22.5. wave — Read and write WAV files¶

The wave module provides a convenient interface to the WAV sound format. It does not support compression/decompression, but it does support mono/stereo.

The wave module defines the following function and exception:

`wave.open`(*file*[, *mode*])¶

If *file* is a string, open the file by that name, other treat it as a seekable file-like object. *mode* can be any of

`'r'`, `'rb'`
Read only mode.
`'w'`, `'wb'`
Write only mode.

Note that it does not allow read/write WAV files.

A *mode* of `'r'` or `'rb'` returns a `Wave_read` object, while a *mode* of `'w'` or `'wb'` returns a `Wave_write` object. If *mode* is omitted and a file-like object is passed as *file*, `file.mode` is used as the default value for *mode* (the `'b'` flag is still added if necessary).

`wave.openfp`(*file*, *mode*)¶
A synonym for `open()`, maintained for backwards compatibility.

exception `wave.Error`¶
An error raised when something is impossible because it violates the WAV specification or hits an implementation deficiency.

22.5.1. Wave_read Objects¶

Wave_read objects, as returned by `open()`, have the following methods:

`Wave_read.close`()¶
Close the stream, and make the instance unusable. This is called automatically on object collection.

`Wave_read.getnchannels`()¶
Returns number of audio channels (1 for mono, 2 for stereo).

`Wave_read.getsampwidth`()¶
Returns sample width in bytes.

`Wave_read.getframerate`()¶
Returns sampling frequency.

`Wave_read.getnframes`()¶
Returns number of audio frames.

`Wave_read.getcomptype`()¶
Returns compression type (`'NONE'` is the only supported type).

`Wave_read.getcompname`()¶
Human-readable version of `getcomptype()`. Usually `'not compressed'` parallels `'NONE'`.

`Wave_read.getparams`()¶
Returns a tuple `(nchannels, sampwidth, framerate, nframes, comptype, compname)`, equivalent to output of the `get*()` methods.

`Wave_read.readframes`(*n*)¶
Reads and returns at most *n* frames of audio, as a string of bytes.

`Wave_read.rewind`()¶
Rewind the file pointer to the beginning of the audio stream.

The following two methods are defined for compatibility with the `aifc` module, and don't do anything interesting.

`Wave_read.getmarkers`()¶
Returns `None`.

`Wave_read.getmark`(*id*)¶
Raise an error.

The following two methods define a term “position” which is compatible between them, and is otherwise implementation dependent.

`Wave_read.setpos`(*pos*)¶
Set the file pointer to the specified position.

`Wave_read.tell`()¶
Return current file pointer position.

22.5.2. Wave_write Objects¶

Wave_write objects, as returned by `open()`, have the following methods:

`Wave_write.close`()¶
Make sure *nframes* is correct, and close the file. This method is called upon deletion.

`Wave_write.setnchannels`(*n*)¶
Set the number of channels.

`Wave_write.setsampwidth`(*n*)¶
Set the sample width to *n* bytes.

`Wave_write.setframerate`(*n*)¶
Set the frame rate to *n*.

`Wave_write.setnframes`(*n*)¶
Set the number of frames to *n*. This will be changed later if more frames are written.

`Wave_write.setcomptype`(*type*, *name*)¶
Set the compression type and description. At the moment, only compression type `NONE` is supported, meaning no compression.

`Wave_write.setparams`(*tuple*)¶
The *tuple* should be `(nchannels, sampwidth, framerate, nframes, comptype, compname)`, with values valid for the `set*()` methods. Sets all parameters.

`Wave_write.tell`()¶
Return current position in the file, with the same disclaimer for the `Wave_read.tell()` and `Wave_read.setpos()` methods.

`Wave_write.writeframesraw`(*data*)¶
Write audio frames, without correcting *nframes*.

`Wave_write.writeframes`(*data*)¶
Write audio frames and make sure *nframes* is correct.

Note that it is invalid to set any parameters after calling `writeframes()` or `writeframesraw()`, and any attempt to do so will raise `wave.Error`.

Table Of Contents

Previous topic

22.4. sunau — Read and write Sun AU files

Next topic

22.6. chunk — Read IFF chunked data

This Page

- Show Source

Navigation

- index
- modules |
- next |
- previous |
- Python v2.6.4 documentation »
- The Python Standard Library »
- 22. Multimedia Services »