# :mod:`tempfile` --- Generate temporary files and directories

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]tempfile.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]tempfile.rst, line 4`)
>
> Unknown directive type "module".
>
> ```
> .. module:: tempfile
>    :synopsis: Generate temporary files and directories.
> ```

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]tempfile.rst, line 7`)
>
> Unknown directive type "sectionauthor".
>
> ```
> .. sectionauthor:: Zack Weinberg <zack@codesourcery.com>
> ```

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

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]tempfile.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]tempfile.rst, line 11`)
>
> Unknown directive type "index".
>
> ```
> .. index::
>    pair: temporary; file name
>    pair: temporary; file
> ```

---

This module creates temporary files and directories. It works on all supported platforms. :class:`TemporaryFile`, :class:`NamedTemporaryFile`, :class:`TemporaryDirectory`, and :class:`SpooledTemporaryFile` are high-level interfaces which provide automatic cleanup and can be used as context managers. :func:`mkstemp` and :func:`mkdtemp` are lower-level functions which require manual cleanup.

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

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

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

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

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

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

All the user-callable functions and constructors take additional arguments which allow direct control over the location and name of temporary files and directories. Files names used by this module include a string of random characters which allows those files to be securely created in shared temporary directories. To maintain backward compatibility, the argument order is somewhat odd; it is recommended to use keyword arguments for clarity.

The module defines the following user-callable items:

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]tempfile.rst, line 34`)
>
> Unknown directive type "function".
>
> ```
> .. function:: TemporaryFile(mode='w+b', buffering=-1, encoding=None, newline=None, suffix=None, prefix=None, dir=None, *, er
>
>    Return a :term:`file-like object` that can be used as a temporary storage area.
>    The file is created securely, using the same rules as :func:`mkstemp`. It will be destroyed as soon
>    as it is closed (including an implicit close when the object is garbage
>    collected).  Under Unix, the directory entry for the file is either not created at all or is removed
>    immediately after the file is created.  Other platforms do not support
>    this; your code should not rely on a temporary file created using this
>    function having or not having a visible name in the file system.
> ```

```
The resulting object can be used as a context manager (see
:ref:`tempfile-examples`).  On completion of the context or
destruction of the file object the temporary file will be removed
from the filesystem.

The *mode* parameter defaults to ``'w+b'`` so that the file created can
be read and written without being closed.  Binary mode is used so that it
behaves consistently on all platforms without regard for the data that is
stored.  *buffering*, *encoding*, *errors* and *newline* are interpreted as for
:func:`open`.

The *dir*, *prefix* and *suffix* parameters have the same meaning and
defaults as with :func:`mkstemp`.

The returned object is a true file object on POSIX platforms.  On other
platforms, it is a file-like object whose :attr:`!file` attribute is the
underlying true file object.

The :py:data:`os.O_TMPFILE` flag is used if it is available and works
(Linux-specific, requires Linux kernel 3.11 or later).

On platforms that are neither Posix nor Cygwin, TemporaryFile is an alias
for NamedTemporaryFile.

.. audit-event:: tempfile.mkstemp fullpath tempfile.TemporaryFile

.. versionchanged:: 3.5

   The :py:data:`os.O_TMPFILE` flag is now used if available.

.. versionchanged:: 3.8
   Added *errors* parameter.
```

---

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]tempfile.rst, **line 78**)

Unknown directive type "function".

```
.. function:: NamedTemporaryFile(mode='w+b', buffering=-1, encoding=None, newline=None, suffix=None, prefix=None, dir=None,

   This function operates exactly as :func:`TemporaryFile` does, except that
   the file is guaranteed to have a visible name in the file system (on
   Unix, the directory entry is not unlinked).  That name can be retrieved
   from the :attr:`name` attribute of the returned
   file-like object.  Whether the name can be
   used to open the file a second time, while the named temporary file is
   still open, varies across platforms (it can be so used on Unix; it cannot
   on Windows NT or later).  If *delete* is true (the default), the file is
   deleted as soon as it is closed.
   The returned object is always a file-like object whose :attr:`!file`
   attribute is the underlying true file object. This file-like object can
   be used in a :keyword:`with` statement, just like a normal file.

   On POSIX (only), a process that is terminated abruptly with SIGKILL
   cannot automatically delete any NamedTemporaryFiles it created.

   .. audit-event:: tempfile.mkstemp fullpath tempfile.NamedTemporaryFile

   .. versionchanged:: 3.8
      Added *errors* parameter.
```

---

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]tempfile.rst, **line 102**)

Invalid class attribute value for "class" directive: "SpooledTemporaryFile(max_size=0, mode='w+b', buffering=-1, encoding=None, newline=None, suffix=None, prefix=None, dir=None, *, errors=None)".

```
.. class:: SpooledTemporaryFile(max_size=0, mode='w+b', buffering=-1, encoding=None, newline=None, suffix=None, prefix=None,

   This class operates exactly as :func:`TemporaryFile` does, except that
   data is spooled in memory until the file size exceeds *max_size*, or
   until the file's :func:`fileno` method is called, at which point the
   contents are written to disk and operation proceeds as with
   :func:`TemporaryFile`.

   The resulting file has one additional method, :func:`rollover`, which
   causes the file to roll over to an on-disk file regardless of its size.

   The returned object is a file-like object whose :attr:`_file` attribute
   is either an :class:`io.BytesIO` or :class:`io.TextIOWrapper` object
   (depending on whether binary or text *mode* was specified) or a true file
   object, depending on whether :func:`rollover` has been called.  This
   file-like object can be used in a :keyword:`with` statement, just like
   a normal file.

   .. versionchanged:: 3.3
      the truncate method now accepts a ``size`` argument.

   .. versionchanged:: 3.8
      Added *errors* parameter.
```

This class securely creates a temporary directory using the same rules as :func:`mkdtemp`. The resulting object can be used as a context manager (see :ref:`tempfile-examples`). On completion of the context or destruction of the temporary directory object, the newly created temporary directory and all its contents are removed from the filesystem.

---

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]tempfile.rst, **line 129**); *backlink*

Unknown interpreted text role "func".

---

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]tempfile.rst, **line 129**); *backlink*

Unknown interpreted text role "ref".

---

The directory name can be retrieved from the :attr:`name` attribute of the returned object. When the returned object is used as a context manager, the :attr:`name` will be assigned to the target of the :keyword:`!as` clause in the :keyword:`with` statement, if there is one.

The directory can be explicitly cleaned up by calling the :func:`cleanup` method. If *ignore_cleanup_errors* is true, any unhandled exceptions during explicit or implicit cleanup (such as a :exc:`PermissionError` removing open files on Windows) will be ignored, and the remaining removable items deleted on a "best-effort" basis. Otherwise, errors will be raised in whatever context cleanup occurs (the :func:`cleanup` call, exiting the context manager, when the object is garbage-collected or during interpreter shutdown).

```
.. audit-event:: tempfile.mkdtemp fullpath tempfile.TemporaryDirectory
```

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

```
.. versionchanged:: 3.10
   Added *ignore_cleanup_errors* parameter.
```

```
.. function:: mkstemp(suffix=None, prefix=None, dir=None, text=False)

   Creates a temporary file in the most secure manner possible.  There are
   no race conditions in the file's creation, assuming that the platform
   properly implements the :const:`os.O_EXCL` flag for :func:`os.open`.  The
   file is readable and writable only by the creating user ID.  If the
   platform uses permission bits to indicate whether a file is executable,
   the file is executable by no one.  The file descriptor is not inherited
   by child processes.

   Unlike :func:`TemporaryFile`, the user of :func:`mkstemp` is responsible
   for deleting the temporary file when done with it.

   If *suffix* is not ``None``, the file name will end with that suffix,
   otherwise there will be no suffix.  :func:`mkstemp` does not put a dot
   between the file name and the suffix; if you need one, put it at the
   beginning of *suffix*.

   If *prefix* is not ``None``, the file name will begin with that prefix;
   otherwise, a default prefix is used.  The default is the return value of
   :func:`gettempprefix` or :func:`gettempprefixb`, as appropriate.

   If *dir* is not ``None``, the file will be created in that directory;
   otherwise, a default directory is used.  The default directory is chosen
   from a platform-dependent list, but the user of the application can
   control the directory location by setting the *TMPDIR*, *TEMP* or *TMP*
   environment variables.  There is thus no guarantee that the generated
   filename will have any nice properties, such as not requiring quoting
   when passed to external commands via ``os.popen()``.

   If any of *suffix*, *prefix*, and *dir* are not
   ``None``, they must be the same type.
   If they are bytes, the returned name will be bytes instead of str.
   If you want to force a bytes return value with otherwise default behavior,
   pass ``suffix=b''``.

   If *text* is specified and true, the file is opened in text mode.
   Otherwise, (the default) the file is opened in binary mode.
```

```
:func:`mkstemp` returns a tuple containing an OS-level handle to an open
file (as would be returned by :func:`os.open`) and the absolute pathname
of that file, in that order.

.. audit-event:: tempfile.mkstemp fullpath tempfile.mkstemp

.. versionchanged:: 3.5
   *suffix*, *prefix*, and *dir* may now be supplied in bytes in order to
   obtain a bytes return value.  Prior to this, only str was allowed.
   *suffix* and *prefix* now accept and default to ``None`` to cause
   an appropriate default value to be used.

.. versionchanged:: 3.6
   The *dir* parameter now accepts a :term:`path-like object`.
```

```
.. function:: mkdtemp(suffix=None, prefix=None, dir=None)

   Creates a temporary directory in the most secure manner possible. There
   are no race conditions in the directory's creation.  The directory is
   readable, writable, and searchable only by the creating user ID.

   The user of :func:`mkdtemp` is responsible for deleting the temporary
   directory and its contents when done with it.

   The *prefix*, *suffix*, and *dir* arguments are the same as for
   :func:`mkstemp`.

   :func:`mkdtemp` returns the absolute pathname of the new directory.

   .. audit-event:: tempfile.mkdtemp fullpath tempfile.mkdtemp

   .. versionchanged:: 3.5
      *suffix*, *prefix*, and *dir* may now be supplied in bytes in order to
      obtain a bytes return value.  Prior to this, only str was allowed.
      *suffix* and *prefix* now accept and default to ``None`` to cause
      an appropriate default value to be used.

   .. versionchanged:: 3.6
      The *dir* parameter now accepts a :term:`path-like object`.
```

```
.. function:: gettempdir()

   Return the name of the directory used for temporary files. This
   defines the default value for the *dir* argument to all functions
   in this module.

   Python searches a standard list of directories to find one which
   the calling user can create files in.  The list is:

   #. The directory named by the :envvar:`TMPDIR` environment variable.

   #. The directory named by the :envvar:`TEMP` environment variable.

   #. The directory named by the :envvar:`TMP` environment variable.

   #. A platform-specific location:

      * On Windows, the directories :file:`C:\\TEMP`, :file:`C:\\TMP`,
        :file:`\\TEMP`, and :file:`\\TMP`, in that order.

      * On all other platforms, the directories :file:`/tmp`, :file:`/var/tmp`, and
        :file:`/usr/tmp`, in that order.

   #. As a last resort, the current working directory.

   The result of this search is cached, see the description of
   :data:`tempdir` below.

   .. versionchanged:: 3.10

      Always returns a str.  Previously it would return any :data:`tempdir`
      value regardless of type so long as it was not ``None``.
```

```
.. function:: gettempdirb()

   Same as :func:`gettempdir` but the return value is in bytes.

   .. versionadded:: 3.5
```

```
.. function:: gettempprefix()

   Return the filename prefix used to create temporary files.  This does not
   contain the directory component.
```

```
   .. function:: gettempprefixb()

      Same as :func:`gettempprefix` but the return value is in bytes.

      .. versionadded:: 3.5
```

The module uses a global variable to store the name of the directory used for temporary files returned by :func:`gettempdir`. It can be set directly to override the selection process, but this is discouraged. All functions in this module take a *dir* argument which can be used to specify the directory. This is the recommended approach that does not surprise other unsuspecting code by changing global API behavior.

```
   .. data:: tempdir

      When set to a value other than ``None``, this variable defines the
      default value for the *dir* argument to the functions defined in this
      module, including its type, bytes or str.  It cannot be a
      :term:`path-like object`.

      If ``tempdir`` is ``None`` (the default) at any call to any of the above
      functions except :func:`gettempprefix` it is initialized following the
      algorithm described in :func:`gettempdir`.

      .. note::

         Beware that if you set ``tempdir`` to a bytes value, there is a
         nasty side effect: The global default return type of
         :func:`mkstemp` and :func:`mkdtemp` changes to bytes when no
         explicit ``prefix``, ``suffix``, or ``dir`` arguments of type
         str are supplied. Please do not write code expecting or
         depending on this. This awkward behavior is maintained for
         compatibility with the historical implementation.
```

## Examples

Here are some examples of typical usage of the :mod:`tempfile` module:

```
>>> import tempfile

# create a temporary file and write some data to it
>>> fp = tempfile.TemporaryFile()
>>> fp.write(b'Hello world!')
# read data from file
>>> fp.seek(0)
>>> fp.read()
b'Hello world!'
# close the file, it will be removed
>>> fp.close()

# create a temporary file using a context manager
>>> with tempfile.TemporaryFile() as fp:
...     fp.write(b'Hello world!')
...     fp.seek(0)
...     fp.read()
b'Hello world!'
>>>
# file is now closed and removed

# create a temporary directory using the context manager
>>> with tempfile.TemporaryDirectory() as tmpdirname:
...     print('created temporary directory', tmpdirname)
>>>
# directory and contents have been removed
```

## Deprecated functions and variables

A historical way to create temporary files was to first generate a file name with the :func:`mktemp` function and then create a file using this name. Unfortunately this is not secure, because a different process may create a file with this name in the time between the call to :func:`mktemp` and the subsequent attempt to create the file by the first process. The solution is to combine the two steps and create the file immediately. This approach is used by :func:`mkstemp` and the other functions described above.

```
   .. function:: mktemp(suffix='', prefix='tmp', dir=None)
```

```
.. deprecated:: 2.3
   Use :func:`mkstemp` instead.

Return an absolute pathname of a file that did not exist at the time the
call is made.  The *prefix*, *suffix*, and *dir* arguments are similar
to those of :func:`mkstemp`, except that bytes file names, ``suffix=None``
and ``prefix=None`` are not supported.

.. warning::

   Use of this function may introduce a security hole in your program.  By
   the time you get around to doing anything with the file name it returns,
   someone else may have beaten you to the punch.  :func:`mktemp` usage can
   be replaced easily with :func:`NamedTemporaryFile`, passing it the
   ``delete=False`` parameter::

      >>> f = NamedTemporaryFile(delete=False)
      >>> f.name
      '/tmp/tmptjujjt'
      >>> f.write(b"Hello World!\n")
      13
      >>> f.close()
      >>> os.unlink(f.name)
      >>> os.path.exists(f.name)
      False
```