sunau — Read and write Sun AU files

Source code: Lib/sunau.py

The sunau module provides a convenient interface to the Sun AU sound format. Note that this module is interface-compatible with the modules aifc and wave.

An audio file consists of a header followed by the data. The fields of the header are:

Field	Contents
magic word	The four bytes .snd.
header size	Size of the header, including info, in bytes.
data size	Physical size of the data, in bytes.
encoding	Indicates how the audio samples are encoded.
sample rate	The sampling rate.
# of channels	The number of channels in the samples.
info	ASCII string giving a description of the audio file (padded with null bytes).

Apart from the info field, all header fields are 4 bytes in size. They are all 32-bit unsigned integers encoded in big-endian byte order.

The sunau module defines the following functions:

sunau.**open**(*file*, *mode*)

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

'r'

Read only mode.

'w'

Write only mode.

Note that it does not allow read/write files.

A *mode* of 'r' returns an AU_read object, while a *mode* of 'w' or 'wb' returns an AU_write object.

sunau.**openfp**(*file*, *mode*)

A synonym for open(), maintained for backwards compatibility.

Deprecated since version 3.7, will be removed in version 3.9.

The sunau module defines the following exception:

exception sunau.**Error**
An error raised when something is impossible because of Sun AU specs or implementation deficiency.

The sunau module defines the following data items:

sunau.**AUDIO_FILE_MAGIC**
An integer every valid Sun AU file begins with, stored in big-endian form. This is the string .snd interpreted as an integer.

sunau.**AUDIO_FILE_ENCODING_MULAW_8**
sunau.**AUDIO_FILE_ENCODING_LINEAR_8**
sunau.**AUDIO_FILE_ENCODING_LINEAR_16**
sunau.**AUDIO_FILE_ENCODING_LINEAR_24**
sunau.**AUDIO_FILE_ENCODING_LINEAR_32**
sunau.**AUDIO_FILE_ENCODING_ALAW_8**
Values of the encoding field from the AU header which are supported by this module.

sunau.**AUDIO_FILE_ENCODING_FLOAT**
sunau.**AUDIO_FILE_ENCODING_DOUBLE**
sunau.**AUDIO_FILE_ENCODING_ADPCM_G721**
sunau.**AUDIO_FILE_ENCODING_ADPCM_G722**
sunau.**AUDIO_FILE_ENCODING_ADPCM_G723_3**
sunau.**AUDIO_FILE_ENCODING_ADPCM_G723_5**
Additional known values of the encoding field from the AU header, but which are not supported by this module.

AU_read Objects

AU_read objects, as returned by open() above, have the following methods:

AU_read.**close**()¶
Close the stream, and make the instance unusable. (This is called automatically on deletion.)

AU_read.**getnchannels**()
Returns number of audio channels (1 for mono, 2 for stereo).

AU_read.**getsampwidth**()
Returns sample width in bytes.

AU_read.**getframerate**()

Returns sampling frequency.

`AU_read.`**`getnframes`**()
Returns number of audio frames.

`AU_read.`**`getcomptype`**()
Returns compression type. Supported compression types are `'ULAW'`, `'ALAW'` and `'NONE'`.

`AU_read.`**`getcompname`**()
Human-readable version of `getcomptype()`. The supported types have the respective names `'CCITT G.711 u-law'`, `'CCITT G.711 A-law'` and `'not compressed'`.

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

`AU_read.`**`readframes`**(*n*)
Reads and returns at most *n* frames of audio, as a `bytes` object. The data will be returned in linear format. If the original data is in u-LAW format, it will be converted.

`AU_read.`**`rewind`**()
Rewind the file pointer to the beginning of the audio stream.

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

`AU_read.`**`setpos`**(*pos*)
Set the file pointer to the specified position. Only values returned from `tell()` should be used for *pos*.

`AU_read.`**`tell`**()
Return current file pointer position. Note that the returned value has nothing to do with the actual position in the file.

The following two functions are defined for compatibility with the `aifc`, and don’t do anything interesting.

`AU_read.`**`getmarkers`**()
Returns `None`.

`AU_read.`**`getmark`**(*id*)
Raise an error.

AU_write Objects

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

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

`AU_write.`**`setsampwidth`**(*n*)
Set the sample width (in bytes.)

Changed in version 3.4: Added support for 24-bit samples.

`AU_write.`**`setframerate`**(*n*)
Set the frame rate.

`AU_write.`**`setnframes`**(*n*)
Set the number of frames. This can be later changed, when and if more frames are written.

`AU_write.`**`setcomptype`**(*type*, *name*)
Set the compression type and description. Only `'NONE'` and `'ULAW'` are supported on output.

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

`AU_write.`**`tell`**()
Return current position in the file, with the same disclaimer for the `AU_read.tell()` and `AU_read.setpos()` methods.

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

Changed in version 3.4: Any bytes-like object is now accepted.

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

Changed in version 3.4: Any bytes-like object is now accepted.

`AU_write.`**`close`**()
Make sure *nframes* is correct, and close the file.

This method is called upon deletion.

Note that it is invalid to set any parameters after calling `writeframes()` or `writeframesraw()`.