Nose Testing - Framework

The nose project was released in 2005, the year after **py.test** received its modern guise. It was written by Jason Pellerin to support the same test idioms that had been pioneered by py.test, but in a package that is easier to install and maintain.

The **nose** module can be installed with the help of pip utility

```
pip install nose
```

This will install the nose module in the current Python distribution as well as a nosetest.exe, which means the test can be run using this utility as well as using –m switch.

```
C:\python>nosetests –v test_sample.py
Or
C:\python>python –m nose test_sample.py
```

nose collects tests from **unittest.TestCase** subclasses, of course. We can also write simple test functions, as well as test classes that are not subclasses of unittest.TestCase. nose also supplies a number of helpful functions for writing timed tests, testing for exceptions, and other common use cases.

nose collects tests automatically. There’s no need to manually collect test cases into test suites. Running tests is responsive, since **nose** begins running tests as soon as the first test module is loaded.

As with the unittest module, **nose** supports fixtures at the package, module, class, and test case level, so expensive initialization can be done as infrequently as possible.

Basic Usage

Let us consider nosetest.py similar to the script used earlier −

```
# content of nosetest.py
def func(x):
   return x + 1

def test_answer():
   assert func(3) == 5
```

In order to run the above test, use the following command line syntax −

```
C:\python>nosetests –v nosetest.py
```

The output displayed on console will be as follows −

```
nosetest.test_answer ... FAIL
================================================================
FAIL: nosetest.test_answer
----------------------------------------------------------------------
Traceback (most recent call last):
   File "C:\Python34\lib\site-packages\nose\case.py", line 198, in runTest
      self.test(*self.arg)
   File "C:\Python34\nosetest.py", line 6, in test_answer
      assert func(3) == 5
AssertionError
----------------------------------------------------------------------
Ran 1 test in 0.000s
FAILED (failures = 1)
```

nose can be integrated with DocTest by using **with-doctest** option in athe bove command line.

```
\nosetests --with-doctest -v nosetest.py
```

You may use **nose** in a test script −

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

If you do not wish 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()
```

The result will be true if the test run is successful, or false if it fails or raises an uncaught exception.

nose supports fixtures (setup and teardown methods) at the package, module, class, and test level. As with py.test or unittest fixtures, setup always runs before any test (or collection of tests for test packages and modules); teardown runs if setup has completed successfully, regardless of the status of the test run.