|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **地平线机器人** | **文档编号** |  | **版本** | A1 | **密级** |  |
| **项目名称** |  | | | | |
| **项目来源** |  | | | | |

**座舱自动化测试**

(内部资料 请勿外传)

|  |  |  |  |
| --- | --- | --- | --- |
| **编 写：** |  | **日 期：** |  |
| **检 查：** |  | **日 期：** |  |
| **审 核：** |  | **日 期：** |  |
| **批 准：** |  | **日 期：** |  |

**地平线机器人**

**版权所有 不得复制**

**文档变更记录**

目录

[1 项目简介 2](#_Toc62568821)

[1.1 项目简介 2](#_Toc62568822)

[1.2 项目GitLab 2](#_Toc62568823)

[2 环境准备 3](#_Toc62568824)

[2.1 安装包 3](#_Toc62568825)

[2.2 安装指南 3](#_Toc62568826)

[3 uiautomatorviewer 使用方法 4](#_Toc62568827)

[4 规范 5](#_Toc62568828)

[4.1 编写格式 5](#_Toc62568829)

[4.2 新增测试模块 5](#_Toc62568830)

[5 pytest用例编写 6](#_Toc62568831)

[5.1 项目目录说明 6](#_Toc62568832)

[5.2 打开项目 7](#_Toc62568833)

[5.3 必要的配置（config） 7](#_Toc62568834)

[5.4 用例的元素定位符（variable） 8](#_Toc62568835)

[5.5 用例的前置与后置（conftest） 9](#_Toc62568836)

[5.6 项目的公共接口（${name}\_lib） 10](#_Toc62568837)

[5.7 模块的公共接口（${module\_name}\_library） 11](#_Toc62568838)

[5.8 测试用例（testcase） 12](#_Toc62568839)

[6 附件 13](#_Toc62568840)

[6.1 git使用 13](#_Toc62568841)

# 项目简介

## 项目GitLab

<http://gitlab.hobot.cc/qa/algorithm/dms_evaluation_tools-/tree/master/DMS_AutoTest/DMS_TEST_PT>

注：对git操作不熟悉的同学，请查看附件“git使用”

# 环境准备

## 环境安装

1. Python、Pycharm：[\\10.64.29.17\software\python](file:///\\\\10.64.29.17\\software\\python)

python-3.7.3-amd64.exe， pycharm-community-2020.2.1.exe

Python、Pycharm 安装指南：<https://www.runoob.com/w3cnote/pycharm-windows-install.html>

1. requirements： [\\10.64.29.17\software\python](file:///\\10.64.29.17\software\python)

放在pip.exe目录下，cmd窗口输入如下指令：pip install -r requirements.txt

1. Java JDK： [\\10.64.29.17\software\jdk](file:///\\10.64.29.17\software\jdk)

JDK 安装指南： <https://jingyan.baidu.com/article/e5c39bf5ca44d639d7603332.html>

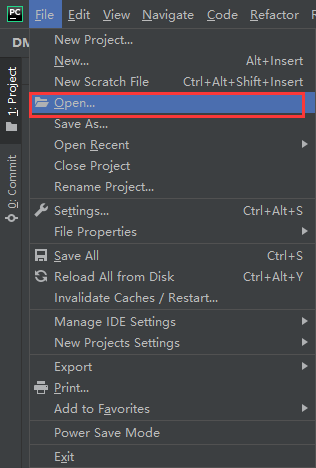
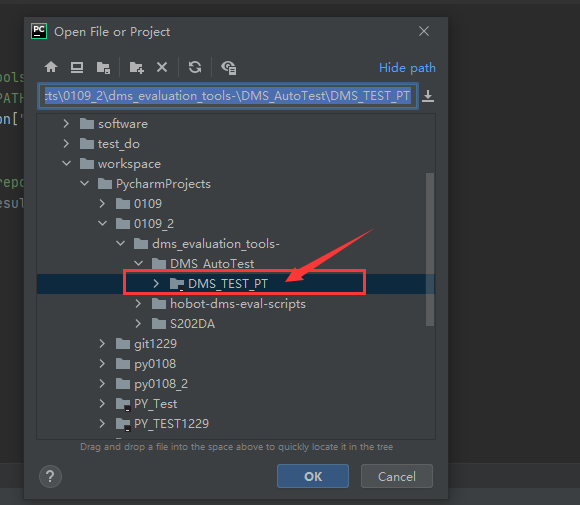
1. Uiautomatorview：[\\10.64.29.17\software\jdk\uiautomatorview](file:///\\\\10.64.29.17\\software\\jdk\\uiautomatorview)

Uiautomatorview为免安装版，本地adb环境配置完成即可

## 工程打开方式

1.双击pycharm图标-》file-》open-》DMS\_TEST\_PT

2.执行程序之前，请确认本地环境配置完成，python,pip,adb,java等环境变量配置正确。

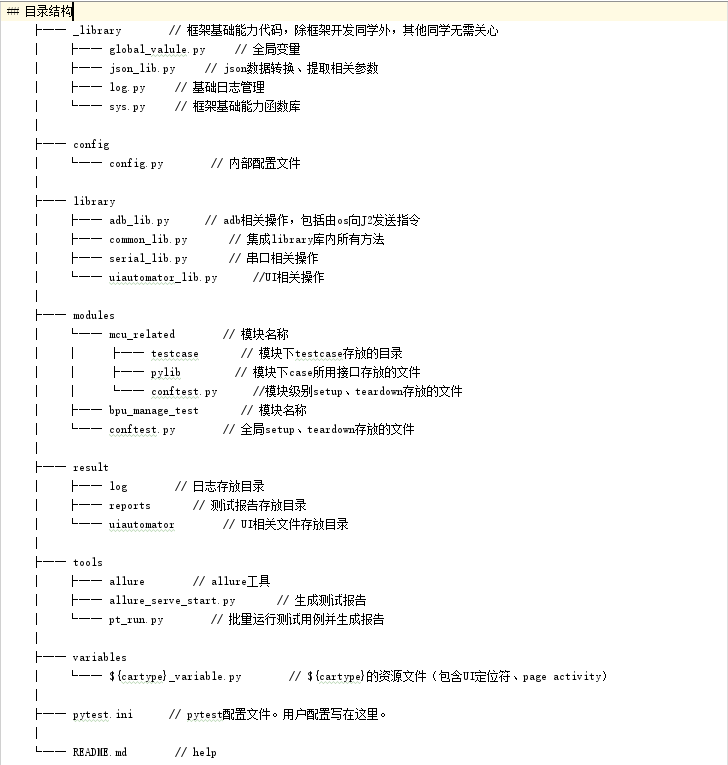
## 项目简介

主要技术：python，pytest，adb，uiautomatorviewer，serial，ini配置文件等等

本工程使用python开源的pytest插件，根据业务定制化实现自动化测试框架，目前工程能够提供的功能包括：

1. 批量自动化运行用例生成清晰的测试报告；
2. 冒烟执行，分标签执行，指定模块执行；
3. 失败用例截图、重试；
4. 结合adb、UIautomator实现UI相关事务性操作；
5. 封装常用adb、串口，文件处理，升级相关操作的高级函数；
6. 定制化日志输出格式、制定日志级别；
7. 根据配置去动态导入不同型号车机参数文件；
8. 已实现软件用例100条；

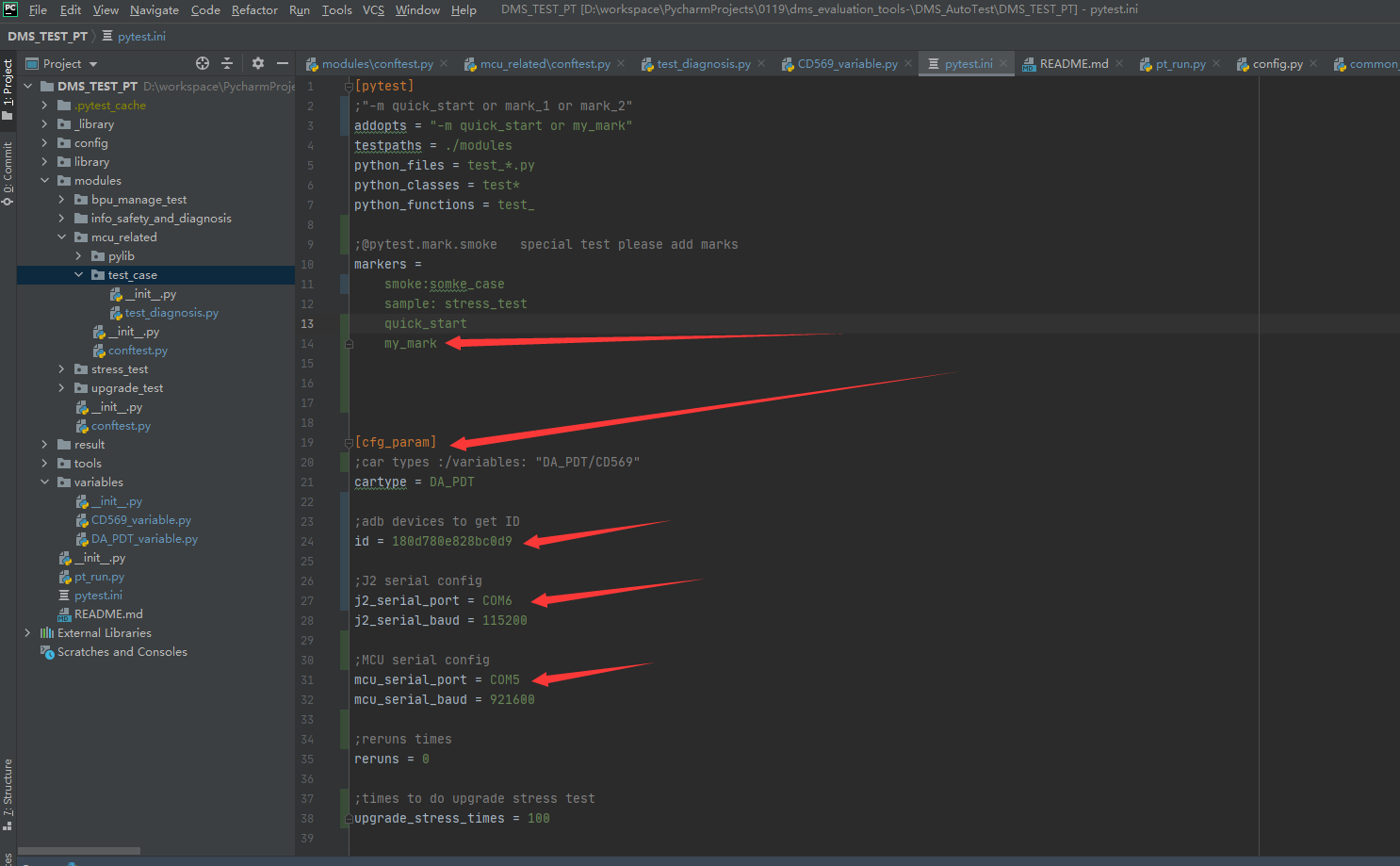
## 项目目录说明



# 工程使用和规范

## 必要的配置（config）

运行代码前，首先配置pytest.ini文件。否则配置与你本地不符，会无法运行成功

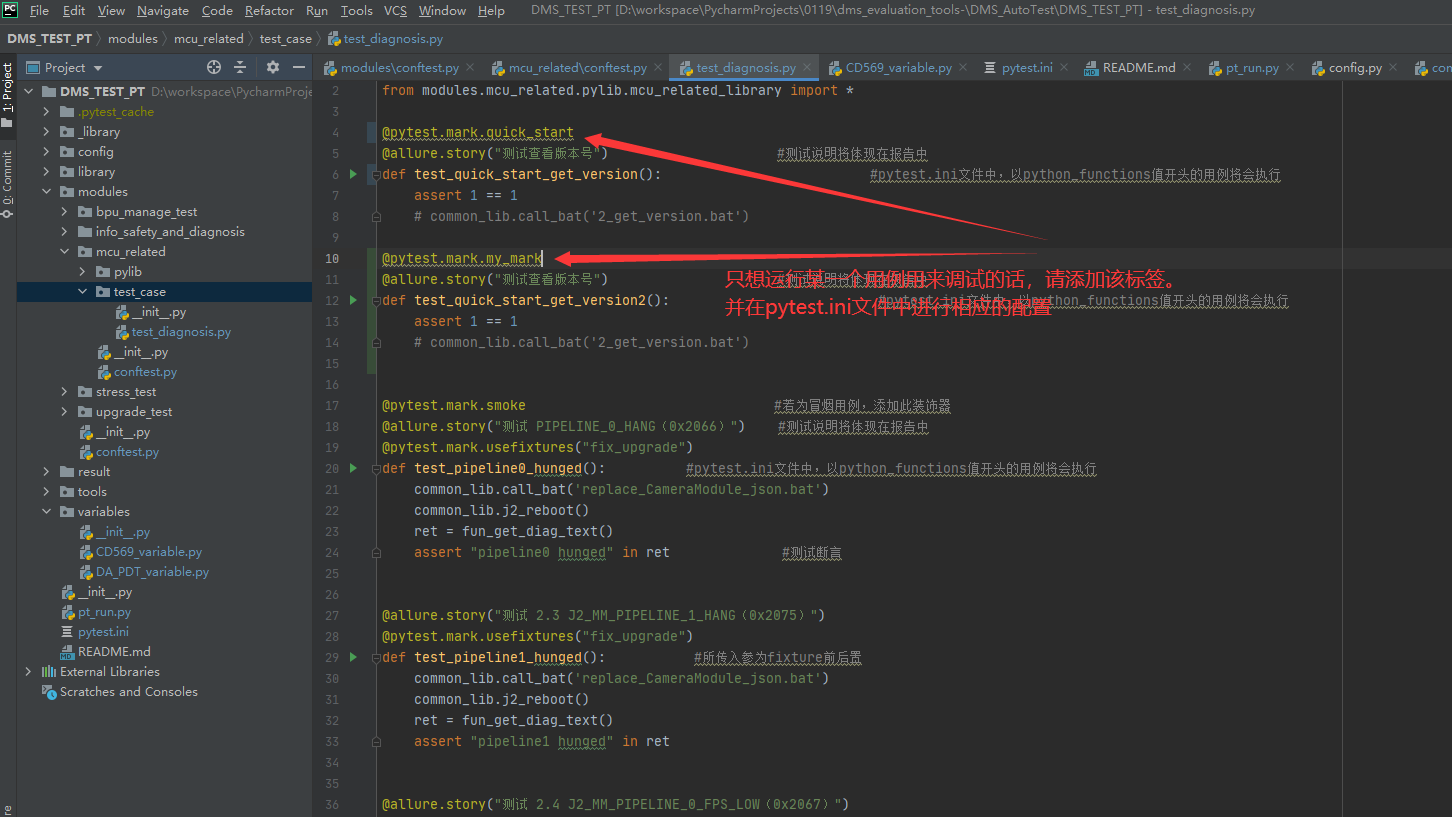


## 编写格式

1. 运行代码前，首先配置pytest.ini文件。否则配置与你本地不符，会无法运行成功
2. 测试用例文件和函数，请使用“test\_”为开头；
3. 除测试用例之外的文件和函数，请不要以“test”为开头或结尾，否则系统会误认为该函数为测试用例！
4. 在执行与调试代码时，请通过添加markers的方式，指定执行某函数。请通过运行pt\_run来运行和调试case
5. 注意：pytest.ini 文件中只能使用纯英文字符，绝对不能使用中文的字符（尤其是冒号和空格）！，ini文件只支持全行注释，用“;”打头
6. 为了统一命名规范：编写的过程中，所有的文件名、类名、函数名、变量名，请全部使用小写，以下划线连接。只有全局变量使用全大写；变量命名尽量“望文生意”，尽量避免“a,aaa”这种命名。
7. 在你上传代码的时候请不要将pytest.ini配置文件上传到gitlab；
8. 在编写的过程中请不要使用print，如果想要输出日志，请使用：log.logger.info(**"XXX"**)；
9. 如果可以我们还希望你能够在代码上传前使用Flake8，进行代码扫描，并对代码格式不正确的地方进行修改；

## 新增测试模块

新增测试用例，若正在编写调试该用例，只想运行这一个用例的话。请给该用例添加标签。并在pytest.ini文件中进行相应的配置。



当你编写一个新模块的用例时，必须创建如下文件（以“mcu\_related”为例）：

1、modules/mcu\_related/ # 在modules目录下面创建一个“mcu\_related”文件，放置专属于此模块的公共方法以及测试用例

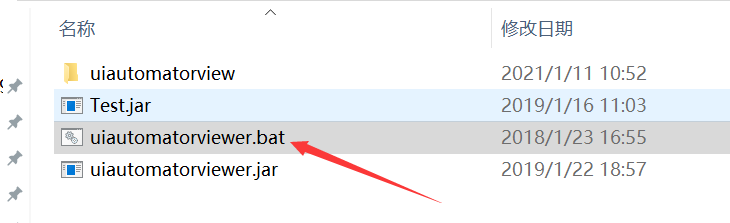
2、mcu\_related/pylib/mcu\_related\_library.py # 公共方法存放

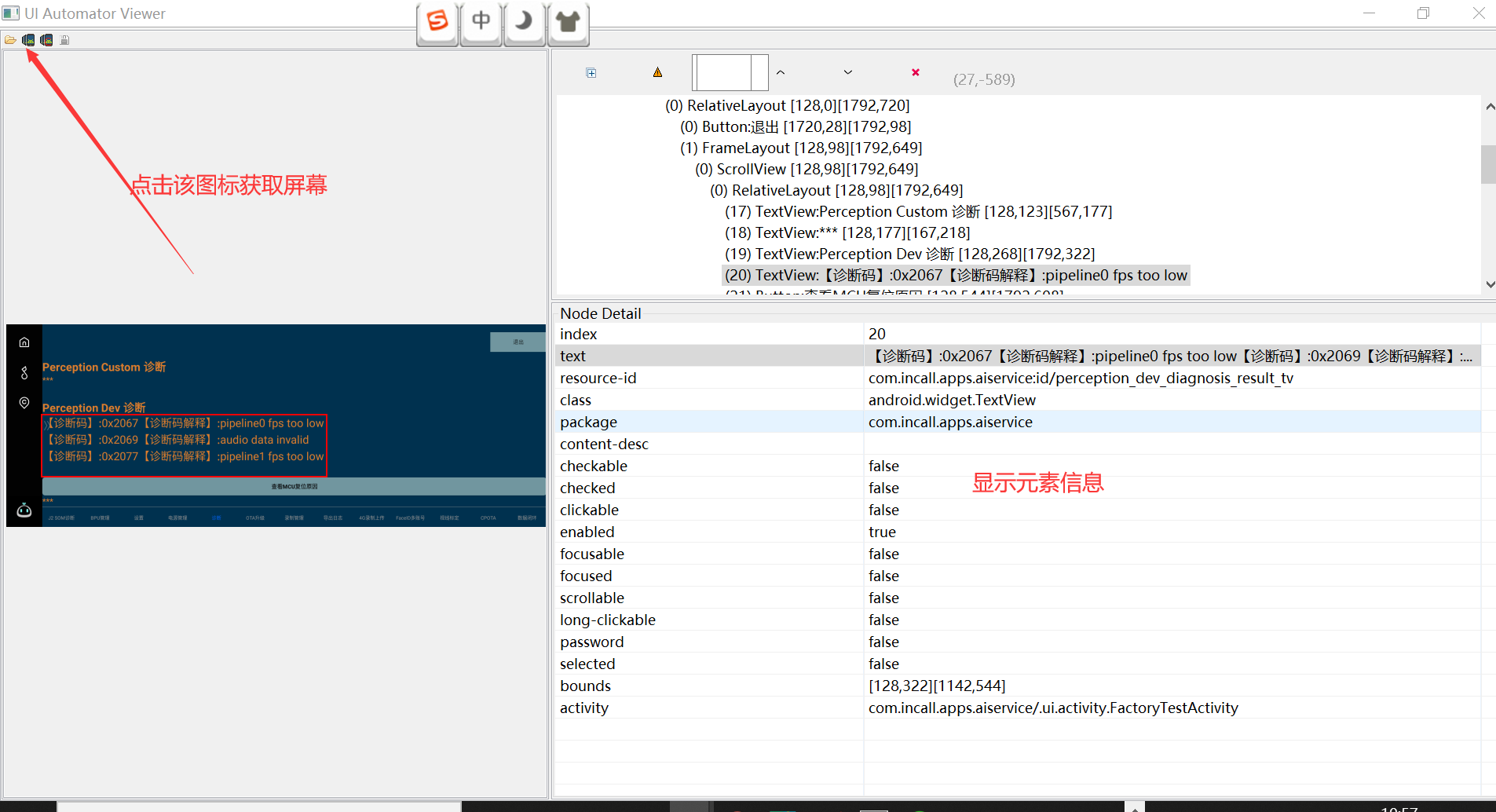
3、mcu\_related/test\_case/ # 测试用例存放

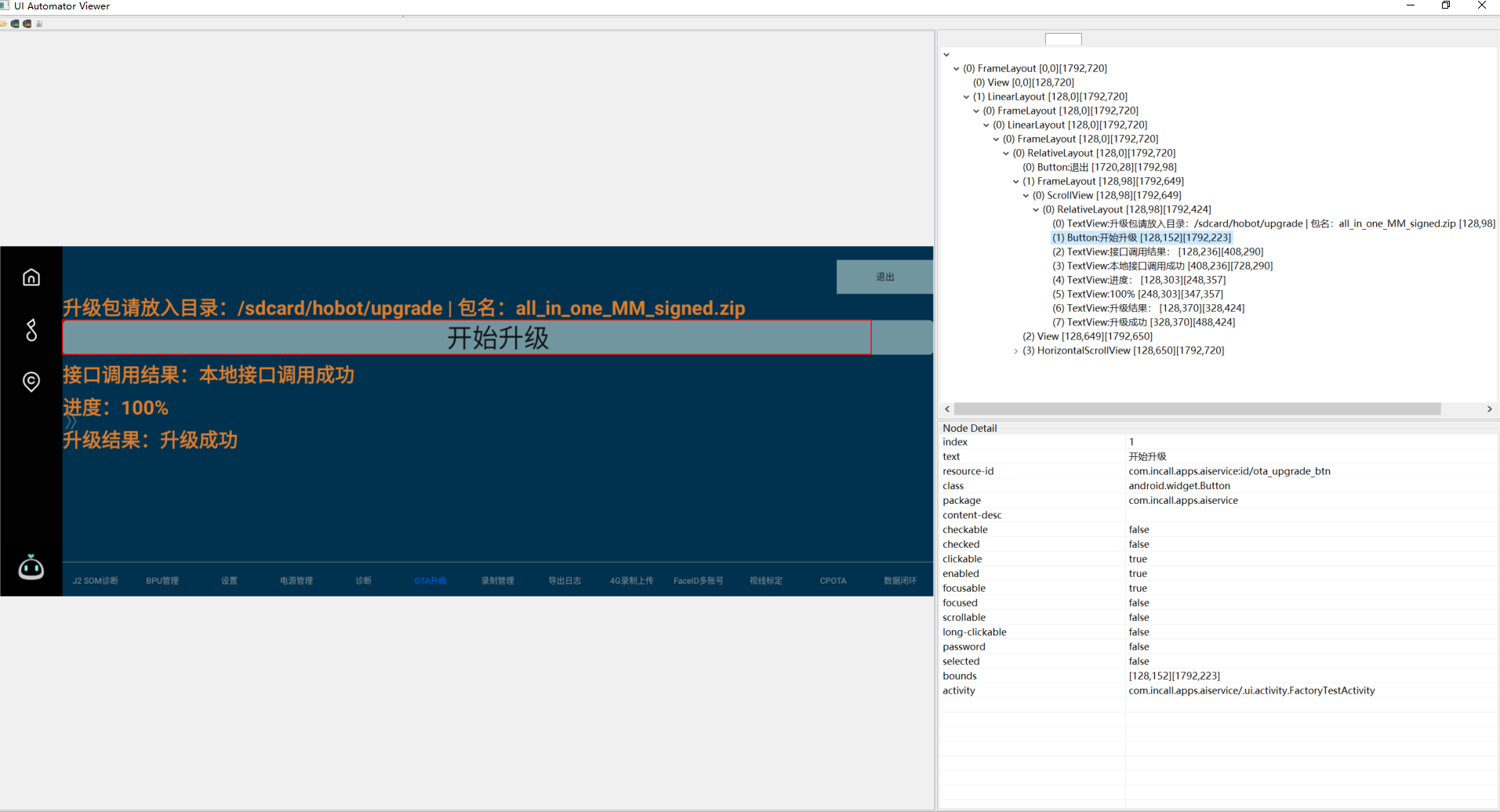
4、mcu\_related/conftest.py # 前置后置存放

# uiautomatorviewer 使用方法

ADB通信正常，安装Java jdk之后。在Uiautomatorview文件夹中，直接双击运行uiautomatorviewer.bat。



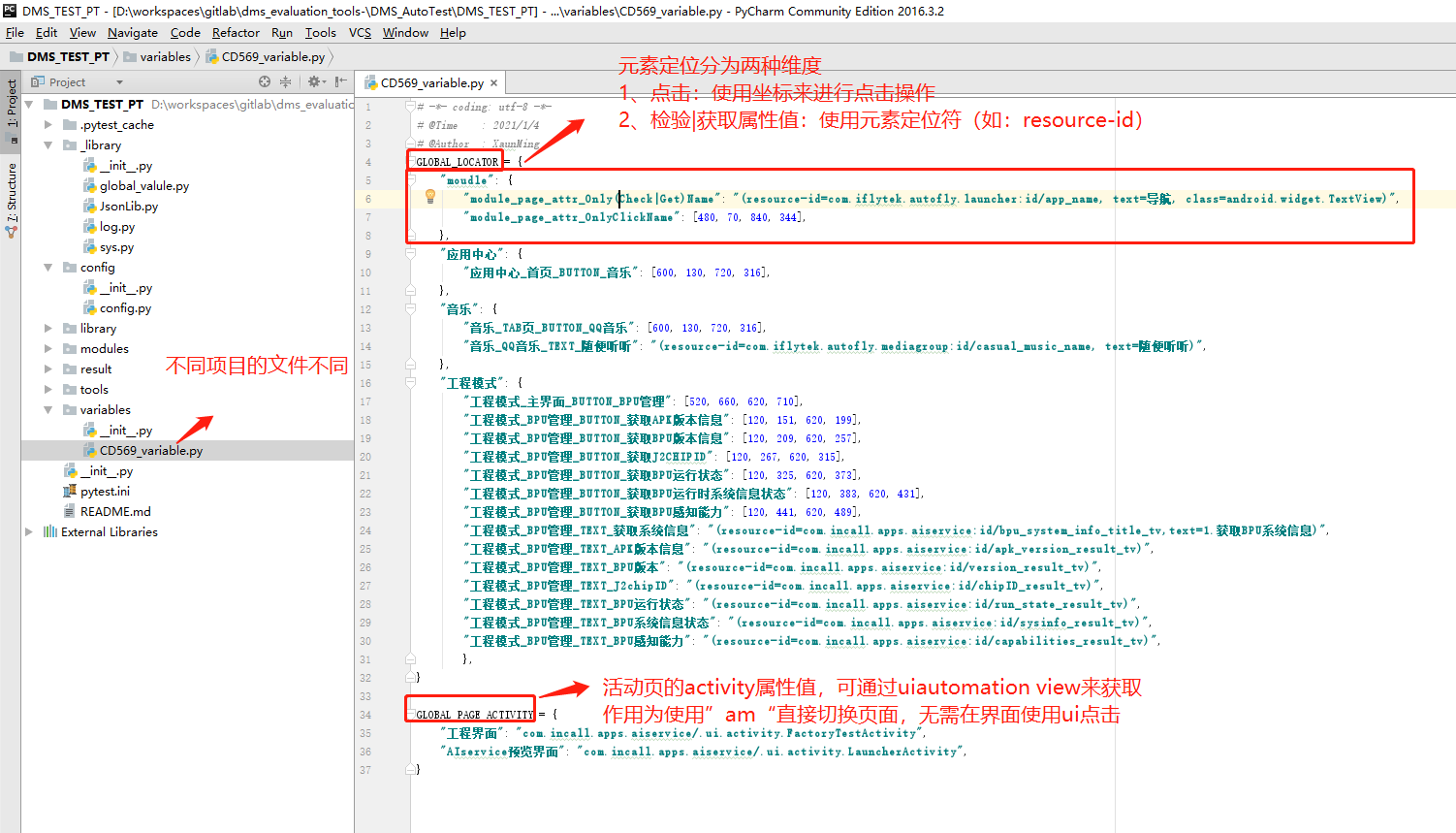


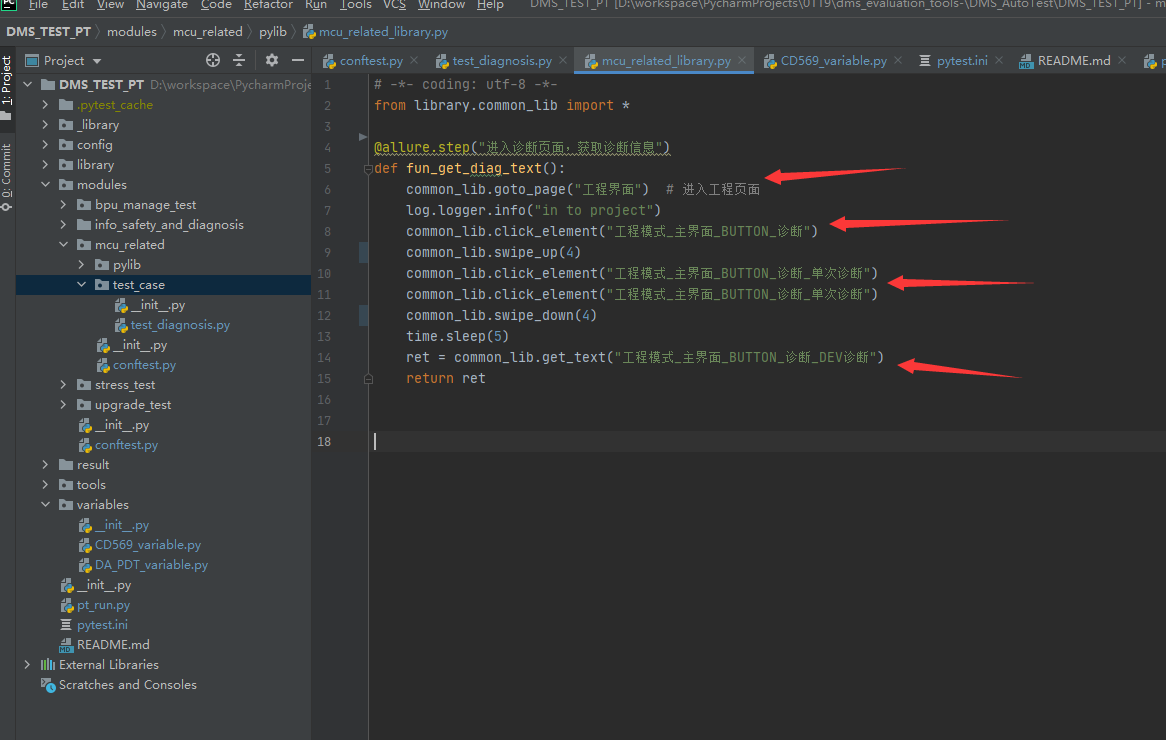


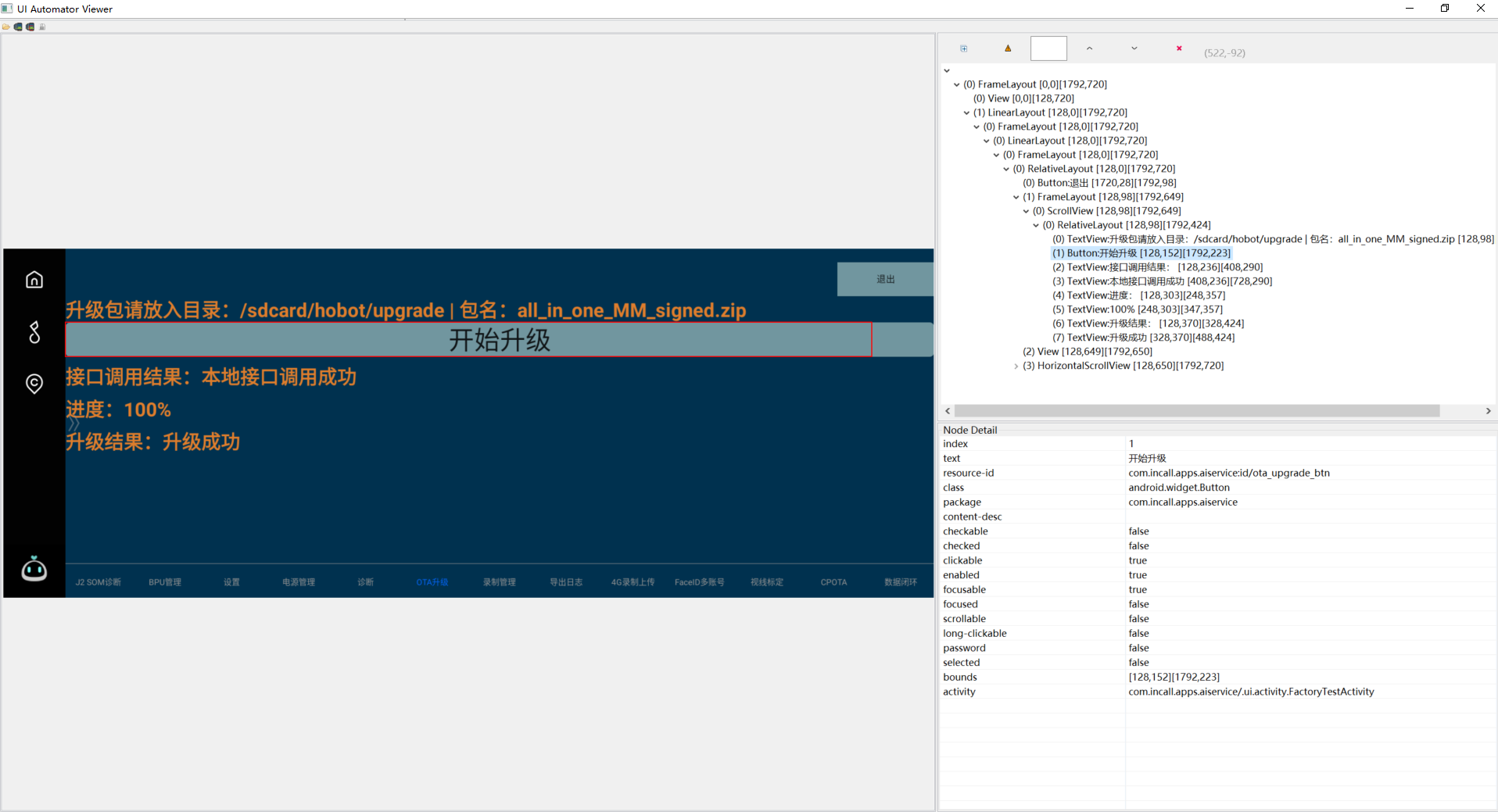
注：若无法成功获取，请通过adb devices查看adb是否连接成功

## 用例的元素定位符（variable）

每一个型号的车机都需要有一个专属的variable文件，当你需要重头编写一个项目的自动化用例时，请你新建一个相应的资源文件。







# 用例的编写

## 用例的前置与后置（conftest）

使用装饰器@pytest.fixture(scope="XXX"，autouse=Ture/False,)，可以将函数设置为setup和teardown，scope参数有如下选项，作用的范围不同，请各位同学根据需要选择。

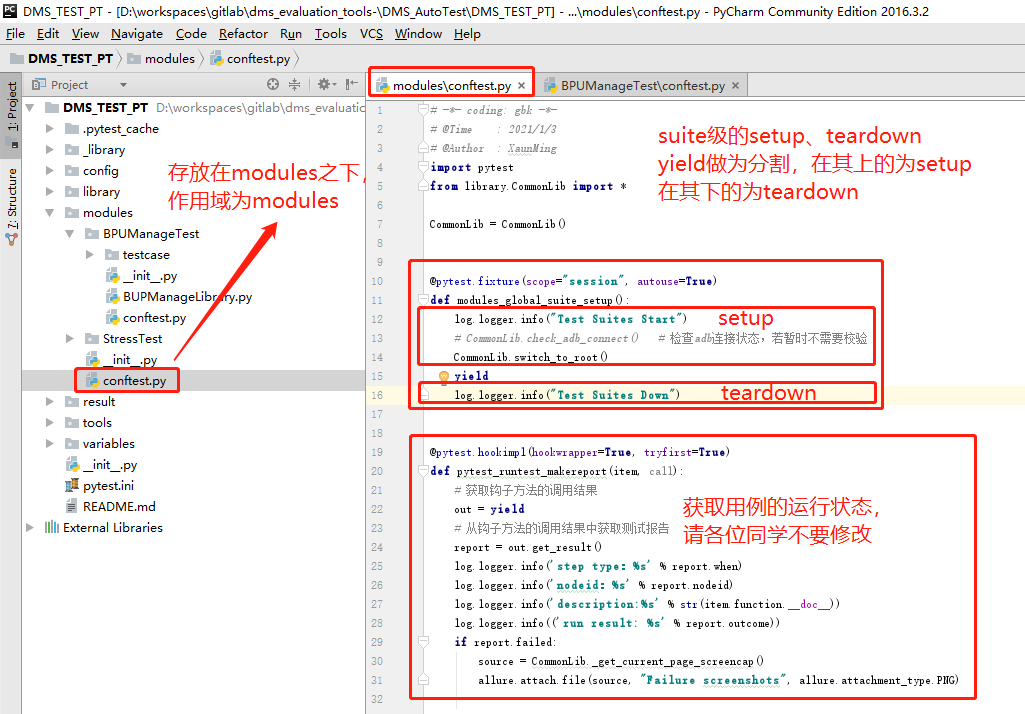
function：函数、方法级别的，每个函数、方法前都会执行；

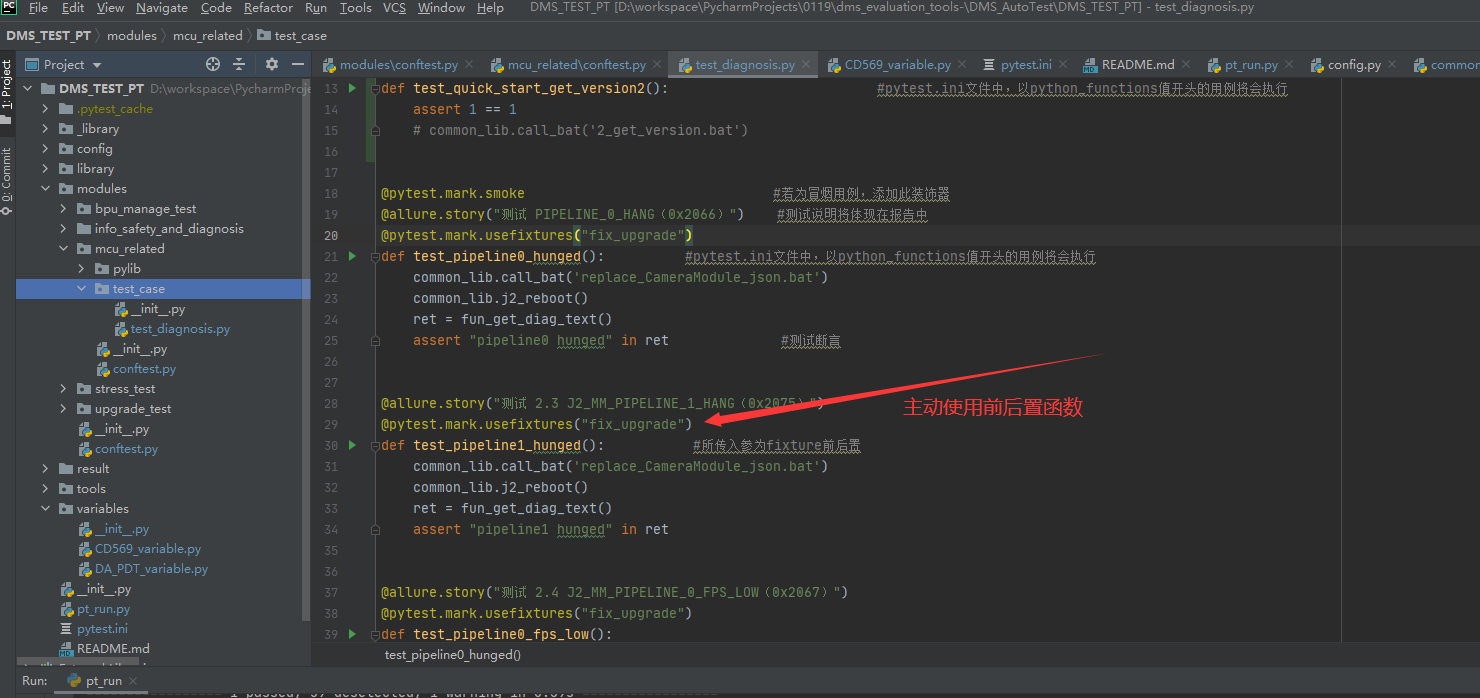
class：类级别的，每个类之前都会执行；

module：可以实现多个.py跨文件共享前置，每个模块（文件）之前都会执行；

package：可以实现整个项目的前置后置，和session的区别待研究；

session：多个.py跨文件使用一个session来完成多个用例，只在作用域前执行一次；





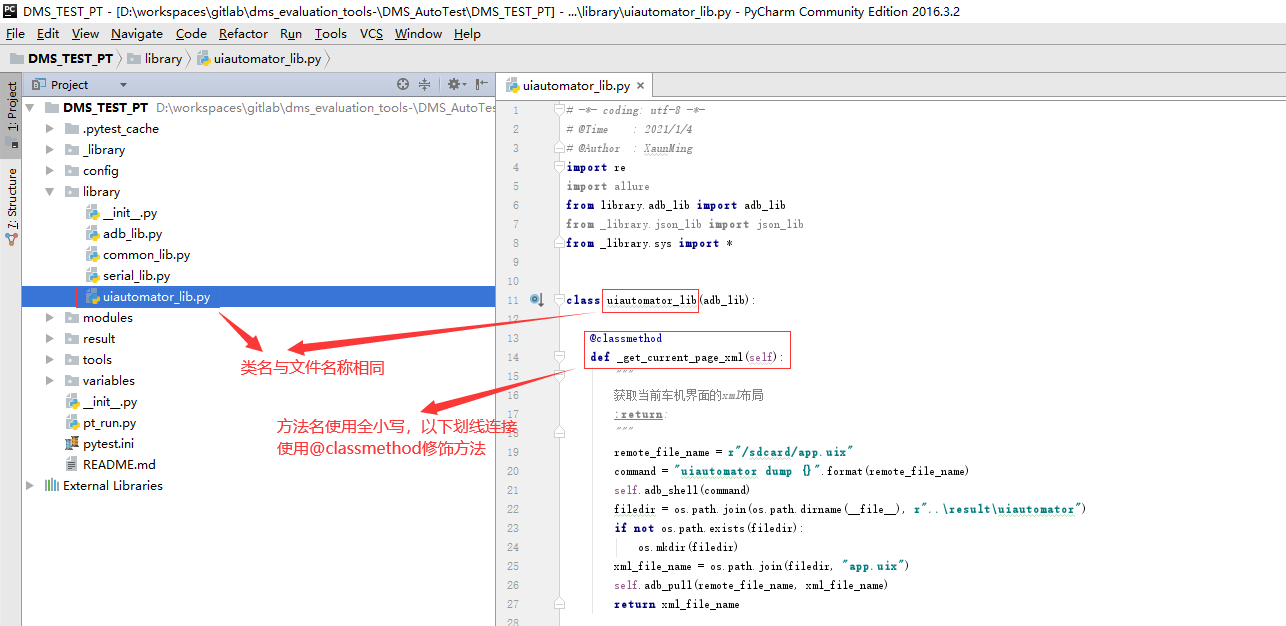
## 项目的公共接口（${name}\_lib）

目录：DMS\_TEST\_PT/ library/

1、文件名：接口名+\_lib.py

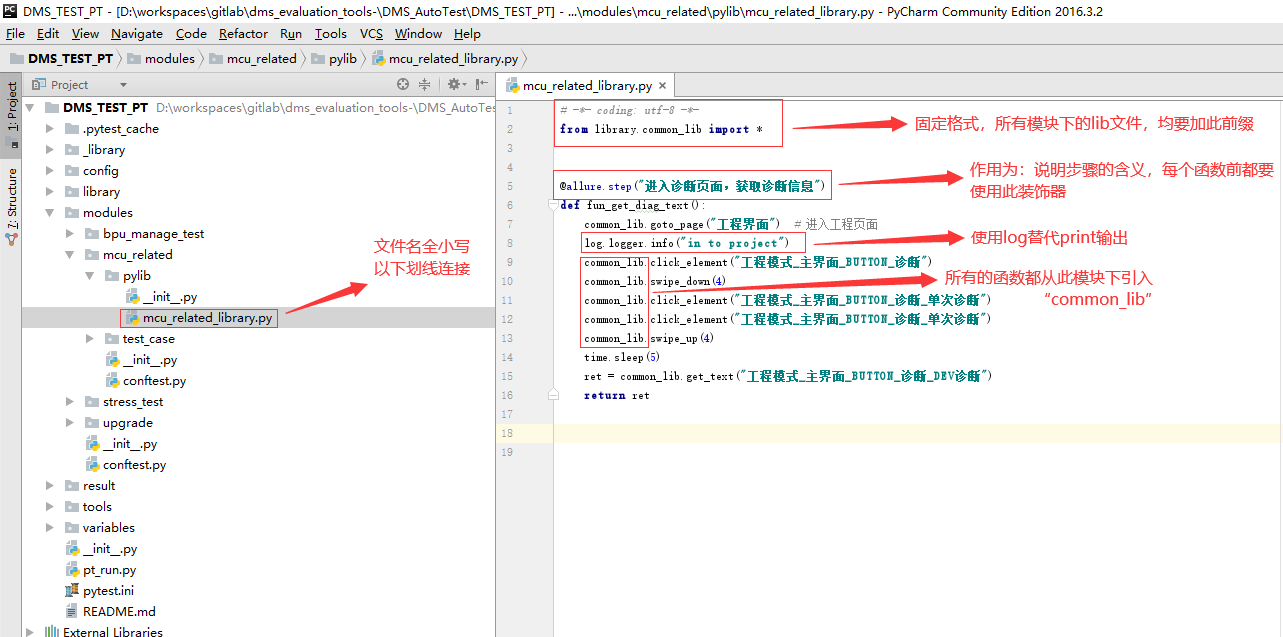
2、类名与文件名相同

3、方法名：使用装饰@classmethod，且方法名均为小写，以下划线连接



## 模块的公共接口（${module\_name}\_library）

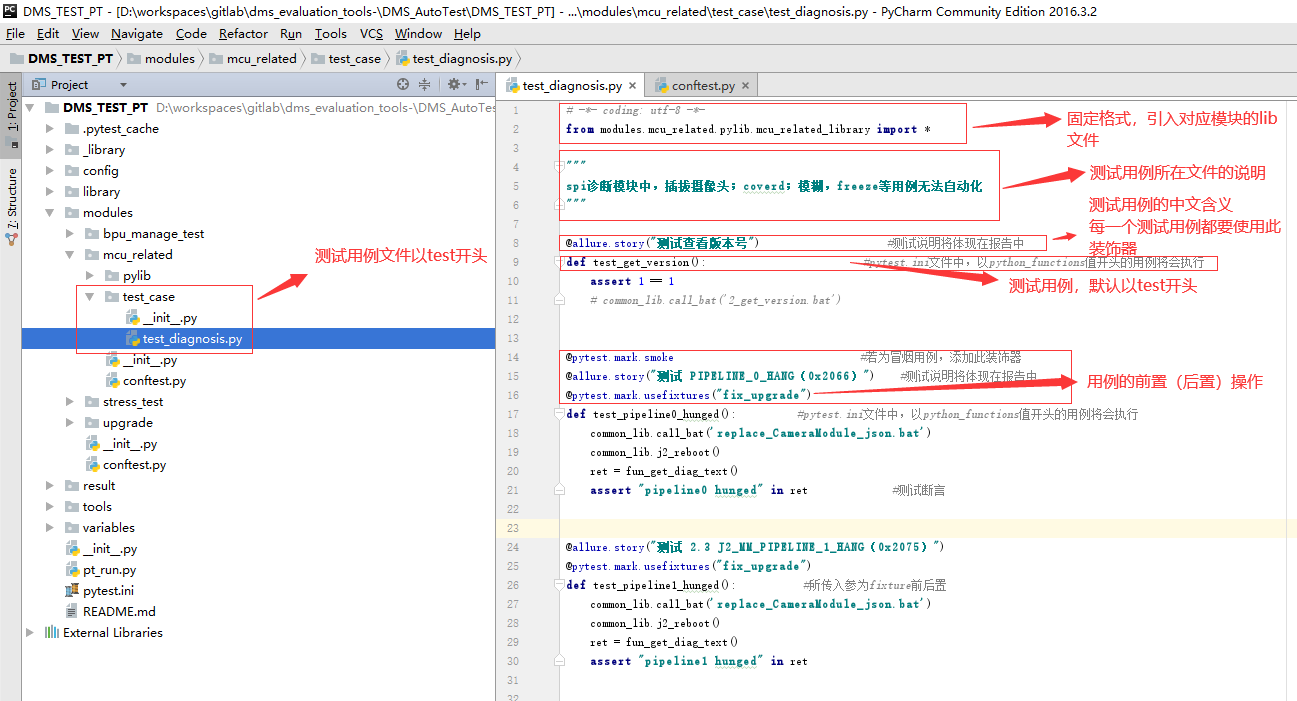
当你开始编写一个新的模块的时候，你需要在modules下面以模块名称创建一个新的文件夹，里面存放有testcase、pylib、conftest.py，其中${module\_name}\_library.py存放的为专属于你这个模块的公共接口，请按照我们固定的格式来编写，可以参考一下mcu\_related \_library.py。



## 测试用例（testcase）

在testcase下请你根据不同的业务类别将该模块下的测试用例放置到不同的py文件下面，当然如果你的用例较少或者你认为他们就是一个类别的，也可以像我一样，都放置到一个py文件下，若你认为这个模块较复杂，也可以在testcase下面建立多级目录，我们的目的是更加方便的管理用例，只要达到这个目的就可以，当然在你编写的时候也请遵寻固定的格式，若不是很清楚可以参考一下mcu\_related下的testcase。

一个用例可以使用多个装饰器来修饰，他们都有不同的功能。



# 附件

## git使用

