Built-in Exceptions

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 6)

Unknown directive type "index".

```
.. index::
   statement: try
   statement: except
```

In Python, all exceptions must be instances of a class that derives from :class:`BaseException`. In a :keyword:`try` statement with an :keyword:`except` clause that mentions a particular class, that clause also handles any exception classes derived from that class (but not exception classes from which *it* is derived). Two exception classes that are not related via subclassing are never equivalent, even if they have the same name.

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "keyword".

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

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 17)

Unknown directive type "index".

```
.. index:: statement: raise
```

The built-in exceptions listed below can be generated by the interpreter or built-in functions. Except where mentioned, they have an "associated value" indicating the detailed cause of the error. This may be a string or a tuple of several items of information (e.g., an error code and a string explaining the code). The associated value is usually passed as arguments to the exception class's constructor.

User code can raise built-in exceptions. This can be used to test an exception handler or to report an error condition "just like" the situation in which the interpreter raises the same exception; but beware that there is nothing to prevent user code from raising an inappropriate error.

The built-in exception classes can be subclassed to define new exceptions; programmers are encouraged to derive new exceptions from the :exc:`Exception` class or one of its subclasses, and not from :exc:`BaseException`. More information on defining exceptions is available in the Python Tutorial under :ref:`tut-userexceptions`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 31); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 31); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 31); *backlink*

Unknown interpreted text role "ref".

Exception context

When raising a new exception while another exception is already being handled, the new exception's :attr:`__context__` attribute is automatically set to the handled exception. An exception may be handled when an :keyword:`except` or :keyword:`finally` clause, or a :keyword:`with` statement, is used.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 41); backlink

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 41); backlink

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 41); backlink

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 41); backlink

Unknown interpreted text role "keyword".

This implicit exception context can be supplemented with an explicit cause by using :keyword:`!from` with :keyword:`raise`:

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

Unknown interpreted text role "keyword".

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

Unknown interpreted text role "keyword".

```
raise new_exc from original_exc
```

The expression following :keyword:`from<raise>` must be an exception or `None`. It will be set as :attr:`__cause__` on the raised exception. Setting :attr:`__cause__` also implicitly sets the :attr:`__suppress_context__` attribute to `True`, so that using `raise new_exc from None` effectively replaces the old exception with the new one for display purposes (e.g. converting :exc:`KeyError` to :exc:`AttributeError`), while leaving the old exception available in :attr:`__context__` for introspection when debugging.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 53); backlink

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 53); backlink

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 53); backlink

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 53); backlink

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 53); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 53); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 53); *backlink*

Unknown interpreted text role "attr".

The default traceback display code shows these chained exceptions in addition to the traceback for the exception itself. An explicitly chained exception in :attr:`__cause__` is always shown when present. An implicitly chained exception in :attr:`__context__` is shown only if :attr:`__cause__` is :const:`None` and :attr:`__suppress_context__` is false.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 62); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 62); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 62); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 62); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 62); *backlink*

Unknown interpreted text role "attr".

In either case, the exception itself is always shown after any chained exceptions so that the final line of the traceback always shows the last exception that was raised.

Inheriting from built-in exceptions

User code can create subclasses that inherit from an exception type. It's recommended to only subclass one exception type at a time to avoid any possible conflicts between how the bases handle the `args` attribute, as well as due to possible memory layout incompatibilities.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 81)

Unknown directive type "impl-detail".

```
.. impl-detail::

   Most built-in exceptions are implemented in C for efficiency, see:
   :source:`Objects/exceptions.c`.  Some have custom memory layouts
   which makes it impossible to create a subclass that inherits from
   multiple exception types. The memory layout of a type is an implementation
   detail and might change between Python versions, leading to new
   conflicts in the future.  Therefore, it's recommended to avoid
   subclassing multiple exception types altogether.
```

Base classes

The following exceptions are used mostly as base classes for other exceptions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 97)

Unknown directive type "exception".

```
.. exception:: BaseException

   The base class for all built-in exceptions.  It is not meant to be directly
   inherited by user-defined classes (for that, use :exc:`Exception`).  If
   :func:`str` is called on an instance of this class, the representation of
   the argument(s) to the instance are returned, or the empty string when
   there were no arguments.

   .. attribute:: args

      The tuple of arguments given to the exception constructor.  Some built-in
      exceptions (like :exc:`OSError`) expect a certain number of arguments and
      assign a special meaning to the elements of this tuple, while others are
      usually called only with a single string giving an error message.

   .. method:: with_traceback(tb)

      This method sets *tb* as the new traceback for the exception and returns
      the exception object.  It was more commonly used before the exception
      chaining features of :pep:`3134` became available.  The following example
      shows how we can convert an instance of ``SomeException`` into an
      instance of ``OtherException`` while preserving the traceback.  Once
      raised, the current frame is pushed onto the traceback of the
      ``OtherException``, as would have happened to the traceback of the
      original ``SomeException`` had we allowed it to propagate to the caller. ::

         try:
             ...
         except SomeException:
             tb = sys.exc_info()[2]
             raise OtherException(...).with_traceback(tb)

   .. attribute:: __note__

      A mutable field which is :const:`None` by default and can be set to a string.
      If it is not :const:`None`, it is included in the traceback. This field can
      be used to enrich exceptions after they have been caught.

   .. versionadded:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 138)

Unknown directive type "exception".

```
.. exception:: Exception

   All built-in, non-system-exiting exceptions are derived from this class.  All
   user-defined exceptions should also be derived from this class.
```

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

Unknown directive type "exception".

```
.. exception:: ArithmeticError

   The base class for those built-in exceptions that are raised for various
   arithmetic errors: :exc:`OverflowError`, :exc:`ZeroDivisionError`,
   :exc:`FloatingPointError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 151)

Unknown directive type "exception".

```
.. exception:: BufferError

   Raised when a :ref:`buffer <bufferobjects>` related operation cannot be
   performed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 157)

Unknown directive type "exception".

```
.. exception:: LookupError

   The base class for the exceptions that are raised when a key or index used on
   a mapping or sequence is invalid: :exc:`IndexError`, :exc:`KeyError`.  This
   can be raised directly by :func:`codecs.lookup`.
```

Concrete exceptions

The following exceptions are the exceptions that are usually raised.

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

Unknown directive type "exception".

```
.. exception:: AssertionError

   .. index:: statement: assert

   Raised when an :keyword:`assert` statement fails.
```

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

Unknown directive type "exception".

```
.. exception:: AttributeError

   Raised when an attribute reference (see :ref:`attribute-references`) or
   assignment fails.  (When an object does not support attribute references or
   attribute assignments at all, :exc:`TypeError` is raised.)

   The :attr:`name` and :attr:`obj` attributes can be set using keyword-only
   arguments to the constructor. When set they represent the name of the attribute
   that was attempted to be accessed and the object that was accessed for said
   attribute, respectively.

   .. versionchanged:: 3.10
      Added the :attr:`name` and :attr:`obj` attributes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 190)

Unknown directive type "exception".

```
.. exception:: EOFError

   Raised when the :func:`input` function hits an end-of-file condition (EOF)
   without reading any data. (N.B.: the :meth:`io.IOBase.read` and
   :meth:`io.IOBase.readline` methods return an empty string when they hit EOF.)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 197)

Unknown directive type "exception".

```
.. exception:: FloatingPointError

   Not currently used.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 202)

Unknown directive type "exception".

```
.. exception:: GeneratorExit

   Raised when a :term:`generator` or :term:`coroutine` is closed;
   see :meth:`generator.close` and :meth:`coroutine.close`.  It
   directly inherits from :exc:`BaseException` instead of :exc:`Exception` since
   it is technically not an error.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 210)

Unknown directive type "exception".

```
.. exception:: ImportError

   Raised when the :keyword:`import` statement has troubles trying to
   load a module.  Also raised when the "from list" in ``from ... import``
   has a name that cannot be found.

   The :attr:`name` and :attr:`path` attributes can be set using keyword-only
   arguments to the constructor. When set they represent the name of the module
   that was attempted to be imported and the path to any file which triggered
   the exception, respectively.

   .. versionchanged:: 3.3
      Added the :attr:`name` and :attr:`path` attributes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 224)

Unknown directive type "exception".

```
.. exception:: ModuleNotFoundError

   A subclass of :exc:`ImportError` which is raised by :keyword:`import`
   when a module could not be located.  It is also raised when ``None``
   is found in :data:`sys.modules`.

   .. versionadded:: 3.6
```

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

Unknown directive type "exception".

```
.. exception:: IndexError

   Raised when a sequence subscript is out of range.  (Slice indices are
   silently truncated to fall in the allowed range; if an index is not an
   integer, :exc:`TypeError` is raised.)

   .. XXX xref to sequences
```

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

Unknown directive type "exception".

```
.. exception:: KeyError

   Raised when a mapping (dictionary) key is not found in the set of existing keys.

   .. XXX xref to mapping objects?
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 249)

Unknown directive type "exception".

```
.. exception:: KeyboardInterrupt

   Raised when the user hits the interrupt key (normally :kbd:`Control-C` or
   :kbd:`Delete`).  During execution, a check for interrupts is made
   regularly. The exception inherits from :exc:`BaseException` so as to not be
   accidentally caught by code that catches :exc:`Exception` and thus prevent
   the interpreter from exiting.

   .. note::

      Catching a :exc:`KeyboardInterrupt` requires special consideration.
      Because it can be raised at unpredictable points, it may, in some
      circumstances, leave the running program in an inconsistent state. It is
      generally best to allow :exc:`KeyboardInterrupt` to end the program as
      quickly as possible or avoid raising it entirely. (See
      :ref:`handlers-and-exceptions`.)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 267)

Unknown directive type "exception".

```
.. exception:: MemoryError

   Raised when an operation runs out of memory but the situation may still be
   rescued (by deleting some objects).  The associated value is a string indicating
   what kind of (internal) operation ran out of memory. Note that because of the
   underlying memory management architecture (C's :c:func:`malloc` function), the
   interpreter may not always be able to completely recover from this situation; it
   nevertheless raises an exception so that a stack traceback can be printed, in
   case a run-away program was the cause.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 278)

Unknown directive type "exception".

```
.. exception:: NameError

   Raised when a local or global name is not found.  This applies only to
   unqualified names.  The associated value is an error message that includes the
   name that could not be found.

   The :attr:`name` attribute can be set using a keyword-only argument to the
   constructor. When set it represent the name of the variable that was attempted
   to be accessed.

   .. versionchanged:: 3.10
      Added the :attr:`name` attribute.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 292)

Unknown directive type "exception".

```
.. exception:: NotImplementedError

   This exception is derived from :exc:`RuntimeError`.  In user defined base
   classes, abstract methods should raise this exception when they require
   derived classes to override the method, or while the class is being
   developed to indicate that the real implementation still needs to be added.

   .. note::

      It should not be used to indicate that an operator or method is not
      meant to be supported at all -- in that case either leave the operator /
      method undefined or, if a subclass, set it to :data:`None`.

   .. note::

      ``NotImplementedError`` and ``NotImplemented`` are not interchangeable,
      even though they have similar names and purposes.  See
      :data:`NotImplemented` for details on when to use it.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 311)

Unknown directive type "exception".

```
.. exception:: OSError([arg])
               OSError(errno, strerror[, filename[, winerror[, filename2]]])

   .. index:: module: errno

   This exception is raised when a system function returns a system-related
   error, including I/O failures such as "file not found" or "disk full"
   (not for illegal argument types or other incidental errors).

   The second form of the constructor sets the corresponding attributes,
   described below.  The attributes default to :const:`None` if not
   specified.  For backwards compatibility, if three arguments are passed,
   the :attr:`~BaseException.args` attribute contains only a 2-tuple
   of the first two constructor arguments.

   The constructor often actually returns a subclass of :exc:`OSError`, as
   described in `OS exceptions`_ below.  The particular subclass depends on
   the final :attr:`.errno` value.  This behaviour only occurs when
   constructing :exc:`OSError` directly or via an alias, and is not
   inherited when subclassing.

   .. attribute:: errno

      A numeric error code from the C variable :c:data:`errno`.

   .. attribute:: winerror

      Under Windows, this gives you the native
      Windows error code.  The :attr:`.errno` attribute is then an approximate
      translation, in POSIX terms, of that native error code.

      Under Windows, if the *winerror* constructor argument is an integer,
      the :attr:`.errno` attribute is determined from the Windows error code,
      and the *errno* argument is ignored.  On other platforms, the
      *winerror* argument is ignored, and the :attr:`winerror` attribute
      does not exist.

   .. attribute:: strerror

      The corresponding error message, as provided by
      the operating system.  It is formatted by the C
      functions :c:func:`perror` under POSIX, and :c:func:`FormatMessage`
      under Windows.

   .. attribute:: filename
                  filename2

      For exceptions that involve a file system path (such as :func:`open` or
      :func:`os.unlink`), :attr:`filename` is the file name passed to the function.
      For functions that involve two file system paths (such as
      :func:`os.rename`), :attr:`filename2` corresponds to the second
      file name passed to the function.
```

```
   .. versionchanged:: 3.3
      :exc:`EnvironmentError`, :exc:`IOError`, :exc:`WindowsError`,
      :exc:`socket.error`, :exc:`select.error` and
      :exc:`mmap.error` have been merged into :exc:`OSError`, and the
      constructor may return a subclass.

   .. versionchanged:: 3.4
      The :attr:`filename` attribute is now the original file name passed to
      the function, instead of the name encoded to or decoded from the
      :term:`filesystem encoding and error handler`. Also, the *filename2*
      constructor argument and attribute was added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 378)

Unknown directive type "exception".

```
.. exception:: OverflowError

   Raised when the result of an arithmetic operation is too large to be
   represented.  This cannot occur for integers (which would rather raise
   :exc:`MemoryError` than give up).  However, for historical reasons,
   OverflowError is sometimes raised for integers that are outside a required
   range.   Because of the lack of standardization of floating point exception
   handling in C, most floating point operations are not checked.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 388)

Unknown directive type "exception".

```
.. exception:: RecursionError

   This exception is derived from :exc:`RuntimeError`.  It is raised when the
   interpreter detects that the maximum recursion depth (see
   :func:`sys.getrecursionlimit`) is exceeded.

   .. versionadded:: 3.5
      Previously, a plain :exc:`RuntimeError` was raised.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 398)

Unknown directive type "exception".

```
.. exception:: ReferenceError

   This exception is raised when a weak reference proxy, created by the
   :func:`weakref.proxy` function, is used to access an attribute of the referent
   after it has been garbage collected. For more information on weak references,
   see the :mod:`weakref` module.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 406)

Unknown directive type "exception".

```
.. exception:: RuntimeError

   Raised when an error is detected that doesn't fall in any of the other
   categories.  The associated value is a string indicating what precisely went
   wrong.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 413)

Unknown directive type "exception".

```
.. exception:: StopIteration

   Raised by built-in function :func:`next` and an :term:`iterator`\'s
   :meth:`~iterator.__next__` method to signal that there are no further
   items produced by the iterator.

   The exception object has a single attribute :attr:`value`, which is
   given as an argument when constructing the exception, and defaults
   to :const:`None`.

   When a :term:`generator` or :term:`coroutine` function
   returns, a new :exc:`StopIteration` instance is
   raised, and the value returned by the function is used as the
   :attr:`value` parameter to the constructor of the exception.

   If a generator code directly or indirectly raises :exc:`StopIteration`,
   it is converted into a :exc:`RuntimeError` (retaining the
   :exc:`StopIteration` as the new exception's cause).

   .. versionchanged:: 3.3
      Added ``value`` attribute and the ability for generator functions to
      use it to return a value.

   .. versionchanged:: 3.5
      Introduced the RuntimeError transformation via
      ``from __future__ import generator_stop``, see :pep:`479`.

   .. versionchanged:: 3.7
      Enable :pep:`479` for all code by default: a :exc:`StopIteration`
      error raised in a generator is transformed into a :exc:`RuntimeError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 444)

Unknown directive type "exception".

```
.. exception:: StopAsyncIteration

   Must be raised by :meth:`__anext__` method of an
   :term:`asynchronous iterator` object to stop the iteration.

   .. versionadded:: 3.5
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 451)

Unknown directive type "exception".

```
.. exception:: SyntaxError(message, details)

   Raised when the parser encounters a syntax error.  This may occur in an
   :keyword:`import` statement, in a call to the built-in functions
   :func:`compile`, :func:`exec`,
   or :func:`eval`, or when reading the initial script or standard input
   (also interactively).

   The :func:`str` of the exception instance returns only the error message.
   Details is a tuple whose members are also available as separate attributes.

   .. attribute:: filename

      The name of the file the syntax error occurred in.

   .. attribute:: lineno

      Which line number in the file the error occurred in. This is
      1-indexed: the first line in the file has a ``lineno`` of 1.

   .. attribute:: offset

      The column in the line where the error occurred. This is
      1-indexed: the first character in the line has an ``offset`` of 1.

   .. attribute:: text
```

```
      The source code text involved in the error.

   .. attribute:: end_lineno

      Which line number in the file the error occurred ends in. This is
      1-indexed: the first line in the file has a ``lineno`` of 1.

   .. attribute:: end_offset

      The column in the end line where the error occurred finishes. This is
      1-indexed: the first character in the line has an ``offset`` of 1.

   For errors in f-string fields, the message is prefixed by "f-string: "
   and the offsets are offsets in a text constructed from the replacement
   expression.  For example, compiling f'Bad {a b} field' results in this
   args attribute: ('f-string: ...', ('', 1, 2, '(a b)\n', 1, 5)).

   .. versionchanged:: 3.10
      Added the :attr:`end_lineno` and :attr:`end_offset` attributes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 498)

Unknown directive type "exception".

```
.. exception:: IndentationError

   Base class for syntax errors related to incorrect indentation.  This is a
   subclass of :exc:`SyntaxError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 504)

Unknown directive type "exception".

```
.. exception:: TabError

   Raised when indentation contains an inconsistent use of tabs and spaces.
   This is a subclass of :exc:`IndentationError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 510)

Unknown directive type "exception".

```
.. exception:: SystemError

   Raised when the interpreter finds an internal error, but the situation does not
   look so serious to cause it to abandon all hope. The associated value is a
   string indicating what went wrong (in low-level terms).

   You should report this to the author or maintainer of your Python interpreter.
   Be sure to report the version of the Python interpreter (``sys.version``; it is
   also printed at the start of an interactive Python session), the exact error
   message (the exception's associated value) and if possible the source of the
   program that triggered the error.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 523)

Unknown directive type "exception".

```
.. exception:: SystemExit

   This exception is raised by the :func:`sys.exit` function.  It inherits from
   :exc:`BaseException` instead of :exc:`Exception` so that it is not accidentally
   caught by code that catches :exc:`Exception`.  This allows the exception to
   properly propagate up and cause the interpreter to exit.  When it is not
   handled, the Python interpreter exits; no stack traceback is printed.  The
   constructor accepts the same optional argument passed to :func:`sys.exit`.
```

```
   If the value is an integer, it specifies the system exit status (passed to
   C's :c:func:`exit` function); if it is ``None``, the exit status is zero; if
   it has another type (such as a string), the object's value is printed and
   the exit status is one.

   A call to :func:`sys.exit` is translated into an exception so that clean-up
   handlers (:keyword:`finally` clauses of :keyword:`try` statements) can be
   executed, and so that a debugger can execute a script without running the risk
   of losing control.  The :func:`os._exit` function can be used if it is
   absolutely positively necessary to exit immediately (for example, in the child
   process after a call to :func:`os.fork`).

   .. attribute:: code

      The exit status or error message that is passed to the constructor.
      (Defaults to ``None``.)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 549)

Unknown directive type "exception".

```
.. exception:: TypeError

   Raised when an operation or function is applied to an object of inappropriate
   type.  The associated value is a string giving details about the type mismatch.

   This exception may be raised by user code to indicate that an attempted
   operation on an object is not supported, and is not meant to be. If an object
   is meant to support a given operation but has not yet provided an
   implementation, :exc:`NotImplementedError` is the proper exception to raise.

   Passing arguments of the wrong type (e.g. passing a :class:`list` when an
   :class:`int` is expected) should result in a :exc:`TypeError`, but passing
   arguments with the wrong value (e.g. a number outside expected boundaries)
   should result in a :exc:`ValueError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 564)

Unknown directive type "exception".

```
.. exception:: UnboundLocalError

   Raised when a reference is made to a local variable in a function or method, but
   no value has been bound to that variable.  This is a subclass of
   :exc:`NameError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 571)

Unknown directive type "exception".

```
.. exception:: UnicodeError

   Raised when a Unicode-related encoding or decoding error occurs.  It is a
   subclass of :exc:`ValueError`.

   :exc:`UnicodeError` has attributes that describe the encoding or decoding
   error.  For example, ``err.object[err.start:err.end]`` gives the particular
   invalid input that the codec failed on.

   .. attribute:: encoding

       The name of the encoding that raised the error.

   .. attribute:: reason

       A string describing the specific codec error.

   .. attribute:: object

       The object the codec was attempting to encode or decode.
```

```
.. attribute:: start

    The first index of invalid data in :attr:`object`.

.. attribute:: end

    The index after the last invalid data in :attr:`object`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 601)

Unknown directive type "exception".

```
.. exception:: UnicodeEncodeError

   Raised when a Unicode-related error occurs during encoding.  It is a subclass of
   :exc:`UnicodeError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 607)

Unknown directive type "exception".

```
.. exception:: UnicodeDecodeError

   Raised when a Unicode-related error occurs during decoding.  It is a subclass of
   :exc:`UnicodeError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 613)

Unknown directive type "exception".

```
.. exception:: UnicodeTranslateError

   Raised when a Unicode-related error occurs during translating.  It is a subclass
   of :exc:`UnicodeError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 619)

Unknown directive type "exception".

```
.. exception:: ValueError

   Raised when an operation or function receives an argument that has the
   right type but an inappropriate value, and the situation is not described by a
   more precise exception such as :exc:`IndexError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 626)

Unknown directive type "exception".

```
.. exception:: ZeroDivisionError

   Raised when the second argument of a division or modulo operation is zero.  The
   associated value is a string indicating the type of the operands and the
   operation.
```

The following exceptions are kept for compatibility with previous versions; starting from Python 3.3, they are aliases of :exc:`OSError`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 633); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 636)

Unknown directive type "exception".

```
.. exception:: EnvironmentError
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 638)

Unknown directive type "exception".

```
.. exception:: IOError
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 640)

Unknown directive type "exception".

```
.. exception:: WindowsError

   Only available on Windows.
```

OS exceptions

The following exceptions are subclasses of :exc:`OSError`, they get raised depending on the system error code.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 648); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 651)

Unknown directive type "exception".

```
.. exception:: BlockingIOError

   Raised when an operation would block on an object (e.g. socket) set
   for non-blocking operation.
   Corresponds to :c:data:`errno` :py:data:`~errno.EAGAIN`, :py:data:`~errno.EALREADY`,
   :py:data:`~errno.EWOULDBLOCK` and :py:data:`~errno.EINPROGRESS`.

   In addition to those of :exc:`OSError`, :exc:`BlockingIOError` can have
   one more attribute:

   .. attribute:: characters_written

      An integer containing the number of characters written to the stream
      before it blocked.  This attribute is available when using the
      buffered I/O classes from the :mod:`io` module.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 667)

Unknown directive type "exception".

```
.. exception:: ChildProcessError

   Raised when an operation on a child process failed.
   Corresponds to :c:data:`errno` :py:data:`~errno.ECHILD`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 672)

Unknown directive type "exception".

```
.. exception:: ConnectionError

   A base class for connection-related issues.

   Subclasses are :exc:`BrokenPipeError`, :exc:`ConnectionAbortedError`,
   :exc:`ConnectionRefusedError` and :exc:`ConnectionResetError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 679)

Unknown directive type "exception".

```
.. exception:: BrokenPipeError

   A subclass of :exc:`ConnectionError`, raised when trying to write on a
   pipe while the other end has been closed, or trying to write on a socket
   which has been shutdown for writing.
   Corresponds to :c:data:`errno` :py:data:`~errno.EPIPE` and :py:data:`~errno.ESHUTDOWN`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 686)

Unknown directive type "exception".

```
.. exception:: ConnectionAbortedError

   A subclass of :exc:`ConnectionError`, raised when a connection attempt
   is aborted by the peer.
   Corresponds to :c:data:`errno` :py:data:`~errno.ECONNABORTED`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 692)

Unknown directive type "exception".

```
.. exception:: ConnectionRefusedError

   A subclass of :exc:`ConnectionError`, raised when a connection attempt
   is refused by the peer.
   Corresponds to :c:data:`errno` :py:data:`~errno.ECONNREFUSED`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 698)

Unknown directive type "exception".

```
.. exception:: ConnectionResetError

   A subclass of :exc:`ConnectionError`, raised when a connection is
   reset by the peer.
   Corresponds to :c:data:`errno` :py:data:`~errno.ECONNRESET`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 704)

Unknown directive type "exception".

```
.. exception:: FileExistsError

   Raised when trying to create a file or directory which already exists.
   Corresponds to :c:data:`errno` :py:data:`~errno.EEXIST`.
```

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

main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 709)

Unknown directive type "exception".

```
.. exception:: FileNotFoundError

   Raised when a file or directory is requested but doesn't exist.
   Corresponds to :c:data:`errno` :py:data:`~errno.ENOENT`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 714)

Unknown directive type "exception".

```
.. exception:: InterruptedError

   Raised when a system call is interrupted by an incoming signal.
   Corresponds to :c:data:`errno` :py:data:`~errno.EINTR`.

   .. versionchanged:: 3.5
      Python now retries system calls when a syscall is interrupted by a
      signal, except if the signal handler raises an exception (see :pep:`475`
      for the rationale), instead of raising :exc:`InterruptedError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 724)

Unknown directive type "exception".

```
.. exception:: IsADirectoryError

   Raised when a file operation (such as :func:`os.remove`) is requested
   on a directory.
   Corresponds to :c:data:`errno` :py:data:`~errno.EISDIR`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 730)

Unknown directive type "exception".

```
.. exception:: NotADirectoryError

   Raised when a directory operation (such as :func:`os.listdir`) is requested on
   something which is not a directory.  On most POSIX platforms, it may also be
   raised if an operation attempts to open or traverse a non-directory file as if
   it were a directory.
   Corresponds to :c:data:`errno` :py:data:`~errno.ENOTDIR`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 738)

Unknown directive type "exception".

```
.. exception:: PermissionError

   Raised when trying to run an operation without the adequate access
   rights - for example filesystem permissions.
   Corresponds to :c:data:`errno` :py:data:`~errno.EACCES` and :py:data:`~errno.EPERM`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 744)

Unknown directive type "exception".

```
.. exception:: ProcessLookupError

   Raised when a given process doesn't exist.
   Corresponds to :c:data:`errno` :py:data:`~errno.ESRCH`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 749)

Unknown directive type "exception".

```
.. exception:: TimeoutError

   Raised when a system function timed out at the system level.
   Corresponds to :c:data:`errno` :py:data:`~errno.ETIMEDOUT`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 754)

Unknown directive type "versionadded".

```
.. versionadded:: 3.3
   All the above :exc:`OSError` subclasses were added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 758)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`3151` - Reworking the OS and IO exception hierarchy
```

Warnings

The following exceptions are used as warning categories; see the :ref:`warning-categories` documentation for more details.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 768); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 771)

Unknown directive type "exception".

```
.. exception:: Warning

   Base class for warning categories.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 776)

Unknown directive type "exception".

```
.. exception:: UserWarning

   Base class for warnings generated by user code.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 781)

Unknown directive type "exception".

```
.. exception:: DeprecationWarning

   Base class for warnings about deprecated features when those warnings are
   intended for other Python developers.
```

```
Ignored by the default warning filters, except in the ``__main__`` module
(:pep:`565`). Enabling the :ref:`Python Development Mode <devmode>` shows
this warning.

The deprecation policy is described in :pep:`387`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 793)

Unknown directive type "exception".

```
.. exception:: PendingDeprecationWarning

   Base class for warnings about features which are obsolete and
   expected to be deprecated in the future, but are not deprecated
   at the moment.

   This class is rarely used as emitting a warning about a possible
   upcoming deprecation is unusual, and :exc:`DeprecationWarning`
   is preferred for already active deprecations.

   Ignored by the default warning filters. Enabling the :ref:`Python
   Development Mode <devmode>` shows this warning.

   The deprecation policy is described in :pep:`387`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 809)

Unknown directive type "exception".

```
.. exception:: SyntaxWarning

   Base class for warnings about dubious syntax.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 814)

Unknown directive type "exception".

```
.. exception:: RuntimeWarning

   Base class for warnings about dubious runtime behavior.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 819)

Unknown directive type "exception".

```
.. exception:: FutureWarning

   Base class for warnings about deprecated features when those warnings are
   intended for end users of applications that are written in Python.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 825)

Unknown directive type "exception".

```
.. exception:: ImportWarning

   Base class for warnings about probable mistakes in module imports.

   Ignored by the default warning filters. Enabling the :ref:`Python
   Development Mode <devmode>` shows this warning.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 833)

Unknown directive type "exception".

```
.. exception:: UnicodeWarning

   Base class for warnings related to Unicode.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 838)

Unknown directive type "exception".

```
.. exception:: EncodingWarning

   Base class for warnings related to encodings.

   See :ref:`io-encoding-warning` for details.

   .. versionadded:: 3.10
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 847)

Unknown directive type "exception".

```
.. exception:: BytesWarning

   Base class 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]exceptions.rst, line 852)

Unknown directive type "exception".

```
.. exception:: ResourceWarning

   Base class for warnings related to resource usage.

   Ignored by the default warning filters. Enabling the :ref:`Python
   Development Mode <devmode>` shows this warning.

   .. versionadded:: 3.2
```

Exception groups

The following are used when it is necessary to raise multiple unrelated exceptions. They are part of the exception hierarchy so they can be handled with :keyword:`except` like all other exceptions. In addition, they are recognised by :keyword:`except* <except_star>`, which matches their subgroups based on the types of the contained exceptions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 865); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 865); *backlink*

Unknown interpreted text role "keyword".

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

main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 871)

Unknown directive type "exception".

```
.. exception:: ExceptionGroup(msg, excs)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 872)

Unknown directive type "exception".

```
.. exception:: BaseExceptionGroup(msg, excs)

   Both of these exception types wrap the exceptions in the sequence ``excs``.
   The ``msg`` parameter must be a string. The difference between the two
   classes is that :exc:`BaseExceptionGroup` extends :exc:`BaseException` and
   it can wrap any exception, while :exc:`ExceptionGroup` extends :exc:`Exception`
   and it can only wrap subclasses of :exc:`Exception`. This design is so that
   ``except Exception`` catches an :exc:`ExceptionGroup` but not
   :exc:`BaseExceptionGroup`.

   The :exc:`BaseExceptionGroup` constructor returns an :exc:`ExceptionGroup`
   rather than a :exc:`BaseExceptionGroup` if all contained exceptions are
   :exc:`Exception` instances, so it can be used to make the selection
   automatic. The :exc:`ExceptionGroup` constructor, on the other hand,
   raises a :exc:`TypeError` if any contained exception is not an
   :exc:`Exception` subclass.

   .. attribute:: message

      The ``msg`` argument to the constructor. This is a read-only attribute.

   .. attribute:: exceptions

      A tuple of the exceptions in the ``excs`` sequence given to the
      constructor. This is a read-only attribute.

   .. method:: subgroup(condition)

      Returns an exception group that contains only the exceptions from the
      current group that match *condition*, or ``None`` if the result is empty.

      The condition can be either a function that accepts an exception and returns
      true for those that should be in the subgroup, or it can be an exception type
      or a tuple of exception types, which is used to check for a match using the
      same check that is used in an ``except`` clause.

      The nesting structure of the current exception is preserved in the result,
      as are the values of its :attr:`message`, :attr:`__traceback__`,
      :attr:`__cause__`, :attr:`__context__` and :attr:`__note__` fields.
      Empty nested groups are omitted from the result.

      The condition is checked for all exceptions in the nested exception group,
      including the top-level and any nested exception groups. If the condition is
      true for such an exception group, it is included in the result in full.

   .. method:: split(condition)

      Like :meth:`subgroup`, but returns the pair ``(match, rest)`` where ``match``
      is ``subgroup(condition)`` and ``rest`` is the remaining non-matching
      part.

   .. method:: derive(excs)

      Returns an exception group with the same :attr:`message`,
      :attr:`__traceback__`, :attr:`__cause__`, :attr:`__context__`
      and :attr:`__note__` but which wraps the exceptions in ``excs``.

      This method is used by :meth:`subgroup` and :meth:`split`. A
      subclass needs to override it in order to make :meth:`subgroup`
      and :meth:`split` return instances of the subclass rather
      than :exc:`ExceptionGroup`. ::

         >>> class MyGroup(ExceptionGroup):
         ...     def derive(self, exc):
         ...         return MyGroup(self.message, exc)
         ...
         >>> MyGroup("eg", [ValueError(1), TypeError(2)]).split(TypeError)
         (MyGroup('eg', [TypeError(2)]), MyGroup('eg', [ValueError(1)]))
```

```
Note that :exc:`BaseExceptionGroup` defines :meth:`__new__`, so
subclasses that need a different constructor signature need to
override that rather than :meth:`__init__`. For example, the following
defines an exception group subclass which accepts an exit_code and
and constructs the group's message from it. ::

   class Errors(ExceptionGroup):
      def __new__(cls, errors, exit_code):
         self = super().__new__(Errors, f"exit code: {exit_code}", errors)
         self.exit_code = exit_code
         return self

      def derive(self, excs):
         return Errors(excs, self.exit_code)

.. versionadded:: 3.11
```

Exception hierarchy

The class hierarchy for built-in exceptions is:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]exceptions.rst, line 964)

Unknown directive type "literalinclude".

```
.. literalinclude:: ../../Lib/test/exception_hierarchy.txt
  :language: text
```