

19.1.12. `email.charset`: Representing character sets

Source code: [Lib/email/charset.py](#)

This module is part of the legacy (Compat32) email API. In the new API only the aliases table is used.

The remaining text in this section is the original documentation of the module.

This module provides a class `Charset` for representing character sets and character set conversions in email messages, as well as a character set registry and several convenience methods for manipulating this registry. Instances of `Charset` are used in several other modules within the `email` package.

Import this class from the `email.charset` module.

```
class email.charset.Charset(input_charset=DEFAULT_CHARSET)
```

Map character sets to their email properties.

This class provides information about the requirements imposed on email for a specific character set. It also provides convenience routines for converting between character sets, given the availability of the applicable codecs. Given a character set, it will do its best to provide information on how to use that character set in an email message in an RFC-compliant way.

Certain character sets must be encoded with quoted-printable or base64 when used in email headers or bodies. Certain character sets must be converted outright, and are not allowed in email.

Optional *input_charset* is as described below; it is always coerced to lower case. After being alias normalized it is also used as a lookup into the registry of character sets to find out the header encoding, body encoding, and output conversion codec to be used for the character set. For example, if *input_charset* is `iso-8859-1`, then headers and bodies will be encoded using quoted-printable and no output conversion codec is necessary. If *input_charset* is `euc-jp`, then headers will be encoded with base64, bodies will not be encoded, but output text will be converted from the `euc-jp` character set to the `iso-2022-jp` character set.

`Charset` instances have the following data attributes:

input_charset

The initial character set specified. Common aliases are converted to their *official* email names (e.g. `latin_1` is converted to `iso-8859-1`). Defaults to 7-bit us-ascii.

header_encoding

If the character set must be encoded before it can be used in an email header, this attribute will be set to `Charset.QP` (for quoted-printable), `Charset.BASE64` (for base64 encoding), or `Charset.SHORTEST` for the shortest of QP or BASE64 encoding. Otherwise, it will be `None`.

body_encoding

Same as *header_encoding*, but describes the encoding for the mail message's body, which indeed may be different than the header encoding. `Charset.SHORTEST` is not allowed for *body_encoding*.

output_charset

Some character sets must be converted before they can be used in email headers or bodies. If the *input_charset* is one of them, this attribute will contain the name of the character set output will be converted to. Otherwise, it will be `None`.

input_codec

The name of the Python codec used to convert the *input_charset* to Unicode. If no conversion codec is necessary, this attribute will be `None`.

output_codec

The name of the Python codec used to convert Unicode to the *output_charset*. If no conversion codec is necessary, this attribute will have the same value as the *input_codec*.

`Charset` instances also have the following methods:

get_body_encoding()

Return the content transfer encoding used for body encoding.

This is either the string quoted-printable or base64 depending on the encoding used, or it is a function, in which case you should call the function with a single argument, the Message object being encoded. The function should then set the *Content-Transfer-Encoding* header itself to whatever is appropriate.

Returns the string quoted-printable if *body_encoding* is QP, returns the string base64 if *body_encoding* is BASE64, and returns the string 7bit otherwise.

get_output_charset()

Return the output character set.

This is the *output_charset* attribute if that is not None, otherwise it is *input_charset*.

header_encode(string)

Header-encode the string *string*.

The type of encoding (base64 or quoted-printable) will be based on the *header_encoding* attribute.

header_encode_lines(string, maxlengths)

Header-encode a *string* by converting it first to bytes.

This is similar to [header_encode\(\)](#) except that the string is fit into maximum line lengths as given by the argument *maxlengths*, which must be an iterator: each element returned from this iterator will provide the next maximum line length.

body_encode(string)

Body-encode the string *string*.

The type of encoding (base64 or quoted-printable) will be based on the *body_encoding* attribute.

The [Charset](#) class also provides a number of methods to support standard operations and built-in functions.

__str__()

Returns *input_charset* as a string coerced to lower case. [__repr__\(\)](#) is an alias for [__str__\(\)](#).

__eq__(other)

This method allows you to compare two [Charset](#) instances for equality.

__ne__(other)

This method allows you to compare two [Charset](#) instances for inequality.

The [email.charset](#) module also provides the following functions for adding new entries to the global character set, alias, and codec registries:

`email.charset.add_charset(charset, header_enc=None, body_enc=None, output_charset=None)`

Add character properties to the global registry.

charset is the input character set, and must be the canonical name of a character set.

Optional *header_enc* and *body_enc* is either `Charset.QP` for quoted-printable, `Charset.BASE64` for base64 encoding, `Charset.SHORTEST` for the shortest of quoted-printable or base64 encoding, or `None` for no encoding. `SHORTEST` is only valid for *header_enc*. The default is `None` for no encoding.

Optional *output_charset* is the character set that the output should be in. Conversions will proceed from input charset, to Unicode, to the output charset when the method `Charset.convert()` is called. The default is to output in the same character set as the input.

Both *input_charset* and *output_charset* must have Unicode codec entries in the module's character set-to-codec mapping; use [add_codec\(\)](#) to add codecs the module does not know about. See the [codecs](#) module's documentation for more information.

The global character set registry is kept in the module global dictionary `CHARSETS`.

`email.charset.add_alias(alias, canonical)`

Add a character set alias. *alias* is the alias name, e.g. `latin-1`. *canonical* is the character set's canonical name, e.g. `iso-8859-1`.

The global charset alias registry is kept in the module global dictionary `ALIASES`.

`email.charset.add_codec(charset, codecname)`

Add a codec that map characters in the given character set to and from Unicode.

charset is the canonical name of a character set. *codecname* is the name of a Python codec, as appropriate for the second argument to the [str](#)'s [encode\(\)](#) method.