zipfile — ZIP Archive Access

Purpose: Read and write ZIP archive files.

The `zipfile` module can be used to manipulate ZIP archive files, the format popularized by the PC program PKZIP.

Testing ZIP Files

The `is_zipfile()` function returns a boolean indicating whether or not the filename passed as an argument refers to a valid ZIP archive.

```
# zipfile_is_zipfile.py

import zipfile

for filename in ['README.txt', 'example.zip',
                 'bad_example.zip', 'notthere.zip']:
    print('{:>15}  {}'.format(
        filename, zipfile.is_zipfile(filename)))
```

If the file does not exist at all, `is_zipfile()` returns `False`.

```
$ python3 zipfile_is_zipfile.py

     README.txt  False
    example.zip  True
bad_example.zip  False
   notthere.zip  False
```

Reading Metadata from an Archive

Use the `ZipFile` class to work directly with a ZIP archive. It supports methods for reading data about existing archives as well as modifying the archives by adding additional files.

```
# zipfile_namelist.py

import zipfile

with zipfile.ZipFile('example.zip', 'r') as zf:
    print(zf.namelist())
```

The `namelist()` method returns the names of the files in an existing archive.

```
$ python3 zipfile_namelist.py

['README.txt']
```

The list of names is only part of the information available from the archive, though. To access all of the metadata about the ZIP contents, use the `infolist()` or `getinfo()` methods.

```
# zipfile_infolist.py

import datetime
import zipfile


def print_info(archive_name):
    with zipfile.ZipFile(archive_name) as zf:
        for info in zf.infolist():
            print(info.filename)
            print('  Comment:', info.comment)
```

```
            print('  Comment     :', info.comment)
            mod_date = datetime.datetime(*info.date_time)
            print('  Modified    :', mod_date)
            if info.create_system == 0:
                system = 'Windows'
            elif info.create_system == 3:
                system = 'Unix'
            else:
                system = 'UNKNOWN'
            print('  System      :', system)
            print('  ZIP version :', info.create_version)
            print('  Compressed  :', info.compress_size, 'bytes')
            print('  Uncompressed:', info.file_size, 'bytes')
            print()


if __name__ == '__main__':
    print_info('example.zip')
```

There are additional fields other than those printed here, but deciphering the values into anything useful requires careful reading of the *PKZIP Application Note* with the ZIP file specification.

```
$ python3 zipfile_infolist.py

README.txt
  Comment     : b''
  Modified    : 2010-11-15 06:48:02
  System      : Unix
  ZIP version : 30
  Compressed  : 65 bytes
  Uncompressed: 76 bytes
```

If the name of the archive member is known in advance, its `ZipInfo` object can be retrieved directly with `getinfo()`.

```
# zipfile_getinfo.py

import zipfile

with zipfile.ZipFile('example.zip') as zf:
    for filename in ['README.txt', 'notthere.txt']:
        try:
            info = zf.getinfo(filename)
        except KeyError:
            print('ERROR: Did not find {} in zip file'.format(
                filename))
        else:
            print('{} is {} bytes'.format(
                info.filename, info.file_size))
```

If the archive member is not present, `getinfo()` raises a `KeyError`.

```
$ python3 zipfile_getinfo.py

README.txt is 76 bytes
ERROR: Did not find notthere.txt in zip file
```

Extracting Archived Files From an Archive

To access the data from an archive member, use the `read()` method, passing the member's name.

```
# zipfile_read.py

import zipfile

with zipfile.ZipFile('example.zip') as zf:
    for filename in ['README.txt', 'notthere.txt']:
        try:
            data = zf.read(filename)
        except KeyError:
```

```
        except KeyError:
            print('ERROR: Did not find {} in zip file'.format(
                filename))
        else:
            print(filename, ':')
            print(data)
        print()
```

The data is automatically decompressed, if necessary.

```
$ python3 zipfile_read.py

README.txt :
b'The examples for the zipfile module use \nthis file and exampl
e.zip as data.\n'

ERROR: Did not find notthere.txt in zip file
```

Creating New Archives

To create a new archive, instantiate the `ZipFile` with a mode of `'w'`. Any existing file is truncated and a new archive is started. To add files, use the `write()` method.

```
# zipfile_write.py
```

```
from zipfile_infolist import print_info
import zipfile

print('creating archive')
with zipfile.ZipFile('write.zip', mode='w') as zf:
    print('adding README.txt')
    zf.write('README.txt')

print()
print_info('write.zip')
```

By default, the contents of the archive are not compressed.

```
$ python3 zipfile_write.py

creating archive
adding README.txt

README.txt
  Comment     : b''
  Modified    : 2016-08-07 13:31:24
  System      : Unix
  ZIP version : 20
  Compressed  : 76 bytes
  Uncompressed: 76 bytes
```

To add compression, the zlib module is required. If zlib is available, the compression mode for individual files or for the archive as a whole can be set using `zipfile.ZIP_DEFLATED`. The default compression mode is `zipfile.ZIP_STORED`, which adds the input data to the archive without compressing it.

```
# zipfile_write_compression.py
```

```
from zipfile_infolist import print_info
import zipfile
try:
    import zlib
    compression = zipfile.ZIP_DEFLATED
except (ImportError, AttributeError):
    compression = zipfile.ZIP_STORED

modes = {
    zipfile.ZIP_DEFLATED: 'deflated',
    zipfile.ZIP_STORED: 'stored',
}
```

```
print('creating archive')
with zipfile.ZipFile('write_compression.zip', mode='w') as zf:
    mode_name = modes[compression]
    print('adding README.txt with compression mode', mode_name)
    zf.write('README.txt', compress_type=compression)

print()
print_info('write_compression.zip')
```

This time, the archive member is compressed.

```
$ python3 zipfile_write_compression.py

creating archive
adding README.txt with compression mode deflated

README.txt
  Comment     : b''
  Modified    : 2016-08-07 13:31:24
  System      : Unix
  ZIP version : 20
  Compressed  : 65 bytes
  Uncompressed: 76 bytes
```

Using Alternate Archive Member Names

Pass an `arcname` value to `write()` to add a file to an archive using a name other than the original filename.

```
# zipfile_write_arcname.py

from zipfile_infolist import print_info
import zipfile

with zipfile.ZipFile('write_arcname.zip', mode='w') as zf:
    zf.write('README.txt', arcname='NOT_README.txt')

print_info('write_arcname.zip')
```

There is no sign of the original filename in the archive.

```
$ python3 zipfile_write_arcname.py

NOT_README.txt
  Comment     : b''
  Modified    : 2016-08-07 13:31:24
  System      : Unix
  ZIP version : 20
  Compressed  : 76 bytes
  Uncompressed: 76 bytes
```

Writing Data from Sources Other Than Files

Sometimes it is necessary to write to a ZIP archive using data that did not come from an existing file. Rather than writing the data to a file, then adding that file to the ZIP archive, use the `writestr()` method to add a string of bytes to the archive directly.

```
# zipfile_writestr.py

from zipfile_infolist import print_info
import zipfile

msg = 'This data did not exist in a file.'
with zipfile.ZipFile('writestr.zip',
                     mode='w',
                     compression=zipfile.ZIP_DEFLATED,
                     ) as zf:
```

```
    zf.writestr('from_string.txt', msg)

print_info('writestr.zip')

with zipfile.ZipFile('writestr.zip', 'r') as zf:
    print(zf.read('from_string.txt'))
```

In this case, the `compress_type` argument to `ZipFile` was used to compress the data, since `writestr()` does not take an argument to specify the compression.

```
$ python3 zipfile_writestr.py

from_string.txt
  Comment     : b''
  Modified    : 2016-12-29 12:14:42
  System      : Unix
  ZIP version : 20
  Compressed  : 36 bytes
  Uncompressed: 34 bytes

b'This data did not exist in a file.'
```

Writing with a ZipInfo Instance

Normally, the modification date is computed when a file or string is added to the archive. A `ZipInfo` instance can be passed to `writestr()` to define the modification date and other metadata.

```
# zipfile_writestr_zipinfo.py

import time
import zipfile
from zipfile_infolist import print_info

msg = b'This data did not exist in a file.'

with zipfile.ZipFile('writestr_zipinfo.zip',
                     mode='w',
                     ) as zf:
    info = zipfile.ZipInfo('from_string.txt',
                           date_time=time.localtime(time.time()),
                           )
    info.compress_type = zipfile.ZIP_DEFLATED
    info.comment = b'Remarks go here'
    info.create_system = 0
    zf.writestr(info, msg)

print_info('writestr_zipinfo.zip')
```

In this example, the modified time is set to the current time, the data is compressed, and false value for `create_system` is used. A simple comment is also associated with the new file.

```
$ python3 zipfile_writestr_zipinfo.py

from_string.txt
  Comment     : b'Remarks go here'
  Modified    : 2016-12-29 12:14:42
  System      : Windows
  ZIP version : 20
  Compressed  : 36 bytes
  Uncompressed: 34 bytes
```

Appending to Files

In addition to creating new archives, it is possible to append to an existing archive or add an archive at the end of an existing file (such as a `.exe` file for a self-extracting archive). To open a file to append to it, use mode `'a'`.

```
# zipfile_append.py
```

```
from zipfile_infolist import print_info
import zipfile

print('creating archive')
with zipfile.ZipFile('append.zip', mode='w') as zf:
    zf.write('README.txt')

print()
print_info('append.zip')

print('appending to the archive')
with zipfile.ZipFile('append.zip', mode='a') as zf:
    zf.write('README.txt', arcname='README2.txt')

print()
print_info('append.zip')
```

The resulting archive contains two members:

```
$ python3 zipfile_append.py

creating archive

README.txt
  Comment     : b''
  Modified    : 2016-08-07 13:31:24
  System      : Unix
  ZIP version : 20
  Compressed  : 76 bytes
  Uncompressed: 76 bytes

appending to the archive

README.txt
  Comment     : b''
  Modified    : 2016-08-07 13:31:24
  System      : Unix
  ZIP version : 20
  Compressed  : 76 bytes
  Uncompressed: 76 bytes

README2.txt
  Comment     : b''
  Modified    : 2016-08-07 13:31:24
  System      : Unix
  ZIP version : 20
  Compressed  : 76 bytes
  Uncompressed: 76 bytes
```

Python ZIP Archives

Python can import modules from inside ZIP archives using zipimport, if those archives appear in `sys.path`. The `PyZipFile` class can be used to construct a module suitable for use in this way. The extra method `writepy()` tells `PyZipFile` to scan a directory for `.py` files and add the corresponding `.pyo` or `.pyc` file to the archive. If neither compiled form exists, a `.pyc` file is created and added.

```
# zipfile_pyzipfile.py

import sys
import zipfile

if __name__ == '__main__':
    with zipfile.PyZipFile('pyzipfile.zip', mode='w') as zf:
        zf.debug = 3
        print('Adding python files')
        zf.writepy('.')
    for name in zf.namelist():
        print(name)
```

```
    print()
    sys.path.insert(0, 'pyzipfile.zip')
    import zipfile_pyzipfile
    print('Imported from:', zipfile_pyzipfile.__file__)
```

With the debug attribute of the `PyZipFile` set to 3, verbose debugging is enabled and output is produced as it compiles each `.py` file it finds.

```
$ python3 zipfile_pyzipfile.py

Adding python files
Adding files from directory .
Compiling ./zipfile_append.py
Adding zipfile_append.pyc
Compiling ./zipfile_getinfo.py
Adding zipfile_getinfo.pyc
Compiling ./zipfile_infolist.py
Adding zipfile_infolist.pyc
Compiling ./zipfile_is_zipfile.py
Adding zipfile_is_zipfile.pyc
Compiling ./zipfile_namelist.py
Adding zipfile_namelist.pyc
Compiling ./zipfile_printdir.py
Adding zipfile_printdir.pyc
Compiling ./zipfile_pyzipfile.py
Adding zipfile_pyzipfile.pyc
Compiling ./zipfile_read.py
Adding zipfile_read.pyc
Compiling ./zipfile_write.py
Adding zipfile_write.pyc
Compiling ./zipfile_write_arcname.py
Adding zipfile_write_arcname.pyc
Compiling ./zipfile_write_compression.py
Adding zipfile_write_compression.pyc
Compiling ./zipfile_writestr.py
Adding zipfile_writestr.pyc
Compiling ./zipfile_writestr_zipinfo.py
Adding zipfile_writestr_zipinfo.pyc
zipfile_append.pyc
zipfile_getinfo.pyc
zipfile_infolist.pyc
zipfile_is_zipfile.pyc
zipfile_namelist.pyc
zipfile_printdir.pyc
zipfile_pyzipfile.pyc
zipfile_read.pyc
zipfile_write.pyc
zipfile_write_arcname.pyc
zipfile_write_compression.pyc
zipfile_writestr.pyc
zipfile_writestr_zipinfo.pyc

Imported from: pyzipfile.zip/zipfile_pyzipfile.pyc
```

Limitations

The `zipfile` module does not support ZIP files with appended comments, or multi-disk archives. It does support ZIP files larger than 4 GB that use the ZIP64 extensions.

See also

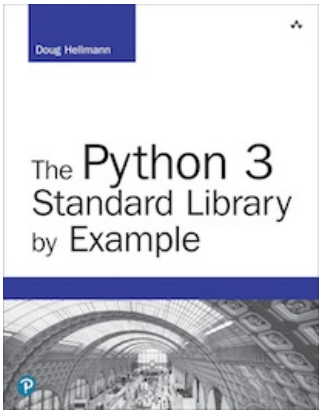
- Standard library documentation for zipfile
- `zlib` – ZIP compression library
- `tarfile` – Read and write tar archives
- `zipimport` – Import Python modules from ZIP archive.
- PKZIP Application Note – Official specification for the ZIP archive format.

Quick Links

This page was last updated 2017-01-29.

Navigation

tarfile — Tar Archive Access
Cryptography



Get the book

The output from all the example programs from PyMOTW-3 has been generated with Python 3.7.1, unless otherwise noted. Some of the features described here may not be available in earlier versions of Python.

Looking for examples for Python 2?

This Site

Module Index
Index

Other Writing

Blog
The Python Standard Library By Example