笔记

> 版权：智泊AI
>
> 作者：Jeff

Python虚拟环境

什么是虚拟环境

虚拟环境（Virtual Environment）是一个 **独立 隔离** 的 Python 运行环境，它能够独立于真实环境存在，并且可以同时有多个互相独立的Python虚拟环境，每个虚拟环境都可以营造一个干净的开发环境，对于项目的依赖、版本的控制有着非常重要的作用。你可以在不同项目中使用不同的 Python 版本和依赖库，而不会相互影响。

虚拟环境的作用

1. **避免依赖冲突**

- 不同项目可能需要不同版本的库。
- 直接在系统环境安装多个版本的库，容易导致兼容性问题，而虚拟环境可以让每个项目独立运行所需的版本。

2. 防止污染全局环境

- 如果所有 Python 项目都共用一个全局环境，安装或卸载某个库可能影响其他项目的运行。
  使用虚拟环境，每个项目的依赖都在自己的目录中，不会污染全局 Python 目录。

3. 保持项目的可移植性

- 在团队协作或部署到服务器时，可以迁移环境，使用 requirements.txt 记录所有依赖，其他人可以用虚拟环境安装相同的依赖，保证代码能在不同环境下运行一致。

官方推荐虚拟环境（venv）

Python 官方提供的 venv 方案十分好用，而且Python 3.3 以上版本自带 venv， 我们大多数情况下使用。

1. 检查venv是否可用

```
python -m venv --help
# 若显示帮助信息说明功能正常
```

2. **创建虚拟环境：**

```
# 创建虚拟环境（推荐在项目根目录创建）
python -m venv .venv
```

3. **激活虚拟环境**

```
# windows
.venv\Scripts\activate
# 激活后提示符变为 (.venv) PS C:\path>

# mac
source .venv/bin/activate
# 激活后提示符变为 (.venv) user@host:~$
```

4. 查看激活状态

```
which python  # Linux/macOS系统
where python  # Windows系统
# 应显示虚拟环境路径下的python
```

5. **退出虚拟环境**

```
deactivate
```

6. 删除虚拟环境（慎用）

```
# 直接删除环境目录即可  Windows

rm -rf .venv   # Linux/macOS
```

7. **环境迁移**

```
# 导出依赖包（已写项目）
pip freeze > requirements.txt

# 安装依赖包（在新环境中）
pip install -r requirements.txt
```