

搜索 Ctrl+K

聊天

语音

Imagine

项目

历史记录

本周

使用uv管理Python版本方法

December

YOLOv8在城市背景下的对比分析

Tr26x5-7 Thread Dimensions

November

神经网络分布与变化分析

OpenMV: Machine Vision D

PowerShell Conda Activation

October

STM32F4烧录失败解决方法

更改UV Python安装路径方法

使用UV环境编译Python的NU

Chrome参数传递方法详解

查看全部

怎么设置uv的python版本

概活

使用uv管理Python版本主要有两种方式：

- 全局默认版本：设置uv默认使用哪个Python（uv default python）
- 项目局部版本：在具体项目中指定使用的Python版本（uv venv + pyproject.toml或.python-version文件）

下面详细说明常用方法（基于最新版uv 0.4+，2025年12月情况）

1. 安装并管理Python版本（uv自带python管理功能）

uv本身可以直接下载和管理Python版本，不需要再装pyenv。

```
Bash
# 列出uv可用的Python版本
uv python list

# 下载并安装特定版本（例如3.12.7、3.11.10等）
uv python install 3.12.7
uv python install 3.11.10
uv python install 3.13.0 # 甚至可以装最新的3.13

# 设置全局默认Python版本（以后uv venv默认用这个）
uv python set-default 3.12.7
```

2. 为当前项目指定Python版本（推荐方式）

方法A：最简单，直接在项目根目录运行（会自动创建.python-version文件）

```
Bash
# 进入项目目录
cd my_project

# 指定项目使用Python 3.12.7（会自动下载如果没有）
uv python pin 3.12.7
```

这会生成一个.python-version文件，内容就是3.12.7，以后在这个目录及子目录运行uv venv或uv pip install时都会自动使用这个版本。

方法B：在pyproject.toml中声明（官方推荐，团队协作更规范）

```
toml
[project]
# 其他配置...
requires-python = ">=3.12, <3.13" # 或者精确版本 = "3.12.7"

# 或者更精确地使用特定patch版本（uv独有功能）
[tool.uv]
python = "3.12.7"
```

然后在项目里执行任意uv命令，它都会自动使用指定的版本。

3. 手动创建虚拟环境时指定版本

```
Bash
# 明确指定Python版本创建虚拟环境
uv venv --python 3.12.7 -venv

# 或者使用已下载的版本别名
uv venv --python 3.12 # 会自动选择最新的3.12.x
```

4. 查看当前使用的Python版本

```
Bash
uv python list --only-installed # 已安装的版本
uv python find 3.12 # 当前环境能找到的3.12版本
uv run python -version # 当前项目实际使用的版本
```

总结最常用的一行命令（推荐）

```
Bash
# 进入项目后，只需这一行就搞定版本锁定
uv python pin 3.12.7
```

以后团队成员clone项目后，只需运行uv sync或uv pip install -r requirements.txt就会自动下载并使用3.12.7，完全不需要手动装Python。

这样设置后，uv的所有命令（venv、pip、run、sync等）都会自动尊重你设置的Python版本，非常方便。

我明把3.11.3、3.12.11和3.13.7安装到D:\uv\python了，但是运行uv python list --only-installed只显示了(base) PS C:\Users\33572\Desktop>uv python list --only-installed cpython-3.13.5-windows-x86_64-none D:\anaconda3\python.exe

概活

你的问题很可能出现在手动“安装”Python版本的方式上：uv的uv python install命令会自动下载并管理Python到其默认目录（Windows上通常是%USERPROFILE%\local\share\uv\python），而手动复制或放置文件到D:\uv\python后，uv无法自动“发现”它们（因为它们不是uv管理的）。结果，uv python list --only-installed只显示了系统发现的Anaconda Python（路径D:\anaconda3\python.exe，版本3.13.5）。解决方案：设置自定义安装目录环境变量UV_PYTHON_INSTALL_DIR=D:\uv\python，然后用uv python install重新安装版本；同时用--managed-python标志过滤列表，只显示uv管理的版本。

详细解释和步骤

1. 为什么只显示Anaconda的Python？

- uv在列出安装版本时，会优先“发现”系统Python（包括Anaconda、官方安装器、Windows注册表中的Python等）。你的Anaconda base环境（3.13.5）被uv检测到并显示，因为在PATH或注册表中。
- 手动放置文件（如果你把3.11.3、3.12.11、3.13.7的Python放到D:\uv\python）不会被uv自动管理或发现。uv只会认为它超过了uv python install下载/安装的“managed”版本，这些版本有特定目录结构（如cpython-3.12.11-windows-x86_64-none）；并在Windows注册表中注册（符合PEP 514）。
- only-installed标志显示所有uv能“找到”的已安装版本（系统+managed），但如果你没有managed版本，它就只剩系统Anaconda了。
- 注意：你的提示符显示(base)，说明你在Anaconda base环境中运行命令，这可能进一步强化uv对Anaconda的偏好（uv会尊重当前激活的环境）。

2. 如何正确安装自定义路径D:\uv\python

uv支持通过环境变量UV_PYTHON_INSTALL_DIR指定managed Python的安装目录。设置后，uv python install会下载并安装到该路径下（路径必须已存在，或uv会创建子目录）。

步骤（在PowerShell或CMD中执行）：

```
PowerShell
# 步骤1：创建目标目录（如果不存在）
mkdir D:\uv\python -Force

# 步骤2：设置环境变量（临时设置，只影响当前会话；想永久，添加用户环境变量）
$env:UV_PYTHON_INSTALL_DIR = "D:\uv\python"

# 步骤3：验证设置（可选，显示uv的存储路径）
uv python dir

# 步骤4：重新安装你的版本（uv会下载到自定义目录，如果已存在会跳过或用--reinstall强制）
uv python install 3.11.3
uv python install 3.12.11
uv python install 3.13.7 # 注意：你之前说3.13.7，但列表显示3.13.5；用你想要的版本

# 输出示例：Installed Python 3.11.3 to: D:\uv\python\cpython-3.11.3-windows-x86_64-none
```

注意：

- 如果目录有非uv结构的文件（如你手动放的），建议先清空D:\uv\python，避免冲突。
- 下载需要网络；uv会自动选择Windows x86_64架构。
- 如果想强制重新安装（即使版本已存在），加--reinstall：uv python install 3.11.3 --reinstall
- 永久设置环境变量：在Windows设置>系统>关于>高级系统设置>环境变量>用户变量>新建UV_PYTHON_INSTALL_DIR=D:\uv\python，然后重启终端。

3. 如何正确列出uv管理的版本

默认的uv python list --only-installed包括系统+managed版本。为了只显示uv管理的（忽略Anaconda），用--managed-python标志：

```
PowerShell
# 只显示uv管理的版本（你的自定义目录下的）
uv python list --only-installed --managed-python
```

输出示例：
cpython-3.11.3-windows-x86_64-none D:\uv\python\cpython-3.11.3-windows-x86_64-
none\python.exe
cpython-3.12.11-windows-x86_64-none D:\uv\python\cpython-3.12.11-windows-x86_64-
none\python.exe
cpython-3.13.7-windows-x86_64-none D:\uv\python\cpython-3.13.7-windows-x86_64-
none\python.exe

其他有用标志：

- no-managed-python：只显示系统版本（如Anaconda）。
- 无标志：显示所有（包括可下载的）。
- 如果还是只显示Anaconda，检查是否在Anaconda环境中（运行conda deactivate退出base环境，或重新启动终端）。

4. 验证和使用

验证安装：

```
PowerShell
# 检查具体版本路径
uv python find 3.12.11 # 应该返回D:\uv\python\...\python.exe

# 测试创建venv用该版本
uv venv -python 3.12.11 .venv
.\venv\Scripts\Activate.ps1 # 激活
python --version # 应显示3.12.11
```

如果想优先uv managed版本（而非系统Anaconda）：在pyproject.toml或uv配置中设置python_preference = "managed"（编辑%APPDATA%\uv\config.toml）。

常见问题排查

- 还是不显示？运行uv python list -v（verbose）看调试输出，检查是否搜索到D:\uv\python。
- Anaconda干扰：如果你不想用Anaconda，考虑用conda deactivate或完全卸载base环境；但uv可以共存。
- 版本不匹配：确保版本号精确（如3.12.11而非3.12）；用uv python list（或--only-installed）看所有下载版本。
- 更新uv：确保uv是最新版（uv self update），因为Python管理功能在迭代中。
- 如果问题继续，检查GitHub issues（如#10049），类似手动安装不被识别的问题常见。

这样设置后，你的自定义目录就会被uv正确管理，列表也会正常显示。もし还有错误输出，贴出来我再帮调试！

我运行uv python set-default 3.11.13显示(base) PS C:\Users\33572\Desktop>uv python set-default 3.11.13
error: unrecognized subcommand 'set-default'
Usage: uvexe python [OPTIONS] <COMMAND>
For more information, try '-help'.

概活

错误是因为uv python set-default不是有效的命令（uv中不存在这个子命令）。uv不支持直接设置“全局默认Python版本”的命令（如pyenv的global），而是通过项目级pinning（使用uv python pin创建.python-version文件）或命令级指针（使用--pyproject标志）来管理版本。你的Anaconda base环境（3.13.5）会优先使用uv优先级，即使pinning后，如果在base环境中运行，uv可能仍偏好系统版。始终conda deactivate后再用uv。

如果想完全忽略系统Python，用环境变量：

```
PowerShell
$env:UV_PYTHON_PREFERENCE = "managed" # 只用uv管理的版本
uv python list --only-installed --managed-python # 只显示你的D:\uv\python中的
```

永久设置：添加用户环境变量UV_PYTHON_PREFERENCE=managed。

常见问题排查

- 版本不存在：用uv python list找最近版，如uv python pin 3.11（自动用最新3.11.x）。
- 还是用Anaconda：运行uv python find 3.11.13检查路径；如果返回Anaconda路径，重启终端或检查PATH环境变量（Anaconda的Scripts目录可能在里面）。
- 更新uv：确保uv是最新版（uv self update），因为Python管理在0.4+版本迭代。
- 如果pinning后venv仍错，用--force：uv venv -python 3.11.13 --force。

这样设置后，uv的所有命令（如uv sync、uv pip install）都会自动用3.11.13，而无需每次指定。项目clone后，团队只需uv sync即可复现环境。如果有具体错误输出，贴出来我再帮调试！

我运行uv python find显示(base) PS C:\Users\33572\Desktop>uv python find
set-default 3.11.13
error: unrecognized subcommand 'set-default'
Usage: uvexe python [OPTIONS] <COMMAND>
For more information, try '-help'.

概活

uv没有“全局默认Python”这个概念（不像pyenv有global），它只有用户级默认版本（通过uv python pin --global命令）。

查看是否设置成功最直接的方法是检查这个文件是否存在，以及在没有任何项目级.python-version的目录下运行uv python find看它实际使用哪个版本。

一键验证（推荐3步搞定）

```
PowerShell
# 1. 查看用户级全局.python-version是否存在以及内容
Get-Content $env:APPDATA\uv\python-version -ErrorAction SilentlyContinue

# 或者uv自己的命令（uv 0.4.21+ 支持）
uv python pin --show-global
```

2. 在一个干净目录（没有任何.python-version文件的目录）下查看uv实际会用哪个版本
随便新建一个文本文件测试
mkdir C:\temp\uv-test-global
cd C:\temp\uv-test-global

uv python find # + 这就是当前“全局默认”生效后的版本
或者uv的命令
uv python find --no-project

3. 再跑一次创建虚拟环境验证（真真真）
uv venv .venv # 不要任何--python参数
.\venv\Scripts\python --version # 看这里是不是你想要的全局版本

正确设置全局默认版本的方法（补全一下）

```
PowerShell
# 确保你想用的版本已经安装好
uv python install 3.11.10

# 设置用户级全局默认（只执行一次）
uv python pin 3.11.10 --global

# 再次验证
uv python pin --show-global # + 应该输出3.11.10
```

执行完毕，任何没有自定义.python-version文件的项目都会自动使用3.11.10。

常见误区和额外检查方法总结

检查方式	命令	说明
查看全局文件内容	Get-Content "\$env:APPDATA\uv\python-version"	存在即设置成功
自带显示全局	uv python pin --show-global	最新uv支持
当前生效版本（无项目文件）	uv python find --no-project	最准确，推荐
创建venv时实际使用的版本	uv venv -> 跑之后 python -V	实际效果验证
列出所有uv管理的版本	uv python list --only-installed --managed-python	确认版本是否真的被uv管理

只要第1步看到文件是你想要的版本，并且第2步uv python find --no-project也显示相同版本，就说明全局默认设置成功了。

如何Can Grok help?