- 大纲
- Python教程
  - Python入门
  - 基础教程
  - 第一个Python程序
  - Python语法
  - 数据类型
  - 变量
  - 字符串和编码
  - 使用list和tuple
  - 条件判断
  - 模式匹配
  - 循环
  - 使用dict和set
  - 函数的调用
  - 函数定义
  - 函数的参数
  - 递归
  - 切片
  - 迭代
  - 列表生成式
  - 生成器
  - 迭代器
  - 函数式编程
  - 模块
  - 面向对象编程
  - 面向对象高级编程
  - 错误处理
  - 调试
  - 单元测试
  - 文档测试
  - 进程和线程
  - 分布式进程
  - 正则表达式
  - 常用内建模块
  - 常用第三方模块
  - virtualenv
  - 图片处理
  - 处理文本
  - 处理数据库
  - 网络编程
  - 电子邮件
  - 网络服务器
  - Web开发
  - 异步IO
  - 实时Web
  - 部署到生产

关于作者

廖雪峰 | 关注 | 1 | 0 | 0 | 0 | 0 | 0

测试题

作者：廖雪峰

如果你刚刚学习了Python的语法（下载Python Development，单元测试模块），

那么现在是检验学习成果的时候了——不要担心这一个测试题会难倒你，这个测试题一定会让你有成就感！

比如你可能会问：如何计算斐波拉契数列的第100个数？

答：斐波拉契数列（Fibonacci sequence）是一个非常简单的递归数列，除了最开始的两个数字1和1，其余的数字可以通过最开始的两个数字一直加到最后一个数字得到。

斐波拉契数列的开始数字是：1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

要计算斐波拉契数列的第100个数字，我们可以先写一个函数来实现：

```
def fibonacci(n):
    a = 1
    b = 1
    for i in range(1, n):
        a, b = b, a + b
    return a
```

fibonacci(100)

可得到结果：

```
354224848179261915075
```

现在，我们来写一个测试模块来测试这个函数是否正确。

```
class MyTestCase(unittest.TestCase):
    def test_one(self):
        self.assertEqual(1, fibonacci(1))
        self.assertEqual(1, fibonacci(2))
        self.assertEqual(2, fibonacci(3))
        self.assertEqual(3, fibonacci(4))
        self.assertEqual(5, fibonacci(5))
    def test_hundred(self):
        self.assertEqual(354224848179261915075, fibonacci(100))
    def test_negative(self):
        self.assertRaises(ValueError, fibonacci, -1)
    def test_zero(self):
        self.assertRaises(ValueError, fibonacci, 0)
    def test_type(self):
        self.assertRaises(TypeError, fibonacci, '1')
    def test_overflow(self):
        self.assertRaises(OverflowError, fibonacci, 1000)
    def test_bigdata(self):
        self.assertEqual(573147844013817084101, fibonacci(100))
    def test_speed(self):
        import time
        t = time.clock()
        fibonacci(100)
        t = time.clock() - t
        print('fibonacci(100) cost %f seconds.' % t)
        self.assertLessThan(t, 0.1)
if __name__ == '__main__':
    unittest.main()
```

▶ 运行

小结

单元测试可以在简单地验证某个程序的操作是否正确——甚至在修改程序以后也能验证程序仍然正确。

单元测试还能让程序设计和实现更有信心，比如，这样的程序设计和实现——

先写测试，然后实现功能，最后不断地运行测试，直到测试全部通过。

单元测试能帮助开发者把精力放在更有趣的事情上——比如如何改进程序的设计和实现，而不是把时间浪费在手工去一遍一遍地测试一个个的小功能。

单元测试是一个开发小组能够持续地保证代码质量（即代码能够通过测试）的最佳方法。

单元测试是一个开发小组能够持续地保证代码质量（即代码能够通过测试）的最佳方法。

参考源码

model.zip

model_test.zip

测试用例可以操作查看模块，请在终端中直接运行测试：

```
./model.py
./model_test.py
```

还可以分别测试：

```
./model.py test
./model_test.py test
```

▶ 运行测试

← 上一页 | 下一页 →

python免费公开课
倒计时 2天
讲师解读：公司高薪offer揭秘，分享最新招聘市场现状工具箱
参加直播

python免费公开课
倒计时 2天
讲师解读：公司高薪offer揭秘，分享最新招聘市场现状工具箱
参加直播

评论

小白
Steven Tsai posted on 2017-12-12 11:11:11

```
class TestMyDict(unittest.TestCase):
    def test_init(self):
        d = MyDict(a=1, b=2)
        self.assertEqual(d.a, 1)
        self.assertEqual(d.b, 2)
        d = MyDict({'x': 1})
        self.assertEqual(d.x, 1)
    def test_value(self):
        d = MyDict()
        d.x = 1
        self.assertEqual(d.x, 1)
    def test_key_error(self):
        d = MyDict()
        with self.assertRaisesRegex(KeyError, 'key: foo'):
            d['foo']
    def test_attr_error(self):
        d = MyDict()
        with self.assertRaisesRegex(AttributeError, 'MyDict object has no attribute: bar'):
            d.bar
    def test_keys(self):
        d = MyDict(a=1, b=2)
        self.assertEqual(d.keys(), ['a', 'b'])
    def test_values(self):
        d = MyDict(a=1, b=2)
        self.assertEqual(d.values(), [1, 2])
    def test_items(self):
        d = MyDict(a=1, b=2)
        self.assertEqual(d.items(), [('a', 1), ('b', 2)])
    def test_empty(self):
        d = MyDict()
        self.assertEqual(len(d.keys()), 0)
if __name__ == '__main__':
    unittest.main()
```

▶ 运行

小结

单元测试可以在简单地验证某个程序的操作是否正确——甚至在修改程序以后也能验证程序仍然正确。

单元测试还能让程序设计和实现更有信心，比如，这样的程序设计和实现——

先写测试，然后实现功能，最后不断地运行测试，直到测试全部通过。

单元测试能帮助开发者把精力放在更有趣的事情上——比如如何改进程序的设计和实现，而不是把时间浪费在手工去一遍一遍地测试一个个的小功能。

单元测试是一个开发小组能够持续地保证代码质量（即代码能够通过测试）的最佳方法。

参考源码

model.zip

model_test.zip

测试用例可以操作查看模块，请在终端中直接运行测试：

```
./model.py
./model_test.py
```

还可以分别测试：

```
./model.py test
./model_test.py test
```

▶ 运行测试

回复 | 点赞

Kevin
Kevin 回复 Steven Tsai posted on 2017-12-12 11:11:11

test

回复 | 点赞

阿的哥哥
Kevin 回复 Kevin posted on 2017-12-12 11:11:11

```
def get_grade(score):
    if score < 0 or score > 100:
        raise ValueError('bad score: %s' % score)
    if score >= 90:
        return 'A'
    if score >= 60:
        return 'B'
    return 'C'
if __name__ == '__main__':
    print('Test get_grade() ...')
    assert get_grade(100) == 'A', get_grade(100)
    assert get_grade(90) == 'A', get_grade(90)
    assert get_grade(89) == 'B', get_grade(89)
    assert get_grade(60) == 'B', get_grade(60)
    assert get_grade(59) == 'C', get_grade(59)
    assert get_grade(0) == 'C', get_grade(0)
    print('Test get_grade() ok.')
    print('Test get_grade() ...')
    try:
        get_grade(-1)
        raise AssertionError('bad score: %s' % score)
    except ValueError:
        pass
    try:
        get_grade(101)
        raise AssertionError('bad score: %s' % score)
    except ValueError:
        pass
    print('Test get_grade() ok.')
```

回复 | 点赞

测试
Kevin posted on 2017-12-12 11:11:11

```
class MyTestCase(unittest.TestCase):
    def test_init(self):
        d = MyDict(a=1, b=2)
        self.assertEqual(d.a, 1)
        self.assertEqual(d.b, 2)
        d = MyDict({'x': 1})
        self.assertEqual(d.x, 1)
    def test_value(self):
        d = MyDict()
        d.x = 1
        self.assertEqual(d.x, 1)
    def test_key_error(self):
        d = MyDict()
        with self.assertRaisesRegex(KeyError, 'key: foo'):
            d['foo']
    def test_attr_error(self):
        d = MyDict()
        with self.assertRaisesRegex(AttributeError, 'MyDict object has no attribute: bar'):
            d.bar
    def test_keys(self):
        d = MyDict(a=1, b=2)
        self.assertEqual(d.keys(), ['a', 'b'])
    def test_values(self):
        d = MyDict(a=1, b=2)
        self.assertEqual(d.values(), [1, 2])
    def test_items(self):
        d = MyDict(a=1, b=2)
        self.assertEqual(d.items(), [('a', 1), ('b', 2)])
    def test_empty(self):
        d = MyDict()
        self.assertEqual(len(d.keys()), 0)
if __name__ == '__main__':
    unittest.main()
```

回复 | 点赞

Kevin
Kevin posted on 2017-12-12 11:11:11

```
class TestMyDict(unittest.TestCase):
    def test_init(self):
        d = MyDict(a=1, b=2)
        self.assertEqual(d.a, 1)
        self.assertEqual(d.b, 2)
        d = MyDict({'x': 1})
        self.assertEqual(d.x, 1)
    def test_value(self):
        d = MyDict()
        d.x = 1
        self.assertEqual(d.x, 1)
    def test_key_error(self):
        d = MyDict()
        with self.assertRaisesRegex(KeyError, 'key: foo'):
            d['foo']
    def test_attr_error(self):
        d = MyDict()
        with self.assertRaisesRegex(AttributeError, 'MyDict object has no attribute: bar'):
            d.bar
    def test_keys(self):
        d = MyDict(a=1, b=2)
        self.assertEqual(d.keys(), ['a', 'b'])
    def test_values(self):
        d = MyDict(a=1, b=2)
        self.assertEqual(d.values(), [1, 2])
    def test_items(self):
        d = MyDict(a=1, b=2)
        self.assertEqual(d.items(), [('a', 1), ('b', 2)])
    def test_empty(self):
        d = MyDict()
        self.assertEqual(len(d.keys()), 0)
if __name__ == '__main__':
    unittest.main()
```

回复 | 点赞

测试