Data marshalling support

These routines allow C code to work with serialized objects using the same data format as the `marshal` module. There are functions to write data into the serialization format, and additional functions that can be used to read the data back. Files used to store marshalled data must be opened in binary mode.

Numeric values are stored with the least significant byte first.

The module supports several versions of the data format; see the `Python module documentation <marshal>` for details.

The following exceptions can be raised by these functions: `ValueError` if the value cannot be marshalled, `ValueError` or `TypeError` if the data is malformed, `EOFError` if the end of the data is reached before the value is complete, `OSError` if reading from or writing to a `FILE*` fails, `KeyboardInterrupt` if reading or writing is interrupted by a signal, and `MemoryError` if memory allocation fails.

3.16 Previously, in functions taking a `FILE*`, the reading functions raised `EOFError` instead of `OSError` and `KeyboardInterrupt`, and the writing functions ignored I/O errors and interruptions.

> The current format version. See `marshal.version`.

> Marshal a `long` integer, *value*, to *file*. This will only write the least-significant 32 bits of *value*; regardless of the size of the native `long` type. *version* indicates the file format.
>
> This function can fail, in which case it sets the error indicator. Use `PyErr_Occurred` to check for that.
>
> > [!NOTE] In Python 3.15 and earlier, this function does not set the error indicator if writing to *file* fails. The same caveat applies as for `PyMarshal_WriteObjectToFile`.

>

> Marshal a Python object, *value*, to *file*. *version* indicates the file format.
>
> The marshalled representation of *value* is built in memory and then written to *file*. Nothing is written if *value* cannot be marshalled.
>
> This function can fail, in which case it sets the error indicator. Use `PyErr_Occurred` to check for that.
>
> > [!NOTE] In Python 3.15 and earlier, this function can write only part of the data to *file* and return without setting the error indicator, for example if *value* contains an object which cannot be marshalled or if writing to *file* fails (see `155907`). Code which has to support these versions cannot rely on `PyErr_Occurred` alone to know that all the data was written. It can use `PyMarshal_WriteObjectToString` and write the result itself, or validate what was written.
>
> 3.16 Previously, the data was written to *file* as it was produced, using a buffer of constant size, and incomplete data was written if *value* could not be marshalled.

> Return a bytes object containing the marshalled representation of *value*. *version* indicates the file format.
>
> On error, raises an exception and returns `NULL`.

The following functions allow marshalled values to be read back in.

> Return a C `long` from the data stream in a `FILE*` opened for reading. Only a 32-bit value can be read in using this function, regardless of the native size of `long`.
>
> On error, raises an exception and returns `-1`.

> Return a C `short` from the data stream in a `FILE*` opened for reading. Only a 16-bit value can be read in using this function, regardless of the native size of `short`.

> On error, raises an exception and returns `-1`.

> Return a Python object from the data stream in a `FILE*` opened for reading.
>
> On error, raises an exception and returns `NULL`.

> Return a Python object from the data stream in a `FILE*` opened for reading. Unlike `PyMarshal_ReadObjectFromFile`, this function assumes that no further objects will be read from the file, allowing it to aggressively load file data into memory so that the de-serialization can operate from data in memory rather than reading a byte at a time from the file. Only use this variant if you are certain that you won't be reading anything else from the file.
>
> On error, raises an exception and returns `NULL`.

> Return a Python object from the data stream in a byte buffer containing *len* bytes pointed to by *data*.
>
> On error, raises an exception and returns `NULL`.

□□□ #2
□ □□□ □□□ 2026-10-31 16:52:33 UTC
□ □□□ □□□ 2026-10-31 16:52:35 UTC