arclib Documentation

Release 0.1

Ryan Gonzalez

Nov 06, 2017

Contents

Python has a very thorough standard library, including a variety of modules for compressing and decompressing data. However, all of them have different APIs for working with them., leading to difficulty in attempting to support multiple archive formats. arclib is an attempt to fix this by providing a common API that bridges the various archive modules in Python.

CHAPTER 1

Rationale

A while back, I was trying to port a Python application from using zip files to using tar files. To say it was “painful” is an understatement. The two modules have conceptually similar but immensely different APIs. Therefore, I started work on arclib to provide a unified API between the two, as well as between the other archive modules in the standard library.

CHAPTER 2

Archive categories

`arclib` divides Python's five archive modules (zipfile, tarfile, lzma, bzip2, and gzip) into three categories:

- Basic, one-shot: the only compression method in this category is gzip. With gzip, the only allowed method of compression and decompression is "one-shot", where all the data must be processed at once.
- Basic, incremental: LZMA and bzip2 fall under here. These not only allow one-shot but also incremental compression and decompression, where the data is fed in in small chunks.
- Complex, file-system: Tar and zip both are considering "complex" in arclib, in that they both allow one to store entire directory trees, not just chunks of data.

CHAPTER 3

Modules

arclib contains the following modules:

3.1 Basic compression/decompression

- **arclib.gz:** A module that exposes a basic, one-shot gzip compression/decompression API.
- **arclib.bz2**: A module that exposes both a basic, one-shot bzip2 compression/decompression and an incremental API.
- **arclib.lzma**: A module that exposes both a basic, one-shot LZMA compression/decompression and an incremental API.

Note that arclib.gz does not expose an incremental API.

3.2 Complex, file-system compression/decompression

Both **arclib.zip** and **arclib.tar** expose an API that allows for manipulation of their respective complex archive formats.

CHAPTER 4

The APIs

4.1 Base

All modules in arclib implement this API:

`open`(**args*, ***kw*)

Returns a `File` with the given arguments.

4.2 Basic, one-shot

All the "basic" modules (gz, bz2, and lzma) implement this API:

detach, truncate https://docs.python.org/3/library/io.html#io.BufferedIOBase

`compress`(*data*)

Compresses data using the corresponding compression method.

Parameters **`data`** (*`bytes`*) – The data to compress.

Returns The compressed data.

Return type bytes

`decompress`(*data*)

Decompresses data using the corresponding decompression method.

Parameters **`data`** (*`bytes`*) – The data to decompress.

Returns The decompressed data.

Return type bytes

In addition, their `open` function is an alias for the corresponding Python module's `open` and therefore returns the module's `File`. For instance, `arclib.gz.open` is an alias for Python's own `gzip.open` and returns `gzip.GzipFile`.

4.3 Basic, incremental

Both arclib.bz2 and arclib.lzma (***not*** arclib.gz) implement this API.

class **`Compressor`**

A class that implements incremental compression. All types of this kind are instances of `arclib.AbstractBasicCompressor`. Example usage:

```
my_compressor = arclib.bz2.Compressor() # The compressor object.
compressed_data = b'' # The resulting compressed data.
compressed_data += my_compressor.compress(b'Something to compress...')
compressed_data += my_compressor.compress(b'More stuff!')
compressed_data += my_compressor.flush() # Always remember the flush call!
```

`compress`(*data*)

Incrementally compresses data using the corresponding compression method.

Parameters **`data`** (*`bytes`*) – The data to compress.

Returns A portion of compressed data, or an empty byte string. Note that this data is **not** considered valid on its own, and must be combined with both other calls to `compress` and the result of *`flush()`*.

Return type bytes

`flush`()

Flushes the compressor's internal buffers.

Returns The rest of the compressed data.

Return type bytes

class **`Decompressor`**

A class that implements incremental compression. All types of this kind are instances of `arclib.AbstractBasicDecompressor`. Example usage:

```
compressed_data = arclib.bz2.compress(b'Some data to compress!')
my_decompressor = arclib.bz2.Decompressor() # The decompressor object.
decompressed_data = b'' # The resulting decompressed data.
# Decompress some data.
decompressed_data += my_decompressor.decompress(compressed_data[:5])
# And some more data!
decompressed_data += my_decompressor.decompress(compressed_data[5:])
assert decompressed_data == b'Some data to compress!'
assert my_decompressor.eof
```

`decompress`(*data*)

Incrementally decompresses data using the corresponding decompression method.

Parameters **`data`** (*`bytes`*) – The data to decompress.

Returns A portion of decompressed data, or an empty byte string. Note that this data is **not** the complete decompressed data, and must b combined with other calls to `decompress`.

Return type bytes

`eof`

Whether or not the end of the compressed data has been reached.

`unused_data`

Any unused data left over after the decompression completed.

4.4 Complex

Both arclib.zip and arclib.tar implement this API.

open(**args*, ***kw*)
Opens an archive. All arguments are passed to the corresponding function; for instance, `arclib.zip.open` passes all its arguments to `zipfile.open`.

Returns The opened archive file.

Return type *File*

openobj(*fileobj*, ***kw*)
Opens the given file object. Whereas *open* opens a file path, *openobj* opens an in-memory file object.

Returns The opened archive file.

Return type *File*

class File
An opened archive file. Can be used as a context manager. Example:

```
import arclib.zip

with arclib.zip.open('myfile.zip') as f:
    # Stuff here.
# f is automatically closed.
```

close()
Close the archive file.

info_for(*member*)
Returns an *Info* object containing information about the given archive member.

Parameters **member** (*str*) – A string describing the path to the archive member, e.g. `x/y/z`.

Returns The member information object.

Return type *Info*

all_info()
Retrieves *Info* objects for all the archive's members.

Returns A list of all the *Info* objects for all the archive's members.

Return type list of *Info*

members()
Retrieves all the archive's members.

Returns A list of strings, one for each archive member.

Return type list of str

dump()
Dump a description archive's contents to standard output.

add(*path*, *arcname=None*, *recursive=True*)
Adds a file or directory to the archive.

Parameters

- **path** (*str*) – The path to add to the archive.

- **arcname** (*str*) – The name to give the file when placing it in the archive. If `None`, then it will be the same as *path*, but with leading roots and the drive removed.
- **recursive** (*bool*) – If *path* is a directory and this is a truthy value, then the directory's contents will also be added to the archive.

add_data (*path*, *data*)

Adds a `bytes` object to the archive.

Parameters

- **path** (*str*) – The name to give the file when placing it in the archive.
- **data** (*bytes*) – The file's contents.

extract (*member*, *path=None*)

Extracts a member from the archive.

Parameters

- **member** (*str*) – The member to extract.
- **path** (*str*) – The target path to extract the member to; if `None`, then it will be the current directory.

`arclib.zip.File.extract` also takes the following keyword argument:

Parameters **pwd** (*str*) – The password to use to extract the file, or `None`.

open (*member*, *universal_newlines=False*)

Extracts a member from the archive into memory rather that onto the disk. Returns a bytes file-like object with the following properties:

- *name* - The name of the member.
- *read(size=-1)* - Read and return *size* bytes from the file.

If `universal_newlines` is `True`, then the file object will be an instance of `io.TextIOWrapper` that also has the `name` property.

Parameters

- **member** (*str*) – The member to extract.
- **universal_newlines** (*str*) – If `True`, returns an `io.TextIOWrapper` that also has a property *name*, which is the name of the member. Otherwise, returns a file-like object as mentioned above.

`arclib.zip.File.extract` also takes the following keyword argument:

Parameters **pwd** (*str*) – The password to use to extract the file, or `None`.

Returns The file-like object as explained above, if the member is present. If it is not present, returns `None`.

class Info

An object containing information about an archive member.

info

The underlying, "true" info object. With `arclib.zip.Info`, this is an instance of `zipfile.ZipInfo`; with `arclib.tar.Info`, this is an instance of `tarfile.TarInfo`.

filename

The name of the file within the archive.

`size`
The number of bytes that the file takes up within the archive.

`mtime`
A `datetime.datetime` object containing the last modification time of the file.

Index