# python实例

**Unittest**

1. 概念

python的标准的单元测试框架，有独立的测试报告框架。

1.test fixture

用于初始化、清理等动作。

2.testcase

测试用例，unittest的最小单元。用以对指定输入的返回结果进行检测。在unittest中提供 了TestCase基类， 用来创建新的测试用例类。

3.test suite

测试套件，一系列测试用例或测试套件的集合。在unittest中由TestSuite类实现。

4.test runner

测试执行器，负责用例执行并生成测试报告，在unittest中提供了命令行模式和GUI模式来执行。

unittest的流程：写好TestCase，然后由TestLoader加载TestCase到TestSuite，然后由TextTestRunner来运行 TestSuite，运行的结果保存在TextTestResult中，我们通过命令行或者unittest.main()执行时，main会调用 TextTestRunner中的run来执行

1. 【简单实例】测试一个加法的函数。 hello.py

import unittest

# 定义被测函数

def add(a, b):

return a + b

# 测试用例 方法必须test开头（demoTest继承unittest.Testcase）

class demoTest(unittest.TestCase):

def test\_add(self):

self.assertEquals(add(4,5),9);

def test\_add2(self):

self.assertEquals(add(5,5),11)

def test\_add3(self):

self.assertEquals(add(6,6),12)

# 当.py文件被直接运行时，if \_\_name\_\_ == '\_\_main\_\_'之下的代码块将被运行；当.py文件以模块形式被导入时， if \_\_name\_\_ == '\_\_main\_\_'之下的代码块不被运行。

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

1. 使用TestSuite 套件 （注意 from \*\* import \*\*\*） test\_1.py

import unittest

from hello import demoTest

if \_\_name\_\_ == '\_\_main\_\_':

suite = unittest.TestSuite() #先创建一个suite实例

tests = [demoTest("test\_add"), demoTest("test\_add2"), demoTest("test\_add3")]

suite.addTests(tests)

#verbosity 0 是简单报告、1 是一般报告、2 是详细报告。

runner = unittest.TextTestRunner(verbosity=1)

runner.run(suite)

1. 测试结果写入文件 test\_2.py

import unittest

from hello import demoTest

if \_\_name\_\_ == '\_\_main\_\_':

suite = unittest.TestSuite()

suite.addTests(unittest.TestLoader().loadTestsFromTestCase(demoTest))

with open('UnittestTextReport.txt', 'a') as f:

runner = unittest.TextTestRunner(stream=f, verbosity=2)

runner.run(suite)

1. 测试 test fixture 比如执行前需要连接数据库（准备环境）、执行完成之后需要还原数据、断开连接（清理环境）

class TestMathFunc(unittest.TestCase):

"""Test mathfuc.py"""

#每次case前，后都会运行一遍

def setUp(self):

print "do something before test.Prepare environment."

def tearDown(self):

print "do something after test.Clean up."

如果只想执行一次,装饰器

@classmethod

def setUpClass(cls):

print "This setUpClass() method only called once."

@classmethod

def tearDownClass(cls):

print "This tearDownClass() method only called once too."

跳过某个case

@unittest.skip("I don't want to run this case.")

def test\_divide(self):

1. 输出HTML报告 下载HTMLTestRunner.py，并放到当前目录下，或者Python27\Lib下

import unittest

from hello import demoTest

from HTMLTestRunner import HTMLTestRunner

if \_\_name\_\_ == '\_\_main\_\_':

suite = unittest.TestSuite()

tests = [demoTest("test\_add"), demoTest("test\_add2"), demoTest("test\_add3")]

suite.addTests(tests)

with open('HTMLReport.html', 'w') as f:

runner = HTMLTestRunner(stream=f,

title='MathFunc Test Report',

description='generated by HTMLTestRunner.',

verbosity=2

)

runner.run(suite)

1. 新的小例子

import unittest

class Converter(object):

def \_\_init\_\_(self, str):

self.mystr = str

def convert(self):

self.mystr = self.mystr[::-1] #反转字符串

return self.mystr

class ConverterTest(unittest.TestCase):

def test\_convert1(self):

value = Converter("abcd")

self.assertEquals("dcba", value.convert())

def test\_convert2(self):

value = Converter("hello world!")

self.assertEquals("!dlrow olleh", value.convert())

if \_\_name\_\_ == "\_\_main\_\_":

suite = unittest.TestSuite()

suite.addTest(unittest.TestLoader().loadTestsFromTestCase(ConverterTest))

runner = unittest.TextTestRunner(verbosity=2)

runner.run(suite)

# 测试 爱壁纸app （appium）

import os

import unittest

from appium import webdriver

from time import sleep

#设置路径信息

PATH = lambda p: os.path.abspath(

os.path.join(os.path.dirname(\_\_file\_\_), p)

)

class LoginAndroidTests(unittest.TestCase):

def setUp(self):

#初始化测试平台

desired\_caps = {}

desired\_caps['platformName'] = 'Android'

desired\_caps['platformVersion'] = '6.0.9.0' #Android版本

desired\_caps['deviceName'] = '127.0.0.1:62001'#告诉appium 夜神的地址

desired\_caps['app'] = 'D:\\bz.apk'#app的安装路径

desired\_caps['appPackage'] = 'com.lovebizhi.wallpaper'#包名

desired\_caps['appActivity'] = 'com.adesk.picasso.view.MainActivity'#activity名称

self.driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub', desired\_caps) #连接appium 4723是appium

def tearDown(self):

self.driver.quit()

def test(self):

print "start test..."

#判断是否安装爱壁纸APP

wallpaper = self.driver.is\_app\_installed("com.lovebizhi.wallpaper")

if wallpaper:

#点击主页

self.driver.find\_element\_by\_id("com.lovebizhi.wallpaper:id/nav\_home\_ll").click()

#点击某一壁纸图片，注意这里是elements list

self.driver.find\_elements\_by\_id("com.lovebizhi.wallpaper:id/thumb")[8].click()

# 点击设置壁纸

self.driver.find\_element\_by\_id("com.lovebizhi.wallpaper:id/set\_wp\_btn").click()

self.driver.find\_element\_by\_id("com.lovebizhi.wallpaper:id/set\_wp\_btn").click()

else:

self.driver.install\_app("D:\\bz.apk")

sleep(30)

if \_\_name\_\_ == '\_\_main\_\_':

suite =unittest.TestLoader().loadTestsFromTestCase(LoginAndroidTests)

unittest.TextTestRunner(verbosity=2).run(suite)

**测试 猫宁考勤app （uiautomator）**

from uiautomator import device as d

import time

import unittest

class MyTestSuite(unittest.TestCase):

# 初始化工作

def setUp(self):

d.press.home()

time.sleep(3)

d(text="猫宁考勤").click()

print "--------------初始化工作"

# 退出清理工作

def tearDown(self):

d.press.back()

print "--------------退出清理工作"

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 判断控件是否存在 & text

def check\_controls\_exists(self, controls\_text):

if d(text=controls\_text).exists:

return 1

else:

return 0

# 判断按钮是否置灰 & text & clickable

def check\_controls\_click\_text(self, controls\_text):

if d(text=controls\_text).info.get("clickable") is True:

return 1

else:

return 0

#assertIn(a, b) a in b

def check\_ainb(self,resourceid,b):

#读出resourceid中的文本

if d(resourceId=resourceid).info.get("text") in b:

return 1

else:

return 0

def check\_controls\_click\_resourceId(self,resourceidj):

#读出resourceid中的文本

if d(resourceId=resourceid).info.get("checked") is True:

return 1

else:

return 0

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

def test\_A\_1(self):

print "-----start=》test\_A\_1"

self.assertEqual(self.check\_controls\_click\_text("注册"),1,"目前页面：猫宁考勤开启全新时代")

d(text="注册").click()

time.sleep(3)

#注册猫宁界面 比较字符要加u ，unicide （该语句比较前两个输入，最后那个只是输出错误信息）

self.assertEqual(self.check\_ainb("com.isentech.attendance:id/regis\_phone",u"请输入手机号码"),

1,"注册页面-》请输入手机号码")

self.assertEqual(self.check\_ainb("com.isentech.attendance:id/regis\_verifycode",u"请输入验证码"),

1,"注册页面-》请输入验证码")

#没有输入手机的情况下，获取验证码的按钮应该不能按

self.assertEqual(self.check\_controls\_click\_text("获取验证码"), 0,"注册页面-》获取验证码")

self.assertEqual(self.check\_controls\_click\_text("《中科爱讯服务协议》"), 1,"注册页面-》《中科爱讯服务协议》")

self.assertEqual(self.check\_controls\_click\_text("注册"), 0,u"注册页面-》注册")

#《中科爱讯服务协议》

d(text="《中科爱讯服务协议》").click()

self.assertEqual(self.check\_ainb("com.isentech.attendance:id/title",u"服务协议"), 1)

d(resourceId="com.isentech.attendance:id/title\_back").click()

time.sleep(1)

#手机号不输入是否能注册

d(text="注册").click()

#只输入5个数字是否能获取验证码

d(resourceId="com.isentech.attendance:id/regis\_phone").set\_text("11111")

self.assertEqual(self.check\_controls\_click\_text("获取验证码"), 0)

time.sleep(1)

d(resourceId="com.isentech.attendance:id/regis\_phone").clear\_text()

time.sleep(1)

#输入正确的验证码&获取验证码是否高亮

d(resourceId="com.isentech.attendance:id/regis\_phone").set\_text("13811112222")

d(resourceId="com.isentech.attendance:id/regis\_verifycode").set\_text("5648")

self.assertEqual(self.check\_controls\_click\_text("获取验证码"), 1)

time.sleep(3)

d(resourceId="com.isentech.attendance:id/regis\_phone").clear\_text()

d(resourceId="com.isentech.attendance:id/regis\_verifycode").clear\_text()

time.sleep(1)

#输入正确的密码是否能注册&我已同意是否打钩

d(resourceId="com.isentech.attendance:id/regis\_phone").set\_text("13811112222")

d(resourceId="com.isentech.attendance:id/regis\_verifycode").set\_text("5648")

d(resourceId="com.isentech.attendance:id/regis\_pass").set\_text("123456")

d(resourceId="com.isentech.attendance:id/regis\_passAgain").set\_text("123456")

time.sleep(1)

self.assertEqual(self.check\_controls\_click\_resourceId("com.isentech.attendance:id/regis\_agree"), 1)

self.assertEqual(self.check\_controls\_click\_text("注册"), 1)

#def test\_A\_2(self):

# print "-----start=》test\_A\_2"

if \_\_name\_\_ == "\_\_main\_\_":

# 测试app

unittest.main()

**接口测试**

1.简单基础

# 导入了 Flask 类。这个类的实例将会是我们的 WSGI 应用程序。

from flask import Flask

#创建一个该类的实例,自己

app = Flask(\_\_name\_\_)

#使用 route() 装饰器告诉 Flask 什么样的URL 能触发我们的函数

@app.route('/')

@app.route('/')

def index():

return 'Index Page'

@app.route('/hello')

def hello():

return 'Hello World'

#确保服务器只会在该脚本被 Python 解释器直接执行的时候才会运行，而不是作为模块导入的时候。按 Ctrl+C关闭服务器

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

访问 127.0.0.1:5000 返回Index Page，127.0.0.1:5000/hello 返回Hello World

【这个可以画图】2.起一个flask服务{}

#!/usr/bin/python

# coding=utf-8

from flask import Flask, request, session, jsonify

USERNAME = 'admin'

PASSWORD = '123456'

app = Flask(\_\_name\_\_)

app.secret\_key = 'pithy'

@app.route('/login', methods=['POST'])

def login():

error = None

if request.method == 'POST':

if request.form['username'] != USERNAME:

error = 'Invalid username'

elif request.form['password'] != PASSWORD:

error = 'Invalid password'

else:

session['logged\_in'] = True

return jsonify({'code': 200, 'msg': 'success'})

return jsonify({'code': 401, 'msg': error}), 401

@app.route('/info', methods=['get'])

def info():

if not session.get('logged\_in'):

return jsonify({'code': 401, 'msg': 'please login !!'})

return jsonify({'code': 200, 'msg': 'success', 'data': 'info'})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

1. 测试代码——1【这个可以画图】

import requests

import unittest

#测试的是服务器的接口 post get

class TestLogin(unittest.TestCase):

@classmethod #装饰器 只执行一次

def setUpClass(cls):

cls.login\_url = 'http://127.0.0.1:5000/login'

cls.info\_url = 'http://127.0.0.1:5000/info'

cls.username = 'admin'

cls.password = '123456'

def test\_login(self):

#测试登录

data = {

'username': self.username,

'password': self.password

}

response = requests.post(self.login\_url, data=data).json()

#登录成功， 接着可以测试info（），此时的session['logged\_in'] = True

assert response['code'] == 200

assert response['msg'] == 'success'

if response['code'] == 200 and response['msg'] == 'success':

print "login\_ok!"

def test\_info(self):

#测试info接口

data = {

'username': self.username,

'password': self.password

}

response\_cookies = requests.post(self.login\_url, data=data).cookies

#会话

session = response\_cookies.get('session')

assert session

info\_cookies = {

'session': session

}

response = requests.get(self.info\_url, cookies=info\_cookies).json()

assert response['code'] == 200

assert response['msg'] == 'success'

assert response['data'] == 'info'

if \_\_name\_\_ == '\_\_main\_\_':

suite =unittest.TestLoader().loadTestsFromTestCase(TestLogin)

unittest.TextTestRunner(verbosity=2).run(suite)

1. 优化后的 测试代码 调用封装成了一个实例方法，实现了复用。

import requests

import unittest

try:

from urlparse import urljoin

except ImportError:

from urllib.parse import urljoin

class DemoApi(object):

def \_\_init\_\_(self, base\_url):

self.base\_url = base\_url

def login(self, username, password):

#登录接口

url = urljoin(self.base\_url, 'login')

data = {

'username': username,

'password': password

}

return requests.post(url, data=data).json()

def get\_cookies(self, username, password):

#获取登录cookies

url = urljoin(self.base\_url, 'login')

data = {

'username': username,

'password': password

}

return requests.post(url, data=data).cookies

def info(self, cookies):

#详情接口

url = urljoin(self.base\_url, 'info')

return requests.get(url, cookies=cookies).json()

class TestLogin(unittest.TestCase):

@classmethod

def setUpClass(cls):

cls.base\_url = 'http://127.0.0.1:5000'

cls.username = 'admin'

cls.password = '123456'

cls.app = DemoApi(cls.base\_url)

def test\_login(self):

#测试登录

response = self.app.login(self.username, self.password)

assert response['code'] == 200

assert response['msg'] == 'success'

def test\_info(self):

#测试获取详情信息

cookies = self.app.get\_cookies(self.username, self.password)

response = self.app.info(cookies)

assert response['code'] == 200

assert response['msg'] == 'success'

assert response['data'] == 'info'

if \_\_name\_\_ == '\_\_main\_\_':

suite =unittest.TestLoader().loadTestsFromTestCase(TestLogin)

unittest.TextTestRunner(verbosity=2).run(suite)

**复杂接口 测试有道翻译**

#“有道”这个接口并不固定 会随着时间推移而不能用 可以再baidu。 百度翻译要注册开发员， 固定ip 做不到。

import requests

import hashlib

import time

import json

#接口可能做了更新 貌似只要这2个数据 post 就能运行

def createData(transStr):

data = {

'i': transStr ,

'doctype': 'json',

}

return data

#接口是translate

url = "http://fanyi.youdao.com/translate"

r = requests.request("post", url, params=createData("handsome boy"))

d = json.loads(r.text)

#解析 r.text {"type":"EN2ZH\_CN","errorCode":0,"elapsedTime":0,"translateResult":[[{"src":"handsome boy","tgt":"帅哥"}]]}

str = d['translateResult'][0][0]['tgt']

print(str)

**Mysql**

import pymysql

conn = pymysql.connect(host='127.0.0.1', port=3306, user='root', passwd='abc', db='samp\_db', charset='utf8')

cursor = conn.cursor()

conn.close()

cursor.execute('create database if not exists samp\_db')

sql = """CREATE TABLE EMPLOYEE (

FIRST\_NAME CHAR(20) NOT NULL,

LAST\_NAME CHAR(20),

AGE INT,

SEX CHAR(1),

INCOME FLOAT )"""

cursor.execute(sql)

**Selenium测试网页应用**

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

from selenium import webdriver

import os

import time

from time import sleep

chromedriver = "C:\Program Files (x86)\Google\Chrome\Application\chromedriver.exe"

driver = webdriver.Chrome(chromedriver)

url = "http://www.baidu.com"

driver.get(url)

sleep(1)

# 方法一

try:

assert u"百度一下" in driver.title

print ('Assertion baidu title pass.')

except Exception as e:

print ('Assertion baidu title fail.', format(e))

sleep(1)

#校验通过，百度一下按钮存在

try:

driver.find\_element\_by\_xpath("//\*[@id='su']")

print "校验通过，百度一下按钮存在"

except NoSuchElementException:

assert 0, "校验不通过"

#实际是 百度一下 你就知道

if u"今日头条\_百度搜索" == driver.title :

print ('Assertion dayevenery title pass.')

else:

print ('Assertion dayevenery title fail.')

def isElementExist(element):

flag=True

try:

driver.find\_element\_by\_css\_selector(element)

return flag

except:

flag=False

return flag

#判断页面上有无id为kw的元素

if isElementExist("#kw") :

driver.find\_element\_by\_id("kw").send\_keys("Python")

if isElementExist("#su") :

driver.find\_element\_by\_id("su").click()

sleep(1)

def ClickRefresh():

#执行1-》3次

for i in range(1,4):

ISOTIMEFORMAT="%Y-%m-%d %X"

strTime = time.strftime( ISOTIMEFORMAT, time.localtime())

print u'正在执行第 ',i,"次 ...",strTime

#刷新浏览器

time.sleep(2)

driver.refresh()

time.sleep(2)

print

print u"已执行完第 ",i,"次"

print

ClickRefresh()

# 浏览器全屏显示,出错原因 需要更新谷歌浏览器驱动到最新版本即可。

#driver.maximize\_window()

#关闭浏览器

#browser.quit()

**Monkey**

1. 测试爱壁纸

import sys

from com.android.monkeyrunner

import MonkeyRunner, MonkeyDevice, MonkeyImage

def call(d):

d.startActivity(component="com.lovebizhi.wallpaper/com.adesk.picasso.view.MainActivity")

print "Start Activity"

def main():

print "Start"

device = MonkeyRunner.waitForConnection()

if not device:

print "Couldn't get connection"

sys.exit()

print "Found device"

call(device)

if \_\_name\_\_ == '\_\_main\_\_':

main()

2.测试百度

因为是按照坐标点击的测试，所以随着网页的改变，位置也会有所不同(可以用UIviewer看大概位置)

import sys

from com.android.monkeyrunner import MonkeyRunner, MonkeyDevice, MonkeyImage

def browse(d):

d.startActivity(uri='http://www.baidu.com',component="com.android.browser/.BrowserActivity")

MonkeyRunner.sleep(4.0)

d.touch(150, 400, 'DOWN\_AND\_UP');

MonkeyRunner.sleep(1.0)

d.type("monkey")

MonkeyRunner.sleep(1.0)

d.touch(980, 200, 'DOWN\_AND\_UP');

MonkeyRunner.sleep(10.0)

d.press('KEYCODE\_HOME', MonkeyDevice.DOWN\_AND\_UP)

def main():

print "Start"

device = MonkeyRunner.waitForConnection()

if not device:

print "Couldn't get connection"

sys.exit()

print "Found device"

browse(device)

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Sample class**

class Employee():

def \_\_init\_\_(self, id, name, income):

self.id = id

self.name = name

self.income = income

def info(self):

print "编号：",emp\_instance.id," ","姓名：",emp\_instance.name," ","收入：",emp\_instance.income

emp\_instance = Employee(1001,"何婷",9000)

emp\_instance.info()

emp\_instance.id = 1002

emp\_instance.name = "何小婷"

emp\_instance.income = 8888

emp\_instance.info()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**编译windows下可执行文件**

pip install pyinstaller

cmd到.py文件目录下

pyinstaller \*\*.py

可执行文件在新生成的 dist中

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**clock 获得两次运行之间的间隔**

t\_start = clock()

t\_end = clock()

**命令行调用iris**

from subprocess import Popen, PIPE, STDOUT, run

dict = {}

dict["address"] = "cosmosaccaddr1m5xm249kpr6j2keyw8wykmllxvru4y7d3hhqnz"

dict["name"] = "testName"

dict["chain-id"] = "test"

dict["amount"] = "10iris"

dict["password"] = "1234567890"+"\n"

def send(param):

send\_faucet = "iriscli.exe send --to={0} --name={1} --chain-id={2} --amount={3} --home=C:\.iriscli".format(param["address"], param["name"], param["chain-id"], param["amount"])

print(send\_faucet)

p = run(send\_faucet, shell=True, stdout=PIPE, input=param["password"].encode())

print(p.stdout)

if \_\_name\_\_ == '\_\_main\_\_':

send(dict)

print("2222")

print(param["password"])

print("333333")

print(param["password"].encode())

=========================================

**线程的使用**

import threading

import time

count = 0

def getCount():

while 1:

start = count

time.sleep(1)

print(count-start)

if \_\_name\_\_ == "\_\_main\_\_":

try:

t1 = threading.Thread(target=getCount,args=())

t1.start()

except:

print()

while 1:

count+=1

=========================================

**pycharm 中如果出现跨文件引用出问题**

右键 project》标记目录为》根源

或者 放到python path中

=========================================

**钉钉python**

Python 实现：

headers = {

'content-type': 'application/json',

}

data = {

"msgtype": "text",

"text": {

"content": "脚本测试数据"

},

}

url = r"https://oapi.dingtalk.com/robot/send?access\_token=99315e8be01b593660314a9decddd53f2bb2b38d5df48e9256a6ca3d3048c6dc"

r = requests.request("post", url, headers=headers, data =json.dumps(data))

<https://oapi.dingtalk.com/robot/send?access_token=f0070a7a8b1e2f3a81d8fba88d360b3a664fb6322980b28aaa2b1f271e556c1d>

=========================================

**Python 一键锁屏打包（exe）**

from ctypes import \*

print('HELLO WORLD!')

user32 = windll.LoadLibrary('user32.dll')

user32.LockWorkStation()

# pyinstaller -F my\_test.py --noconsole

# 生成快捷方式， 右键exe