm">&bull; Professional</a></p>

    <p><a href="training.htm">&bull; Education</a></p>

    <p><a href="gov.htm">&bull; Government</a></p>

    <p><a href="jobs.htm">&bull; Jobs</a></p>
  </div>

  <div id="Content">
    <h1>American Institute of Chemical Engineers<br>
    Southern California Section</h1>
    <hr>

    <h2>Upcoming Events</h2>
    <hr>

    <table summary="Overall wrapper">"""

#  HTML constant text at the bottom of the page
```

```
foot = """    </table>
    <hr>
    <i>Although we try our best, if any of the information herein is
    incorrect or incomplete, please let me know ASAP at</i> <a href=
    "mailto:gsmith@well.com"><i>WebMaster</i></a> <i>so I can correct it.
    Thanks!</i><br>
    (Revision """ +\
    time.strftime("%m") + "/" + time.strftime("%d") + "/" +\
    time.strftime("%Y") +\
    """) &copy; Southern California Section of AIChE, 2004<br>

    <center>
      <div id="Bottomlink">
        <a href="meeting.htm">&lt; Back</a> | <a href="index.htm">Home</a> |
        <a href="execcomm.htm">Next &gt;</a>
      </div>
    </center>
  </div>
</body>
</html>"""

month_str = ( "January ", "February ", "March ", "April ", "May ",
              "June ", "July ", "August ", "September ", "October ",
              "November ", "December " )

def eachmonth ( cal_lines ):
    """Generate an HTML table from the output of
    the Unix cal command.
    cal_line:  A list containing the outpuf of the cal command.
    return:  The first Tuesday of the month"""

    # First line is name of the month
    aline = string.rstrip(cal_lines[0],'\n')
    print ' '*10 + '<table border="5" summary="Month of ' + aline + '">'
    print ' '*12 + '<tr><th colspan="7">' + aline + '</th></tr>'

    # Followed by days of the week
    print ' '*12 + '<tr>'
    for idx in range(7):
        print ' '*14 + '<td align="right">' +\
              cal_lines[1][idx*3:idx*3+2] + '</td>'
    print ' '*12 + '</tr>'

    # Followed by four to six weeks worth of data
    first_tues = 0
    for date_line in cal_lines[2:]:
        if date_line == "\n":  continue   # Nothing in last week--skip it
        print ' '*12 + '<tr>'
        for idx in range(7):
            col = date_line[idx*3:idx*3+2]
            if col == "  " or col == "":
                print ' '*14 + '<td align="right"> </td>'
            else:
                print ' '*14 + '<td align="right">' + col + '</td>'
                if idx == 2 and first_tues == 0:
                    first_tues = col
        print ' '*12 + '</tr>'
    print ' '*10 + '</table>'
```

```
    return int(first_tues)

#  Redirect the output to the file
sys.stdout = open ( "schedule.htm", "w" )

#  Output the constant HTML at the beginning of the page
print head

#  Generate the twelve months of data
for month in range(12):
    print ' '*6 + '<tr>  <!-- month ' + str(month+1) + ' -->'
    print ' '*8 + '<td>  <!-- month in left column -->'

    # execute the Unix cal command to get the data for the HTML
    handle = os.popen ( "cal " + str(month+1) + year )
    cal_lines =  handle.readlines()

    # convert the output to an HTML table
    first = eachmonth ( cal_lines )
    print ' '*8 + '</td>'
    print ' '*8 + '<td>  <!-- call outs in right column -->'
    print ' '*10 + '<p>' + month_str[month] + str(first) +\
          ': Executive committee meeting'
    print ' '*10 + 'at Carrow\'s Restaurant in Montebello, 6:15 P.M.</p>'
    print ' '*10 + '<p>' + month_str[month] + str(first+topic[month][0]) +\
          ': ' + topic[month][1] + '</p>'
    print ' '*8 + '</td>'
    print ' '*6 + '</tr>'

#  Finish with the constant HTML at the end of the page
print foot
```

Appendix B

Program Listing for Generate_Schedule_1.4.py

```
#!/usr/bin/python
#
#  Generate the schedule page for the Southern California
#  AIChE web site.  This script calls on the Unix 'cal'
#  command to determine the months for the specified year
#  and generates the HTML from the 'cal' output.

import sys, string, calendar, time

#  Edit the topics with the information as it becomes available.  The
#  entry for each month is a tuple with two values.  The first item in
#  the tuple is the number of days after the first Tuesday for the
#  date of the meeting.  The second entry is the text describing the
#  meeting topic.
year = 2005
pad = '\n' + ' '*10
topic = ( ( 21,    # January topic ... fourth Tuesday
            pad + 'Joint meeting with Orange County Section at Khoury\'s' +
            pad + 'Restaurant in Long Beach, 5:30 P.M.' +
            pad + 'Topic: Dredging in Lake Tahoe and Escaping to Tell' +
            pad + 'the Tale--Environmental and Regulatory Hurdles.'),
          ( 16,    # February topic ... third Thursday (special date)
          pad + 'Local Section General Meeting at the Quiet Cannon' +
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Wasewater Treatment in the San Gabriel Valley.'),
          ( 14,    # March topic ... third Tuesday
          pad + 'Local Section General Meeting at the Quiet Cannon' +
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 21,    # April topic ... third Tuesday
            pad + 'Joint meeting with Orange County Section at Khoury\'s' +
            pad + 'Restaurant in Long Beach, 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 14,    # May topic ... third Tuesday
          pad + 'Local Section General Meeting at the Quiet Cannon' +
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
          pad + 'Topic: Open.'),
          ( 14,    # June topic ... third Tuesday
          pad + 'Local Section General Meeting at the Quiet Cannon' +
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 21,    # July topic ... third Tuesday
            pad + 'Joint meeting with Orange County Section at Khoury\'s' +
            pad + 'Restaurant in Long Beach, 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 14,    # August topic ... third Tuesday
          pad + 'Local Section General Meeting at the Quiet Cannon' +
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 14,    # September topic ... third Tuesday
          pad + 'Local Section General Meeting at the Quiet Cannon' +
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 21,    # October topic ... third Tuesday
            pad + 'Joint meeting with Orange County Section at Khoury\'s' +
            pad + 'Restaurant in Long Beach, 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 14,    # November topic ... third Tuesday
          pad + 'Local Section General Meeting at the Quiet Cannon' +
```

```
            pad + 'Restaurant (Montebello Country Club), 5:30 P.M.' +
            pad + 'Topic: Open.'),
          ( 14,    # December topic ... third Tuesday
            pad + 'No Local Section General Meeting is currently' +
            pad + 'scheduled for December.') )

#  HTML constant text at the top of the page

head = """<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
  <title>Southern California AIChE Section Schedule</title>
  <meta http-equiv="Content-Type" content="text/html; charset=us-ascii">
  <link rel="stylesheet" type="text/css" href="sitestyle.css">
</head>

<body background="backgd.gif">
  <div id="Menu">
    <img src="aiche.gif" width="139" height="142" alt="(AIChE Logo)">

    <p><a href="index.htm">Home</a></p>

    <p><a href="about.htm">About the Section</a></p>

    <p><a href="message.htm">Chair's Message</a></p>

    <p><a href="meeting.htm">Next Meeting</a></p>

    <p><a href="schedule.htm">Schedule</a></p>

    <p><a href="execcomm.htm">Executive Committee</a></p>

    <p><a href="links.htm">Other Resources</a></p>

    <p><a href="past.htm">&bull; Past Meetings</a></p>

    <p><a href="student.htm">&bull; Student Chapters</a></p>

    <p><a href="society.htm">&bull; Professional</a></p>

    <p><a href="training.htm">&bull; Education</a></p>

    <p><a href="gov.htm">&bull; Government</a></p>

    <p><a href="jobs.htm">&bull; Jobs</a></p>
  </div>

  <div id="Content">
    <h1>American Institute of Chemical Engineers<br>
    Southern California Section</h1>
    <hr>

    <h2>Upcoming Events</h2>
    <hr>

    <table summary="Overall wrapper">"""

#  HTML constant text at the bottom of the page
```

```
foot = """    </table>
    <hr>
    <i>Although we try our best, if any of the information herein is
    incorrect or incomplete, please let me know ASAP at</i> <a href=
    "mailto:gsmith@well.com"><i>WebMaster</i></a> <i>so I can correct it.
    Thanks!</i><br>
    (Revision """ +\
    time.strftime("%m") + "/" + time.strftime("%d") + "/" +\
    time.strftime("%y") +\
    """) &copy; Southern California Section of AIChE, 2005<br>

    <center>
      <div id="Bottomlink">
        <a href="meeting.htm">&lt; Back</a> | <a href="index.htm">Home</a> |
        <a href="execcomm.htm">Next &gt;</a>
      </div>
    </center>
  </div>
</body>
</html>"""

month_str = ( "January ", "February ", "March ", "April ", "May ",
              "June ", "July ", "August ", "September ", "October ",
              "November ", "December " )

def eachmonth ( cal_lines ):
    """Generate an HTML table from the output of
    the Unix cal command.
    cal_line:  A list containing the outpuf of the cal command.
    return:  The first Tuesday of the month"""

    # First line is name of the month
    print ' '*10 + '<table border="5" summary="Month of ' + cal_lines[0] + '">'
    print ' '*12 + '<tr><th colspan="7">' + cal_lines[0] + '</th></tr>'

    # Followed by days of the week
    print ' '*12 + '<tr>'
    for idx in range(7):
        print ' '*14 + '<td align="right">' +\
              cal_lines[1][idx*3:idx*3+2] + '</td>'
    print ' '*12 + '</tr>'

    # Followed by four to six weeks worth of data
    first_tues = 0
    for date_line in cal_lines[2:]:
        print ' '*12 + '<tr>'
        for idx in range(7):
            col = date_line[idx*3:idx*3+2]
            if col == "  " or col == "":
                print ' '*14 + '<td align="right"> </td>'
            else:
                print ' '*14 + '<td align="right">' + col + '</td>'
                if idx == 2 and first_tues == 0:
                    first_tues = col
        print ' '*12 + '</tr>'
    print ' '*10 + '</table>'
    return int(first_tues)
```

```
#  Redirect the output to the file
sys.stdout = open ( "schedule.htm", "w" )

#  Output the constant HTML at the beginning of the page
print head

#  Generate the twelve months of data
for month in range(12):
    print ' '*6 + '<tr>  <!-- month ' + str(month+1) + ' -->'
    print ' '*8 + '<td>  <!-- month in left column -->'

    # call the calendar module to get the data for the HTML
    calendar.setfirstweekday(calendar.SUNDAY)
    cal_lines = string.split(calendar.month(year,month+1),"\n")
    cal_lines.pop()  # throw away last blank line

    # convert the output to an HTML table
    first = eachmonth ( cal_lines )
    print ' '*8 + '</td>'
    print ' '*8 + '<td>  <!-- call outs in right column -->'
    print ' '*10 + '<p>' + month_str[month] + str(first) +\
          ': Executive committee meeting'
    print ' '*10 + 'at Carrow\'s Restaurant in Montebello, 6:15 P.M.</p>'
    print ' '*10 + '<p>' + month_str[month] + str(first+topic[month][0]) +\
          ': ' + topic[month][1] + '</p>'
    print ' '*8 + '</td>'
    print ' '*6 + '</tr>'

#  Finish with the constant HTML at the end of the page
print foot
```

Appendix C

Cascading Stylesheet sitestyle.css

```
body {
      margin:0px;
      padding:0px;
      font-family:verdana, arial, helvetica, sans-serif;
      background-color:white;
      }

h1 {
      font-size: 140%;
      text-align: center;
      }

h2 {
      font-size: 120%;
      text-align: center;
      }

h3, h4, h5, h6 {
      font-size: 100%;
      }

a {
      color:#09c;
      text-decoration:none;
      font-weight:600;
      font-family:verdana, arial, helvetica, sans-serif;
      }

a:link {
      color:#09c;
      }


a:visited {
      color:#666;
      }

a:hover {
      background-color:#fff;
      text-decoration: underline;
      }

#Content {
      margin:0px 10px 10px 170px;
      padding:10px;
      }

#Menu {
      position:absolute;
      top:0px;
      left:0px;
      width:90px;
      padding:10px;
      background-color:#0ff;
```

```
      border:none;
      line-height:17px;
      }

#Bottomlink {
      padding:10px;
      width:250px;
      text-align:center;
      background-color:#0ff;
      border:3px solid #999;
      line-height:17px;
      }

#ECTable {
      padding:10px;
      bottom-margin:20px;
      width:425px;
      text-align:center;
      border:3px solid #999;
      line-height:17px;
      }

.EC {
      margin-top: 50px;
      margin-bottom: 50px;
      }

.ECPic {
      float: left;
        width: 80px;
        height: 80px;
      }

.ECEntry1 {
      float: right;
      width: 325px;
      font-weight: 600;
        left-margin: 100px;
      border-width: 3px 3px 0px 3px;
        border-style: solid;
        border-color: #999;
      }

.ECEntry2 {
      float: right;
      width: 325px;
        left-margin: 100px;
      border-width: 3px 3px 0px 3px;
        border-style: solid;
        border-color: #999;
      }

.ECEntry3 {
      float: right;
```

```
width: 325px;
  left-margin: 100px;
border-width: 3px 3px 3px 3px;
  border-style: solid;
  border-color: #999;
}
```