# :mod:`fileinput` --- Iterate over lines from multiple input streams

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

Unknown directive type "module".

```
.. module:: fileinput
   :synopsis: Loop over standard input or a list of files.
```

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

Unknown directive type "moduleauthor".

```
.. moduleauthor:: Guido van Rossum <guido@python.org>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 8)

Unknown directive type "sectionauthor".

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

**Source code:** :source:`Lib/fileinput.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 10); *backlink*

Unknown interpreted text role "source".

---

This module implements a helper class and functions to quickly write a loop over standard input or a list of files. If you just want to read or write one file see :func:`open`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 14); *backlink*

Unknown interpreted text role "func".

The typical use is:

```
import fileinput
for line in fileinput.input(encoding="utf-8"):
    process(line)
```

This iterates over the lines of all files listed in `sys.argv[1:]`, defaulting to `sys.stdin` if the list is empty. If a filename is `'-'`, it is also replaced by `sys.stdin` and the optional arguments *mode* and *openhook* are ignored. To specify an alternative list of filenames, pass it as the first argument to :func:`.input`. A single file name is also allowed.

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

Unknown interpreted text role "func".

All files are opened in text mode by default, but you can override this by specifying the *mode* parameter in the call to :func:`.input` or :class:`FileInput`. If an I/O error occurs during opening or reading a file, :exc:`OSError` is raised.

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

Unknown interpreted text role "func".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 35)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   :exc:`IOError` used to be raised; it is now an alias of :exc:`OSError`.
```

If `sys.stdin` is used more than once, the second and further use will return no lines, except perhaps for interactive use, or if it has been explicitly reset (e.g. using `sys.stdin.seek(0)`).

Empty files are opened and immediately closed; the only time their presence in the list of filenames is noticeable at all is when the last file opened is empty.

Lines are returned with any newlines intact, which means that the last line in a file may not have one.

You can control how files are opened by providing an opening hook via the *openhook* parameter to :func:`fileinput.input` or :class:`FileInput()`. The hook must be a function that takes two arguments, *filename* and *mode*, and returns an accordingly opened file-like object. If *encoding* and/or *errors* are specified, they will be passed to the hook as additional keyword arguments. This module provides a :func:`hook_compressed` to support compressed files.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 49); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 49); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 49); *backlink*

Unknown interpreted text role "func".

The following function is the primary interface of this module:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 59)

Unknown directive type "function".

```
.. function:: input(files=None, inplace=False, backup='', *, mode='r', openhook=None, encoding=None, erro

   Create an instance of the :class:`FileInput` class.  The instance will be used
   as global state for the functions of this module, and is also returned to use
   during iteration.  The parameters to this function will be passed along to the
   constructor of the :class:`FileInput` class.

   The :class:`FileInput` instance can be used as a context manager in the
   :keyword:`with` statement.  In this example, *input* is closed after the
   :keyword:`!with` statement is exited, even if an exception occurs::

      with fileinput.input(files=('spam.txt', 'eggs.txt'), encoding="utf-8") as f:
          for line in f:
              process(line)

   .. versionchanged:: 3.2
      Can be used as a context manager.

   .. versionchanged:: 3.8
      The keyword parameters *mode* and *openhook* are now keyword-only.

   .. versionchanged:: 3.10
      The keyword-only parameter *encoding* and *errors* are added.
```

The following functions use the global state created by :func:`fileinput.input`; if there is no active state, :exc:`RuntimeError` is raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 84); *backlink*

Unknown interpreted text role "func".

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

**main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 84); *backlink***

Unknown interpreted text role "exc".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 88)**

Unknown directive type "function".

```
.. function:: filename()

   Return the name of the file currently being read.  Before the first line has
   been read, returns ``None``.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 94)**

Unknown directive type "function".

```
.. function:: fileno()

   Return the integer "file descriptor" for the current file. When no file is
   opened (before the first line and between files), returns ``-1``.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 100)**

Unknown directive type "function".

```
.. function:: lineno()

   Return the cumulative line number of the line that has just been read.  Before
   the first line has been read, returns ``0``.  After the last line of the last
   file has been read, returns the line number of that line.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 107)**

Unknown directive type "function".

```
.. function:: filelineno()

   Return the line number in the current file.  Before the first line has been
   read, returns ``0``.  After the last line of the last file has been read,
   returns the line number of that line within the file.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 114)**

Unknown directive type "function".

```
.. function:: isfirstline()

   Return ``True`` if the line just read is the first line of its file, otherwise
   return ``False``.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 120)**

Unknown directive type "function".

```
.. function:: isstdin()

   Return ``True`` if the last line was read from ``sys.stdin``, otherwise return
   ``False``.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 126)**

Unknown directive type "function".

```
.. function:: nextfile()

   Close the current file so that the next iteration will read the first line from
   the next file (if any); lines not read from the file will not count towards the
```

```
cumulative line count.  The filename is not changed until after the first line
of the next file has been read.  Before the first line has been read, this
function has no effect; it cannot be used to skip the first file.  After the
last line of the last file has been read, this function has no effect.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)fileinput.rst, line 136)**

Unknown directive type "function".

```
.. function:: close()

   Close the sequence.
```

The class which implements the sequence behavior provided by the module is available for subclassing as well:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)fileinput.rst, line 144)**

Invalid class attribute value for "class" directive: "FileInput(files=None, inplace=False, backup='', *, mode='r', openhook=None, encoding=None, errors=None)".

```
.. class:: FileInput(files=None, inplace=False, backup='', *, mode='r', openhook=None, encoding=None, erro

   Class :class:`FileInput` is the implementation; its methods :meth:`filename`,
   :meth:`fileno`, :meth:`lineno`, :meth:`filelineno`, :meth:`isfirstline`,
   :meth:`isstdin`, :meth:`nextfile` and :meth:`close` correspond to the
   functions of the same name in the module. In addition it is :term:`iterable`
   and has a :meth:`~io.TextIOBase.readline` method which returns the next
   input line. The sequence must be accessed in strictly sequential order;
   random access and :meth:`~io.TextIOBase.readline` cannot be mixed.

   With *mode* you can specify which file mode will be passed to :func:`open`. It
   must be one of ``'r'`` and ``'rb'``.

   The *openhook*, when given, must be a function that takes two arguments,
   *filename* and *mode*, and returns an accordingly opened file-like object. You
   cannot use *inplace* and *openhook* together.

   You can specify *encoding* and *errors* that is passed to :func:`open` or *openhook*.

   A :class:`FileInput` instance can be used as a context manager in the
   :keyword:`with` statement.  In this example, *input* is closed after the
   :keyword:`!with` statement is exited, even if an exception occurs::

      with FileInput(files=('spam.txt', 'eggs.txt')) as input:
          process(input)

   .. versionchanged:: 3.2
      Can be used as a context manager.

   .. versionchanged:: 3.8
      The keyword parameter *mode* and *openhook* are now keyword-only.

   .. versionchanged:: 3.10
      The keyword-only parameter *encoding* and *errors* are added.

   .. versionchanged:: 3.11
      The ``'rU'`` and ``'U'`` modes and the :meth:`__getitem__` method have
      been removed.
```

**Optional in-place filtering:** if the keyword argument `inplace=True` is passed to :func:`fileinput.input` or to the :class:`FileInput` constructor, the file is moved to a backup file and standard output is directed to the input file (if a file of the same name as the backup file already exists, it will be replaced silently). This makes it possible to write a filter that rewrites its input file in place. If the *backup* parameter is given (typically as `backup='.<some extension>'`), it specifies the extension for the backup file, and the backup file remains around; by default, the extension is `'.bak'` and it is deleted when the output file is closed. In-place filtering is disabled when standard input is read.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)fileinput.rst, line 184); *backlink***

Unknown interpreted text role "func".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)fileinput.rst, line 184); *backlink***

Unknown interpreted text role "class".

The two following opening hooks are provided by this module:

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

**main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 198)**

Unknown directive type "function".

```
.. function:: hook_compressed(filename, mode, *, encoding=None, errors=None)

   Transparently opens files compressed with gzip and bzip2 (recognized by the
   extensions ``'.gz'`` and ``'.bz2'``) using the :mod:`gzip` and :mod:`bz2`
   modules.  If the filename extension is not ``'.gz'`` or ``'.bz2'``, the file is
   opened normally (ie, using :func:`open` without any decompression).

   The *encoding* and *errors* values are passed to :class:`io.TextIOWrapper`
   for compressed files and open for normal files.

   Usage example:  ``fi = fileinput.FileInput(openhook=fileinput.hook_compressed, encoding="utf-8")``

   .. versionchanged:: 3.10
      The keyword-only parameter *encoding* and *errors* are added.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)fileinput.rst, line 214)**

Unknown directive type "function".

```
.. function:: hook_encoded(encoding, errors=None)

   Returns a hook which opens each file with :func:`open`, using the given
   *encoding* and *errors* to read the file.

   Usage example: ``fi =
   fileinput.FileInput(openhook=fileinput.hook_encoded("utf-8",
   "surrogateescape"))``

   .. versionchanged:: 3.6
      Added the optional *errors* parameter.

   .. deprecated:: 3.10
      This function is deprecated since :func:`input` and :class:`FileInput`
      now have *encoding* and *errors* parameters.
```