6/7/2021 🡪 6/22/2021

9/3/21

1/13/22

1. Get

import requests  
  
'''  
 URL Parameters Request method: URL parameter  
 For example: request http://httpbin.org/get?first\_name=hello&last\_name=word  
'''  
  
params = {"first\_name": "hello", "last\_name": "word"}  
responds = requests.get("http://httpbin.org/get", params=params)  
print(responds.text)  
print(responds.url)

1. Post

import requests  
  
params = {"first\_name": "hello", "last\_name": "word"}  
headers = {"Content-Tpye": "application/x-www-form-urlencoded"}  
responds = requests.post("http://httpbin.org/post", data=params, headers=headers)  
print(responds.text)  
print(responds.url)  
print(responds.request)

1. Json post

import requests  
  
'''  
 URL Parameters Request method: URL parameter  
 For example: request http://httpbin.org/get?first\_name=hello&last\_name=word in a POST method  
'''  
  
params = {"first\_name": "hello", "last\_name": "word"}  
headers = {"Content-Tpye": "application/json"}  
responds = requests.post("http://httpbin.org/post", json=params, headers=headers)  
print(responds.text)  
print(responds.url)  
print(responds.request)

1. **Packaging Request Method**

* Directly call a general function

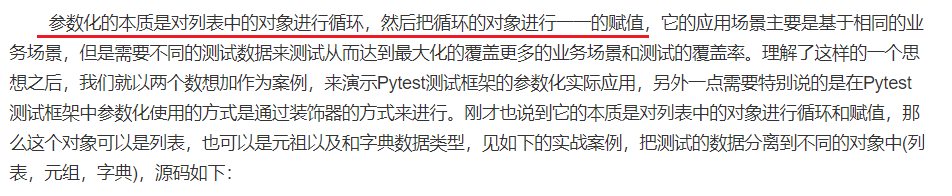
import requests  
import json  
  
  
class ApiRequest(object):  
  
 # ----- The first request method package the Request library, the call can be used according to the actual situation  
 def send\_requests(self, method, url, data=None, params=None, headers=None, cookies=None, json=None, files=None, timeout=None):  
 self.r = requests.request(method, url, data=data, params=params, headers=headers, cookies=cookies, json=json,  
 files=files, timeout=timeout)  
 return self.r

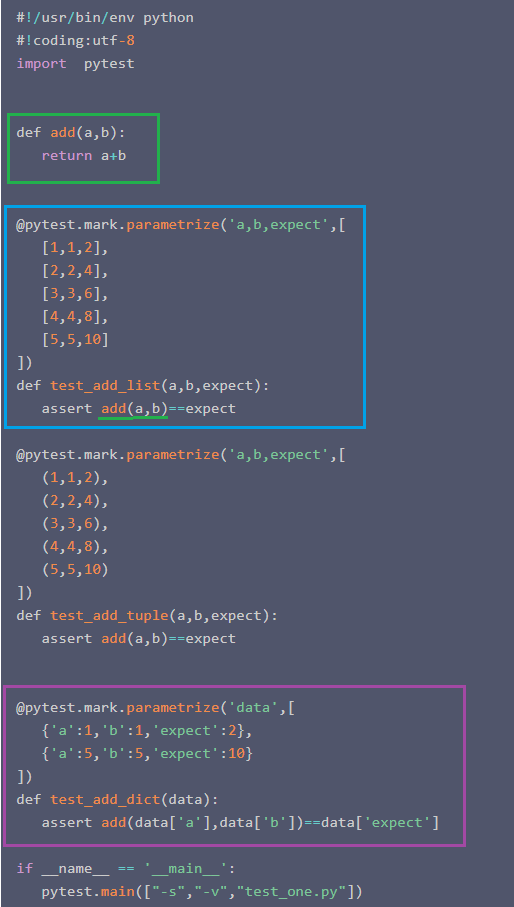
* Call individual method function through a filter/gen function

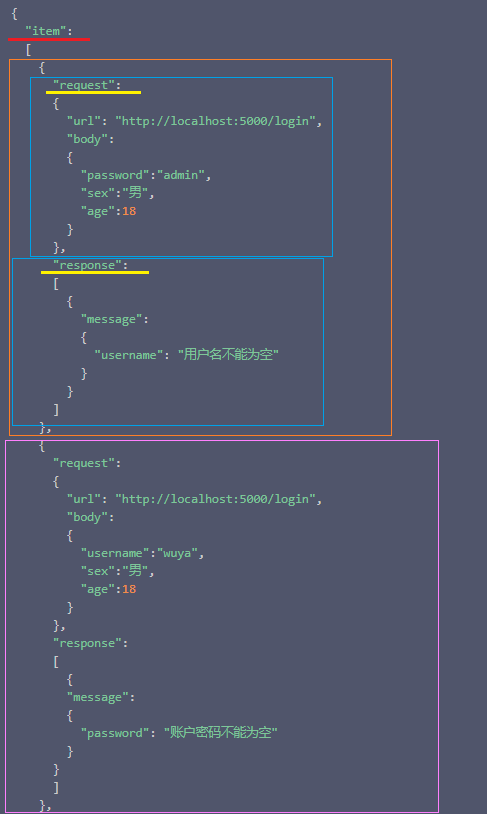
import requests  
import json  
  
  
class ApiRequest(object):  
  
 def get(self, url, data=None, headers=None):  
 if headers is not None:  
 res = requests.get(url=url, data=data, headers=headers)  
 else:  
 res = requests.get(url=url)  
 return res.json()  
  
 "" "POST request" ""  
  
  
def post(self, url, data, headers):  
 if headers is not None:  
 res = requests.post(url=url, data=data, headers=headers)  
 else:  
 res = requests.post(url=url, data=data)  
 if str(res) == "<Response [200]>":  
 return res.json()  
 else:  
 return res.text()  
  
 "" "" "" "" ""  
  
  
def all\_method(self, method, url, data=None, headers=None):  
 if method == 'get' or method == 'GET':  
 res = self.get(url, data, headers)  
 elif method == 'post' or method == 'POST':  
 res = self.post(url, data, headers)  
 elif method == 'put' or method == 'PUT':  
 res = self.post(url, data, headers)  
 elif method == 'delete' or method == 'DELETE':  
 res = self.delete(url, data, headers)  
 else:  
 Res = 'request mode is incorrect'  
 return json.dump(res, ensure\_ascii=False, indent=4, sort\_keys=True, separators=(',', ':'))

# Pytest实战

<https://cloud.tencent.com/developer/article/1730701>

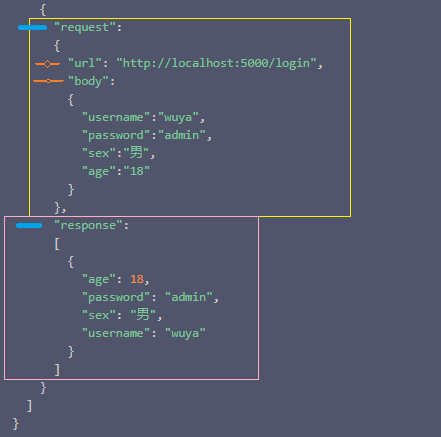




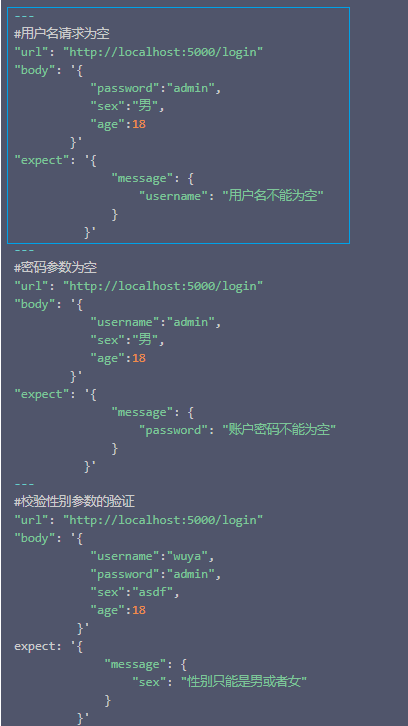


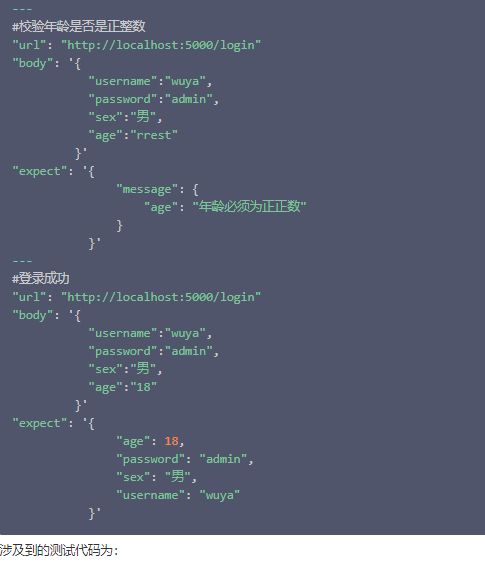
























# Pytest接口自动化测试框架搭建模板

<https://blog.csdn.net/VXadmin/article/details/112966633?utm_medium=distribute.pc_relevant.none-task-blog-baidujs_baidulandingword-4&spm=1001.2101.3001.4242>















# Python+unittest+requests 接口自动化测试框架搭建 完整的框架搭建过程 实战

<https://blog.csdn.net/songlh1234/article/details/84317617?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-4.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-4.control>



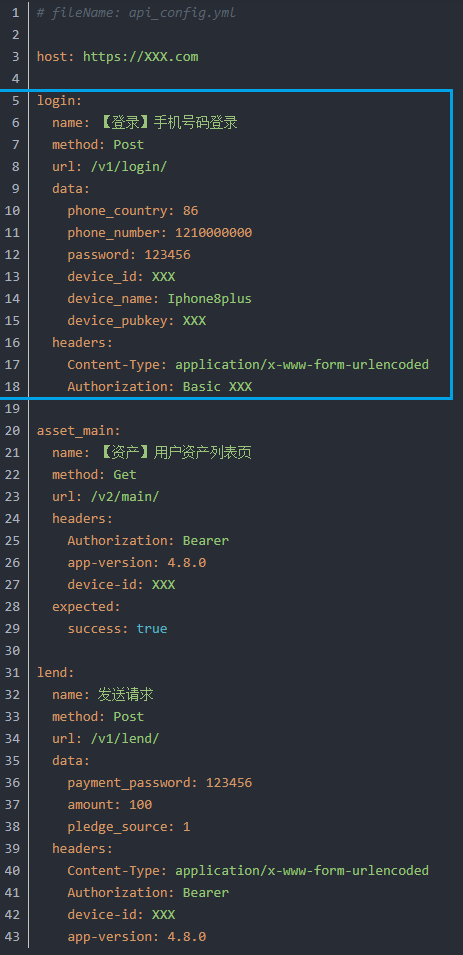
# Python+Requests+Pytest 接口自动化测试脚本总结

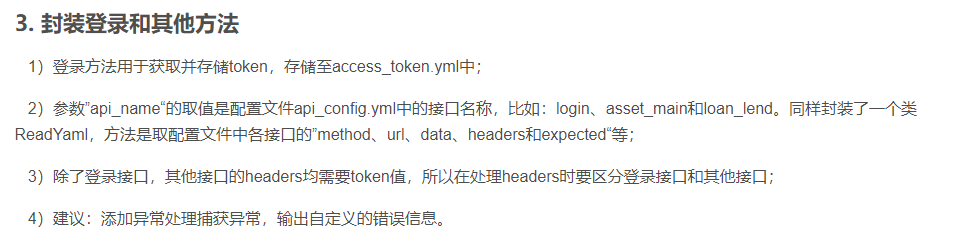
<https://blog.csdn.net/changyixue/article/details/105362848>

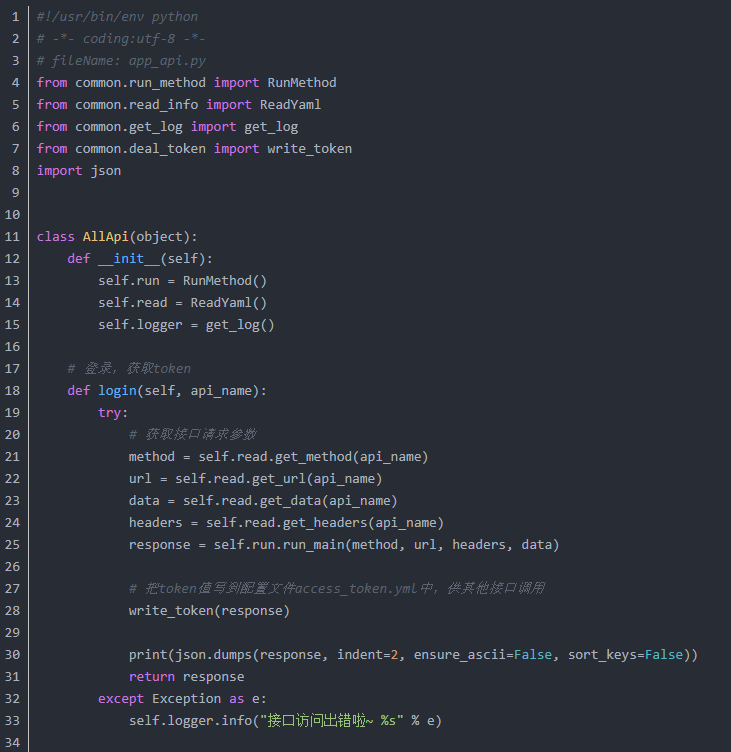


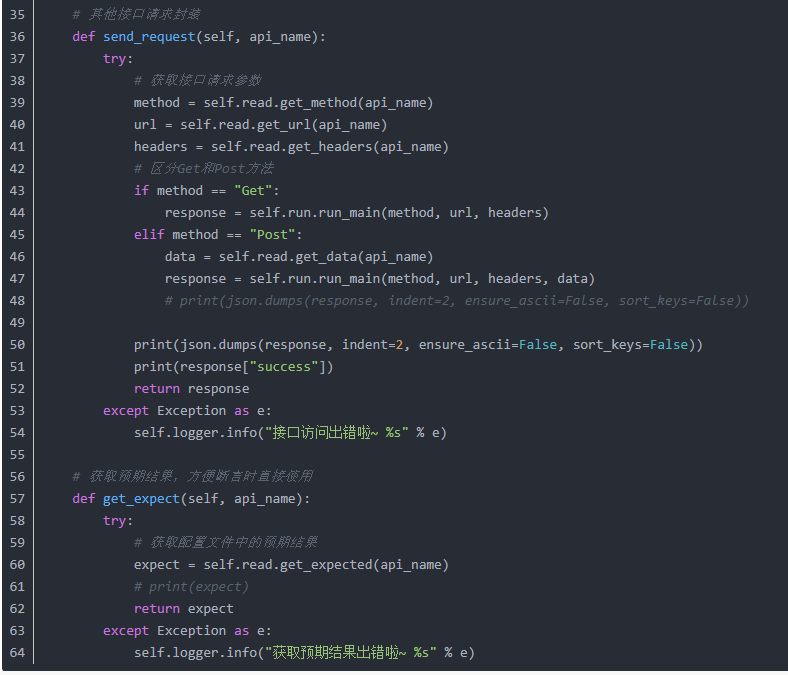
### 2. 存储接口信息

使用yml文件作为配置文件，存储每一个接口信息，主要包括”接口名称、method、url、data和headers“，以及预期结果”expected“

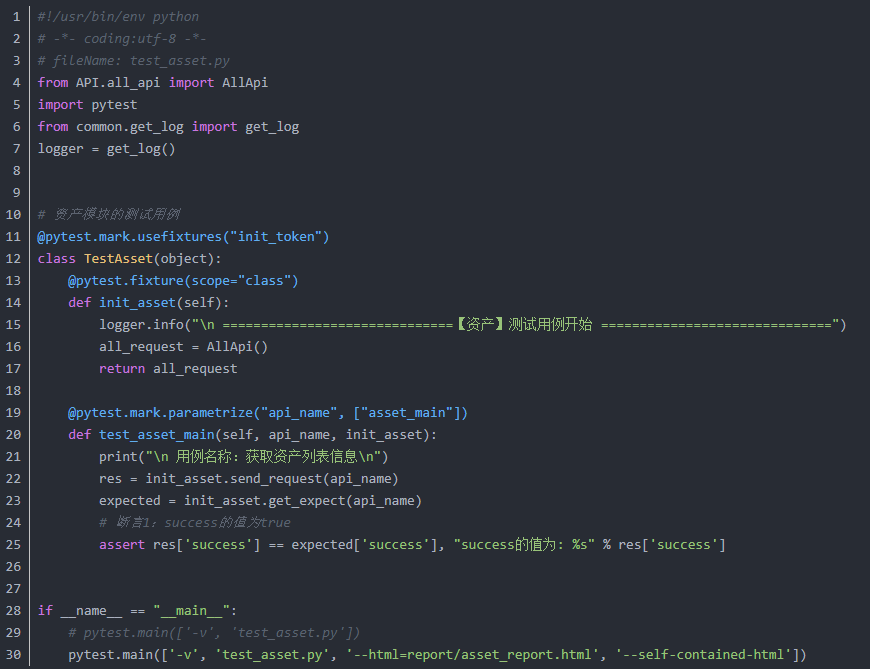


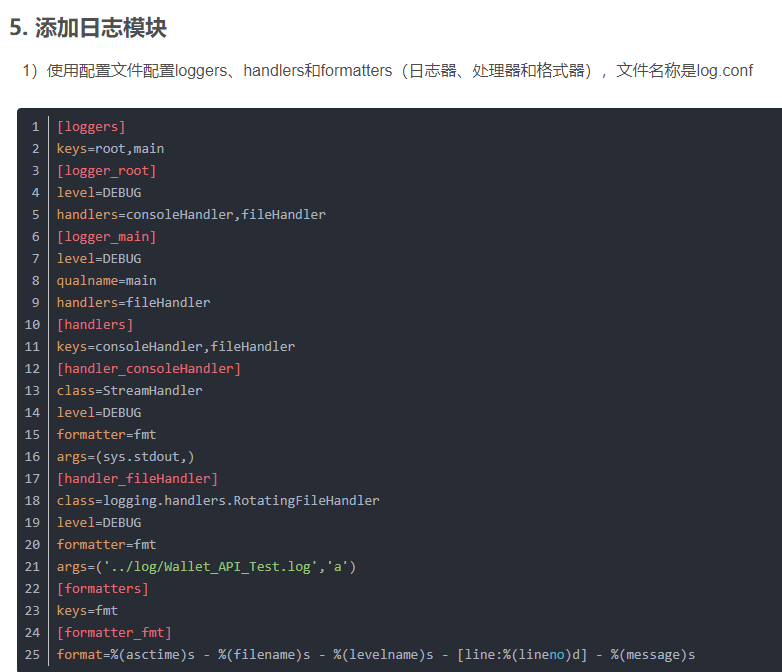




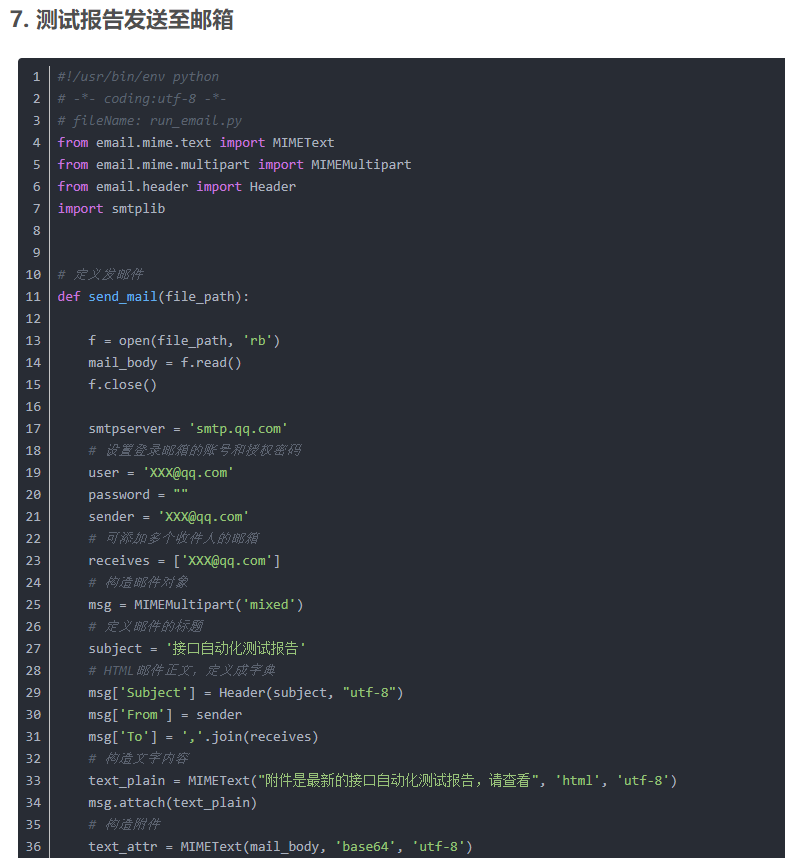


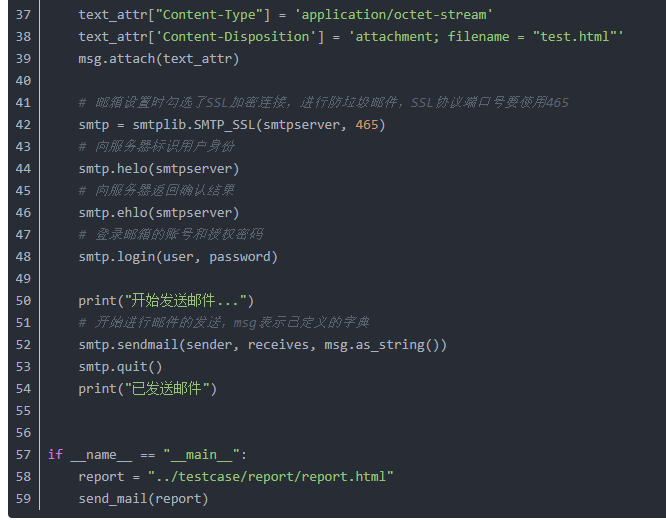


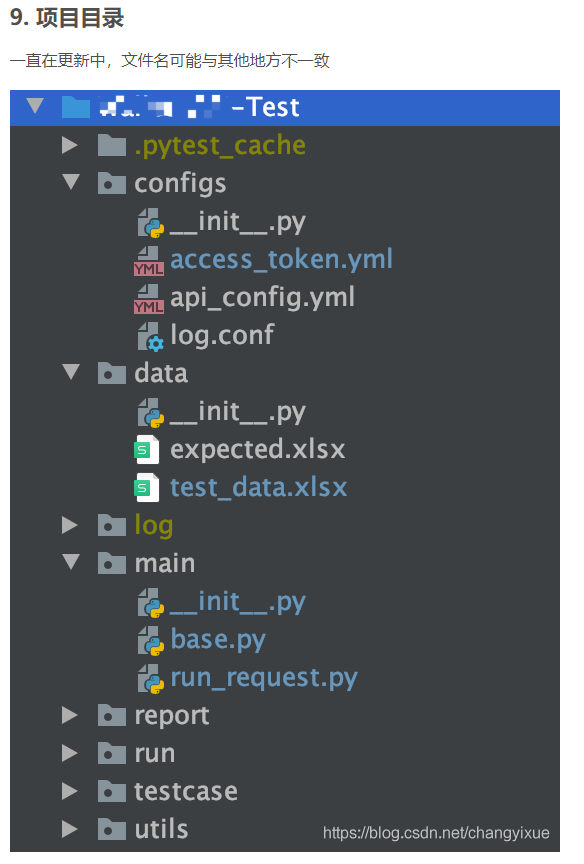




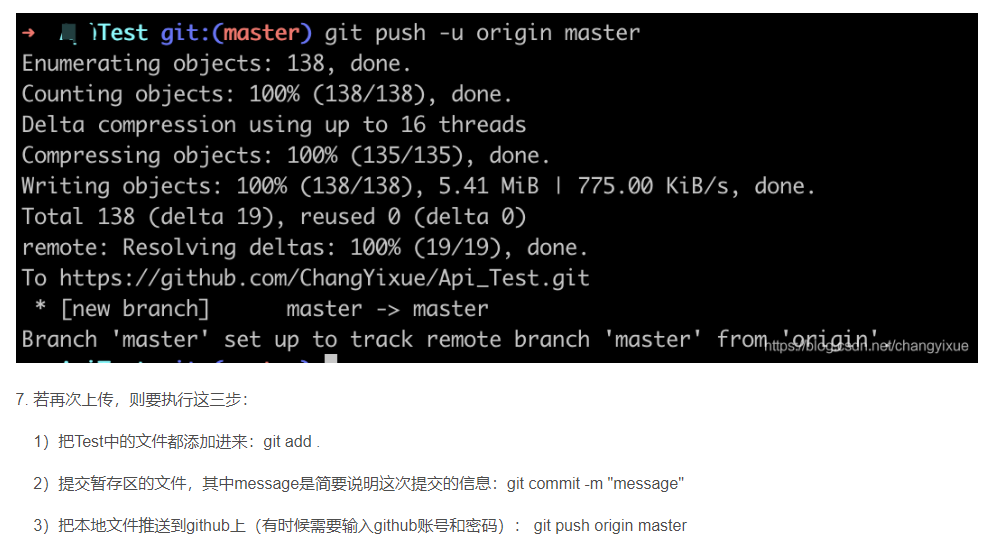




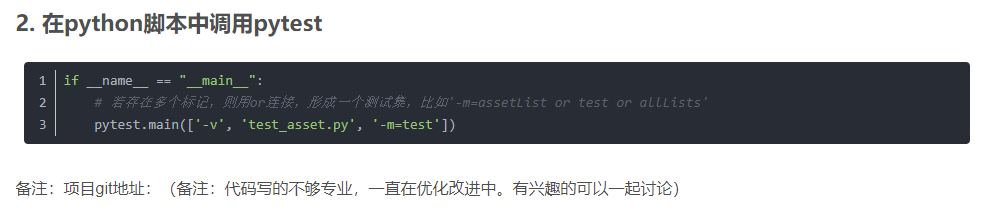












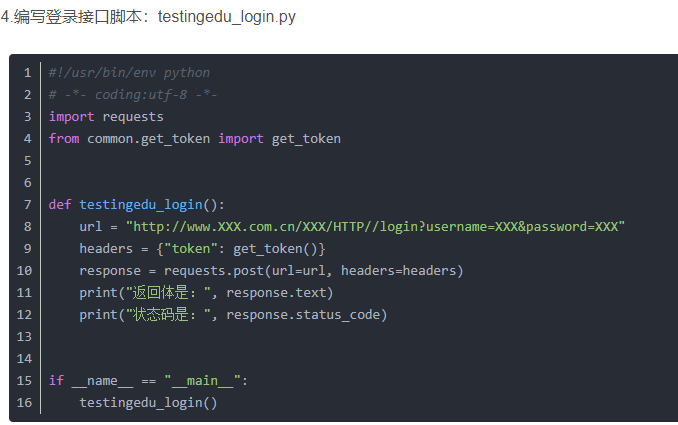
<https://github.com/ChangYixue/Test.git>

# 用python脚本实现一次获取token，多次使用token

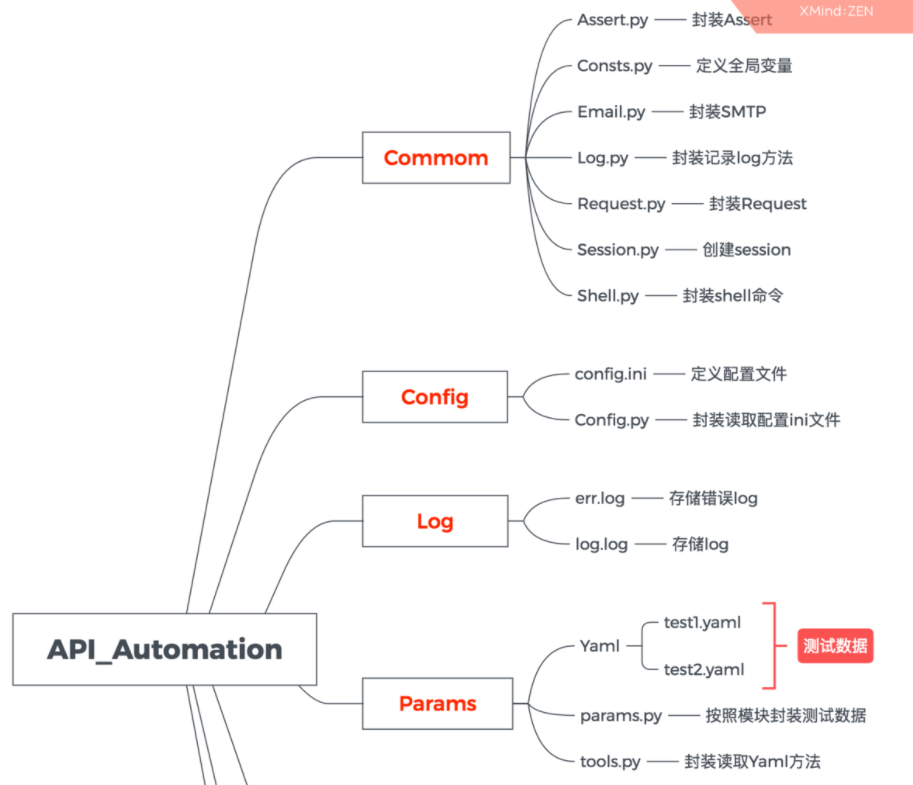


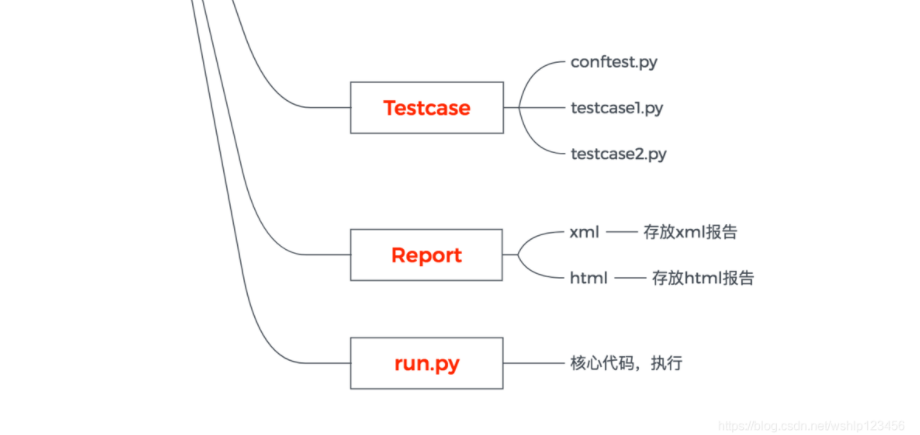


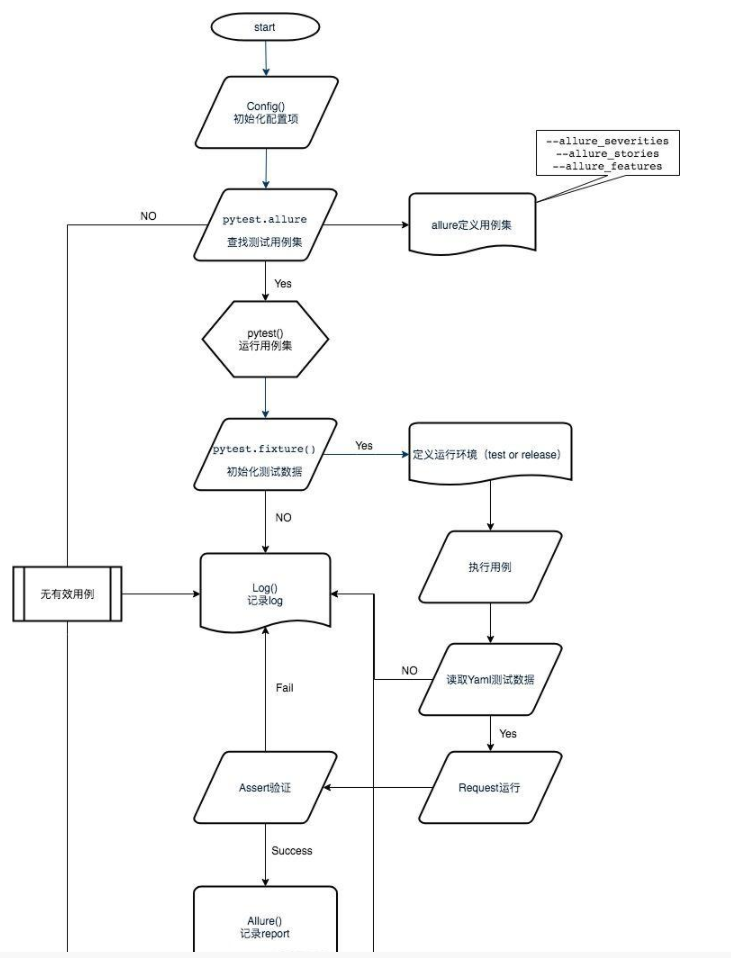


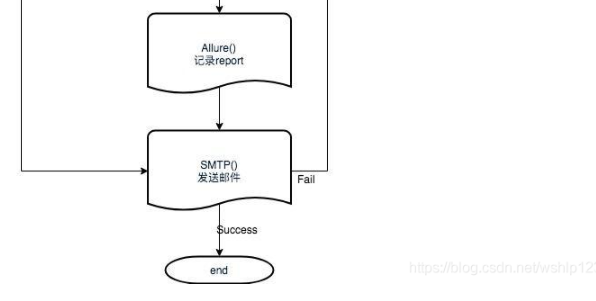


1. **pytest接口自动化测试框架+项目实例**











# Pytest学习笔记7——接口自动化测试框架实战附源码(开荒篇)

# 

# 

# 

# 

# 

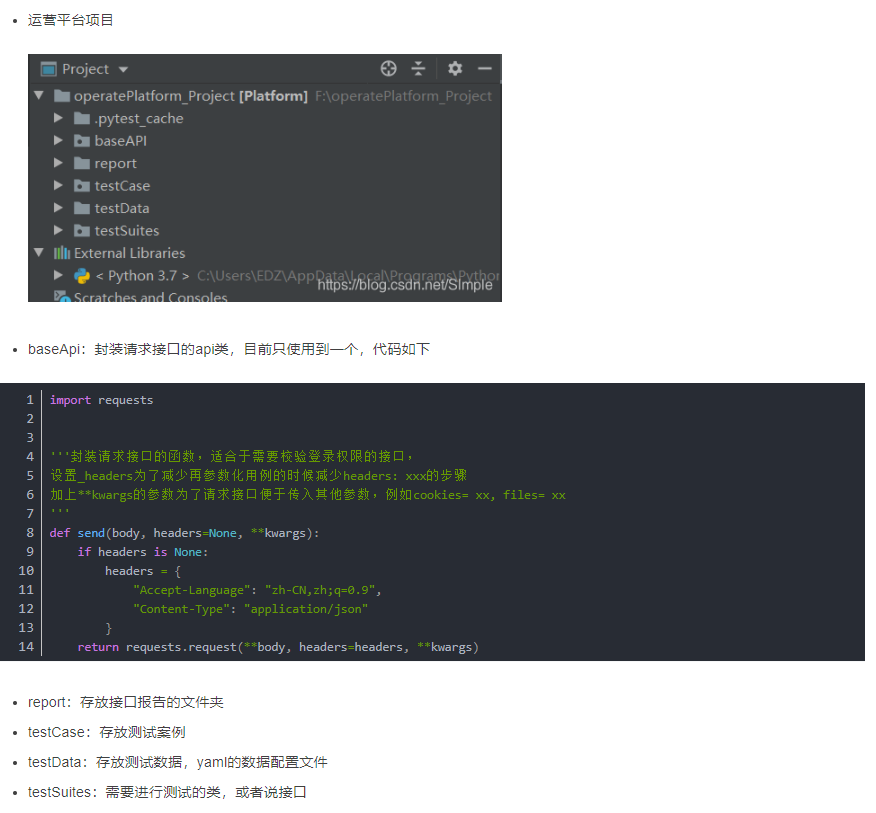
# 

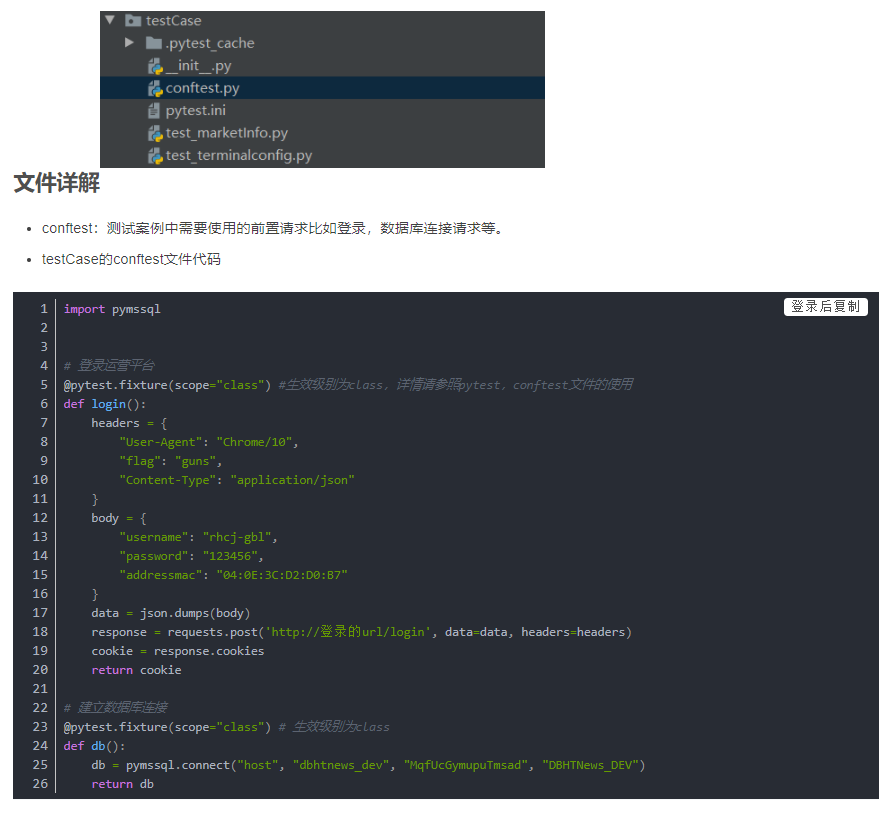
# 

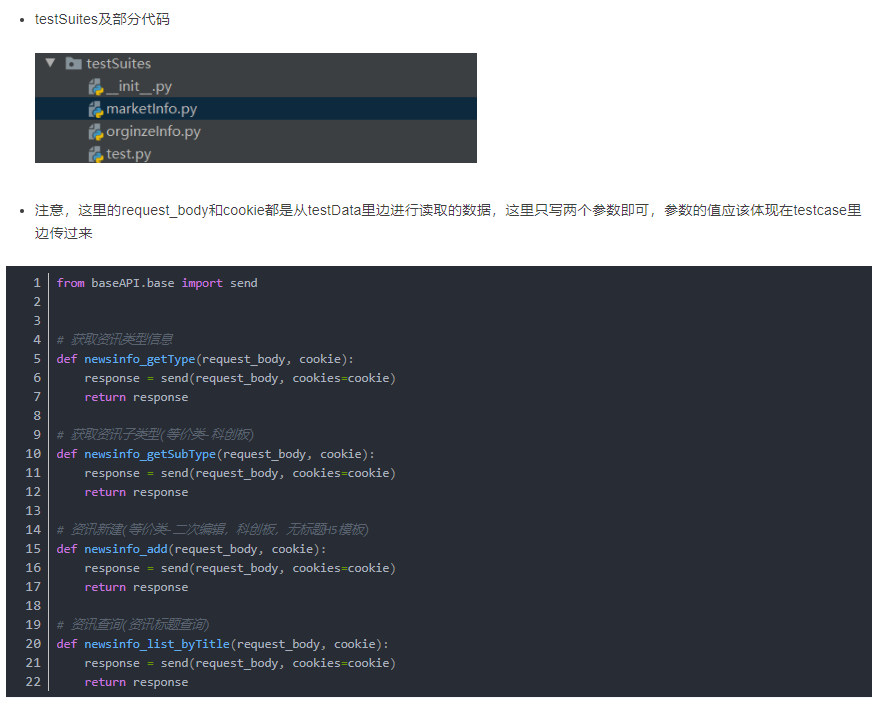
# 

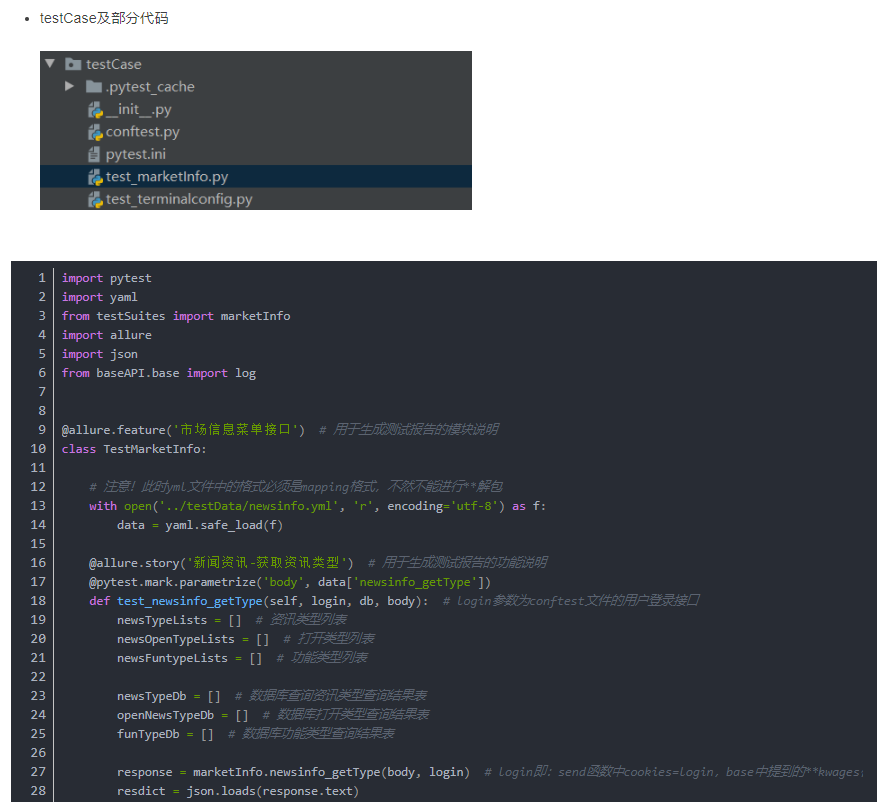
1. **Python pytest+allure的接口自动化测试框架**

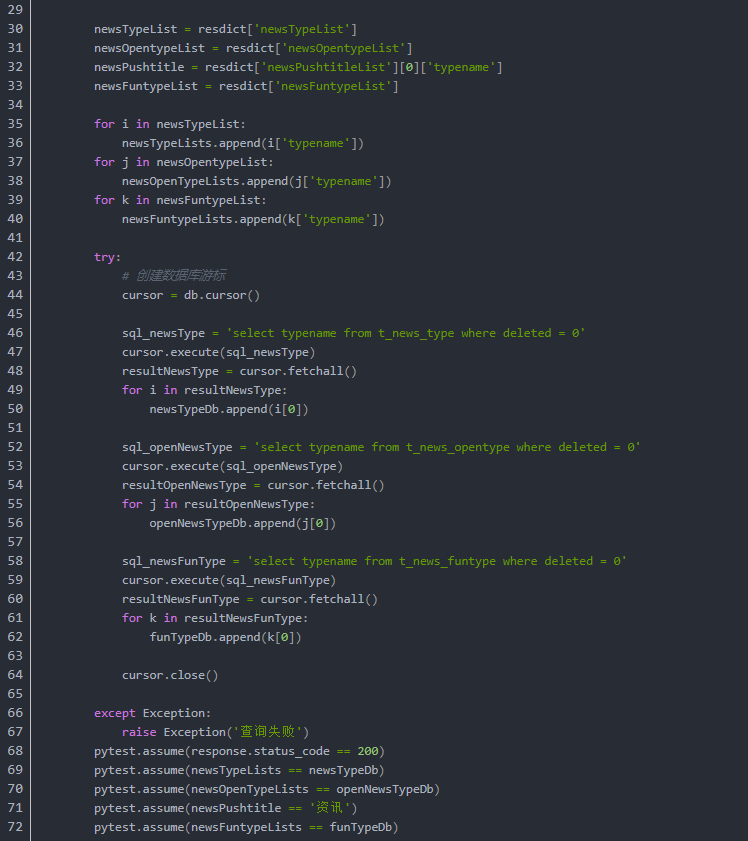
[**https://blog.csdn.net/Slmple/article/details/107641925?utm\_medium=distribute.pc\_relevant.none-task-blog-baidujs\_baidulandingword-4&spm=1001.2101.3001.4242**](https://blog.csdn.net/Slmple/article/details/107641925?utm_medium=distribute.pc_relevant.none-task-blog-baidujs_baidulandingword-4&spm=1001.2101.3001.4242)

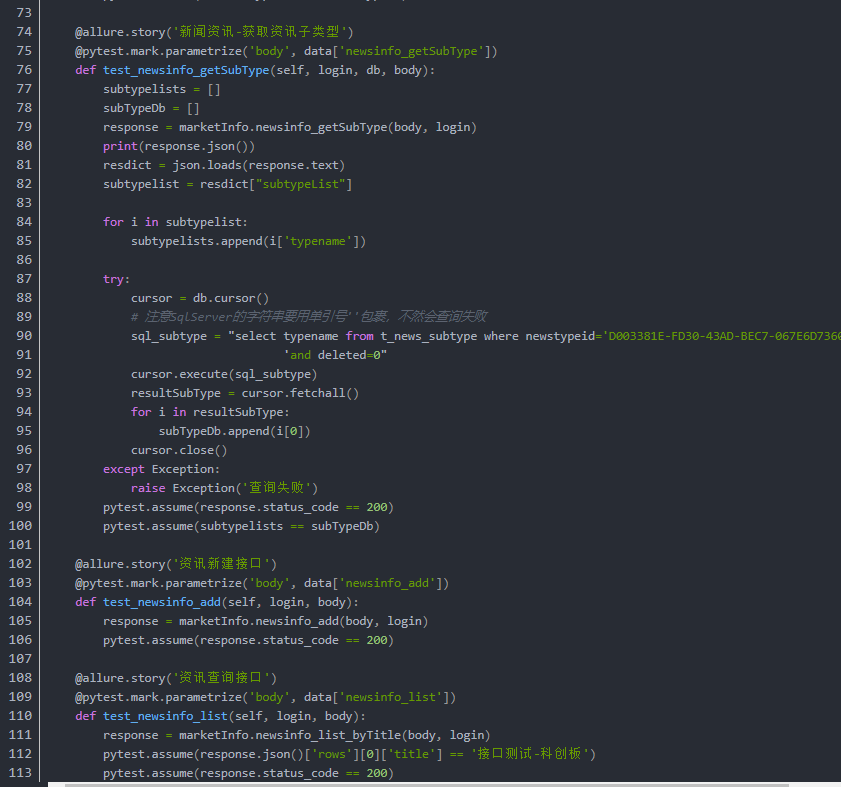
****

****

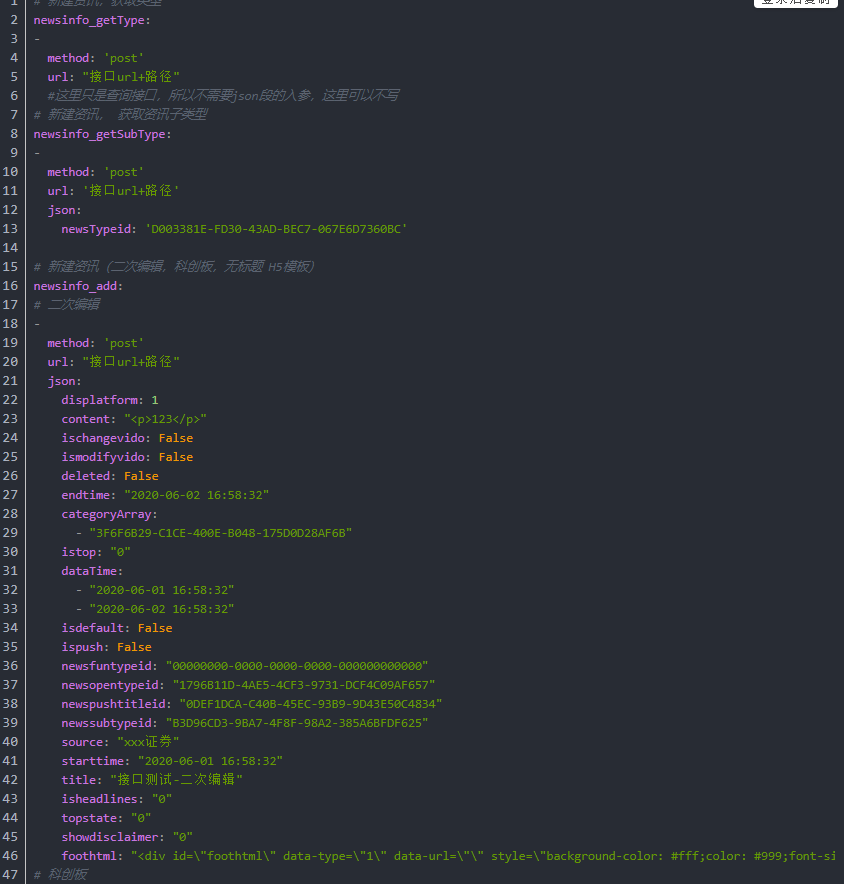
****

****

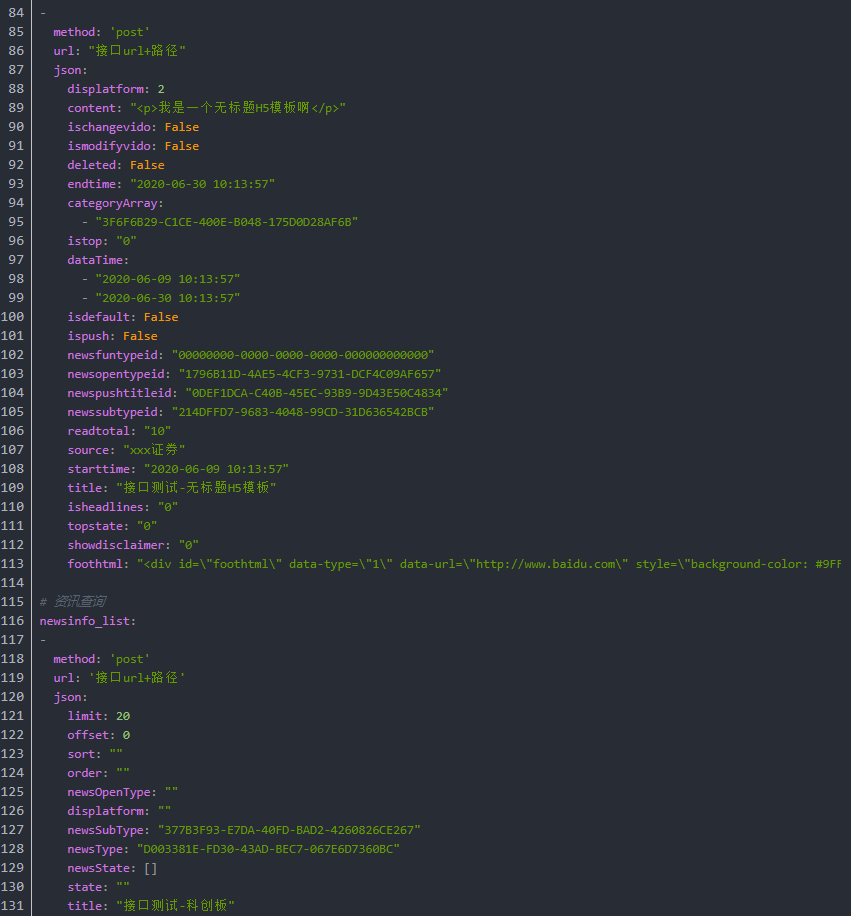
****

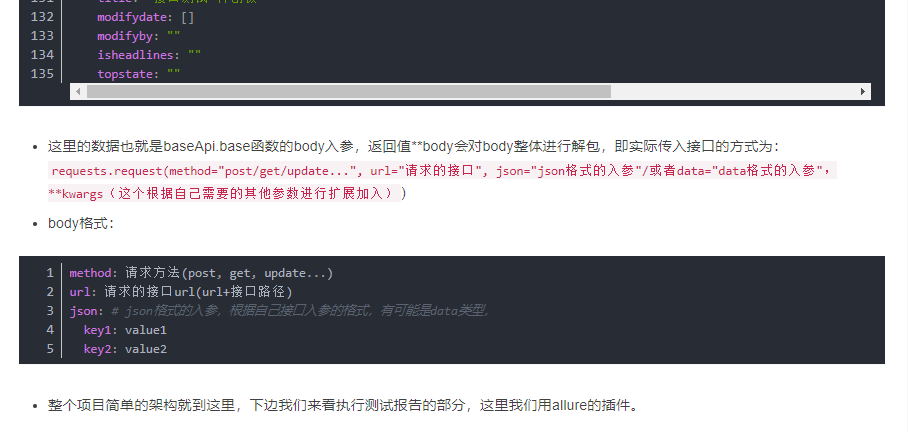
****

****

****

****

****

****

****

****

1. yaml文件在每个冒号后都要加空格
2. 每个request请求url都要加前缀http://
3. 如果用的是mysql数据库，连接语句为db =pymysql.connect(host=“你的数据库地址”,user=“用户名”,password=“密码”,database=“数据库名”,charset=“utf8”)
4. yml配置文件中增加集合类型属性的话 需要用横线(-)进行分割，注意博客中的yaml文件也是用了横线(-)
5. 请求头是headers不是header
6. 请求如果不是json格式的，例如post请求入参是x-www-form-urlencoded格式的，不能使用json.dumps()处理入参，
7. 在testSuite中，调用send方法是入参中的cookies是cookies=cookie，而不是直接传个cookie，否则send方法处理的时候，会将cookie当成headers，而不是\*\*kwargs
8. 如果入参不是json格式的，例如post请求入参是x-www-form-urlencoded格式的，在testData的yml文件中要把json改为data（这个博主讲过了，我当时没细看）

# python接口自动化实战+unitest+pytest+jenkins+allure测试报告

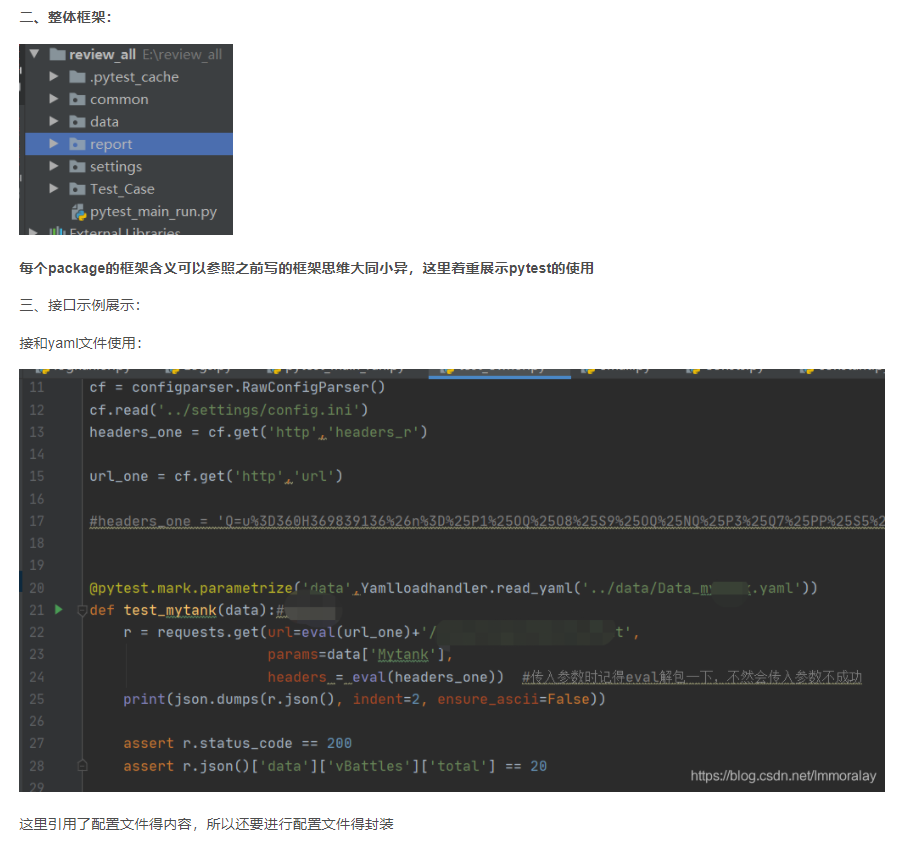
# <https://github.com/heboqiang/python->

# 

pytest测试搜索的规则为：

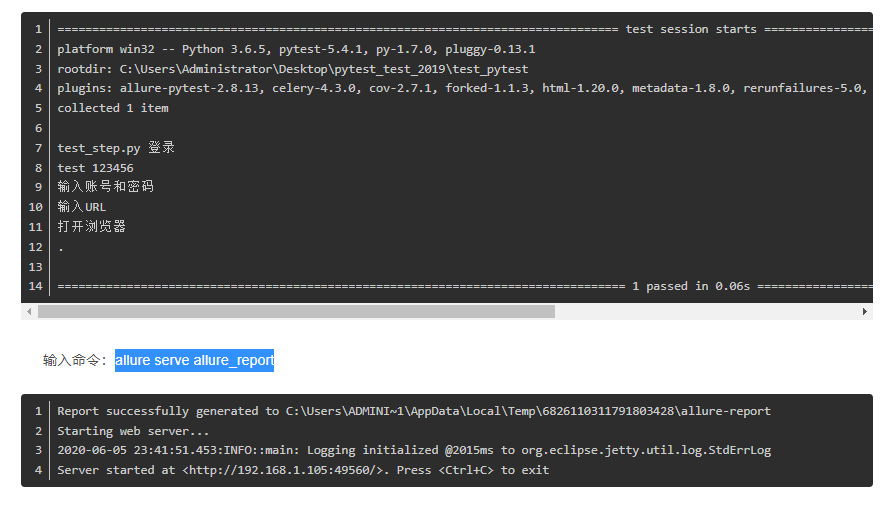
1. 测试文件最好是test开头或者是test为结尾
2. 测试函数或者测试类方法应该当是test开头或者是test为结尾
3. 测试类应当命名为Test<SomeThing>

# 因为用的yaml文件做数据驱动

****









# Pytest接口自动化测试框架搭建模板

<https://blog.csdn.net/VXadmin/article/details/112966633?utm_medium=distribute.pc_relevant_download.none-task-blog-baidujs-4.nonecase&depth_1-utm_source=distribute.pc_relevant_download.none-task-blog-baidujs-4.nonecase>

# Allure测试报告用例详细描述

<https://blog.csdn.net/liudinglong1989/article/details/106597048>