Python 2.7 Unittest Discovery

TriZPUG April 2012

Chris Calloway

Simple Way to Run Doctests

```
> python -m doctest -h

usage: doctest [-v] file ...

>
```

Simple Way to Run Doctests

```
> python -m doctest myflawlessmodule.py

>
```

Simple Way to Run Doctests

> python -m doctest -v myflawlessmodule.py

```
Trying:
    import ...
Expecting nothing
ok
...
1 items had no tests:
    ...
1 items passed all tests:
   31 tests in list_gliders._main
31 tests in 2 items.
31 passed and 0 failed.
Test passed.
```

>

Not So Simple Way to Run Doctests

```
> python -m doctest myflawlessmodule1.py

> python -m doctest myflawlessmodule2.py

> python -m doctest myflawlessmodule3.py

> python -m doctest myflawlessmodule4.py

> python -m doctest myflawlessmodule5.py

>
```

New in Python 2.7

```
> python -m unittest -h
Usage: python -m unittest [options] [tests]

Options:
  -h, --help       Show this message
  -v, --verbose    Verbose output
  -q, --quiet      Minimal output
  -f, --failfast   Stop on first failure
  -c, --catch      Catch control-C and display results
  -b, --buffer     Buffer stdout and stderr during test runs

Examples:
  python -m unittest test_module               - run tests from test_module
  python -m unittest module.TestClass          - run tests from module.TestClass
  python -m unittest module.Class.test_method  - run specified test method

[tests] can be a list of any number of test modules, classes and test
methods.
```

New in Python 2.7

```
Alternative Usage: python -m unittest discover [options]

Options:
  -v, --verbose    Verbose output
  -f, --failfast   Stop on first failure
  -c, --catch      Catch control-C and display results
  -b, --buffer     Buffer stdout and stderr during test runs
  -s directory     Directory to start discovery ('.' default)
  -p pattern       Pattern to match test files ('test*.py' default)
  -t directory     Top level directory of project (default to
                   start directory)

For test discovery all test modules must be importable from the top
level directory of the project.
>
```

The load_tests Protocol

- 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 `TestLoader.loadTestsFromModule()` with the following arguments:

```
load_tests(loader, standard_tests,
           pattern="test*.py")
```

- `load_tests` should return a `TestSuite`.

The load_tests Protocol

- *loader* is the instance of `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 *pattern* argument is used when loading packages as part of test discovery.

The load_tests Protocol

- If discovery is started, either from the command line or by calling `TestLoader.discover()`, with a pattern that matches a package name then the package `__init__.py` will be checked for `load_tests`.

- If the package `__init__.py` defines `load_tests` then it will be called and discovery not continued into the package. `load_tests` is called.

Simple Way to Run Doctests

```
> python -m unittest discover -p "tests"

...........

----------------------------------------------------------------------
Ran 11 tests in 0.285s

>
```

Unittest Discovery

- http://docs.python.org/library/unittest.html#load-tests-protocol