# 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. 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)

**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

**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)

**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()

**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

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

**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