# 创建插件

参考：https://blog.csdn.net/wg8ofk08/article/details/130944359

创建项目

选择项目位置

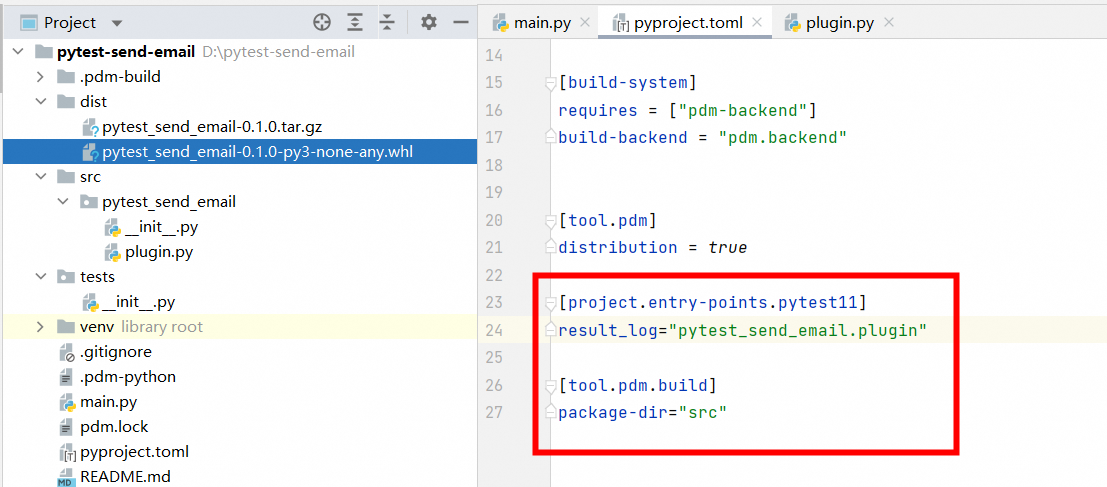
创建venv

安装pdm pip install pdm

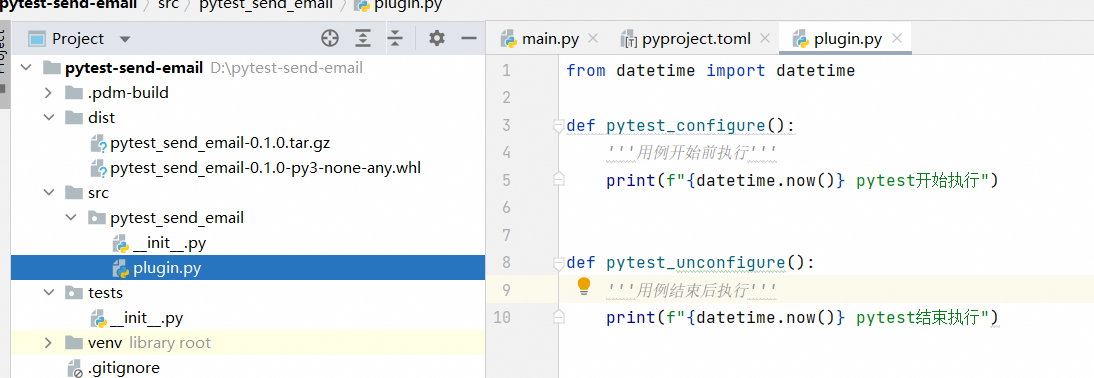
使用pdm初始化项目

Pdm init

配置



编写代码：



添加依赖： 执行命令 pdm add pytest

打包：执行命令 pdm build

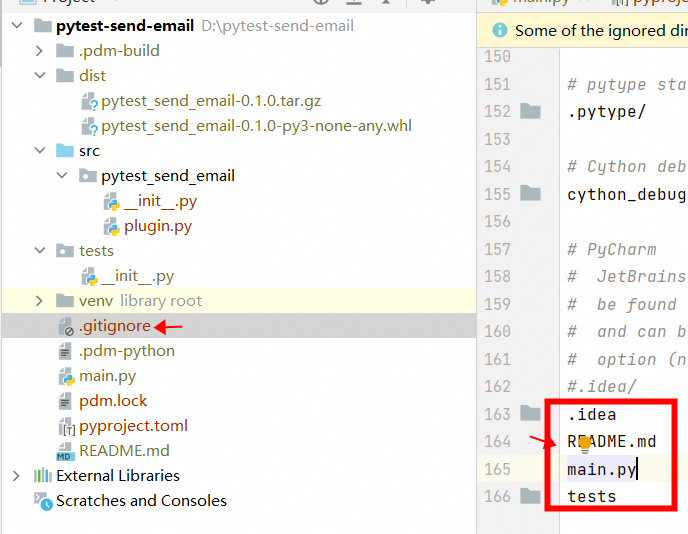
验证：将whl包复制粘贴到新的项目，执行pip install 包名

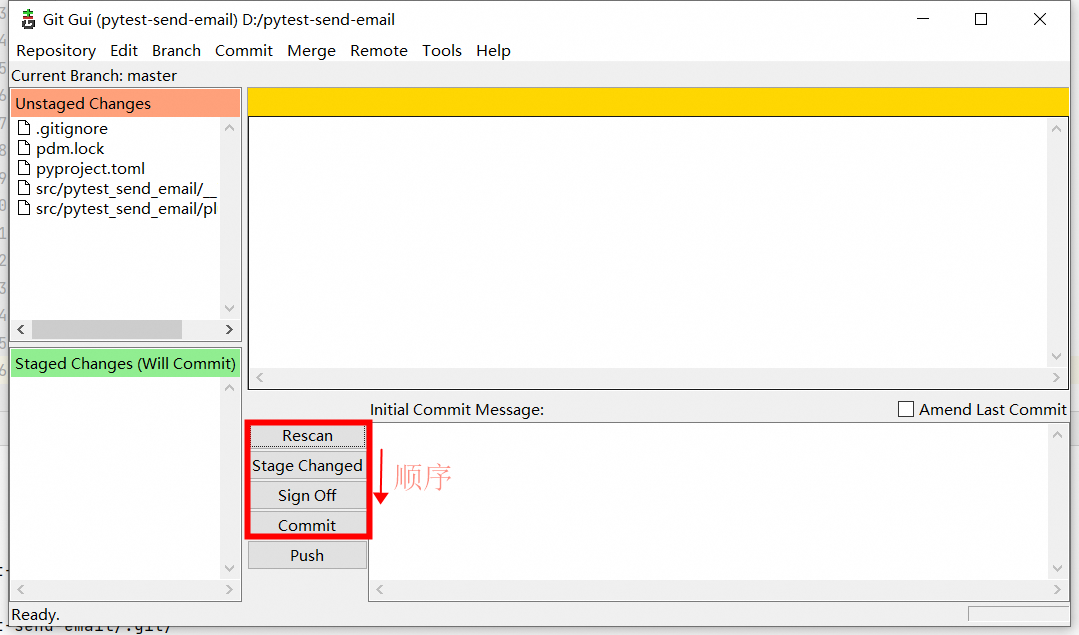
2.创建git 库 命令：git init

打开git gui

在根目录下的.gitignore文件配置需要过滤的文件，添加即可过滤

然后在gui里按照顺序提交

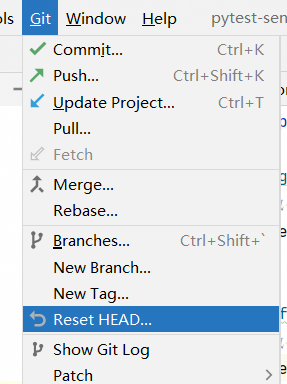




回退版本：

Reset-head

选择hard 及填上版本号提交



3.使用black，isort，flake8 代码质量检测

分别是：代码格式、import排序 、质量分析

添加依赖 pdm add black isort flake8 -d

方法1：black aa.py 终端执行命令如black+文件名，

方法2：一次性检测，如下

1配置脚手架：在pyproject.toml里添加

[tool.isort]  
profile="black"  
multi\_line\_output=3  
[tool.black]  
exclude='''  
/(  
 \.git  
| \.hg  
| \.tox  
| venv  
| buck-out  
| build  
| dist  
)/  
'''  
[tool.pdm.scripts]  
check={composite=["isort .","black .","flake8 ."]}  
test="pytest"

2在根目录添加.flake8文本文件输入

[flake8]  
max-line-length=88  
ignore=E501,E203,W503  
exclude=.git,\_\_pycache\_\_,hypothesis,.pytest\_cache,venv,.venv,build,dist

3都配置好后，只用执行pdm check 即可

3.代码编写

1.主要代码

from datetime import datetime  
import pytest  
import requests  
  
data = {  
 "passed": 0,  
 "failed": 0,  
}  
  
def pytest\_runtest\_logreport(report: pytest.TestReport):  
 *"""记录运行结果"""* if report.when == "call":  
 data[report.outcome] += 1  
  
def pytest\_collection\_finish(session: pytest.Session):  
 *"""用例加载完成之后执行，包含全部用例"""* data["total"] = len(session.items)  
  
def pytest\_configure():  
 *"""用例开始前执行"""* data["start\_time"] = datetime.now()  
 print(f"{datetime.now()} pytest开始执行")  
  
def pytest\_unconfigure():  
 *"""用例结束后执行"""* data["end\_time"] = datetime.now()  
 print(f"{datetime.now()} pytest结束执行")  
 data["run\_time"] = data["end\_time"] - data["start\_time"]  
 data["pass\_radio"] = data["passed"] / data["total"] \* 100  
 data["pass\_radio"] = f"{data['pass\_radio']:.2f}%"  
 *# assert data['total']==2* url = <https://oapi.dingtalk.com/robot/send?access_token=05493a7395f447d8e9a477dfc07aa85c73665145bdfbf14f51a8ab4ffd64b2ef>

content = f"""  
 自动化测试结果 \n>  
 测试开始时间：{data["start\_time"]}\n>  
 测试执行时间：{data["run\_time"]}\n>  
 用例数：{data['total']}\n>  
 通过：<font color=#00FF00>\*\*{data['passed']}\*\*</font>\n>  
 失败：\*\*<font color=#FF0000>{data['failed']}</font>\*\*\n>  
 通过率：{data['pass\_radio']}"""  
 requests.post(  
 url,  
 json={  
 "msgtype": "markdown",  
 "markdown": {"title": "自动化测试结果", "text": content},  
 },  
 )

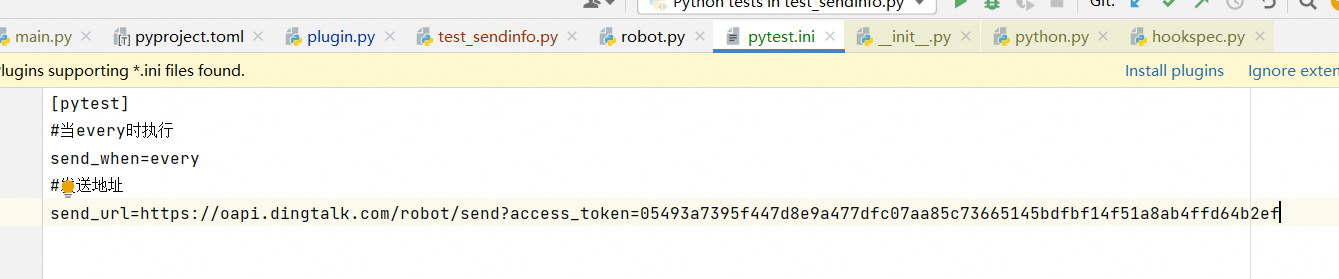
2.如果运行失败缺少插件



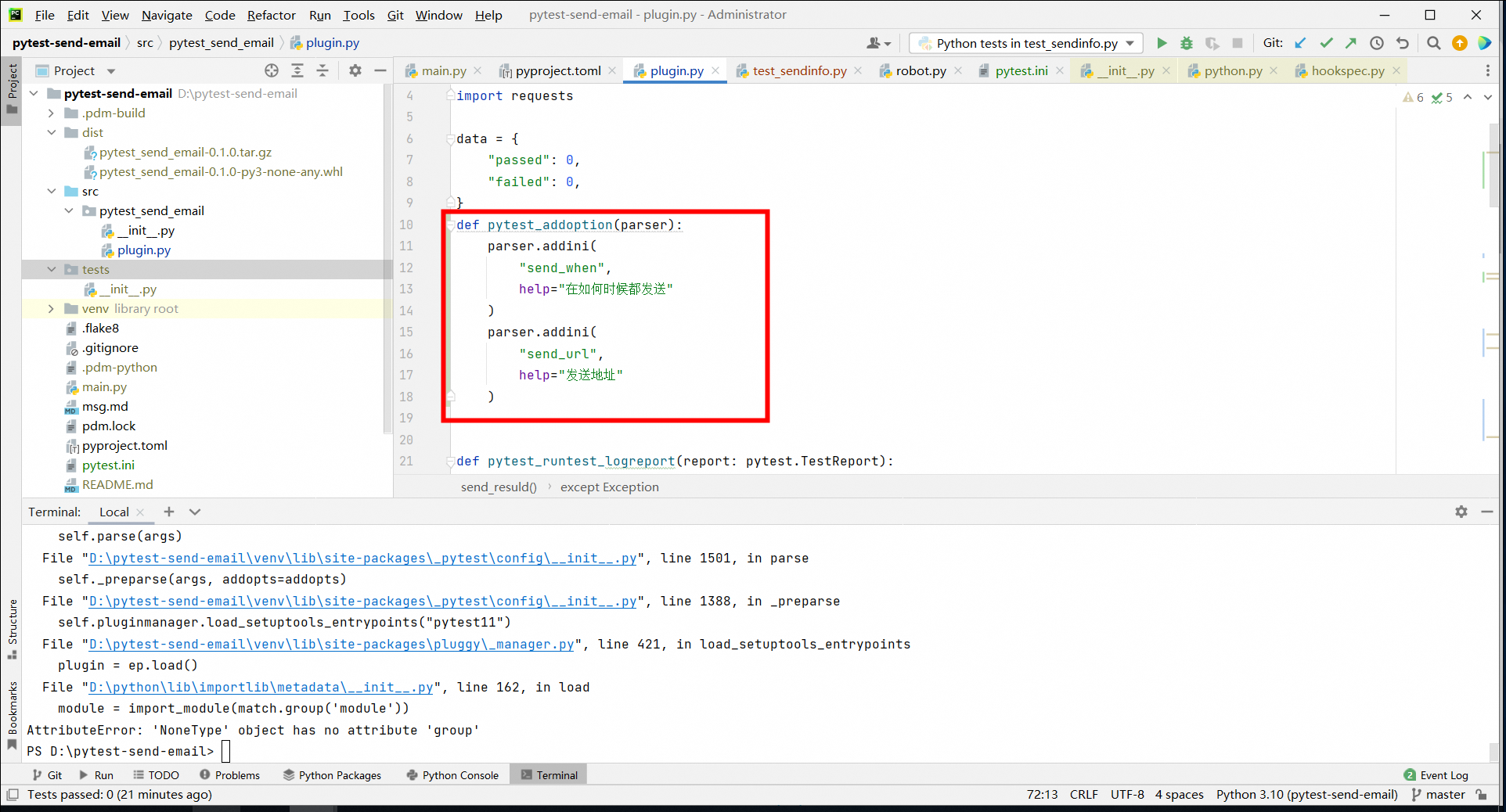
3.可以在新的文件打包安装测试

## 4.设置配置

1.添加配置项



2.识别配置项



3.使用配置，补全代码-红色为新加内容

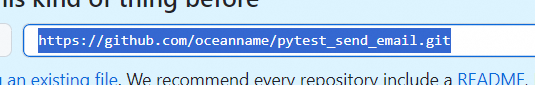
def pytest\_configure(config: pytest.Config):  
 *"""用例开始前执行"""* data["start\_time"] = datetime.now()  
 print(f"{datetime.now()} pytest开始执行")  
 data["send\_when"]=config.getini("when") #将提取的值写入全局变量，供其他地方也可以使用  
 data["send\_url"]=config.getini("url")  
  
def pytest\_unconfigure():  
 *"""用例结束后执行"""* data["end\_time"] = datetime.now()  
 print(f"{datetime.now()} pytest结束执行")  
 data["run\_time"] = data["end\_time"] - data["start\_time"]  
 data["pass\_radio"] = data["passed"] / data["total"] \* 100  
 data["pass\_radio"] = f"{data['pass\_radio']:.2f}%"  
 *# assert data['total']==2* send\_resuld()  
def send\_resuld():  
 if not data["send\_url"]:  
 return  
 if data["send\_when"]!="every":  
 return  
 url = data["send\_url"]  
 content = f"""  
 自动化测试结果 \n>  
 测试开始时间：{data["start\_time"]}\n>  
 测试执行时间：{data["run\_time"]}\n>  
 用例数：{data['total']}\n>  
 通过：<font color=#00FF00>\*\*{data['passed']}\*\*</font>\n>  
 失败：\*\*<font color=#FF0000>{data['failed']}</font>\*\*\n>  
 通过率：{data['pass\_radio']}"""  
 try: #try的作用，防止网络不行的情况下  
 requests.post(  
 url,  
 json={  
 "msgtype": "markdown",  
 "markdown": {"title": "自动化测试结果", "text": content},  
 },  
 )  
 except Exception:  
 pass

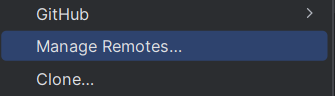
4. 添加测试用例

## 5.推送到远程仓库

创建仓库

登录github，new创建仓库，设置名称，create

将添加到git

或者在pycharm的git下找到添加

之后就可以push上传到github6.使用

6．使用  
在新的文件中使用，要添加pytest.ini文件，

[pytest]  
#当every时执行  
send\_when=every  
#发送地址  
send\_url=https://oapi.dingtalk.com/robot/send?access\_token=05493a7395f447d8e9a477dfc07aa85c73665145bdfbf14f51a8ab4ffd64b2ef