:mod:`warnings` --- Warning control

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

Unknown directive type "module".

```
.. module:: warnings
   :synopsis: Issue warning messages and control their disposition.
```

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

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

Unknown interpreted text role "source".

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

Unknown directive type "index".

```
.. index:: single: warnings
```

Warning messages are typically issued in situations where it is useful to alert the user of some condition in a program, where that condition (normally) doesn't warrant raising an exception and terminating the program. For example, one might want to issue a warning when a program uses an obsolete module.

Python programmers issue warnings by calling the :func:`warn` function defined in this module. (C programmers use :c:func:`PyErr_WarnEx`; see :ref:`exceptionhandling` for details).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 18); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 18); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 18); *backlink*

Unknown interpreted text role "ref".

Warning messages are normally written to :data:`sys.stderr`, but their disposition can be changed flexibly, from ignoring all warnings to turning them into exceptions. The disposition of warnings can vary based on the :ref:`warning category <warning-categories>`, the text of the warning message, and the source location where it is issued. Repetitions of a particular warning for the same source location are typically suppressed.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 22); *backlink*

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 22); *backlink*

Unknown interpreted text role "ref".

There are two stages in warning control: first, each time a warning is issued, a determination is made whether a message should be issued or not; next, if a message is to be issued, it is formatted and printed using a user-settable hook.

The determination whether to issue a warning message is controlled by the :ref:`warning filter <warning-filter>`, which is a sequence of matching rules and actions. Rules can be added to the filter by calling :func:`filterwarnings` and reset to its default state by calling :func:`resetwarnings`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 33); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 33); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 33); *backlink*

Unknown interpreted text role "func".

The printing of warning messages is done by calling :func:`showwarning`, which may be overridden; the default implementation of this function formats the message by calling :func:`formatwarning`, which is also available for use by custom implementations.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 38); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 38); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 43)

Unknown directive type "seealso".

```
.. seealso::
   :func:`logging.captureWarnings` allows you to handle all warnings with
   the standard logging infrastructure.
```

Warning Categories

There are a number of built-in exceptions that represent warning categories. This categorization is useful to be able to filter out groups of warnings.

While these are technically :ref:`built-in exceptions <warning-categories-as-exceptions>`, they are documented here, because conceptually they belong to the warnings mechanism.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 56); *backlink*

Unknown interpreted text role "ref".

User code can define additional warning categories by subclassing one of the standard warning categories. A warning category must always be a subclass of the :exc:`Warning` class.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 60); *backlink*

Unknown interpreted text role "exc".

The following warnings category classes are currently defined:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 66)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |l|p{0.6\linewidth}|
```

Class	Description
:exc:`Warning` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 72); *backlink* Unknown interpreted text role "exc".	This is the base class of all warning category classes. It is a subclass of :exc:`Exception`. System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 72); *backlink* Unknown interpreted text role "exc".

Class	Description
:exc:`UserWarning` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 76); *backlink* Unknown interpreted text role "exc".	The default category for :func:`warn`. System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 76); *backlink* Unknown interpreted text role "func".
:exc:`DeprecationWarning` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 78); *backlink* Unknown interpreted text role "exc".	Base category for warnings about deprecated features when those warnings are intended for other Python developers (ignored by default, unless triggered by code in `__main__`).
:exc:`SyntaxWarning` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 83); *backlink* Unknown interpreted text role "exc".	Base category for warnings about dubious syntactic features.
:exc:`RuntimeWarning` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 86); *backlink* Unknown interpreted text role "exc".	Base category for warnings about dubious runtime features.
:exc:`FutureWarning` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 89); *backlink* Unknown interpreted text role "exc".	Base category for warnings about deprecated features when those warnings are intended for end users of applications that are written in Python.
:exc:`PendingDeprecationWarning` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 94); *backlink* Unknown interpreted text role "exc".	Base category for warnings about features that will be deprecated in the future (ignored by default).

Class	Description
:exc:`ImportWarning` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 98); *backlink* Unknown interpreted text role "exc".	Base category for warnings triggered during the process of importing a module (ignored by default).
:exc:`UnicodeWarning` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 102); *backlink* Unknown interpreted text role "exc".	Base category for warnings related to Unicode.
:exc:`BytesWarning` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 105); *backlink* Unknown interpreted text role "exc".	Base category for warnings related to :class:`bytes` and :class:`bytearray`. System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 105); *backlink* Unknown interpreted text role "class". System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 105); *backlink* Unknown interpreted text role "class".
:exc:`ResourceWarning` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 108); *backlink* Unknown interpreted text role "exc".	Base category for warnings related to resource usage (ignored by default).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 111)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   Previously :exc:`DeprecationWarning` and :exc:`FutureWarning` were
   distinguished based on whether a feature was being removed entirely or
   changing its behaviour. They are now distinguished based on their
   intended audience and the way they're handled by the default warnings
   filters.
```

The Warnings Filter

The warnings filter controls whether warnings are ignored, displayed, or turned into errors (raising an exception).

Conceptually, the warnings filter maintains an ordered list of filter specifications; any specific warning is matched against each filter specification in the list in turn until a match is found; the filter determines the disposition of the match. Each entry is a tuple of the form (*action*, *message*, *category*, *module*, *lineno*), where:

- *action* is one of the following strings:

Value	Disposition

Value	Disposition
`"default"`	print the first occurrence of matching warnings for each location (module + line number) where the warning is issued
`"error"`	turn matching warnings into exceptions
`"ignore"`	never print matching warnings
`"always"`	always print matching warnings
`"module"`	print the first occurrence of matching warnings for each module where the warning is issued (regardless of line number)
`"once"`	print only the first occurrence of matching warnings, regardless of location

- *message* is a string containing a regular expression that the start of the warning message must match. The expression is compiled to always be case-insensitive.
- *category* is a class (a subclass of :exc:`Warning`) of which the warning category must be a subclass in order to match.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 160); *backlink*

  Unknown interpreted text role "exc".

- *module* is a string containing a regular expression that the module name must match. The expression is compiled to be case-sensitive.
- *lineno* is an integer that the line number where the warning occurred must match, or `0` to match all line numbers.

Since the :exc:`Warning` class is derived from the built-in :exc:`Exception` class, to turn a warning into an error we simply raise `category(message)`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 169); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 169); *backlink*

Unknown interpreted text role "exc".

If a warning is reported and doesn't match any registered filter then the "default" action is applied (hence its name).

Describing Warning Filters

The warnings filter is initialized by :option:`-W` options passed to the Python interpreter command line and the :envvar:`PYTHONWARNINGS` environment variable. The interpreter saves the arguments for all supplied entries without interpretation in :data:`sys.warnoptions`; the :mod:`warnings` module parses these when it is first imported (invalid options are ignored, after printing a message to :data:`sys.stderr`).

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

Unknown interpreted text role "option".

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

Unknown interpreted text role "envvar".

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

Unknown interpreted text role "data".

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

Unknown interpreted text role "mod".

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

Unknown interpreted text role "data".

Individual warnings filters are specified as a sequence of fields separated by colons:

```
action:message:category:module:line
```

The meaning of each of these fields is as described in :ref:`warning-filter`. When listing multiple filters on a single line (as for :envvar:`PYTHONWARNINGS`), the individual filters are separated by commas and the filters listed later take precedence over those listed before them (as they're applied left-to-right, and the most recently applied filters take precedence over earlier ones).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 193); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 193); *backlink*

Unknown interpreted text role "envvar".

Commonly used warning filters apply to either all warnings, warnings in a particular category, or warnings raised by particular modules or packages. Some examples:

```
default                      # Show all warnings (even those ignored by default)
ignore                       # Ignore all warnings
error                        # Convert all warnings to errors
error::ResourceWarning       # Treat ResourceWarning messages as errors
default::DeprecationWarning  # Show DeprecationWarning messages
ignore,default:::mymodule    # Only report warnings triggered by "mymodule"
error:::mymodule[.*]         # Convert warnings to errors in "mymodule"
                             # and any subpackages of "mymodule"
```

Default Warning Filter

By default, Python installs several warning filters, which can be overridden by the :option:`-W` command-line option, the :envvar:`PYTHONWARNINGS` environment variable and calls to :func:`filterwarnings`.

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

Unknown interpreted text role "option".

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

Unknown interpreted text role "envvar".

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

Unknown interpreted text role "func".

In regular release builds, the default warning filter has the following entries (in order of precedence):

```
default::DeprecationWarning:__main__
ignore::DeprecationWarning
ignore::PendingDeprecationWarning
ignore::ImportWarning
ignore::ResourceWarning
```

In a :ref:`debug build <debug-build>`, the list of default warning filters is empty.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 232); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 234)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.2
   :exc:`DeprecationWarning` is now ignored by default in addition to
   :exc:`PendingDeprecationWarning`.
```

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

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
  :exc:`DeprecationWarning` is once again shown by default when triggered
  directly by code in ``__main__``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 242)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
  :exc:`BytesWarning` no longer appears in the default filter list and is
  instead configured via :data:`sys.warnoptions` when :option:`-b` is specified
  twice.
```

Overriding the default filter

Developers of applications written in Python may wish to hide *all* Python level warnings from their users by default, and only display them when running tests or otherwise working on the application. The :data:`sys.warnoptions` attribute used to pass filter configurations to the interpreter can be used as a marker to indicate whether or not warnings should be disabled:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 253); *backlink*

Unknown interpreted text role "data".

```
import sys

if not sys.warnoptions:
    import warnings
    warnings.simplefilter("ignore")
```

Developers of test runners for Python code are advised to instead ensure that *all* warnings are displayed by default for the code under test, using code like:

```
import sys

if not sys.warnoptions:
    import os, warnings
    warnings.simplefilter("default") # Change the filter in this process
    os.environ["PYTHONWARNINGS"] = "default" # Also affect subprocesses
```

Finally, developers of interactive shells that run user code in a namespace other than `__main__` are advised to ensure that :exc:`DeprecationWarning` messages are made visible by default, using code like the following (where `user_ns` is the module used to execute code entered interactively):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 276); *backlink*

Unknown interpreted text role "exc".

```
import warnings
warnings.filterwarnings("default", category=DeprecationWarning,
                                   module=user_ns.get("__name__"))
```

Temporarily Suppressing Warnings

If you are using code that you know will raise a warning, such as a deprecated function, but do not want to see the warning (even when warnings have been explicitly configured via the command line), then it is possible to suppress the warning using the :class:`catch_warnings` context manager:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 291); *backlink*

Unknown interpreted text role "class".

```
import warnings

def fxn():
    warnings.warn("deprecated", DeprecationWarning)

with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    fxn()
```

While within the context manager all warnings will simply be ignored. This allows you to use known-deprecated code without having to see the warning while not suppressing the warning for other code that might not be aware of its use of deprecated code. Note: this can only be guaranteed in a single-threaded application. If two or more threads use the :class:`catch_warnings` context manager at the same time, the behavior is undefined.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 305); *backlink*

Unknown interpreted text role "class".

Testing Warnings

To test warnings raised by code, use the :class:`catch_warnings` context manager. With it you can temporarily mutate the warnings filter to facilitate your testing. For instance, do the following to capture all raised warnings to check:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 319); *backlink*

Unknown interpreted text role "class".

```
import warnings

def fxn():
    warnings.warn("deprecated", DeprecationWarning)

with warnings.catch_warnings(record=True) as w:
    # Cause all warnings to always be triggered.
    warnings.simplefilter("always")
    # Trigger a warning.
    fxn()
    # Verify some things
    assert len(w) == 1
    assert issubclass(w[-1].category, DeprecationWarning)
    assert "deprecated" in str(w[-1].message)
```

One can also cause all warnings to be exceptions by using `error` instead of `always`. One thing to be aware of is that if a warning has already been raised because of a `once`/`default` rule, then no matter what filters are set the warning will not be seen again unless the

warnings registry related to the warning has been cleared.

Once the context manager exits, the warnings filter is restored to its state when the context was entered. This prevents tests from changing the warnings filter in unexpected ways between tests and leading to indeterminate test results. The :func:`showwarning` function in the module is also restored to its original value. Note: this can only be guaranteed in a single-threaded application. If two or more threads use the :class:`catch_warnings` context manager at the same time, the behavior is undefined.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst`, line 345); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst`, line 345); *backlink*

Unknown interpreted text role "class".

When testing multiple operations that raise the same kind of warning, it is important to test them in a manner that confirms each operation is raising a new warning (e.g. set warnings to be raised as exceptions and check the operations raise exceptions, check that the length of the warning list continues to increase after each operation, or else delete the previous entries from the warnings list before each new operation).

Updating Code For New Versions of Dependencies

Warning categories that are primarily of interest to Python developers (rather than end users of applications written in Python) are ignored by default.

Notably, this "ignored by default" list includes :exc:`DeprecationWarning` (for every module except `__main__`), which means developers should make sure to test their code with typically ignored warnings made visible in order to receive timely notifications of future breaking API changes (whether in the standard library or third party packages).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst`, line 369); *backlink*

Unknown interpreted text role "exc".

In the ideal case, the code will have a suitable test suite, and the test runner will take care of implicitly enabling all warnings when running tests (the test runner provided by the :mod:`unittest` module does this).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst`, line 375); *backlink*

Unknown interpreted text role "mod".

In less ideal cases, applications can be checked for use of deprecated interfaces by passing :option:`-Wd <-W>` to the Python interpreter (this is shorthand for :option:`!-W default`) or setting `PYTHONWARNINGS=default` in the environment. This enables default handling for all warnings, including those that are ignored by default. To change what action is taken for encountered warnings you can change what argument is passed to :option:`-W` (e.g. :option:`!-W error`). See the :option:`-W` flag for more details on what is possible.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst`, line 379); *backlink*

Unknown interpreted text role "option".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst`, line 379); *backlink*

Unknown interpreted text role "option".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst`, line 379); *backlink*

Unknown interpreted text role "option".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst`, line 379); *backlink*

Unknown interpreted text role "option".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst`, line 379); *backlink*

Unknown interpreted text role "option".

Available Functions

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst`, line 395)

Unknown directive type "function".

```
.. function:: warn(message, category=None, stacklevel=1, source=None)
```

```
   Issue a warning, or maybe ignore it or raise an exception.  The *category*
   argument, if given, must be a :ref:`warning category class <warning-categories>`; it
   defaults to :exc:`UserWarning`.  Alternatively, *message* can be a :exc:`Warning` instance,
   in which case *category* will be ignored and ``message.__class__`` will be used.
   In this case, the message text will be ``str(message)``. This function raises an
   exception if the particular warning issued is changed into an error by the
   :ref:`warnings filter <warning-filter>`.  The *stacklevel* argument can be used by wrapper
   functions written in Python, like this::

      def deprecation(message):
          warnings.warn(message, DeprecationWarning, stacklevel=2)

   This makes the warning refer to :func:`deprecation`'s caller, rather than to the
   source of :func:`deprecation` itself (since the latter would defeat the purpose
   of the warning message).

   *source*, if supplied, is the destroyed object which emitted a
   :exc:`ResourceWarning`.

   .. versionchanged:: 3.6
      Added *source* parameter.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 420)

Unknown directive type "function".

```
.. function:: warn_explicit(message, category, filename, lineno, module=None, registry=None, module_globals=None

   This is a low-level interface to the functionality of :func:`warn`, passing in
   explicitly the message, category, filename and line number, and optionally the
   module name and the registry (which should be the ``__warningregistry__``
   dictionary of the module).  The module name defaults to the filename with
   ``.py`` stripped; if no registry is passed, the warning is never suppressed.
   *message* must be a string and *category* a subclass of :exc:`Warning` or
   *message* may be a :exc:`Warning` instance, in which case *category* will be
   ignored.

   *module_globals*, if supplied, should be the global namespace in use by the code
   for which the warning is issued.  (This argument is used to support displaying
   source for modules found in zipfiles or other non-filesystem import
   sources).

   *source*, if supplied, is the destroyed object which emitted a
   :exc:`ResourceWarning`.

   .. versionchanged:: 3.6
      Add the *source* parameter.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 443)

Unknown directive type "function".

```
.. function:: showwarning(message, category, filename, lineno, file=None, line=None)

   Write a warning to a file.  The default implementation calls
   ``formatwarning(message, category, filename, lineno, line)`` and writes the
   resulting string to *file*, which defaults to :data:`sys.stderr`.  You may replace
   this function with any callable by assigning to ``warnings.showwarning``.
   *line* is a line of source code to be included in the warning
   message; if *line* is not supplied, :func:`showwarning` will
   try to read the line specified by *filename* and *lineno*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 454)

Unknown directive type "function".

```
.. function:: formatwarning(message, category, filename, lineno, line=None)

   Format a warning the standard way.  This returns a string which may contain
   embedded newlines and ends in a newline.  *line* is a line of source code to
   be included in the warning message; if *line* is not supplied,
   :func:`formatwarning` will try to read the line specified by *filename* and
   *lineno*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 463)

Unknown directive type "function".

```
.. function:: filterwarnings(action, message='', category=Warning, module='', lineno=0, append=False)

   Insert an entry into the list of :ref:`warnings filter specifications
   <warning-filter>`.  The entry is inserted at the front by default; if
   *append* is true, it is inserted at the end.  This checks the types of the
   arguments, compiles the *message* and *module* regular expressions, and
   inserts them as a tuple in the list of warnings filters.  Entries closer to
   the front of the list override entries later in the list, if both match a
```

```
particular warning.  Omitted arguments default to a value that matches
everything.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 475)

Unknown directive type "function".

```
.. function:: simplefilter(action, category=Warning, lineno=0, append=False)

   Insert a simple entry into the list of :ref:`warnings filter specifications
   <warning-filter>`.  The meaning of the function parameters is as for
   :func:`filterwarnings`, but regular expressions are not needed as the filter
   inserted always matches any message in any module as long as the category and
   line number match.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 484)

Unknown directive type "function".

```
.. function:: resetwarnings()

   Reset the warnings filter.  This discards the effect of all previous calls to
   :func:`filterwarnings`, including that of the :option:`-W` command line options
   and calls to :func:`simplefilter`.
```

Available Context Managers

A context manager that copies and, upon exit, restores the warnings filter and the :func:`showwarning` function. If the *record* argument is :const:`False` (the default) the context manager returns :class:`None` on entry. If *record* is :const:`True`, a list is returned that is progressively populated with objects as seen by a custom :func:`showwarning` function (which also suppresses output to `sys.stdout`). Each object in the list has attributes with the same names as the arguments to :func:`showwarning`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 496); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 496); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 496); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 496); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 496); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 496); *backlink*

Unknown interpreted text role "func".

The *module* argument takes a module that will be used instead of the module returned when you import :mod:`warnings` whose filter will be protected. This argument exists primarily for testing the :mod:`warnings` module itself.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 505); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst, line 505); *backlink*

Unknown interpreted text role "mod".

Note

The :class:`catch_warnings` manager works by replacing and then later restoring the module's :func:`showwarning`

function and internal list of filter specifications. This means the context manager is modifying global state and therefore is not thread-safe.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst`, **line 512**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]warnings.rst`, **line 512**); ***backlink***

Unknown interpreted text role "func".