Python for Automatic Web Page Generation

Greg Smith
gsmith@well.com

Presentation to the Southern California
OS/2 User Group
Programming SIG

February 19, 2005

Introduction

- Needed schedule page for web site
  http://www.geocities.com/aiche_sc/

- Two meetings each month
  Executive committee always meets on the first Tuesday of the month
  General meeting usually meets on the third or fourth Tuesday of the month

Programming for Web Page Generation

- Web page generated offline
  - Occasional web page updates
  - Site has limited CGI programming ability
- Python
  - Automates tedious HTML layout
- Unix utility: cal
  - Don't reinvent the wheel—Steal the blueprints

The 'cal' Command

CAL(1) BSD General Commands Manual CAL(1)

NAME
cal - displays a calendar

SYNOPSIS
cal [-jy] [[month] year]

DESCRIPTION
cal displays a simple calendar. If arguments are not specified, the current month is displayed. The options are as follows:

-j Display julian dates (days one-based, numbered from January 1).

-y Display a calendar for the current year.

A single parameter specifies the year (1 - 9999) to be displayed; note the year must be fully specified: ``cal 89'' will not display a calendar for 1989. Two parameters denote the month (1 - 12) and year. If no parameters are specified, the current month's calendar is displayed.

A year starts on Jan 1.

The 'cal' Command (continued)

- Fixed format output
  - First line has name of month and year
  - Second line has days of week with 1 and 2 letter abbreviations
  - Third through eighth lines have the days of the month
- Specific months can be generated as needed

Sample 'cal' Output

```
$ 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
```

Python Hooks
Os Module Popen

popen(*command*[, *mode*[, *bufsize*]])

Open a pipe to or from *command*. The return value is an open file object connected to the pipe, which can be read or written depending on whether *mode* is `'r'` (default) or `'w'`. The *bufsize* argument has the same meaning as the corresponding argument to the built-in `open()` function. The exit status of the command (encoded in the format specified for `wait()`) is available as the return value of the `close()` method of the file object, except that when the exit status is zero (termination without errors), `None` is returned. Availability: Unix, Windows.

Popen Example 1

```
>>> 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
```

Popen Example 2

```
>>> 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']
```

Making the HTML Table

- Use popen to run the 'cal' command
- Strip newline characters where necessary
- First line of 'cal' output spans 7 columns
- Use slices to extract 7 days of the week
- Use slices to extract data for calendar
  - Replace blanks with
  - Skip over blank lines at end

Code Sample 1

```
import os, string
filehandle = os.popen("cal 2 2009")
cal_lines = filehandle.readlines()

# 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>'
```

Code Sample 2

```
# 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>'
```

Generated 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>

  ...

</table>
```

Completing the Web Page

- Generate HTML heading
- Data structure for monthly call outs
- Loop through 12 months
  - Embed 'cal' HTML in wrapper table
  - Embed call outs in wrapper table
- Generate HTML footing

Page Layout

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

HTML for Wrapper

```
<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>

          .  .  .

</table>
```

Code Review

- Variable 'topic' data structure: offset, text
- Variable 'head' with constant HTML
- Variable 'foot' with constant HTML
- Variable month_str
- Function def for eachmonth (cal_lines)
- File redirection
- Output 'head' with constant HTML
- Main loop
- Output 'foot' with constant HTML

There Is More Than One Way to Do It: Code Evolution

- 'cal' command + text editor → web page
- 'cal' command + throw-away scripts + text editor → web page
- 'cal' command + Python program → web page
- Python program with standard library calls → web page

Python Standard Library: The Calendar Module

month(*theyear, themonth*[, *w*[, *l*]])

Returns a month's calendar in a multi-line string. If *w* is provided, it specifies the width of the date columns, which are centered. If *l* is given, it specifies the number of lines that each week will use. Depends on the first weekday as set by setfirstweekday(). New in version 2.0

Month Example

```
>>> 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

>>>
```

Month Example 2

```
>>> 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']
>>>
```

The 'Pure' Python Solution

- Does not need the external 'cal' command
- The `calendar` module has the `month` function which returns the same information as 'cal' command
- The `month` function default behavior follows European conventions
- Newlines '\n' go away when used to parse the string returned by `calendar.month`

Conclusions

- The Python program generates valid HTML
- One place to update meeting information in the 'topic' data structure
- Open source
  - Independent of OS
  - Independent of Web service
  - Independent of Web Browser