:mod:`doctest` --- Test interactive Python examples

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

Unknown directive type "module".

```
.. module:: doctest
   :synopsis: Test pieces of code within docstrings.
```

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

Unknown directive type "moduleauthor".

```
.. moduleauthor:: Tim Peters <tim@python.org>
```

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

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Tim Peters <tim@python.org>
```

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

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Moshe Zadka <moshez@debian.org>
```

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

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Edward Loper <edloper@users.sourceforge.net>
```

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 12); *backlink*

Unknown interpreted text role "source".

The :mod:`doctest` module searches for pieces of text that look like interactive Python sessions, and then executes those sessions to verify that they work exactly as shown. There are several common ways to use doctest:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 16); *backlink*

Unknown interpreted text role "mod".

- To check that a module's docstrings are up-to-date by verifying that all interactive examples still work as documented.
- To perform regression testing by verifying that interactive examples from a test file or a test object work as expected.
- To write tutorial documentation for a package, liberally illustrated with input-output examples. Depending on whether the examples or the expository text are emphasized, this has the flavor of "literate testing" or "executable documentation".

Here's a complete but small example module:

```
"""
This is the "example" module.

The example module supplies one function, factorial().  For example,

>>> factorial(5)
120
"""

def factorial(n):
    """Return the factorial of n, an exact integer >= 0.

    >>> [factorial(n) for n in range(6)]
    [1, 1, 2, 6, 24, 120]
    >>> factorial(30)
    265252859812191058636308480000000
    >>> factorial(-1)
    Traceback (most recent call last):
        ...
    ValueError: n must be >= 0

    Factorials of floats are OK, but the float must be an exact integer:
    >>> factorial(30.1)
    Traceback (most recent call last):
        ...
    ValueError: n must be exact integer
    >>> factorial(30.0)
    265252859812191058636308480000000

    It must also not be ridiculously large:
    >>> factorial(1e100)
    Traceback (most recent call last):
        ...
    OverflowError: n too large
    """

    import math
    if not n >= 0:
        raise ValueError("n must be >= 0")
    if math.floor(n) != n:
        raise ValueError("n must be exact integer")
    if n+1 == n:  # catch a value like 1e300
        raise OverflowError("n too large")
    result = 1
    factor = 2
    while factor <= n:
```

```
        result *= factor
        factor += 1
    return result


if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

If you run :file:`example.py` directly from the command line, :mod:`doctest` works its magic:

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

Unknown interpreted text role "file".

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

Unknown interpreted text role "mod".

```
$ python example.py
$
```

There's no output! That's normal, and it means all the examples worked. Pass `-v` to the script, and :mod:`doctest` prints a detailed log of what it's trying, and prints a summary at the end:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 96); backlink

Unknown interpreted text role "mod".

```
$ python example.py -v
Trying:
    factorial(5)
Expecting:
    120
ok
Trying:
    [factorial(n) for n in range(6)]
Expecting:
    [1, 1, 2, 6, 24, 120]
ok
```

And so on, eventually ending with:

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 116)

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

```
.. code-block:: none

   Trying:
       factorial(1e100)
   Expecting:
       Traceback (most recent call last):
           ...
       OverflowError: n too large
   ok
   2 items passed all tests:
      1 tests in __main__
      8 tests in __main__.factorial
   9 tests in 2 items.
   9 passed and 0 failed.
   Test passed.
   $
```

That's all you need to know to start making productive use of :mod:`doctest`! Jump in. The following sections provide full details. Note that there are many examples of doctests in the standard Python test suite and libraries. Especially useful examples can be found in the standard test file :file:`Lib/test/test_doctest.py`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 133); backlink

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 133); backlink

Unknown interpreted text role "file".

Simple Usage: Checking Examples in Docstrings

The simplest way to start using doctest (but not necessarily the way you'll continue to do it) is to end each module :mod:`M` with:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 145); backlink

Unknown interpreted text role "mod".

```
if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

:mod:`doctest` then examines docstrings in module :mod:`M`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 152); backlink

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 152); backlink

Unknown interpreted text role "mod".

Running the module as a script causes the examples in the docstrings to get executed and verified:

```
python M.py
```

This won't display anything unless an example fails, in which case the failing example(s) and the cause(s) of the failure(s) are printed to stdout, and the final line of output is `***Test Failed*** N failures.`, where *N* is the number of examples that failed.

Run it with the `-v` switch instead:

```
python M.py -v
```

and a detailed report of all examples tried is printed to standard output, along with assorted summaries at the end.

You can force verbose mode by passing `verbose=True` to :func:`testmod`, or prohibit it by passing `verbose=False`. In either of those cases, `sys.argv` is not examined by :func:`testmod` (so passing `-v` or not has no effect).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 171); backlink

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 171); backlink

Unknown interpreted text role "func".

There is also a command line shortcut for running :func:`testmod`. You can instruct the Python interpreter to run the doctest module directly from the standard library and pass the module name(s) on the command line:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 176); backlink

Unknown interpreted text role "func".

```
python -m doctest -v example.py
```

This will import :file:`example.py` as a standalone module and run :func:`testmod` on it. Note that this may not work correctly if the file is part of a package and imports other submodules from that package.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 182); backlink

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 182); backlink

Unknown interpreted text role "func".

For more information on :func:`testmod`, see section :ref:`doctest-basic-api`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 186); backlink

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 186); backlink

Unknown interpreted text role "ref".

Simple Usage: Checking Examples in a Text File

Another simple application of doctest is testing interactive examples in a text file. This can be done with the :func:`testfile` function:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 194); backlink

Unknown interpreted text role "func".

```
import doctest
doctest.testfile("example.txt")
```

That short script executes and verifies any interactive Python examples contained in the file :file:`example.txt`. The file content is treated as if it were a single giant docstring; the file doesn't need to contain a Python program! For example, perhaps :file:`example.txt` contains this:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 200); backlink

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 200); backlink

Unknown interpreted text role "file".

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 205)

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

```
.. code-block:: none

   The ``example`` module
   ======================

   Using ``factorial``
   -------------------

   This is an example text file in reStructuredText format.  First import
   ``factorial`` from the ``example`` module:

       >>> from example import factorial

   Now use it:

       >>> factorial(6)
       120
```

Running `doctest.testfile("example.txt")` then finds the error in this documentation:

```
File "./example.txt", line 14, in example.txt
Failed example:
    factorial(6)
Expected:
    120
Got:
    720
```

As with :func:`testmod`, :func:`testfile` won't display anything unless an example fails. If an example does fail, then the failing example(s) and the cause(s) of the failure(s) are printed to stdout, using the same format as :func:`testmod`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 234); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 234); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 234); *backlink*

Unknown interpreted text role "func".

By default, :func:`testfile` looks for files in the calling module's directory. See section :ref:`doctest-basic-api` for a description of the optional arguments that can be used to tell it to look for files in other locations.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 239); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 239); *backlink*

Unknown interpreted text role "ref".

Like :func:`testmod`, :func:`testfile`'s verbosity can be set with the `-v` command-line switch or with the optional keyword argument *verbose*.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 243); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 243); *backlink*

Unknown interpreted text role "func".

There is also a command line shortcut for running :func:`testfile`. You can instruct the Python interpreter to run the doctest module directly from the standard library and pass the file name(s) on the command line:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 247); *backlink*

Unknown interpreted text role "func".

```
python -m doctest -v example.txt
```

Because the file name does not end with :file:`.py`, :mod:`doctest` infers that it must be run with :func:`testfile`, not :func:`testmod`.

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

Unknown interpreted text role "file".

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

Unknown interpreted text role "mod".

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

Unknown interpreted text role "func".

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

Unknown interpreted text role "func".

For more information on :func:`testfile`, see section :ref:`doctest-basic-api`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 256); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 256); *backlink*

Unknown interpreted text role "ref".

How It Works

This section examines in detail how doctest works: which docstrings it looks at, how it finds interactive examples, what execution context it uses, how it handles exceptions, and how option flags can be used to control its behavior. This is the information that you need to know to write doctest examples; for information about actually running doctest on these examples, see the following sections.

Which Docstrings Are Examined?

The module docstring, and all function, class and method docstrings are searched. Objects imported into the module are not searched.

In addition, if `M.__test__` exists and "is true", it must be a dict, and each entry maps a (string) name to a function object, class object, or string. Function and class object docstrings found from `M.__test__` are searched, and strings are treated as if they were docstrings. In output, a key `K` in `M.__test__` appears with name

```
<name of M>.__test__.K
```

Any classes found are recursively searched similarly, to test docstrings in their contained methods and nested classes.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 291)

Unknown directive type "impl-detail".

```
.. impl-detail::
   Prior to version 3.4, extension modules written in C were not fully
   searched by doctest.
```

How are Docstring Examples Recognized?

In most cases a copy-and-paste of an interactive console session works fine, but doctest isn't trying to do an exact emulation of any specific Python shell.

```
>>> # comments are ignored
>>> x = 12
>>> x
12
>>> if x == 13:
...     print("yes")
... else:
...     print("no")
...     print("NO")
...     print("NO!!!")
...
no
NO
NO!!!
>>>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 322)

Unknown directive type "index".

```
.. index::
   single: >>>; interpreter prompt
   single: ...; interpreter prompt
```

Any expected output must immediately follow the final `'>>> '` or `'... '` line containing the code, and the expected output (if any) extends to the next `'>>> '` or all-whitespace line.

The fine print:

- Expected output cannot contain an all-whitespace line, since such a line is taken to signal the end of expected output. If expected output does contain a blank line, put `<BLANKLINE>` in your doctest example each place a blank line is expected.
- All hard tab characters are expanded to spaces, using 8-column tab stops. Tabs in output generated by the tested code are not modified. Because any hard tabs in the sample output *are* expanded, this means that if the code output includes hard tabs, the only way the doctest can pass is if the :const:`NORMALIZE_WHITESPACE` option or :ref:`directive <doctest-directives>` is in effect. Alternatively, the test can be rewritten to capture the output and compare it to an expected value as part of the test. This handling of tabs in the source was arrived at through trial and error, and has proven to be the least error prone way of handling them. It is possible to use a different algorithm for handling tabs by writing a custom :class:`DocTestParser` class.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 337); *backlink*

  Unknown interpreted text role "const".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 337); *backlink*

  Unknown interpreted text role "ref".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 337); *backlink*

  Unknown interpreted text role "class".

- Output to stdout is captured, but not output to stderr (exception tracebacks are captured via a different means).
- If you continue a line via backslashing in an interactive session, or for any other reason use a backslash, you should use a raw docstring, which will preserve your backslashes exactly as you type them:

  ```
  >>> def f(x):
  ...     r'''Backslashes in a raw docstring: m\n'''
  >>> print(f.__doc__)
  Backslashes in a raw docstring: m\n
  ```

  Otherwise, the backslash will be interpreted as part of the string. For example, the `\n` above would be interpreted as a newline character. Alternatively, you can double each backslash in the doctest version (and not use a raw string):

  ```
  >>> def f(x):
  ...     '''Backslashes in a raw docstring: m\\n'''
  >>> print(f.__doc__)
  Backslashes in a raw docstring: m\n
  ```

- The starting column doesn't matter:

  ```
  >>> assert "Easy!"
        >>> import math
            >>> math.floor(1.9)
            1
  ```

  and as many leading whitespace characters are stripped from the expected output as appeared in the initial `'>>> '` line that started the example.

What's the Execution Context?

By default, each time :mod:`doctest` finds a docstring to test, it uses a *shallow copy* of :mod:`M`'s globals, so that running tests doesn't change the module's real globals, and so that one test in :mod:`M` can't leave behind crumbs that accidentally allow another test to work. This means examples can freely use any names defined at top-level in :mod:`M`, and names defined earlier in the docstring being run. Examples cannot see names defined in other docstrings.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 386); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 386); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 386); backlink

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 386); backlink

Unknown interpreted text role "mod".

You can force use of your own dict as the execution context by passing `globs=your_dict` to :func:`testmod` or :func:`testfile` instead.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 394); backlink

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 394); backlink

Unknown interpreted text role "func".

What About Exceptions?

No problem, provided that the traceback is the only output produced by the example: just paste in the traceback. [1] Since tracebacks contain details that are likely to change rapidly (for example, exact file paths and line numbers), this is one case where doctest works hard to be flexible in what it accepts.

Simple example:

```
>>> [1, 2, 3].remove(42)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: list.remove(x): x not in list
```

That doctest succeeds if :exc:`ValueError` is raised, with the `list.remove(x): x not in list` detail as shown.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 416); backlink

Unknown interpreted text role "exc".

The expected output for an exception must start with a traceback header, which may be either of the following two lines, indented the same as the first line of the example:

```
Traceback (most recent call last):
Traceback (innermost last):
```

The traceback header is followed by an optional traceback stack, whose contents are ignored by doctest. The traceback stack is typically omitted, or copied verbatim from an interactive session.

The traceback stack is followed by the most interesting part: the line(s) containing the exception type and detail. This is usually the last line of a traceback, but can extend across multiple lines if the exception has a multi-line detail:

```
>>> raise ValueError('multi\n    line\ndetail')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: multi
    line
detail
```

The last three lines (starting with :exc:`ValueError`) are compared against the exception's type and detail, and the rest are ignored.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 442); backlink

Unknown interpreted text role "exc".

Best practice is to omit the traceback stack, unless it adds significant documentation value to the example. So the last example is probably better as:

```
>>> raise ValueError('multi\n    line\ndetail')
Traceback (most recent call last):
    ...
ValueError: multi
    line
detail
```

Note that tracebacks are treated very specially. In particular, in the rewritten example, the use of `...` is independent of doctest's :const:`ELLIPSIS` option. The ellipsis in that example could be left out, or could just as well be three (or three hundred) commas or digits, or an indented transcript of a Monty Python skit.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 455); backlink

Unknown interpreted text role "const".

Some details you should read once, but won't need to remember:

- Doctest can't guess whether your expected output came from an exception traceback or from ordinary printing. So, e.g., an example that expects `ValueError: 42 is prime` will pass whether :exc:`ValueError` is actually raised or if the example merely prints that traceback text. In practice, ordinary output rarely begins with a traceback header line, so this doesn't create real problems.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 463); backlink

  Unknown interpreted text role "exc".

- Each line of the traceback stack (if present) must be indented further than the first line of the example, *or* start with a non-alphanumeric character. The first line following the traceback header indented the same and starting with an alphanumeric is taken to be the start of the exception detail. Of course this does the right thing for genuine tracebacks.
- When the :const:`IGNORE_EXCEPTION_DETAIL` doctest option is specified, everything following the leftmost colon and any module information in the exception name is ignored.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 476); backlink

Unknown interpreted text role "const".

- The interactive shell omits the traceback header line for some :exc:`SyntaxError`s. But doctest uses the traceback header line to distinguish exceptions from non-exceptions. So in the rare case where you need to test a :exc:`SyntaxError` that omits the traceback header, you will need to manually add the traceback header line to your test example.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 480); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 480); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 486)

Unknown directive type "index".

```
.. index:: single: ^ (caret); marker
```

- For some exceptions, Python displays the position of the error using ^ markers and tildes:

```
>>> 1 + None
  File "<stdin>", line 1
    1 + None
    ~~^~~~~
TypeError: unsupported operand type(s) for +: 'int' and 'NoneType'
```

Since the lines showing the position of the error come before the exception type and detail, they are not checked by doctest. For example, the following test would pass, even though it puts the ^ marker in the wrong location:

```
>>> 1 + None
  File "<stdin>", line 1
    1 + None
    ^~~~~~~
TypeError: unsupported operand type(s) for +: 'int' and 'NoneType'
```

Option Flags

A number of option flags control various aspects of doctest's behavior. Symbolic names for the flags are supplied as module constants, which can be :ref:`bitwise ORed <bitwise>` together and passed to various functions. The names can also be used in :ref:`doctest directives <doctest-directives>`, and may be passed to the doctest command line interface via the -o option.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 514); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 514); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 520)

Unknown directive type "versionadded".

```
.. versionadded:: 3.4
   The ``-o`` command line option.
```

The first group of options define test semantics, controlling aspects of how doctest decides whether actual output matches an example's expected output:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 527)

Unknown directive type "data".

```
.. data:: DONT_ACCEPT_TRUE_FOR_1

   By default, if an expected output block contains just ``1``, an actual output
   block containing just ``1`` or just ``True`` is considered to be a match, and
   similarly for ``0`` versus ``False``.  When :const:`DONT_ACCEPT_TRUE_FOR_1` is
   specified, neither substitution is allowed.  The default behavior caters to that
   Python changed the return type of many functions from integer to boolean;
   doctests expecting "little integer" output still work in these cases.  This
   option will probably go away, but not for several years.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 538)

Unknown directive type "index".

```
.. index:: single: <BLANKLINE>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 539)

Unknown directive type "data".

```
.. data:: DONT_ACCEPT_BLANKLINE

   By default, if an expected output block contains a line containing only the
   string ``<BLANKLINE>``, then that line will match a blank line in the actual
   output.  Because a genuinely blank line delimits the expected output, this is
   the only way to communicate that a blank line is expected.  When
   :const:`DONT_ACCEPT_BLANKLINE` is specified, this substitution is not allowed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 548)

Unknown directive type "data".

```
.. data:: NORMALIZE_WHITESPACE

   When specified, all sequences of whitespace (blanks and newlines) are treated as
   equal.  Any sequence of whitespace within the expected output will match any
   sequence of whitespace within the actual output. By default, whitespace must
   match exactly. :const:`NORMALIZE_WHITESPACE` is especially useful when a line of
   expected output is very long, and you want to wrap it across multiple lines in
   your source.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 558)

Unknown directive type "index".

```
.. index:: single: ...; in doctests
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 559)

Unknown directive type "data".

```
.. data:: ELLIPSIS

   When specified, an ellipsis marker (``...``) in the expected output can match
   any substring in the actual output.  This includes substrings that span line
   boundaries, and empty substrings, so it's best to keep usage of this simple.
   Complicated uses can lead to the same kinds of "oops, it matched too much!"
   surprises that ``.*`` is prone to in regular expressions.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 568)

Unknown directive type "data".

```
.. data:: IGNORE_EXCEPTION_DETAIL

   When specified, an example that expects an exception passes if an exception of
   the expected type is raised, even if the exception detail does not match.  For
   example, an example expecting ``ValueError: 42`` will pass if the actual
   exception raised is ``ValueError: 3*14``, but will fail, e.g., if
   :exc:`TypeError` is raised.

   It will also ignore the module name used in Python 3 doctest reports. Hence
   both of these variations will work with the flag specified, regardless of
   whether the test is run under Python 2.7 or Python 3.2 (or later versions)::

      >>> raise CustomError('message')
      Traceback (most recent call last):
      CustomError: message

      >>> raise CustomError('message')
      Traceback (most recent call last):
      my_module.CustomError: message

   Note that :const:`ELLIPSIS` can also be used to ignore the
   details of the exception message, but such a test may still fail based
   on whether or not the module details are printed as part of the
   exception name. Using :const:`IGNORE_EXCEPTION_DETAIL` and the details
   from Python 2.3 is also the only clear way to write a doctest that doesn't
   care about the exception detail yet continues to pass under Python 2.3 or
   earlier (those releases do not support :ref:`doctest directives
   <doctest-directives>` and ignore them as irrelevant comments). For example::

      >>> (1, 2)[3] = 'moo'
      Traceback (most recent call last):
        File "<stdin>", line 1, in <module>
      TypeError: object doesn't support item assignment

   passes under Python 2.3 and later Python versions with the flag specified,
   even though the detail
   changed in Python 2.4 to say "does not" instead of "doesn't".

   .. versionchanged:: 3.2
      :const:`IGNORE_EXCEPTION_DETAIL` now also ignores any information relating
      to the module containing the exception under test.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 611)

Unknown directive type "data".

```
.. data:: SKIP

   When specified, do not run the example at all.  This can be useful in contexts
   where doctest examples serve as both documentation and test cases, and an
   example should be included for documentation purposes, but should not be
   checked.  E.g., the example's output might be random; or the example might
   depend on resources which would be unavailable to the test driver.

   The SKIP flag can also be used for temporarily "commenting out" examples.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 622)

Unknown directive type "data".

```
.. data:: COMPARISON_FLAGS

   A bitmask or'ing together all the comparison flags above.
```

The second group of options controls how test failures are reported:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 629)

Unknown directive type "data".

```
.. data:: REPORT_UDIFF

   When specified, failures that involve multi-line expected and actual outputs are
   displayed using a unified diff.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 635)

Unknown directive type "data".

```
.. data:: REPORT_CDIFF

   When specified, failures that involve multi-line expected and actual outputs
   will be displayed using a context diff.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 641)

Unknown directive type "data".

```
.. data:: REPORT_NDIFF

   When specified, differences are computed by ``difflib.Differ``, using the same
   algorithm as the popular :file:`ndiff.py` utility. This is the only method that
   marks differences within lines as well as across lines.  For example, if a line
   of expected output contains digit ``1`` where actual output contains letter
   ``l``, a line is inserted with a caret marking the mismatching column positions.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 650)

Unknown directive type "data".

```
.. data:: REPORT_ONLY_FIRST_FAILURE

   When specified, display the first failing example in each doctest, but suppress
   output for all remaining examples.  This will prevent doctest from reporting
   correct examples that break because of earlier failures; but it might also hide
   incorrect examples that fail independently of the first failure.  When
   :const:`REPORT_ONLY_FIRST_FAILURE` is specified, the remaining examples are
   still run, and still count towards the total number of failures reported; only
   the output is suppressed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 661)

Unknown directive type "data".

```
.. data:: FAIL_FAST

   When specified, exit after the first failing example and don't attempt to run
   the remaining examples. Thus, the number of failures reported will be at most
   1.  This flag may be useful during debugging, since examples after the first
   failure won't even produce debugging output.

   The doctest command line accepts the option ``-f`` as a shorthand for ``-o
   FAIL_FAST``.

   .. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 674)

Unknown directive type "data".

```
.. data:: REPORTING_FLAGS

   A bitmask or'ing together all the reporting flags above.
```

There is also a way to register new option flag names, though this isn't useful unless you intend to extend :mod:`doctest` internals via subclassing:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 679); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 683)

Unknown directive type "function".

```
.. function:: register_optionflag(name)

   Create a new option flag with a given name, and return the new flag's integer
   value.  :func:`register_optionflag` can be used when subclassing
   :class:`OutputChecker` or :class:`DocTestRunner` to create new options that are
   supported by your subclasses.  :func:`register_optionflag` should always be
   called using the following idiom::

      MY_FLAG = register_optionflag('MY_FLAG')
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 694)

Unknown directive type "index".

```
.. index::
   single: # (hash); in doctests
   single: + (plus); in doctests
   single: - (minus); in doctests
```

Directives

Doctest directives may be used to modify the :ref:`option flags <doctest-options>` for an individual example. Doctest directives are special Python comments following an example's source code:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 703); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 707)

Unknown directive type "productionlist".

```
.. productionlist:: doctest
   directive: "#" "doctest:" `directive_options`
   directive_options: `directive_option` ("," `directive_option`)\*
   directive_option: `on_or_off` `directive_option_name`
   on_or_off: "+" \| "-"
   directive_option_name: "DONT_ACCEPT_BLANKLINE" \| "NORMALIZE_WHITESPACE" \| ...
```

Whitespace is not allowed between the `+` or `-` and the directive option name. The directive option name can be any of the option flag names explained above.

An example's doctest directives modify doctest's behavior for that single example. Use `+` to enable the named behavior, or `-` to disable it.

For example, this test passes:

```
>>> print(list(range(20))) # doctest: +NORMALIZE_WHITESPACE
[0,   1,  2,  3,  4,  5,  6,  7,  8,  9,
10,  11, 12, 13, 14, 15, 16, 17, 18, 19]
```

Without the directive it would fail, both because the actual output doesn't have two blanks before the single-digit list elements, and because the actual output is on a single line. This test also passes, and also requires a directive to do so:

```
>>> print(list(range(20))) # doctest: +ELLIPSIS
[0, 1, ..., 18, 19]
```

Multiple directives can be used on a single physical line, separated by commas:

```
>>> print(list(range(20))) # doctest: +ELLIPSIS, +NORMALIZE_WHITESPACE
[0,    1, ...,   18,    19]
```

If multiple directive comments are used for a single example, then they are combined:

```
>>> print(list(range(20))) # doctest: +ELLIPSIS
...                        # doctest: +NORMALIZE_WHITESPACE
[0,    1, ...,   18,    19]
```

As the previous example shows, you can add `...` lines to your example containing only directives. This can be useful when an example is too long for a directive to comfortably fit on the same line:

```
>>> print(list(range(5)) + list(range(10, 20)) + list(range(30, 40)))
... # doctest: +ELLIPSIS
[0, ..., 4, 10, ..., 19, 30, ..., 39]
```

Note that since all options are disabled by default, and directives apply only to the example they appear in, enabling options (via `+` in a directive) is usually the only meaningful choice. However, option flags can also be passed to functions that run doctests, establishing different defaults. In such cases, disabling an option via `-` in a directive can be useful.

Warnings

:mod:`doctest` is serious about requiring exact matches in expected output. If even a single character doesn't match, the test fails. This will probably surprise you a few times, as you learn exactly what Python does and doesn't guarantee about output. For example, when printing a set, Python doesn't guarantee that the element is printed in any particular order, so a test like

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

Unknown interpreted text role "mod".

```
>>> foo()
{"Hermione", "Harry"}
```

is vulnerable! One workaround is to do

```
>>> foo() == {"Hermione", "Harry"}
True
```

instead. Another is to do

```
>>> d = sorted(foo())
>>> d
['Harry', 'Hermione']
```

> **Note**
>
> Before Python 3.6, when printing a dict, Python did not guarantee that the key-value pairs was printed in any particular order.

There are others, but you get the idea.

Another bad idea is to print things that embed an object address, like

```
>>> id(1.0) # certain to fail some of the time
7948648
>>> class C: pass
>>> C()   # the default repr() for instances embeds an address
<__main__.C instance at 0x00AC18F0>
```

The :const:`ELLIPSIS` directive gives a nice approach for the last example:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 803); *backlink*

Unknown interpreted text role "const".

```
>>> C() #doctest: +ELLIPSIS
<__main__.C instance at 0x...>
```

Floating-point numbers are also subject to small output variations across platforms, because Python defers to the platform C library for float formatting, and C libraries vary widely in quality here.

```
>>> 1./7  # risky
0.14285714285714285
>>> print(1./7) # safer
0.142857142857
>>> print(round(1./7, 6)) # much safer
0.142857
```

Numbers of the form `I/2.**J` are safe across all platforms, and I often contrive doctest examples to produce numbers of that form:

```
>>> 3./4  # utterly safe
0.75
```

Simple fractions are also easier for people to understand, and that makes for better documentation.

Basic API

The functions :func:`testmod` and :func:`testfile` provide a simple interface to doctest that should be sufficient for most basic uses. For

a less formal introduction to these two functions, see sections :ref:`doctest-simple-testmod` and :ref:`doctest-simple-testfile`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 834); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 834); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 834); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 834); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 840)

Unknown directive type "function".

```
.. function:: testfile(filename, module_relative=True, name=None, package=None, globs=None, verbose=None, report=True, optionflags=

   All arguments except *filename* are optional, and should be specified in keyword
   form.

   Test examples in the file named *filename*.  Return ``(failure_count,
   test_count)``.

   Optional argument *module_relative* specifies how the filename should be
   interpreted:

   * If *module_relative* is ``True`` (the default), then *filename* specifies an
     OS-independent module-relative path.  By default, this path is relative to the
     calling module's directory; but if the *package* argument is specified, then it
     is relative to that package.  To ensure OS-independence, *filename* should use
     ``/`` characters to separate path segments, and may not be an absolute path
     (i.e., it may not begin with ``/``).

   * If *module_relative* is ``False``, then *filename* specifies an OS-specific
     path.  The path may be absolute or relative; relative paths are resolved with
     respect to the current working directory.

   Optional argument *name* gives the name of the test; by default, or if ``None``,
   ``os.path.basename(filename)`` is used.

   Optional argument *package* is a Python package or the name of a Python package
   whose directory should be used as the base directory for a module-relative
   filename.  If no package is specified, then the calling module's directory is
   used as the base directory for module-relative filenames.  It is an error to
   specify *package* if *module_relative* is ``False``.

   Optional argument *globs* gives a dict to be used as the globals when executing
   examples.  A new shallow copy of this dict is created for the doctest, so its
   examples start with a clean slate. By default, or if ``None``, a new empty dict
   is used.

   Optional argument *extraglobs* gives a dict merged into the globals used to
   execute examples.  This works like :meth:`dict.update`:  if *globs* and
   *extraglobs* have a common key, the associated value in *extraglobs* appears in
   the combined dict.  By default, or if ``None``, no extra globals are used.  This
   is an advanced feature that allows parameterization of doctests.  For example, a
   doctest can be written for a base class, using a generic name for the class,
   then reused to test any number of subclasses by passing an *extraglobs* dict
   mapping the generic name to the subclass to be tested.

   Optional argument *verbose* prints lots of stuff if true, and prints only
   failures if false; by default, or if ``None``, it's true if and only if ``'-v'``
   is in ``sys.argv``.

   Optional argument *report* prints a summary at the end when true, else prints
   nothing at the end.  In verbose mode, the summary is detailed, else the summary
   is very brief (in fact, empty if all tests passed).

   Optional argument *optionflags* (default value 0) takes the
   :ref:`bitwise OR <bitwise>` of option flags.
   See section :ref:`doctest-options`.

   Optional argument *raise_on_error* defaults to false.  If true, an exception is
   raised upon the first failure or unexpected exception in an example.  This
   allows failures to be post-mortem debugged. Default behavior is to continue
   running examples.

   Optional argument *parser* specifies a :class:`DocTestParser` (or subclass) that
   should be used to extract tests from the files.  It defaults to a normal parser
   (i.e., ``DocTestParser()``).

   Optional argument *encoding* specifies an encoding that should be used to
   convert the file to unicode.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 910)

Unknown directive type "function".

```
.. function:: testmod(m=None, name=None, globs=None, verbose=None, report=True, optionflags=0, extraglobs=None, raise_on_error=Fal

   All arguments are optional, and all except for *m* should be specified in
   keyword form.

   Test examples in docstrings in functions and classes reachable from module *m*
   (or module :mod:`__main__` if *m* is not supplied or is ``None``), starting with
   ``m.__doc__``.

   Also test examples reachable from dict ``m.__test__``, if it exists and is not
   ``None``.  ``m.__test__`` maps names (strings) to functions, classes and
   strings; function and class docstrings are searched for examples; strings are
   searched directly, as if they were docstrings.

   Only docstrings attached to objects belonging to module *m* are searched.

   Return ``(failure_count, test_count)``.

   Optional argument *name* gives the name of the module; by default, or if
   ``None``, ``m.__name__`` is used.
```

```
Optional argument *exclude_empty* defaults to false.  If true, objects for which
no doctests are found are excluded from consideration. The default is a backward
compatibility hack, so that code still using :meth:`doctest.master.summarize` in
conjunction with :func:`testmod` continues to get output for objects with no
tests. The *exclude_empty* argument to the newer :class:`DocTestFinder`
constructor defaults to true.

Optional arguments *extraglobs*, *verbose*, *report*, *optionflags*,
*raise_on_error*, and *globs* are the same as for function :func:`testfile`
above, except that *globs* defaults to ``m.__dict__``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 943)

Unknown directive type "function".

```
.. function:: run_docstring_examples(f, globs, verbose=False, name="NoName", compileflags=None, optionflags=0)

   Test examples associated with object *f*; for example, *f* may be a string,
   a module, a function, or a class object.

   A shallow copy of dictionary argument *globs* is used for the execution context.

   Optional argument *name* is used in failure messages, and defaults to
   ``"NoName"``.

   If optional argument *verbose* is true, output is generated even if there are no
   failures.  By default, output is generated only in case of an example failure.

   Optional argument *compileflags* gives the set of flags that should be used by
   the Python compiler when running the examples.  By default, or if ``None``,
   flags are deduced corresponding to the set of future features found in *globs*.

   Optional argument *optionflags* works as for function :func:`testfile` above.
```

Unittest API

As your collection of doctest'ed modules grows, you'll want a way to run all their doctests systematically. :mod:`doctest` provides two functions that can be used to create :mod:`unittest` test suites from modules and text files containing doctests. To integrate with :mod:`unittest` test discovery, include a :func:`load_tests` function in your test module:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 968); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 968); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 968); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 968); *backlink*

Unknown interpreted text role "func".

```
import unittest
import doctest
import my_module_with_doctests

def load_tests(loader, tests, ignore):
    tests.addTests(doctest.DocTestSuite(my_module_with_doctests))
    return tests
```

There are two main functions for creating :class:`unittest.TestSuite` instances from text files and modules with doctests:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 982); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 986)

Unknown directive type "function".

```
.. function:: DocFileSuite(*paths, module_relative=True, package=None, setUp=None, tearDown=None, globs=None, optionflags=0, parse

   Convert doctest tests from one or more text files to a
   :class:`unittest.TestSuite`.

   The returned :class:`unittest.TestSuite` is to be run by the unittest framework
   and runs the interactive examples in each file.  If an example in any file
   fails, then the synthesized unit test fails, and a :exc:`failureException`
   exception is raised showing the name of the file containing the test and a
   (sometimes approximate) line number.

   Pass one or more paths (as strings) to text files to be examined.

   Options may be provided as keyword arguments:

   Optional argument *module_relative* specifies how the filenames in *paths*
   should be interpreted:

   * If *module_relative* is ``True`` (the default), then each filename in
     *paths* specifies an OS-independent module-relative path.  By default, this
     path is relative to the calling module's directory; but if the *package*
     argument is specified, then it is relative to that package.  To ensure
     OS-independence, each filename should use ``/`` characters to separate path
     segments, and may not be an absolute path (i.e., it may not begin with
     ``/``).

   * If *module_relative* is ``False``, then each filename in *paths* specifies
     an OS-specific path.  The path may be absolute or relative; relative paths
     are resolved with respect to the current working directory.

   Optional argument *package* is a Python package or the name of a Python
   package whose directory should be used as the base directory for
   module-relative filenames in *paths*.  If no package is specified, then the
   calling module's directory is used as the base directory for module-relative
```

```
   filenames.  It is an error to specify *package* if *module_relative* is
   ``False``.

   Optional argument *setUp* specifies a set-up function for the test suite.
   This is called before running the tests in each file.  The *setUp* function
   will be passed a :class:`DocTest` object.  The setUp function can access the
   test globals as the *globs* attribute of the test passed.

   Optional argument *tearDown* specifies a tear-down function for the test
   suite.  This is called after running the tests in each file.  The *tearDown*
   function will be passed a :class:`DocTest` object.  The setUp function can
   access the test globals as the *globs* attribute of the test passed.

   Optional argument *globs* is a dictionary containing the initial global
   variables for the tests.  A new copy of this dictionary is created for each
   test.  By default, *globs* is a new empty dictionary.

   Optional argument *optionflags* specifies the default doctest options for the
   tests, created by or-ing together individual option flags.  See section
   :ref:`doctest-options`. See function :func:`set_unittest_reportflags` below
   for a better way to set reporting options.

   Optional argument *parser* specifies a :class:`DocTestParser` (or subclass)
   that should be used to extract tests from the files.  It defaults to a normal
   parser (i.e., ``DocTestParser()``).

   Optional argument *encoding* specifies an encoding that should be used to
   convert the file to unicode.

   The global ``__file__`` is added to the globals provided to doctests loaded
   from a text file using :func:`DocFileSuite`.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1053)

Unknown directive type "function".

```
.. function:: DocTestSuite(module=None, globs=None, extraglobs=None, test_finder=None, setUp=None, tearDown=None, checker=None)

   Convert doctest tests for a module to a :class:`unittest.TestSuite`.

   The returned :class:`unittest.TestSuite` is to be run by the unittest framework
   and runs each doctest in the module.  If any of the doctests fail, then the
   synthesized unit test fails, and a :exc:`failureException` exception is raised
   showing the name of the file containing the test and a (sometimes approximate)
   line number.

   Optional argument *module* provides the module to be tested.  It can be a module
   object or a (possibly dotted) module name.  If not specified, the module calling
   this function is used.

   Optional argument *globs* is a dictionary containing the initial global
   variables for the tests.  A new copy of this dictionary is created for each
   test.  By default, *globs* is a new empty dictionary.

   Optional argument *extraglobs* specifies an extra set of global variables, which
   is merged into *globs*.  By default, no extra globals are used.

   Optional argument *test_finder* is the :class:`DocTestFinder` object (or a
   drop-in replacement) that is used to extract doctests from the module.

   Optional arguments *setUp*, *tearDown*, and *optionflags* are the same as for
   function :func:`DocFileSuite` above.

   This function uses the same search technique as :func:`testmod`.

   .. versionchanged:: 3.5
      :func:`DocTestSuite` returns an empty :class:`unittest.TestSuite` if *module*
      contains no docstrings instead of raising :exc:`ValueError`.
```

Under the covers, :func:`DocTestSuite` creates a :class:`unittest.TestSuite` out of :class:`doctest.DocTestCase` instances, and :class:`DocTestCase` is a subclass of :class:`unittest.TestCase`. :class:`DocTestCase` isn't documented here (it's an internal detail), but studying its code can answer questions about the exact details of :mod:`unittest` integration.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1087); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1087); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1087); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1087); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1087); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1087); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1087); *backlink*

Unknown interpreted text role "mod".

Similarly, :func:`DocFileSuite` creates a :class:`unittest.TestSuite` out of :class:`doctest.DocFileCase` instances, and :class:`DocFileCase` is a subclass of :class:`DocTestCase`.

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

main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1093); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1093); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1093); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1093); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1093); *backlink*

Unknown interpreted text role "class".

So both ways of creating a :class:`unittest.TestSuite` run instances of :class:`DocTestCase`. This is important for a subtle reason: when you run :mod:`doctest` functions yourself, you can control the :mod:`doctest` options in use directly, by passing option flags to :mod:`doctest` functions. However, if you're writing a :mod:`unittest` framework, :mod:`unittest` ultimately controls when and how tests get run. The framework author typically wants to control :mod:`doctest` reporting options (perhaps, e.g., specified by command line options), but there's no way to pass options through :mod:`unittest` to :mod:`doctest` test runners.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1097); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1097); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1097); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1097); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1097); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1097); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1097); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1097); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1097); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1097); *backlink*

Unknown interpreted text role "mod".

For this reason, :mod:`doctest` also supports a notion of :mod:`doctest` reporting flags specific to :mod:`unittest` support, via this function:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1107); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1107); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1107); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) doctest.rst, line 1111)

Unknown directive type "function".

```
.. function:: set_unittest_reportflags(flags)
```

```
Set the :mod:`doctest` reporting flags to use.

Argument *flags* takes the :ref:`bitwise OR <bitwise>` of option flags.  See
section :ref:`doctest-options`.  Only "reporting flags" can be used.

This is a module-global setting, and affects all future doctests run by module
:mod:`unittest`:  the :meth:`runTest` method of :class:`DocTestCase` looks at
the option flags specified for the test case when the :class:`DocTestCase`
instance was constructed.  If no reporting flags were specified (which is the
typical and expected case), :mod:`doctest`'s :mod:`unittest` reporting flags are
:ref:`bitwise ORed <bitwise>` into the option flags, and the option flags
so augmented are passed to the :class:`DocTestRunner` instance created to
run the doctest.  If any reporting flags were specified when the
:class:`DocTestCase` instance was constructed, :mod:`doctest`'s
:mod:`unittest` reporting flags are ignored.

The value of the :mod:`unittest` reporting flags in effect before the function
was called is returned by the function.
```

Advanced API

The basic API is a simple wrapper that's intended to make doctest easy to use. It is fairly flexible, and should meet most users' needs; however, if you require more fine-grained control over testing, or wish to extend doctest's capabilities, then you should use the advanced API.

The advanced API revolves around two container classes, which are used to store the interactive examples extracted from doctest cases:

- :class:`Example`: A single Python :term:`statement`, paired with its expected output.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1146); *backlink*

  Unknown interpreted text role "class".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1146); *backlink*

  Unknown interpreted text role "term".

- :class:`DocTest`: A collection of :class:`Example`s, typically extracted from a single docstring or text file.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1149); *backlink*

  Unknown interpreted text role "class".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1149); *backlink*

  Unknown interpreted text role "class".

Additional processing classes are defined to find, parse, and run, and check doctest examples:

- :class:`DocTestFinder`: Finds all docstrings in a given module, and uses a :class:`DocTestParser` to create a :class:`DocTest` from every docstring that contains interactive examples.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1155); *backlink*

  Unknown interpreted text role "class".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1155); *backlink*

  Unknown interpreted text role "class".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1155); *backlink*

  Unknown interpreted text role "class".

- :class:`DocTestParser`: Creates a :class:`DocTest` object from a string (such as an object's docstring).

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1159); *backlink*

  Unknown interpreted text role "class".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1159); *backlink*

  Unknown interpreted text role "class".

- :class:`DocTestRunner`: Executes the examples in a :class:`DocTest`, and uses an :class:`OutputChecker` to verify their output.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1162); *backlink*

  Unknown interpreted text role "class".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1162); *backlink*

  Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1162); *backlink*

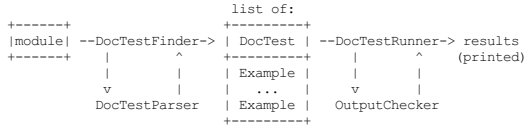
Unknown interpreted text role "class".

- :class:`OutputChecker`: Compares the actual output from a doctest example with the expected output, and decides whether they match.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1165); *backlink*

Unknown interpreted text role "class".

The relationships among these processing classes are summarized in the following diagram:



DocTest Objects

A collection of doctest examples that should be run in a single namespace. The constructor arguments are used to initialize the attributes of the same names.

:class:`DocTest` defines the following attributes. They are initialized by the constructor, and should not be modified directly.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1193); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1197)

Unknown directive type "attribute".

```
.. attribute:: examples

   A list of :class:`Example` objects encoding the individual interactive Python
   examples that should be run by this test.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1203)

Unknown directive type "attribute".

```
.. attribute:: globs

   The namespace (aka globals) that the examples should be run in. This is a
   dictionary mapping names to values.  Any changes to the namespace made by the
   examples (such as binding new variables) will be reflected in :attr:`globs`
   after the test is run.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1211)

Unknown directive type "attribute".

```
.. attribute:: name

   A string name identifying the :class:`DocTest`.  Typically, this is the name
   of the object or file that the test was extracted from.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1217)

Unknown directive type "attribute".

```
.. attribute:: filename

   The name of the file that this :class:`DocTest` was extracted from; or
   ``None`` if the filename is unknown, or if the :class:`DocTest` was not
   extracted from a file.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1224)

Unknown directive type "attribute".

```
.. attribute:: lineno

   The line number within :attr:`filename` where this :class:`DocTest` begins, or
   ``None`` if the line number is unavailable.  This line number is zero-based
   with respect to the beginning of the file.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1231)

Unknown directive type "attribute".

```
.. attribute:: docstring

   The string that the test was extracted from, or ``None`` if the string is
   unavailable, or if the test was not extracted from a string.
```

Example Objects

A single interactive example, consisting of a Python statement and its expected output. The constructor arguments are used to initialize the attributes of the same names.

:class:`Example` defines the following attributes. They are initialized by the constructor, and should not be modified directly.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1250); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1254)

Unknown directive type "attribute".

```
.. attribute:: source

   A string containing the example's source code.  This source code consists of a
   single Python statement, and always ends with a newline; the constructor adds
   a newline when necessary.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1261)

Unknown directive type "attribute".

```
.. attribute:: want

   The expected output from running the example's source code (either from
   stdout, or a traceback in case of exception).  :attr:`want` ends with a
   newline unless no output is expected, in which case it's an empty string.  The
   constructor adds a newline when necessary.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1269)

Unknown directive type "attribute".

```
.. attribute:: exc_msg

   The exception message generated by the example, if the example is expected to
   generate an exception; or ``None`` if it is not expected to generate an
   exception.  This exception message is compared against the return value of
   :func:`traceback.format_exception_only`.  :attr:`exc_msg` ends with a newline
   unless it's ``None``.  The constructor adds a newline if needed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1278)

Unknown directive type "attribute".

```
.. attribute:: lineno

   The line number within the string containing this example where the example
   begins.  This line number is zero-based with respect to the beginning of the
   containing string.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1285)

Unknown directive type "attribute".

```
.. attribute:: indent

   The example's indentation in the containing string, i.e., the number of space
   characters that precede the example's first prompt.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1291)

Unknown directive type "attribute".

```
.. attribute:: options

   A dictionary mapping from option flags to ``True`` or ``False``, which is used
   to override default options for this example.  Any option flags not contained
   in this dictionary are left at their default value (as specified by the
   :class:`DocTestRunner`'s :attr:`optionflags`). By default, no options are set.
```

DocTestFinder objects

A processing class used to extract the :class:`DocTest`s that are relevant to a given object, from its docstring and the docstrings of its contained objects. :class:`DocTest`s can be extracted from modules, classes, functions, methods, staticmethods, classmethods, and properties.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1307); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1307); *backlink*

Unknown interpreted text role "class".

The optional argument *verbose* can be used to display the objects searched by the finder. It defaults to `False` (no output).

The optional argument *parser* specifies the :class:`DocTestParser` object (or a drop-in replacement) that is used to extract doctests from docstrings.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1315); *backlink*

Unknown interpreted text role "class".

If the optional argument *recurse* is false, then :meth:`DocTestFinder.find` will only examine the given object, and not any contained objects.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1318); *backlink*

Unknown interpreted text role "meth".

If the optional argument *exclude_empty* is false, then :meth:`DocTestFinder.find` will include tests for objects with empty docstrings.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1321); *backlink*

Unknown interpreted text role "meth".

:class:`DocTestFinder` defines the following method:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1325); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1328)

Unknown directive type "method".

```
.. method:: find(obj[, name][, module][, globs][, extraglobs])

   Return a list of the :class:`DocTest`\ s that are defined by *obj*'s
   docstring, or by any of its contained objects' docstrings.

   The optional argument *name* specifies the object's name; this name will be
   used to construct names for the returned :class:`DocTest`\ s.  If *name* is
   not specified, then ``obj.__name__`` is used.

   The optional parameter *module* is the module that contains the given object.
   If the module is not specified or is ``None``, then the test finder will attempt
   to automatically determine the correct module.  The object's module is used:

   * As a default namespace, if *globs* is not specified.

   * To prevent the DocTestFinder from extracting DocTests from objects that are
     imported from other modules.  (Contained objects with modules other than
     *module* are ignored.)

   * To find the name of the file containing the object.

   * To help find the line number of the object within its file.

   If *module* is ``False``, no attempt to find the module will be made.  This is
   obscure, of use mostly in testing doctest itself: if *module* is ``False``, or
   is ``None`` but cannot be found automatically, then all objects are considered
   to belong to the (non-existent) module, so all contained objects will
   (recursively) be searched for doctests.

   The globals for each :class:`DocTest` is formed by combining *globs* and
   *extraglobs* (bindings in *extraglobs* override bindings in *globs*).  A new
   shallow copy of the globals dictionary is created for each :class:`DocTest`.
   If *globs* is not specified, then it defaults to the module's *__dict__*, if
   specified, or ``{}`` otherwise.  If *extraglobs* is not specified, then it
   defaults to ``{}``.
```

DocTestParser objects

A processing class used to extract interactive examples from a string, and use them to create a :class:`DocTest` object.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1373); *backlink*

Unknown interpreted text role "class".

:class:`DocTestParser` defines the following methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1377); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1380)

Unknown directive type "method".

```
.. method:: get_doctest(string, globs, name, filename, lineno)

   Extract all doctest examples from the given string, and collect them into a
   :class:`DocTest` object.

   *globs*, *name*, *filename*, and *lineno* are attributes for the new
   :class:`DocTest` object.  See the documentation for :class:`DocTest` for more
   information.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1390)

Unknown directive type "method".

```
.. method:: get_examples(string, name='<string>')

   Extract all doctest examples from the given string, and return them as a list
   of :class:`Example` objects.  Line numbers are 0-based.  The optional argument
   *name* is a name identifying this string, and is only used for error messages.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1397)

Unknown directive type "method".

```
.. method:: parse(string, name='<string>')

   Divide the given string into examples and intervening text, and return them as
   a list of alternating :class:`Example`\ s and strings. Line numbers for the
   :class:`Example`\ s are 0-based.  The optional argument *name* is a name
   identifying this string, and is only used for error messages.
```

DocTestRunner objects

A processing class used to execute and verify the interactive examples in a :class:`DocTest`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1413); *backlink*

Unknown interpreted text role "class".

The comparison between expected outputs and actual outputs is done by an :class:`OutputChecker`. This comparison may be customized with a number of option flags; see section :ref:`doctest-options` for more information. If the option flags are insufficient, then the comparison may also be customized by passing a subclass of :class:`OutputChecker` to the constructor.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1416); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1416); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1416); *backlink*

Unknown interpreted text role "class".

The test runner's display output can be controlled in two ways. First, an output function can be passed to :meth:`TestRunner.run`; this function will be called with strings that should be displayed. It defaults to `sys.stdout.write`. If capturing the output is not sufficient, then the display output can be also customized by subclassing DocTestRunner, and overriding the methods :meth:`report_start`, :meth:`report_success`, :meth:`report_unexpected_exception`, and :meth:`report_failure`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1422); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1422); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1422); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1422); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1422); *backlink*

Unknown interpreted text role "meth".

The optional keyword argument *checker* specifies the :class:`OutputChecker` object (or drop-in replacement) that should be used to compare the expected outputs to the actual outputs of doctest examples.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1430); *backlink*

Unknown interpreted text role "class".

The optional keyword argument *verbose* controls the :class:`DocTestRunner`'s verbosity. If *verbose* is `True`, then information is printed about each example, as it is run. If *verbose* is `False`, then only failures are printed. If *verbose* is unspecified, or `None`, then verbose output is used iff the command-line switch `-v` is used.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1434); *backlink*

Unknown interpreted text role "class".

The optional keyword argument *optionflags* can be used to control how the test runner compares expected output to actual output, and how it displays failures. For more information, see section :ref:`doctest-options`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1440); *backlink*

Unknown interpreted text role "ref".

:class:`DocTestParser` defines the following methods:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1445); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1448)

Unknown directive type "method".

```
.. method:: report_start(out, test, example)

   Report that the test runner is about to process the given example. This method
   is provided to allow subclasses of :class:`DocTestRunner` to customize their
   output; it should not be called directly.

   *example* is the example about to be processed.  *test* is the test
   *containing example*.  *out* is the output function that was passed to
   :meth:`DocTestRunner.run`.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst`, line 1459)

Unknown directive type "method".

```
.. method:: report_success(out, test, example, got)

   Report that the given example ran successfully.  This method is provided to
   allow subclasses of :class:`DocTestRunner` to customize their output; it
   should not be called directly.

   *example* is the example about to be processed.  *got* is the actual output
   from the example.  *test* is the test containing *example*.  *out* is the
   output function that was passed to :meth:`DocTestRunner.run`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1470)

Unknown directive type "method".

```
.. method:: report_failure(out, test, example, got)

   Report that the given example failed.  This method is provided to allow
   subclasses of :class:`DocTestRunner` to customize their output; it should not
   be called directly.

   *example* is the example about to be processed.  *got* is the actual output
   from the example.  *test* is the test containing *example*.  *out* is the
   output function that was passed to :meth:`DocTestRunner.run`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1481)

Unknown directive type "method".

```
.. method:: report_unexpected_exception(out, test, example, exc_info)

   Report that the given example raised an unexpected exception. This method is
   provided to allow subclasses of :class:`DocTestRunner` to customize their
   output; it should not be called directly.

   *example* is the example about to be processed. *exc_info* is a tuple
   containing information about the unexpected exception (as returned by
   :func:`sys.exc_info`). *test* is the test containing *example*.  *out* is the
   output function that was passed to :meth:`DocTestRunner.run`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1493)

Unknown directive type "method".

```
.. method:: run(test, compileflags=None, out=None, clear_globs=True)

   Run the examples in *test* (a :class:`DocTest` object), and display the
   results using the writer function *out*.

   The examples are run in the namespace ``test.globs``.  If *clear_globs* is
   true (the default), then this namespace will be cleared after the test runs,
   to help with garbage collection. If you would like to examine the namespace
   after the test completes, then use *clear_globs=False*.

   *compileflags* gives the set of flags that should be used by the Python
   compiler when running the examples.  If not specified, then it will default to
   the set of future-import flags that apply to *globs*.

   The output of each example is checked using the :class:`DocTestRunner`'s
   output checker, and the results are formatted by the
   :meth:`DocTestRunner.report_\*` methods.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1512)

Unknown directive type "method".

```
.. method:: summarize(verbose=None)

   Print a summary of all the test cases that have been run by this DocTestRunner,
   and return a :term:`named tuple` ``TestResults(failed, attempted)``.

   The optional *verbose* argument controls how detailed the summary is.  If the
   verbosity is not specified, then the :class:`DocTestRunner`'s verbosity is
   used.
```

OutputChecker objects

A class used to check the whether the actual output from a doctest example matches the expected output. :class:`OutputChecker` defines two methods: :meth:`check_output`, which compares a given pair of outputs, and returns `True` if they match; and :meth:`output_difference`, which returns a string describing the differences between two outputs.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1529); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1529); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1529); *backlink*

Unknown interpreted text role "meth".

:class:`OutputChecker` defines the following methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1536); *backlink*

Unknown interpreted text role "class".

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1538)

Unknown directive type "method".

```
.. method:: check_output(want, got, optionflags)

   Return ``True`` iff the actual output from an example (*got*) matches the
   expected output (*want*).  These strings are always considered to match if
   they are identical; but depending on what option flags the test runner is
   using, several non-exact match types are also possible.  See section
   :ref:`doctest-options` for more information about option flags.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1547)

Unknown directive type "method".

```
.. method:: output_difference(example, got, optionflags)

   Return a string describing the differences between the expected output for a
   given example (*example*) and the actual output (*got*).  *optionflags* is the
   set of option flags used to compare *want* and *got*.
```

Debugging

Doctest provides several mechanisms for debugging doctest examples:

- Several functions convert doctests to executable Python programs, which can be run under the Python debugger, :mod:`pdb`.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1561); *backlink*

  Unknown interpreted text role "mod".

- The :class:`DebugRunner` class is a subclass of :class:`DocTestRunner` that raises an exception for the first failing example, containing information about that example. This information can be used to perform post-mortem debugging on the example.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1564); *backlink*

  Unknown interpreted text role "class".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1564); *backlink*

  Unknown interpreted text role "class".

- The :mod:`unittest` cases generated by :func:`DocTestSuite` support the :meth:`debug` method defined by :class:`unittest.TestCase`.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1569); *backlink*

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1569); *backlink*

  Unknown interpreted text role "func".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1569); *backlink*

  Unknown interpreted text role "meth".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1569); *backlink*

  Unknown interpreted text role "class".

- You can add a call to :func:`pdb.set_trace` in a doctest example, and you'll drop into the Python debugger when that line is executed. Then you can inspect current values of variables, and so on. For example, suppose :file:`a.py` contains just this module docstring:

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1572); *backlink*

  Unknown interpreted text role "func".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1572); *backlink*

  Unknown interpreted text role "file".

  ```
  """
  >>> def f(x):
  ...     g(x*2)
  >>> def g(x):
  ...     print(x+3)
  ...     import pdb; pdb.set_trace()
  >>> f(3)
  9
  """
  ```

  Then an interactive Python session may look like this:

  ```
  >>> import a, doctest
  >>> doctest.testmod(a)
  --Return--
  > <doctest a[1]>(3)g()->None
  ```

```
-> import pdb; pdb.set_trace()
(Pdb) list
  1     def g(x):
  2         print(x+3)
  3  ->     import pdb; pdb.set_trace()
[EOF]
(Pdb) p x
6
(Pdb) step
--Return--
> <doctest a[0]>(2)f()->None
-> g(x*2)
(Pdb) list
  1     def f(x):
  2  ->     g(x*2)
[EOF]
(Pdb) p x
3
(Pdb) step
--Return--
> <doctest a[2]>(1)?()->None
-> f(3)
(Pdb) cont
(0, 3)
>>>
```

Functions that convert doctests to Python code, and possibly run the synthesized code under the debugger:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst, line 1624)

Unknown directive type "function".

```
.. function:: script_from_examples(s)

   Convert text with examples to a script.

   Argument *s* is a string containing doctest examples.  The string is converted
   to a Python script, where doctest examples in *s* are converted to regular code,
   and everything else is converted to Python comments.  The generated script is
   returned as a string. For example, ::

      import doctest
      print(doctest.script_from_examples(r"""
          Set x and y to 1 and 2.
          >>> x, y = 1, 2

          Print their sum:
          >>> print(x+y)
          3
      """))

   displays::

      # Set x and y to 1 and 2.
      x, y = 1, 2
      #
      # Print their sum:
      print(x+y)
      # Expected:
      ## 3

   This function is used internally by other functions (see below), but can also be
   useful when you want to transform an interactive Python session into a Python
   script.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst, line 1658)

Unknown directive type "function".

```
.. function:: testsource(module, name)

   Convert the doctest for an object to a script.

   Argument *module* is a module object, or dotted name of a module, containing the
   object whose doctests are of interest.  Argument *name* is the name (within the
   module) of the object with the doctests of interest.  The result is a string,
   containing the object's docstring converted to a Python script, as described for
   :func:`script_from_examples` above.  For example, if module :file:`a.py`
   contains a top-level function :func:`f`, then ::

      import a, doctest
      print(doctest.testsource(a, "a.f"))

   prints a script version of function :func:`f`'s docstring, with doctests
   converted to code, and the rest placed in comments.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst, line 1676)

Unknown directive type "function".

```
.. function:: debug(module, name, pm=False)

   Debug the doctests for an object.

   The *module* and *name* arguments are the same as for function
   :func:`testsource` above.  The synthesized Python script for the named object's
   docstring is written to a temporary file, and then that file is run under the
   control of the Python debugger, :mod:`pdb`.

   A shallow copy of ``module.__dict__`` is used for both local and global
   execution context.

   Optional argument *pm* controls whether post-mortem debugging is used.  If *pm*
   has a true value, the script file is run directly, and the debugger gets
   involved only if the script terminates via raising an unhandled exception.  If
   it does, then post-mortem debugging is invoked, via :func:`pdb.post_mortem`,
   passing the traceback object from the unhandled exception.  If *pm* is not
   specified, or is false, the script is run under the debugger from the start, via
   passing an appropriate :func:`exec` call to :func:`pdb.run`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)doctest.rst, line 1697)

Unknown directive type "function".

```
.. function:: debug_src(src, pm=False, globs=None)

   Debug the doctests in a string.
```

```
This is like function :func:`debug` above, except that a string containing
doctest examples is specified directly, via the *src* argument.

Optional argument *pm* has the same meaning as in function :func:`debug` above.

Optional argument *globs* gives a dictionary to use as both local and global
execution context.  If not specified, or ``None``, an empty dictionary is used.
If specified, a shallow copy of the dictionary is used.
```

The :class:`DebugRunner` class, and the special exceptions it may raise, are of most interest to testing framework authors, and will only be sketched here. See the source code, and especially :class:`DebugRunner`'s docstring (which is a doctest!) for more details:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1711); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1711); *backlink*

Unknown interpreted text role "class".

A subclass of :class:`DocTestRunner` that raises an exception as soon as a failure is encountered. If an unexpected exception occurs, an :exc:`UnexpectedException` exception is raised, containing the test, the example, and the original exception. If the output doesn't match, then a :exc:`DocTestFailure` exception is raised, containing the test, the example, and the actual output.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1719); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1719); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1719); *backlink*

Unknown interpreted text role "exc".

For information about the constructor parameters and methods, see the documentation for :class:`DocTestRunner` in section :ref:`doctest-advanced-api`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1726); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1726); *backlink*

Unknown interpreted text role "ref".

There are two exceptions that may be raised by :class:`DebugRunner` instances:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1729); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1732)

Unknown directive type "exception".

```
.. exception:: DocTestFailure(test, example, got)

   An exception raised by :class:`DocTestRunner` to signal that a doctest example's
   actual output did not match its expected output. The constructor arguments are
   used to initialize the attributes of the same names.
```

:exc:`DocTestFailure` defines the following attributes:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1738); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1741)

Unknown directive type "attribute".

```
.. attribute:: DocTestFailure.test

   The :class:`DocTest` object that was being run when the example failed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1746)

Unknown directive type "attribute".

```
.. attribute:: DocTestFailure.example

   The :class:`Example` that failed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1751)

Unknown directive type "attribute".

```
.. attribute:: DocTestFailure.got
```

```
The example's actual output.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1756)

Unknown directive type "exception".

```
.. exception:: UnexpectedException(test, example, exc_info)

   An exception raised by :class:`DocTestRunner` to signal that a doctest
   example raised an unexpected exception.  The constructor arguments are used
   to initialize the attributes of the same names.
```

:exc:`UnexpectedException` defines the following attributes:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1762); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1765)

Unknown directive type "attribute".

```
.. attribute:: UnexpectedException.test

   The :class:`DocTest` object that was being run when the example failed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1770)

Unknown directive type "attribute".

```
.. attribute:: UnexpectedException.example

   The :class:`Example` that failed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1775)

Unknown directive type "attribute".

```
.. attribute:: UnexpectedException.exc_info

   A tuple containing information about the unexpected exception, as returned by
   :func:`sys.exc_info`.
```

Soapbox

As mentioned in the introduction, :mod:`doctest` has grown to have three primary uses:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1786); *backlink*

Unknown interpreted text role "mod".

1. Checking examples in docstrings.
2. Regression testing.
3. Executable documentation / literate testing.

These uses have different requirements, and it is important to distinguish them. In particular, filling your docstrings with obscure test cases makes for bad documentation.

When writing a docstring, choose docstring examples with care. There's an art to this that needs to be learned---it may not be natural at first. Examples should add genuine value to the documentation. A good example can often be worth many words. If done with care, the examples will be invaluable for your users, and will pay back the time it takes to collect them many times over as the years go by and things change. I'm still amazed at how often one of my :mod:`doctest` examples stops working after a "harmless" change.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1799); *backlink*

Unknown interpreted text role "mod".

Doctest also makes an excellent tool for regression testing, especially if you don't skimp on explanatory text. By interleaving prose and examples, it becomes much easier to keep track of what's actually being tested, and why. When a test fails, good prose can make it much easier to figure out what the problem is, and how it should be fixed. It's true that you could write extensive comments in code-based testing, but few programmers do. Many have found that using doctest approaches instead leads to much clearer tests. Perhaps this is simply because doctest makes writing prose a little easier than writing code, while writing comments in code is a little harder. I think it goes deeper than just that: the natural attitude when writing a doctest-based test is that you want to explain the fine points of your software, and illustrate them with examples. This in turn naturally leads to test files that start with the simplest features, and logically progress to complications and edge cases. A coherent narrative is the result, instead of a collection of isolated functions that test isolated bits of functionality seemingly at random. It's a different attitude, and produces different results, blurring the distinction between testing and explaining.

Regression testing is best confined to dedicated objects or files. There are several options for organizing tests:

- Write text files containing test cases as interactive examples, and test the files using :func:`testfile` or :func:`DocFileSuite`. This is recommended, although is easiest to do for new projects, designed from the start to use doctest.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1828); *backlink*

  Unknown interpreted text role "func".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)doctest.rst, line 1828); *backlink*

  Unknown interpreted text role "func".

- Define functions named `_regrtest_topic` that consist of single docstrings, containing test cases for the named topics. These functions can be included in the same file as the module, or separated out into a separate test file.

- Define a `__test__` dictionary mapping from regression test topics to docstrings containing test cases.

When you have placed your tests in a module, the module can itself be the test runner. When a test fails, you can arrange for your test runner to re-run only the failing doctest while you debug the problem. Here is a minimal example of such a test runner:

```
if __name__ == '__main__':
    import doctest
    flags = doctest.REPORT_NDIFF|doctest.FAIL_FAST
    if len(sys.argv) > 1:
        name = sys.argv[1]
        if name in globals():
            obj = globals()[name]
        else:
            obj = __test__[name]
        doctest.run_docstring_examples(obj, globals(), name=name,
                                       optionflags=flags)
    else:
        fail, total = doctest.testmod(optionflags=flags)
        print("{} failures out of {} tests".format(fail, total))
```

Footnotes

[1] Examples containing both expected output and an exception are not supported. Trying to guess where one ends and the other begins is too error-prone, and that also makes for a confusing test.