

Basic usage

Use the `nosetests` script (after installation by `setuptools`):

```
nosetests [options] [(optional) test files or directories]
```

In addition to passing command-line options, you may also put configuration options in a `.noserc` or `nose.cfg` file in your home directory. These are standard `.ini`-style config files. Put your `nosetests` configuration in a `[nosetests]` section, with the `--` prefix removed:

```
[nosetests]
verbosity=3
with-doctest=1
```

There is also possibility to disable configuration files loading (might be useful when running i.e. `tox` and you don't want your global nose config file to be used by `tox`). In order to ignore those configuration files simply set an environment variable `NOSE_IGNORE_CONFIG_FILES`.

There are several other ways to use the nose test runner besides the `nosetests` script. You may use nose in a test script:

```
import nose
nose.main()
```

If you don't want the test script to exit with 0 on success and 1 on failure (like `unittest.main()`), use `nose.run()` instead:



```
import nose
result = nose.run()
```

`result` will be true if the test run succeeded, or false if any test failed or raised an uncaught exception. Lastly, you can run `nose.core` directly, which will run `nose.main()`:

```
python /path/to/nose/core.py
```

Please see the usage message for the `nosetests` script for information about how to control which tests nose runs, which plugins are loaded, and the test output.

Extended usage

nose collects tests automatically from python source files, directories and packages found in its working directory (which defaults to the current working directory). Any python source file, directory or package that matches the `testMatch` regular expression (by default: `(?:\b|_)[Tt]est`) will be collected as a test (or source for collection of tests). In addition, all other packages found in the working directory will be examined for python source files or directories that match `testMatch`. Package discovery descends all the way down the tree, so `package.tests` and `package`  `v: latest`  all be collected.

Within a test directory or package, any python source file matching `testMatch` will be collected as a test. Within a module, functions and classes whose names match `testMatch` and `TestCase` subclasses will be executed as tests. Tests may use the `assert` keyword or raise `AssertionErrors` to indicate failure. Tests may also use the various `TestCase` methods available.

It is important to note that the default behavior of nose is to not include tests from such files, remove their executable bit or use the `-exe` flag (see `--help`).

Selecting Tests

To specify which tests to run, pass test names on the command line:

```
nosetests only_test_this.py
```

Test names specified may be file or module names, and may optionally indicate the test case name with a colon. Filenames may be relative or absolute. Examples:

Versions

latest

Downloads

PDF HTML Epub

On Read the Docs

Project Home Builds Downloads

On GitHub

View

Search

Hosted by [Read the Docs](#) · [Privacy Policy](#)

```
nosetests test.module
nosetests another.test:TestCase.test_method
nosetests a.test:TestCase
nosetests /path/to/test/file.py:test_function
```

You may also change the working directory where nose looks for tests by using the `-w` switch:

```
nosetests -w /path/to/tests
```

Note, however, that support for multiple `-w` arguments is now deprecated and will be removed in a future release. As of nose 0.10, you can get the same behavior by specifying the target directories *without* the `-w` switch:

```
nosetests /path/to/tests /another/path/to/tests
```

Further customization of test selection and loading is possible through the use of plugins.

Test result output is identical to that of unittest, except for the additional features (error classes, and plugin-supplied features such as output capture and assert introspection) detailed in the options below.

Configuration

In addition to passing command-line options, you may also put configuration options in your project’s `setup.cfg` file, or a `.noserc` or `nose.cfg` file in your home directory. In any of these standard ini-style config files, you put your nosetests configuration in a `[nosetests]` section. Options are the same as on the command line, with the `-` prefix removed. For options that are simple switches, you must supply a value:

```
[nosetests]
verbosity=3
with-doctest=1
```


All configuration files that are found will be loaded and their options combined. You can override the standard config file loading with the `-c` option.

Using Plugins

There are numerous nose plugins available via `easy_install` and elsewhere. To use a plugin, just install it. The plugin will add command line options to nosetests. To verify that the plugin is installed, run:

```
nosetests --plugins
```

You can add `-v` or `-vv` to that command to show more information about each plugin.

If you are running `nose.main()` or `nose.run()` from a script, you can specify a list o  with the `plugins` keyword argument.

v: latest ▾

0.9 plugins

nose 1.0 can use SOME plugins that were written for nose 0.9. The default plugi around 0.9 plugins that adapts the changed plugin api calls. However, plugins tl especially if they attempt to access test case or test suite classes. For example, plu startTest is an individual test or a suite will fail, partly because suites are no longe likely that the plugin is trying to find out if the test is an instance of a class that no

0.10 and 0.11 plugins

All plugins written for nose 0.10 and 0.11 should work with nose 1.0.

Options

-V , --version
Output nose version and exit

Versions

latest

Downloads

PDF HTML Epub

On Read the Docs

Project Home Builds Downloads

On GitHub

View

Search

Search docs

Hosted by [Read the Docs](#) · [Privacy Policy](#)

-p , --plugins

Output list of available plugins and exit. Combine with higher verbosity for greater detail

-v =DEFAULT , --verbose =DEFAULT

Be more verbose. [NOSE_VERBOSE]

--verbosity =VERBOSITY

Set verbosity; `-verbosity=2` is the same as `-v`

-q =DEFAULT , --quiet =DEFAULT

Be less verbose

-c =FILES , --config =FILES

Load configuration from config file(s). May be specified multiple times; in that case, all config files will be loaded and combined

-w =WHERE , --where =WHERE

Look for tests in this directory. May be specified multiple times. The first directory passed will be used as the working directory, in place of the current working directory, which is the default. Others will be added to the list of tests to execute. [NOSE_WHERE]

--py3where =PY3WHERE

Look for tests in this directory under Python 3.x. Functions the same as ‘where’, but only applies if running under Python 3.x or above. Note that, if present under 3.x, this option completely replaces any directories specified with ‘where’, so the ‘where’ option becomes ineffective. [NOSE_PY3WHERE]

-m =REGEX , --match =REGEX , --testmatch =REGEX

Files, directories, function names, and class names that match this regular expression are considered tests. Default: `(?:\b|_)[Tt]est` [NOSE_TESTMATCH]

--tests =NAMES

Run these tests (comma-separated list). This argument is useful mainly from configuration files; on the command line, just pass the tests to run as additional arguments with no switch.

-l =DEFAULT , --debug =DEFAULT

Activate debug logging for one or more systems. Available debug loggers: `nose`, `nose.importer`, `nose.inspector`, `nose.plugins`, `nose.result` and `nose.selector`. Separate multiple names with a comma.

--debug-log =FILE

Log debug messages to this file (default: `sys.stderr`)

--logging-config =FILE , --log-config =FILE

Load logging config from this file – bypasses all other logging config setting

-I =REGEX , --ignore-files =REGEX

Completely ignore any file that matches this regular expression. Takes precedence over `--include-files`. Specifying this option will replace the default setting. Specify this option multiple times for multiple regular expressions [NOSE_IGNORE_FILES]

-e =REGEX , --exclude =REGEX

Don’t run tests that match regular expression [NOSE_EXCLUDE]

-i =REGEX , --include =REGEX

This regular expression will be applied to files, directories, function names, and class names. Specify this option multiple times for multiple regular expressions. Specify this option multiple times for multiple regular expressions [NOSE_INCLUDE]


-x , --stop

Stop running tests after the first error or failure

-P , --no-path-adjustment

Don’t make any changes to `sys.path` when loading tests [NOSE_NOPATH]

--exe


v: latest ▾

Versions

latest

Downloads

PDF HTML Epub

On Read the Docs

Project Home Builds Downloads

On GitHub

View

Search

Search docs

Look for tests in python modules that are executable. Normal behavior is to exclude executable modules, since they may not be import-safe [NOSE_INCLUDE_EXE]

--noexe

DO NOT look for tests in python modules that are executable. (The default on the windows platform is to do so.)

--traverse-namespace

Traverse through all path entries of a namespace package

--first-package-wins , --first-pkg-wins , --1st-pkg-wins

nose's importer will normally evict a package from sys.modules if it sees a package with the same name in a different location. Set this option to disable that behavior.

--no-byte-compile

Prevent nose from byte-compiling the source into .pyc files while nose is scanning for and running tests.

-a=ATTR , --attr=ATTR

Run only tests that have attributes specified by ATTR [NOSE_ATTR]

-A=EXPR , --eval-attr=EXPR

Run only tests for whose attributes the Python expression EXPR evaluates to True [NOSE_EVAL_ATTR]

-s , --nocapture

Don't capture stdout (any stdout output will be printed immediately) [NOSE_NOCAPTURE]

--nologcapture

Disable logging capture plugin. Logging configuration will be left intact. [NOSE_NOLOGCAPTURE]

--logging-format=FORMAT

Specify custom format to print statements. Uses the same format as used by standard logging handlers. [NOSE_LOGFORMAT]

--logging-datefmt=FORMAT

Specify custom date/time format to print statements. Uses the same format as used by standard logging handlers. [NOSE_LOGDATEFMT]

--logging-filter=FILTER

Specify which statements to filter in/out. By default, everything is captured. If the output is too verbose, use this option to filter out needless output. Example: filter=foo will capture statements issued ONLY to foo or foo.what.ever.sub but not foobar or other logger. Specify multiple loggers with comma: filter=foo,bar,baz. If any logger name is prefixed with a minus, eg filter=-foo, it will be excluded rather than included. Default: exclude logging messages from nose itself (-nose). [NOSE_LOGFILTER]

--logging-clear-handlers

Clear all other logging handlers

--logging-level=DEFAULT

Set the log level to capture

--with-coverage

Enable plugin Coverage: Activate a coverage report using [NOSE_WITH_COVERAGE]

--cover-package=PACKAGE

Restrict coverage output to selected packages [NOSE_COVER_PACKAGE]

--cover-erase

Erase previously collected coverage statistics before run

--cover-tests

Include test modules in coverage report [NOSE_COVER_TESTS]

--cover-min-percentage=DEFAULT

Minimum percentage of coverage for tests to pass [NOSE_COVER_MIN_PERCENTAGE]



v: latest ▾

Versions

latest

Downloads

PDF HTML Epub

On Read the Docs

Project Home Builds Downloads

On GitHub

View

Search

Hosted by [Read the Docs](#) · [Privacy Policy](#)

--cover-inclusive

Include all python files under working directory in coverage report. Useful for discovering holes in test coverage if not all files are imported by the test suite. [NOSE_COVER_INCLUSIVE]

--cover-html

Produce HTML coverage information

--cover-html-dir=DIR

Produce HTML coverage information in dir

--cover-branches

Include branch coverage in coverage report [NOSE_COVER_BRANCHES]

--cover-xml

Produce XML coverage information

--cover-xml-file=FILE

Produce XML coverage information in file

--cover-config-file=DEFAULT

Location of coverage config file [NOSE_COVER_CONFIG_FILE]

--cover-no-print

Suppress printing of coverage information

--pdb

Drop into debugger on failures or errors

--pdb-failures

Drop into debugger on failures

--pdb-errors

Drop into debugger on errors

--no-deprecated

Disable special handling of DeprecatedTest exceptions.

--with-doctest

Enable plugin Doctest: Activate doctest plugin to find and run doctests in non-test modules. [NOSE_WITH_DOCTEST]

--doctest-tests

Also look for doctests in test modules. Note that classes, methods and fun doctest tests, not both. [NOSE_DOCTEST_TESTS]

--doctest-extension=EXT

Also look for doctests in files with this extension [NOSE_DOCTEST_EXTE

--doctest-result-variable=VAR

Change the variable name set to the result of the last interpreter command conflicts with the _() function used for text translation. [NOSE_DOCTEST_I

--doctest-fixtures=SUFFIX

Find fixtures for a doctest file in module with this name appended to the base

--doctest-options=OPTIONS

Specify options to pass to doctest. Eg. '+ELLIPSIS,+NORMALIZE_WHITE

--with-isolation

Enable plugin IsolationPlugin: Activate the isolation plugin to isolate cha module or package. The isolation plugin resets the contents of sys.modules a state before the test. PLEASE NOTE that this plugin should not be used with the coverage plugin, or in any other case where module reloading may produce undesirable side-effects. [NOSE_WITH_ISOLATION]

v: latest ▾

Versions

latest

Downloads

PDF HTML Epub

On Read the Docs

Project Home Builds Downloads

On GitHub

View

Search

Search docs

Hosted by [Read the Docs](#) · [Privacy Policy](#)

