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 two versions of the data format: version 0 is the historical version, version 1 shares interned strings in the file, and upon unmarshalling. Version 2 uses a binary format for floating point numbers. *Py_MARSHAL_VERSION* indicates the current file format (currently 2).

void **PyMarshal_WriteLongToFile**(long *value*, FILE **file*, int *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.

void **PyMarshal_WriteObjectToFile**(PyObject **value*, FILE **file*, int *version*)

Marshal a Python object, *value*, to *file*. *version* indicates the file format.

PyObject* **PyMarshal_WriteObjectToString**(PyObject **value*, int *version*)

Return value: New reference.
Return a bytes object containing the marshalled representation of *value*. *version* indicates the file format.

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

XXX What about error detection? It appears that reading past the end of the file will always result in a negative numeric value (where that's relevant), but it's not clear that negative values won't be handled properly when there's no error. What's the right way to tell? Should only non-negative values be written using these routines?

long **PyMarshal_ReadLongFromFile**(FILE **file*)

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, raise an exception and return `-1`.

int **PyMarshal_ReadShortFromFile**(FILE **file*)

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, raise an exception and return `-1`.

PyObject* **PyMarshal_ReadObjectFromFile**(FILE **file*)

Return value: New reference.

Return a Python object from the data stream in a `FILE*` opened for reading.

On error, sets the appropriate exception (`EOFError` or `TypeError`) and returns *NULL*.

PyObject* **PyMarshal_ReadLastObjectFromFile**(FILE **file*)

Return value: New reference.

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 these variant if you are certain that you won't be reading anything else from the file.

On error, sets the appropriate exception (`EOFError` or `TypeError`) and returns *NULL*.

PyObject* **PyMarshal_ReadObjectFromString**(const char **data*, Py_ssize_t *len*)

Return value: New reference.

Return a Python object from the data stream in a byte buffer containing *len* bytes pointed to by *data*.

On error, sets the appropriate exception (`EOFError` or `TypeError`) and returns *NULL*.