:mod:`smtplib` --- SMTP protocol client

> **System Message: ERROR/3 (**`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)smtplib.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)smtplib.rst`**, line 4)**
>
> Unknown directive type "module".
>
> ```
> .. module:: smtplib
>    :synopsis: SMTP protocol client (requires sockets).
> ```

> **System Message: ERROR/3 (**`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)smtplib.rst`**, line 7)**
>
> Unknown directive type "sectionauthor".
>
> ```
> .. sectionauthor:: Eric S. Raymond <esr@snark.thyrsus.com>
> ```

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

> **System Message: ERROR/3 (**`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)smtplib.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)smtplib.rst`**, line 11)**
>
> Unknown directive type "index".
>
> ```
> .. index::
>    pair: SMTP; protocol
>    single: Simple Mail Transfer Protocol
> ```

The :mod:`smtplib` module defines an SMTP client session object that can be used to send mail to any internet machine with an SMTP or ESMTP listener daemon. For details of SMTP and ESMTP operation, consult RFC 821 (Simple Mail Transfer Protocol) and RFC 1869 (SMTP Service Extensions).

> **System Message: ERROR/3 (**`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)smtplib.rst`**, line 17);** *backlink*
>
> Unknown interpreted text role "mod".

An :class:`SMTP` instance encapsulates an SMTP connection. It has methods that support a full repertoire of SMTP and ESMTP operations. If the optional host and port parameters are given, the SMTP :meth:`connect` method is called with those parameters during initialization. If specified, *local_hostname* is used as the FQDN of the local host in the HELO/EHLO command. Otherwise, the local hostname is found using :func:`socket.getfqdn`. If the :meth:`connect` call returns anything other than a success code, an :exc:`SMTPConnectError` is raised. The optional *timeout* parameter specifies a timeout in seconds for blocking operations like the connection attempt (if not specified, the global default timeout setting will be used). If the timeout expires, :exc:`TimeoutError` is raised. The optional source_address parameter allows binding to some specific source address in a machine with multiple network interfaces, and/or to some specific source TCP port. It takes a 2-tuple (host, port), for the socket to bind to as its source address before connecting. If omitted (or if host or port are `''` and/or 0 respectively) the OS default behavior will be used.

> **System Message: ERROR/3 (**`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)smtplib.rst`**, line 25);** *backlink*
>
> Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)smtplib.rst`, **line 25**); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)smtplib.rst`, **line 25**); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)smtplib.rst`, **line 25**); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)smtplib.rst`, **line 25**); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)smtplib.rst`, **line 25**); *backlink*

Unknown interpreted text role "exc".

For normal use, you should only require the initialization/connect, :meth:`sendmail`, and :meth:`SMTP.quit` methods. An example is included below.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)smtplib.rst`, **line 43**); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)smtplib.rst`, **line 43**); *backlink*

Unknown interpreted text role "meth".

The :class:`SMTP` class supports the :keyword:`with` statement. When used like this, the SMTP `QUIT` command is issued automatically when the :keyword:`!with` statement exits. E.g.:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)smtplib.rst`, **line 47**); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)smtplib.rst`, **line 47**); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)smtplib.rst`, **line 47**); *backlink*

Unknown interpreted text role "keyword".

```
>>> from smtplib import SMTP
>>> with SMTP("domain.org") as smtp:
...     smtp.noop()
...
(250, b'Ok')
>>>
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)smtplib.rst`, **line 58**)

Unknown directive type "audit-event".

```
.. audit-event:: smtplib.send self,data smtplib.SMTP

   All commands will raise an :ref:`auditing event <auditing>`
   ``smtplib.SMTP.send`` with arguments ``self`` and ``data``,
   where ``data`` is the bytes about to be sent to the remote host.
```

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   Support for the :keyword:`with` statement was added.
```

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   source_address argument was added.
```

Unknown directive type "versionadded".

```
.. versionadded:: 3.5
   The SMTPUTF8 extension (:rfc:`6531`) is now supported.
```

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.9
   If the *timeout* parameter is set to be zero, it will raise a
   :class:`ValueError` to prevent the creation of a non-blocking socket
```

Invalid class attribute value for "class" directive: "SMTP_SSL(host=", port=0, local_hostname=None, keyfile=None, \ certfile=None [, timeout], context=None, \ source_address=None)".

```
.. class:: SMTP_SSL(host='', port=0, local_hostname=None, keyfile=None, \
                    certfile=None [, timeout], context=None, \
                    source_address=None)

   An :class:`SMTP_SSL` instance behaves exactly the same as instances of
   :class:`SMTP`. :class:`SMTP_SSL` should be used for situations where SSL is
   required from the beginning of the connection and using :meth:`starttls` is
   not appropriate. If *host* is not specified, the local host is used. If
   *port* is zero, the standard SMTP-over-SSL port (465) is used.  The optional
   arguments *local_hostname*, *timeout* and *source_address* have the same
   meaning as they do in the :class:`SMTP` class. *context*, also optional,
   can contain a :class:`~ssl.SSLContext` and allows configuring various
   aspects of the secure connection.  Please read :ref:`ssl-security` for
   best practices.

   *keyfile* and *certfile* are a legacy alternative to *context*, and can
   point to a PEM formatted private key and certificate chain file for the
   SSL connection.

   .. versionchanged:: 3.3
      *context* was added.

   .. versionchanged:: 3.3
      source_address argument was added.
```

```
.. versionchanged:: 3.4
   The class now supports hostname check with
   :attr:`ssl.SSLContext.check_hostname` and *Server Name Indication* (see
   :data:`ssl.HAS_SNI`).

.. deprecated:: 3.6

   *keyfile* and *certfile* are deprecated in favor of *context*.
   Please use :meth:`ssl.SSLContext.load_cert_chain` instead, or let
   :func:`ssl.create_default_context` select the system's trusted CA
   certificates for you.

.. versionchanged:: 3.9
   If the *timeout* parameter is set to be zero, it will raise a
   :class:`ValueError` to prevent the creation of a non-blocking socket
```

```
.. class:: LMTP(host='', port=LMTP_PORT, local_hostname=None, \
              source_address=None[, timeout])

   The LMTP protocol, which is very similar to ESMTP, is heavily based on the
   standard SMTP client. It's common to use Unix sockets for LMTP, so our
   :meth:`connect` method must support that as well as a regular host:port
   server. The optional arguments local_hostname and source_address have the
   same meaning as they do in the :class:`SMTP` class. To specify a Unix
   socket, you must use an absolute path for *host*, starting with a '/'.

   Authentication is supported, using the regular SMTP mechanism. When using a
   Unix socket, LMTP generally don't support or require any authentication, but
   your mileage might vary.

   .. versionchanged:: 3.9
      The optional *timeout* parameter was added.
```

A nice selection of exceptions is defined as well:

```
.. exception:: SMTPException

   Subclass of :exc:`OSError` that is the base exception class for all
   the other exceptions provided by this module.

   .. versionchanged:: 3.4
      SMTPException became subclass of :exc:`OSError`
```

```
.. exception:: SMTPServerDisconnected

   This exception is raised when the server unexpectedly disconnects, or when an
   attempt is made to use the :class:`SMTP` instance before connecting it to a
   server.
```

```
.. exception:: SMTPResponseException

   Base class for all exceptions that include an SMTP error code. These exceptions
   are generated in some instances when the SMTP server returns an error code.  The
   error code is stored in the :attr:`smtp_code` attribute of the error, and the
   :attr:`smtp_error` attribute is set to the error message.
```

```
.. exception:: SMTPSenderRefused

   Sender address refused.  In addition to the attributes set by on all
   :exc:`SMTPResponseException` exceptions, this sets 'sender' to the string that
   the SMTP server refused.
```

```
.. exception:: SMTPRecipientsRefused

   All recipient addresses refused.  The errors for each recipient are accessible
   through the attribute :attr:`recipients`, which is a dictionary of exactly the
   same sort as :meth:`SMTP.sendmail` returns.
```

```
.. exception:: SMTPDataError

   The SMTP server refused to accept the message data.
```

```
.. exception:: SMTPConnectError

   Error occurred during establishment of a connection  with the server.
```

```
.. exception:: SMTPHeloError

   The server refused our ``HELO`` message.
```

```
.. exception:: SMTPNotSupportedError
```

```
     The command or option attempted is not supported by the server.

     .. versionadded:: 3.5
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)smtplib.rst, line 199)

Unknown directive type "exception".

```
   .. exception:: SMTPAuthenticationError

      SMTP authentication went wrong.  Most probably the server didn't accept the
      username/password combination provided.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)smtplib.rst, line 205)

Unknown directive type "seealso".

```
   .. seealso::

      :rfc:`821` - Simple Mail Transfer Protocol
         Protocol definition for SMTP.  This document covers the model, operating
         procedure, and protocol details for SMTP.

      :rfc:`1869` - SMTP Service Extensions
         Definition of the ESMTP extensions for SMTP.  This describes a framework for
         extending SMTP with new commands, supporting dynamic discovery of the commands
         provided by the server, and defines a few additional commands.
```

SMTP Objects

An :class:`SMTP` instance has the following methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)smtplib.rst, line 222); *backlink*

Unknown interpreted text role "class".

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

Unknown directive type "method".

```
   .. method:: SMTP.set_debuglevel(level)

      Set the debug output level.  A value of 1 or ``True`` for *level* results in
      debug messages for connection and for all messages sent to and received from
      the server.  A value of 2 for *level* results in these messages being
      timestamped.

      .. versionchanged:: 3.5 Added debuglevel 2.
```

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

Unknown directive type "method".

```
   .. method:: SMTP.docmd(cmd, args='')

      Send a command *cmd* to the server.  The optional argument *args* is simply
      concatenated to the command, separated by a space.

      This returns a 2-tuple composed of a numeric response code and the actual
      response line (multiline responses are joined into one long line.)

      In normal operation it should not be necessary to call this method explicitly.
```

It is used to implement other methods and may be useful for testing private
extensions.

If the connection to the server is lost while waiting for the reply,
:exc:`SMTPServerDisconnected` will be raised.

Unknown directive type "method".

```
.. method:: SMTP.connect(host='localhost', port=0)

   Connect to a host on a given port.  The defaults are to connect to the local
   host at the standard SMTP port (25). If the hostname ends with a colon (``':'``)
   followed by a number, that suffix will be stripped off and the number
   interpreted as the port number to use. This method is automatically invoked by
   the constructor if a host is specified during instantiation.  Returns a
   2-tuple of the response code and message sent by the server in its
   connection response.

   .. audit-event:: smtplib.connect self,host,port smtplib.SMTP.connect
```

Unknown directive type "method".

```
.. method:: SMTP.helo(name='')

   Identify yourself to the SMTP server using ``HELO``.  The hostname argument
   defaults to the fully qualified domain name of the local host.
   The message returned by the server is stored as the :attr:`helo_resp` attribute
   of the object.

   In normal operation it should not be necessary to call this method explicitly.
   It will be implicitly called by the :meth:`sendmail` when necessary.
```

Unknown directive type "method".

```
.. method:: SMTP.ehlo(name='')

   Identify yourself to an ESMTP server using ``EHLO``.  The hostname argument
   defaults to the fully qualified domain name of the local host.  Examine the
   response for ESMTP option and store them for use by :meth:`has_extn`.
   Also sets several informational attributes: the message returned by
   the server is stored as the :attr:`ehlo_resp` attribute, :attr:`does_esmtp`
   is set to ``True`` or ``False`` depending on whether the server supports
   ESMTP, and :attr:`esmtp_features` will be a dictionary containing the names
   of the SMTP service extensions this server supports, and their parameters
   (if any).

   Unless you wish to use :meth:`has_extn` before sending mail, it should not be
   necessary to call this method explicitly.  It will be implicitly called by
   :meth:`sendmail` when necessary.
```

Unknown directive type "method".

```
.. method:: SMTP.ehlo_or_helo_if_needed()

   This method calls :meth:`ehlo` and/or :meth:`helo` if there has been no
   previous ``EHLO`` or ``HELO`` command this session.  It tries ESMTP ``EHLO``
   first.
```

```
:exc:`SMTPHeloError`
   The server didn't reply properly to the ``HELO`` greeting.
```

```
.. method:: SMTP.has_extn(name)

   Return :const:`True` if *name* is in the set of SMTP service extensions returned
   by the server, :const:`False` otherwise. Case is ignored.
```

```
.. method:: SMTP.verify(address)

   Check the validity of an address on this server using SMTP ``VRFY``. Returns a
   tuple consisting of code 250 and a full :rfc:`822` address (including human
   name) if the user address is valid. Otherwise returns an SMTP error code of 400
   or greater and an error string.

   .. note::

      Many sites disable SMTP ``VRFY`` in order to foil spammers.
```

```
.. method:: SMTP.login(user, password, *, initial_response_ok=True)

   Log in on an SMTP server that requires authentication. The arguments are the
   username and the password to authenticate with. If there has been no previous
   ``EHLO`` or ``HELO`` command this session, this method tries ESMTP ``EHLO``
   first. This method will return normally if the authentication was successful, or
   may raise the following exceptions:

   :exc:`SMTPHeloError`
      The server didn't reply properly to the ``HELO`` greeting.

   :exc:`SMTPAuthenticationError`
      The server didn't accept the username/password combination.

   :exc:`SMTPNotSupportedError`
      The ``AUTH`` command is not supported by the server.

   :exc:`SMTPException`
      No suitable authentication method was found.

   Each of the authentication methods supported by :mod:`smtplib` are tried in
   turn if they are advertised as supported by the server.  See :meth:`auth`
   for a list of supported authentication methods.  *initial_response_ok* is
   passed through to :meth:`auth`.

   Optional keyword argument *initial_response_ok* specifies whether, for
   authentication methods that support it, an "initial response" as specified
   in :rfc:`4954` can be sent along with the ``AUTH`` command, rather than
   requiring a challenge/response.

   .. versionchanged:: 3.5
      :exc:`SMTPNotSupportedError` may be raised, and the
      *initial_response_ok* parameter was added.
```

Unknown directive type "method".

```
.. method:: SMTP.auth(mechanism, authobject, *, initial_response_ok=True)

   Issue an ``SMTP`` ``AUTH`` command for the specified authentication
   *mechanism*, and handle the challenge response via *authobject*.

   *mechanism* specifies which authentication mechanism is to
   be used as argument to the ``AUTH`` command; the valid values are
   those listed in the ``auth`` element of :attr:`esmtp_features`.

   *authobject* must be a callable object taking an optional single argument:

     data = authobject(challenge=None)

   If optional keyword argument *initial_response_ok* is true,
   ``authobject()`` will be called first with no argument.  It can return the
   :rfc:`4954` "initial response" ASCII ``str`` which will be encoded and sent with
   the ``AUTH`` command as below.  If the ``authobject()`` does not support an
   initial response (e.g. because it requires a challenge), it should return
   ``None`` when called with ``challenge=None``.  If *initial_response_ok* is
   false, then ``authobject()`` will not be called first with ``None``.

   If the initial response check returns ``None``, or if *initial_response_ok* is
   false, ``authobject()`` will be called to process the server's challenge
   response; the *challenge* argument it is passed will be a ``bytes``.  It
   should return ASCII ``str`` *data* that will be base64 encoded and sent to the
   server.

   The ``SMTP`` class provides ``authobjects`` for the ``CRAM-MD5``, ``PLAIN``,
   and ``LOGIN`` mechanisms; they are named ``SMTP.auth_cram_md5``,
   ``SMTP.auth_plain``, and ``SMTP.auth_login`` respectively.  They all require
   that the ``user`` and ``password`` properties of the ``SMTP`` instance are
   set to appropriate values.

   User code does not normally need to call ``auth`` directly, but can instead
   call the :meth:`login` method, which will try each of the above mechanisms
   in turn, in the order listed.  ``auth`` is exposed to facilitate the
   implementation of authentication methods not (or not yet) supported
   directly by :mod:`smtplib`.

   .. versionadded:: 3.5
```

Unknown directive type "method".

```
.. method:: SMTP.starttls(keyfile=None, certfile=None, context=None)

   Put the SMTP connection in TLS (Transport Layer Security) mode.  All SMTP
   commands that follow will be encrypted.  You should then call :meth:`ehlo`
   again.

   If *keyfile* and *certfile* are provided, they are used to create an
   :class:`ssl.SSLContext`.

   Optional *context* parameter is an :class:`ssl.SSLContext` object; This is
   an alternative to using a keyfile and a certfile and if specified both
   *keyfile* and *certfile* should be ``None``.

   If there has been no previous ``EHLO`` or ``HELO`` command this session,
   this method tries ESMTP ``EHLO`` first.

   .. deprecated:: 3.6

       *keyfile* and *certfile* are deprecated in favor of *context*.
       Please use :meth:`ssl.SSLContext.load_cert_chain` instead, or let
       :func:`ssl.create_default_context` select the system's trusted CA
       certificates for you.

   :exc:`SMTPHeloError`
      The server didn't reply properly to the ``HELO`` greeting.

   :exc:`SMTPNotSupportedError`
     The server does not support the STARTTLS extension.

   :exc:`RuntimeError`
```

```
       SSL/TLS support is not available to your Python interpreter.

   .. versionchanged:: 3.3
      *context* was added.

   .. versionchanged:: 3.4
      The method now supports hostname check with
      :attr:`SSLContext.check_hostname` and *Server Name Indicator* (see
      :data:`~ssl.HAS_SNI`).

   .. versionchanged:: 3.5
      The error raised for lack of STARTTLS support is now the
      :exc:`SMTPNotSupportedError` subclass instead of the base
      :exc:`SMTPException`.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)smtplib.rst,` **line 441)**

Unknown directive type "method".

```
   .. method:: SMTP.sendmail(from_addr, to_addrs, msg, mail_options=(), rcpt_options=())

      Send mail.  The required arguments are an :rfc:`822` from-address string, a list
      of :rfc:`822` to-address strings (a bare string will be treated as a list with 1
      address), and a message string.  The caller may pass a list of ESMTP options
      (such as ``8bitmime``) to be used in ``MAIL FROM`` commands as *mail_options*.
      ESMTP options (such as ``DSN`` commands) that should be used with all ``RCPT``
      commands can be passed as *rcpt_options*.  (If you need to use different ESMTP
      options to different recipients you have to use the low-level methods such as
      :meth:`mail`, :meth:`rcpt` and :meth:`data` to send the message.)

      .. note::

         The *from_addr* and *to_addrs* parameters are used to construct the message
         envelope used by the transport agents.  ``sendmail`` does not modify the
         message headers in any way.

      *msg* may be a string containing characters in the ASCII range, or a byte
      string.  A string is encoded to bytes using the ascii codec, and lone ``\r``
      and ``\n`` characters are converted to ``\r\n`` characters.  A byte string is
      not modified.

      If there has been no previous ``EHLO`` or ``HELO`` command this session, this
      method tries ESMTP ``EHLO`` first. If the server does ESMTP, message size and
      each of the specified options will be passed to it (if the option is in the
      feature set the server advertises).  If ``EHLO`` fails, ``HELO`` will be tried
      and ESMTP options suppressed.

      This method will return normally if the mail is accepted for at least one
      recipient. Otherwise it will raise an exception.  That is, if this method does
      not raise an exception, then someone should get your mail. If this method does
      not raise an exception, it returns a dictionary, with one entry for each
      recipient that was refused.  Each entry contains a tuple of the SMTP error code
      and the accompanying error message sent by the server.

      If ``SMTPUTF8`` is included in *mail_options*, and the server supports it,
      *from_addr* and *to_addrs* may contain non-ASCII characters.

      This method may raise the following exceptions:

      :exc:`SMTPRecipientsRefused`
         All recipients were refused.  Nobody got the mail.  The :attr:`recipients`
         attribute of the exception object is a dictionary with information about the
         refused recipients (like the one returned when at least one recipient was
         accepted).

      :exc:`SMTPHeloError`
         The server didn't reply properly to the ``HELO`` greeting.

      :exc:`SMTPSenderRefused`
         The server didn't accept the *from_addr*.

      :exc:`SMTPDataError`
         The server replied with an unexpected error code (other than a refusal of a
         recipient).

      :exc:`SMTPNotSupportedError`
         ``SMTPUTF8`` was given in the *mail_options* but is not supported by the
```

```
                  server.

            Unless otherwise noted, the connection will be open even after an exception is
            raised.

            .. versionchanged:: 3.2
               *msg* may be a byte string.

            .. versionchanged:: 3.5
               ``SMTPUTF8`` support added, and :exc:`SMTPNotSupportedError` may be
               raised if ``SMTPUTF8`` is specified but the server does not support it.
```

```
      .. method:: SMTP.send_message(msg, from_addr=None, to_addrs=None, \
                                    mail_options=(), rcpt_options=())

          This is a convenience method for calling :meth:`sendmail` with the message
          represented by an :class:`email.message.Message` object.  The arguments have
          the same meaning as for :meth:`sendmail`, except that *msg* is a ``Message``
          object.

          If *from_addr* is ``None`` or *to_addrs* is ``None``, ``send_message`` fills
          those arguments with addresses extracted from the headers of *msg* as
          specified in :rfc:`5322`\: *from_addr* is set to the :mailheader:`Sender`
          field if it is present, and otherwise to the :mailheader:`From` field.
          *to_addrs* combines the values (if any) of the :mailheader:`To`,
          :mailheader:`Cc`, and :mailheader:`Bcc` fields from *msg*.  If exactly one
          set of :mailheader:`Resent-*` headers appear in the message, the regular
          headers are ignored and the :mailheader:`Resent-*` headers are used instead.
          If the message contains more than one set of :mailheader:`Resent-*` headers,
          a :exc:`ValueError` is raised, since there is no way to unambiguously detect
          the most recent set of :mailheader:`Resent-` headers.

          ``send_message`` serializes *msg* using
          :class:`~email.generator.BytesGenerator` with ``\r\n`` as the *linesep*, and
          calls :meth:`sendmail` to transmit the resulting message.  Regardless of the
          values of *from_addr* and *to_addrs*, ``send_message`` does not transmit any
          :mailheader:`Bcc` or :mailheader:`Resent-Bcc` headers that may appear
          in *msg*.  If any of the addresses in *from_addr* and *to_addrs* contain
          non-ASCII characters and the server does not advertise ``SMTPUTF8`` support,
          an :exc:`SMTPNotSupported` error is raised.  Otherwise the ``Message`` is
          serialized with a clone of its :mod:`~email.policy` with the
          :attr:`~email.policy.EmailPolicy.utf8` attribute set to ``True``, and
          ``SMTPUTF8`` and ``BODY=8BITMIME`` are added to *mail_options*.

          .. versionadded:: 3.2

          .. versionadded:: 3.5
             Support for internationalized addresses (``SMTPUTF8``).
```

```
      .. method:: SMTP.quit()

          Terminate the SMTP session and close the connection.  Return the result of
          the SMTP ``QUIT`` command.
```

Low-level methods corresponding to the standard SMTP/ESMTP commands HELP, RSET, NOOP, MAIL, RCPT, and DATA are also supported. Normally these do not need to be called directly, so they are not documented here. For details, consult the module code.

SMTP Example

This example prompts the user for addresses needed in the message envelope ('To' and 'From' addresses), and the message to be delivered. Note that the headers to be included with the message must be included in the message as entered; this example doesn't do

any processing of the RFC 822 headers. In particular, the 'To' and 'From' addresses must be included in the message headers explicitly.

```python
import smtplib

def prompt(prompt):
    return input(prompt).strip()

fromaddr = prompt("From: ")
toaddrs  = prompt("To: ").split()
print("Enter message, end with ^D (Unix) or ^Z (Windows):")

# Add the From: and To: headers at the start!
msg = ("From: %s\r\nTo: %s\r\n\r\n"
       % (fromaddr, ", ".join(toaddrs)))
while True:
    try:
        line = input()
    except EOFError:
        break
    if not line:
        break
    msg = msg + line

print("Message length is", len(msg))

server = smtplib.SMTP('localhost')
server.set_debuglevel(1)
server.sendmail(fromaddr, toaddrs, msg)
server.quit()
```

> **Note**
>
> In general, you will want to use the :mod:`email` package's features to construct an email message, which you can then send via :meth:`~smtplib.SMTP.send_message`; see :ref:`email-examples`.
>
> > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)smtplib.rst, line 603); *backlink***
> >
> > Unknown interpreted text role "mod".
>
> > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)smtplib.rst, line 603); *backlink***
> >
> > Unknown interpreted text role "meth".
>
> > **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)smtplib.rst, line 603); *backlink***
> >
> > Unknown interpreted text role "ref".