霍格沃兹测试学院

用Python做自动化测试

--unittest/pytest

目录

单元测试的概念

单元测试覆盖率

Unittest/pytest框架

unittest 的编写规范

pytest优势和编写规范

pytest实例

pytest几个主要插件

Pytest 高级用法

unittest/pytest项目实战

单元测试的概念

霍格沃兹测试学院

- 什么是单元测试?
  - 单元测试的价值
  - 单元测试的难点
- unittest和pytest 都是python单元测试框架；
- 目前国内大厂单元测试的现状；
- 应用unittest/pytest于自动化测试.

单元测试的覆盖率

霍格沃兹测试学院

- 语句覆盖（Statement Coverage）
- 判断覆盖（Decision Coverage）
- 条件覆盖（Condition Coverage）
- 路径覆盖 （Path Coverage）

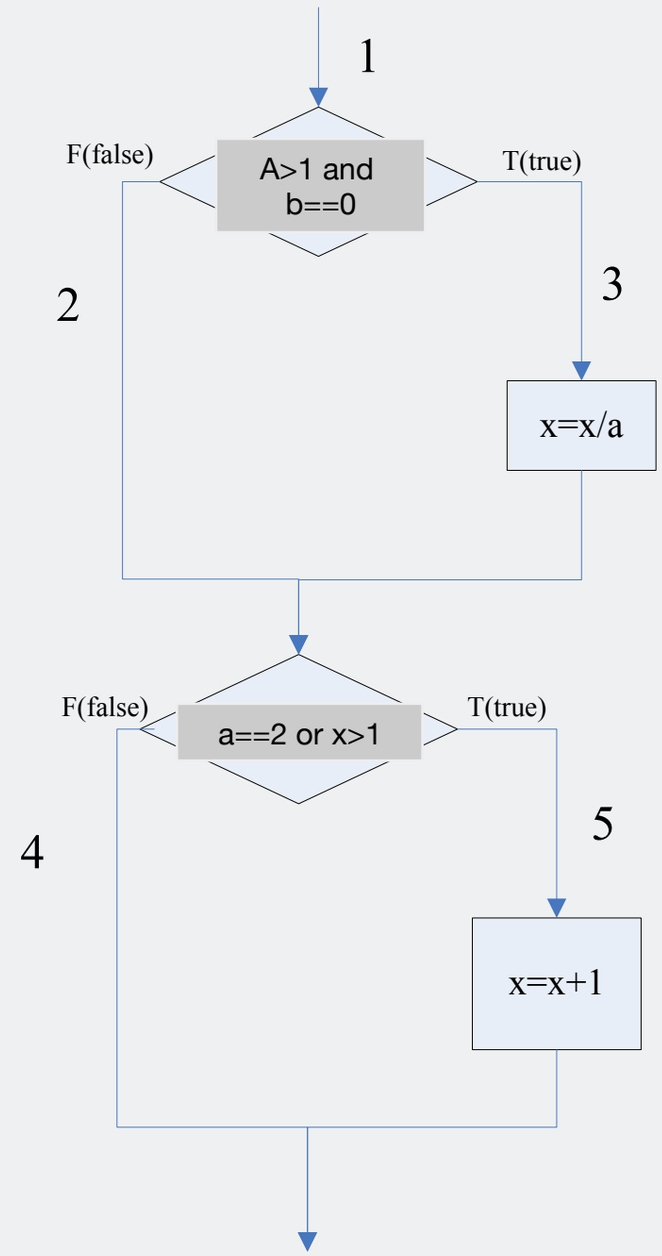
1.代码覆盖率也被用于自动化测试和手工测试，来度量测试是否全面的指标之一
2. 应用覆盖率的思想增强测试用例的设计

被测试函数

```
def demo_method(a, b, x):

    if (a > 1 and b == 0):
        x= x/a

if (a == 2 or x > 1):
    x = x+1
return x
```



语句覆盖

- 语句覆盖定义：
  - 运行测试用例的过程被击中的代码行即称为被覆盖的语句
- 测试用例：
  - a=3, b=0,x=3
- 漏洞：
  - and -> or

判断覆盖

霍格沃兹测试学院

- 判断覆盖定义：
  - 运行测试用例的过程被击中的判定语句
- 测试用例：

Test Cases	a b x	(a>1)&&(b==0)	a==2||x>1	ExecutePath
Case1	2 0 3	T	T	135
Case2	1 0 1	F	F	124
Case3	3 0 3	T	F	134
Case4	2 1 1	F	T	125

- 漏洞：
  - a==2 or x>1 ->a==2 or x<1

条件覆盖

- 条件覆盖定义：
  - 每一个判断覆盖语句中，每个可能的子条件或者合并条件都覆盖到
- 测试用例：if (a > 1 and b == 0)

Test Cases	**a>1**	**b==0**
Case1	T	T
Case2	T	F
Case3	F	T
Case4	F	F

- 缺陷：
  - 测试用例指数级增加（2**conditions）

路径覆盖

- 路径覆盖定义：
  - 每一个可能的路径都覆盖
- 测试用例：

Test Cases	a b x	ExecutePath
Case1	2 0 3	135
Case2	1 0 1	124
Case3	3 0 3	134
Case4	2 1 1	125

应用这些方法设计测试用例！！！

Unittest 与编写规范

霍格沃兹测试学院

- Unittest是python内嵌的测试框架，原名为PyUnit
- Unittest提供了test cases、test suites、test fixtures、test runner相关的组件
- 编写规范
  - 测试模块首先 import unittest
  - 测试类必须继承 unittest.TestCase
  - 测试方法必须以“test_”开头
  - 模块名字，类名没有要求

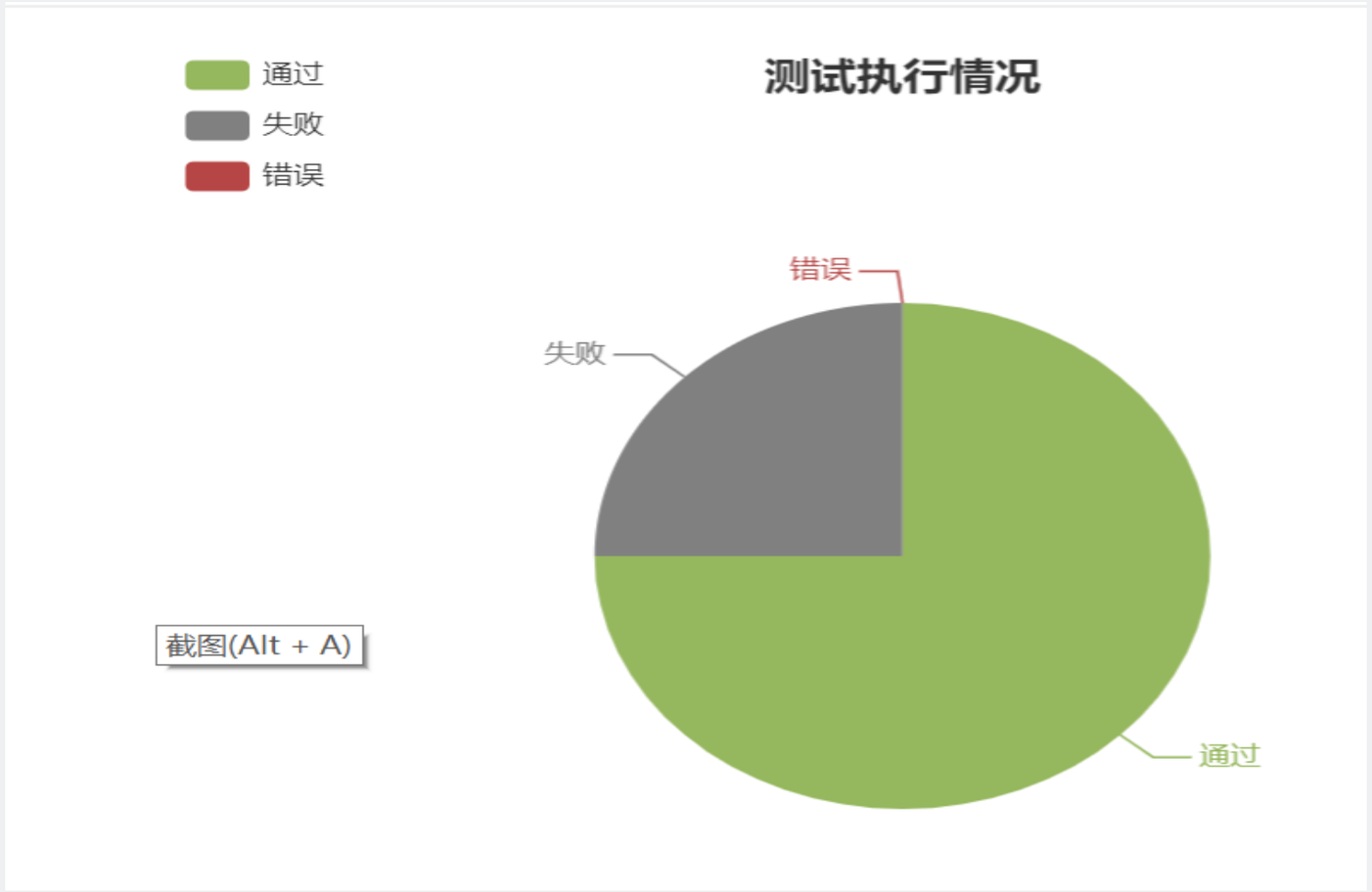
setUp, tearDown

- 基于测试方法级别的setUp, tearDown
  - 执行每个测试方法的时候都会执行一次setUp 和tearDown
- 基于类级别的 setUpClass, tearDownClass
  - 执行这个类里面的所有测试方法只有一次执行setup, tearDown
- 基于模块级别的 setUpModule，tearDownModule
  - 执行此模块里的所有类里的测试方法，只执行一次setup和teardown

setUp, tearDown，测试报告

霍格沃兹测试学院

- HtmlTestrunner 生成测试报告



Pytest优势

- 简单灵活，像写Python代码一样写测试用例；
- 为测试方法输入不同参数化；
- 自动重试失败的测试用例；
- 支持allure2 测试报告；
- 具有很多第三方插件(http://plugincompat.herokuapp.com/,目前已有565个插件)，并且可以自定义扩展；

Pytest编写规范

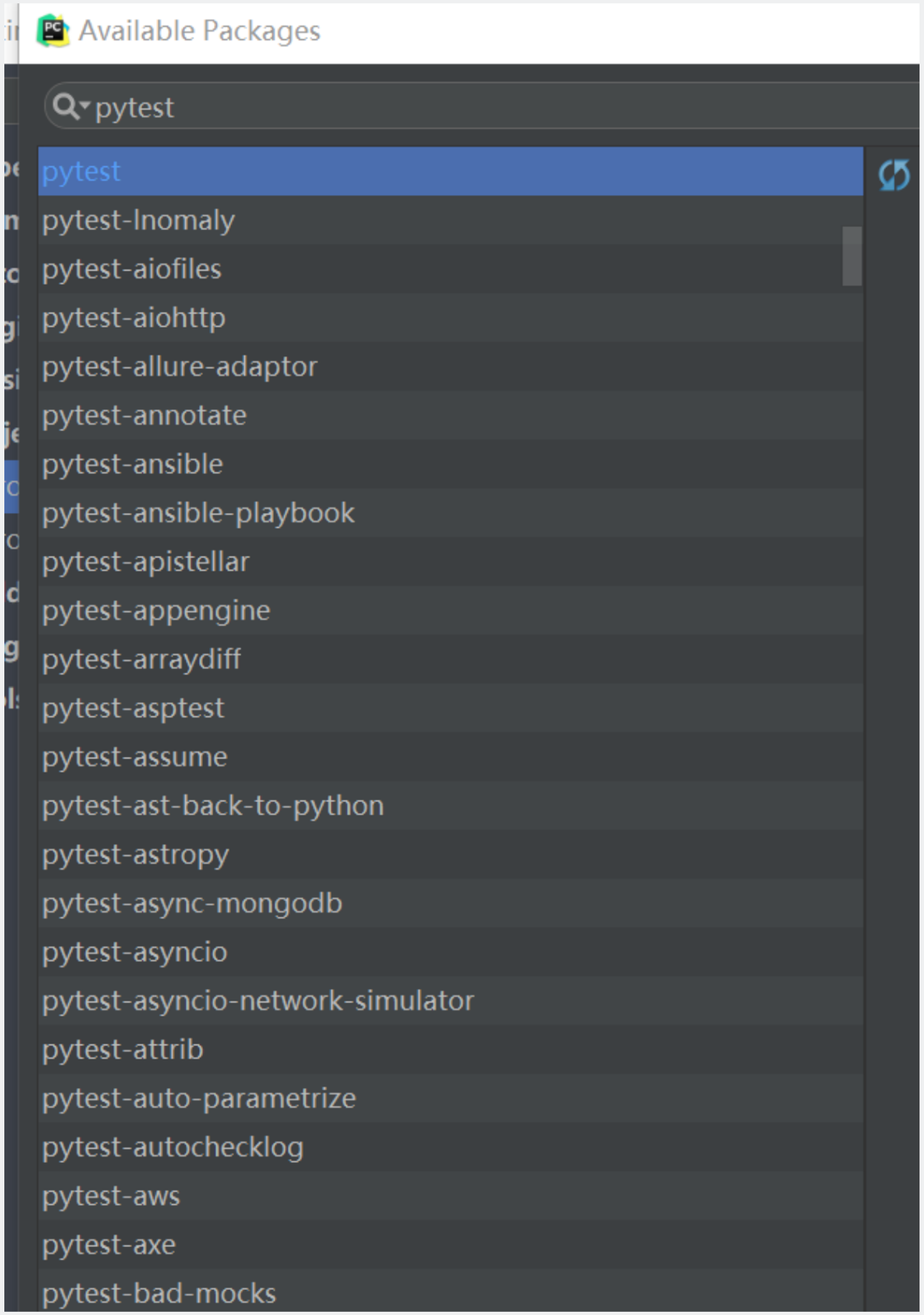
- 编写规范：
  - 测试文件以 test_ 开头（以 _test 结尾也可以）
  - 测试类以 Test 开头，并且不能带有 __init__ 方法
  - 测试函数以 test_ 开头

Pytest是最好的Python自动化测试框架！

没有之一！！！

Pytest第三方组件

霍格沃兹测试学院



- pytest-sugar
- pytest-assume
- pytest-ordering
- pytest-selenium
- pytest-play
- pytest-rerunfailures
- pytest-allure
- pytest-datadir
- pytest-datafiles

Pytest实例讲解

- **pytest: 3.8.0**
- 运行 api>pytest test_pyexample_1.py

```
#test_pyexample_1.py
def add(x,y):
    return x+y
def test_add():
    assert add(1, 2) == 3
def test_add2():
    print("I am 2")
    assert add(1.2, 3.1) == 4.3
```

Pytest-参数化

- 测试场景：
  - 测试登录成功，登录失败（账号错误，密码错误）
  - 创建多种账号: 中文账号，英文账号
- Copy多份代码 or 读入参数?
- 一次性执行多个输入参数
- pytest.mark.parametrize

```
#test_parameters.py
import pytest
@pytest.mark.parametrize("x,y", [
    (3+5, 8),
    (2+4, 6),
    (6*9, 42),
])
def test_add(x, y):
    assert x == y
```

Pytest-多个Assert

- 测试场景：
  - 一个测试用例中有多个数据需要比较；

- 一个个比较 or 放到一个数据结构里面全量比较?
  - 全量比较，无法知道那个值不对
  - 一个个比较数值，第一个失败就退出

```
#test_pyexample_1.py
import time
def add(x,y):
    return x+y
def test_add():
    assert add(1, 2) == 3
def test_add2():
    print("I am 2")
    time.sleep(3)
    assert  add(1.2, 3.1) == 5.3
    assert add(1,2) == 3
```

Pytest-rerunfails

霍格沃兹测试学院

- 测试场景：
  - 在web、APP自动化测试中，经常出现超时导致测试失败
- 加成等待时间 or 重新执行?

```
#test_pyexample_rerun.py
import random
def add(x,y):
    return x+y
def test_add():
    assert add(1, 2) == 3
def test_add2():
    random_value =
random.randint(2,5)
print("random_value:"+str(random
_value))
    assert  random_value== 3
```

Pytest-ordering

- 测试场景：
  - 在web测试中，上下测试用例页面切换有依赖关系；
  - 在修改信息的页面中，依赖于前面用例已经创建好的信息，比如修改账号信息，依赖于已经创建好的数据

```
#test_order.py
import time
import pytest
value=0
def test_add2():
    print("I am 2")
    time.sleep(2)
    assert value == 10
def test_add():
    global value
    value =10
    assert value ==10
```

Pytest 练习


```
#pytest_practice.py

#! /usr/bin/env python

#coding=utf-8

import random

def bubble_sort(nums):
    for i in range(len(nums)-1):
        for j in range(len(nums)-i-1):
            if nums[j] > nums[j+1]:
                nums[j], nums[j+1] = nums[j+1], nums[j]
    return random.choice([nums,None,10])
```

Pytest高级用法-自定义marks

霍格沃兹测试学院

- 测试场景：
  - 在大厂里面，一个系统的测试用例好几千个，每次发版都执行，来不及；
  - 通过关键字来运行测试用例

    pytest –k，缺陷--难于维护、无法多重打标
  - 通过标签来运行测试用例，即可以作用于函数也可以作用于类

```
#pytest.ini
[pytest]
markers =
    webtest:  Run the webtest case
    hello: Run the hello case
    P0:  run P0 cases
    P1:   Run P1 cases
```

Pytest高级用法-fixtures

霍格沃兹测试学院

- 什么是fixtures：
  - 类似于unittest里的setUp, tearDown；
  - Pytest fixtures调用起来比unittest 的setUp, tearDown方便
  - 支持不同级别的fixtures(session, class, function)
  - 调用更灵活，支持直接函数名调用、decorator调用、autouse

Pytest fixtures三种调用方法

- 什么是fixtures
  - fixture就是为了测试用例的执行，而初始化一些数据和方法
  - 实现了unittest里面的setUp, tearDown功能，但比setUp, tearDown更加灵活

- 直接通过函数名字调用
- 使用usefixtures decorateor
- 使用autouse

Pytest fixtures--名字调用

- yield 之前相当于setup
- yield 之后相当于tearDown

```
#test_fixturebyname.py
import pytest
@pytest.fixture()
def loginandlogout():
    print('do login action\n')
    yield
    print("do logout action")
class TestSample:
    def test_answer(self,
loginandlogout):
        ....
def test_answer2(self,
loginandlogout):
```

Pytest fixtures-decorator调用

霍格沃兹测试学院

- @pytest.mark.usefixtures ('loginandlogout’)
- Loginandlogout是fixture函数名

```
#test_decorator.py
import pytest
@pytest.fixture()
def loginandlogout():
    print('do login action\n')
    yield
    print("do logout action")
class TestSample:

@pytest.mark.usefixtures('loginandlogout')
 def test_answer(self):
        ....
```

Pytest fixtures--autouse

霍格沃兹测试学院

- Fixture scope
  - module
  - session
  - class
  - package

```
#test_autouse.py
@pytest.fixture(scope='module',autouse=True)
def loginandlogout():
    print('do login action\n')
    yield
    print("do logout action\n")
@pytest.fixture(scope='class',autouse=True)
def clickhome():
    print('click home button\n')
    yield
    print("end click home  link\n")
class TestSample:
    def test_answer(self):
    def test_answer2(self):
class TestSampleTwo:
    def test_two_answer(self):
```

Pytest高级用法-Conftest.py

- conftest.py 文件中定义共享的fixture;
- conftest.py文件一般放在testcases目录下，每个子目录下也可以存在conftest.py
- 如果子目录下有conftest.py,子目录下的conftest.py中的

fixture优先

pytest 执行测试用例的方式（1）

霍格沃兹测试学院

- 执行一个module
  - pytest -v src/testcases/api/test_autouse.py
- 执行一个类，一个方法
  - pytest -v src/testcases/api/test_autouse.py::TestSample
  - pytest -v src/testcases/api/

    test_autouse.py::TestSample::test_answer
- 执行一个目录或者package
  - pytest -v src/testcases/api
- 通过标签来运行测试用例
  - pytest -m P0 src/testcases/api/

pytest 执行测试用例的方式（2）

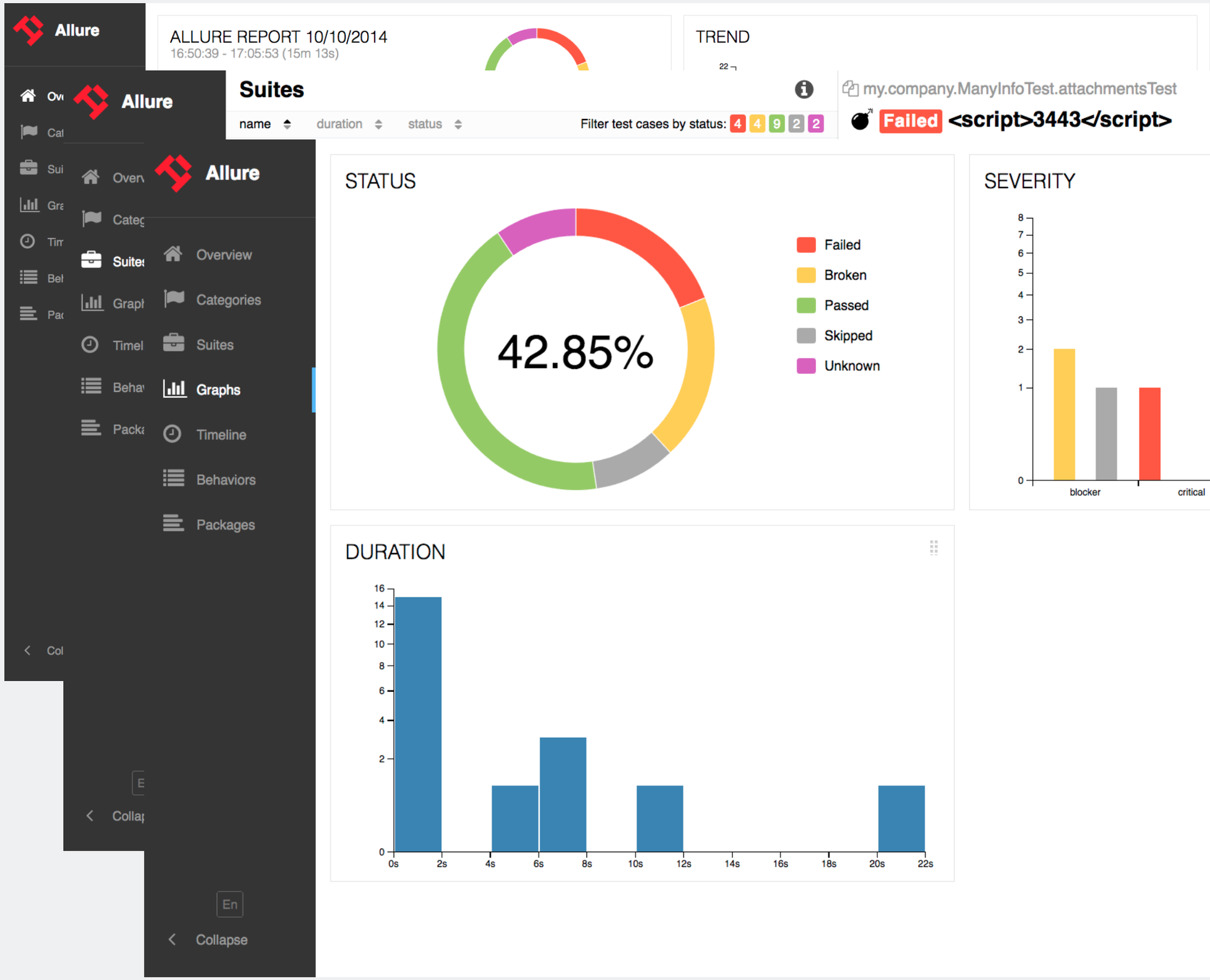
- 通过pytest.main来执行，所有的参数和pytest命令行方式是一样的
    - pytest.main(['-v', '--instafail', 'testcases/api/test_example.py', '-m=P2'])

unitest pytest项目实战

- https://github.com/lxneng/xpinyin
- 作业
  - 使用pytest实现对get_pinyin, get_initial, get_initials的单元测试
  - 三个方法的行覆盖率要达到80%（什么工具？）

Pytest report

霍格沃兹测试学院



休息时间