# **使用 pip 和 venv 在虚拟环境中安装软件包**

本指南讨论如何使用 标准库的虚拟环境工具 [venv](https://packaging.python.org/en/latest/key_projects/" \l "venv) 和安装包。 该指南介绍了如何：

创建和激活虚拟环境

准备点子

使用命令将软件包安装到虚拟环境中pip

使用和创建需求文件

注意

本指南适用于受支持的 Python 版本，当前为 3.8 和更高。

注意

本指南使用术语****“****包”来指代分发包，该[分发包](https://packaging.python.org/en/latest/glossary/" \l "term-Distribution-Package)通常从外部安装 主机。这与术语[“导入包](https://packaging.python.org/en/latest/glossary/" \l "term-Import-Package)”不同，后者指的是 在 Python 源代码中导入模块。

重要

本指南的先决条件是，您使用的是从 <<https://www.python.org/downloads/>>。如果您正在使用您的操作 系统的包管理器来安装 Python，请确保 Python 是 在继续执行这些步骤之前安装。

## **创建和使用虚拟环境**

### **创建新的虚拟环境**

[venv](https://packaging.python.org/en/latest/key_projects/" \l "venv)（用于 Python 3）允许您管理单独的软件包安装 不同的项目。它创建一个“虚拟”隔离的 Python 安装。什么时候 切换项目，可以创建一个隔离的新虚拟环境 从其他虚拟环境。您将从虚拟环境中获益 因为软件包可以放心地安装并且不会干扰 另一个项目的环境。

提示

建议在使用第三种设备时使用虚拟环境 派对套餐。

要创建虚拟环境，请转到项目的目录并运行 。这将在本地文件夹中创建一个新的虚拟环境：venv.venv

Unix/macOS 操作系统

python3 -m venv .venv

窗户

第二个参数是创建虚拟环境的位置。一般来说，你 可以在您的项目中创建它并调用它。.venv

venv将在文件夹中创建虚拟 Python 安装。.venv

注意

应从版本中排除虚拟环境目录 控制系统使用或类似。.gitignore

### **激活虚拟环境**

在开始在虚拟环境中安装或使用软件包之前，您需要 需要它。激活虚拟环境会将 特定于虚拟环境的可执行文件和可执行文件放入 壳牌的 .activatepythonpipPATH

Unix/macOS 操作系统

source .venv/bin/activate

窗户

要确认虚拟环境已激活，请检查 Python 解释器：

Unix/macOS 操作系统

which python

窗户

激活虚拟环境后，该位置将包括 目录：.venv

Unix/macOS 操作系统

.../.venv/bin/python

窗户

当虚拟环境被激活时，pip 会将软件包安装到其中 特定环境。这使您能够在 Python 应用程序。

### **停用虚拟环境**

如果要切换项目或离开虚拟环境，则环境：deactivate

deactivate

注意

关闭 shell 将停用虚拟环境。如果 您打开了一个新的 shell 窗口，并希望使用虚拟环境， 重新激活它。

### **重新激活虚拟环境**

如果要重新激活现有虚拟环境，请按照相同的操作 有关激活虚拟环境的说明。无需创建 一个新的虚拟环境。

## **准备点子**

[pip](https://packaging.python.org/en/latest/key_projects/" \l "pip) 是参考 Python 包管理器。 它用于在虚拟环境中安装和更新包。

Unix/macOS 操作系统

macOS 的 Python 安装程序包括 pip。在 Linux 上，您可能需要安装 一个附加包，例如 .您可以确保 pip 是 通过运行以下命令更新：python3-pip

python3 -m pip install --upgrade pip

python3 -m pip --version

之后，您应该在 用户站点：

pip 23.3.1 from .../.venv/lib/python3.9/site-packages (python 3.9)

窗户

## **使用 pip 安装软件包**

激活虚拟环境后，可以安装软件包。使用命令安装软件包。pip install

### **安装软件包**

例如，让我们从 [Python 包索引 （PyPI）](https://packaging.python.org/en/latest/glossary/" \l "term-Python-Package-Index-PyPI) 安装 [Requests](https://pypi.org/project/requests/) 库：

Unix/macOS 操作系统

python3 -m pip install requests

窗户

pip 应该下载请求及其所有依赖项并安装它们：

Collecting requests

Using cached requests-2.18.4-py2.py3-none-any.whl

Collecting chardet<3.1.0,>=3.0.2 (from requests)

Using cached chardet-3.0.4-py2.py3-none-any.whl

Collecting urllib3<1.23,>=1.21.1 (from requests)

Using cached urllib3-1.22-py2.py3-none-any.whl

Collecting certifi>=2017.4.17 (from requests)

Using cached certifi-2017.7.27.1-py2.py3-none-any.whl

Collecting idna<2.7,>=2.5 (from requests)

Using cached idna-2.6-py2.py3-none-any.whl

Installing collected packages: chardet, urllib3, certifi, idna, requests

Successfully installed certifi-2017.7.27.1 chardet-3.0.4 idna-2.6 requests-2.18.4 urllib3-1.22

### **安装特定软件包版本**

pip 允许您使用[版本说明符](https://packaging.python.org/en/latest/glossary/" \l "term-Version-Specifier)指定要安装的包版本。例如，要安装 的特定版本：requests

Unix/macOS 操作系统

python3 -m pip install 'requests==2.18.4'

窗户

要安装最新版本的请求，请执行以下操作：2.x

Unix/macOS 操作系统

python3 -m pip install 'requests>=2.0.0,<3.0.0'

窗户

若要安装包的预发布版本，请使用以下标志：--pre

Unix/macOS 操作系统

python3 -m pip install --pre requests

窗户

### **安装附加功能**

Some packages have optional [extras](https://setuptools.readthedocs.io/en/latest/userguide/dependency_management.html" \l "optional-dependencies). You can tell pip to install these by specifying the extra in brackets:

Unix/macOS

python3 -m pip install 'requests[security]'

Windows

### **Install a package from source**

pip can install a package directly from its source code. For example, to install the source code in the directory:google-auth

Unix/macOS

cd google-auth

python3 -m pip install .

Windows

Additionally, pip can install packages from source in [development mode](https://setuptools.pypa.io/en/latest/userguide/development_mode.html" \o "(in setuptools v68.2.2.post20231119)), meaning that changes to the source directory will immediately affect the installed package without needing to re-install:

Unix/macOS

python3 -m pip install --editable .

Windows

### **Install from version control systems**

pip can install packages directly from their version control system. For example, you can install directly from a git repository:

google-auth @ git+https://github.com/GoogleCloudPlatform/google-auth-library-python.git

For more information on supported version control systems and syntax, see pip’s documentation on [VCS Support](https://pip.pypa.io/en/latest/topics/vcs-support/" \l "vcs-support" \o "(in pip v24.0)).

### **Install from local archives**

If you have a local copy of a [Distribution Package](https://packaging.python.org/en/latest/glossary/" \l "term-Distribution-Package)’s archive (a zip, wheel, or tar file) you can install it directly with pip:

Unix/macOS

python3 -m pip install requests-2.18.4.tar.gz

Windows

If you have a directory containing archives of multiple packages, you can tell pip to look for packages there and not to use the [Python Package Index (PyPI)](https://packaging.python.org/en/latest/glossary/" \l "term-Python-Package-Index-PyPI) at all:

Unix/macOS

python3 -m pip install --no-index --find-links**=**/local/dir/ requests

Windows

This is useful if you are installing packages on a system with limited connectivity or if you want to strictly control the origin of distribution packages.

### **Install from other package indexes**

If you want to download packages from a different index than the [Python Package Index (PyPI)](https://packaging.python.org/en/latest/glossary/" \l "term-Python-Package-Index-PyPI), you can use the flag:--index-url

Unix/macOS

python3 -m pip install --index-url http://index.example.com/simple/ SomeProject

Windows

If you want to allow packages from both the [Python Package Index (PyPI)](https://packaging.python.org/en/latest/glossary/" \l "term-Python-Package-Index-PyPI) and a separate index, you can use the flag instead:--extra-index-url

Unix/macOS

python3 -m pip install --extra-index-url http://index.example.com/simple/ SomeProject

Windows

## **Upgrading packages**

pip can upgrade packages in-place using the flag. For example, to install the latest version of and all of its dependencies:--upgraderequests

Unix/macOS

python3 -m pip install --upgrade requests

Windows

## **Using a requirements file**

Instead of installing packages individually, pip allows you to declare all dependencies in a [Requirements File](https://pip.pypa.io/en/latest/user_guide/" \l "requirements-files" \o "(in pip v24.0)). For example you could create a file containing:requirements.txt

requests==2.18.4

google-auth==1.1.0

并告诉 pip 使用以下标志安装此文件中的所有包：-r

Unix/macOS 操作系统

python3 -m pip install -r requirements.txt

窗户

## **冻结依赖项**[#](https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/" \l "freezing-dependencies" \o "Link to this heading)

Pip 可以使用以下命令导出所有已安装软件包及其版本的列表：freeze

Unix/macOS 操作系统

python3 -m pip freeze

窗户

这将输出包说明符列表，例如：

cachetools==2.0.1

certifi==2017.7.27.1

chardet==3.0.4

google-auth==1.1.1

idna==2.6

pyasn1==0.3.6

pyasn1-modules==0.1.4

requests==2.18.4

rsa==3.4.2

six==1.11.0

urllib3==1.22

该命令可用于创建需求文件，这些[文件](https://pip.pypa.io/en/latest/user_guide/" \l "requirements-files" \o "(in pip v24.0))可以重新创建环境中安装的所有软件包的确切版本。pip freeze