哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

Lab 5：测试用例设计与单元测试

实验目标

- 按Lab1和Lab2的分组，两人共同完成实验，针对自己在Lab1中完成的程序；
- 针对Lab1开发的服务组合算法，设计测试用例
- 在PyUnit环境下撰写测试代码并执行测试
  - 使用Pyunit进行一般Python程序单元测试
- 使用Coverage.py统计测试的覆盖度

针对Lab 1：设计黑盒测试用例

- **利用黑盒测试的等价类划分和边界值分析方法，为Lab1待测程序设计一组测试用例(输入数据+期望输出)。**
  - 测试对象：主程序，其输入为3个TXT文件(SERVICE.TXT、 PROCESS.TXT、REQ.TXT)；
  - 测试用例的**输入部分**为1个文件：SERVICE.TXT，需要根据测试用例修改；其他两个文件(PROCESS.TXT、REQ.TXT)无需改动，直接使用Lab5中提供的两个文本文件（**注：不是Lab1的**）；
  - 测试用例的**期望输出**部分为1个文件：RESULT.TXT，其中包含的数据由学生根据输入的SERVICE.TXT手工计算得到。

针对Lab 1：设计黑盒测试用例

- **在Lab1实验要求基础上，对输入文件SERVICE.TXT中包含的数据做如下额外限定：**
  - 每个活动的候选服务数目在2到4之间；
  - 全部候选服务的数目不低于11个；
  - 活动的第二个QoS指标(可靠性)小于1，小数点之后的位数为1或2，最后一位不能为0；
  - 活动的第四个QoS指标(价格)的取值范围为[3,7)，最多2位小数，也可为整数。
- **对Lab1实验要求的输出文件格式做如下简化：**
  - (A-1,B-20), (A-1,D-35), (B-20,C-102), (D-35,E-290), (C-102,F-7), (E-290,F-7), Reliability=XXX,Cost=YYY,Q=ZZZ
  - 无需输出算法的执行时间。

针对Lab 1：设计白盒测试用例

- 任选你的Lab1中的一个独立函数，为其设计白盒测试用例。
- 所选函数应满足以下条件：
  - 不低于40行代码；
  - 至少包含1个循环；
  - 至少包含3个判定；
  - 不能是用于从文件中读取数据和向文件写入结果的函数。
- 采用基本路径方法设计测试用例；
- 测试用例由输入数据和期望的输出数据构成。

Pyunit

- A standard part in Python.
- Beyond Python 1.5.2x and up.

```
C:\>python
Python 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on wi
n32
Type "help", "copyright", "credits" or "license" for more information.
>>> import unittest
>>>
```

- PyUnit 使用参考：http://www.oschina.net/question/12_27127

Test Case

- 单元测试是由一些测试用例（Test Cases）构建组成的。在PyUnit中，**unittest**模块中的**TestCase** 类代表测试用例。
- **通过覆盖runTest方法即可得到最简单的测试用例子类以运行某些代码**

```
1  import unittest
2
3  class DefaultWidgetSizeTestCase(unittest.TestCase):
4      def runTest(self):
5          widget = Widget("The widget")
6          assert widget.size() == (50,50), 'incorrect default size'
```

- **“assert” phrase is used for test 通过断言来判断是否通过测试**
- **testCase = DefaultWidgetSizeTestCase() 实例化测试用例**

Combine Test Cases(1)

- 采用**setUp()**方法来初始化多个测试用例的共性内容，设置测试环境

```
1  import unittest
2
3  class SimpleWidgetTestCase(unittest.TestCase):
4      def setUp(self):
5      self.widget = Widget("The widget")
6
7  class DefaultWidgetSizeTestCase(SimpleWidgetTestCase):
8      def runTest(self):
9      assert self.widget.size() == (50,50), 'incorrect default size'
10
11 class WidgetResizeTestCase(SimpleWidgetTestCase):
12     def runTest(self):
13     self.widget.resize(100,150)
14     assert self.widget.size() == (100,150), \
15            'wrong size after resize'
```

- 用**tearDown()**方法进行测试后的环境清除工作

```
1  import unittest
2
3  class SimpleWidgetTestCase(unittest.TestCase):
4      def setUp(self):
5      self.widget = Widget("The widget")
6      def tearDown(self):
7      self.widget.dispose()
8      self.widget = None
```

Combine Test Cases(2)

- 包含多个测试用例的类

```
import unittest


class WidgetTestCase(unittest.TestCase):
    def setUp(self):
        self.widget = Widget("The widget")
    def tearDown(self):
        self.widget.dispose()
        self.widget = None
    def testDefaultSize(self):
        assert self.widget.size() == (50,50), 'incorrect default size'
    def testResize(self):
        self.widget.resize(100,150)
        assert self.widget.size() == (100,150), \
               'wrong size after resize'
```

```
defaultSizeTestCase = WidgetTestCase("testDefaultSize")
resizeTestCase = WidgetTestCase("testResize")
```

Testsuite

- 将多个测试用例合成测试套件 **TestSuite**

```
widgetTestSuite = unittest.TestSuite()
widgetTestSuite.addTest(WidgetTestCase("testDefaultSize"))
widgetTestSuite.addTest(WidgetTestCase("testResize"))
```

```
def suite():
    suite = unittest.TestSuite()
    suite.addTest(WidgetTestCase("testDefaultSize"))
    suite.addTest(WidgetTestCase("testResize"))
    return suite
```

```
suite = unittest.makeSuite(WidgetTestCase,'test')
```

从命令行运行测试

- **python unittest.py widgettests.WidgetTestSuite**

Coverage.py

- **Coverage.py is a tool for measuring code coverage of Python programs. It monitors your program, noting which parts of the code have been executed, then analyzes the source to identify code that could have been executed but was not.**
- **Coverage measurement is typically used to gauge the effectiveness of tests. It can show which parts of your code are being exercised by tests, and which are not.**

- 网站：http://coverage.readthedocs.org/en/latest/
- 安装：**pip install coverage**

Coverage Test

- **Run**
  - coverage run my_program.py arg1 arg2
  - Generate a file
- **report**
  - coverage report
- **Html**
  - coverage html -d covhtml

```
$ coverage report
Name                      Stmts   Exec  Cover
---------------------------------------------
my_program                   20     16    80%
my_module                    15     13    86%
my_other_module              56     50    89%
---------------------------------------------
TOTAL                        91     79    87%
```

实验评判标准

- 是否可正确配置Pyunit、coverage.py；
- 所设计的测试用例是否满足前述要求；
- 是否能够针对不同测试用例完成覆盖度分析；
- 是否正确书写了Pyunit测试代码并执行，以获得测试结果。

提交方式

- 请遵循实验报告模板撰写。
- 设计测试用例之前不能修改**Lab1**的代码。

- 提交日期：第**11**周周四晚**(11**月**26**日 **23:55)**
- 提交一个文件到**CMS**：
  - 实验报告：命名规则“学号-Lab5-report.doc”，具体请参见模板
  - 同组的两人提交一份即可。

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

结束