# Nose Testing - Tools

The nose.tools module provides a number of testing aids that you may find useful, including decorators for restricting test execution time and testing for exceptions, and all of the same assertX methods found in unittest.TestCase.

- **nose.tools.ok_(expr, msg = None)** − Shorthand for assert.
- **nose.tools.eq_(a, b, msg = None)** − Shorthand for 'assert a == b, "%r != %r" % (a, b)
- **nose.tools.make_decorator(func)** − Wraps a test decorator so as to properly replicate metadata of the decorated function, including nose's additional stuff (namely, setup and teardown).
- **nose.tools.raises(*exceptions)** − Test must raise one of expected exceptions to pass.
- **nose.tools.timed(limit)** − Test must finish within specified time limit to pass
- **nose.tools.istest(func)** − Decorator to mark a function or method as a test
- **nose.tools.nottest(func)** − Decorator to mark a function or method as not a test

## Parameterized Testing

Python's testing framework, unittest, doesn't have a simple way of running parametrized test cases. In other words, you can't easily pass arguments into a **unittest.TestCase** from outside.

However, pytest module ports test parametrization in several well-integrated ways −

- **pytest.fixture()** allows you to define parametrization at the level of fixture functions.
- **@pytest.mark.parametrize** allows to define parametrization at the function or class level. It provides multiple argument/fixture sets for a particular test function or class.
- **pytest_generate_tests** enables implementing your own custom dynamic parametrization scheme or extensions.

A third party module 'nose-parameterized' allows Parameterized testing with any Python test framework. It can be downloaded from this link − https://github.com/wolever/nose-parameterized