StringIO and cStringIO – Work with text buffers using file-like API

Purpose: Work with text buffers using file-like API

Available In: StringIO: 1.4, cStringIO: 1.5

`StringIO` provides a convenient means of working with text in memory using the file API (read, write. etc.). There are two separate implementations. The `cStringIO` version is written in C for speed, while `StringIO` is written in Python for portability. Using `cStringIO` to build large strings can offer performance savings over some other string conctatenation techniques.

Example

Here are some pretty standard, simple, examples of using `StringIO` buffers:

```
# Find the best implementation available on this platform
try:
    from cStringIO import StringIO
except:
    from StringIO import StringIO

# Writing to a buffer
output = StringIO()
output.write('This goes into the buffer. ')
print >>output, 'And so does this.'

# Retrieve the value written
print output.getvalue()

output.close() # discard buffer memory

# Initialize a read buffer
input = StringIO('Inital value for read buffer')

# Read from the buffer
print input.read()
```

This example uses **`read()`**, but the `readline()` and **`readlines()`** methods are also available. The `StringIO` class also provides a **`seek()`** method so it is possible to jump

Page Contents

Navigation

Table of Contents
Previous: string – Working with text
Next: re – Regular Expressions

This Page

Show Source

Examples

The output from all the example programs from PyMOTW has been generated with Python 2.7.8, unless otherwise noted. Some of the features described here may not be available in earlier versions of Python.

If you are looking for examples that work under Python 3, please refer to the PyMOTW-3 section of the site.

around in a buffer while reading, which can be useful for rewinding if you are using some sort of look-ahead parsing algorithm.

```
$ python stringio_examples.py

This goes into the buffer. And so does this.

Inital value for read buffer
```

Real world applications of `StringIO` include a web application stack where various parts of the stack may add text to the response, or testing the output generated by parts of a program which typically write to a file.

The application we are building at work includes a shell scripting interface in the form of several command line programs. Some of these programs are responsible for pulling data from the database and dumping it on the console (either to show the user, or so the text can serve as input to another command). The commands share a set of formatter plugins to produce a text representation of an object in a variety of ways (XML, bash syntax, human readable, etc.). Since the formatters normally write to standard output, testing the results would be a little tricky without the StringIO module. Using StringIO to intercept the output of the formatter gives us an easy way to collect the output in memory to compare against expected results.

See also:

StringIO

Standard library documentation for this module.

The StringIO module ::: www.effbot.org

effbot's examples with StringIO

Efficient String Concatenation in Python

Examines various methods of combining strings and their relative merits.