Codec registry and support functions

int **`PyCodec_Register`**(PyObject **search_function*)

Register a new codec search function.

As side effect, this tries to load the `encodings` package, if not yet done, to make sure that it is always first in the list of search functions.

int **`PyCodec_KnownEncoding`**(const char **encoding*)

Return `1` or `0` depending on whether there is a registered codec for the given *encoding*.

PyObject* **`PyCodec_Encode`**(PyObject **object*, const char **encoding*, const char **errors*)

Generic codec based encoding API.

object is passed through the encoder function found for the given *encoding* using the error handling method defined by *errors*. *errors* may be *NULL* to use the default method defined for the codec. Raises a `LookupError` if no encoder can be found.

PyObject* **`PyCodec_Decode`**(PyObject **object*, const char **encoding*, const char **errors*)

Generic codec based decoding API.

object is passed through the decoder function found for the given *encoding* using the error handling method defined by *errors*. *errors* may be *NULL* to use the default method defined for the codec. Raises a `LookupError` if no encoder can be found.

Codec lookup API

In the following functions, the *encoding* string is looked up converted to all lower-case characters, which makes encodings looked up through this mechanism effectively case-insensitive. If no codec is found, a `KeyError` is set and *NULL* returned.

PyObject* **`PyCodec_Encoder`**(const char **encoding*)

Get an encoder function for the given *encoding*.

PyObject* **`PyCodec_Decoder`**(const char **encoding*)

Get a decoder function for the given *encoding*.

PyObject* **PyCodec_IncrementalEncoder**(const char **encoding*, const char **errors*)

Get an `IncrementalEncoder` object for the given *encoding*.

PyObject* **PyCodec_IncrementalDecoder**(const char **encoding*, const char **errors*)

Get an `IncrementalDecoder` object for the given *encoding*.

PyObject* **PyCodec_StreamReader**(const char **encoding*, PyObject **stream*, const char **errors*)

Get a `StreamReader` factory function for the given *encoding*.

PyObject* **PyCodec_StreamWriter**(const char **encoding*, PyObject **stream*, const char **errors*)

Get a `StreamWriter` factory function for the given *encoding*.

Registry API for Unicode encoding error handlers

int **PyCodec_RegisterError**(const char **name*, PyObject **error*)

Register the error handling callback function *error* under the given *name*. This callback function will be called by a codec when it encounters unencodable characters/undecodable bytes and *name* is specified as the error parameter in the call to the encode/decode function.

The callback gets a single argument, an instance of `UnicodeEncodeError`, `UnicodeDecodeError` or `UnicodeTranslateError` that holds information about the problematic sequence of characters or bytes and their offset in the original string (see Unicode Exception Objects for functions to extract this information). The callback must either raise the given exception, or return a two-item tuple containing the replacement for the problematic sequence, and an integer giving the offset in the original string at which encoding/decoding should be resumed.

Return `0` on success, `-1` on error.

PyObject* **PyCodec_LookupError**(const char **name*)

Lookup the error handling callback function registered under *name*. As a special case *NULL* can be passed, in which case the error handling callback for “strict” will be returned.

PyObject* **PyCodec_StrictErrors**(PyObject **exc*)

Raise *exc* as an exception.

PyObject* **PyCodec_IgnoreErrors**(PyObject **exc*)

Ignore the unicode error, skipping the faulty input.

PyObject* **`PyCodec_ReplaceErrors`**(PyObject **exc*)

Replace the unicode encode error with `?` or `U+FFFD`.

PyObject* **`PyCodec_XMLCharRefReplaceErrors`**(PyObject **exc*)

Replace the unicode encode error with XML character references.

PyObject* **`PyCodec_BackslashReplaceErrors`**(PyObject **exc*)

Replace the unicode encode error with backslash escapes (`\x`, `\u` and `\U`).

PyObject* **`PyCodec_NameReplaceErrors`**(PyObject **exc*)

Replace the unicode encode error with `\N{...}` escapes.

New in version 3.5.