:mod:`unittest` --- Unit testing framework

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

Unknown directive type "module".

```
.. module:: unittest
   :synopsis: Unit testing framework for Python.
```

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

Unknown directive type "moduleauthor".

```
.. moduleauthor:: Steve Purcell <stephen_purcell@yahoo.com>
```

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

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Steve Purcell <stephen_purcell@yahoo.com>
```

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

Unknown directive type "sectionauthor".

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

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

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Raymond Hettinger <python@rcn.com>
```

Source code: :source:`Lib/unittest/__init__.py`

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

Unknown interpreted text role "source".

(If you are already familiar with the basic concepts of testing, you might want to skip to :ref:`the list of assert methods <assert-methods>`.)

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

Unknown interpreted text role "ref".

The :mod:`unittest` unit testing framework was originally inspired by JUnit and has a similar flavor as major unit testing frameworks in other languages. It supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 19); *backlink*

Unknown interpreted text role "mod".

To achieve this, :mod:`unittest` supports some important concepts in an object-oriented way:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 25); *backlink*

Unknown interpreted text role "mod".

test fixture

> A :dfn:`test fixture` represents the preparation needed to perform one or more tests, and any associated cleanup actions. This may involve, for example, creating temporary or proxy databases, directories, or starting a server process.
>
> System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 29); *backlink*

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

Unknown interpreted text role "dfn".

test case

A :dfn:`test case` is the individual unit of testing. It checks for a specific response to a particular set of inputs. :mod:`unittest` provides a base class, :class:`TestCase`, which may be used to create new test cases.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 35); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 35); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 35); *backlink*

Unknown interpreted text role "class".

test suite

A :dfn:`test suite` is a collection of test cases, test suites, or both. It is used to aggregate tests that should be executed together.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 40); *backlink*

Unknown interpreted text role "dfn".

test runner

A :dfn:`test runner` is a component which orchestrates the execution of tests and provides the outcome to the user. The runner may use a graphical interface, a textual interface, or return a special value to indicate the results of executing the tests.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 44); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 50)

Unknown directive type "seealso".

```
.. seealso::

   Module :mod:`doctest`
      Another test-support module with a very different flavor.

   `Simple Smalltalk Testing: With Patterns <https://web.archive.org/web/20150315073817/http://www.xprogramming.co
      Kent Beck's original paper on testing frameworks using the pattern shared
      by :mod:`unittest`.

   `pytest <https://docs.pytest.org/>`_
      Third-party unittest framework with a lighter-weight syntax for writing
      tests.  For example, ``assert func(10) == 42``.

   `The Python Testing Tools Taxonomy <https://wiki.python.org/moin/PythonTestingToolsTaxonomy>`_
      An extensive list of Python testing tools including functional testing
      frameworks and mock object libraries.

   `Testing in Python Mailing List <http://lists.idyll.org/listinfo/testing-in-python>`_
      A special-interest-group for discussion of testing, and testing tools,
      in Python.

   The script :file:`Tools/unittestgui/unittestgui.py` in the Python source distribution is
   a GUI tool for test discovery and execution.  This is intended largely for ease of use
   for those new to unit testing.  For production environments it is
   recommended that tests be driven by a continuous integration system such as
   `Buildbot <https://buildbot.net/>`_, `Jenkins <https://jenkins.io/>`_,
   `GitHub Actions <https://github.com/features/actions>`_, or
   `AppVeyor <https://www.appveyor.com/>`_.
```

Basic example

The :mod:`unittest` module provides a rich set of tools for constructing and running tests. This section demonstrates that a small subset of the tools suffice to meet the needs of most users.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 85); *backlink*

Unknown interpreted text role "mod".

Here is a short script to test three string methods:

```
import unittest

class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        # check that s.split fails when the separator is not a string
        with self.assertRaises(TypeError):
            s.split(2)

if __name__ == '__main__':
    unittest.main()
```

A testcase is created by subclassing :class:`unittest.TestCase`. The three individual tests are defined with methods whose names start with the letters `test`. This naming convention informs the test runner about which methods represent tests.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 113); *backlink*
>
> Unknown interpreted text role "class".

The crux of each test is a call to :meth:`~TestCase.assertEqual` to check for an expected result; :meth:`~TestCase.assertTrue` or :meth:`~TestCase.assertFalse` to verify a condition; or :meth:`~TestCase.assertRaises` to verify that a specific exception gets raised. These methods are used instead of the :keyword:`assert` statement so the test runner can accumulate all test results and produce a report.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 118); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 118); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 118); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 118); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 118); *backlink*
>
> Unknown interpreted text role "keyword".

The :meth:`~TestCase.setUp` and :meth:`~TestCase.tearDown` methods allow you to define instructions that will be executed before and after each test method. They are covered in more detail in the section :ref:`organizing-tests`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 125); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 125); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 125); *backlink*
>
> Unknown interpreted text role "ref".

The final block shows a simple way to run the tests. :func:`unittest.main` provides a command-line interface to the test script. When run from the command line, the above script produces an output that looks like this:

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 129); *backlink*
>
> Unknown interpreted text role "func".

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s
```

```
OK
```

Passing the -v option to your test script will instruct :func:`unittest.main` to enable a higher level of verbosity, and produce the following output:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 139); *backlink***
>
> Unknown interpreted text role "func".

```
test_isupper (__main__.TestStringMethods.test_isupper) ... ok
test_split (__main__.TestStringMethods.test_split) ... ok
test_upper (__main__.TestStringMethods.test_upper) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```

The above examples show the most commonly used :mod:`unittest` features which are sufficient to meet many everyday testing needs. The remainder of the documentation explores the full feature set from first principles.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 151); *backlink***
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 155)**
>
> Unknown directive type "versionchanged".
>
> ```
> .. versionchanged:: 3.11
>    The behavior of returning a value from a test method (other than the default
>    ``None`` value), is now deprecated.
> ```

Command-Line Interface

The unittest module can be used from the command line to run tests from modules, classes or even individual test methods:

```
python -m unittest test_module1 test_module2
python -m unittest test_module.TestClass
python -m unittest test_module.TestClass.test_method
```

You can pass in a list with any combination of module names, and fully qualified class or method names.

Test modules can be specified by file path as well:

```
python -m unittest tests/test_something.py
```

This allows you to use the shell filename completion to specify the test module. The file specified must still be importable as a module. The path is converted to a module name by removing the '.py' and converting path separators into '.'. If you want to execute a test file that isn't importable as a module you should execute the file directly instead.

You can run tests with more detail (higher verbosity) by passing in the -v flag:

```
python -m unittest -v test_module
```

When executed without arguments :ref:`unittest-test-discovery` is started:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 189); *backlink***
>
> Unknown interpreted text role "ref".

```
python -m unittest
```

For a list of all the command-line options:

```
python -m unittest -h
```

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 197)**
>
> Unknown directive type "versionchanged".
>
> ```
> .. versionchanged:: 3.2
>    In earlier versions it was only possible to run individual test methods and
>    not modules or classes.
> ```

Command-line options

:program:`unittest` supports these command-line options:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 205); *backlink***
>
> Unknown interpreted text role "program".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 207)**

Unknown directive type "program".

```
.. program:: unittest
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 209)

Unknown directive type "cmdoption".

```
.. cmdoption:: -b, --buffer

   The standard output and standard error streams are buffered during the test
   run. Output during a passing test is discarded. Output is echoed normally
   on test fail or error and is added to the failure messages.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 215)

Unknown directive type "cmdoption".

```
.. cmdoption:: -c, --catch

   :kbd:`Control-C` during the test run waits for the current test to end and then
   reports all the results so far. A second :kbd:`Control-C` raises the normal
   :exc:`KeyboardInterrupt` exception.

   See `Signal Handling`_ for the functions that provide this functionality.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 223)

Unknown directive type "cmdoption".

```
.. cmdoption:: -f, --failfast

   Stop the test run on the first error or failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 227)

Unknown directive type "cmdoption".

```
.. cmdoption:: -k

   Only run test methods and classes that match the pattern or substring.
   This option may be used multiple times, in which case all test cases that
   match any of the given patterns are included.

   Patterns that contain a wildcard character (``*``) are matched against the
   test name using :meth:`fnmatch.fnmatchcase`; otherwise simple case-sensitive
   substring matching is used.

   Patterns are matched against the fully qualified test method name as
   imported by the test loader.

   For example, ``-k foo`` matches ``foo_tests.SomeTest.test_something``,
   ``bar_tests.SomeTest.test_foo``, but not ``bar_tests.FooTest.test_something``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 243)

Unknown directive type "cmdoption".

```
.. cmdoption:: --locals

   Show local variables in tracebacks.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 247)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
   The command-line options ``-b``, ``-c`` and ``-f`` were added.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 250)

Unknown directive type "versionadded".

```
.. versionadded:: 3.5
   The command-line option ``--locals``.
```

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

Unknown directive type "versionadded".

```
.. versionadded:: 3.7
   The command-line option ``-k``.
```

The command line can also be used for test discovery, for running all of the tests in a project or just a subset.

Test Discovery

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 265)

Unknown directive type "versionadded".

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

Unittest supports simple test discovery. In order to be compatible with test discovery, all of the test files must be :ref:`modules <tut-modules>` or :ref:`packages <tut-packages>` importable from the top-level directory of the project (this means that their filenames must be valid :ref:`identifiers <identifiers>`).

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

Unknown interpreted text role "ref".

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

Unknown interpreted text role "ref".

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

Unknown interpreted text role "ref".

Test discovery is implemented in :meth:`TestLoader.discover`, but can also be used from the command line. The basic command-line usage is:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 273); *backlink*

Unknown interpreted text role "meth".

```
cd project_directory
python -m unittest discover
```

Note

As a shortcut, `python -m unittest` is the equivalent of `python -m unittest discover`. If you want to pass arguments to test discovery the `discover` sub-command must be used explicitly.

The `discover` sub-command has the following options:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 287)

Unknown directive type "program".

```
.. program:: unittest discover
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 289)

Unknown directive type "cmdoption".

```
.. cmdoption:: -v, --verbose

   Verbose output
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 293)

Unknown directive type "cmdoption".

```
.. cmdoption:: -s, --start-directory directory

   Directory to start discovery (``.`` default)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 297)

Unknown directive type "cmdoption".

```
.. cmdoption:: -p, --pattern pattern

   Pattern to match test files (``test*.py`` default)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 301)

Unknown directive type "cmdoption".

```
.. cmdoption:: -t, --top-level-directory directory

   Top level directory of project (defaults to start directory)
```

The :option:`-s`, :option:`-p`, and :option:`-t` options can be passed in as positional arguments in that order. The following two command lines are equivalent:

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

Unknown interpreted text role "option".

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

Unknown interpreted text role "option".

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

Unknown interpreted text role "option".

```
python -m unittest discover -s project_directory -p "*_test.py"
python -m unittest discover project_directory "*_test.py"
```

As well as being a path it is possible to pass a package name, for example `myproject.subpackage.test`, as the start directory. The package name you supply will then be imported and its location on the filesystem will be used as the start directory.

Caution!

Test discovery loads tests by importing them. Once test discovery has found all the test files from the start directory you specify it turns the paths into package names to import. For example :file:`foo/bar/baz.py` will be imported as `foo.bar.baz`.

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

Unknown interpreted text role "file".

If you have a package installed globally and attempt test discovery on a different copy of the package then the import *could* happen from the wrong place. If this happens test discovery will warn you and exit.

If you supply the start directory as a package name rather than a path to a directory then discover assumes that whichever location it imports from is the location you intended, so you will not get the warning.

Test modules and packages can customize test loading and discovery by through the load_tests protocol.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 336)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.4
   Test discovery supports :term:`namespace packages <namespace package>`
   for the start directory. Note that you need to specify the top level
   directory too (e.g.
   ``python -m unittest discover -s root/namespace -t root``).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 342)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.11
   Python 3.11 dropped the :term:`namespace packages <namespace package>`
   support. It has been broken since Python 3.7. Start directory and
   subdirectories containing tests must be regular package that have
   ``__init__.py`` file.

   Directories containing start directory still can be a namespace package.
   In this case, you need to specify start directory as dotted package name,
   and target directory explicitly. For example::

      # proj/  <-- current directory
      #   namespace/
      #     mypkg/
      #       __init__.py
      #       test_mypkg.py

      python -m unittest discover -s namespace.mypkg -t .
```

Organizing test code

The basic building blocks of unit testing are :dfn:`test cases` --- single scenarios that must be set up and checked for correctness. In :mod:`unittest`, test cases are represented by :class:`unittest.TestCase` instances. To make your own test cases you must write subclasses of :class:`TestCase` or use :class:`FunctionTestCase`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 366); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 366); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 366); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 366); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 366); *backlink*

Unknown interpreted text role "class".

The testing code of a :class:`TestCase` instance should be entirely self contained, such that it can be run either in isolation or in arbitrary combination with any number of other test cases.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 372); *backlink*

Unknown interpreted text role "class".

The simplest :class:`TestCase` subclass will simply implement a test method (i.e. a method whose name starts with `test`) in order to perform specific testing code:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 376); *backlink*

Unknown interpreted text role "class".

```
import unittest

class DefaultWidgetSizeTestCase(unittest.TestCase):
    def test_default_widget_size(self):
        widget = Widget('The widget')
        self.assertEqual(widget.size(), (50, 50))
```

Note that in order to test something, we use one of the :meth:`assert\*` methods provided by the :class:`TestCase` base class. If the test fails, an exception will be raised with an explanatory message, and :mod:`unittest` will identify the test case as a :dfn:`failure`. Any other exceptions will be treated as :dfn:`errors`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 387); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 387); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 387); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 387); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 387); *backlink*

Unknown interpreted text role "dfn".

Tests can be numerous, and their set-up can be repetitive. Luckily, we can factor out set-up code by implementing a method called :meth:`~TestCase.setUp`, which the testing framework will automatically call for every single test we run:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 393); *backlink*

Unknown interpreted text role "meth".

```
import unittest
```

```
class WidgetTestCase(unittest.TestCase):
    def setUp(self):
        self.widget = Widget('The widget')

    def test_default_widget_size(self):
        self.assertEqual(self.widget.size(), (50,50),
                         'incorrect default size')

    def test_widget_resize(self):
        self.widget.resize(100,150)
        self.assertEqual(self.widget.size(), (100,150),
                         'wrong size after resize')
```

> **Note**
>
> The order in which the various tests will be run is determined by sorting the test method names with respect to the built-in ordering for strings.

If the :meth:`~TestCase.setUp` method raises an exception while the test is running, the framework will consider the test to have suffered an error, and the test method will not be executed.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 418); *backlink*

Unknown interpreted text role "meth".

Similarly, we can provide a :meth:`~TestCase.tearDown` method that tidies up after the test method has been run:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 422); *backlink*

Unknown interpreted text role "meth".

```
import unittest

class WidgetTestCase(unittest.TestCase):
    def setUp(self):
        self.widget = Widget('The widget')

    def tearDown(self):
        self.widget.dispose()
```

If :meth:`~TestCase.setUp` succeeded, :meth:`~TestCase.tearDown` will be run whether the test method succeeded or not.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 434); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 434); *backlink*

Unknown interpreted text role "meth".

Such a working environment for the testing code is called a :dfn:`test fixture`. A new TestCase instance is created as a unique test fixture used to execute each individual test method. Thus :meth:`~TestCase.setUp`, :meth:`~TestCase.tearDown`, and :meth:`~TestCase.__init__` will be called once per test.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 437); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 437); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 437); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 437); *backlink*

Unknown interpreted text role "meth".

It is recommended that you use TestCase implementations to group tests together according to the features they test. :mod:`unittest` provides a mechanism for this: the :dfn:`test suite`, represented by :mod:`unittest`'s :class:`TestSuite` class. In most cases, calling :func:`unittest.main` will do the right thing and collect all the module's test cases for you and execute them.

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

Unknown interpreted text role "mod".

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]unittest.rst, line 443); *backlink*

Unknown interpreted text role "dfn".

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

Unknown interpreted text role "mod".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "func".

However, should you want to customize the building of your test suite, you can do it yourself:

```
def suite():
    suite = unittest.TestSuite()
    suite.addTest(WidgetTestCase('test_default_widget_size'))
    suite.addTest(WidgetTestCase('test_widget_resize'))
    return suite

if __name__ == '__main__':
    runner = unittest.TextTestRunner()
    runner.run(suite())
```

You can place the definitions of test cases and test suites in the same modules as the code they are to test (such as :file:`widget.py`), but there are several advantages to placing the test code in a separate module, such as :file:`test_widget.py`:

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

Unknown interpreted text role "file".

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

Unknown interpreted text role "file".

- The test module can be run standalone from the command line.
- The test code can more easily be separated from shipped code.
- There is less temptation to change test code to fit the code it tests without a good reason.
- Test code should be modified much less frequently than the code it tests.
- Tested code can be refactored more easily.
- Tests for modules written in C must be in separate modules anyway, so why not be consistent?
- If the testing strategy changes, there is no need to change the source code.

Re-using old test code

Some users will find that they have existing test code that they would like to run from :mod:`unittest`, without converting every old test function to a :class:`TestCase` subclass.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 490); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 490); *backlink*

Unknown interpreted text role "class".

For this reason, :mod:`unittest` provides a :class:`FunctionTestCase` class. This subclass of :class:`TestCase` can be used to wrap an existing test function. Set-up and tear-down functions can also be provided.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 494); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 494); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 494); *backlink*

Unknown interpreted text role "class".

Given the following test function:

```
def testSomething():
```

```
        something = makeSomething()
        assert something.name is not None
        # ...
```

one can create an equivalent test case instance as follows, with optional set-up and tear-down methods:

```
testcase = unittest.FunctionTestCase(testSomething,
                                     setUp=makeSomethingDB,
                                     tearDown=deleteSomethingDB)
```

Note

Even though :class:`FunctionTestCase` can be used to quickly convert an existing test base over to a :mod:`unittest`-based system, this approach is not recommended. Taking the time to set up proper :class:`TestCase` subclasses will make future test refactorings infinitely easier.

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "mod".

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

Unknown interpreted text role "class".

In some cases, the existing tests may have been written using the :mod:`doctest` module. If so, :mod:`doctest` provides a :class:`DocTestSuite` class that can automatically build :class:`unittest.TestSuite` instances from the existing :mod:`doctest`-based tests.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 519); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 519); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 519); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 519); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 519); *backlink*

Unknown interpreted text role "mod".

Skipping tests and expected failures

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 530)

Unknown directive type "versionadded".

```
.. versionadded:: 3.1
```

Unittest supports skipping individual test methods and even whole classes of tests. In addition, it supports marking a test as an "expected failure," a test that is broken and will fail, but shouldn't be counted as a failure on a :class:`TestResult`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 532); *backlink*

Unknown interpreted text role "class".

Skipping a test is simply a matter of using the :func:`skip` :term:`decorator` or one of its conditional variants, calling :meth:`TestCase.skipTest` within a :meth:`~TestCase.setUp` or test method, or raising :exc:`SkipTest` directly.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 537); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 537); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 537); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 537); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 537); *backlink*

Unknown interpreted text role "exc".

Basic skipping looks like this:

```
class MyTestCase(unittest.TestCase):

    @unittest.skip("demonstrating skipping")
    def test_nothing(self):
        self.fail("shouldn't happen")

    @unittest.skipIf(mylib.__version__ < (1, 3),
                     "not supported in this library version")
    def test_format(self):
        # Tests that work for only a certain version of the library.
        pass

    @unittest.skipUnless(sys.platform.startswith("win"), "requires Windows")
    def test_windows_support(self):
        # windows specific testing code
        pass

    def test_maybe_skipped(self):
        if not external_resource_available():
            self.skipTest("external resource not available")
        # test code that depends on the external resource
        pass
```

This is the output of running the example above in verbose mode:

```
test_format (__main__.MyTestCase.test_format) ... skipped 'not supported in this library version'
test_nothing (__main__.MyTestCase.test_nothing) ... skipped 'demonstrating skipping'
test_maybe_skipped (__main__.MyTestCase.test_maybe_skipped) ... skipped 'external resource not available'
test_windows_support (__main__.MyTestCase.test_windows_support) ... skipped 'requires Windows'

----------------------------------------------------------------------
Ran 4 tests in 0.005s

OK (skipped=4)
```

Classes can be skipped just like methods:

```
@unittest.skip("showing class skipping")
class MySkippedTestCase(unittest.TestCase):
    def test_not_run(self):
        pass
```

:meth:`TestCase.setUp` can also skip the test. This is useful when a resource that needs to be set up is not available.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 585); *backlink*

Unknown interpreted text role "meth".

Expected failures use the :func:`expectedFailure` decorator.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 588); *backlink*

Unknown interpreted text role "func".

```
class ExpectedFailureTestCase(unittest.TestCase):
    @unittest.expectedFailure
    def test_fail(self):
        self.assertEqual(1, 0, "broken")
```

It's easy to roll your own skipping decorators by making a decorator that calls :func:`skip` on the test when it wants it to be skipped. This decorator skips the test unless the passed object has a certain attribute:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 595); *backlink*

Unknown interpreted text role "func".

```
def skipUnlessHasattr(obj, attr):
    if hasattr(obj, attr):
```

```
    return lambda func: func
return unittest.skip("{!r} doesn't have {!r}".format(obj, attr))
```

The following decorators and exception implement test skipping and expected failures:

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 606)
>
> Unknown directive type "decorator".
>
> ```
> .. decorator:: skip(reason)
>
>    Unconditionally skip the decorated test.  *reason* should describe why the
>    test is being skipped.
> ```

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 611)
>
> Unknown directive type "decorator".
>
> ```
> .. decorator:: skipIf(condition, reason)
>
>    Skip the decorated test if *condition* is true.
> ```

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 615)
>
> Unknown directive type "decorator".
>
> ```
> .. decorator:: skipUnless(condition, reason)
>
>    Skip the decorated test unless *condition* is true.
> ```

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 619)
>
> Unknown directive type "decorator".
>
> ```
> .. decorator:: expectedFailure
>
>    Mark the test as an expected failure or error.  If the test fails or errors
>    in the test function itself (rather than in one of the :dfn:`test fixture`
>    methods) then it will be considered a success.  If the test passes, it will
>    be considered a failure.
> ```

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 626)
>
> Unknown directive type "exception".
>
> ```
> .. exception:: SkipTest(reason)
>
>    This exception is raised to skip a test.
>
>    Usually you can use :meth:`TestCase.skipTest` or one of the skipping
>    decorators instead of raising this directly.
> ```

Skipped tests will not have :meth:`~TestCase.setUp` or :meth:`~TestCase.tearDown` run around them. Skipped classes will not have :meth:`~TestCase.setUpClass` or :meth:`~TestCase.tearDownClass` run. Skipped modules will not have :func:`setUpModule` or :func:`tearDownModule` run.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 633); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 633); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 633); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 633); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst`, line 633); *backlink*
>
> Unknown interpreted text role "func".

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

Unknown interpreted text role "func".

Distinguishing test iterations using subtests

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 643)

Unknown directive type "versionadded".

```
.. versionadded:: 3.4
```

When there are very small differences among your tests, for instance some parameters, unittest allows you to distinguish them inside the body of a test method using the :meth:`~TestCase.subTest` context manager.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 645); *backlink*

Unknown interpreted text role "meth".

For example, the following test:

```
class NumbersTest(unittest.TestCase):

    def test_even(self):
        """
        Test that numbers between 0 and 5 are all even.
        """
        for i in range(0, 6):
            with self.subTest(i=i):
                self.assertEqual(i % 2, 0)
```

will produce the following output:

```
======================================================================
FAIL: test_even (__main__.NumbersTest.test_even) (i=1)
Test that numbers between 0 and 5 are all even.
----------------------------------------------------------------------
Traceback (most recent call last):
  File "subtests.py", line 11, in test_even
    self.assertEqual(i % 2, 0)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: 1 != 0

======================================================================
FAIL: test_even (__main__.NumbersTest.test_even) (i=3)
Test that numbers between 0 and 5 are all even.
----------------------------------------------------------------------
Traceback (most recent call last):
  File "subtests.py", line 11, in test_even
    self.assertEqual(i % 2, 0)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: 1 != 0

======================================================================
FAIL: test_even (__main__.NumbersTest.test_even) (i=5)
Test that numbers between 0 and 5 are all even.
----------------------------------------------------------------------
Traceback (most recent call last):
  File "subtests.py", line 11, in test_even
    self.assertEqual(i % 2, 0)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: 1 != 0
```

Without using a subtest, execution would stop after the first failure, and the error would be less easy to diagnose because the value of `i` wouldn't be displayed:

```
======================================================================
FAIL: test_even (__main__.NumbersTest.test_even)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "subtests.py", line 32, in test_even
    self.assertEqual(i % 2, 0)
AssertionError: 1 != 0
```

Classes and functions

This section describes in depth the API of :mod:`unittest`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 711); *backlink*

Unknown interpreted text role "mod".

Test cases

Instances of the :class:`TestCase` class represent the logical test units in the :mod:`unittest` universe. This class is intended to be used as a base class, with specific tests being implemented by concrete subclasses. This class implements the interface needed by the test runner to allow it to drive the tests, and methods that the test code can use to check for and report various kinds of failure.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 721); *backlink*

Unknown interpreted text role "class".

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

main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 721); *backlink*

Unknown interpreted text role "mod".

Each instance of :class:`TestCase` will run a single base method: the method named *methodName*. In most uses of :class:`TestCase`, you will neither change the *methodName* nor reimplement the default `runTest()` method.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 728); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 728); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 733)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.2
   :class:`TestCase` can be instantiated successfully without providing a
   *methodName*. This makes it easier to experiment with :class:`TestCase`
   from the interactive interpreter.
```

:class:`TestCase` instances provide three groups of methods: one group used to run the test, another used by the test implementation to check conditions and report failures, and some inquiry methods allowing information about the test itself to be gathered.

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

Unknown interpreted text role "class".

Methods in the first group (running the test) are:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 745)

Unknown directive type "method".

```
.. method:: setUp()

   Method called to prepare the test fixture.  This is called immediately
   before calling the test method; other than :exc:`AssertionError` or :exc:`SkipTest`,
   any exception raised by this method will be considered an error rather than
   a test failure. The default implementation does nothing.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 753)

Unknown directive type "method".

```
.. method:: tearDown()

   Method called immediately after the test method has been called and the
   result recorded.  This is called even if the test method raised an
   exception, so the implementation in subclasses may need to be particularly
   careful about checking internal state.  Any exception, other than
   :exc:`AssertionError` or :exc:`SkipTest`, raised by this method will be
   considered an additional error rather than a test failure (thus increasing
   the total number of reported errors). This method will only be called if
   the :meth:`setUp` succeeds, regardless of the outcome of the test method.
   The default implementation does nothing.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 766)

Unknown directive type "method".

```
.. method:: setUpClass()

   A class method called before tests in an individual class are run.
   ``setUpClass`` is called with the class as the only argument
   and must be decorated as a :func:`classmethod`::

     @classmethod
     def setUpClass(cls):
         ...

   See `Class and Module Fixtures`_ for more details.

   .. versionadded:: 3.2
```

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

Unknown directive type "method".

```
.. method:: tearDownClass()

   A class method called after tests in an individual class have run.
   ``tearDownClass`` is called with the class as the only argument
   and must be decorated as a :meth:`classmethod`::

     @classmethod
     def tearDownClass(cls):
         ...

   See `Class and Module Fixtures`_ for more details.

   .. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 796)

Unknown directive type "method".

```
.. method:: run(result=None)

   Run the test, collecting the result into the :class:`TestResult` object
   passed as *result*.  If *result* is omitted or ``None``, a temporary
   result object is created (by calling the :meth:`defaultTestResult`
   method) and used. The result object is returned to :meth:`run`'s
   caller.

   The same effect may be had by simply calling the :class:`TestCase`
   instance.

   .. versionchanged:: 3.3
      Previous versions of ``run`` did not return the result. Neither did
      calling an instance.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 811)

Unknown directive type "method".

```
.. method:: skipTest(reason)

   Calling this during a test method or :meth:`setUp` skips the current
   test.  See :ref:`unittest-skipping` for more information.

   .. versionadded:: 3.1
```

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

Unknown directive type "method".

```
.. method:: subTest(msg=None, **params)

   Return a context manager which executes the enclosed code block as a
   subtest.  *msg* and *params* are optional, arbitrary values which are
   displayed whenever a subtest fails, allowing you to identify them
   clearly.

   A test case can contain any number of subtest declarations, and
   they can be arbitrarily nested.

   See :ref:`subtests` for more information.

   .. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 834)

Unknown directive type "method".

```
.. method:: debug()

   Run the test without collecting the result.  This allows exceptions raised
   by the test to be propagated to the caller, and can be used to support
   running tests under a debugger.
```

The :class:`TestCase` class provides several assert methods to check for and report failures. The following table lists the most commonly used methods (see the tables below for more assert methods):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 842); *backlink*

Unknown interpreted text role "class".

Method	Checks that	New in

Method	Checks that	New in
:meth:`assertEqual(a, b) <TestCase.assertEqual>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 850); *backlink* Unknown interpreted text role "meth".	`a == b`	
:meth:`assertNotEqual(a, b) <TestCase.assertNotEqual>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 853); *backlink* Unknown interpreted text role "meth".	`a != b`	
:meth:`assertTrue(x) <TestCase.assertTrue>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 856); *backlink* Unknown interpreted text role "meth".	`bool(x) is True`	
:meth:`assertFalse(x) <TestCase.assertFalse>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 859); *backlink* Unknown interpreted text role "meth".	`bool(x) is False`	
:meth:`assertIs(a, b) <TestCase.assertIs>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 862); *backlink* Unknown interpreted text role "meth".	`a is b`	3.1
:meth:`assertIsNot(a, b) <TestCase.assertIsNot>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 865); *backlink* Unknown interpreted text role "meth".	`a is not b`	3.1
:meth:`assertIsNone(x) <TestCase.assertIsNone>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 868); *backlink* Unknown interpreted text role "meth".	`x is None`	3.1

Method	Checks that	New in
:meth:`assertIsNotNone(x) <TestCase.assertIsNotNone>` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 871); *backlink* Unknown interpreted text role "meth".	`x is not None`	3.1
:meth:`assertIn(a, b) <TestCase.assertIn>` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 874); *backlink* Unknown interpreted text role "meth".	`a in b`	3.1
:meth:`assertNotIn(a, b) <TestCase.assertNotIn>` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 877); *backlink* Unknown interpreted text role "meth".	`a not in b`	3.1
:meth:`assertIsInstance(a, b) <TestCase.assertIsInstance>` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 880); *backlink* Unknown interpreted text role "meth".	`isinstance(a, b)`	3.2
:meth:`assertNotIsInstance(a, b) <TestCase.assertNotIsInstance>` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 883); *backlink* Unknown interpreted text role "meth".	`not isinstance(a, b)`	3.2

All the assert methods accept a *msg* argument that, if specified, is used as the error message on failure (see also :data:`longMessage`). Note that the *msg* keyword argument can be passed to :meth:`assertRaises`, :meth:`assertRaisesRegex`, :meth:`assertWarns`, :meth:`assertWarnsRegex` only when they are used as a context manager.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 886); *backlink*

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 886); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 886); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 886); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 886); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 892)

Unknown directive type "method".

```
.. method:: assertEqual(first, second, msg=None)

   Test that *first* and *second* are equal.  If the values do not
   compare equal, the test will fail.

   In addition, if *first* and *second* are the exact same type and one of
   list, tuple, dict, set, frozenset or str or any type that a subclass
   registers with :meth:`addTypeEqualityFunc` the type-specific equality
   function will be called in order to generate a more useful default
   error message (see also the :ref:`list of type-specific methods
   <type-specific-methods>`).

   .. versionchanged:: 3.1
      Added the automatic calling of type-specific equality function.

   .. versionchanged:: 3.2
      :meth:`assertMultiLineEqual` added as the default type equality
      function for comparing strings.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 912)

Unknown directive type "method".

```
.. method:: assertNotEqual(first, second, msg=None)

   Test that *first* and *second* are not equal.  If the values do
   compare equal, the test will fail.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 917)

Unknown directive type "method".

```
.. method:: assertTrue(expr, msg=None)
            assertFalse(expr, msg=None)

   Test that *expr* is true (or false).

   Note that this is equivalent to ``bool(expr) is True`` and not to ``expr
   is True`` (use ``assertIs(expr, True)`` for the latter).  This method
   should also be avoided when more specific methods are available (e.g.
   ``assertEqual(a, b)`` instead of ``assertTrue(a == b)``), because they
   provide a better error message in case of failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 929)

Unknown directive type "method".

```
.. method:: assertIs(first, second, msg=None)
            assertIsNot(first, second, msg=None)

   Test that *first* and *second* are (or are not) the same object.

   .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 937)

Unknown directive type "method".

```
.. method:: assertIsNone(expr, msg=None)
            assertIsNotNone(expr, msg=None)

   Test that *expr* is (or is not) ``None``.

   .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 945)

Unknown directive type "method".

```
.. method:: assertIn(member, container, msg=None)
            assertNotIn(member, container, msg=None)

   Test that *member* is (or is not) in *container*.

   .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 953)

Unknown directive type "method".

```
.. method:: assertIsInstance(obj, cls, msg=None)
            assertNotIsInstance(obj, cls, msg=None)

   Test that *obj* is (or is not) an instance of *cls* (which can be a
   class or a tuple of classes, as supported by :func:`isinstance`).
   To check for the exact type, use :func:`assertIs(type(obj), cls) <assertIs>`.

   .. versionadded:: 3.2
```

It is also possible to check the production of exceptions, warnings, and log messages using the following methods:

Method	Checks that	New in
:meth:`assertRaises(exc, fun, *args, **kwds) <TestCase.assertRaises>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 971); *backlink* Unknown interpreted text role "meth".	`fun(*args, **kwds)` raises *exc*	
:meth:`assertRaisesRegex(exc, r, fun, *args, **kwds) <TestCase.assertRaisesRegex>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 974); *backlink* Unknown interpreted text role "meth".	`fun(*args, **kwds)` raises *exc* and the message matches regex *r*	3.1
:meth:`assertWarns(warn, fun, *args, **kwds) <TestCase.assertWarns>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 977); *backlink* Unknown interpreted text role "meth".	`fun(*args, **kwds)` raises *warn*	3.2
:meth:`assertWarnsRegex(warn, r, fun, *args, **kwds) <TestCase.assertWarnsRegex>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 980); *backlink* Unknown interpreted text role "meth".	`fun(*args, **kwds)` raises *warn* and the message matches regex *r*	3.2
:meth:`assertLogs(logger, level) <TestCase.assertLogs>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 983); *backlink* Unknown interpreted text role "meth".	The `with` block logs on *logger* with minimum *level*	3.4
:meth:`assertNoLogs(logger, level) <TestCase.assertNoLogs>` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 986); *backlink* Unknown interpreted text role "meth".	The `with` block does not log on *logger* with minimum *level*	3.10

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 989)

Unknown directive type "method".

```
.. method:: assertRaises(exception, callable, *args, **kwds)
            assertRaises(exception, *, msg=None)

   Test that an exception is raised when *callable* is called with any
   positional or keyword arguments that are also passed to
   :meth:`assertRaises`.  The test passes if *exception* is raised, is an
   error if another exception is raised, or fails if no exception is raised.
   To catch any of a group of exceptions, a tuple containing the exception
   classes may be passed as *exception*.

   If only the *exception* and possibly the *msg* arguments are given,
   return a context manager so that the code under test can be written
   inline rather than as a function::

      with self.assertRaises(SomeException):
          do_something()

   When used as a context manager, :meth:`assertRaises` accepts the
   additional keyword argument *msg*.

   The context manager will store the caught exception object in its
   :attr:`exception` attribute.  This can be useful if the intention
   is to perform additional checks on the exception raised::

      with self.assertRaises(SomeException) as cm:
          do_something()

      the_exception = cm.exception
      self.assertEqual(the_exception.error_code, 3)

   .. versionchanged:: 3.1
      Added the ability to use :meth:`assertRaises` as a context manager.

   .. versionchanged:: 3.2
      Added the :attr:`exception` attribute.

   .. versionchanged:: 3.3
      Added the *msg* keyword argument when used as a context manager.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1029)

Unknown directive type "method".

```
.. method:: assertRaisesRegex(exception, regex, callable, *args, **kwds)
            assertRaisesRegex(exception, regex, *, msg=None)

   Like :meth:`assertRaises` but also tests that *regex* matches
   on the string representation of the raised exception.  *regex* may be
   a regular expression object or a string containing a regular expression
   suitable for use by :func:`re.search`.  Examples::

      self.assertRaisesRegex(ValueError, "invalid literal for.*XYZ'$",
                             int, 'XYZ')

   or::

      with self.assertRaisesRegex(ValueError, 'literal'):
         int('XYZ')

   .. versionadded:: 3.1
      Added under the name ``assertRaisesRegexp``.

   .. versionchanged:: 3.2
      Renamed to :meth:`assertRaisesRegex`.

   .. versionchanged:: 3.3
      Added the *msg* keyword argument when used as a context manager.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1055)

Unknown directive type "method".

```
.. method:: assertWarns(warning, callable, *args, **kwds)
            assertWarns(warning, *, msg=None)

   Test that a warning is triggered when *callable* is called with any
   positional or keyword arguments that are also passed to
   :meth:`assertWarns`.  The test passes if *warning* is triggered and
   fails if it isn't.  Any exception is an error.
   To catch any of a group of warnings, a tuple containing the warning
   classes may be passed as *warnings*.

   If only the *warning* and possibly the *msg* arguments are given,
   return a context manager so that the code under test can be written
   inline rather than as a function::

      with self.assertWarns(SomeWarning):
          do_something()

   When used as a context manager, :meth:`assertWarns` accepts the
   additional keyword argument *msg*.

   The context manager will store the caught warning object in its
   :attr:`warning` attribute, and the source line which triggered the
   warnings in the :attr:`filename` and :attr:`lineno` attributes.
   This can be useful if the intention is to perform additional checks
```

```
   on the warning caught::

      with self.assertWarns(SomeWarning) as cm:
          do_something()

      self.assertIn('myfile.py', cm.filename)
      self.assertEqual(320, cm.lineno)

   This method works regardless of the warning filters in place when it
   is called.

   .. versionadded:: 3.2

   .. versionchanged:: 3.3
      Added the *msg* keyword argument when used as a context manager.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1096)

Unknown directive type "method".

```
.. method:: assertWarnsRegex(warning, regex, callable, *args, **kwds)
            assertWarnsRegex(warning, regex, *, msg=None)

   Like :meth:`assertWarns` but also tests that *regex* matches on the
   message of the triggered warning.  *regex* may be a regular expression
   object or a string containing a regular expression suitable for use
   by :func:`re.search`.  Example::

      self.assertWarnsRegex(DeprecationWarning,
                            r'legacy_function\(\) is deprecated',
                            legacy_function, 'XYZ')

   or::

      with self.assertWarnsRegex(RuntimeWarning, 'unsafe frobnicating'):
          frobnicate('/etc/passwd')

   .. versionadded:: 3.2

   .. versionchanged:: 3.3
      Added the *msg* keyword argument when used as a context manager.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1118)

Unknown directive type "method".

```
.. method:: assertLogs(logger=None, level=None)

   A context manager to test that at least one message is logged on
   the *logger* or one of its children, with at least the given
   *level*.

   If given, *logger* should be a :class:`logging.Logger` object or a
   :class:`str` giving the name of a logger.  The default is the root
   logger, which will catch all messages that were not blocked by a
   non-propagating descendent logger.

   If given, *level* should be either a numeric logging level or
   its string equivalent (for example either ``"ERROR"`` or
   :attr:`logging.ERROR`).  The default is :attr:`logging.INFO`.

   The test passes if at least one message emitted inside the ``with``
   block matches the *logger* and *level* conditions, otherwise it fails.

   The object returned by the context manager is a recording helper
   which keeps tracks of the matching log messages.  It has two
   attributes:

   .. attribute:: records

      A list of :class:`logging.LogRecord` objects of the matching
      log messages.

   .. attribute:: output

      A list of :class:`str` objects with the formatted output of
      matching messages.

   Example::

      with self.assertLogs('foo', level='INFO') as cm:
         logging.getLogger('foo').info('first message')
         logging.getLogger('foo.bar').error('second message')
      self.assertEqual(cm.output, ['INFO:foo:first message',
                                   'ERROR:foo.bar:second message'])

   .. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1160)

Unknown directive type "method".

```
.. method:: assertNoLogs(logger=None, level=None)

   A context manager to test that no messages are logged on
   the *logger* or one of its children, with at least the given
```

```
*level*.

If given, *logger* should be a :class:`logging.Logger` object or a
:class:`str` giving the name of a logger.  The default is the root
logger, which will catch all messages.

If given, *level* should be either a numeric logging level or
its string equivalent (for example either ``"ERROR"`` or
:attr:`logging.ERROR`).  The default is :attr:`logging.INFO`.

Unlike :meth:`assertLogs`, nothing will be returned by the context
manager.

.. versionadded:: 3.10
```

There are also other methods used to perform more specific checks, such as:

Method	Checks that	New in
:meth:`assertAlmostEqual(a, b) <TestCase.assertAlmostEqual>` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1185); *backlink* Unknown interpreted text role "meth".	`round(a-b, 7) == 0`	
:meth:`assertNotAlmostEqual(a, b) <TestCase.assertNotAlmostEqual>` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1188); *backlink* Unknown interpreted text role "meth".	`round(a-b, 7) != 0`	
:meth:`assertGreater(a, b) <TestCase.assertGreater>` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1191); *backlink* Unknown interpreted text role "meth".	`a > b`	3.1
:meth:`assertGreaterEqual(a, b) <TestCase.assertGreaterEqual>` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1194); *backlink* Unknown interpreted text role "meth".	`a >= b`	3.1
:meth:`assertLess(a, b) <TestCase.assertLess>` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1197); *backlink* Unknown interpreted text role "meth".	`a < b`	3.1
:meth:`assertLessEqual(a, b) <TestCase.assertLessEqual>` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1200); *backlink* Unknown interpreted text role "meth".	`a <= b`	3.1

Method	Checks that	New in
:meth:`assertRegex(s, r) <TestCase.assertRegex>` **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1203); *backlink*** Unknown interpreted text role "meth".	`r.search(s)`	3.1
:meth:`assertNotRegex(s, r) <TestCase.assertNotRegex>` **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1206); *backlink*** Unknown interpreted text role "meth".	`not r.search(s)`	3.2
:meth:`assertCountEqual(a, b) <TestCase.assertCountEqual>` **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1209); *backlink*** Unknown interpreted text role "meth".	*a* and *b* have the same elements in the same number, regardless of their order.	3.2

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1214)

Unknown directive type "method".

```
.. method:: assertAlmostEqual(first, second, places=7, msg=None, delta=None)
            assertNotAlmostEqual(first, second, places=7, msg=None, delta=None)

   Test that *first* and *second* are approximately (or not approximately)
   equal by computing the difference, rounding to the given number of
   decimal *places* (default 7), and comparing to zero.  Note that these
   methods round the values to the given number of *decimal places* (i.e.
   like the :func:`round` function) and not *significant digits*.

   If *delta* is supplied instead of *places* then the difference
   between *first* and *second* must be less or equal to (or greater than) *delta*.

   Supplying both *delta* and *places* raises a :exc:`TypeError`.

   .. versionchanged:: 3.2
      :meth:`assertAlmostEqual` automatically considers almost equal objects
      that compare equal.  :meth:`assertNotAlmostEqual` automatically fails
      if the objects compare equal.  Added the *delta* keyword argument.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1234)

Unknown directive type "method".

```
.. method:: assertGreater(first, second, msg=None)
            assertGreaterEqual(first, second, msg=None)
            assertLess(first, second, msg=None)
            assertLessEqual(first, second, msg=None)

   Test that *first* is respectively >, >=, < or <= than *second* depending
   on the method name.  If not, the test will fail::

      >>> self.assertGreaterEqual(3, 4)
      AssertionError: "3" unexpectedly not greater than or equal to "4"

   .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1248)

Unknown directive type "method".

```
.. method:: assertRegex(text, regex, msg=None)
            assertNotRegex(text, regex, msg=None)

   Test that a *regex* search matches (or does not match) *text*.  In case
   of failure, the error message will include the pattern and the *text* (or
   the pattern and the part of *text* that unexpectedly matched).  *regex*
   may be a regular expression object or a string containing a regular
   expression suitable for use by :func:`re.search`.
```

```
   .. versionadded:: 3.1
      Added under the name ``assertRegexpMatches``.
   .. versionchanged:: 3.2
      The method ``assertRegexpMatches()`` has been renamed to
      :meth:`.assertRegex`.
   .. versionadded:: 3.2
      :meth:`.assertNotRegex`.
   .. versionadded:: 3.5
      The name ``assertNotRegexpMatches`` is a deprecated alias
      for :meth:`.assertNotRegex`.
```

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

Unknown directive type "method".

```
.. method:: assertCountEqual(first, second, msg=None)

   Test that sequence *first* contains the same elements as *second*,
   regardless of their order. When they don't, an error message listing the
   differences between the sequences will be generated.

   Duplicate elements are *not* ignored when comparing *first* and
   *second*. It verifies whether each element has the same count in both
   sequences. Equivalent to:
   ``assertEqual(Counter(list(first)), Counter(list(second)))``
   but works with sequences of unhashable objects as well.

   .. versionadded:: 3.2
```

The :meth:`assertEqual` method dispatches the equality check for objects of the same type to different type-specific methods. These methods are already implemented for most of the built-in types, but it's also possible to register new methods using :meth:`addTypeEqualityFunc`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1286); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1286); *backlink*

Unknown interpreted text role "meth".

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

Unknown directive type "method".

```
.. method:: addTypeEqualityFunc(typeobj, function)

   Registers a type-specific method called by :meth:`assertEqual` to check
   if two objects of exactly the same *typeobj* (not subclasses) compare
   equal.  *function* must take two positional arguments and a third msg=None
   keyword argument just as :meth:`assertEqual` does.  It must raise
   :data:`self.failureException(msg) <failureException>` when inequality
   between the first two parameters is detected -- possibly providing useful
   information and explaining the inequalities in details in the error
   message.

   .. versionadded:: 3.1
```

The list of type-specific methods automatically used by :meth:`~TestCase.assertEqual` are summarized in the following table. Note that it's usually not necessary to invoke these methods directly.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1304); *backlink*

Unknown interpreted text role "meth".

Method	Used to compare	New in
:meth:`assertMultiLineEqual(a, b) <TestCase.assertMultiLineEqual>` **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1312); *backlink*** Unknown interpreted text role "meth".	strings	3.1

Method	Used to compare	New in
:meth:`assertSequenceEqual(a, b) <TestCase.assertSequenceEqual>` **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1315); *backlink*** Unknown interpreted text role "meth".	sequences	3.1
:meth:`assertListEqual(a, b) <TestCase.assertListEqual>` **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1318); *backlink*** Unknown interpreted text role "meth".	lists	3.1
:meth:`assertTupleEqual(a, b) <TestCase.assertTupleEqual>` **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1321); *backlink*** Unknown interpreted text role "meth".	tuples	3.1
:meth:`assertSetEqual(a, b) <TestCase.assertSetEqual>` **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1324); *backlink*** Unknown interpreted text role "meth".	sets or frozensets	3.1
:meth:`assertDictEqual(a, b) <TestCase.assertDictEqual>` **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1327); *backlink*** Unknown interpreted text role "meth".	dicts	3.1

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1332)

Unknown directive type "method".

```
.. method:: assertMultiLineEqual(first, second, msg=None)

   Test that the multiline string *first* is equal to the string *second*.
   When not equal a diff of the two strings highlighting the differences
   will be included in the error message. This method is used by default
   when comparing strings with :meth:`assertEqual`.

   .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1342)

Unknown directive type "method".

```
.. method:: assertSequenceEqual(first, second, msg=None, seq_type=None)

   Tests that two sequences are equal.  If a *seq_type* is supplied, both
   *first* and *second* must be instances of *seq_type* or a failure will
   be raised.  If the sequences are different an error message is
   constructed that shows the difference between the two.

   This method is not called directly by :meth:`assertEqual`, but
   it's used to implement :meth:`assertListEqual` and
   :meth:`assertTupleEqual`.

   .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1356)

Unknown directive type "method".

```
.. method:: assertListEqual(first, second, msg=None)
            assertTupleEqual(first, second, msg=None)

   Tests that two lists or tuples are equal.  If not, an error message is
   constructed that shows only the differences between the two.  An error
   is also raised if either of the parameters are of the wrong type.
   These methods are used by default when comparing lists or tuples with
   :meth:`assertEqual`.

   .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1368)

Unknown directive type "method".

```
.. method:: assertSetEqual(first, second, msg=None)

   Tests that two sets are equal.  If not, an error message is constructed
   that lists the differences between the sets.  This method is used by
   default when comparing sets or frozensets with :meth:`assertEqual`.

   Fails if either of *first* or *second* does not have a :meth:`set.difference`
   method.

   .. versionadded:: 3.1
```

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

Unknown directive type "method".

```
.. method:: assertDictEqual(first, second, msg=None)

   Test that two dictionaries are equal.  If not, an error message is
   constructed that shows the differences in the dictionaries. This
   method will be used by default to compare dictionaries in
   calls to :meth:`assertEqual`.

   .. versionadded:: 3.1
```

Finally the :class:`TestCase` provides the following methods and attributes:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1393); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1396)

Unknown directive type "method".

```
.. method:: fail(msg=None)

   Signals a test failure unconditionally, with *msg* or ``None`` for
   the error message.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1402)

Unknown directive type "attribute".

```
.. attribute:: failureException

   This class attribute gives the exception raised by the test method.  If a
   test framework needs to use a specialized exception, possibly to carry
   additional information, it must subclass this exception in order to "play
   fair" with the framework.  The initial value of this attribute is
   :exc:`AssertionError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1411)

Unknown directive type "attribute".

```
.. attribute:: longMessage

   This class attribute determines what happens when a custom failure message
   is passed as the msg argument to an assertXYY call that fails.
```

```
``True`` is the default value. In this case, the custom message is appended
to the end of the standard failure message.
When set to ``False``, the custom message replaces the standard message.

The class setting can be overridden in individual test methods by assigning
an instance attribute, self.longMessage, to ``True`` or ``False`` before
calling the assert methods.

The class setting gets reset before each test call.

.. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1428)

Unknown directive type "attribute".

```
.. attribute:: maxDiff

   This attribute controls the maximum length of diffs output by assert
   methods that report diffs on failure. It defaults to 80*8 characters.
   Assert methods affected by this attribute are
   :meth:`assertSequenceEqual` (including all the sequence comparison
   methods that delegate to it), :meth:`assertDictEqual` and
   :meth:`assertMultiLineEqual`.

   Setting ``maxDiff`` to ``None`` means that there is no maximum length of
   diffs.

   .. versionadded:: 3.2
```

Testing frameworks can use the following methods to collect information on the test:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1447)

Unknown directive type "method".

```
.. method:: countTestCases()

   Return the number of tests represented by this test object.  For
   :class:`TestCase` instances, this will always be ``1``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1453)

Unknown directive type "method".

```
.. method:: defaultTestResult()

   Return an instance of the test result class that should be used for this
   test case class (if no other result instance is provided to the
   :meth:`run` method).

   For :class:`TestCase` instances, this will always be an instance of
   :class:`TestResult`; subclasses of :class:`TestCase` should override this
   as necessary.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1464)

Unknown directive type "method".

```
.. method:: id()

   Return a string identifying the specific test case.  This is usually the
   full name of the test method, including the module and class name.
```

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

Unknown directive type "method".

```
.. method:: shortDescription()

   Returns a description of the test, or ``None`` if no description
   has been provided.  The default implementation of this method
   returns the first line of the test method's docstring, if available,
   or ``None``.

   .. versionchanged:: 3.1
      In 3.1 this was changed to add the test name to the short description
      even in the presence of a docstring.  This caused compatibility issues
      with unittest extensions and adding the test name was moved to the
      :class:`TextTestResult` in Python 3.2.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1484)

Unknown directive type "method".

```
.. method:: addCleanup(function, /, *args, **kwargs)

   Add a function to be called after :meth:`tearDown` to cleanup resources
   used during the test. Functions will be called in reverse order to the
   order they are added (:abbr:`LIFO (last-in, first-out)`).  They
   are called with any arguments and keyword arguments passed into
   :meth:`addCleanup` when they are added.

   If :meth:`setUp` fails, meaning that :meth:`tearDown` is not called,
   then any cleanup functions added will still be called.

   .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1498)

Unknown directive type "method".

```
.. method:: doCleanups()

   This method is called unconditionally after :meth:`tearDown`, or
   after :meth:`setUp` if :meth:`setUp` raises an exception.

   It is responsible for calling all the cleanup functions added by
   :meth:`addCleanup`. If you need cleanup functions to be called
   *prior* to :meth:`tearDown` then you can call :meth:`doCleanups`
   yourself.

   :meth:`doCleanups` pops methods off the stack of cleanup
   functions one at a time, so it can be called at any time.

   .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1513)

Unknown directive type "classmethod".

```
.. classmethod:: addClassCleanup(function, /, *args, **kwargs)

   Add a function to be called after :meth:`tearDownClass` to cleanup
   resources used during the test class. Functions will be called in reverse
   order to the order they are added (:abbr:`LIFO (last-in, first-out)`).
   They are called with any arguments and keyword arguments passed into
   :meth:`addClassCleanup` when they are added.

   If :meth:`setUpClass` fails, meaning that :meth:`tearDownClass` is not
   called, then any cleanup functions added will still be called.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1527)

Unknown directive type "classmethod".

```
.. classmethod:: doClassCleanups()

   This method is called unconditionally after :meth:`tearDownClass`, or
   after :meth:`setUpClass` if :meth:`setUpClass` raises an exception.

   It is responsible for calling all the cleanup functions added by
   :meth:`addClassCleanup`. If you need cleanup functions to be called
   *prior* to :meth:`tearDownClass` then you can call
   :meth:`doClassCleanups` yourself.

   :meth:`doClassCleanups` pops methods off the stack of cleanup
   functions one at a time, so it can be called at any time.

   .. versionadded:: 3.8
```

This class provides an API similar to :class:`TestCase` and also accepts coroutines as test functions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1545); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1548)

Unknown directive type "versionadded".

```
.. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1550)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: asyncSetUp()
```

```
Method called to prepare the test fixture. This is called after :meth:`setUp`.
This is called immediately before calling the test method; other than
:exc:`AssertionError` or :exc:`SkipTest`, any exception raised by this method
will be considered an error rather than a test failure. The default implementation
does nothing.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1558)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: asyncTearDown()

   Method called immediately after the test method has been called and the
   result recorded.  This is called before :meth:`tearDown`. This is called even if
   the test method raised an exception, so the implementation in subclasses may need
   to be particularly careful about checking internal state.  Any exception, other than
   :exc:`AssertionError` or :exc:`SkipTest`, raised by this method will be
   considered an additional error rather than a test failure (thus increasing
   the total number of reported errors). This method will only be called if
   the :meth:`asyncSetUp` succeeds, regardless of the outcome of the test method.
   The default implementation does nothing.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1570)

Unknown directive type "method".

```
.. method:: addAsyncCleanup(function, /, *args, **kwargs)

   This method accepts a coroutine that can be used as a cleanup function.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1574)

Unknown directive type "method".

```
.. method:: run(result=None)

   Sets up a new event loop to run the test, collecting the result into
   the :class:`TestResult` object passed as *result*.  If *result* is
   omitted or ``None``, a temporary result object is created (by calling
   the :meth:`defaultTestResult` method) and used. The result object is
   returned to :meth:`run`'s caller. At the end of the test all the tasks
   in the event loop are cancelled.
```

An example illustrating the order:

```
from unittest import IsolatedAsyncioTestCase

events = []


class Test(IsolatedAsyncioTestCase):


    def setUp(self):
        events.append("setUp")

    async def asyncSetUp(self):
        self._async_connection = await AsyncConnection()
        events.append("asyncSetUp")

    async def test_response(self):
        events.append("test_response")
        response = await self._async_connection.get("https://example.com")
        self.assertEqual(response.status_code, 200)
        self.addAsyncCleanup(self.on_cleanup)

    def tearDown(self):
        events.append("tearDown")

    async def asyncTearDown(self):
        await self._async_connection.close()
        events.append("asyncTearDown")

    async def on_cleanup(self):
        events.append("cleanup")

if __name__ == "__main__":
    unittest.main()
```

After running the test, `events` would contain `["setUp", "asyncSetUp", "test_response", "asyncTearDown", "tearDown", "cleanup"]`.

This class implements the portion of the :class:`TestCase` interface which allows the test runner to drive the test, but does not provide the methods which test code can use to check and report errors. This is used to create test cases using legacy test code, allowing it to be integrated into a :mod:`unittest`-based test framework.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1625); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1625); *backlink*

Unknown interpreted text role "mod".

Deprecated aliases

For historical reasons, some of the :class:`TestCase` methods had one or more aliases that are now deprecated. The following table lists the correct names along with their deprecated aliases:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1637); *backlink*

Unknown interpreted text role "class".

Method Name	Deprecated alias	Deprecated alias
:meth:`.assertEqual` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1645); *backlink* Unknown interpreted text role "meth".	failUnlessEqual	assertEquals
:meth:`.assertNotEqual` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1646); *backlink* Unknown interpreted text role "meth".	failIfEqual	assertNotEquals
:meth:`.assertTrue` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1647); *backlink* Unknown interpreted text role "meth".	failUnless	assert_
:meth:`.assertFalse` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1648); *backlink* Unknown interpreted text role "meth".	failIf	

Method Name	Deprecated alias	Deprecated alias
:meth:`.assertRaises` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1649); *backlink* Unknown interpreted text role "meth".	failUnlessRaises	
:meth:`.assertAlmostEqual` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1650); *backlink* Unknown interpreted text role "meth".	failUnlessAlmostEqual	assertAlmostEquals
:meth:`.assertNotAlmostEqual` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1651); *backlink* Unknown interpreted text role "meth".	failIfAlmostEqual	assertNotAlmostEquals
:meth:`.assertRegex` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1652); *backlink* Unknown interpreted text role "meth".		assertRegexpMatches
:meth:`.assertNotRegex` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1653); *backlink* Unknown interpreted text role "meth".		assertNotRegexpMatches

Method Name	Deprecated alias	Deprecated alias
:meth:`.assertRaisesRegex` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1654); *backlink* Unknown interpreted text role "meth".		assertRaisesRegexp

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1656)

Unknown directive type "deprecated".

```
.. deprecated:: 3.1
      The fail* aliases listed in the second column have been deprecated.
```

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

Unknown directive type "deprecated".

```
.. deprecated:: 3.2
      The assert* aliases listed in the third column have been deprecated.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1660)

Unknown directive type "deprecated".

```
.. deprecated:: 3.2
      ``assertRegexpMatches`` and ``assertRaisesRegexp`` have been renamed to
      :meth:`.assertRegex` and :meth:`.assertRaisesRegex`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1663)

Unknown directive type "deprecated".

```
.. deprecated:: 3.5
      The ``assertNotRegexpMatches`` name is deprecated in favor of :meth:`.assertNotRegex`.
```

Grouping tests

This class represents an aggregation of individual test cases and test suites. The class presents the interface needed by the test runner to allow it to be run as any other test case. Running a :class:`TestSuite` instance is the same as iterating over the suite, running each test individually.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1673); *backlink*

Unknown interpreted text role "class".

If *tests* is given, it must be an iterable of individual test cases or other test suites that will be used to build the suite initially. Additional methods are provided to add test cases and suites to the collection later on.

:class:`TestSuite` objects behave much like :class:`TestCase` objects, except they do not actually implement a test. Instead, they are used to aggregate tests into groups of tests that should be run together. Some additional methods are available to add tests to :class:`TestSuite` instances:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1682); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1682); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1682); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1688)

Unknown directive type "method".

```
.. method:: TestSuite.addTest(test)

   Add a :class:`TestCase` or :class:`TestSuite` to the suite.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1693)

Unknown directive type "method".

```
.. method:: TestSuite.addTests(tests)

   Add all the tests from an iterable of :class:`TestCase` and :class:`TestSuite`
   instances to this test suite.

   This is equivalent to iterating over *tests*, calling :meth:`addTest` for
   each element.
```

:class:`TestSuite` shares the following methods with :class:`TestCase`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1701); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1701); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1704)

Unknown directive type "method".

```
.. method:: run(result)

   Run the tests associated with this suite, collecting the result into the
   test result object passed as *result*.  Note that unlike
   :meth:`TestCase.run`, :meth:`TestSuite.run` requires the result object to
   be passed in.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1712)

Unknown directive type "method".

```
.. method:: debug()

   Run the tests associated with this suite without collecting the
   result. This allows exceptions raised by the test to be propagated to the
   caller and can be used to support running tests under a debugger.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1719)

Unknown directive type "method".

```
.. method:: countTestCases()

   Return the number of tests represented by this test object, including all
   individual tests and sub-suites.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1725)

Unknown directive type "method".

```
.. method:: __iter__()

   Tests grouped by a :class:`TestSuite` are always accessed by iteration.
   Subclasses can lazily provide tests by overriding :meth:`__iter__`. Note
   that this method may be called several times on a single suite (for
   example when counting tests or comparing for equality) so the tests
   returned by repeated iterations before :meth:`TestSuite.run` must be the
   same for each call iteration. After :meth:`TestSuite.run`, callers should
   not rely on the tests returned by this method unless the caller uses a
   subclass that overrides :meth:`TestSuite._removeTestAtIndex` to preserve
   test references.

   .. versionchanged:: 3.2
      In earlier versions the :class:`TestSuite` accessed tests directly rather
```

```
   than through iteration, so overriding :meth:`__iter__` wasn't sufficient
   for providing tests.

.. versionchanged:: 3.4
   In earlier versions the :class:`TestSuite` held references to each
   :class:`TestCase` after :meth:`TestSuite.run`. Subclasses can restore
   that behavior by overriding :meth:`TestSuite._removeTestAtIndex`.
```

In the typical usage of a :class:`TestSuite` object, the :meth:`run` method is invoked by a :class:`TestRunner` rather than by the end-user test harness.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1747); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1747); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1747); *backlink*

Unknown interpreted text role "class".

Loading and running tests

The :class:`TestLoader` class is used to create test suites from classes and modules. Normally, there is no need to create an instance of this class; the :mod:`unittest` module provides an instance that can be shared as :data:`unittest.defaultTestLoader`. Using a subclass or instance, however, allows customization of some configurable properties.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1756); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1756); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1756); *backlink*

Unknown interpreted text role "data".

:class:`TestLoader` objects have the following attributes:

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

Unknown interpreted text role "class".

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

Unknown directive type "attribute".

```
.. attribute:: errors

   A list of the non-fatal errors encountered while loading tests. Not reset
   by the loader at any point. Fatal errors are signalled by the relevant
   a method raising an exception to the caller. Non-fatal errors are also
   indicated by a synthetic test that will raise the original error when
   run.

   .. versionadded:: 3.5
```

:class:`TestLoader` objects have the following methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1776); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1779)

Unknown directive type "method".

```
.. method:: loadTestsFromTestCase(testCaseClass)

   Return a suite of all test cases contained in the :class:`TestCase`\ -derived
   :class:`testCaseClass`.

   A test case instance is created for each method named by
   :meth:`getTestCaseNames`. By default these are the method names
```

```
   beginning with ``test``. If :meth:`getTestCaseNames` returns no
   methods, but the :meth:`runTest` method is implemented, a single test
   case is created for that method instead.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1791)

Unknown directive type "method".

```
.. method:: loadTestsFromModule(module, pattern=None)

   Return a suite of all test cases contained in the given module. This
   method searches *module* for classes derived from :class:`TestCase` and
   creates an instance of the class for each test method defined for the
   class.

   .. note::

      While using a hierarchy of :class:`TestCase`\ -derived classes can be
      convenient in sharing fixtures and helper functions, defining test
      methods on base classes that are not intended to be instantiated
      directly does not play well with this method.  Doing so, however, can
      be useful when the fixtures are different and defined in subclasses.

   If a module provides a ``load_tests`` function it will be called to
   load the tests. This allows modules to customize test loading.
   This is the `load_tests protocol`_.  The *pattern* argument is passed as
   the third argument to ``load_tests``.

   .. versionchanged:: 3.2
      Support for ``load_tests`` added.

   .. versionchanged:: 3.5
      The undocumented and unofficial *use_load_tests* default argument is
      deprecated and ignored, although it is still accepted for backward
      compatibility.  The method also now accepts a keyword-only argument
      *pattern* which is passed to ``load_tests`` as the third argument.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1821)

Unknown directive type "method".

```
.. method:: loadTestsFromName(name, module=None)

   Return a suite of all test cases given a string specifier.

   The specifier *name* is a "dotted name" that may resolve either to a
   module, a test case class, a test method within a test case class, a
   :class:`TestSuite` instance, or a callable object which returns a
   :class:`TestCase` or :class:`TestSuite` instance.  These checks are
   applied in the order listed here; that is, a method on a possible test
   case class will be picked up as "a test method within a test case class",
   rather than "a callable object".

   For example, if you have a module :mod:`SampleTests` containing a
   :class:`TestCase`\ -derived class :class:`SampleTestCase` with three test
   methods (:meth:`test_one`, :meth:`test_two`, and :meth:`test_three`), the
   specifier ``'SampleTests.SampleTestCase'`` would cause this method to
   return a suite which will run all three test methods. Using the specifier
   ``'SampleTests.SampleTestCase.test_two'`` would cause it to return a test
   suite which will run only the :meth:`test_two` test method. The specifier
   can refer to modules and packages which have not been imported; they will
   be imported as a side-effect.

   The method optionally resolves *name* relative to the given *module*.

   .. versionchanged:: 3.5
      If an :exc:`ImportError` or :exc:`AttributeError` occurs while traversing
      *name* then a synthetic test that raises that error when run will be
      returned. These errors are included in the errors accumulated by
      self.errors.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1852)

Unknown directive type "method".

```
.. method:: loadTestsFromNames(names, module=None)

   Similar to :meth:`loadTestsFromName`, but takes a sequence of names rather
   than a single name.  The return value is a test suite which supports all
   the tests defined for each name.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1859)

Unknown directive type "method".

```
.. method:: getTestCaseNames(testCaseClass)

   Return a sorted sequence of method names found within *testCaseClass*;
   this should be a subclass of :class:`TestCase`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1865)

Unknown directive type "method".

```
.. method:: discover(start_dir, pattern='test*.py', top_level_dir=None)

   Find all the test modules by recursing into subdirectories from the
   specified start directory, and return a TestSuite object containing them.
   Only test files that match *pattern* will be loaded. (Using shell style
   pattern matching.) Only module names that are importable (i.e. are valid
   Python identifiers) will be loaded.

   All test modules must be importable from the top level of the project. If
   the start directory is not the top level directory then the top level
   directory must be specified separately.

   If importing a module fails, for example due to a syntax error, then
   this will be recorded as a single error and discovery will continue.  If
   the import failure is due to :exc:`SkipTest` being raised, it will be
   recorded as a skip instead of an error.

   If a package (a directory containing a file named :file:`__init__.py`) is
   found, the package will be checked for a ``load_tests`` function. If this
   exists then it will be called
   ``package.load_tests(loader, tests, pattern)``. Test discovery takes care
   to ensure that a package is only checked for tests once during an
   invocation, even if the load_tests function itself calls
   ``loader.discover``.

   If ``load_tests`` exists then discovery does *not* recurse into the
   package, ``load_tests`` is responsible for loading all tests in the
   package.

   The pattern is deliberately not stored as a loader attribute so that
   packages can continue discovery themselves. *top_level_dir* is stored so
   ``load_tests`` does not need to pass this argument in to
   ``loader.discover()``.

   *start_dir* can be a dotted module name as well as a directory.

   .. versionadded:: 3.2

   .. versionchanged:: 3.4
      Modules that raise :exc:`SkipTest` on import are recorded as skips,
      not errors.

   .. versionchanged:: 3.4
      *start_dir* can be a :term:`namespace packages <namespace package>`.

   .. versionchanged:: 3.4
      Paths are sorted before being imported so that execution order is the
      same even if the underlying file system's ordering is not dependent
      on file name.

   .. versionchanged:: 3.5
      Found packages are now checked for ``load_tests`` regardless of
      whether their path matches *pattern*, because it is impossible for
      a package name to match the default pattern.

   .. versionchanged:: 3.11
      *start_dir* can not be a :term:`namespace packages <namespace package>`.
      It has been broken since Python 3.7 and Python 3.11 officially remove it.
```

The following attributes of a :class:`TestLoader` can be configured either by subclassing or assignment on an instance:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1925); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1929)

Unknown directive type "attribute".

```
.. attribute:: testMethodPrefix

   String giving the prefix of method names which will be interpreted as test
   methods.  The default value is ``'test'``.

   This affects :meth:`getTestCaseNames` and all the :meth:`loadTestsFrom\*`
   methods.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1938)

Unknown directive type "attribute".

```
.. attribute:: sortTestMethodsUsing

   Function to be used to compare method names when sorting them in
   :meth:`getTestCaseNames` and all the :meth:`loadTestsFrom\*` methods.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1944)

Unknown directive type "attribute".

```
.. attribute:: suiteClass

   Callable object that constructs a test suite from a list of tests. No
   methods on the resulting object are needed.  The default value is the
   :class:`TestSuite` class.

   This affects all the :meth:`loadTestsFrom\*` methods.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1952)

Unknown directive type "attribute".

```
.. attribute:: testNamePatterns

   List of Unix shell-style wildcard test name patterns that test methods
   have to match to be included in test suites (see ``-v`` option).

   If this attribute is not ``None`` (the default), all test methods to be
   included in test suites must match one of the patterns in this list.
   Note that matches are always performed using :meth:`fnmatch.fnmatchcase`,
   so unlike patterns passed to the ``-v`` option, simple substring patterns
   will have to be converted using ``*`` wildcards.

   This affects all the :meth:`loadTestsFrom\*` methods.

   .. versionadded:: 3.7
```

This class is used to compile information about which tests have succeeded and which have failed.

A :class:`TestResult` object stores the results of a set of tests. The :class:`TestCase` and :class:`TestSuite` classes ensure that results are properly recorded; test authors do not need to worry about recording the outcome of tests.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1973); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1973); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1973); *backlink*

Unknown interpreted text role "class".

Testing frameworks built on top of :mod:`unittest` may want access to the :class:`TestResult` object generated by running a set of tests for reporting purposes; a :class:`TestResult` instance is returned by the :meth:`TestRunner.run` method for this purpose.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1978); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1978); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1978); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1978); *backlink*

Unknown interpreted text role "meth".

:class:`TestResult` instances have the following attributes that will be of interest when inspecting the results of running a set of tests:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1983); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1987)

Unknown directive type "attribute".

```
.. attribute:: errors
```

```
      A list containing 2-tuples of :class:`TestCase` instances and strings
      holding formatted tracebacks. Each tuple represents a test which raised an
      unexpected exception.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1993)

Unknown directive type "attribute".

```
   .. attribute:: failures

      A list containing 2-tuples of :class:`TestCase` instances and strings
      holding formatted tracebacks. Each tuple represents a test where a failure
      was explicitly signalled using the :meth:`TestCase.assert\*` methods.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 1999)

Unknown directive type "attribute".

```
   .. attribute:: skipped

      A list containing 2-tuples of :class:`TestCase` instances and strings
      holding the reason for skipping the test.

      .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2006)

Unknown directive type "attribute".

```
   .. attribute:: expectedFailures

      A list containing 2-tuples of :class:`TestCase` instances and strings
      holding formatted tracebacks.  Each tuple represents an expected failure
      or error of the test case.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2012)

Unknown directive type "attribute".

```
   .. attribute:: unexpectedSuccesses

      A list containing :class:`TestCase` instances that were marked as expected
      failures, but succeeded.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2017)

Unknown directive type "attribute".

```
   .. attribute:: shouldStop

      Set to ``True`` when the execution of tests should stop by :meth:`stop`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2021)

Unknown directive type "attribute".

```
   .. attribute:: testsRun

      The total number of tests run so far.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2025)

Unknown directive type "attribute".

```
   .. attribute:: buffer

      If set to true, ``sys.stdout`` and ``sys.stderr`` will be buffered in between
      :meth:`startTest` and :meth:`stopTest` being called. Collected output will
      only be echoed onto the real ``sys.stdout`` and ``sys.stderr`` if the test
      fails or errors. Any output is also attached to the failure / error message.

      .. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2034)

Unknown directive type "attribute".

```
   .. attribute:: failfast

      If set to true :meth:`stop` will be called on the first failure or error,
      halting the test run.
```

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2041)

Unknown directive type "attribute".

```
.. attribute:: tb_locals

   If set to true then local variables will be shown in tracebacks.

   .. versionadded:: 3.5
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2047)

Unknown directive type "method".

```
.. method:: wasSuccessful()

   Return ``True`` if all tests run so far have passed, otherwise returns
   ``False``.

   .. versionchanged:: 3.4
      Returns ``False`` if there were any :attr:`unexpectedSuccesses`
      from tests marked with the :func:`expectedFailure` decorator.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2056)

Unknown directive type "method".

```
.. method:: stop()

   This method can be called to signal that the set of tests being run should
   be aborted by setting the :attr:`shouldStop` attribute to ``True``.
   :class:`TestRunner` objects should respect this flag and return without
   running any additional tests.

   For example, this feature is used by the :class:`TextTestRunner` class to
   stop the test framework when the user signals an interrupt from the
   keyboard.  Interactive tools which provide :class:`TestRunner`
   implementations can use this in a similar manner.
```

The following methods of the :class:`TestResult` class are used to maintain the internal data structures, and may be extended in subclasses to support additional reporting requirements. This is particularly useful in building tools which support interactive reporting while tests are being run.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2068); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2074)

Unknown directive type "method".

```
.. method:: startTest(test)

   Called when the test case *test* is about to be run.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2078)

Unknown directive type "method".

```
.. method:: stopTest(test)

   Called after the test case *test* has been executed, regardless of the
   outcome.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2083)

Unknown directive type "method".

```
.. method:: startTestRun()

   Called once before any tests are executed.

   .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2090)

Unknown directive type "method".

```
.. method:: stopTestRun()
```

```
   Called once after all tests are executed.

   .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2097)

Unknown directive type "method".

```
.. method:: addError(test, err)

   Called when the test case *test* raises an unexpected exception. *err* is a
   tuple of the form returned by :func:`sys.exc_info`: ``(type, value,
   traceback)``.

   The default implementation appends a tuple ``(test, formatted_err)`` to
   the instance's :attr:`errors` attribute, where *formatted_err* is a
   formatted traceback derived from *err*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2108)

Unknown directive type "method".

```
.. method:: addFailure(test, err)

   Called when the test case *test* signals a failure. *err* is a tuple of
   the form returned by :func:`sys.exc_info`: ``(type, value, traceback)``.

   The default implementation appends a tuple ``(test, formatted_err)`` to
   the instance's :attr:`failures` attribute, where *formatted_err* is a
   formatted traceback derived from *err*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2118)

Unknown directive type "method".

```
.. method:: addSuccess(test)

   Called when the test case *test* succeeds.

   The default implementation does nothing.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2125)

Unknown directive type "method".

```
.. method:: addSkip(test, reason)

   Called when the test case *test* is skipped.  *reason* is the reason the
   test gave for skipping.

   The default implementation appends a tuple ``(test, reason)`` to the
   instance's :attr:`skipped` attribute.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2134)

Unknown directive type "method".

```
.. method:: addExpectedFailure(test, err)

   Called when the test case *test* fails or errors, but was marked with
   the :func:`expectedFailure` decorator.

   The default implementation appends a tuple ``(test, formatted_err)`` to
   the instance's :attr:`expectedFailures` attribute, where *formatted_err*
   is a formatted traceback derived from *err*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2144)

Unknown directive type "method".

```
.. method:: addUnexpectedSuccess(test)

   Called when the test case *test* was marked with the
   :func:`expectedFailure` decorator, but succeeded.

   The default implementation appends the test to the instance's
   :attr:`unexpectedSuccesses` attribute.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2153)

Unknown directive type "method".

```
.. method:: addSubTest(test, subtest, outcome)

   Called when a subtest finishes.  *test* is the test case
   corresponding to the test method.  *subtest* is a custom
   :class:`TestCase` instance describing the subtest.

   If *outcome* is :const:`None`, the subtest succeeded.  Otherwise,
   it failed with an exception where *outcome* is a tuple of the form
   returned by :func:`sys.exc_info`: ``(type, value, traceback)``.

   The default implementation does nothing when the outcome is a
   success, and records subtest failures as normal failures.

   .. versionadded:: 3.4
```

A concrete implementation of :class:`TestResult` used by the :class:`TextTestRunner`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2171); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2171); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2174)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
   This class was previously named ``_TextTestResult``. The old name still
   exists as an alias but is deprecated.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2179)

Unknown directive type "data".

```
.. data:: defaultTestLoader

   Instance of the :class:`TestLoader` class intended to be shared.  If no
   customization of the :class:`TestLoader` is needed, this instance can be used
   instead of repeatedly creating new instances.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2186)

Invalid class attribute value for "class" directive: "TextTestRunner(stream=None, descriptions=True, verbosity=1, failfast=False, \ buffer=False, resultclass=None, warnings=None, *, tb_locals=False)".

```
.. class:: TextTestRunner(stream=None, descriptions=True, verbosity=1, failfast=False, \
                          buffer=False, resultclass=None, warnings=None, *, tb_locals=False)

   A basic test runner implementation that outputs results to a stream. If *stream*
   is ``None``, the default, :data:`sys.stderr` is used as the output stream. This class
   has a few configurable parameters, but is essentially very simple.  Graphical
   applications which run test suites should provide alternate implementations. Such
   implementations should accept ``**kwargs`` as the interface to construct runners
   changes when features are added to unittest.

   By default this runner shows :exc:`DeprecationWarning`,
   :exc:`PendingDeprecationWarning`, :exc:`ResourceWarning` and
   :exc:`ImportWarning` even if they are :ref:`ignored by default
   <warning-ignored>`. Deprecation warnings caused by :ref:`deprecated unittest
   methods <deprecated-aliases>` are also special-cased and, when the warning
   filters are ``'default'`` or ``'always'``, they will appear only once
   per-module, in order to avoid too many warning messages.  This behavior can
   be overridden using Python's :option:`!-Wd` or :option:`!-Wa` options
   (see :ref:`Warning control <using-on-warnings>`) and leaving
   *warnings* to ``None``.

   .. versionchanged:: 3.2
      Added the ``warnings`` argument.

   .. versionchanged:: 3.2
      The default stream is set to :data:`sys.stderr` at instantiation time rather
      than import time.

   .. versionchanged:: 3.5
      Added the tb_locals parameter.

   .. method:: _makeResult()

      This method returns the instance of ``TestResult`` used by :meth:`run`.
      It is not intended to be called directly, but can be overridden in
      subclasses to provide a custom ``TestResult``.
```

```
        ``_makeResult()`` instantiates the class or callable passed in the
        ``TextTestRunner`` constructor as the ``resultclass`` argument. It
        defaults to :class:`TextTestResult` if no ``resultclass`` is provided.
        The result class is instantiated with the following arguments::

          stream, descriptions, verbosity

    .. method:: run(test)

        This method is the main public interface to the ``TextTestRunner``. This
        method takes a :class:`TestSuite` or :class:`TestCase` instance. A
        :class:`TestResult` is created by calling
        :func:`_makeResult` and the test(s) are run and the
        results printed to stdout.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2239)

Unknown directive type "function".

```
.. function:: main(module='__main__', defaultTest=None, argv=None, testRunner=None, \
                   testLoader=unittest.defaultTestLoader, exit=True, verbosity=1, \
                   failfast=None, catchbreak=None, buffer=None, warnings=None)

   A command-line program that loads a set of tests from *module* and runs them;
   this is primarily for making test modules conveniently executable.
   The simplest use for this function is to include the following line at the
   end of a test script::

      if __name__ == '__main__':
          unittest.main()

   You can run tests with more detailed information by passing in the verbosity
   argument::

      if __name__ == '__main__':
          unittest.main(verbosity=2)

   The *defaultTest* argument is either the name of a single test or an
   iterable of test names to run if no test names are specified via *argv*.  If
   not specified or ``None`` and no test names are provided via *argv*, all
   tests found in *module* are run.

   The *argv* argument can be a list of options passed to the program, with the
   first element being the program name.  If not specified or ``None``,
   the values of :data:`sys.argv` are used.

   The *testRunner* argument can either be a test runner class or an already
   created instance of it. By default ``main`` calls :func:`sys.exit` with
   an exit code indicating success or failure of the tests run.

   The *testLoader* argument has to be a :class:`TestLoader` instance,
   and defaults to :data:`defaultTestLoader`.

   ``main`` supports being used from the interactive interpreter by passing in the
   argument ``exit=False``. This displays the result on standard output without
   calling :func:`sys.exit`::

      >>> from unittest import main
      >>> main(module='test_module', exit=False)

   The *failfast*, *catchbreak* and *buffer* parameters have the same
   effect as the same-name `command-line options`_.

   The *warnings* argument specifies the :ref:`warning filter <warning-filter>`
   that should be used while running the tests.  If it's not specified, it will
   remain ``None`` if a :option:`!-W` option is passed to :program:`python`
   (see :ref:`Warning control <using-on-warnings>`),
   otherwise it will be set to ``'default'``.

   Calling ``main`` actually returns an instance of the ``TestProgram`` class.
   This stores the result of the tests run as the ``result`` attribute.

   .. versionchanged:: 3.1
      The *exit* parameter was added.

   .. versionchanged:: 3.2
      The *verbosity*, *failfast*, *catchbreak*, *buffer*
      and *warnings* parameters were added.

   .. versionchanged:: 3.4
      The *defaultTest* parameter was changed to also accept an iterable of
      test names.
```

load_tests Protocol

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2307)

Unknown directive type "versionadded".

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

Modules or packages can customize how tests are loaded from them during normal test runs or test discovery by implementing a function called `load_tests`.

If a test module defines `load_tests` it will be called by :meth:`TestLoader.loadTestsFromModule` with the following arguments:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2312); backlink

Unknown interpreted text role "meth".

```
load_tests(loader, standard_tests, pattern)
```

where *pattern* is passed straight through from `loadTestsFromModule`. It defaults to `None`.

It should return a :class:`TestSuite`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2320); backlink

Unknown interpreted text role "class".

loader is the instance of :class:`TestLoader` doing the loading. *standard_tests* are the tests that would be loaded by default from the module. It is common for test modules to only want to add or remove tests from the standard set of tests. The third argument is used when loading packages as part of test discovery.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2322); backlink

Unknown interpreted text role "class".

A typical `load_tests` function that loads tests from a specific set of :class:`TestCase` classes may look like:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2328); backlink

Unknown interpreted text role "class".

```
test_cases = (TestCase1, TestCase2, TestCase3)

def load_tests(loader, tests, pattern):
    suite = TestSuite()
    for test_class in test_cases:
        tests = loader.loadTestsFromTestCase(test_class)
        suite.addTests(tests)
    return suite
```

If discovery is started in a directory containing a package, either from the command line or by calling :meth:`TestLoader.discover`, then the package :file:`__init__.py` will be checked for `load_tests`. If that function does not exist, discovery will recurse into the package as though it were just another directory. Otherwise, discovery of the package's tests will be left up to `load_tests` which is called with the following arguments:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2340); backlink

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2340); backlink

Unknown interpreted text role "file".

```
load_tests(loader, standard_tests, pattern)
```

This should return a :class:`TestSuite` representing all the tests from the package. (`standard_tests` will only contain tests collected from :file:`__init__.py`.)

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2349); backlink

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2349); backlink

Unknown interpreted text role "file".

Because the pattern is passed into `load_tests` the package is free to continue (and potentially modify) test discovery. A 'do nothing' `load_tests` function for a test package would look like:

```
def load_tests(loader, standard_tests, pattern):
    # top level directory cached on loader instance
    this_dir = os.path.dirname(__file__)
    package_tests = loader.discover(start_dir=this_dir, pattern=pattern)
    standard_tests.addTests(package_tests)
    return standard_tests
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2364)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.5
   Discovery no longer checks package names for matching *pattern* due to the
   impossibility of package names matching the default pattern.
```

Class and Module Fixtures

Class and module level fixtures are implemented in :class:`TestSuite`. When the test suite encounters a test from a new class then :meth:`tearDownClass` from the previous class (if there is one) is called, followed by :meth:`setUpClass` from the new class.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2373); *backlink*
>
> Unknown interpreted text role "class".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2373); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2373); *backlink*
>
> Unknown interpreted text role "meth".

Similarly if a test is from a different module from the previous test then `tearDownModule` from the previous module is run, followed by `setUpModule` from the new module.

After all the tests have run the final `tearDownClass` and `tearDownModule` are run.

Note that shared fixtures do not play well with [potential] features like test parallelization and they break test isolation. They should be used with care.

The default ordering of tests created by the unittest test loaders is to group all tests from the same modules and classes together. This will lead to `setUpClass` / `setUpModule` (etc) being called exactly once per class and module. If you randomize the order, so that tests from different modules and classes are adjacent to each other, then these shared fixture functions may be called multiple times in a single test run.

Shared fixtures are not intended to work with suites with non-standard ordering. A `BaseTestSuite` still exists for frameworks that don't want to support shared fixtures.

If there are any exceptions raised during one of the shared fixture functions the test is reported as an error. Because there is no corresponding test instance an `_ErrorHolder` object (that has the same interface as a :class:`TestCase`) is created to represent the error. If you are just using the standard unittest test runner then this detail doesn't matter, but if you are a framework author it may be relevant.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2399); *backlink*
>
> Unknown interpreted text role "class".

setUpClass and tearDownClass

These must be implemented as class methods:

```
import unittest

class Test(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls._connection = createExpensiveConnectionObject()

    @classmethod
    def tearDownClass(cls):
        cls._connection.destroy()
```

If you want the `setUpClass` and `tearDownClass` on base classes called then you must call up to them yourself. The implementations in :class:`TestCase` are empty.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2423); *backlink*
>
> Unknown interpreted text role "class".

If an exception is raised during a `setUpClass` then the tests in the class are not run and the `tearDownClass` is not run. Skipped classes will not have `setUpClass` or `tearDownClass` run. If the exception is a :exc:`SkipTest` exception then the class will be reported as having been skipped instead of as an error.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2427); *backlink*
>
> Unknown interpreted text role "exc".

setUpModule and tearDownModule

These should be implemented as functions:

```
def setUpModule():
    createConnection()

def tearDownModule():
    closeConnection()
```

If an exception is raised in a `setUpModule` then none of the tests in the module will be run and the `tearDownModule` will not be run. If the exception is a :exc:`SkipTest` exception then the module will be reported as having been skipped instead of as an error.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2445); *backlink*

Unknown interpreted text role "exc".

To add cleanup code that must be run even in the case of an exception, use `addModuleCleanup`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2454)

Unknown directive type "function".

```
.. function:: addModuleCleanup(function, /, *args, **kwargs)

   Add a function to be called after :func:`tearDownModule` to cleanup
   resources used during the test class. Functions will be called in reverse
   order to the order they are added (:abbr:`LIFO (last-in, first-out)`).
   They are called with any arguments and keyword arguments passed into
   :meth:`addModuleCleanup` when they are added.

   If :meth:`setUpModule` fails, meaning that :func:`tearDownModule` is not
   called, then any cleanup functions added will still be called.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2468)

Unknown directive type "function".

```
.. function:: doModuleCleanups()

   This function is called unconditionally after :func:`tearDownModule`, or
   after :func:`setUpModule` if :func:`setUpModule` raises an exception.

   It is responsible for calling all the cleanup functions added by
   :func:`addCleanupModule`. If you need cleanup functions to be called
   *prior* to :func:`tearDownModule` then you can call
   :func:`doModuleCleanups` yourself.

   :func:`doModuleCleanups` pops methods off the stack of cleanup
   functions one at a time, so it can be called at any time.

   .. versionadded:: 3.8
```

Signal Handling

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2486)

Unknown directive type "versionadded".

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

The :option:`-c/--catch <unittest -c>` command-line option to unittest, along with the `catchbreak` parameter to :func:`unittest.main()`, provide more friendly handling of control-C during a test run. With catch break behavior enabled control-C will allow the currently running test to complete, and the test run will then end and report all the results so far. A second control-c will raise a :exc:`KeyboardInterrupt` in the usual way.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2488); *backlink*

Unknown interpreted text role "option".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2488); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2488); *backlink*

Unknown interpreted text role "exc".

The control-c handling signal handler attempts to remain compatible with code or tests that install their own :const:`signal.SIGINT` handler. If the `unittest` handler is called but *isn't* the installed :const:`signal.SIGINT` handler, i.e. it has been replaced by the system under test and delegated to, then it calls the default handler. This will normally be the expected behavior by code that replaces an installed handler and delegates to it. For individual tests that need `unittest` control-c handling disabled the :func:`removeHandler` decorator can be used.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2495); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2495); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2495); *backlink*

Unknown interpreted text role "func".

There are a few utility functions for framework authors to enable control-c handling functionality within test frameworks.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2507)

Unknown directive type "function".

```
.. function:: installHandler()

   Install the control-c handler. When a :const:`signal.SIGINT` is received
   (usually in response to the user pressing control-c) all registered results
   have :meth:`~TestResult.stop` called.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2514)

Unknown directive type "function".

```
.. function:: registerResult(result)

   Register a :class:`TestResult` object for control-c handling. Registering a
   result stores a weak reference to it, so it doesn't prevent the result from
   being garbage collected.

   Registering a :class:`TestResult` object has no side-effects if control-c
   handling is not enabled, so test frameworks can unconditionally register
   all results they create independently of whether or not handling is enabled.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2525)

Unknown directive type "function".

```
.. function:: removeResult(result)

   Remove a registered result. Once a result has been removed then
   :meth:`~TestResult.stop` will no longer be called on that result object in
   response to a control-c.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]unittest.rst, line 2532)

Unknown directive type "function".

```
.. function:: removeHandler(function=None)

   When called without arguments this function removes the control-c handler
   if it has been installed. This function can also be used as a test decorator
   to temporarily remove the handler while the test is being executed::

      @unittest.removeHandler
      def test_signal_handling(self):
          ...
```