在 Windows 10 下删除 Pipenv 已安装包的详细步骤

方法一：卸载单个包（推荐）

```bash

# 1. 进入包含 Pipfile 的项目目录

cd your\_project\_folder

2. 卸载指定包（例如卸载 requests 包）

pipenv uninstall requests

3. 更新锁文件（可选但推荐）

pipenv lock

方法二：卸载所有包（清空虚拟环境）

1. 进入项目目录

cd your\_project\_folder

2. 删除所有已安装包（保留 Pipfile）

pipenv uninstall --all

方法三：完全删除虚拟环境（彻底清除）

1. 进入项目目录

cd your\_project\_folder

2. 删除整个虚拟环境

pipenv --rm

3. 手动删除包记录（可选）

删除 Pipfile.lock 文件（下次 install 会重新生成）

删除 Pipfile 中的 [packages] 部分内容

关键操作说明

1. 卸载后检查状态

查看已安装包列表

pipenv graph

检查包依赖关系

pipenv check

2. 清理残留文件

卸载后可能需要手动操作：

1. 删除 `\_\_pycache\_\_` 文件夹

2. 清除项目中的 `.pytest\_cache`（如果使用 pytest）

3. 删除 `\*.egg-info` 文件夹（某些包会生成）

3. 特殊情况处理

\*\*场景：包卸载失败\*\*

强制卸载

pipenv uninstall --skip-lock package\_name

手动编辑 Pipfile 后重试

1. 打开 Pipfile 删除对应包名

2. 执行 pipenv clean

\*\*场景：需要保留某些依赖\*\*

选择性卸载（例如保留测试依赖）

pipenv uninstall package\_name --keep-outdated

注意事项

1. \*\*权限问题\*\*：

- 以管理员身份运行 CMD/PowerShell

- 如果报权限错误，添加 `--verbose` 查看详情：

```bash

pipenv uninstall package\_name --verbose

2. \*\*依赖关系\*\*：

- 卸载主包时其依赖不会自动移除（使用 `pipenv clean` 清理孤立依赖）

pipenv clean

3. \*\*环境隔离\*\*：

- 确保在正确的虚拟环境中操作（命令行提示符前应有 `(your\_project\_name)` 标识）

- 验证环境位置：

pipenv --venv

4. \*\*文件恢复\*\*：

- 误删后可通过 `pipenv install` 重新安装 Pipfile 中的包

- 使用版本控制恢复 Pipfile.lock：

git checkout -- Pipfile.lock

> 建议在操作前备份 Pipfile 和 Pipfile.lock 文件，如需完全重置环境，可删除整个虚拟环境后重新安装：

> `pipenv --rm && pipenv install`