13.3. `bz2` — Support for **bzip2** compression

Source code: Lib/bz2.py

This module provides a comprehensive interface for compressing and decompressing data using the bzip2 compression algorithm.

The `bz2` module contains:

- The `open()` function and `BZ2File` class for reading and writing compressed files.
- The `BZ2Compressor` and `BZ2Decompressor` classes for incremental (de)compression.
- The `compress()` and `decompress()` functions for one-shot (de)compression.

All of the classes in this module may safely be accessed from multiple threads.

13.3.1. (De)compression of files

`bz2.`**`open`**(*filename*, *mode='r'*, *compresslevel=9*, *encoding=None*, *errors=None*, *newline=None*)

Open a bzip2-compressed file in binary or text mode, returning a file object.

As with the constructor for `BZ2File`, the *filename* argument can be an actual filename (a `str` or `bytes` object), or an existing file object to read from or write to.

The *mode* argument can be any of `'r'`, `'rb'`, `'w'`, `'wb'`, `'x'`, `'xb'`, `'a'` or `'ab'` for binary mode, or `'rt'`, `'wt'`, `'xt'`, or `'at'` for text mode. The default is `'rb'`.

The *compresslevel* argument is an integer from 1 to 9, as for the `BZ2File` constructor.

For binary mode, this function is equivalent to the `BZ2File` constructor: `BZ2File (filename, mode, compresslevel=compresslevel)`. In this case, the *encoding*, *errors* and *newline* arguments must not be provided.

For text mode, a `BZ2File` object is created, and wrapped in an `io.TextIOWrapper` instance with the specified encoding, error handling behavior, and line ending(s).

New in version 3.3.

Changed in version 3.4: The `'x'` (exclusive creation) mode was added.

Changed in version 3.6: Accepts a path-like object.

class `bz2.`**`BZ2File`**(*filename*, *mode='r'*, *buffering=None*, *compresslevel=9*)

Open a bzip2-compressed file in binary mode.

If *filename* is a `str` or `bytes` object, open the named file directly. Otherwise, *filename* should be a file object, which will be used to read or write the compressed data.

The *mode* argument can be either `'r'` for reading (default), `'w'` for overwriting, `'x'` for exclusive creation, or `'a'` for appending. These can equivalently be given as `'rb'`, `'wb'`, `'xb'` and `'ab'` respectively.

If *filename* is a file object (rather than an actual file name), a mode of `'w'` does not truncate the file, and is instead equivalent to `'a'`.

The *buffering* argument is ignored. Its use is deprecated.

If *mode* is `'w'` or `'a'`, *compresslevel* can be a number between `1` and `9` specifying the level of compression: `1` produces the least compression, and `9` (default) produces the most compression.

If *mode* is `'r'`, the input file may be the concatenation of multiple compressed streams.

`BZ2File` provides all of the members specified by the `io.BufferedIOBase`, except for `detach()` and `truncate()`. Iteration and the `with` statement are supported.

`BZ2File` also provides the following method:

`peek`([*n*])

Return buffered data without advancing the file position. At least one byte of data will be returned (unless at EOF). The exact number of bytes returned is unspecified.

Note: While calling `peek()` does not change the file position of the `BZ2File`, it may change the position of the underlying file object (e.g. if the `BZ2File` was constructed by passing a file object for *filename*).

New in version 3.3.

Changed in version 3.1: Support for the `with` statement was added.

Changed in version 3.3: The `fileno()`, `readable()`, `seekable()`, `writable()`, `read1()` and `readinto()` methods were added.

Changed in version 3.3: Support was added for *filename* being a file object instead of an actual filename.

Changed in version 3.3: The `'a'` (append) mode was added, along with support for reading multi-stream files.

Changed in version 3.4: The `'x'` (exclusive creation) mode was added.

Changed in version 3.5: The `read()` method now accepts an argument of `None`.

Changed in version 3.6: Accepts a path-like object.

13.3.2. Incremental (de)compression

class `bz2.`**`BZ2Compressor`**(*compresslevel=9*)

Create a new compressor object. This object may be used to compress data incrementally. For one-shot compression, use the `compress()` function instead.

compresslevel, if given, must be a number between `1` and `9`. The default is `9`.

`compress`(*data*)

Provide data to the compressor object. Returns a chunk of compressed data if possible, or an empty byte string otherwise.

When you have finished providing data to the compressor, call the `flush()` method to finish the compression process.

`flush`()

Finish the compression process. Returns the compressed data left in internal buffers.

The compressor object may not be used after this method has been called.

class `bz2.`**`BZ2Decompressor`**

Create a new decompressor object. This object may be used to decompress data incrementally. For one-shot compression, use the `decompress()` function instead.

Note: This class does not transparently handle inputs containing multiple compressed streams, unlike `decompress()` and `BZ2File`. If you need to decompress a multi-stream input with `BZ2Decompressor`, you must use a new decompressor for each stream.

`decompress`(*data*, *max_length=-1*)

Decompress *data* (a bytes-like object), returning uncompressed data as bytes. Some of *data* may be buffered internally, for use in later calls to `decompress()`. The returned data should be concatenated with the output of any previous calls to `decompress()`.

If *max_length* is nonnegative, returns at most *max_length* bytes of decompressed data. If this limit is reached and further output can be produced, the `needs_input` attribute will be set to `False`. In this case, the next call to `decompress()` may provide *data* as `b''` to obtain more of the output.

If all of the input data was decompressed and returned (either because this was less than *max_length* bytes, or because *max_length* was negative), the `needs_input` attribute will be set to `True`.

Attempting to decompress data after the end of stream is reached raises an *EOFError*. Any data found after the end of the stream is ignored and saved in the `unused_data` attribute.

Changed in version 3.5: Added the *max_length* parameter.

`eof`

`True` if the end-of-stream marker has been reached.

New in version 3.3.

`unused_data`

Data found after the end of the compressed stream.

If this attribute is accessed before the end of the stream has been reached, its value will be `b''`.

`needs_input`

`False` if the `decompress()` method can provide more decompressed data before requiring new uncompressed input.

New in version 3.5.

13.3.3. One-shot (de)compression

`bz2.`**`compress`**(*data*, *compresslevel=9*)

Compress *data*.

compresslevel, if given, must be a number between `1` and `9`. The default is `9`.

For incremental compression, use a `BZ2Compressor` instead.

`bz2.`**`decompress`**(*data*)

Decompress *data*.

If *data* is the concatenation of multiple compressed streams, decompress all of the streams.

For incremental decompression, use a `BZ2Decompressor` instead.

Changed in version 3.3: Support for multi-stream inputs was added.