:mod:`urllib.parse` --- Parse URLs into components

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

Unknown directive type "module".

```
.. module:: urllib.parse
   :synopsis: Parse URLs into or assemble them from components.
```

Source code: :source:`Lib/urllib/parse.py`

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

Unknown interpreted text role "source".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 9)

Unknown directive type "index".

```
.. index::
   single: WWW
   single: World Wide Web
   single: URL
   pair: URL; parsing
   pair: relative; URL
```

This module defines a standard interface to break Uniform Resource Locator (URL) strings up in components (addressing scheme, network location, path etc.), to combine the components back into a URL string, and to convert a "relative URL" to an absolute URL given a "base URL."

The module has been designed to match the internet RFC on Relative Uniform Resource Locators. It supports the following URL schemes: `file`, `ftp`, `gopher`, `hdl`, `http`, `https`, `imap`, `mailto`, `mms`, `news`, `nntp`, `prospero`, `rsync`, `rtsp`, `rtspu`, `sftp`, `shttp`, `sip`, `sips`, `snews`, `svn`, `svn+ssh`, `telnet`, `wais`, `ws`, `wss`.

The :mod:`urllib.parse` module defines functions that fall into two broad categories: URL parsing and URL quoting. These are covered in detail in the following sections.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 30); *backlink*

Unknown interpreted text role "mod".

URL Parsing

The URL parsing functions focus on splitting a URL string into its components, or on combining URL components into a URL string.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 40)

Unknown directive type "function".

```
.. function:: urlparse(urlstring, scheme='', allow_fragments=True)

   Parse a URL into six components, returning a 6-item :term:`named tuple`.  This
   corresponds to the general structure of a URL:
   ``scheme://netloc/path;parameters?query#fragment``.
   Each tuple item is a string, possibly empty. The components are not broken up
   into smaller parts (for example, the network location is a single string), and %
   escapes are not expanded. The delimiters as shown above are not part of the
   result, except for a leading slash in the *path* component, which is retained if
   present.  For example:

   .. doctest::
      :options: +NORMALIZE_WHITESPACE

      >>> from urllib.parse import urlparse
      >>> urlparse("scheme://netloc/path;parameters?query#fragment")
      ParseResult(scheme='scheme', netloc='netloc', path='/path;parameters', params='',
                  query='query', fragment='fragment')
      >>> o = urlparse("http://docs.python.org:80/3/library/urllib.parse.html?"
      ...              "highlight=params#url-parsing")
      >>> o
      ParseResult(scheme='http', netloc='docs.python.org:80',
                  path='/3/library/urllib.parse.html', params='',
                  query='highlight=params', fragment='url-parsing')
      >>> o.scheme
      'http'
      >>> o.netloc
      'docs.python.org:80'
      >>> o.hostname
      'docs.python.org'
      >>> o.port
      80
      >>> o._replace(fragment="").geturl()
      'http://docs.python.org:80/3/library/urllib.parse.html?highlight=params'

   Following the syntax specifications in :rfc:`1808`, urlparse recognizes
   a netloc only if it is properly introduced by '//'.  Otherwise the
   input is presumed to be a relative URL and thus to start with
   a path component.

   .. doctest::
      :options: +NORMALIZE_WHITESPACE
```

```
   >>> from urllib.parse import urlparse
   >>> urlparse('//www.cwi.nl:80/%7Eguido/Python.html')
   ParseResult(scheme='', netloc='www.cwi.nl:80', path='/%7Eguido/Python.html',
               params='', query='', fragment='')
   >>> urlparse('www.cwi.nl/%7Eguido/Python.html')
   ParseResult(scheme='', netloc='', path='www.cwi.nl/%7Eguido/Python.html',
               params='', query='', fragment='')
   >>> urlparse('help/Python.html')
   ParseResult(scheme='', netloc='', path='help/Python.html', params='',
               query='', fragment='')

The *scheme* argument gives the default addressing scheme, to be
used only if the URL does not specify one.  It should be the same type
(text or bytes) as *urlstring*, except that the default value ``''`` is
always allowed, and is automatically converted to ``b''`` if appropriate.

If the *allow_fragments* argument is false, fragment identifiers are not
recognized.  Instead, they are parsed as part of the path, parameters
or query component, and :attr:`fragment` is set to the empty string in
the return value.

The return value is a :term:`named tuple`, which means that its items can
be accessed by index or as named attributes, which are:

+------------------+-------+-------------------------+-----------------------+
| Attribute        | Index | Value                   | Value if not present  |
+==================+=======+=========================+=======================+
| :attr:`scheme`   | 0     | URL scheme specifier    | *scheme* parameter    |
+------------------+-------+-------------------------+-----------------------+
| :attr:`netloc`   | 1     | Network location part   | empty string          |
+------------------+-------+-------------------------+-----------------------+
| :attr:`path`     | 2     | Hierarchical path       | empty string          |
+------------------+-------+-------------------------+-----------------------+
| :attr:`params`   | 3     | No longer used          | always an empty string|
+------------------+-------+-------------------------+-----------------------+
| :attr:`query`    | 4     | Query component         | empty string          |
+------------------+-------+-------------------------+-----------------------+
| :attr:`fragment` | 5     | Fragment identifier     | empty string          |
+------------------+-------+-------------------------+-----------------------+
| :attr:`username` |       | User name               | :const:`None`         |
+------------------+-------+-------------------------+-----------------------+
| :attr:`password` |       | Password                | :const:`None`         |
+------------------+-------+-------------------------+-----------------------+
| :attr:`hostname` |       | Host name (lower case)  | :const:`None`         |
+------------------+-------+-------------------------+-----------------------+
| :attr:`port`     |       | Port number as integer, | :const:`None`         |
|                  |       | if present              |                       |
+------------------+-------+-------------------------+-----------------------+

Reading the :attr:`port` attribute will raise a :exc:`ValueError` if
an invalid port is specified in the URL.  See section
:ref:`urlparse-result-object` for more information on the result object.

Unmatched square brackets in the :attr:`netloc` attribute will raise a
:exc:`ValueError`.

Characters in the :attr:`netloc` attribute that decompose under NFKC
normalization (as used by the IDNA encoding) into any of ``/``, ``?``,
``#``, ``@``, or ``:`` will raise a :exc:`ValueError`. If the URL is
decomposed before parsing, no error will be raised.

As is the case with all named tuples, the subclass has a few additional methods
and attributes that are particularly useful. One such method is :meth:`_replace`.
The :meth:`_replace` method will return a new ParseResult object replacing specified
fields with new values.

.. doctest::
   :options: +NORMALIZE_WHITESPACE

   >>> from urllib.parse import urlparse
   >>> u = urlparse('//www.cwi.nl:80/%7Eguido/Python.html')
   >>> u
   ParseResult(scheme='', netloc='www.cwi.nl:80', path='/%7Eguido/Python.html',
               params='', query='', fragment='')
   >>> u._replace(scheme='http')
   ParseResult(scheme='http', netloc='www.cwi.nl:80', path='/%7Eguido/Python.html',
               params='', query='', fragment='')


.. versionchanged:: 3.2
   Added IPv6 URL parsing capabilities.

.. versionchanged:: 3.3
   The fragment is now parsed for all URL schemes (unless *allow_fragment* is
   false), in accordance with :rfc:`3986`.  Previously, an allowlist of
   schemes that support fragments existed.

.. versionchanged:: 3.6
   Out-of-range port numbers now raise :exc:`ValueError`, instead of
   returning :const:`None`.

.. versionchanged:: 3.8
   Characters that affect netloc parsing under NFKC normalization will
   now raise :exc:`ValueError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 179)

Unknown directive type "function".

```
.. function:: parse_qs(qs, keep_blank_values=False, strict_parsing=False, encoding='utf-8', errors='replace', max_num_fiel

   Parse a query string given as a string argument (data of type
   :mimetype:`application/x-www-form-urlencoded`).  Data are returned as a
   dictionary.  The dictionary keys are the unique query variable names and the
   values are lists of values for each name.

   The optional argument *keep_blank_values* is a flag indicating whether blank
   values in percent-encoded queries should be treated as blank strings. A true value
   indicates that blanks should be retained as  blank strings.  The default false
   value indicates that blank values are to be ignored and treated as if they were
   not included.

   The optional argument *strict_parsing* is a flag indicating what to do with
```

```
   parsing errors.  If false (the default), errors are silently ignored.  If true,
   errors raise a :exc:`ValueError` exception.

   The optional *encoding* and *errors* parameters specify how to decode
   percent-encoded sequences into Unicode characters, as accepted by the
   :meth:`bytes.decode` method.

   The optional argument *max_num_fields* is the maximum number of fields to
   read. If set, then throws a :exc:`ValueError` if there are more than
   *max_num_fields* fields read.

   The optional argument *separator* is the symbol to use for separating the
   query arguments. It defaults to ``&``.

   Use the :func:`urllib.parse.urlencode` function (with the ``doseq``
   parameter set to ``True``) to convert such dictionaries into query
   strings.


   .. versionchanged:: 3.2
      Add *encoding* and *errors* parameters.

   .. versionchanged:: 3.8
      Added *max_num_fields* parameter.

   .. versionchanged:: 3.10
      Added *separator* parameter with the default value of ``&``. Python
      versions earlier than Python 3.10 allowed using both ``;`` and ``&`` as
      query parameter separator. This has been changed to allow only a single
      separator key, with ``&`` as the default separator.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 225)

Unknown directive type "function".

```
.. function:: parse_qsl(qs, keep_blank_values=False, strict_parsing=False, encoding='utf-8', errors='replace', max_num_fie

   Parse a query string given as a string argument (data of type
   :mimetype:`application/x-www-form-urlencoded`).  Data are returned as a list of
   name, value pairs.

   The optional argument *keep_blank_values* is a flag indicating whether blank
   values in percent-encoded queries should be treated as blank strings. A true value
   indicates that blanks should be retained as  blank strings.  The default false
   value indicates that blank values are to be ignored and treated as if they were
   not included.

   The optional argument *strict_parsing* is a flag indicating what to do with
   parsing errors.  If false (the default), errors are silently ignored.  If true,
   errors raise a :exc:`ValueError` exception.

   The optional *encoding* and *errors* parameters specify how to decode
   percent-encoded sequences into Unicode characters, as accepted by the
   :meth:`bytes.decode` method.

   The optional argument *max_num_fields* is the maximum number of fields to
   read. If set, then throws a :exc:`ValueError` if there are more than
   *max_num_fields* fields read.

   The optional argument *separator* is the symbol to use for separating the
   query arguments. It defaults to ``&``.

   Use the :func:`urllib.parse.urlencode` function to convert such lists of pairs into
   query strings.

   .. versionchanged:: 3.2
      Add *encoding* and *errors* parameters.

   .. versionchanged:: 3.8
      Added *max_num_fields* parameter.

   .. versionchanged:: 3.10
      Added *separator* parameter with the default value of ``&``. Python
      versions earlier than Python 3.10 allowed using both ``;`` and ``&`` as
      query parameter separator. This has been changed to allow only a single
      separator key, with ``&`` as the default separator.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 268)

Unknown directive type "function".

```
.. function:: urlunparse(parts)

   Construct a URL from a tuple as returned by ``urlparse()``. The *parts*
   argument can be any six-item iterable. This may result in a slightly
   different, but equivalent URL, if the URL that was parsed originally had
   unnecessary delimiters (for example, a ``?`` with an empty query; the RFC
   states that these are equivalent).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 277)

Unknown directive type "function".

```
.. function:: urlsplit(urlstring, scheme='', allow_fragments=True)

   This is similar to :func:`urlparse`, but does not split the params from the URL.
   This should generally be used instead of :func:`urlparse` if the more recent URL
   syntax allowing parameters to be applied to each segment of the *path* portion
   of the URL (see :rfc:`2396`) is wanted.  A separate function is needed to
   separate the path segments and parameters.  This function returns a 5-item
   :term:`named tuple`::

      (addressing scheme, network location, path, query, fragment identifier).

   The return value is a :term:`named tuple`, its items can be accessed by index
```

```
or as named attributes:

+------------------+-------+-------------------------+----------------------+
| Attribute        | Index | Value                   | Value if not present |
+==================+=======+=========================+======================+
| :attr:`scheme`   | 0     | URL scheme specifier    | *scheme* parameter   |
+------------------+-------+-------------------------+----------------------+
| :attr:`netloc`   | 1     | Network location part   | empty string         |
+------------------+-------+-------------------------+----------------------+
| :attr:`path`     | 2     | Hierarchical path       | empty string         |
+------------------+-------+-------------------------+----------------------+
| :attr:`query`    | 3     | Query component         | empty string         |
+------------------+-------+-------------------------+----------------------+
| :attr:`fragment` | 4     | Fragment identifier     | empty string         |
+------------------+-------+-------------------------+----------------------+
| :attr:`username` |       | User name               | :const:`None`        |
+------------------+-------+-------------------------+----------------------+
| :attr:`password` |       | Password                | :const:`None`        |
+------------------+-------+-------------------------+----------------------+
| :attr:`hostname` |       | Host name (lower case)  | :const:`None`        |
+------------------+-------+-------------------------+----------------------+
| :attr:`port`     |       | Port number as integer, | :const:`None`        |
|                  |       | if present              |                      |
+------------------+-------+-------------------------+----------------------+

Reading the :attr:`port` attribute will raise a :exc:`ValueError` if
an invalid port is specified in the URL.  See section
:ref:`urlparse-result-object` for more information on the result object.

Unmatched square brackets in the :attr:`netloc` attribute will raise a
:exc:`ValueError`.

Characters in the :attr:`netloc` attribute that decompose under NFKC
normalization (as used by the IDNA encoding) into any of ``/``, ``?``,
``#``, ``@``, or ``:`` will raise a :exc:`ValueError`. If the URL is
decomposed before parsing, no error will be raised.

Following the `WHATWG spec`_ that updates RFC 3986, ASCII newline
``\n``, ``\r`` and tab ``\t`` characters are stripped from the URL.

.. versionchanged:: 3.6
   Out-of-range port numbers now raise :exc:`ValueError`, instead of
   returning :const:`None`.

.. versionchanged:: 3.8
   Characters that affect netloc parsing under NFKC normalization will
   now raise :exc:`ValueError`.

.. versionchanged:: 3.10
   ASCII newline and tab characters are stripped from the URL.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 342)

Unknown directive type "function".

```
.. function:: urlunsplit(parts)

   Combine the elements of a tuple as returned by :func:`urlsplit` into a
   complete URL as a string. The *parts* argument can be any five-item
   iterable. This may result in a slightly different, but equivalent URL, if the
   URL that was parsed originally had unnecessary delimiters (for example, a ?
   with an empty query; the RFC states that these are equivalent).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 351)

Unknown directive type "function".

```
.. function:: urljoin(base, url, allow_fragments=True)

   Construct a full ("absolute") URL by combining a "base URL" (*base*) with
   another URL (*url*).  Informally, this uses components of the base URL, in
   particular the addressing scheme, the network location and (part of) the
   path, to provide missing components in the relative URL.  For example:

      >>> from urllib.parse import urljoin
      >>> urljoin('http://www.cwi.nl/%7Eguido/Python.html', 'FAQ.html')
      'http://www.cwi.nl/%7Eguido/FAQ.html'

   The *allow_fragments* argument has the same meaning and default as for
   :func:`urlparse`.

   .. note::

      If *url* is an absolute URL (that is, it starts with ``//`` or ``scheme://``),
      the *url*'s hostname and/or scheme will be present in the result.  For example:

      .. doctest::

         >>> urljoin('http://www.cwi.nl/%7Eguido/Python.html',
         ...         '//www.python.org/%7Eguido')
         'http://www.python.org/%7Eguido'

      If you do not want that behavior, preprocess the *url* with :func:`urlsplit` and
      :func:`urlunsplit`, removing possible *scheme* and *netloc* parts.


   .. versionchanged:: 3.5

      Behavior updated to match the semantics defined in :rfc:`3986`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 385)

Unknown directive type "function".

```
.. function:: urldefrag(url)
```

```
   If *url* contains a fragment identifier, return a modified version of *url*
   with no fragment identifier, and the fragment identifier as a separate
   string.  If there is no fragment identifier in *url*, return *url* unmodified
   and an empty string.

   The return value is a :term:`named tuple`, its items can be accessed by index
   or as named attributes:

   +------------------+-------+-------------------------+----------------------+
   | Attribute        | Index | Value                   | Value if not present |
   +==================+=======+=========================+======================+
   | :attr:`url`      | 0     | URL with no fragment    | empty string         |
   +------------------+-------+-------------------------+----------------------+
   | :attr:`fragment` | 1     | Fragment identifier     | empty string         |
   +------------------+-------+-------------------------+----------------------+

   See section :ref:`urlparse-result-object` for more information on the result
   object.

   .. versionchanged:: 3.2
      Result is a structured object rather than a simple 2-tuple.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 409)

Unknown directive type "function".

```
.. function:: unwrap(url)

   Extract the url from a wrapped URL (that is, a string formatted as
   ``<URL:scheme://host/path>``, ``<scheme://host/path>``, ``URL:scheme://host/path``
   or ``scheme://host/path``). If *url* is not a wrapped URL, it is returned
   without changes.
```

Parsing ASCII Encoded Bytes

The URL parsing functions were originally designed to operate on character strings only. In practice, it is useful to be able to manipulate properly quoted and encoded URLs as sequences of ASCII bytes. Accordingly, the URL parsing functions in this module all operate on :class:`bytes` and :class:`bytearray` objects in addition to :class:`str` objects.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 421); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 421); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 421); *backlink*

Unknown interpreted text role "class".

If :class:`str` data is passed in, the result will also contain only :class:`str` data. If :class:`bytes` or :class:`bytearray` data is passed in, the result will contain only :class:`bytes` data.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 427); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 427); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 427); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 427); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 427); *backlink*

Unknown interpreted text role "class".

Attempting to mix :class:`str` data with :class:`bytes` or :class:`bytearray` in a single function call will result in a :exc:`TypeError` being raised, while attempting to pass in non-ASCII byte values will trigger :exc:`UnicodeDecodeError`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 431); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 431); *backlink*

Unknown interpreted text role "class".

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

main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 431); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 431); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 431); *backlink*

Unknown interpreted text role "exc".

To support easier conversion of result objects between :class:`str` and :class:`bytes`, all return values from URL parsing functions provide either an :meth:`encode` method (when the result contains :class:`str` data) or a :meth:`decode` method (when the result contains :class:`bytes` data). The signatures of these methods match those of the corresponding :class:`str` and :class:`bytes` methods (except that the default encoding is `'ascii'` rather than `'utf-8'`). Each produces a value of a corresponding type that contains either :class:`bytes` data (for :meth:`encode` methods) or :class:`str` data (for :meth:`decode` methods).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 436); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 436); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 436); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 436); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 436); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 436); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 436); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 436); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 436); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 436); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 436); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 436); *backlink*

Unknown interpreted text role "meth".

Applications that need to operate on potentially improperly quoted URLs that may contain non-ASCII data will need to do their own decoding from bytes to characters before invoking the URL parsing methods.

The behaviour described in this section applies only to the URL parsing functions. The URL quoting functions use their own rules when producing or consuming byte sequences as detailed in the documentation of the individual URL quoting functions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 456)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.2
   URL parsing functions now accept ASCII encoded byte sequences
```

Structured Parse Results

The result objects from the :func:`urlparse`, :func:`urlsplit` and :func:`urldefrag` functions are subclasses of the :class:`tuple` type. These subclasses add the attributes listed in the documentation for those functions, the encoding and decoding support described in the previous section, as well as an additional method:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 465); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 465); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 465); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 465); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 471)

Unknown directive type "method".

```
.. method:: urllib.parse.SplitResult.geturl()

   Return the re-combined version of the original URL as a string. This may
   differ from the original URL in that the scheme may be normalized to lower
   case and empty components may be dropped. Specifically, empty parameters,
   queries, and fragment identifiers will be removed.

   For :func:`urldefrag` results, only empty fragment identifiers will be removed.
   For :func:`urlsplit` and :func:`urlparse` results, all noted changes will be
   made to the URL returned by this method.

   The result of this method remains unchanged if passed back through the original
   parsing function:

      >>> from urllib.parse import urlsplit
      >>> url = 'HTTP://www.Python.org/doc/#'
      >>> r1 = urlsplit(url)
      >>> r1.geturl()
      'http://www.Python.org/doc/'
      >>> r2 = urlsplit(r1.geturl())
      >>> r2.geturl()
      'http://www.Python.org/doc/'
```

The following classes provide the implementations of the structured parse results when operating on :class:`str` objects:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 495); *backlink*

Unknown interpreted text role "class".

Concrete class for :func:`urldefrag` results containing :class:`str` data. The :meth:`encode` method returns a :class:`DefragResultBytes` instance.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 500); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 500); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 500); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 500); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 504)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
```

Concrete class for :func:`urlparse` results containing :class:`str` data. The :meth:`encode` method returns a :class:`ParseResultBytes` instance.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 508); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 508); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 508); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 508); *backlink*

Unknown interpreted text role "class".

Concrete class for :func:`urlsplit` results containing :class:`str` data. The :meth:`encode` method returns a :class:`SplitResultBytes` instance.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 514); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 514); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 514); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 514); *backlink*

Unknown interpreted text role "class".

The following classes provide the implementations of the parse results when operating on :class:`bytes` or :class:`bytearray` objects:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 519); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 519); *backlink*

Unknown interpreted text role "class".

Concrete class for :func:`urldefrag` results containing :class:`bytes` data. The :meth:`decode` method returns a :class:`DefragResult` instance.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 524); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 524); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 524); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 524); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 528)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
```

Concrete class for :func:`urlparse` results containing :class:`bytes` data. The :meth:`decode` method returns a :class:`ParseResult` instance.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 532); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 532); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 532); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 532); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 536)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
```

Concrete class for :func:`urlsplit` results containing :class:`bytes` data. The :meth:`decode` method returns a :class:`SplitResult` instance.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 540); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 540); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 540); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 540); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 544)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
```

URL Quoting

The URL quoting functions focus on taking program data and making it safe for use as URL components by quoting special characters and appropriately encoding non-ASCII text. They also support reversing these operations to recreate the original data from the contents of a URL component if that task isn't already covered by the URL parsing functions above.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 556)

Unknown directive type "function".

```
.. function:: quote(string, safe='/', encoding=None, errors=None)

   Replace special characters in *string* using the ``%xx`` escape. Letters,
   digits, and the characters ``'_.-~'`` are never quoted. By default, this
   function is intended for quoting the path section of a URL. The optional
   *safe* parameter specifies additional ASCII characters that should not be
   quoted --- its default value is ``'/'``.

   *string* may be either a :class:`str` or a :class:`bytes` object.

   .. versionchanged:: 3.7
      Moved from :rfc:`2396` to :rfc:`3986` for quoting URL strings. "~" is now
      included in the set of unreserved characters.

   The optional *encoding* and *errors* parameters specify how to deal with
   non-ASCII characters, as accepted by the :meth:`str.encode` method.
   *encoding* defaults to ``'utf-8'``.
   *errors* defaults to ``'strict'``, meaning unsupported characters raise a
   :class:`UnicodeEncodeError`.
   *encoding* and *errors* must not be supplied if *string* is a
   :class:`bytes`, or a :class:`TypeError` is raised.

   Note that ``quote(string, safe, encoding, errors)`` is equivalent to
   ``quote_from_bytes(string.encode(encoding, errors), safe)``.

   Example: ``quote('/El NiÃ±o/')`` yields ``'/El%20Ni%C3%B1o/'``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst, line 584)

Unknown directive type "function".

```
.. function:: quote_plus(string, safe='', encoding=None, errors=None)

   Like :func:`quote`, but also replace spaces with plus signs, as required for
   quoting HTML form values when building up a query string to go into a URL.
   Plus signs in the original string are escaped unless they are included in
   *safe*.  It also does not have *safe* default to ``'/'``.
```

```
   Example: ``quote_plus('/El NiÃ±o/')`` yields ``'%2FEl+Ni%C3%B1o%2F'``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 594)

Unknown directive type "function".

```
.. function:: quote_from_bytes(bytes, safe='/')

   Like :func:`quote`, but accepts a :class:`bytes` object rather than a
   :class:`str`, and does not perform string-to-bytes encoding.

   Example: ``quote_from_bytes(b'a&\xef')`` yields
   ``'a%26%EF'``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 603)

Unknown directive type "function".

```
.. function:: unquote(string, encoding='utf-8', errors='replace')

   Replace ``%xx`` escapes with their single-character equivalent.
   The optional *encoding* and *errors* parameters specify how to decode
   percent-encoded sequences into Unicode characters, as accepted by the
   :meth:`bytes.decode` method.

   *string* may be either a :class:`str` or a :class:`bytes` object.

   *encoding* defaults to ``'utf-8'``.
   *errors* defaults to ``'replace'``, meaning invalid sequences are replaced
   by a placeholder character.

   Example: ``unquote('/El%20Ni%C3%B1o/')`` yields ``'/El NiÃ±o/'``.

   .. versionchanged:: 3.9
      *string* parameter supports bytes and str objects (previously only str).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 624)

Unknown directive type "function".

```
.. function:: unquote_plus(string, encoding='utf-8', errors='replace')

   Like :func:`unquote`, but also replace plus signs with spaces, as required
   for unquoting HTML form values.

   *string* must be a :class:`str`.

   Example: ``unquote_plus('/El+Ni%C3%B1o/')`` yields ``'/El NiÃ±o/'``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 634)

Unknown directive type "function".

```
.. function:: unquote_to_bytes(string)

   Replace ``%xx`` escapes with their single-octet equivalent, and return a
   :class:`bytes` object.

   *string* may be either a :class:`str` or a :class:`bytes` object.

   If it is a :class:`str`, unescaped non-ASCII characters in *string*
   are encoded into UTF-8 bytes.

   Example: ``unquote_to_bytes('a%26%EF')`` yields ``b'a&\xef'``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)urllib.parse.rst, line 647)

Unknown directive type "function".

```
.. function:: urlencode(query, doseq=False, safe='', encoding=None, \
                        errors=None, quote_via=quote_plus)

   Convert a mapping object or a sequence of two-element tuples, which may
   contain :class:`str` or :class:`bytes` objects, to a percent-encoded ASCII
   text string.  If the resultant string is to be used as a *data* for POST
   operation with the :func:`~urllib.request.urlopen` function, then
   it should be encoded to bytes, otherwise it would result in a
   :exc:`TypeError`.

   The resulting string is a series of ``key=value`` pairs separated by ``'&'``
   characters, where both *key* and *value* are quoted using the *quote_via*
   function.  By default, :func:`quote_plus` is used to quote the values, which
   means spaces are quoted as a ``'+'`` character and '/' characters are
   encoded as ``%2F``, which follows the standard for GET requests
   (``application/x-www-form-urlencoded``).  An alternate function that can be
   passed as *quote_via* is :func:`quote`, which will encode spaces as ``%20``
   and not encode '/' characters.  For maximum control of what is quoted, use
   ``quote`` and specify a value for *safe*.

   When a sequence of two-element tuples is used as the *query*
   argument, the first element of each tuple is a key and the second is a
   value. The value element in itself can be a sequence and in that case, if
   the optional parameter *doseq* evaluates to ``True``, individual
   ``key=value`` pairs separated by ``'&'`` are generated for each element of
   the value sequence for the key.  The order of parameters in the encoded
```

```
   string will match the order of parameter tuples in the sequence.

   The *safe*, *encoding*, and *errors* parameters are passed down to
   *quote_via* (the *encoding* and *errors* parameters are only passed
   when a query element is a :class:`str`).

   To reverse this encoding process, :func:`parse_qs` and :func:`parse_qsl` are
   provided in this module to parse query strings into Python data structures.

   Refer to :ref:`urllib examples <urllib-examples>` to find out how the
   :func:`urllib.parse.urlencode` method can be used for generating the query
   string of a URL or data for a POST request.

   .. versionchanged:: 3.2
      *query* supports bytes and string objects.

   .. versionadded:: 3.5
      *quote_via* parameter.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)urllib.parse.rst`, line 693)

Unknown directive type "seealso".

```
.. seealso::

   `WHATWG`_ -  URL Living standard
      Working Group for the URL Standard that defines URLs, domains, IP addresses, the
      application/x-www-form-urlencoded format, and their API.

   :rfc:`3986` - Uniform Resource Identifiers
      This is the current standard (STD66). Any changes to urllib.parse module
      should conform to this. Certain deviations could be observed, which are
      mostly for backward compatibility purposes and for certain de-facto
      parsing requirements as commonly observed in major browsers.

   :rfc:`2732` - Format for Literal IPv6 Addresses in URL's.
      This specifies the parsing requirements of IPv6 URLs.

   :rfc:`2396` - Uniform Resource Identifiers (URI): Generic Syntax
      Document describing the generic syntactic requirements for both Uniform Resource
      Names (URNs) and Uniform Resource Locators (URLs).

   :rfc:`2368` - The mailto URL scheme.
      Parsing requirements for mailto URL schemes.

   :rfc:`1808` - Relative Uniform Resource Locators
      This Request For Comments includes the rules for joining an absolute and a
      relative URL, including a fair number of "Abnormal Examples" which govern the
      treatment of border cases.

   :rfc:`1738` - Uniform Resource Locators (URL)
      This specifies the formal syntax and semantics of absolute URLs.
```