

# mimetypes — Map filenames to MIME types

**Source code:** [Lib/mimetypes.py](#)

The `mimetypes` module converts between a filename or URL and the MIME type associated with the filename extension. Conversions are provided from filename to MIME type and from MIME type to filename extension; encodings are not supported for the latter conversion.

The module provides one class and a number of convenience functions. The functions are the normal interface to this module, but some applications may be interested in the class as well.

The functions described below provide the primary interface for this module. If the module has not been initialized, they will call `init()` if they rely on the information `init()` sets up.

`mimetypes.guess_type(url, strict=True)`

Guess the type of a file based on its filename, path or URL, given by *url*. URL can be a string or a [path-like object](#).

The return value is a tuple (*type*, *encoding*) where *type* is `None` if the type can't be guessed (missing or unknown suffix) or a string of the form 'type/subtype', usable for a MIME *content-type* header.

*encoding* is `None` for no encoding or the name of the program used to encode (e.g. **compress** or **gzip**). The encoding is suitable for use as a *Content-Encoding* header, **not** as a *Content-Transfer-Encoding* header. The mappings are table driven. Encoding suffixes are case sensitive; type suffixes are first tried case sensitively, then case insensitively.

The optional *strict* argument is a flag specifying whether the list of known MIME types is limited to only the official types [registered with IANA](#). When *strict* is `True` (the default), only the IANA types are supported; when *strict* is `False`, some additional non-standard but commonly used MIME types are also recognized.

*Changed in version 3.8:* Added support for url being a [path-like object](#).

`mimetypes.guess_all_extensions(type, strict=True)`

Guess the extensions for a file based on its MIME type, given by *type*. The return value is a list of strings giving all possible filename extensions, including the leading dot ('.'). The extensions are not guaranteed to have been associated with any particular data stream, but would be mapped to the MIME type *type* by `guess_type()`.

The optional *strict* argument has the same meaning as with the `guess_type()` function.

`mimetypes.guess_extension(type, strict=True)`

Guess the extension for a file based on its MIME type, given by *type*. The return value is a string giving a filename extension, including the leading dot ('.'). The extension is not guaranteed to have been associated with any particular data stream, but would be mapped to the MIME type *type* by `guess_type()`. If no extension can be guessed for *type*, `None` is returned.

The optional *strict* argument has the same meaning as with the `guess_type()` function.

Some additional functions and data items are available for controlling the behavior of the module.

#### `mimetypes.init(files=None)`

Initialize the internal data structures. If given, *files* must be a sequence of file names which should be used to augment the default type map. If omitted, the file names to use are taken from `knownfiles`; on Windows, the current registry settings are loaded. Each file named in *files* or `knownfiles` takes precedence over those named before it. Calling `init()` repeatedly is allowed.

Specifying an empty list for *files* will prevent the system defaults from being applied: only the well-known values will be present from a built-in list.

If *files* is `None` the internal data structure is completely rebuilt to its initial default value. This is a stable operation and will produce the same results when called multiple times.

*Changed in version 3.2:* Previously, Windows registry settings were ignored.

#### `mimetypes.read_mime_types(filename)`

Load the type map given in the file *filename*, if it exists. The type map is returned as a dictionary mapping filename extensions, including the leading dot ('.'), to strings of the form 'type/subtype'. If the file *filename* does not exist or cannot be read, `None` is returned.

#### `mimetypes.add_type(type, ext, strict=True)`

Add a mapping from the MIME type *type* to the extension *ext*. When the extension is already known, the new type will replace the old one. When the type is already known the extension will be added to the list of known extensions.

When *strict* is `True` (the default), the mapping will be added to the official MIME types, otherwise to the non-standard ones.

#### `mimetypes.inited`

Flag indicating whether or not the global data structures have been initialized. This is set to `True` by `init()`.

#### `mimetypes.knownfiles`

List of type map file names commonly installed. These files are typically named `mime.types` and are installed in different locations by different packages.

**mimetypes.suffix\_map**

Dictionary mapping suffixes to suffixes. This is used to allow recognition of encoded files for which the encoding and the type are indicated by the same extension. For example, the `.tgz` extension is mapped to `.tar.gz` to allow the encoding and type to be recognized separately.

**mimetypes.encodings\_map**

Dictionary mapping filename extensions to encoding types.

**mimetypes.types\_map**

Dictionary mapping filename extensions to MIME types.

**mimetypes.common\_types**

Dictionary mapping filename extensions to non-standard, but commonly found MIME types.

An example usage of the module:

```
>>> import mimetypes
>>> mimetypes.init()
>>> mimetypes.knownfiles
['/etc/mime.types', '/etc/httpd/mime.types', ... ]
>>> mimetypes.suffix_map['.tgz']
'.tar.gz'
>>> mimetypes.encodings_map['.gz']
'gzip'
>>> mimetypes.types_map['.tgz']
'application/x-tar-gz'
```

## MimeTypes Objects

The `MimeTypes` class may be useful for applications which may want more than one MIME-type database; it provides an interface similar to the one of the `mimetypes` module.

**`class mimetypes.MimeTypes(filenames=(), strict=True)`**

This class represents a MIME-types database. By default, it provides access to the same database as the rest of this module. The initial database is a copy of that provided by the module, and may be extended by loading additional `mime.types`-style files into the database using the `read()` or `readfp()` methods. The mapping dictionaries may also be cleared before loading additional data if the default data is not desired.

The optional `filenames` parameter can be used to cause additional files to be loaded “on top” of the default database.

**suffix\_map**

Dictionary mapping suffixes to suffixes. This is used to allow recognition of encoded files for which the encoding and the type are indicated by the same extension. For example,

the `.tgz` extension is mapped to `.tar.gz` to allow the encoding and type to be recognized separately. This is initially a copy of the global `suffix_map` defined in the module.

## **encodings\_map**

Dictionary mapping filename extensions to encoding types. This is initially a copy of the global `encodings_map` defined in the module.

## **types\_map**

Tuple containing two dictionaries, mapping filename extensions to MIME types: the first dictionary is for the non-standards types and the second one is for the standard types. They are initialized by `common_types` and `types_map`.

## **types\_map\_inv**

Tuple containing two dictionaries, mapping MIME types to a list of filename extensions: the first dictionary is for the non-standards types and the second one is for the standard types. They are initialized by `common_types` and `types_map`.

## **guess\_extension(*type*, *strict*=True)**

Similar to the `guess_extension()` function, using the tables stored as part of the object.

## **guess\_type(*url*, *strict*=True)**

Similar to the `guess_type()` function, using the tables stored as part of the object.

## **guess\_all\_extensions(*type*, *strict*=True)**

Similar to the `guess_all_extensions()` function, using the tables stored as part of the object.

## **read(*filename*, *strict*=True)**

Load MIME information from a file named *filename*. This uses `readfp()` to parse the file.

If *strict* is `True`, information will be added to list of standard types, else to the list of non-standard types.

## **readfp(*fp*, *strict*=True)**

Load MIME type information from an open file *fp*. The file must have the format of the standard `mime.types` files.

If *strict* is `True`, information will be added to the list of standard types, else to the list of non-standard types.

## **read\_windows\_registry(*strict*=True)**

Load MIME type information from the Windows registry.

**Availability:** Windows.

If *strict* is `True`, information will be added to the list of standard types, else to the list of non-standard types.

*New in version 3.2.*