:mod:`cmd` --- Support for line-oriented command interpreters

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

Unknown directive type "module".

```
.. module:: cmd
   :synopsis: Build line-oriented command interpreters.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 7)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Eric S. Raymond <esr@snark.thyrsus.com>
```

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

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

Unknown interpreted text role "source".

The :class:`Cmd` class provides a simple framework for writing line-oriented command interpreters. These are often useful for test harnesses, administrative tools, and prototypes that will later be wrapped in a more sophisticated interface.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 13); *backlink*

Unknown interpreted text role "class".

A :class:`Cmd` instance or subclass instance is a line-oriented interpreter framework. There is no good reason to instantiate :class:`Cmd` itself; rather, it's useful as a superclass of an interpreter class you define yourself in order to inherit :class:`Cmd`'s methods and encapsulate action methods.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 20); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 20); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 20); *backlink*

Unknown interpreted text role "class".

The optional argument *completekey* is the :mod:`readline` name of a completion key; it defaults to :kbd:`Tab`. If *completekey* is not :const:`None` and :mod:`readline` is available, command completion is done automatically.

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

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

Unknown interpreted text role "mod".

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

Unknown interpreted text role "kbd".

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

Unknown interpreted text role "const".

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

Unknown interpreted text role "mod".

The optional arguments *stdin* and *stdout* specify the input and output file objects that the Cmd instance or subclass instance will use for input and output. If not specified, they will default to :data:`sys.stdin` and :data:`sys.stdout`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 29); *backlink*

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 29); *backlink*

Unknown interpreted text role "data".

If you want a given *stdin* to be used, make sure to set the instance's :attr:`use_rawinput` attribute to `False`, otherwise *stdin* will be ignored.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 34); *backlink*

Unknown interpreted text role "attr".

Cmd Objects

A :class:`Cmd` instance has the following methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 44); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 47)

Unknown directive type "method".

```
.. method:: Cmd.cmdloop(intro=None)

   Repeatedly issue a prompt, accept input, parse an initial prefix off the
   received input, and dispatch to action methods, passing them the remainder of
   the line as argument.

   The optional argument is a banner or intro string to be issued before the first
   prompt (this overrides the :attr:`intro` class attribute).

   If the :mod:`readline` module is loaded, input will automatically inherit
   :program:`bash`\ -like history-list editing (e.g. :kbd:`Control-P` scrolls back
   to the last command, :kbd:`Control-N` forward to the next one, :kbd:`Control-F`
   moves the cursor to the right non-destructively, :kbd:`Control-B` moves the
   cursor to the left non-destructively, etc.).
```

```
An end-of-file on input is passed back as the string ``'EOF'``.

.. index::
   single: ? (question mark); in a command interpreter
   single: ! (exclamation); in a command interpreter

An interpreter instance will recognize a command name ``foo`` if and only if it
has a method :meth:`do_foo`.  As a special case, a line beginning with the
character ``'?'`` is dispatched to the method :meth:`do_help`.  As another
special case, a line beginning with the character ``'!'`` is dispatched to the
method :meth:`do_shell` (if such a method is defined).

This method will return when the :meth:`postcmd` method returns a true value.
The *stop* argument to :meth:`postcmd` is the return value from the command's
corresponding :meth:`do_\*` method.

If completion is enabled, completing commands will be done automatically, and
completing of commands args is done by calling :meth:`complete_foo` with
arguments *text*, *line*, *begidx*, and *endidx*.  *text* is the string prefix
we are attempting to match: all returned matches must begin with it. *line* is
the current input line with leading whitespace removed, *begidx* and *endidx*
are the beginning and ending indexes of the prefix text, which could be used to
provide different completion depending upon which position the argument is in.

All subclasses of :class:`Cmd` inherit a predefined :meth:`do_help`.  This
method, called with an argument ``'bar'``, invokes the corresponding method
:meth:`help_bar`, and if that is not present, prints the docstring of
:meth:`do_bar`, if available.  With no argument, :meth:`do_help` lists all
available help topics (that is, all commands with corresponding
:meth:`help_\*` methods or commands that have docstrings), and also lists any
undocumented commands.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 95)

Unknown directive type "method".

```
.. method:: Cmd.onecmd(str)

   Interpret the argument as though it had been typed in response to the prompt.
   This may be overridden, but should not normally need to be; see the
   :meth:`precmd` and :meth:`postcmd` methods for useful execution hooks.  The
   return value is a flag indicating whether interpretation of commands by the
   interpreter should stop.  If there is a :meth:`do_\*` method for the command
   *str*, the return value of that method is returned, otherwise the return value
   from the :meth:`default` method is returned.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 106)

Unknown directive type "method".

```
.. method:: Cmd.emptyline()

   Method called when an empty line is entered in response to the prompt. If this
   method is not overridden, it repeats the last nonempty command entered.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 112)

Unknown directive type "method".

```
.. method:: Cmd.default(line)

   Method called on an input line when the command prefix is not recognized. If
   this method is not overridden, it prints an error message and returns.
```

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

main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 118)

Unknown directive type "method".

```
.. method:: Cmd.completedefault(text, line, begidx, endidx)

   Method called to complete an input line when no command-specific
   :meth:`complete_\*` method is available.  By default, it returns an empty list.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 124)

Unknown directive type "method".

```
.. method:: Cmd.columnize(list, displaywidth=80)

   Method called to display a list of strings as a compact set of columns.
   Each column is only as wide as necessary.
   Columns are separated by two spaces for readability.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 131)

Unknown directive type "method".

```
.. method:: Cmd.precmd(line)

   Hook method executed just before the command line *line* is interpreted, but
   after the input prompt is generated and issued.  This method is a stub in
   :class:`Cmd`; it exists to be overridden by subclasses.  The return value is
   used as the command which will be executed by the :meth:`onecmd` method; the
   :meth:`precmd` implementation may re-write the command or simply return *line*
   unchanged.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 141)

Unknown directive type "method".

```
.. method:: Cmd.postcmd(stop, line)

   Hook method executed just after a command dispatch is finished.  This method is
   a stub in :class:`Cmd`; it exists to be overridden by subclasses.  *line* is the
   command line which was executed, and *stop* is a flag which indicates whether
   execution will be terminated after the call to :meth:`postcmd`; this will be the
   return value of the :meth:`onecmd` method.  The return value of this method will
   be used as the new value for the internal flag which corresponds to *stop*;
   returning false will cause interpretation to continue.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 152)

Unknown directive type "method".

```
.. method:: Cmd.preloop()

   Hook method executed once when :meth:`cmdloop` is called.  This method is a stub
   in :class:`Cmd`; it exists to be overridden by subclasses.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 158)

Unknown directive type "method".

```
.. method:: Cmd.postloop()
```

```
Hook method executed once when :meth:`cmdloop` is about to return. This method
is a stub in :class:`Cmd`; it exists to be overridden by subclasses.
```

Instances of :class:`Cmd` subclasses have some public instance variables:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 164); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 166)

Unknown directive type "attribute".

```
.. attribute:: Cmd.prompt

   The prompt issued to solicit input.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 171)

Unknown directive type "attribute".

```
.. attribute:: Cmd.identchars

   The string of characters accepted for the command prefix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 176)

Unknown directive type "attribute".

```
.. attribute:: Cmd.lastcmd

   The last nonempty command prefix seen.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 181)

Unknown directive type "attribute".

```
.. attribute:: Cmd.cmdqueue

   A list of queued input lines.  The cmdqueue list is checked in
   :meth:`cmdloop` when new input is needed; if it is nonempty, its elements
   will be processed in order, as if entered at the prompt.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 188)

Unknown directive type "attribute".

```
.. attribute:: Cmd.intro

   A string to issue as an intro or banner.  May be overridden by giving the
   :meth:`cmdloop` method an argument.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 194)

Unknown directive type "attribute".

```
.. attribute:: Cmd.doc_header

   The header to issue if the help output has a section for documented commands.
```

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

Unknown directive type "attribute".

```
.. attribute:: Cmd.misc_header

   The header to issue if the help output has a section for miscellaneous  help
   topics (that is, there are :meth:`help_\*` methods without corresponding
   :meth:`do_\*` methods).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 206)

Unknown directive type "attribute".

```
.. attribute:: Cmd.undoc_header

   The header to issue if the help output has a section for undocumented  commands
   (that is, there are :meth:`do_\*` methods without corresponding :meth:`help_\*`
   methods).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 213)

Unknown directive type "attribute".

```
.. attribute:: Cmd.ruler

   The character used to draw separator lines under the help-message headers.  If
   empty, no ruler line is drawn.  It defaults to ``'='``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 219)

Unknown directive type "attribute".

```
.. attribute:: Cmd.use_rawinput

   A flag, defaulting to true.  If true, :meth:`cmdloop` uses :func:`input` to
   display a prompt and read the next command; if false, :meth:`sys.stdout.write`
   and :meth:`sys.stdin.readline` are used. (This means that by importing
   :mod:`readline`, on systems that support it, the interpreter will automatically
   support :program:`Emacs`\ -like line editing  and command-history keystrokes.)
```

Cmd Example

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 233)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Raymond Hettinger <python at rcn dot com>
```

The :mod:`cmd` module is mainly useful for building custom shells that let a user work with a program interactively.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 235); *backlink*

Unknown interpreted text role "mod".

This section presents a simple example of how to build a shell around a few of the commands in the :mod:`turtle` module.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 238); *backlink*

Unknown interpreted text role "mod".

Basic turtle commands such as :meth:`~turtle.forward` are added to a :class:`Cmd` subclass with method named :meth:`do_forward`. The argument is converted to a number and dispatched to the turtle module. The docstring is used in the help utility provided by the shell.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 241); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 241); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 241); *backlink*

Unknown interpreted text role "meth".

The example also includes a basic record and playback facility implemented with the :meth:`~Cmd.precmd` method which is responsible for converting the input to lowercase and writing the commands to a file. The :meth:`do_playback` method reads the file and adds the recorded commands to the :attr:`cmdqueue` for immediate playback:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 246); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 246); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 246); *backlink*

Unknown interpreted text role "attr".

```
import cmd, sys
from turtle import *

class TurtleShell(cmd.Cmd):
    intro = 'Welcome to the turtle shell.   Type help or ? to list commands.\n'
    prompt = '(turtle) '
    file = None

    # ----- basic turtle commands -----
    def do_forward(self, arg):
        'Move the turtle forward by the specified distance:  FORWARD 10'
        forward(*parse(arg))
    def do_right(self, arg):
        'Turn turtle right by given number of degrees:  RIGHT 20'
        right(*parse(arg))
    def do_left(self, arg):
        'Turn turtle left by given number of degrees:  LEFT 90'
        left(*parse(arg))
    def do_goto(self, arg):
        'Move turtle to an absolute position with changing orientation.  GOTO 100 200'
        goto(*parse(arg))
    def do_home(self, arg):
```

```
            'Return turtle to the home position:  HOME'
            home()
        def do_circle(self, arg):
            'Draw circle with given radius an options extent and steps:  CIRCLE 50'
            circle(*parse(arg))
        def do_position(self, arg):
            'Print the current turtle position:  POSITION'
            print('Current position is %d %d\n' % position())
        def do_heading(self, arg):
            'Print the current turtle heading in degrees:  HEADING'
            print('Current heading is %d\n' % (heading(),))
        def do_color(self, arg):
            'Set the color:  COLOR BLUE'
            color(arg.lower())
        def do_undo(self, arg):
            'Undo (repeatedly) the last turtle action(s):  UNDO'
        def do_reset(self, arg):
            'Clear the screen and return turtle to center:  RESET'
            reset()
        def do_bye(self, arg):
            'Stop recording, close the turtle window, and exit:  BYE'
            print('Thank you for using Turtle')
            self.close()
            bye()
            return True

        # ----- record and playback -----
        def do_record(self, arg):
            'Save future commands to filename:  RECORD rose.cmd'
            self.file = open(arg, 'w')
        def do_playback(self, arg):
            'Playback commands from a file:  PLAYBACK rose.cmd'
            self.close()
            with open(arg) as f:
                self.cmdqueue.extend(f.read().splitlines())
        def precmd(self, line):
            line = line.lower()
            if self.file and 'playback' not in line:
                print(line, file=self.file)
            return line
        def close(self):
            if self.file:
                self.file.close()
                self.file = None

    def parse(arg):
        'Convert a series of zero or more numbers to an argument tuple'
        return tuple(map(int, arg.split()))

    if __name__ == '__main__':
        TurtleShell().cmdloop()
```

Here is a sample session with the turtle shell showing the help functions, using blank lines to repeat commands, and the simple record and playback facility:

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]cmd.rst, line 330)

Cannot analyze code. No Pygments lexer found for "none".

```
    .. code-block:: none

        Welcome to the turtle shell.   Type help or ? to list commands.

        (turtle) ?

        Documented commands (type help <topic>):
        ========================================
        bye     color    goto     home  playback  record  right
        circle  forward  heading  left  position  reset   undo

        (turtle) help forward
        Move the turtle forward by the specified distance:  FORWARD 10
        (turtle) record spiral.cmd
        (turtle) position
        Current position is 0 0

        (turtle) heading
        Current heading is 0
```

```
(turtle) reset
(turtle) circle 20
(turtle) right 30
(turtle) circle 40
(turtle) right 30
(turtle) circle 60
(turtle) right 30
(turtle) circle 80
(turtle) right 30
(turtle) circle 100
(turtle) right 30
(turtle) circle 120
(turtle) right 30
(turtle) circle 120
(turtle) heading
Current heading is 180

(turtle) forward 100
(turtle)
(turtle) right 90
(turtle) forward 100
(turtle)
(turtle) right 90
(turtle) forward 400
(turtle) right 90
(turtle) forward 500
(turtle) right 90
(turtle) forward 400
(turtle) right 90
(turtle) forward 300
(turtle) playback spiral.cmd
Current position is 0 0

Current heading is 0

Current heading is 180

(turtle) bye
Thank you for using Turtle
```