14.5 忽略或期望测试失败¶

问题¶

你想在单元测试中忽略或标记某些测试会按照预期运行失败。

解决方案¶

`unittest` 模块有装饰器可用来控制对指定测试方法的处理，例如：

```
import unittest
import os
import platform

class Tests(unittest.TestCase):
    def test_0(self):
        self.assertTrue(True)

    @unittest.skip('skipped test')
    def test_1(self):
        self.fail('should have failed!')

    @unittest.skipIf(os.name=='posix', 'Not supported on Unix')
    def test_2(self):
        import winreg

    @unittest.skipUnless(platform.system() == 'Darwin', 'Mac specific test')
    def test_3(self):
        self.assertTrue(True)

    @unittest.expectedFailure
    def test_4(self):
        self.assertEqual(2+2, 5)

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

如果你在Mac上运行这段代码，你会得到如下输出：

```
bash % python3 testsample.py -v
test_0 (__main__.Tests) ... ok
test_1 (__main__.Tests) ... skipped 'skipped test'
test_2 (__main__.Tests) ... skipped 'Not supported on Unix'
test_3 (__main__.Tests) ... ok
test_4 (__main__.Tests) ... expected failure

----------------------------------------------------------------------
Ran 5 tests in 0.002s

OK (skipped=2, expected failures=1)
```

讨论¶

`skip()` 装饰器能被用来忽略某个你不想运行的测试。 `skipIf()` 和 `skipUnless()` 对于你只想在某个特定平台或Python版本或其他依赖成立时才运行测试的时候非常有用。使用 `@expected` 的失败装饰器来标记那些确定会失败的测试，并且对这些

其他依赖成立时才进行测试的时候非常有用。使用 @expected 的失败装饰器来标记那些确定会失败的测试，并且对这些测试你不想让测试框架打印更多信息。

忽略方法的装饰器还可以被用来装饰整个测试类，比如：

```
@unittest.skipUnless(platform.system() == 'Darwin', 'Mac specific tests')
class DarwinTests(unittest.TestCase):
    pass
```