Introduction 介绍
Getting started 开始
Installation 安装
First steps 第一步
Features 特征
Getting help 获取帮助
Guides 指南
Installing Python 安装 Python
Running scripts 运行脚本
Using tools 使用工具
Working on projects 从事项目
Publishing packages 发布包
Migration 迁移 >
Integrations 集成 >
Concepts 概念
Projects 项目 >
Tools 工具
Python versions Python 版本
Configuration files 配置文件
Package indexes 包索引
Resolution 分辨率
Build backend 构建后端
Authentication 认证
Caching 缓存
Preview features 预览功能
The pip interface pip 接口 >
Reference 参考

Running tools 运行工具
Commands with different package names
具有不同包名称的命令
Requesting specific versions
请求特定版本
Requesting extras
请求额外服务
Requesting different sources
请求不同的来源
Commands with plugins
带有插件的命令
Installing tools 安装工具
Upgrading tools 升级工具
Requesting Python versions
请求 Python 版本
Legacy Windows Scripts
旧版 Windows 脚本
Next steps 后续步骤

Guides 指南

Using tools 使用工具

Many Python packages provide applications that can be used as tools. uv has specialized support for easily invoking and installing tools.

许多 Python 包提供了可用作工具的应用程序。UV 具有专门的支持，可轻松调用和安装工具。

Running tools 运行工具

The `uvx` command invokes a tool without installing it.

`uvx` 命令调用工具而不安装它。

For example, to run `ruff`:

例如，要运行 `ruff`：

```
$ uvx ruff
```

> **Note 注意**
>
> This is exactly equivalent to:
>
> 这完全等同于：
>
> ```
> $ uv tool run ruff
> ```
>
> `uvx` is provided as an alias for convenience.
>
> 为方便起见，`uvx` 作为别名提供。

Arguments can be provided after the tool name:

可以在工具名称后面提供参数：

```
$ uvx pycowsay hello from uv

  -------------
< hello from uv >
  -------------
   \   ^__^
    \  (oo)\_______
       (__)\       )\/\
           ||----w |
           ||     ||
```

Tools are installed into temporary, isolated environments when using `uvx`.

使用 `uvx` 时，工具被安装到临时的隔离环境中。

> **Note 注意**
>
> If you are running a tool in a *project* and the tool requires that your project is installed, e.g., when using `pytest` or `mypy`, you'll want to use `uv run` instead of `uvx`. Otherwise, the tool will be run in a virtual environment that is isolated from your project.
>
> 如果你在*项目*中运行一个工具，并且该工具需要安装你的项目，例如，在使用 `pytest` 或 `mypy` 时，你需要使用 `uv run` 而不是 `uvx`。否则，该工具将在与项目隔离的虚拟环境中运行。
>
> If your project has a flat structure, e.g., instead of using a `src` directory for modules, the project itself does not need to be installed and `uvx` is fine. In this case, using `uv run` is only beneficial if you want to pin the version of the tool in the project's dependencies.
>
> 如果您的项目具有扁平结构，例如，无需将 `src` 目录用于模块，则不需要安装项目本身，并且 `uvx` 就可以了。在这种情况下，使用 `UV 运行` 仅在要将工具版本固定在项目的依赖项中时才有用。

Commands with different package names
具有不同包名称的命令

When `uvx ruff` is invoked, uv installs the `ruff` package which provides the `ruff` command. However, sometimes the package and command names differ.

调用 `uvx ruff` 时，uv 会安装提供 `ruff` 命令的 `ruff` 包。但是，有时包和命令名称不同。

The `--from` option can be used to invoke a command from a specific package, e.g., `http` which is provided by `httpie`:

`--from` 选项可用于从特定包中调用命令，例如 `httpie` 提供的 `http`：

```
$ uvx --from httpie http
```

Requesting specific versions
请求特定版本

To run a tool at a specific version, use `command@<version>`:

若要在特定版本下运行工具，请使用 `command@<version>`：

```
$ uvx ruff@0.3.0 check
```

To run a tool at the latest version, use `command@latest`:

要运行最新版本的工具，请使用 `command@latest`：

```
$ uvx ruff@latest check
```

The `--from` option can also be used to specify package versions, as above:

`--from` 选项也可用于指定包版本，如上所示：

```
$ uvx --from 'ruff==0.3.0' ruff check
```

Or, to constrain to a range of versions:

或者，要限制为一系列版本：

```
$ uvx --from 'ruff>0.2.0,<0.3.0' ruff check
```

Note the `@` syntax cannot be used for anything other than an exact version.

请注意，`@` 语法不能用于精确版本以外的任何内容。

Requesting extras 请求额外服务

The `--from` option can be used to run a tool with extras:

`--from` 选项可用于运行具有附加功能的工具：

```
$ uvx --from 'mypy[faster-cache,reports]' mypy --xml-report mypy_report
```

This can also be combined with version selection:

这也可以与版本选择相结合：

```
$ uvx --from 'mypy[faster-cache,reports]==1.13.0' mypy --xml-report mypy_report
```

Requesting different sources
请求不同的来源

The `--from` option can also be used to install from alternative sources.

`--from` 选项也可用于从替代源安装。

For example, to pull from git:

例如，要从 git 中提取：

```
$ uvx --from git+https://github.com/httpie/cli httpie
```

You can also pull the latest commit from a specific named branch:

您还可以从特定的命名分支中提取最新的提交：

```
$ uvx --from git+https://github.com/httpie/cli@master httpie
```

Or pull a specific tag:

或者拉取特定标签：

```
$ uvx --from git+https://github.com/httpie/cli@3.2.4 httpie
```

Or even a specific commit:

甚至是特定的提交：

```
$ uvx --from git+https://github.com/httpie/cli@2843b87 httpie
```

Commands with plugins 带有插件的命令

Additional dependencies can be included, e.g., to include `mkdocs-material` when running `mkdocs`:

可以包含其他依赖项，例如，在运行 `mkdocs` 时包含 `mkdocs-material`：

```
$ uvx --with mkdocs-material mkdocs --help
```

Installing tools 安装工具

If a tool is used often, it is useful to install it to a persistent environment and add it to the `PATH` instead of invoking `uvx` repeatedly.

如果经常使用某个工具，将其安装到持久环境并将其添加到 `PATH` 而不是重复调用 `uvx`。

> **Tip 提示**
>
> `uvx` is a convenient alias for `uv tool run`. All of the other commands for interacting with tools require the full `uv tool` prefix.
>
> `UVX` 是 `UV 工具运行` 的便捷别名。与工具交互的所有其他命令都需要完整的 `uv 工具` 前缀。

To install `ruff`:

要安装 `ruff`：

```
$ uv tool install ruff
```

When a tool is installed, its executables are placed in a `bin` directory in the `PATH` which allows the tool to be run without uv. If it's not on the `PATH`, a warning will be displayed and `uv tool update-shell` can be used to add it to the `PATH`.

安装工具后，其可执行文件被放置在 `PATH` 中的 `bin` 目录中，这允许该工具在没有 uv 的情况下运行。如果它不在 `PATH` 上，则会显示警告，并且 `uv 工具 update-shell` 可用于将其添加到 `PATH`。

After installing `ruff`, it should be available:

安装 `ruff` 后，它应该可用：

```
$ ruff --version
```

Unlike `uv pip install`, installing a tool does not make its modules available in the current environment. For example, the following command will fail:

与 `uv pip install` 不同，安装工具不会使其模块在当前环境中可用。例如，以下命令将失败：

```
$ python -c "import ruff"
```

This isolation is important for reducing interactions and conflicts between dependencies of tools, scripts, and projects.

这种隔离对于减少工具、脚本和项目依赖项之间的交互和冲突非常重要。

Unlike `uvx`, `uv tool install` operates on a *package* and will install all executables provided by the tool.

与 `uvx` 不同，`uv tool install` 在*包*上运行，并将安装该工具提供的所有可执行文件。

For example, the following will install the `http`, `https`, and `httpie` executables:

例如，以下内容将安装 `http`、`https` 和 `httpie` 可执行文件：

```
$ uv tool install httpie
```

Additionally, package versions can be included without `--from`:

此外，包版本可以包含在不使用 `--from`：

```
$ uv tool install 'httpie>0.1.0'
```

And, similarly, for package sources:

同样，对于包源：

```
$ uv tool install git+https://github.com/httpie/cli
```

As with `uvx`, installations can include additional packages:

与 `uvx` 一样，安装可以包括其他包：

```
$ uv tool install mkdocs --with mkdocs-material
```

Upgrading tools 升级工具

To upgrade a tool, use `uv tool upgrade`:

要升级工具，请使用 `uv 工具升级` ：

```
$ uv tool upgrade ruff
```

Tool upgrades will respect the version constraints provided when installing the tool. For example, `uv tool install ruff >=0.3,<0.4` followed by `uv tool upgrade ruff` will upgrade Ruff to the latest version in the range `>=0.3,<0.4`.

工具升级将遵循安装工具时提供的版本约束。例如 `uv tool install ruff >=0.3,<0.4` 然后是 `uv 工具升级 ruff 将 ruff` 升级到 `>=0.3，<0.4` 范围内的最新版本。

To instead replace the version constraints, re-install the tool with `uv tool install`:

要替换版本约束，请使用 `uv tool install` 重新安装该工具：

```
$ uv tool install ruff>=0.4
```

To instead upgrade all tools:

要升级所有工具，请执行以下作：

```
$ uv tool upgrade --all
```

Requesting Python versions
请求 Python 版本

By default, uv will use your default Python interpreter (the first it finds) when running, installing, or upgrading tools. You can specify the Python interpreter to use with the `--python` option.

默认情况下，uv 在运行、安装或升级工具时将使用默认的 Python 解释器（它找到的第一个）。您可以指定要与 `--python` 一起使用的 Python 解释器 选择。

For example, to request a specific Python version when running a tool:

例如，要在运行工具时请求特定的 Python 版本：

```
$ uvx --python 3.10 ruff
```

Or, when installing a tool:

或者，在安装工具时：

```
$ uv tool install --python 3.10 ruff
```

Or, when upgrading a tool:

或者，在升级工具时：

```
$ uv tool upgrade --python 3.10 ruff
```

For more details on requesting Python versions, see the Python version concept page.

有关请求 Python 版本的更多详细信息，请参阅 Python 版本概念页。

Legacy Windows Scripts 旧版 Windows 脚本

Tools also support running legacy setuptools scripts. These scripts are available via `$(uv tool dir)\<tool-name>\Scripts` when installed.

工具还支持运行 旧版 SetupTools 脚本 。这些脚本在安装时可通过。 `$(uv tool dir)\<tool-name>\Scripts`

Currently only legacy scripts with the `.ps1`, `.cmd`, and `.bat` extensions are supported.

目前仅支持扩展名为 `.ps1`、`.cmd` 和 `.bat` 的旧脚本。

For example, below is an example running a Command Prompt script.

例如，下面是运行命令提示符脚本的示例。

```
$ uv tool run --from nuitka==2.6.7 nuitka.cmd --version
```

In addition, you don't need to specify the extension. `uvx` will automatically look for files ending in `.ps1`, `.cmd`, and `.bat` in that order of execution on your behalf.

此外，无需指定扩展名。`UVX` 将代表您自动查找以 `.ps1`、`.cmd` 和 `.bat` 结尾的文件，并按照该执行顺序。

```
$ uv tool run --from nuitka==2.6.7 nuitka --version
```

Next steps 后续步骤

To learn more about managing tools with uv, see the Tools concept page and the command reference.

要了解有关使用 uv 管理工具的更多信息，请参阅工具概念页面和命令参考 。

Or, read on to learn how to work on projects.

或者，继续阅读以了解如何处理项目 。