The sgmllib module

This module provides an basic SGML parser. It works pretty much like the **xmllib** parser, but is less restrictive (and less complete).

Like in **xmllib**, this parser calls methods in itself to deal with things like start tags, data sections, end tags, and entities. If you're only interested in a few tags, you can define special **start** and **end** methods:

Example: Using the sgmllib module to extract the title element

```
# File: sgmllib-example-1.py

import sgmllib
import string

class FoundTitle(Exception):
    pass

class ExtractTitle(sgmllib.SGMLParser):

    def __init__(self, verbose=0):
        sgmllib.SGMLParser.__init__(self, verbose)
        self.title = self.data = None

    def handle_data(self, data):
        if self.data is not None:
            self.data.append(data)

    def start_title(self, attrs):
        self.data = []

    def end_title(self):
        self.title = string.join(self.data, "")
        raise FoundTitle # abort parsing!

def extract(file):
    # extract title from an HTML/SGML stream
    p = ExtractTitle()
    try:
        while 1:
            # read small chunks
            s = file.read(512)
            if not s:
                break
            p.feed(s)
        p.close()
    except FoundTitle:
        return p.title
    return None

#
# try it out

print "html", "=>", extract(open("samples/sample.htm"))
print "sgml", "=>", extract(open("samples/sample.sgm"))


html => A Title.
sgml => Quotations
```

To handle all tags, overload the **unknown_starttag** and **unknown_endtag** methods instead:

Example: Using the sgmllib module to format an SGML document

```
# File: sgmllib-example-2.py

import sgmllib
```

```
import cgi, sys

class PrettyPrinter(sgmllib.SGMLParser):
    # a simple SGML pretty printer

    def __init__(self):
        # initialize base class
        sgmllib.SGMLParser.__init__(self)
        self.flag = 0

    def newline(self):
        # force newline, if necessary
        if self.flag:
            sys.stdout.write("\n")
        self.flag = 0

    def unknown_starttag(self, tag, attrs):
        # called for each start tag

        # the attrs argument is a list of (attr, value)
        # tuples. convert it to a string.
        text = ""
        for attr, value in attrs:
            text = text + " %s='%s'" % (attr, cgi.escape(value))

        self.newline()
        sys.stdout.write("<%s%s>\n" % (tag, text))

    def handle_data(self, text):
        # called for each text section
        sys.stdout.write(text)
        self.flag = (text[-1:] != "\n")

    def handle_entityref(self, text):
        # called for each entity
        sys.stdout.write("&%s;" % text)

    def unknown_endtag(self, tag):
        # called for each end tag
        self.newline()
        sys.stdout.write("<%s>" % tag)

#
# try it out

file = open("samples/sample.sgm")

p = PrettyPrinter()
p.feed(file.read())
p.close()
```

```
<chapter>
<title>
Quotations
<title>
<epigraph>
<attribution>
eff-bot, June 1997
<attribution>
<para>
<quote>
Nobody expects the Spanish Inquisition! Amongst
our weaponry are such diverse elements as fear, surprise,
ruthless efficiency, and an almost fanatical devotion to
Guido, and nice red uniforms &mdash; oh, damn!
<quote>
<para>
<epigraph>
<chapter>
```

The following example checks if an SGML document is “well-formed”, in the XML sense. In a well-formed document, all elements are properly nested, and there’s one end tag for each start

tag.

To check this, we simply keep a list of open tags, and check that each end tag closes a matching start tag, and that there are no open tags when we reach the end of the document.

Example: Using the sgmllib module to check if an SGML document is well-formed

```
# File: sgmllib-example-3.py

import sgmllib

class WellFormednessChecker(sgmllib.SGMLParser):
    # check that an SGML document is 'well formed'
    # (in the XML sense).

    def __init__(self, file=None):
        sgmllib.SGMLParser.__init__(self)
        self.tags = []
        if file:
            self.load(file)

    def load(self, file):
        while 1:
            s = file.read(8192)
            if not s:
                break
            self.feed(s)
        self.close()

    def close(self):
        sgmllib.SGMLParser.close(self)
        if self.tags:
            raise SyntaxError, "start tag %s not closed" % self.tags[-1]

    def unknown_starttag(self, start, attrs):
        self.tags.append(start)

    def unknown_endtag(self, end):
        start = self.tags.pop()
        if end != start:
            raise SyntaxError, "end tag %s does't match start tag %s" %\
                  (end, start)

try:
    c = WellFormednessChecker()
    c.load(open("samples/sample.htm"))
except SyntaxError:
    raise # report error
else:
    print "document is wellformed"


Traceback (innermost last):
...
SyntaxError: end tag head does't match start tag meta
```

Finally, here's a class that allows you to filter HTML and SGML documents. To use this class, create your own base class, and implement the **start** and **end** methods.

Example: Using the sgmllib module to filter SGML documents

```
# File: sgmllib-example-4.py

import sgmllib
import cgi, string, sys

class SGMLFilter(sgmllib.SGMLParser):
    # sgml filter.  override start/end to manipulate
    # document elements
```

```
    def __init__(self, outfile=None, infile=None):
        sgmllib.SGMLParser.__init__(self)
        if not outfile:
            outfile = sys.stdout
        self.write = outfile.write
        if infile:
            self.load(infile)

    def load(self, file):
        while 1:
            s = file.read(8192)
            if not s:
                break
            self.feed(s)
        self.close()

    def handle_entityref(self, name):
        self.write("&%s;" % name)

    def handle_data(self, data):
        self.write(cgi.escape(data))

    def unknown_starttag(self, tag, attrs):
        tag, attrs = self.start(tag, attrs)
        if tag:
            if not attrs:
                self.write("<%s>" % tag)
            else:
                self.write("<%s" % tag)
                for k, v in attrs:
                    self.write(" %s=%s" % (k, repr(v)))
                self.write(">")

    def unknown_endtag(self, tag):
        tag = self.end(tag)
        if tag:
            self.write("</%s>" % tag)

    def start(self, tag, attrs):
        return tag, attrs # override

    def end(self, tag):
        return tag # override

class Filter(SGMLFilter):

    def fixtag(self, tag):
        if tag == "em":
            tag = "i"
        if tag == "string":
            tag = "b"
        return string.upper(tag)

    def start(self, tag, attrs):
        return self.fixtag(tag), attrs

    def end(self, tag):
        return self.fixtag(tag)

c = Filter()
c.load(open("samples/sample.htm"))
```
