1. **前置条件(环境部署)**

1.具体参考文档《接口自动化环境部署文档》

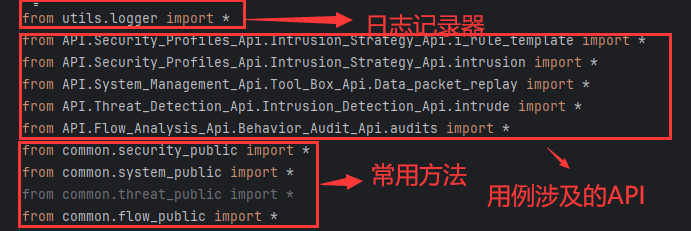
1. **用例编写步骤**
2. 新建测试用例文件

·用例文件一般位于testcases目录下，命名规范以test\_开头具体规则参考pytest.ini文件或《接口自动化框架搭建分享PPT》文件

·在指定目录下创建python文件并命名为

1. 导入模块

·根据用例设计的模块区域导入对应的API,常用方法以及log日志生成器



建议在编写用例过程中导入，防止遗漏

具体目录所在位置可以查看README.md文件

1. 用例标题(以入侵策略功能检测为例)

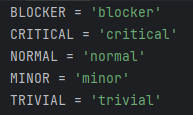
·标题分为两类，采用allure模块编辑标题,使用@allure.feature()命名主要标题，通常放于类名前，表示用例所属模块。例如:用例入侵策略功能检测属于入侵检测模块.使用

@allure.story()命名为用例标题，通常放在用例名前，表示具体用例的标题。例如

1. 测试类名,用例标题,标签

·测试类名以Test\*开头，用例标题以test\_开头

·标签以@pytest.mark.\*\*\*(‘这是一条\*\*\*用例’)命名，如，对用例进行分类，方便之后的筛选

·额外选项:给用例设置优先级，，级别分为5个不等(非必选项，根据需求使用)

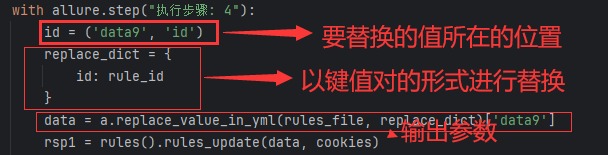
1. 用例编写

·日志记录器，以用例标题+时间戳的格式命名用例日志，调用request方法格式如下:



·编写用例步骤，使用with allure.step(‘执行步骤: n’)(n表示顺序)

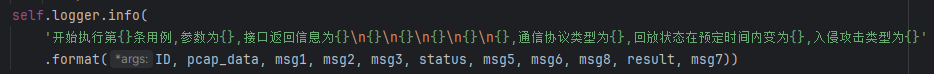
然后在步骤下添加参数信息，调用对应的API获取响应数据



·必要时使用try except获取报错信息



·设置日志输出以self.logger.info(‘\*\*\*’.format(具体参数))格式



·添加断言，作为判断用例通过的标志，在用例结尾处使用assert添加断言信息

多个断言用and连接

1. 参数化

·使用@pytest.mark.parametrize()进行参数化传递数据(以上传数据包为例)



记得导入参数至标题

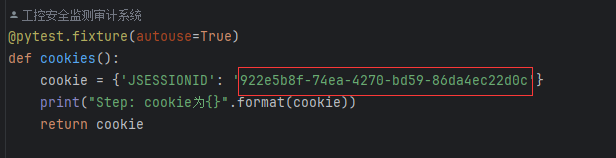
1. 关键文件

·conftest.py文件:存放自定义钩子函数。该文件的主要功能是存放自定义函数，比如获取cookie，获取ip，mac地址等方法(PPT中有相关介绍)

1. 获取cookie



登录后截取cookie保存在conftest.py::cookie目录下



·导入cookie至用例



将cookie参数化传递至用例中，以便后续接口调用



1. 获取随机ip、mac地址

·在调用获取随机ip地址或者mac方法时，需要使用@pytest.mark.file\_path(文件路径)来定位需要修改ip、mac地址的文件(参考testcases/security\_configuration/Baseline\_strategy/test\_assetbaseline.py文件)

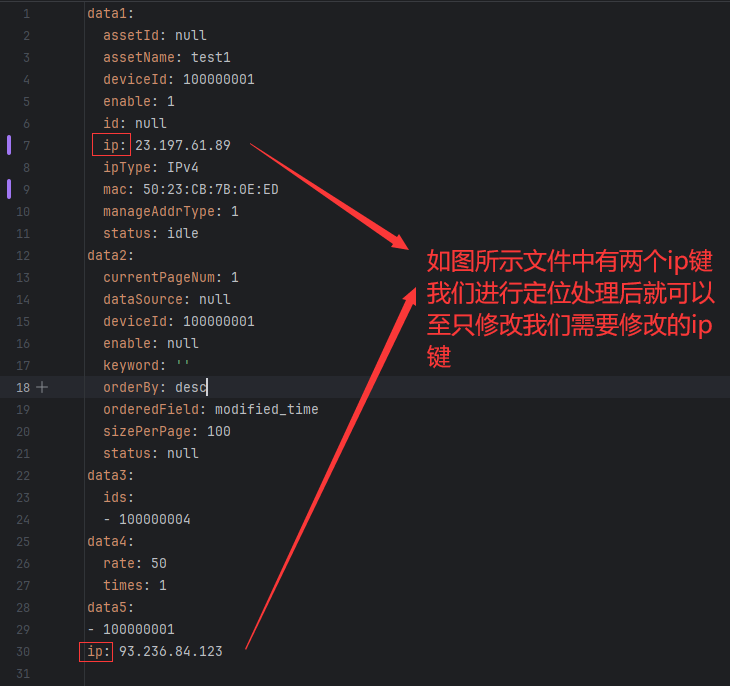


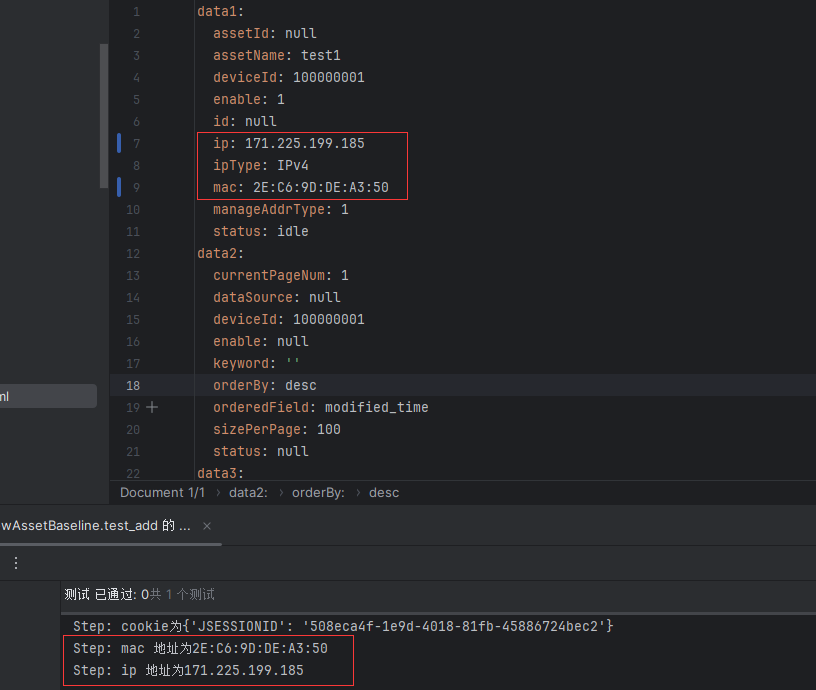
·参数化导入用例

其中mac\_index和ip\_index为前述方法所需的额外参数需要使用

为额外参数赋值

如图代表文件的第一个'mac'键和第一个'ip'键



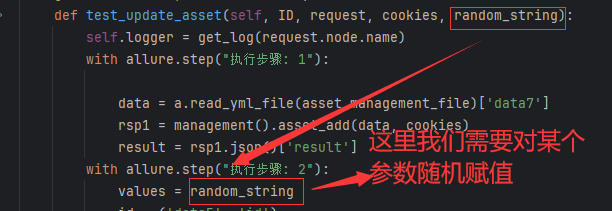


运行后发现mac，ip在源文件中发生了变化

1. 获取随机字符串

·这里比较容易理解,将conftest.py文件中获取随机字符串的方法以参数化的形式添加至用例中

后续需要使用时



如图所示直接调用即可(文件中也包含了获取随机mac,ip地址方法同理)

8.完整用例

·参考目录testcases/security\_configuration/intrusion\_strategy/test\_intrusion.py文件

1. **测试用例**
2. 运行单条用例

·右键运行(这里不进行演示)

·点击用例标题旁的运行按钮(这里不进行演示)

·主函数模式运行:pytest.main([‘-vs’,’./文件路径/模块名.py::类名::方法名’])

·终端运行用例:pytest ‘./文件路径/模块名.py::类名::方法名’

1. 运行同类型用例(这里指的使用@pytest.mark.标签名的用例)

·终端运行:pytest -m 标签名

1. 运行所有用例

·终端运行所有用例: pytest ./文件路径

·主函数模式运行:pytest.main([‘-vs’,’./文件路径’])

1. 运行规则

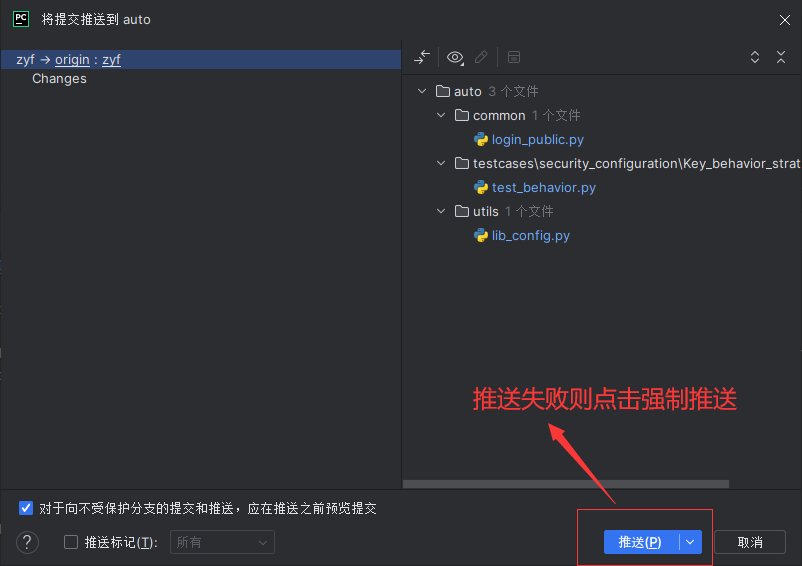
·不管是主函数模式还是命令行模式，都会去读取pytest.ini配置文件

1. **提交代码**(以pycharm提交代码举例)
2. 前置条件

·连接gitlab仓库:左上角点击git-->管理远程-->添加gitlab地址

1. 提交代码

·点击左上角后提交文件其中更改表示已经提交过但是有修改后的文件。未进行版本管理的文件表示新增的文件。根据具体需求提交并推送



·提交分支代码至主分支

1. 切换到分支: git checkout 分支名
2. 拉取代码至本地: git pull
3. 切换到主分支: git checkout 主分支名
4. 提交分支代码到主分支: git merge 分支名
5. 推送代码: git push