:mod:`mimetypes` --- Map filenames to MIME types

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, **line 1**); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, **line 4**)

Unknown directive type "module".

```
.. module:: mimetypes
   :synopsis: Mapping of filename extensions to MIME types.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, **line 7**)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Fred L. Drake, Jr. <fdrake@acm.org>
```

Source code: :source:`Lib/mimetypes.py`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, **line 9**); *backlink*

Unknown interpreted text role "source".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, **line 11**)

Unknown directive type "index".

```
.. index:: pair: MIME; content type
```

The :mod:`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.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, **line 15**); *backlink*

Unknown interpreted text role "mod".

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 :func:`init` if they rely on the information :func:`init` sets up.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, **line 24**); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, **line 24**); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-

```
    .. function:: guess_type(url, strict=True)

        .. index:: pair: MIME; headers

        Guess the type of a file based on its filename, path or URL, given by *url*.
        URL can be a string or a :term:`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 :mailheader:`content-type` header.

        *encoding* is ``None`` for no encoding or the name of the program used to encode
        (e.g. :program:`compress` or :program:`gzip`). The encoding is suitable for use
        as a :mailheader:`Content-Encoding` header, **not** as a
        :mailheader:`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
        <https://www.iana.org/assignments/media-types/media-types.xhtml>`_.
        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.

        .. versionchanged:: 3.8
           Added support for url being a :term:`path-like object`.
```

```
    .. function:: 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 :func:`guess_type`.

        The optional *strict* argument has the same meaning as with the :func:`guess_type` function.
```

```
    .. function:: 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
        :func:`guess_type`.  If no extension can be guessed for *type*, ``None`` is
        returned.

        The optional *strict* argument has the same meaning as with the :func:`guess_type` function.
```

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

```
    .. function:: 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 :const:`knownfiles`; on Windows, the
current registry settings are loaded. Each file named in *files* or
:const:`knownfiles` takes precedence over those named before it. Calling
:func:`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.

.. versionchanged:: 3.2
   Previously, Windows registry settings were ignored.

Unknown directive type "function".

```
.. function:: 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.

Unknown directive type "function".

```
.. function:: 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.

Unknown directive type "data".

```
.. data:: inited
```

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

Unknown directive type "data".

```
.. data:: knownfiles

   .. index:: single: file; mime.types
```

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

Unknown directive type "data".

```
.. data:: 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 :file:`.tgz` extension is mapped to :file:`.tar.gz`
   to allow the encoding and type to be recognized separately.
```

Unknown directive type "data".

```
.. data:: encodings_map

   Dictionary mapping filename extensions to encoding types.
```

Unknown directive type "data".

```
.. data:: types_map

   Dictionary mapping filename extensions to MIME types.
```

Unknown directive type "data".

```
.. data:: 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 :class:`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 :mod:`mimetypes` module.

Unknown interpreted text role "class".

Unknown interpreted text role "mod".

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 :file:`mime.types`-style files into the database using the :meth:`read` or :meth:`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.

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

```
.. attribute:: MimeTypes.encodings_map

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

```
.. attribute:: MimeTypes.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 :data:`common_types` and
   :data:`types_map`.
```

```
.. attribute:: MimeTypes.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
:data:`common_types` and :data:`types_map`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, line 229)

Unknown directive type "method".

```
.. method:: MimeTypes.guess_extension(type, strict=True)

   Similar to the :func:`guess_extension` function, using the tables stored as part
   of the object.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, line 235)

Unknown directive type "method".

```
.. method:: MimeTypes.guess_type(url, strict=True)

   Similar to the :func:`guess_type` function, using the tables stored as part of
   the object.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, line 241)

Unknown directive type "method".

```
.. method:: MimeTypes.guess_all_extensions(type, strict=True)

   Similar to the :func:`guess_all_extensions` function, using the tables stored
   as part of the object.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, line 247)

Unknown directive type "method".

```
.. method:: MimeTypes.read(filename, strict=True)

   Load MIME information from a file named *filename*.  This uses :meth:`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.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, line 256)

Unknown directive type "method".

```
.. method:: MimeTypes.readfp(fp, strict=True)

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)mimetypes.rst`, line 265)

Unknown directive type "method".

```
.. method:: MimeTypes.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.

   .. versionadded:: 3.2
```