The zipfile module

(New in 2.0) This module allows you to read and write files in the popular ZIP archive format.

Listing the contents

To list the contents of an existing archive, you can use the **namelist** and **infolist** methods. The former returns a list of filenames, the latter a list of **ZipInfo** instances.

Example: Using the zipfile module to list files in a ZIP file

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

import zipfile

file = zipfile.ZipFile("samples/sample.zip", "r")

# list filenames
for name in file.namelist():
    print name,
print

# list file information
for info in file.infolist():
    print info.filename, info.date_time, info.file_size


$ python zipfile-example-1.py
sample.txt sample.jpg
sample.txt (1999, 9, 11, 20, 11, 8) 302
sample.jpg (1999, 9, 18, 16, 9, 44) 4762
```

Reading data from a ZIP file

To read data from an archive, simply use the **read** method. It takes a filename as an argument, and returns the data as a string.

Example: Using the zipfile module to read data from a ZIP file

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

import zipfile

file = zipfile.ZipFile("samples/sample.zip", "r")

for name in file.namelist():
    data = file.read(name)
    print name, len(data), repr(data[:10])


$ python zipfile-example-2.py
sample.txt 302 'We will pe'
sample.jpg 4762 '\377\330\377\340\000\020JFIF'
```

Writing data to a ZIP file

Adding files to an archive is easy. Just pass the file name, and the name you want that file to have in the archive, to the **write** method.

The following script creates a ZIP file containing all files in the **samples** directory.

Example: Using the zipfile module to store files in a ZIP file

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

```
import zipfile
import glob, os

# open the zip file for writing, and write stuff to it

file = zipfile.ZipFile("test.zip", "w")

for name in glob.glob("samples/*"):
    file.write(name, os.path.basename(name), zipfile.ZIP_DEFLATED)

file.close()

# open the file again, to see what's in it

file = zipfile.ZipFile("test.zip", "r")
for info in file.infolist():
    print info.filename, info.date_time, info.file_size, info.compress_size


$ python zipfile-example-3.py
sample.wav (1999, 8, 15, 21, 26, 46) 13260 10985
sample.jpg (1999, 9, 18, 16, 9, 44) 4762 4626
sample.au (1999, 7, 18, 20, 57, 34) 1676 1103
...
```

The third, optional argument to the **write** method controls what compression method to use. Or rather, it controls whether data should be compressed at all. The default is **zipfile.ZIP_STORED**, which stores the data in the archive without any compression at all. If the **zlib** module is installed, you can also use **zipfile.ZIP_DEFLATED**, which gives you "deflate" compression.

The **zipfile** module also allows you to add strings to the archive. However, adding data from a string is a bit tricky; instead of just passing in the archive name and the data, you have to create a **ZipInfo** instance and configure it correctly. Here's a simple example:

Example: Using the zipfile module to store strings in a ZIP file

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

import zipfile
import glob, os, time

file = zipfile.ZipFile("test.zip", "w")

now = time.localtime(time.time())[:6]

for name in ("life", "of", "brian"):
    info = zipfile.ZipInfo(name)
    info.date_time = now
    info.compress_type = zipfile.ZIP_DEFLATED
    file.writestr(info, name*1000)

file.close()

# open the file again, to see what's in it

file = zipfile.ZipFile("test.zip", "r")

for info in file.infolist():
    print info.filename, info.date_time, info.file_size, info.compress_size


$ python zipfile-example-4.py
life (2000, 12, 1, 0, 12, 1) 4000 26
of (2000, 12, 1, 0, 12, 1) 2000 18
brian (2000, 12, 1, 0, 12, 1) 5000 31
```