21.19. telnetlib — Telnet client

Source code: Lib/telnetlib.py

The telnetlib module provides a Telnet class that implements the Telnet protocol. See **RFC 854** for details about the protocol. In addition, it provides symbolic constants for the protocol characters (see below), and for the telnet options. The symbolic names of the telnet options follow the definitions in arpa/telnet.h, with the leading TELOPT_ removed. For symbolic names of options which are traditionally not included in arpa/telnet.h, see the module source itself.

The symbolic constants for the telnet commands are: IAC, DONT, DO, WONT, WILL, SE (Subnegotiation End), NOP (No Operation), DM (Data Mark), BRK (Break), IP (Interrupt process), AO (Abort output), AYT (Are You There), EC (Erase Character), EL (Erase Line), GA (Go Ahead), SB (Subnegotiation Begin).

class telnetlib.**Telnet**(*host=None*, *port=0*[, *timeout*])

Telnet represents a connection to a Telnet server. The instance is initially not connected by default; the open() method must be used to establish a connection. Alternatively, the host name and optional port number can be passed to the constructor too, in which case the connection to the server will be established before the constructor returns. 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).

Do not reopen an already connected instance.

This class has many read_*() methods. Note that some of them raise EOFError when the end of the connection is read, because they can return an empty string for other reasons. See the individual descriptions below.

A Telnet object is a context manager and can be used in a with statement. When the with block ends, the close() method is called:

```
>>> from telnetlib import Telnet
>>> with Telnet('localhost', 23) as tn:
...     tn.interact()
...
```

Changed in version 3.6: Context manager support added

See also:

RFC 854 - Telnet Protocol Specification
Definition of the Telnet protocol.

21.19.1. Telnet Objects

`Telnet` instances have the following methods:

`Telnet.`**`read_until`**(*expected*, *timeout=None*)

Read until a given byte string, *expected*, is encountered or until *timeout* seconds have passed.

When no match is found, return whatever is available instead, possibly empty bytes. Raise `EOFError` if the connection is closed and no cooked data is available.

`Telnet.`**`read_all`**()

Read all data until EOF as bytes; block until connection closed.

`Telnet.`**`read_some`**()

Read at least one byte of cooked data unless EOF is hit. Return `b''` if EOF is hit. Block if no data is immediately available.

`Telnet.`**`read_very_eager`**()

Read everything that can be without blocking in I/O (eager).

Raise `EOFError` if connection closed and no cooked data available. Return `b''` if no cooked data available otherwise. Do not block unless in the midst of an IAC sequence.

`Telnet.`**`read_eager`**()

Read readily available data.

Raise `EOFError` if connection closed and no cooked data available. Return `b''` if no cooked data available otherwise. Do not block unless in the midst of an IAC sequence.

`Telnet.`**`read_lazy`**()

Process and return data already in the queues (lazy).

Raise `EOFError` if connection closed and no data available. Return `b''` if no cooked data available otherwise. Do not block unless in the midst of an IAC sequence.

`Telnet.`**`read_very_lazy`**()

Return any data available in the cooked queue (very lazy).

Raise `EOFError` if connection closed and no data available. Return `b''` if no cooked data available otherwise. This method never blocks.

`Telnet.`**`read_sb_data`**()

Return the data collected between a SB/SE pair (suboption begin/end). The callback should access these data when it was invoked with a `SE` command. This method never blocks.

`Telnet.`**`open`**(*host*, *port=0*[, *timeout*])

Connect to a host. The optional second argument is the port number, which defaults to the standard Telnet port (23). 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).

Do not try to reopen an already connected instance.

`Telnet.`**`msg`**(*msg*, **args*)

Print a debug message when the debug level is > 0. If extra arguments are present, they are substituted in the message using the standard string formatting operator.

`Telnet.`**`set_debuglevel`**(*debuglevel*)

Set the debug level. The higher the value of *debuglevel*, the more debug output you get (on `sys.stdout`).

`Telnet.`**`close`**()

Close the connection.

`Telnet.`**`get_socket`**()

Return the socket object used internally.

`Telnet.`**`fileno`**()

Return the file descriptor of the socket object used internally.

`Telnet.`**`write`**(*buffer*)

Write a byte string to the socket, doubling any IAC characters. This can block if the connection is blocked. May raise `OSError` if the connection is closed.

Changed in version 3.3: This method used to raise `socket.error`, which is now an alias of `OSError`.

`Telnet.`**`interact`**()

Interaction function, emulates a very dumb Telnet client.

`Telnet.`**`mt_interact`**`()`

Multithreaded version of `interact()`.

`Telnet.`**`expect`**`(`*list*, *timeout=None*`)`

Read until one from a list of a regular expressions matches.

The first argument is a list of regular expressions, either compiled (regex objects) or uncompiled (byte strings). The optional second argument is a timeout, in seconds; the default is to block indefinitely.

Return a tuple of three items: the index in the list of the first regular expression that matches; the match object returned; and the bytes read up till and including the match.

If end of file is found and no bytes were read, raise `EOFError`. Otherwise, when nothing matches, return `(-1, None, data)` where *data* is the bytes received so far (may be empty bytes if a timeout happened).

If a regular expression ends with a greedy match (such as `.*`) or if more than one expression can match the same input, the results are non-deterministic, and may depend on the I/O timing.

`Telnet.`**`set_option_negotiation_callback`**`(`*callback*`)`

Each time a telnet option is read on the input flow, this *callback* (if set) is called with the following parameters: callback(telnet socket, command (DO/DONT/WILL/WONT), option). No other action is done afterwards by telnetlib.

21.19.2. Telnet Example

A simple example illustrating typical use:

```python
import getpass
import telnetlib

HOST = "localhost"
user = input("Enter your remote account: ")
password = getpass.getpass()

tn = telnetlib.Telnet(HOST)

tn.read_until(b"login: ")
tn.write(user.encode('ascii') + b"\n")
if password:
    tn.read_until(b"Password: ")
    tn.write(password.encode('ascii') + b"\n")
tn.write(b"ls\n")
tn.write(b"exit\n")

print(tn.read_all().decode('ascii'))
```