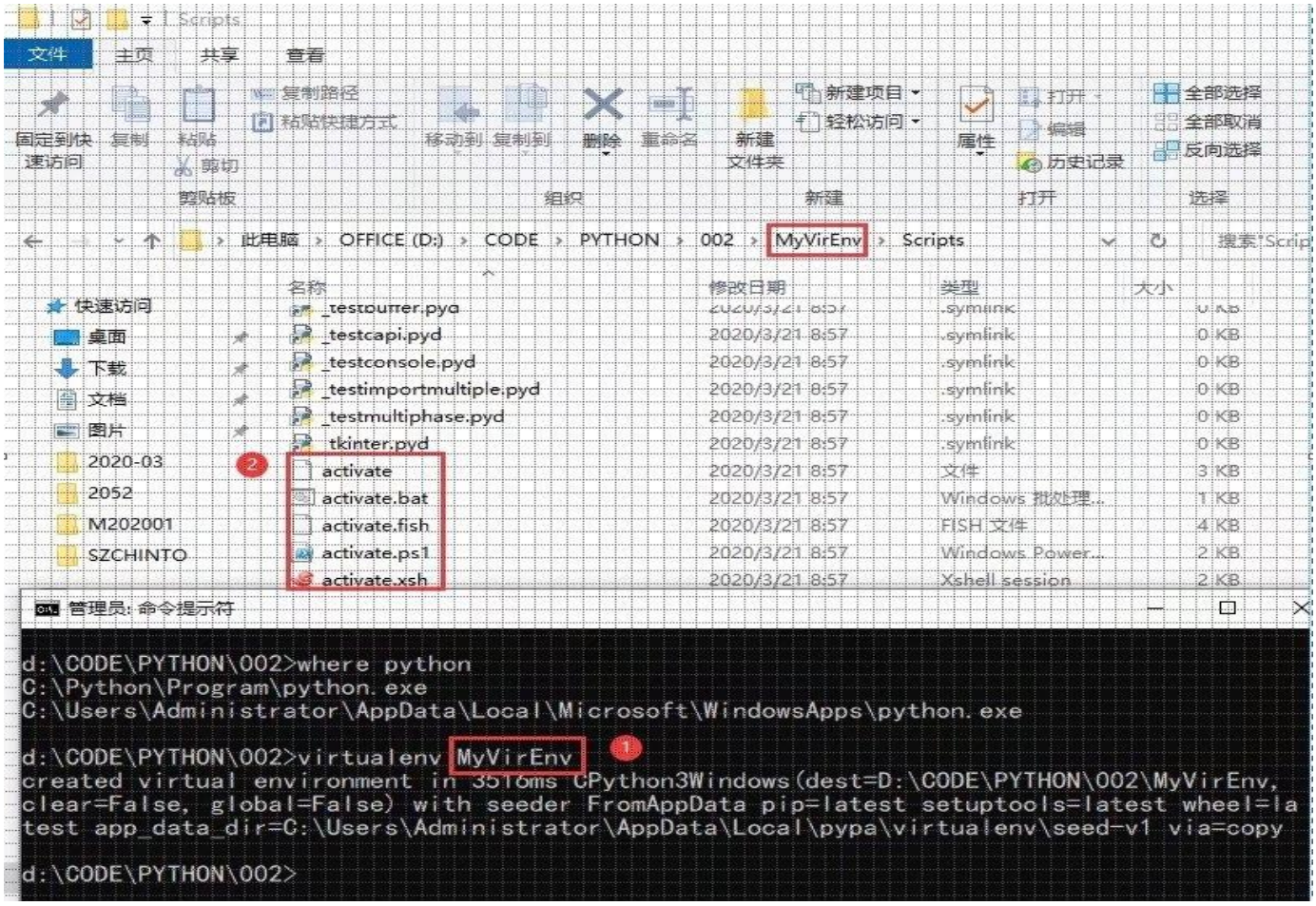
在 Windows 下 Virtualenv 和 virtualenvwrapper-win 这两个包都是管理虚拟环境的工具包，Linux 下这两个工具包的名称为 Virtualenv 和 virtualenvwrapper.【注在 Windows 下 Virtualenv 并不能很好地隔离虚拟环境】

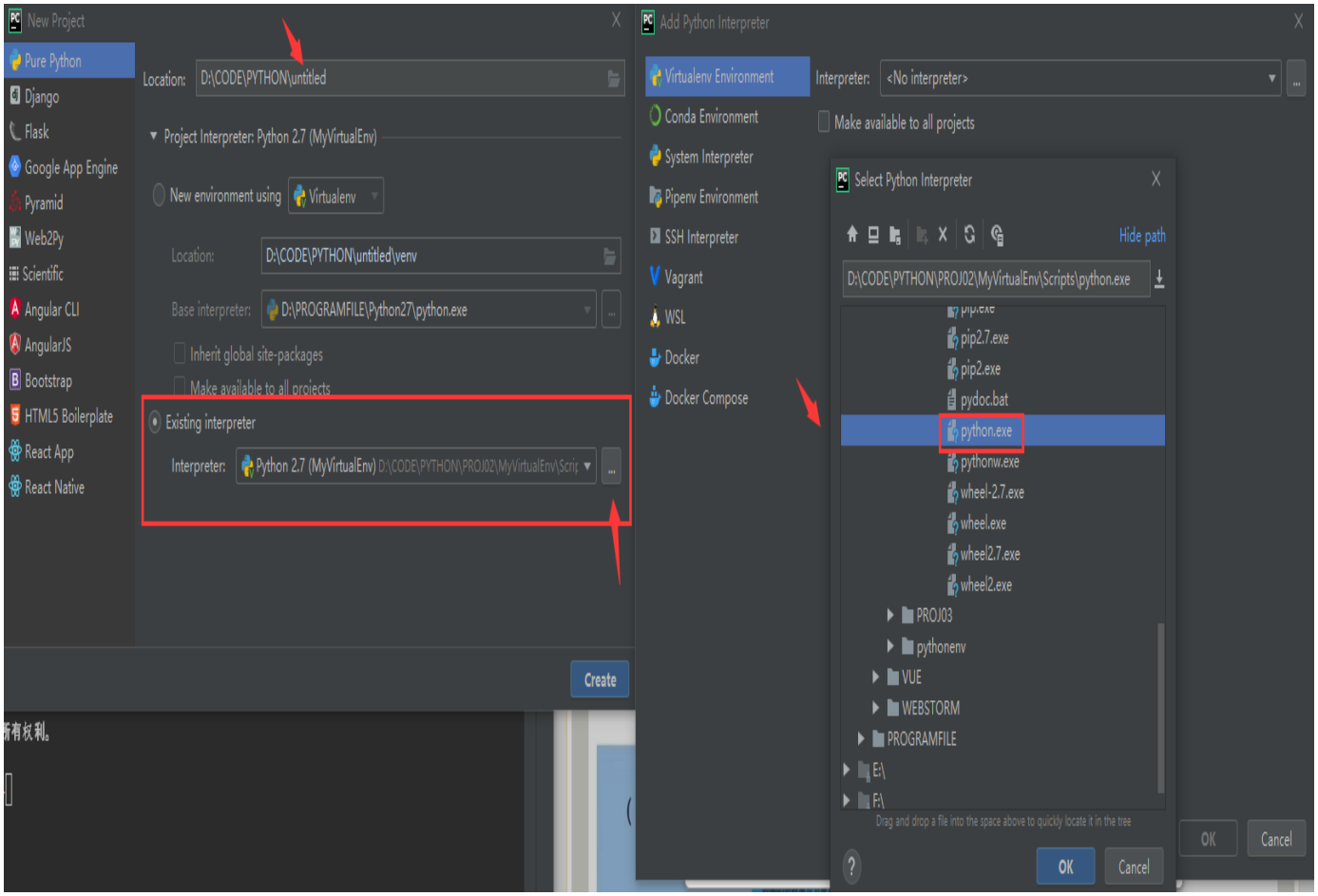
安装 virtualenv 和安装其他包的方法一样，pip install virtualenv
在 python 2.7 下安装 就用 pip2 install virtualenv
在 python 3.6 下安装就用 pip3 install virtualenv

虚拟环境在 Windows 下安装配置
1、 安装虚拟环境
pip install virtualenv
pip install virtualenvwrapper-win
2、 配置虚拟环境路径存放的目录
WORKON_HOME 变量[适用于 virtaulenvwrapper pipenv poetry]
Virtualenv D:\ENVS [直接指定存放目录]
3、 virtualenvwrapper-win 创建虚拟环境
mkvirtualenv 虚拟环境名字
4、 切换虚拟环境
workon 环境名字
workon 不带参数查询环境名字
5、 退出虚拟环境
deactivate
6、 删除虚拟环境
rmvirtualenv 环境名字

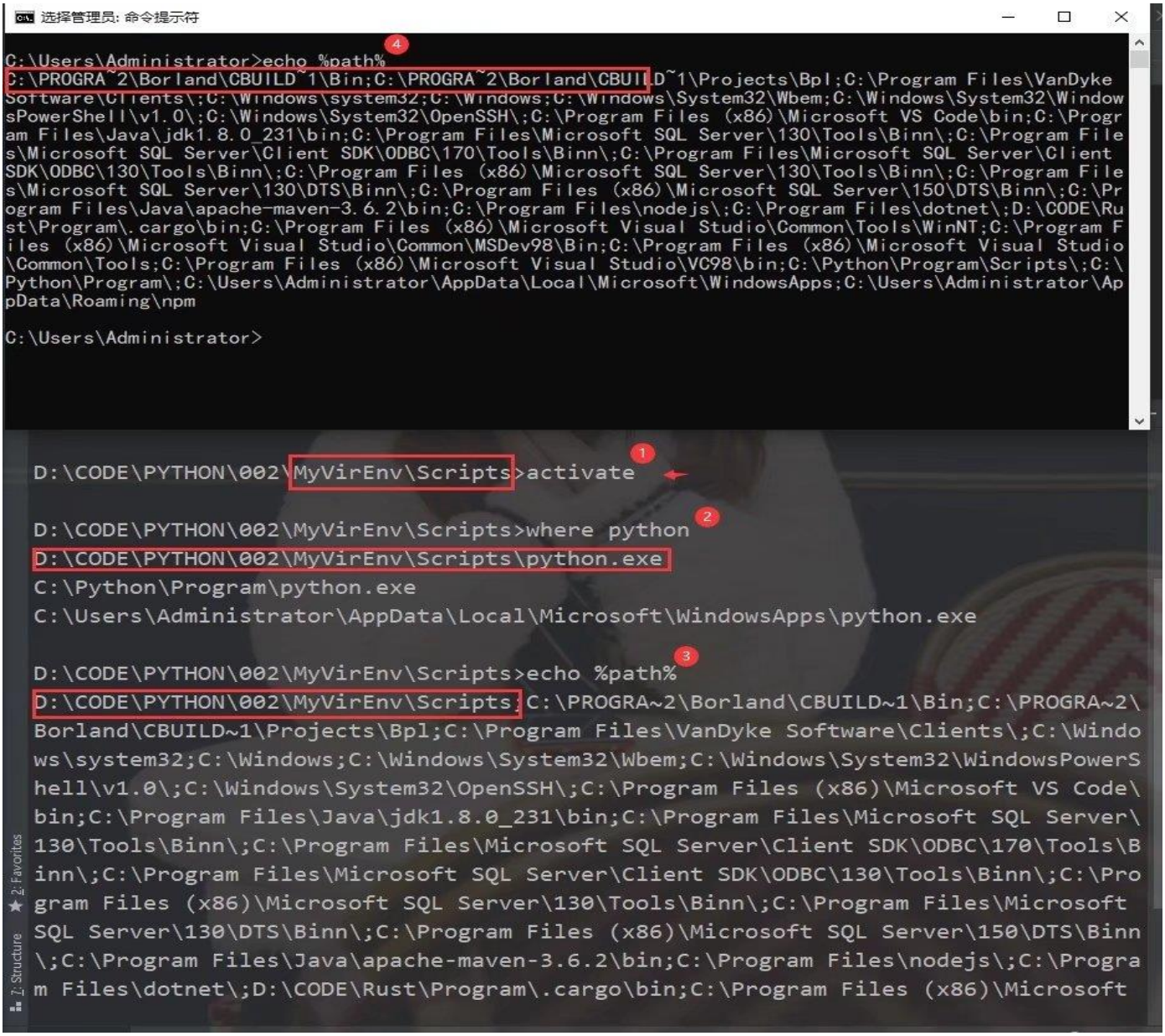
1.virtualenv



pycharm 中选择已存在的虚拟环境

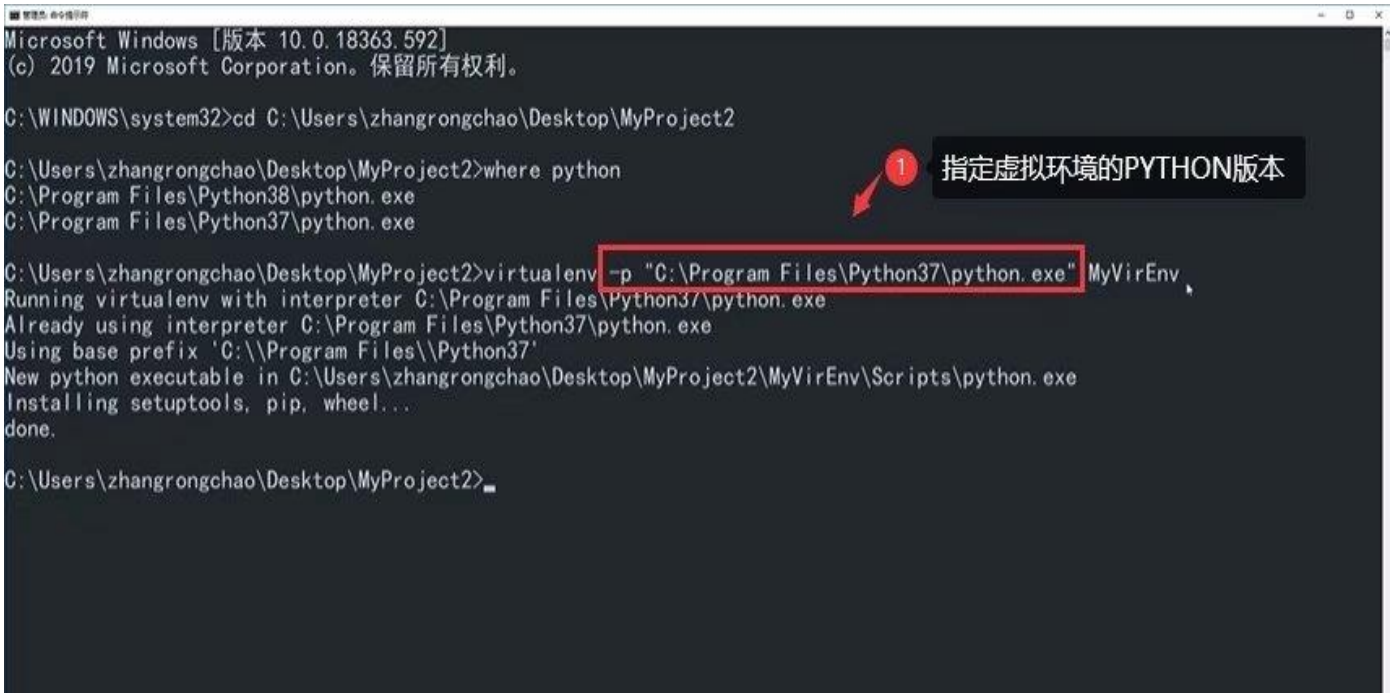

New Project
Pure Python
Django
Flask
Google App Engine
Pyramid
Web2Py
Scientific
Angular CLI
AngularJS
Bootstrap
HTML5 Boilerplate
React App
React Native
Location: D:\CODE\PYTHON\untitled
Project Interpreter: Python 2.7 (MyVirtualEnv)
New environment using Virtualenv
Location: D:\CODE\PYTHON\untitled\venv
Base interpreter: D:\PROGRAMFILE\Python27\python.exe
Inherit global site-packages
Make available to all projects
Existing interpreter
Interpreter: Python 2.7 (MyVirtualEnv) D:\CODE\PYTHON\PROJ02\MyVirtualEnv\Scrip
Create
Add Python Interpreter
Virtualenv Environment
Conda Environment
System Interpreter
Pipenv Environment
SSH Interpreter
Vagrant
WSL
Docker
Docker Compose
Interpreter: <No interpreter>
Make available to all projects
Select Python Interpreter
Hide path
D:\CODE\PYTHON\PROJ02\MyVirtualEnv\Scripts\python.exe
pip2.7.exe
pip2.exe
pydoc.bat
python.exe
pythonw.exe
wheel-2.7.exe
wheel.exe
wheel2.7.exe
wheel2.exe
PROJ03
pythonenv
VUE
WEBSTORM
PROGRAMFILE
E:\
F:\
Drag and drop a file into the space above to quickly locate it in the tree
OK
Cancel
OK
Cancel


Python运行环境
（对应于Python的安装目录）
Python解释器
Python标准库
工具（pip、easy-install，等等）
site-packages目录中
安装的所有第三方库及其版本
pygame（1.9.6）
requests（2.22.0）
pymysql（0.8.1）
numpy（1.18.1）
pandas（0.24.2）
.....
虚拟的Python运行环境
（根据Python的安装目录虚拟出来的）
Python解释器
Python标准库
工具（pip、easy-install，等等）
site-packages目录中
安装的所有第三方库及其版本
pymysql（0.9.3）
项目1
pymysql（0.9.3）
虚拟的Python运行环境
（根据Python的安装目录虚拟出来的）
Python解释器
Python标准库
工具（pip、easy-install，等等）
site-packages目录中
安装的所有第三方库及其版本
pymysql（0.7.1）
项目2
pymysql（0.7.1）
虚拟的Python运行环境
（根据Python的安装目录虚拟出来的）
Python解释器
Python标准库
工具（pip、easy-install，等等）
site-packages目录中
安装的所有第三方库及其版本
pymysql（0.6.2）
项目3
pymysql（0.6.2）

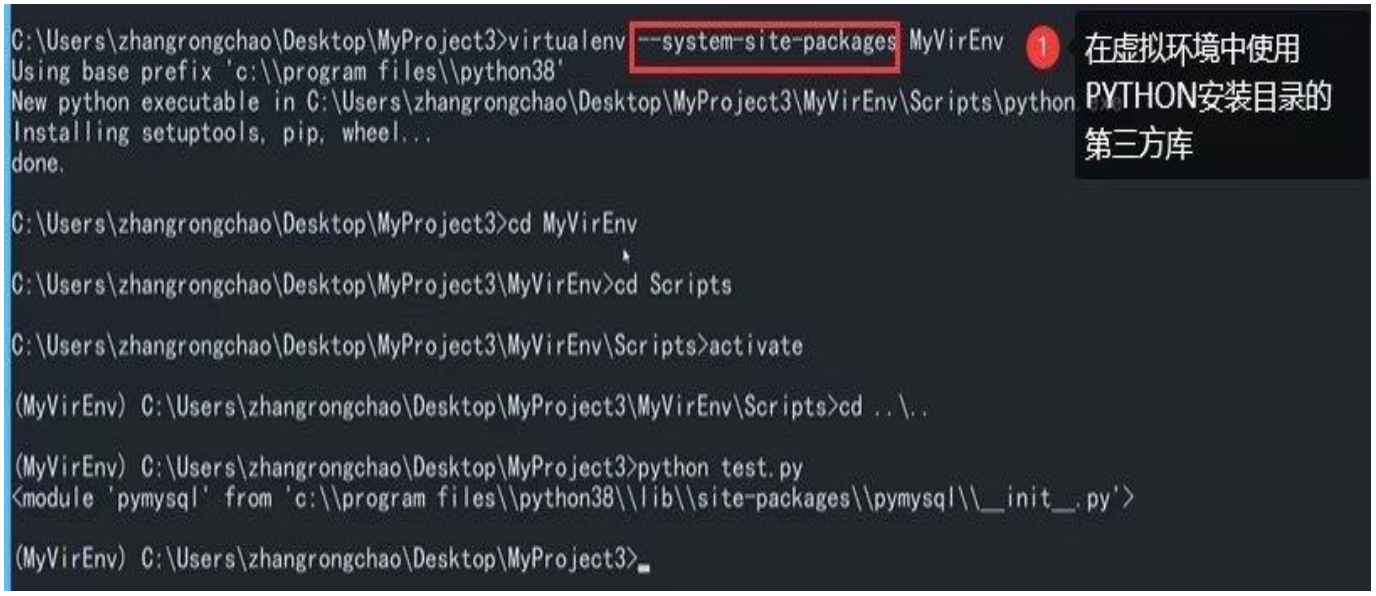


上图进入虚拟环境后，echo %path%会显示当前的环境，且置于查找路径的最前面。

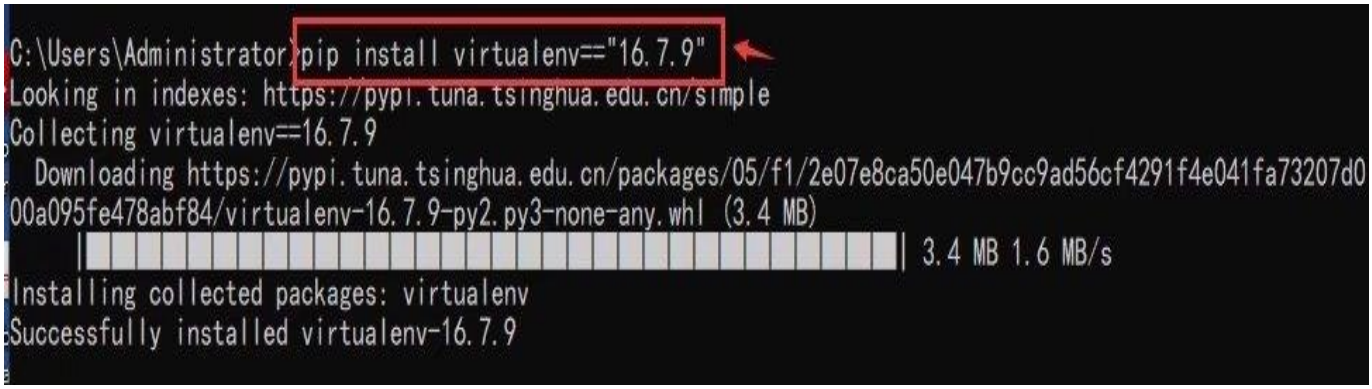
指定 python 版本创建虚拟环境

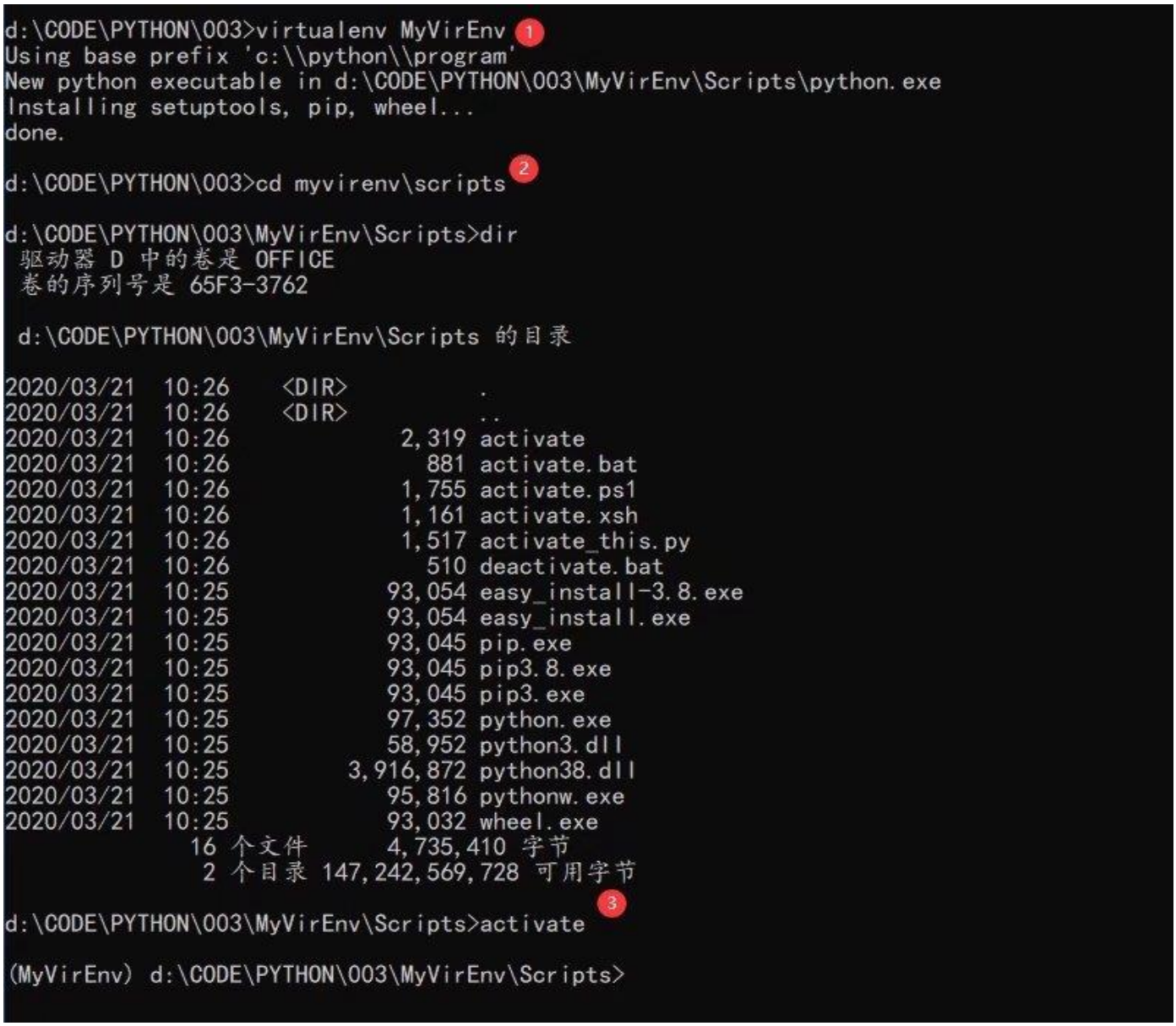


--system-site-packages 参数为当前 Python 虚拟环境的加载当前的第三方库。



安装特定的版本使用==





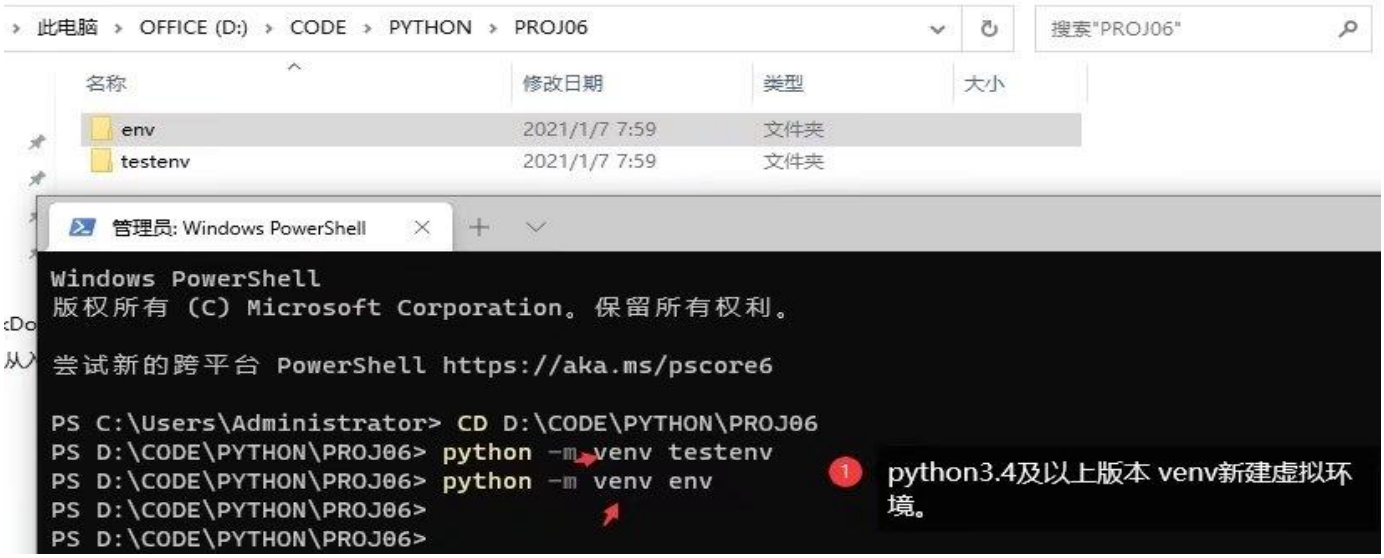
D:\CODE\PYTHON\PROJ02\MyVirtualEnv\Scripts>activate

(MyVirtualEnv) D:\CODE\PYTHON\PROJ02\MyVirtualEnv\Scripts>

```
管理员: 命令提示符
C:\Users\zhangrongchao\Desktop\MyProject>virtualenv MyVirEnv
Using base prefix 'c:\\program files\\python38'
New python executable in C:\Users\zhangrongchao\Desktop\MyProject\MyVirEnv\Scripts\python.exe
Installing setuptools, pip, wheel...
done.

C:\Users\zhangrongchao\Desktop\MyProject>cd MyVirEnv\Scripts

C:\Users\zhangrongchao\Desktop\MyProject\MyVirEnv\Scripts>activate

(MyVirEnv) C:\Users\zhangrongchao\Desktop\MyProject\MyVirEnv\Scripts>cd ..\..

(MyVirEnv) C:\Users\zhangrongchao\Desktop\MyProject>pip install -r requirements.txt
Collecting pymongo==3.10.1
  Using cached pymongo-3.10.1-cp38-cp38-win_amd64.whl (355 kB)
Collecting PyMySQL==0.9.3
  Using cached PyMySQL-0.9.3-py2.py3-none-any.whl (47 kB)
Installing collected packages: pymongo, PyMySQL
Successfully installed PyMySQL-0.9.3 pymongo-3.10.1

(MyVirEnv) C:\Users\zhangrongchao\Desktop\MyProject>
```

1 pip freeze > requirements.txt

python 3.4 或以上的版也可以使用 venv 命令来创建 python 虚拟环境，其功能与 virtualenv 等效。

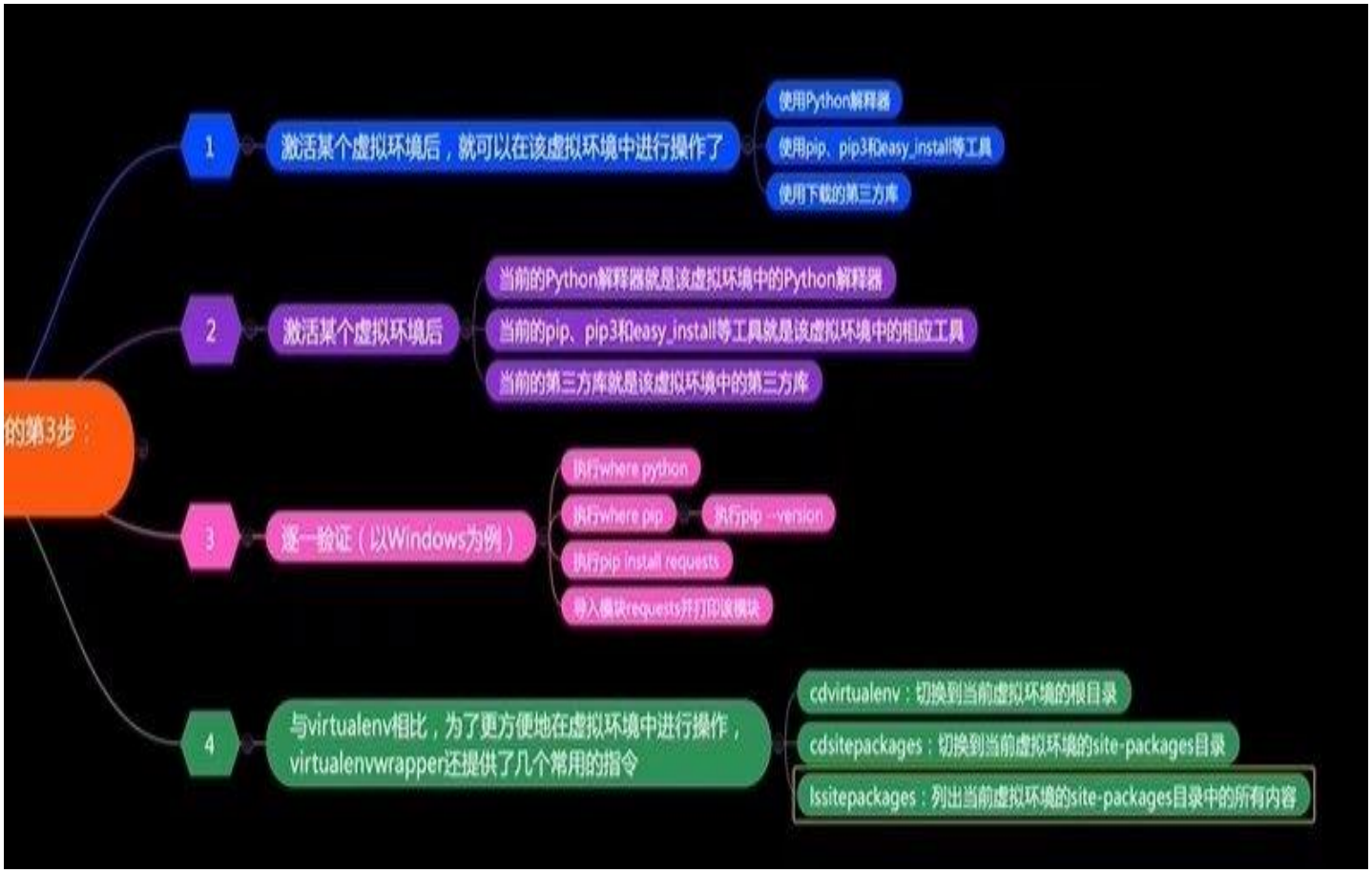



```
Microsoft Windows [版本 10.0.22000.376]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>lsvirtualenv
dir /b /ad "D:\PROGRAMFILES\python-env"
==========================================================================

C:\Users\Administrator>
```

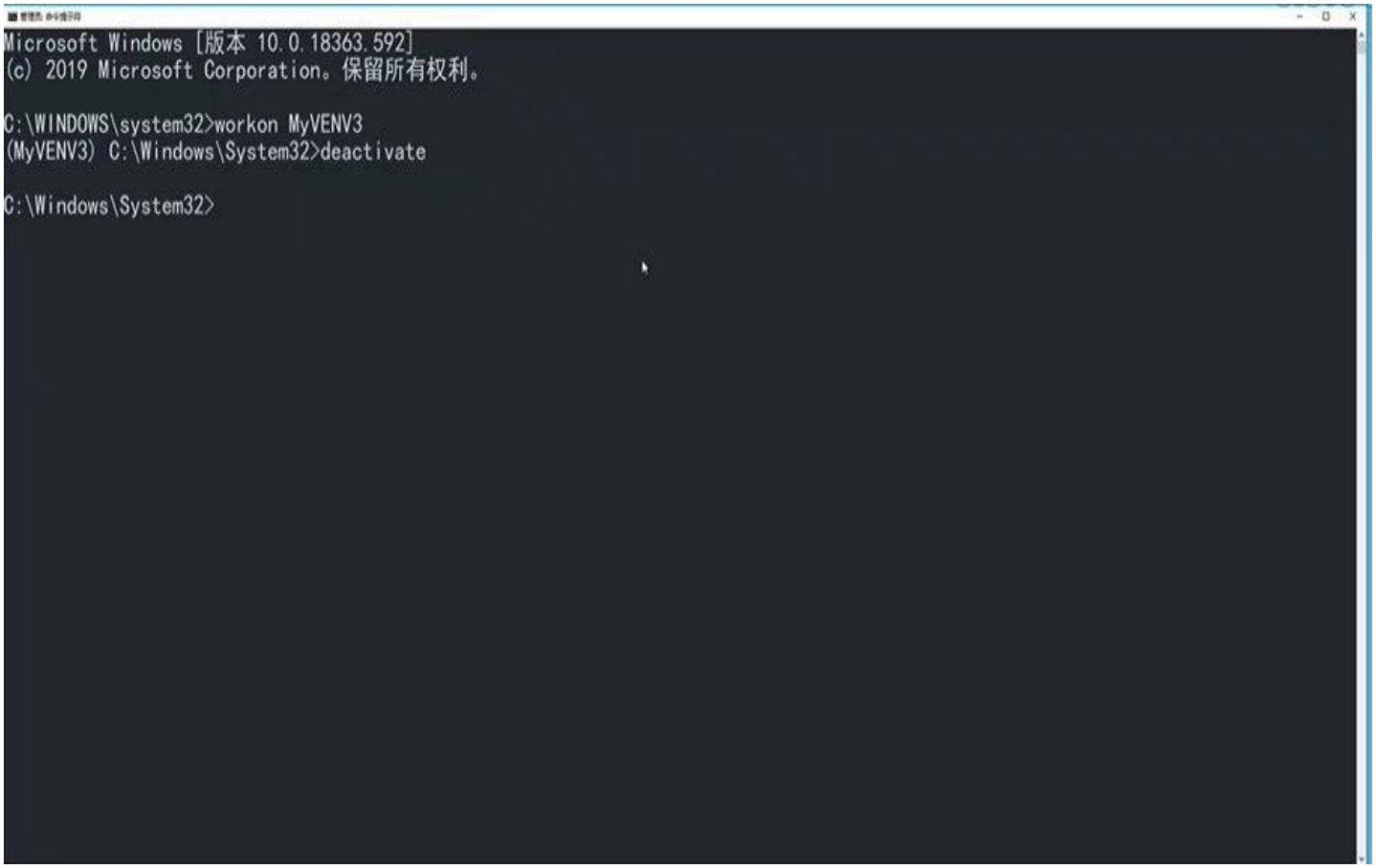
1 查询创建的虚拟环境【并不准确】

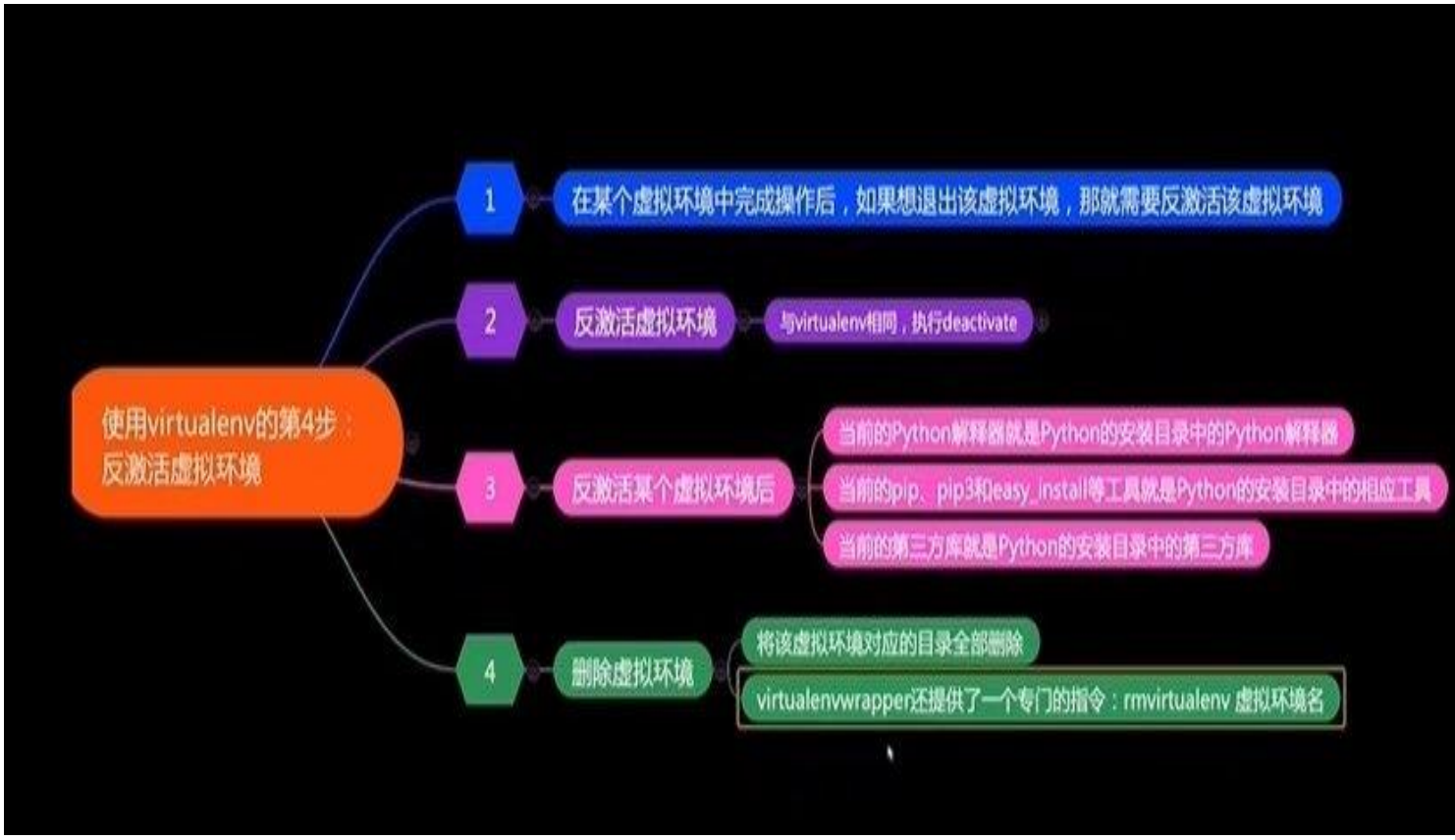
2.virtualenvwrapper-win(virtualenvwrapper)

```
管理员: 命令提示符
Microsoft Windows [版本 10.0.18363.720]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>lsvirtualenv

dir /b /ad "C:\Users\Administrator\Envs"
==========================================================
找不到文件

C:\Users\Administrator>workon

Pass a name to activate one of the following virtualenvs:
==========================================================
找不到文件

C:\Users\Administrator>
```

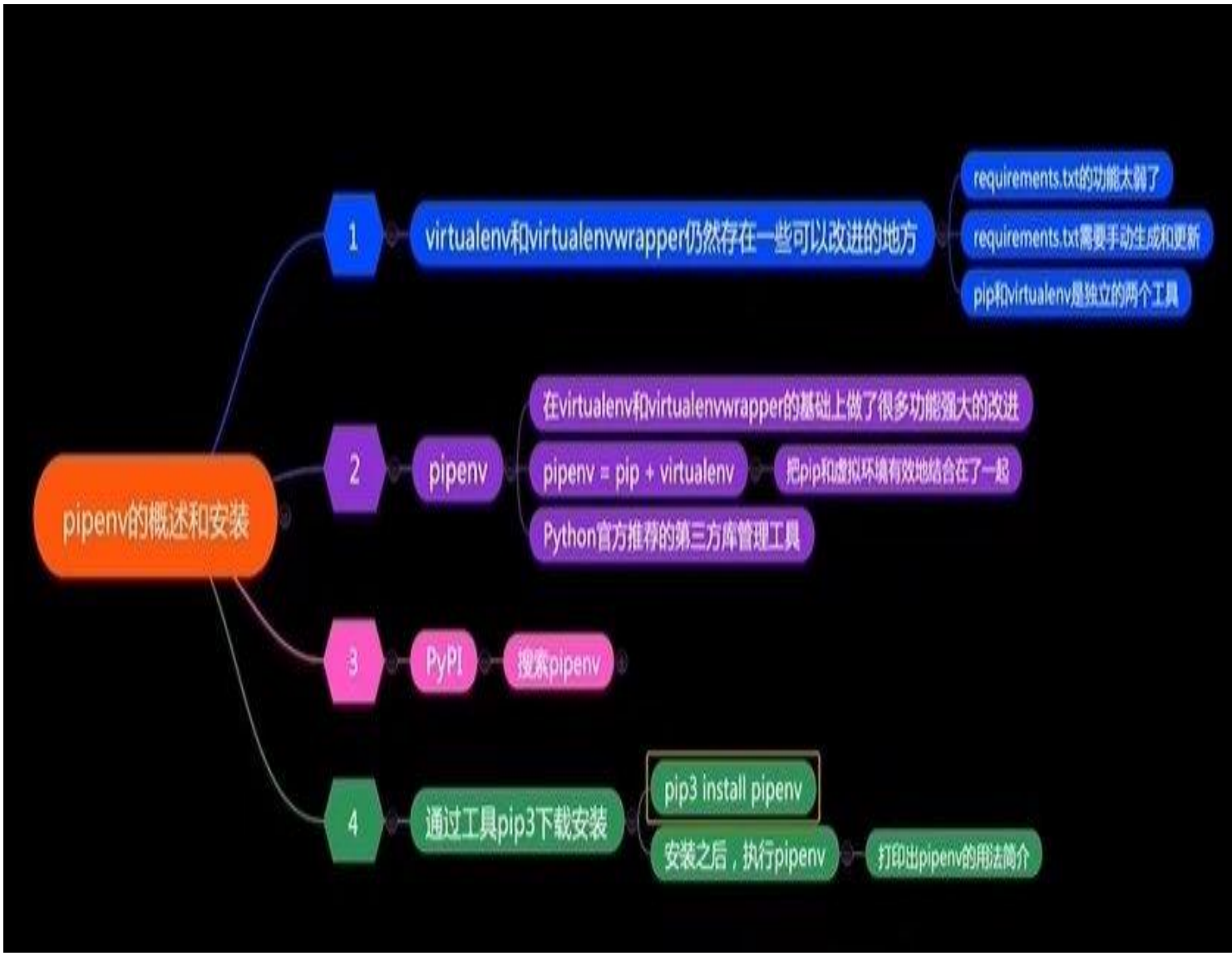


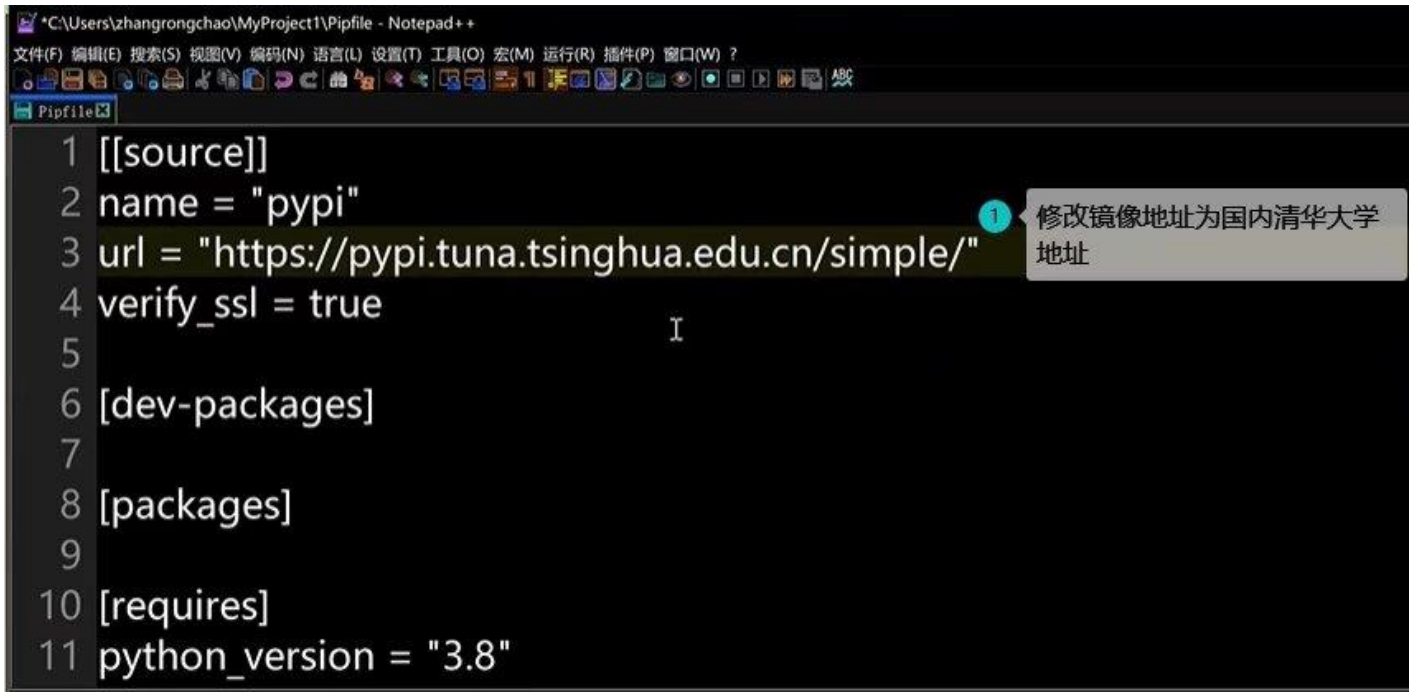
workon 虚拟环境名--激活虚拟环境，deactivate 退出虚拟环境



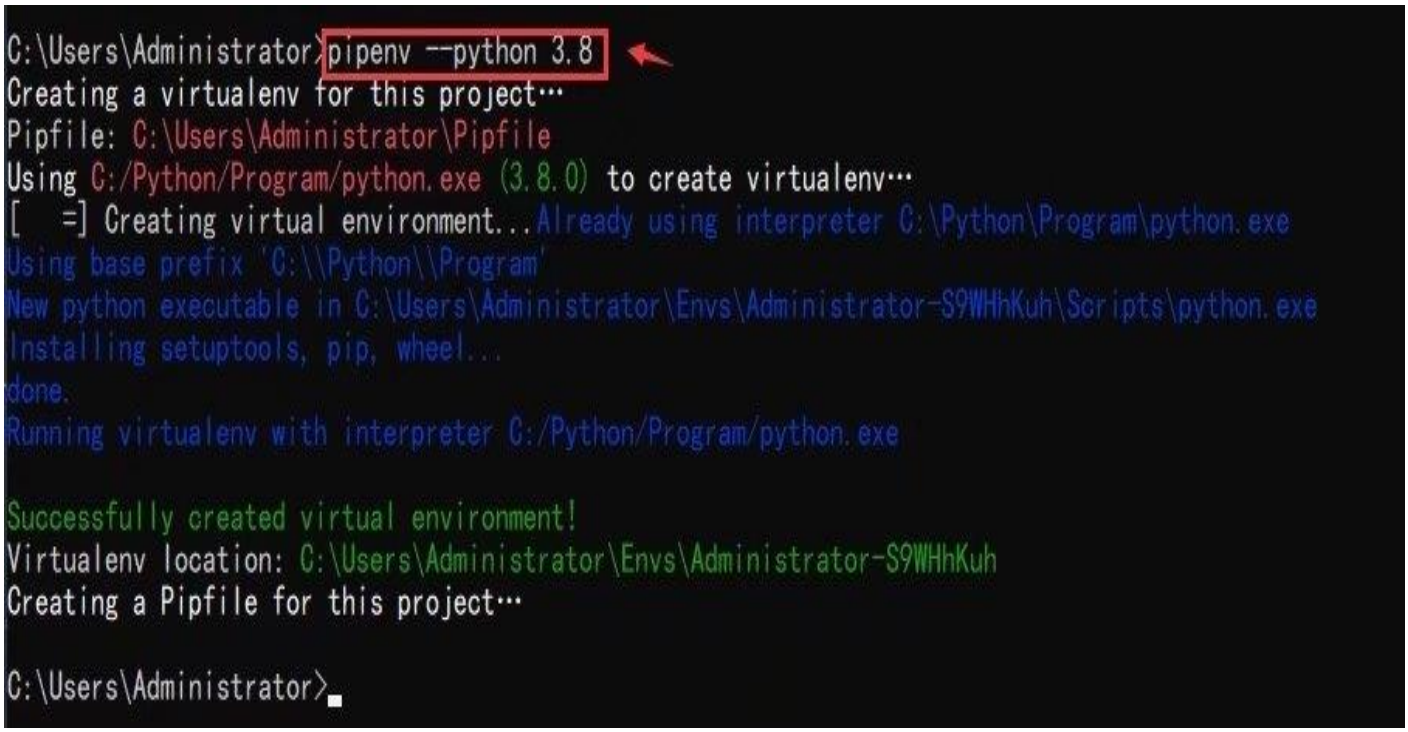

使用virtualenv的第4步：
反激活虚拟环境
1
在某个虚拟环境中完成操作后，如果想退出该虚拟环境，那就需要反激活该虚拟环境
2
反激活虚拟环境
与virtualenv相同，执行deactivate
3
反激活某个虚拟环境后
当前的Python解释器就是Python的安装目录中的Python解释器
当前的pip、pip3和easy_install等工具就是Python的安装目录中的相应工具
当前的第三方库就是Python的安装目录中的第三方库
4
删除虚拟环境
将该虚拟环境对应的目录全部删除
virtualenvwrapper还提供了一个专门的指令：rmvirtualenv 虚拟环境名

3.pipenv

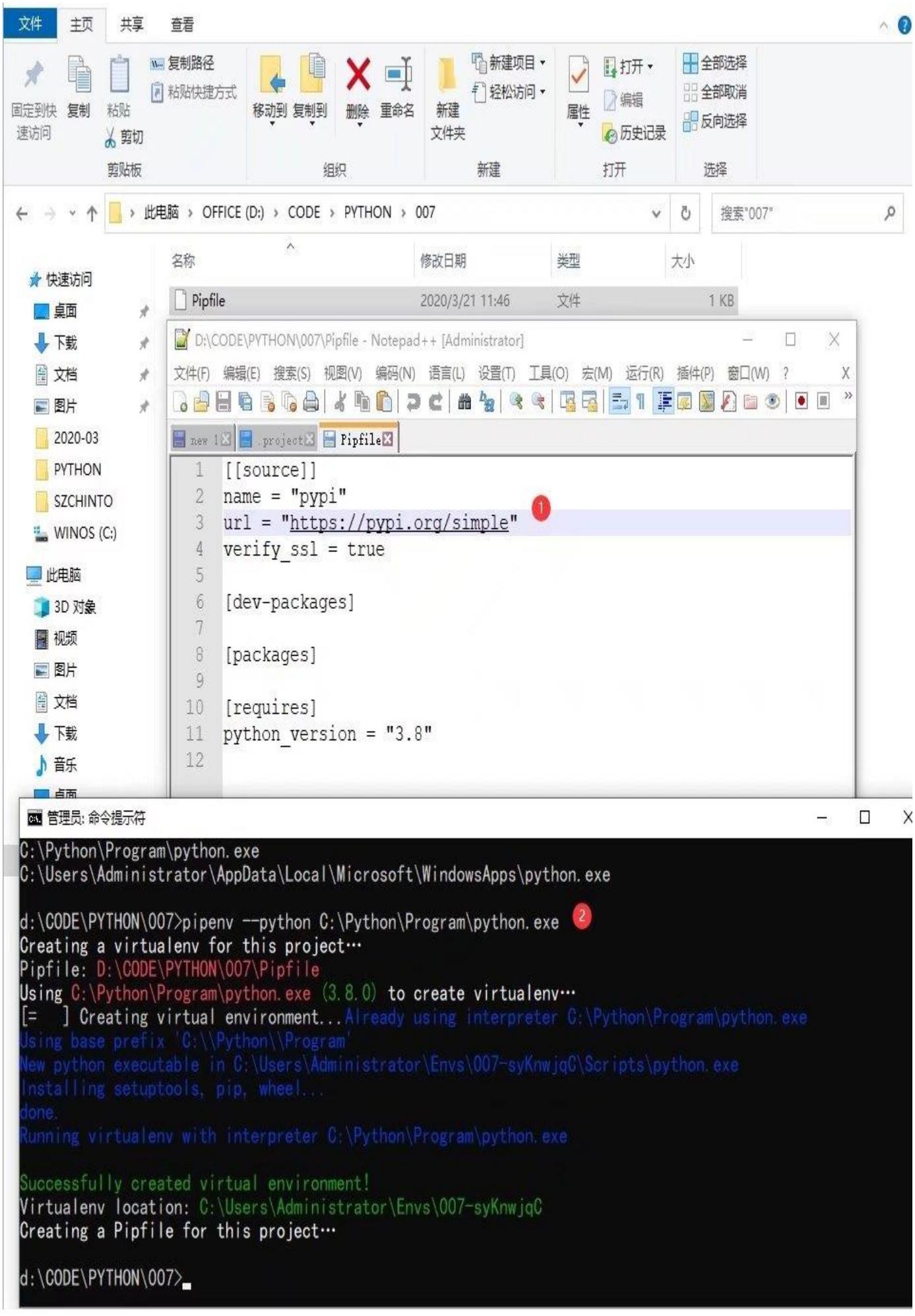

pipenv的概述和安装
1
virtualenv和virtualenvwrapper仍然存在一些可以改进的地方
requirements.txt的功能太弱了
requirements.txt需要手动生成和更新
pip和virtualenv是独立的两个工具
2
pipenv
在virtualenv和virtualenvwrapper的基础上做了很多功能强大的改进
pipenv = pip + virtualenv
把pip和虚拟环境有效地结合在了一起
Python官方推荐的第三方库管理工具
3
PyPI
搜索pipenv
4
通过工具pip3下载安装
pip3 install pipenv
安装之后，执行pipenv
打印出pipenv的用法简介

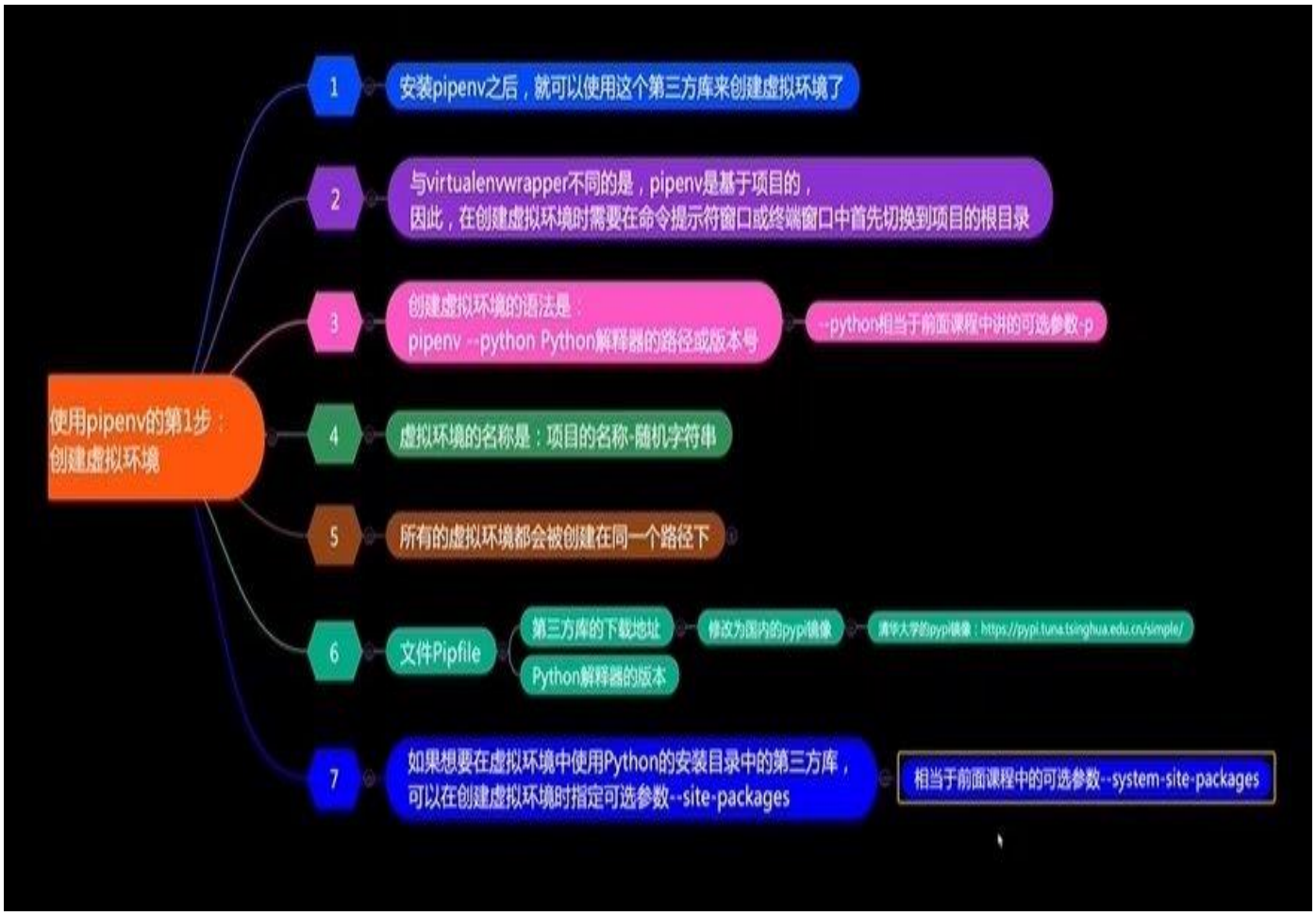

*C:\Users\zhangrongchao\MyProject1\Pipfile - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
Pipfile
1 [[source]]
2 name = "pypi"
3 url = "https://pypi.tuna.tsinghua.edu.cn/simple/"
4 verify_ssl = true
5
6 [dev-packages]
7
8 [packages]
9
10 [requires]
11 python_version = "3.8"
1 修改镜像地址为国内清华大学地址

pipenv --python 版本号是从本地版本中创建

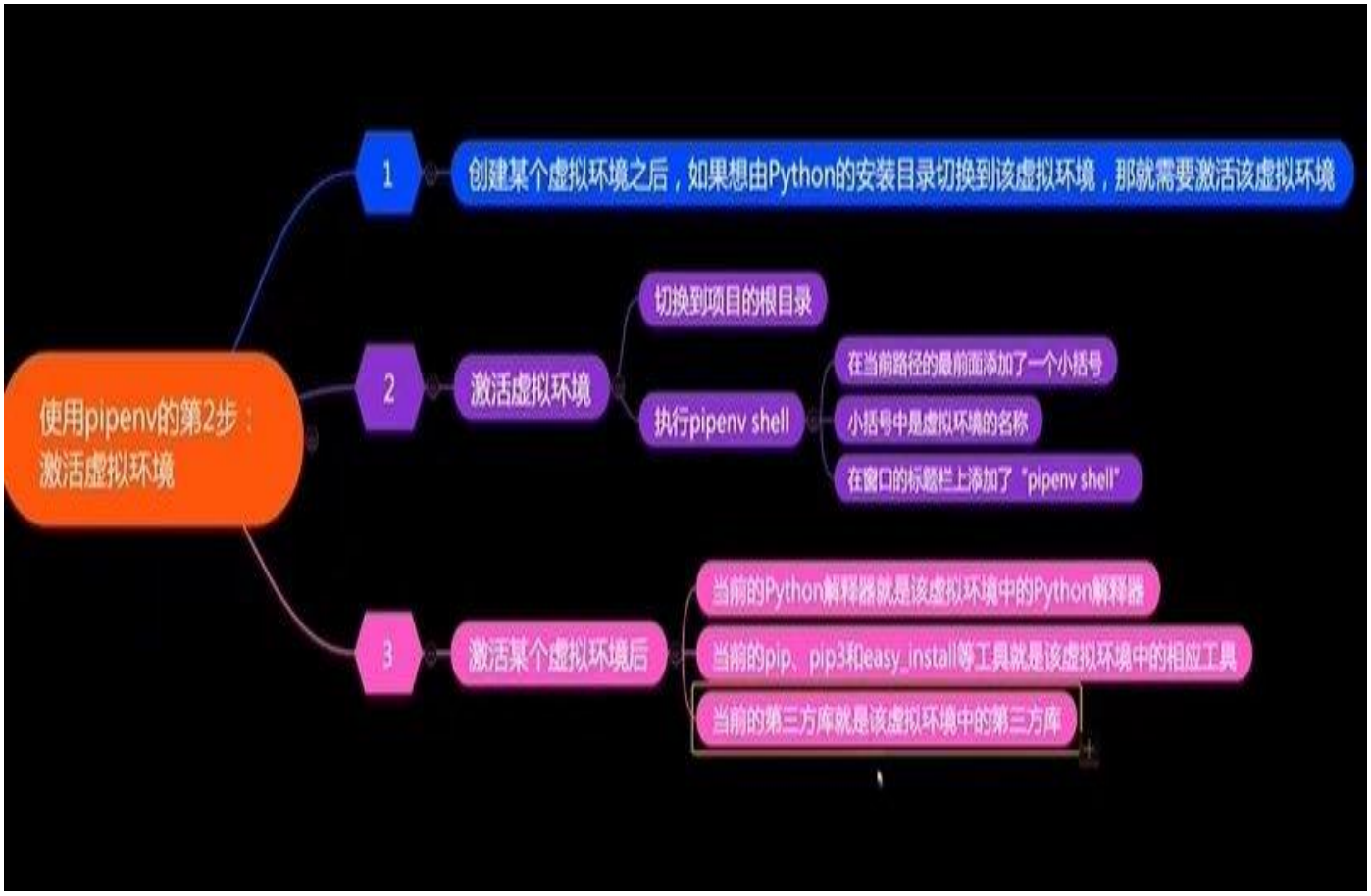


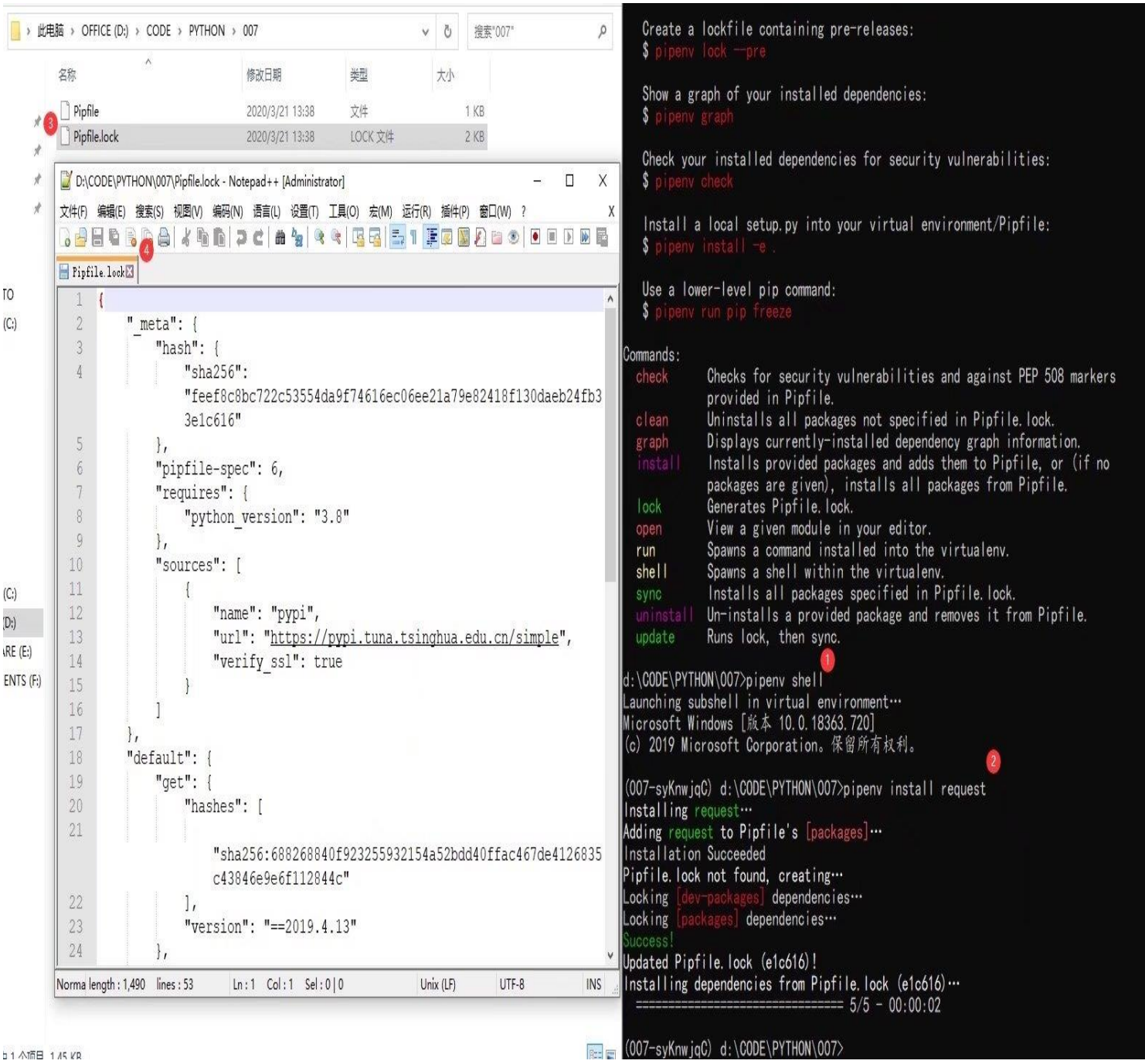
url="https://pypi.tuna.tsinghua.edu.cn/simple"

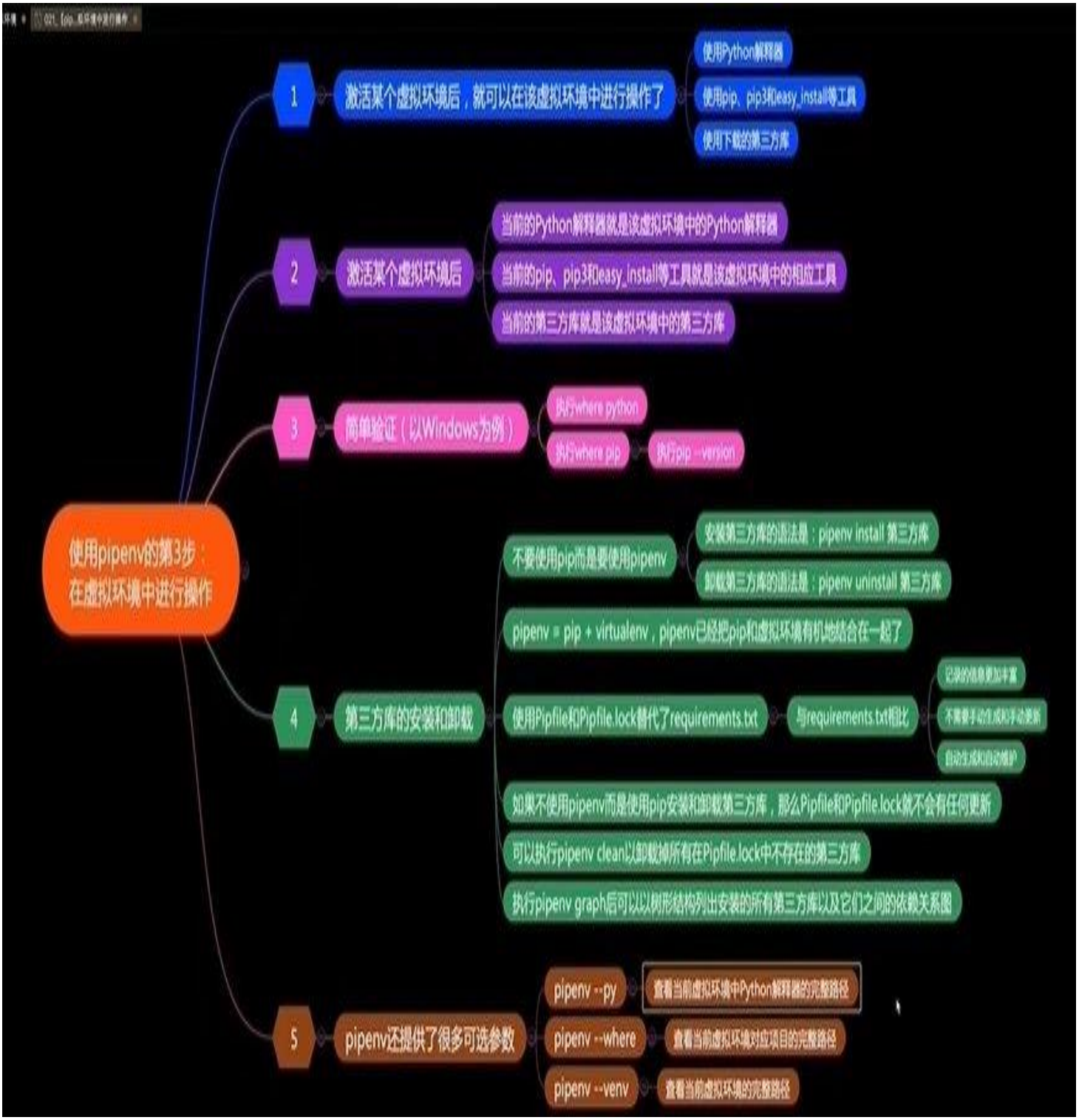

文件
主页
共享
查看
固定到快速访问
复制
粘贴
复制路径
粘贴快捷方式
剪切
剪贴板
移动到
复制到
删除
重命名
组织
新建项目
轻松访问
新建文件夹
新建
属性
打开
编辑
历史记录
打开
全部选择
全部取消
反向选择
选择
此电脑 › OFFICE (D:) › CODE › PYTHON › 007
搜索"007"
名称
修改日期
类型
大小
Pipfile
2020/3/21 11:46
文件
1 KB
快速访问
桌面
下载
文档
图片
2020-03
PYTHON
SZCHINTO
WINOS (C:)
此电脑
3D 对象
视频
图片
文档
下载
音乐
D:\CODE\PYTHON\007\Pipfile - Notepad++ [Administrator]
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
new 1
.project
Pipfile
1 [[source]]
2 name = "pypi"
3 url = "https://pypi.org/simple"
4 verify_ssl = true
5
6 [dev-packages]
7
8 [packages]
9
10 [requires]
11 python_version = "3.8"
12
1
管理员: 命令提示符
C:\Python\Program\python.exe
C:\Users\Administrator\AppData\Local\Microsoft\WindowsApps\python.exe
d:\CODE\PYTHON\007>pipenv --python C:\Python\Program\python.exe
2
Creating a virtualenv for this project…
Pipfile: D:\CODE\PYTHON\007\Pipfile
Using C:\Python\Program\python.exe (3.8.0) to create virtualenv…
[=] Creating virtual environment...Already using interpreter C:\Python\Program\python.exe
Using base prefix 'C:\\Python\\Program'
New python executable in C:\Users\Administrator\Envs\007-syKnwjqC\Scripts\python.exe
Installing setuptools, pip, wheel...
done.
Running virtualenv with interpreter C:\Python\Program\python.exe
Successfully created virtual environment!
Virtualenv location: C:\Users\Administrator\Envs\007-syKnwjqC
Creating a Pipfile for this project…
d:\CODE\PYTHON\007>

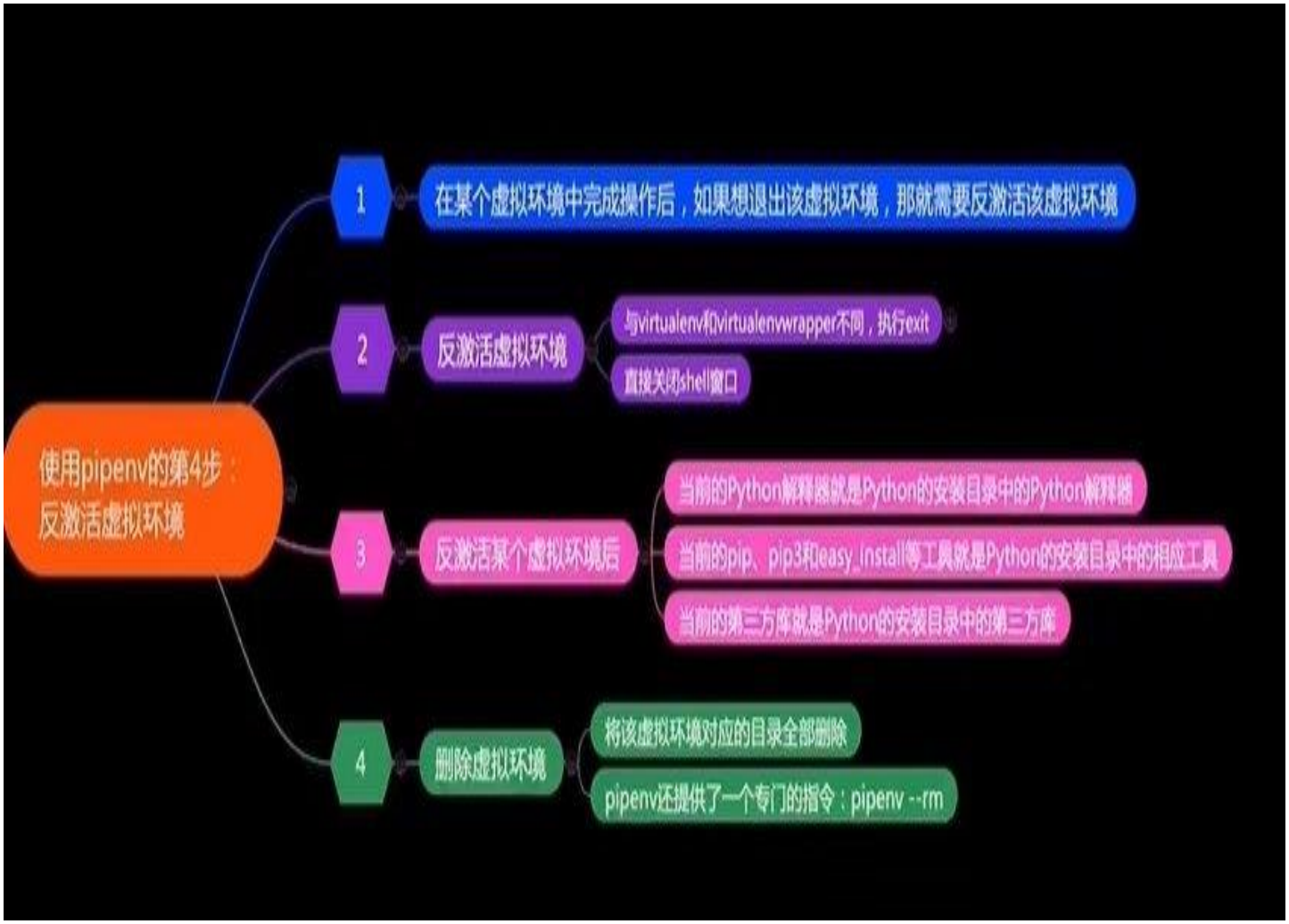


pipenv shell 激活虚拟环境，exit 离开虚拟环境


使用pipenv的第2步：
激活虚拟环境
1
创建某个虚拟环境之后，如果想由Python的安装目录切换到该虚拟环境，那就需要激活该虚拟环境
2
激活虚拟环境
切换到项目的根目录
执行pipenv shell
在当前路径的最前面添加了一个小括号
小括号中是虚拟环境的名称
在窗口的标题栏上添加了“pipenv shell”
3
激活某个虚拟环境后
当前的Python解释器就是该虚拟环境中的Python解释器
当前的pip、pip3和easy_install等工具就是该虚拟环境中的相应工具
当前的第三方库就是该虚拟环境中的第三方库


此电脑 › OFFICE (D:) › CODE › PYTHON › 007
搜索"007"
名称 修改日期 类型 大小
Pipfile 2020/3/21 13:38 文件 1 KB
Pipfile.lock 2020/3/21 13:38 LOCK 文件 2 KB
D:\CODE\PYTHON\007\Pipfile.lock - Notepad++ [Administrator]
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
Pipfile.lock
{
"_meta": {
"hash": {
"sha256":
"feef8c8bc722c53554da9f74616ec06ee21a79e82418f130daeb24fb33e1c616"
},
"pipfile-spec": 6,
"requires": {
"python_version": "3.8"
},
"sources": [
{
"name": "pypi",
"url": "https://pypi.tuna.tsinghua.edu.cn/simple",
"verify_ssl": true
}
]
},
"default": {
"get": {
"hashes": [
"sha256:688268840f923255932154a52bdd40ffac467de4126835c43846e9e6f112844c"
],
"version": "==2019.4.13"
},
Norma length : 1,490 lines : 53 Ln : 1 Col : 1 Sel : 0 | 0 Unix (LF) UTF-8 INS
Create a lockfile containing pre-releases:
$ pipenv lock --pre
Show a graph of your installed dependencies:
$ pipenv graph
Check your installed dependencies for security vulnerabilities:
$ pipenv check
Install a local setup.py into your virtual environment/Pipfile:
$ pipenv install -e .
Use a lower-level pip command:
$ pipenv run pip freeze
Commands:
check Checks for security vulnerabilities and against PEP 508 markers provided in Pipfile.
clean Uninstalls all packages not specified in Pipfile.lock.
graph Displays currently-installed dependency graph information.
install Installs provided packages and adds them to Pipfile, or (if no packages are given), installs all packages from Pipfile.
lock Generates Pipfile.lock.
open View a given module in your editor.
run Spawns a command installed into the virtualenv.
shell Spawns a shell within the virtualenv.
sync Installs all packages specified in Pipfile.lock.
uninstall Un-installs a provided package and removes it from Pipfile.
update Runs lock, then sync.
d:\CODE\PYTHON\007>pipenv shell
Launching subshell in virtual environment…
Microsoft Windows [版本 10.0.18363.720]
(c) 2019 Microsoft Corporation。保留所有权利。
(007-syKnwjqC) d:\CODE\PYTHON\007>pipenv install request
Installing request…
Adding request to Pipfile's [packages]…
Installation Succeeded
Pipfile.lock not found, creating…
Locking [dev-packages] dependencies…
Locking [packages] dependencies…
Success!
Updated Pipfile.lock (e1c616)!
Installing dependencies from Pipfile.lock (e1c616)…
========== 5/5 - 00:00:02
(007-syKnwjqC) d:\CODE\PYTHON\007>


使用pipenv的第3步：
在虚拟环境中进行操作
1
激活某个虚拟环境后，就可以在该虚拟环境中进行操作了
使用Python解释器
使用pip、pip3和easy_install等工具
使用下载的第三方库
2
激活某个虚拟环境后
当前的Python解释器就是该虚拟环境中的Python解释器
当前的pip、pip3和easy_install等工具就是该虚拟环境中的相应工具
当前的第三方库就是该虚拟环境中的第三方库
3
简单验证（以Windows为例）
执行where python
执行where pip
执行pip --version
4
第三方库的安装和卸载
不要使用pip而是要使用pipenv
安装第三方库的语法是：pipenv install 第三方库
卸载第三方库的语法是：pipenv uninstall 第三方库
pipenv = pip + virtualenv，pipenv已经把pip和虚拟环境有机地结合在一起了
使用Pipfile和Pipfile.lock替代了requirements.txt
与requirements.txt相比
记录的信息更加丰富
不需要手动生成和手动更新
自动生成和自动维护
如果不使用pipenv而是使用pip安装和卸载第三方库，那么Pipfile和Pipfile.lock就不会有任何更新
可以执行pipenv clean以卸载掉所有在Pipfile.lock中不存在的第三方库
执行pipenv graph后可以以树形结构列出安装的所有第三方库以及它们之间的依赖关系图
5
pipenv还提供了很多可选参数
pipenv --py
查看当前虚拟环境中Python解释器的完整路径
pipenv --where
查看当前虚拟环境对应项目的完整路径
pipenv --venv
查看当前虚拟环境的完整路径




```
Microsoft Windows [版本 10.0.18363.592]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\WINDOWS\system32>cd C:\Users\[redacted]\Desktop\MyP1

C:\Users\[redacted]\Desktop\MyP1>pipenv install --python 3.8
Creating a virtualenv for this project…
Pipfile: C:\Users\[redacted]\Desktop\MyP1\Pipfile
Using C:/Program Files/Python38/python.exe (3.8.1) to create virtualenv…
[=== ] Creating virtual environment...Already using interpreter C:\Program Files\Python38\python.exe
Using base prefix 'C:\\Program Files\\Python38'
New python executable in C:\Users\[redacted]\MyVirEnvs\MyP1-ewNaBP9D\Scripts\python.exe
Installing setuptools, pip, wheel...
done.
Running virtualenv with interpreter C:/Program Files/Python38/python.exe

Successfully created virtual environment!
Virtualenv location: C:\Users\[redacted]\MyVirEnvs\MyP1-ewNaBP9D
Installing dependencies from Pipfile.lock (2cabd8)…
  ================================ 1/1 - 00:00:08
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.

C:\Users\[redacted]\Desktop\MyP1>pipenv shell
Launching subshell in virtual environment…
Microsoft Windows [版本 10.0.18363.592]
(c) 2019 Microsoft Corporation。保留所有权利。

(MyP1-ewNaBP9D) C:\Users\[redacted]\Desktop\MyP1>
```

pipfile 的内容

```
[[source]]
name = "pypi"
url = "https://pypi.tuna.tsinghua.edu.cn/simple"
verify_ssl = true

[dev-packages]

[packages]

[requires]
python_version = "3.8"
```

4.poetry

poetry 与 pipenv 类似.支持打包上传功能，支持更强大更复杂的功能。

Poetry 安装

python install-poetry.py 或

python install-poetry.py --git https://gitee.com/builderzou/poetry

[install-poetry.py 文件内容]

```
==================================
"""
This script will install Poetry and its dependencies.

It does, in order:

  - Downloads the virtualenv package to a temporary directory and add it to sys.path.
  - Creates a virtual environment in the correct OS data dir which will be
      - `%APPDATA%\\pypoetry` on Windows
      -  ~/Library/Application Support/pypoetry on MacOS
      - `${XDG_DATA_HOME}/pypoetry` (or `~/.local/share/pypoetry` if it's not set) on UNIX systems
      - In `${POETRY_HOME}` if it's set.
  - Installs the latest or given version of Poetry inside this virtual environment.
  - Installs a `poetry` script in the Python user directory (or `${POETRY_HOME/bin}` if `POETRY_HOME`
is set).
"""

import argparse
import json
import os
import re
import shutil
import site
import subprocess
import sys
import tempfile

from contextlib import closing
from contextlib import contextmanager
from functools import cmp_to_key
from io import UnsupportedOperation
from pathlib import Path
from typing import Optional
```

```
from urllib.request import Request
from urllib.request import urlopen


SHELL = os.getenv("SHELL", "")
WINDOWS = sys.platform.startswith("win") or (sys.platform == "cli" and os.name == "nt")
MACOS = sys.platform == "darwin"

FOREGROUND_COLORS = {
    "black": 30,
    "red": 31,
    "green": 32,
    "yellow": 33,
    "blue": 34,
    "magenta": 35,
    "cyan": 36,
    "white": 37,
}

BACKGROUND_COLORS = {
    "black": 40,
    "red": 41,
    "green": 42,
    "yellow": 43,
    "blue": 44,
    "magenta": 45,
    "cyan": 46,
    "white": 47,
}

OPTIONS = {"bold": 1, "underscore": 4, "blink": 5, "reverse": 7, "conceal": 8}


def style(fg, bg, options):
    codes = []

    if fg:
        codes.append(FOREGROUND_COLORS[fg])

    if bg:
```

```
        codes.append(BACKGROUND_COLORS[bg])

    if options:
        if not isinstance(options, (list, tuple)):
            options = [options]

        for option in options:
            codes.append(OPTIONS[option])

    return "\033[{}m".format(";".join(map(str, codes)))


STYLES = {
    "info": style("cyan", None, None),
    "comment": style("yellow", None, None),
    "success": style("green", None, None),
    "error": style("red", None, None),
    "warning": style("yellow", None, None),
    "b": style(None, None, ("bold",)),
}


def is_decorated():
    if WINDOWS:
        return (
            os.getenv("ANSICON") is not None
            or "ON" == os.getenv("ConEmuANSI")
            or "xterm" == os.getenv("Term")
        )

    if not hasattr(sys.stdout, "fileno"):
        return False

    try:
        return os.isatty(sys.stdout.fileno())
    except UnsupportedOperation:
        return False


def is_interactive():
```

```
    if not hasattr(sys.stdin, "fileno"):
        return False

    try:
        return os.isatty(sys.stdin.fileno())
    except UnsupportedOperation:
        return False


def colorize(style, text):
    if not is_decorated():
        return text

    return "{}{}\033[0m".format(STYLES[style], text)


def string_to_bool(value):
    value = value.lower()

    return value in {"true", "1", "y", "yes"}


def data_dir(version: Optional[str] = None) -> Path:
    if os.getenv("POETRY_HOME"):
        return Path(os.getenv("POETRY_HOME")).expanduser()

    if WINDOWS:
        const = "CSIDL_APPDATA"
        path = os.path.normpath(_get_win_folder(const))
        path = os.path.join(path, "pypoetry")
    elif MACOS:
        path = os.path.expanduser("~/Library/Application Support/pypoetry")
    else:
        path = os.getenv("XDG_DATA_HOME", os.path.expanduser("~/.local/share"))
        path = os.path.join(path, "pypoetry")

    if version:
        path = os.path.join(path, version)

    return Path(path)
```

```
def bin_dir(version: Optional[str] = None) -> Path:
    if os.getenv("POETRY_HOME"):
        return Path(os.getenv("POETRY_HOME"), "bin").expanduser()

    user_base = site.getuserbase()

    if WINDOWS:
        bin_dir = os.path.join(user_base, "Scripts")
    else:
        bin_dir = os.path.join(user_base, "bin")

    return Path(bin_dir)


def _get_win_folder_from_registry(csidl_name):
    import winreg as _winreg

    shell_folder_name = {
        "CSIDL_APPDATA": "AppData",
        "CSIDL_COMMON_APPDATA": "Common AppData",
        "CSIDL_LOCAL_APPDATA": "Local AppData",
    }[csidl_name]

    key = _winreg.OpenKey(
        _winreg.HKEY_CURRENT_USER,
        r"Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders",
    )
    dir, type = _winreg.QueryValueEx(key, shell_folder_name)

    return dir


def _get_win_folder_with_ctypes(csidl_name):
    import ctypes

    csidl_const = {
        "CSIDL_APPDATA": 26,
        "CSIDL_COMMON_APPDATA": 35,
```

```
        "CSIDL_LOCAL_APPDATA": 28,
    }[csidl_name]

    buf = ctypes.create_unicode_buffer(1024)
    ctypes.windll.shell32.SHGetFolderPathW(None, csidl_const, None, 0, buf)

    # Downgrade to short path name if have highbit chars. See
    # <"" target="_blank">http://bugs.activestate.com/show_bug.cgi?id=85099>;.
    has_high_char = False
    for c in buf:
        if ord(c) > 255:
            has_high_char = True
            break
    if has_high_char:
        buf2 = ctypes.create_unicode_buffer(1024)
        if ctypes.windll.kernel32.GetShortPathNameW(buf.value, buf2, 1024):
            buf = buf2

    return buf.value


if WINDOWS:
    try:
        from ctypes import windll  # noqa

        _get_win_folder = _get_win_folder_with_ctypes
    except ImportError:
        _get_win_folder = _get_win_folder_from_registry


@contextmanager
def temporary_directory(*args, **kwargs):
    try:
        from tempfile import TemporaryDirectory
    except ImportError:
        name = tempfile.mkdtemp(*args, **kwargs)

        yield name

        shutil.rmtree(name)
```

```
    else:
        with TemporaryDirectory(*args, **kwargs) as name:
            yield name


PRE_MESSAGE = """# Welcome to {poetry}!

This will download and install the latest version of {poetry},
a dependency and package manager for Python.

It will add the `poetry` command to {poetry}'s bin directory, located at:

{poetry_home_bin}

You can uninstall at any time by executing this script with the --uninstall option,
and these changes will be reverted.
"""

POST_MESSAGE = """{poetry} ({version}) is installed now. Great!

You can test that everything is set up by executing:

`{test_command}`
"""

POST_MESSAGE_NOT_IN_PATH = """{poetry} ({version}) is installed now. Great!

To get started you need {poetry}'s bin directory ({poetry_home_bin}) in your `PATH`
environment variable.
{configure_message}
Alternatively, you can call {poetry} explicitly with `{poetry_executable}`.

You can test that everything is set up by executing:

`{test_command}`
"""

POST_MESSAGE_CONFIGURE_UNIX = """
Add `export PATH="{poetry_home_bin}:$PATH"` to your shell configuration file.
"""
```

```
POST_MESSAGE_CONFIGURE_FISH = """
You can execute `set -U fish_user_paths {poetry_home_bin} $fish_user_paths`
"""

POST_MESSAGE_CONFIGURE_WINDOWS = """"""


class Cursor:
    def __init__(self) -> None:
        self._output = sys.stdout

    def move_up(self, lines: int = 1) -> "Cursor":
        self._output.write("\x1b[{}A".format(lines))

        return self

    def move_down(self, lines: int = 1) -> "Cursor":
        self._output.write("\x1b[{}B".format(lines))

        return self

    def move_right(self, columns: int = 1) -> "Cursor":
        self._output.write("\x1b[{}C".format(columns))

        return self

    def move_left(self, columns: int = 1) -> "Cursor":
        self._output.write("\x1b[{}D".format(columns))

        return self

    def move_to_column(self, column: int) -> "Cursor":
        self._output.write("\x1b[{}G".format(column))

        return self

    def move_to_position(self, column: int, row: int) -> "Cursor":
        self._output.write("\x1b[{};{}H".format(row + 1, column))
```

```
        return self

    def save_position(self) -> "Cursor":
        self._output.write("\x1b7")

        return self

    def restore_position(self) -> "Cursor":
        self._output.write("\x1b8")

        return self

    def hide(self) -> "Cursor":
        self._output.write("\x1b[?25l")

        return self

    def show(self) -> "Cursor":
        self._output.write("\x1b[?25h\x1b[?0c")

        return self

    def clear_line(self) -> "Cursor":
        """
        Clears all the output from the current line.
        """
        self._output.write("\x1b[2K")

        return self

    def clear_line_after(self) -> "Cursor":
        """
        Clears all the output from the current line after the current position.
        """
        self._output.write("\x1b[K")

        return self

    def clear_output(self) -> "Cursor":
        """
```

```
        Clears all the output from the cursors' current position
        to the end of the screen.
        """
        self._output.write("\x1b[0J")

        return self

    def clear_screen(self) -> "Cursor":
        """
        Clears the entire screen.
        """
        self._output.write("\x1b[2J")

        return self


class Installer:
    METADATA_URL = "https://pypi.org/pypi/poetry/json"
    VERSION_REGEX = re.compile(
        r"v?(\d+)(?:\.(\d+))?(?:\.(\d+))?(?:\.(\d+))?"
        "("
        "[._-]?"
        r"(?:(stable|beta|b|rc|RC|alpha|a|patch|pl|p)((?:[.-]?\d+)*)?)?"
        "([.-]?dev)?"
        ")?"
        r"(?:\+[^\s]+)?"
    )

    def __init__(
        self,
        version: Optional[str] = None,
        preview: bool = False,
        force: bool = False,
        accept_all: bool = False,
        git: Optional[str] = None,
        path: Optional[str] = None,
    ) -> None:
        self._version = version
        self._preview = preview
        self._force = force
```

```
        self._accept_all = accept_all
        self._git = git
        self._path = path
        self._data_dir = data_dir()
        self._bin_dir = bin_dir()
        self._cursor = Cursor()

    def allows_prereleases(self) -> bool:
        return self._preview

    def run(self) -> int:
        if self._git:
            version = self._git
        elif self._path:
            version = self._path
        else:
            version, current_version = self.get_version()

        if version is None:
            return 0

        self.display_pre_message()
        self.ensure_directories()

        def _is_self_upgrade_supported(x):
            mx = self.VERSION_REGEX.match(x)

            if mx is None:
                # the version is not semver, perhaps scm or file, we assume upgrade is supported
                return True

            vx = tuple(int(p) for p in mx.groups()[:3]) + (mx.group(5),)
            return vx >= (1, 1, 7)

        if version and not _is_self_upgrade_supported(version):
            self._write(
                colorize(
                    "warning",
                    f"You are installing {version}. When using the current installer, this version does not
support "
```

```
                f"updating using the 'self update' command. Please use 1.1.7 or later.",
            )
        )
        if not self._accept_all:
            continue_install = input("Do you want to continue? ([y]/n) ") or "y"
            if continue_install.lower() in {"n", "no"}:
                return 0

    try:
        self.install(version)
    except subprocess.CalledProcessError as e:
        print(
            colorize("error", f"\nAn error has occurred: {e}\n{e.stderr.decode()}")
        )

        return e.returncode

    self._write("")
    self.display_post_message(version)

    return 0

def install(self, version, upgrade=False):
    """
    Installs Poetry in $POETRY_HOME.
    """
    self._write(
        "Installing {} ({})".format(
            colorize("info", "Poetry"), colorize("info", version)
        )
    )

    env_path = self.make_env(version)
    self.install_poetry(version, env_path)
    self.make_bin(version)

    self._overwrite(
        "Installing {} ({}): {}".format(
            colorize("info", "Poetry"),
            colorize("b", version),
```

```
            colorize("success", "Done"),
        )
    )

    self._data_dir.joinpath("VERSION").write_text(version)

    return 0

def uninstall(self) -> int:
    if not self._data_dir.exists():
        self._write(
            "{} is not currently installed.".format(colorize("info", "Poetry"))
        )

        return 1

    version = None
    if self._data_dir.joinpath("VERSION").exists():
        version = self._data_dir.joinpath("VERSION").read_text().strip()

    if version:
        self._write(
            "Removing {} ({})".format(
                colorize("info", "Poetry"), colorize("b", version)
            )
        )
    else:
        self._write("Removing {}".format(colorize("info", "Poetry")))

    shutil.rmtree(str(self._data_dir))
    for script in ["poetry", "poetry.bat"]:
        if self._bin_dir.joinpath(script).exists():
            self._bin_dir.joinpath(script).unlink()

    return 0

def make_env(self, version: str) -> Path:
    self._overwrite(
        "Installing {} ({}): {}".format(
            colorize("info", "Poetry"),
```

```
            colorize("b", version),
            colorize("comment", "Creating environment"),
        )
    )

    env_path = self._data_dir.joinpath("venv")

    with temporary_directory() as tmp_dir:
        subprocess.call(
            [sys.executable, "-m", "pip", "install", "virtualenv", "-t", tmp_dir],
            stdout=subprocess.PIPE,
            stderr=subprocess.STDOUT,
        )

        sys.path.insert(0, tmp_dir)

        import virtualenv

        virtualenv.cli_run([str(env_path), "--clear"])

    return env_path

def make_bin(self, version: str) -> None:
    self._overwrite(
        "Installing {} ({}): {}".format(
            colorize("info", "Poetry"),
            colorize("b", version),
            colorize("comment", "Creating script"),
        )
    )

    self._bin_dir.mkdir(parents=True, exist_ok=True)

    script = "poetry"
    target_script = "venv/bin/poetry"
    if WINDOWS:
        script = "poetry.exe"
        target_script = "venv/Scripts/poetry.exe"

    if self._bin_dir.joinpath(script).exists():
```

```
            self._bin_dir.joinpath(script).unlink()

        try:
            self._bin_dir.joinpath(script).symlink_to(
                self._data_dir.joinpath(target_script)
            )
        except OSError:
            # This can happen if the user
            # does not have the correct permission on Windows
            shutil.copy(
                self._data_dir.joinpath(target_script), self._bin_dir.joinpath(script)
            )

    def install_poetry(self, version: str, env_path: Path) -> None:
        self._overwrite(
            "Installing {} ({}): {}".format(
                colorize("info", "Poetry"),
                colorize("b", version),
                colorize("comment", "Installing Poetry"),
            )
        )

        if WINDOWS:
            python = env_path.joinpath("Scripts/python.exe")
        else:
            python = env_path.joinpath("bin/python")

        if self._git:
            specification = "git+" + version
        elif self._path:
            specification = version
        else:
            specification = f"poetry=={version}"

        subprocess.run(
            [str(python), "-m", "pip", "install", specification],
            stdout=subprocess.PIPE,
            stderr=subprocess.STDOUT,
            check=True,
        )
```

```
def display_pre_message(self) -> None:
    kwargs = {
        "poetry": colorize("info", "Poetry"),
        "poetry_home_bin": colorize("comment", self._bin_dir),
    }
    self._write(PRE_MESSAGE.format(**kwargs))

def display_post_message(self, version: str) -> None:
    if WINDOWS:
        return self.display_post_message_windows(version)

    if SHELL == "fish":
        return self.display_post_message_fish(version)

    return self.display_post_message_unix(version)

def display_post_message_windows(self, version: str) -> None:
    path = self.get_windows_path_var()

    message = POST_MESSAGE_NOT_IN_PATH
    if path and str(self._bin_dir) in path:
        message = POST_MESSAGE

    self._write(
        message.format(
            poetry=colorize("info", "Poetry"),
            version=colorize("b", version),
            poetry_home_bin=colorize("comment", self._bin_dir),
            poetry_executable=colorize("b", self._bin_dir.joinpath("poetry")),
            configure_message=POST_MESSAGE_CONFIGURE_WINDOWS.format(
                poetry_home_bin=colorize("comment", self._bin_dir)
            ),
            test_command=colorize("b", "poetry --version"),
        )
    )

def get_windows_path_var(self) -> Optional[str]:
    import winreg
```

```
    with winreg.ConnectRegistry(None, winreg.HKEY_CURRENT_USER) as root:
        with winreg.OpenKey(root, "Environment", 0, winreg.KEY_ALL_ACCESS) as key:
            path, _ = winreg.QueryValueEx(key, "PATH")

            return path

def display_post_message_fish(self, version: str) -> None:
    fish_user_paths = subprocess.check_output(
        ["fish", "-c", "echo $fish_user_paths"]
    ).decode("utf-8")

    message = POST_MESSAGE_NOT_IN_PATH
    if fish_user_paths and str(self._bin_dir) in fish_user_paths:
        message = POST_MESSAGE

    self._write(
        message.format(
            poetry=colorize("info", "Poetry"),
            version=colorize("b", version),
            poetry_home_bin=colorize("comment", self._bin_dir),
            poetry_executable=colorize("b", self._bin_dir.joinpath("poetry")),
            configure_message=POST_MESSAGE_CONFIGURE_FISH.format(
                poetry_home_bin=colorize("comment", self._bin_dir)
            ),
            test_command=colorize("b", "poetry --version"),
        )
    )

def display_post_message_unix(self, version: str) -> None:
    paths = os.getenv("PATH", "").split(":")

    message = POST_MESSAGE_NOT_IN_PATH
    if paths and str(self._bin_dir) in paths:
        message = POST_MESSAGE

    self._write(
        message.format(
            poetry=colorize("info", "Poetry"),
            version=colorize("b", version),
            poetry_home_bin=colorize("comment", self._bin_dir),
```

```
            poetry_executable=colorize("b", self._bin_dir.joinpath("poetry")),
            configure_message=POST_MESSAGE_CONFIGURE_UNIX.format(
                poetry_home_bin=colorize("comment", self._bin_dir)
            ),
            test_command=colorize("b", "poetry --version"),
        )
    )

def ensure_directories(self) -> None:
    self._data_dir.mkdir(parents=True, exist_ok=True)
    self._bin_dir.mkdir(parents=True, exist_ok=True)

def get_version(self):
    current_version = None
    if self._data_dir.joinpath("VERSION").exists():
        current_version = self._data_dir.joinpath("VERSION").read_text().strip()

    self._write(colorize("info", "Retrieving Poetry metadata"))

    metadata = json.loads(self._get(self.METADATA_URL).decode())

    def _compare_versions(x, y):
        mx = self.VERSION_REGEX.match(x)
        my = self.VERSION_REGEX.match(y)

        vx = tuple(int(p) for p in mx.groups()[:3]) + (mx.group(5),)
        vy = tuple(int(p) for p in my.groups()[:3]) + (my.group(5),)

        if vx < vy:
            return -1
        elif vx > vy:
            return 1

        return 0

    self._write("")
    releases = sorted(
        metadata["releases"].keys(), key=cmp_to_key(_compare_versions)
    )
```

```
    if self._version and self._version not in releases:
        self._write(
            colorize("error", "Version {} does not exist.".format(self._version))
        )

        return None, None

    version = self._version
    if not version:
        for release in reversed(releases):
            m = self.VERSION_REGEX.match(release)
            if m.group(5) and not self.allows_prereleases():
                continue

            version = release

            break

    if current_version == version and not self._force:
        self._write(
            "The latest version ({}) is already installed.".format(
                colorize("b", version)
            )
        )

        return None, current_version

    return version, current_version

def _write(self, line) -> None:
    sys.stdout.write(line + "\n")

def _overwrite(self, line) -> None:
    if not is_decorated():
        return self._write(line)

    self._cursor.move_up()
    self._cursor.clear_line()
    self._write(line)
```

```
    def _get(self, url):
        request = Request(url, headers={"User-Agent": "Python Poetry"})

        with closing(urlopen(request)) as r:
            return r.read()


def main():
    parser = argparse.ArgumentParser(
        description="Installs the latest (or given) version of poetry"
    )
    parser.add_argument(
        "-p",
        "--preview",
        help="install preview version",
        dest="preview",
        action="store_true",
        default=False,
    )
    parser.add_argument("--version", help="install named version", dest="version")
    parser.add_argument(
        "-f",
        "--force",
        help="install on top of existing version",
        dest="force",
        action="store_true",
        default=False,
    )
    parser.add_argument(
        "-y",
        "--yes",
        help="accept all prompts",
        dest="accept_all",
        action="store_true",
        default=False,
    )
    parser.add_argument(
        "--uninstall",
        help="uninstall poetry",
        dest="uninstall",
```

```
    action="store_true",
    default=False,
)
parser.add_argument(
    "--path",
    dest="path",
    action="store",
    help=(
        "Install from a given path (file or directory) instead of "
        "fetching the latest version of Poetry available online."
    ),
)
parser.add_argument(
    "--git",
    dest="git",
    action="store",
    help=(
        "Install from a git repository instead of fetching the latest version "
        "of Poetry available online."
    ),
)

args = parser.parse_args()

installer = Installer(
    version=args.version or os.getenv("POETRY_VERSION"),
    preview=args.preview or string_to_bool(os.getenv("POETRY_PREVIEW", "0")),
    force=args.force,
    accept_all=args.accept_all
    or string_to_bool(os.getenv("POETRY_ACCEPT", "0"))
    or not is_interactive(),
    path=args.path,
    git=args.git,
)

if args.uninstall or string_to_bool(os.getenv("POETRY_UNINSTALL", "0")):
    return installer.uninstall()

return installer.run()
```

```
if __name__ == "__main__":
    sys.exit(main())
```

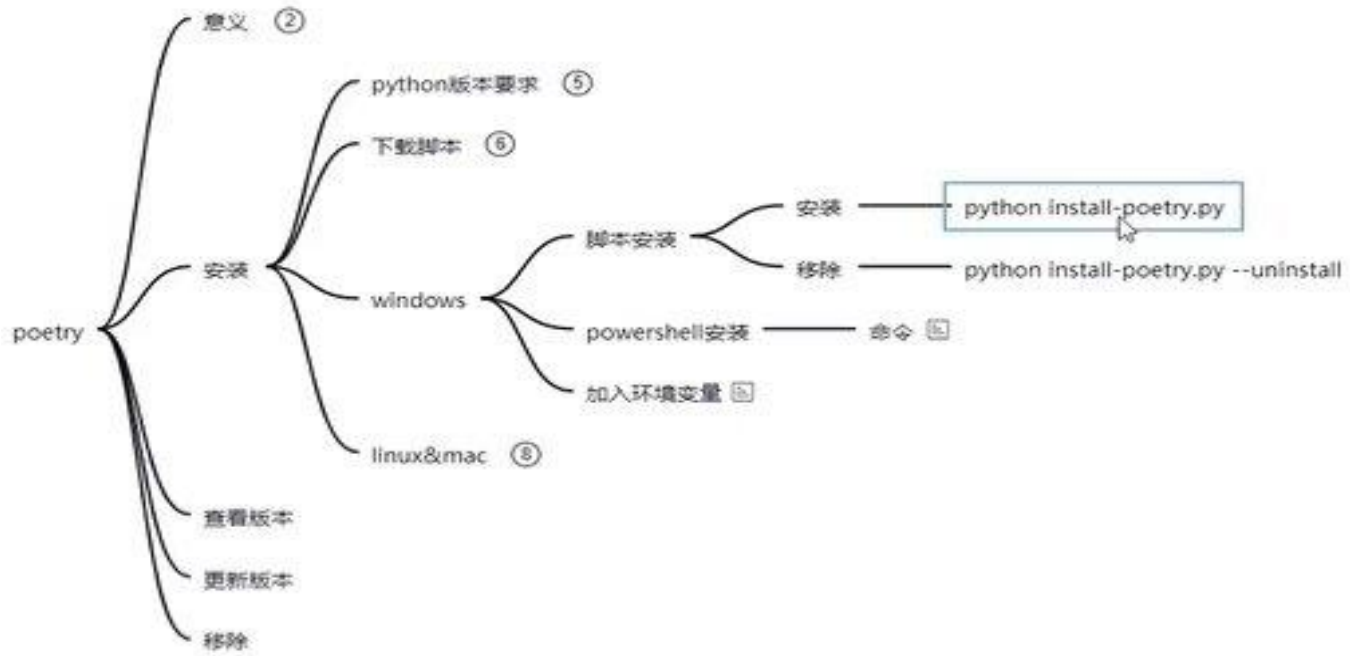
==========

toml 文件增加

[[tool.poetry.source]]
name = "aliyun"
url = "https://mirrors.aliyun.com/pypi/simple/"
poetry init
poetry install
poetry shell

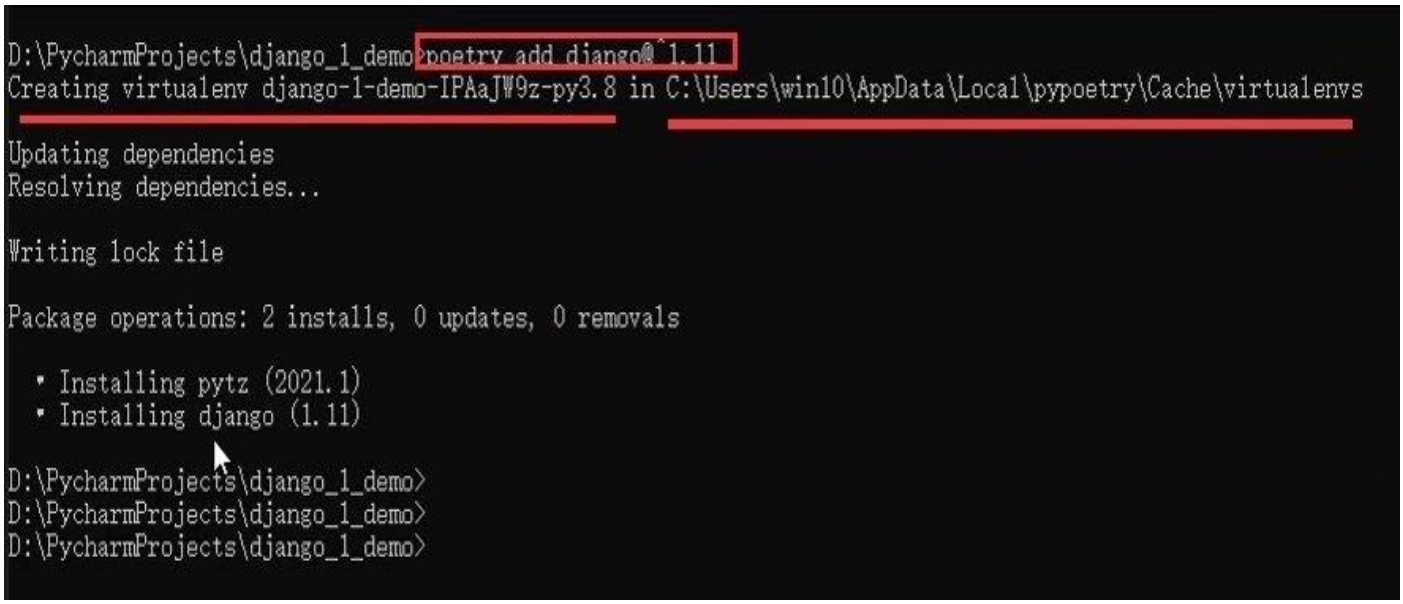
poetry show -t #以树状显示包的依赖关系统
poetry config --list

卸载

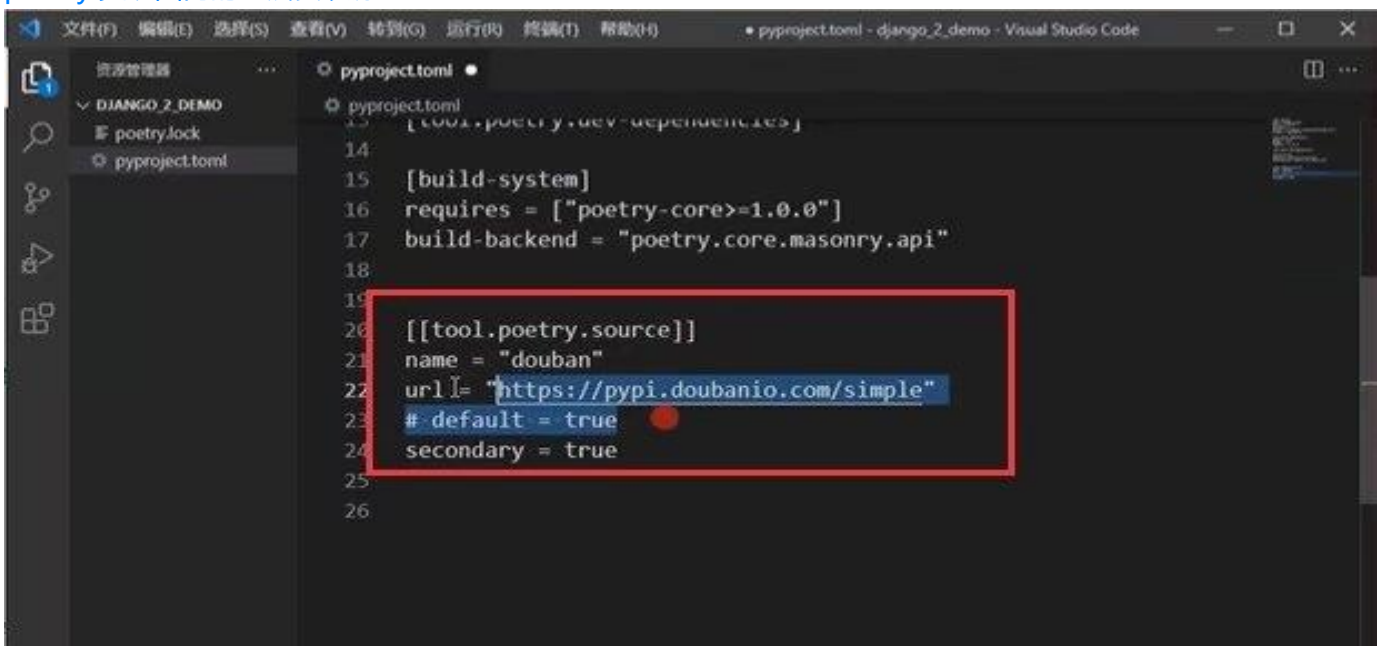



```
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

export PYENV_ROOT="$HOME/.pyenv"
export PATH="$PYENV_ROOT/bin:$PATH"
eval "$(pyenv init --path)"

if command -v pyenv >/dev/null; then
    eval "$(pyenv init -)"
fi

export PATH="/root/.local/bin:$PATH"
~
~
```

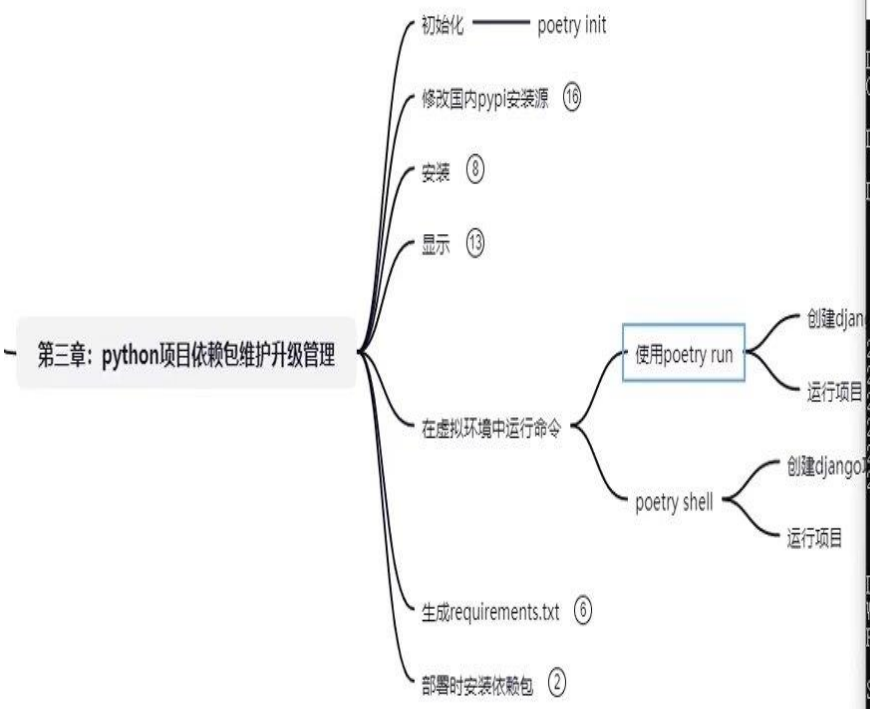
Poetry 安装包 [poetry add]

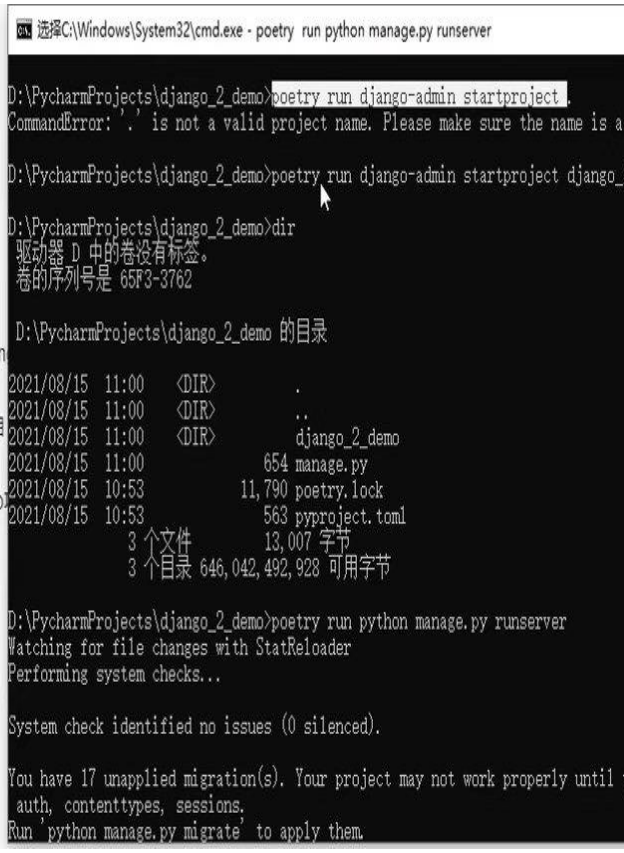


poetry 更改国内的豆瓣安装源

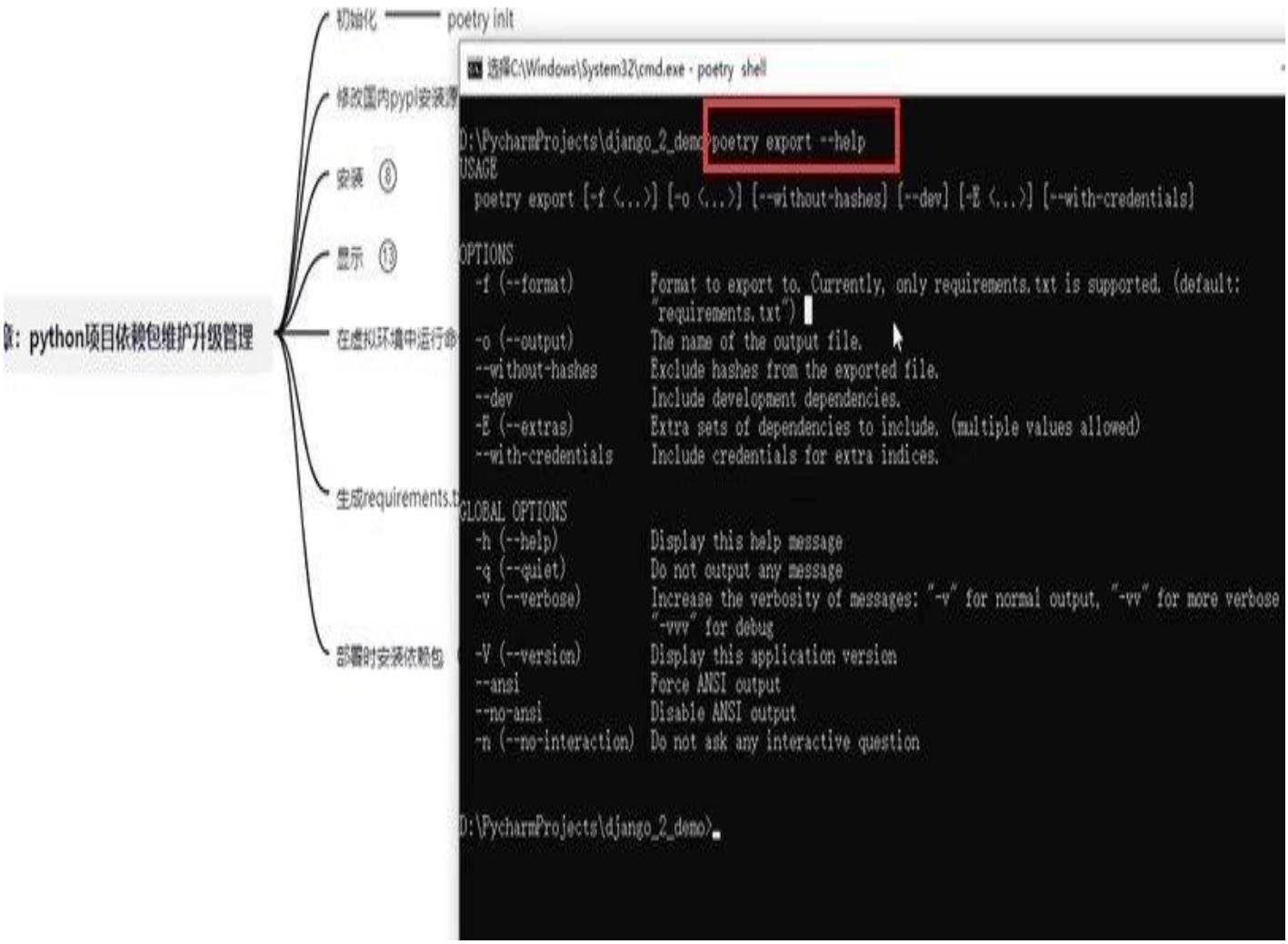


poetry run 运行命令

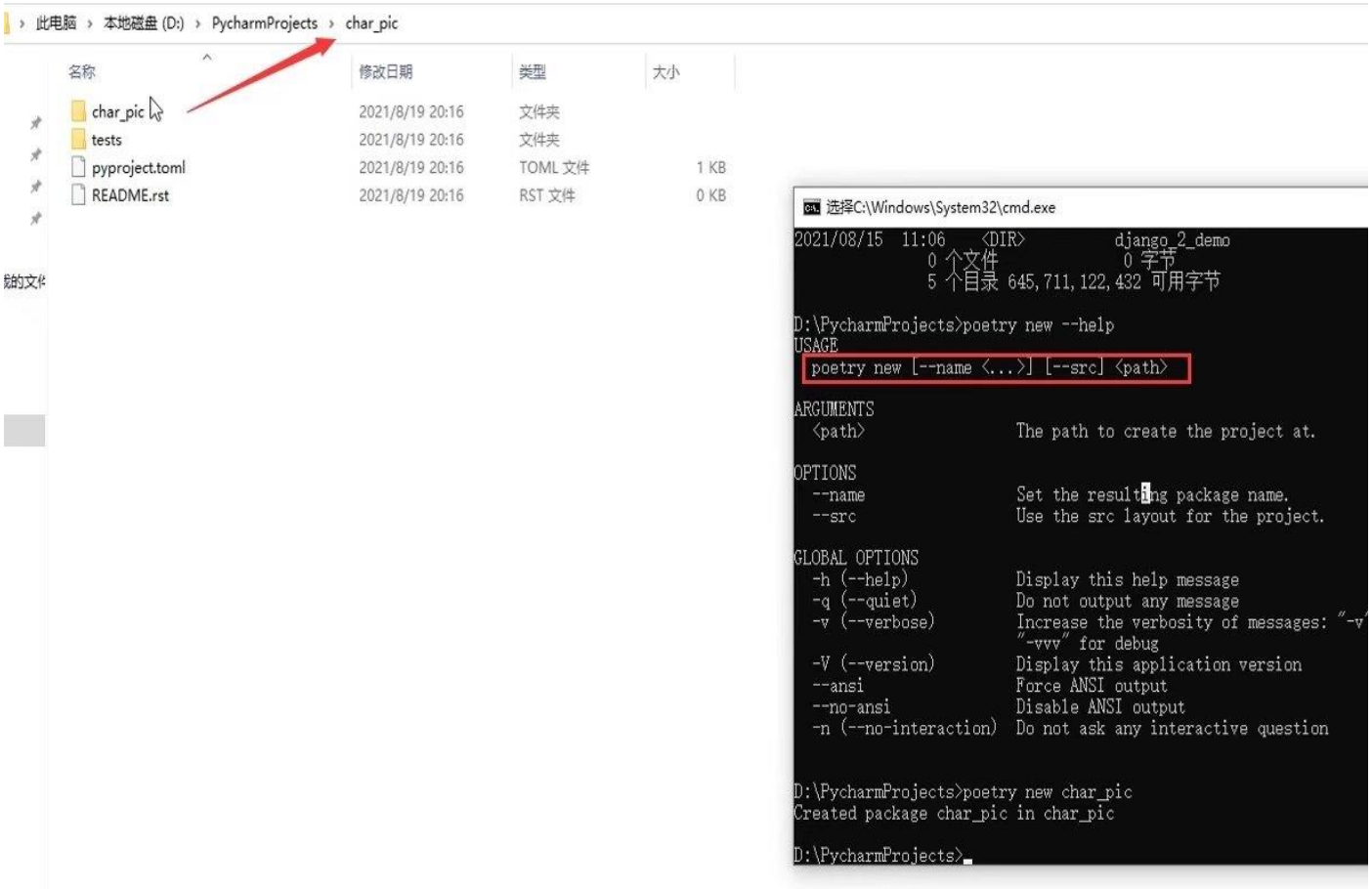


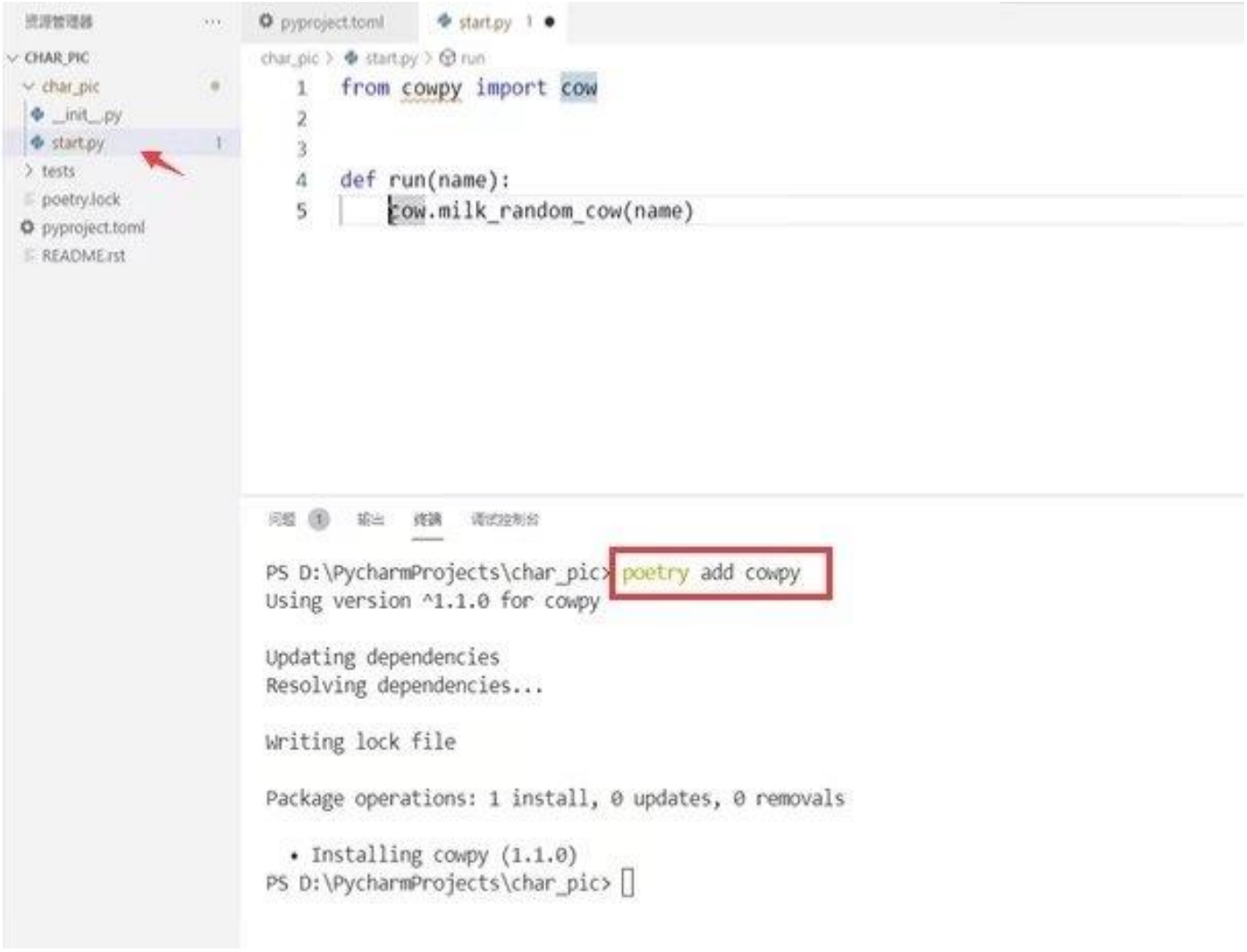


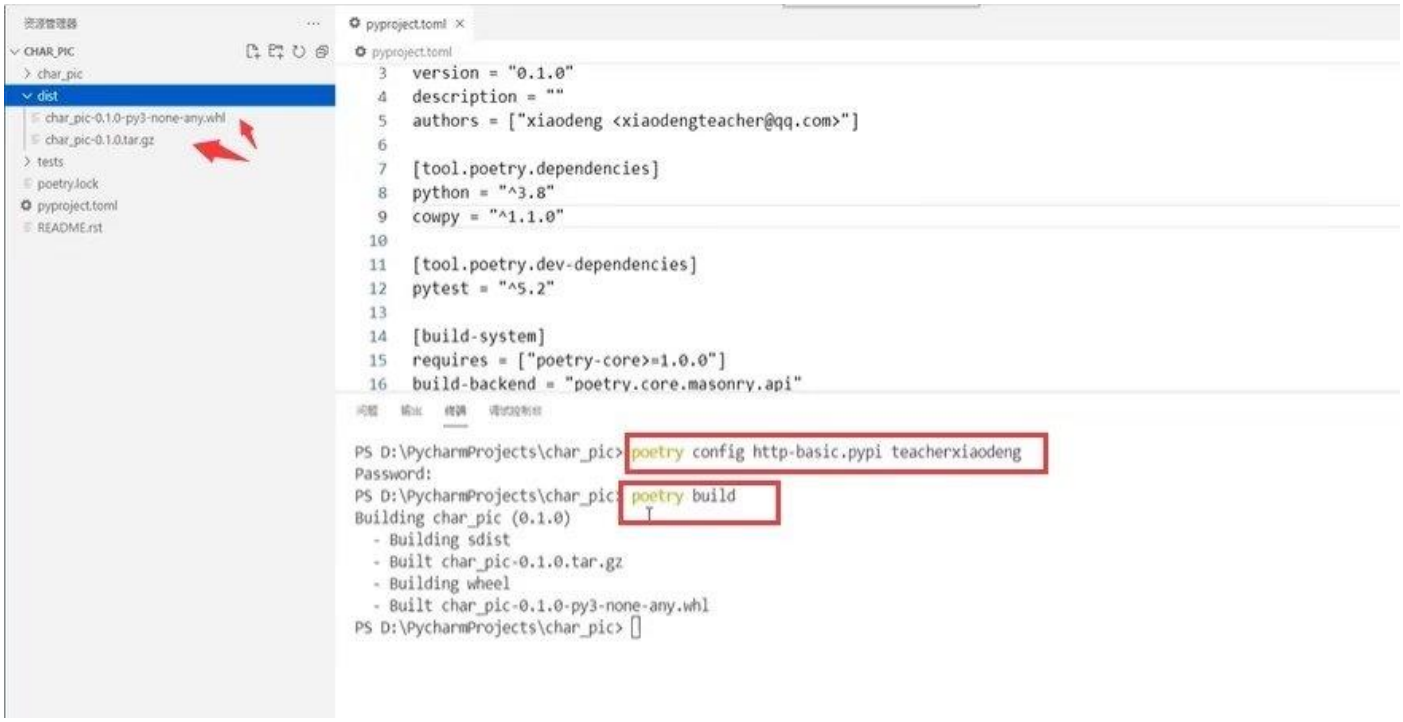
poetry –help 查看帮助




```
1 xiaodeng  2 xiaodeng
db.sqlite3  django_2_demo  manage.py  pyproject.toml  req.txt  requirements.txt
[root@instance-k0fsji48 django_2_demo]# poetry install
Creating virtualenv django-2-demo-Wm76rbA7-py3.6 in /root/.cache/pypoetry/virtualenvs
Updating dependencies
Resolving dependencies... (10.4s)

  SolverProblemError

  The current project's Python requirement (>=3,<4) is not compatible with some of the required packages Python
    - djangorestframework requiresIPython >=3.5, so it will not be satisfied for Python >=3,<3.5

  Because no versions of djangorestframework match >3.12.4,<4.0.0
   and djangorestframework (3.12.4) requires Python >=3.5, djangorestframework is forbidden.
  So, because django-2-demo depends on djangorestframework (^3.12.4), version solving failed.
```


此电脑 › 本地磁盘 (D:) › PycharmProjects › char_pic
名称
修改日期
类型
大小
char_pic 2021/8/19 20:16 文件夹
tests 2021/8/19 20:16 文件夹
pyproject.toml 2021/8/19 20:16 TOML 文件 1 KB
README.rst 2021/8/19 20:16 RST 文件 0 KB
选择C:\Windows\System32\cmd.exe
2021/08/15 11:06 <DIR> django_2_demo
0 个文件 0 字节
5 个目录 645,711,122,432 可用字节
D:\PycharmProjects>poetry new --help
USAGE
poetry new [--name <...>] [--src] <path>
ARGUMENTS
<path> The path to create the project at.
OPTIONS
--name Set the resulting package name.
--src Use the src layout for the project.
GLOBAL OPTIONS
-h (--help) Display this help message
-q (--quiet) Do not output any message
-v (--verbose) Increase the verbosity of messages: "-v
"-vvv" for debug
-V (--version) Display this application version
--ansi Force ANSI output
--no-ansi Disable ANSI output
-n (--no-interaction) Do not ask any interactive question
D:\PycharmProjects>poetry new char_pic
Created package char_pic in char_pic
D:\PycharmProjects>

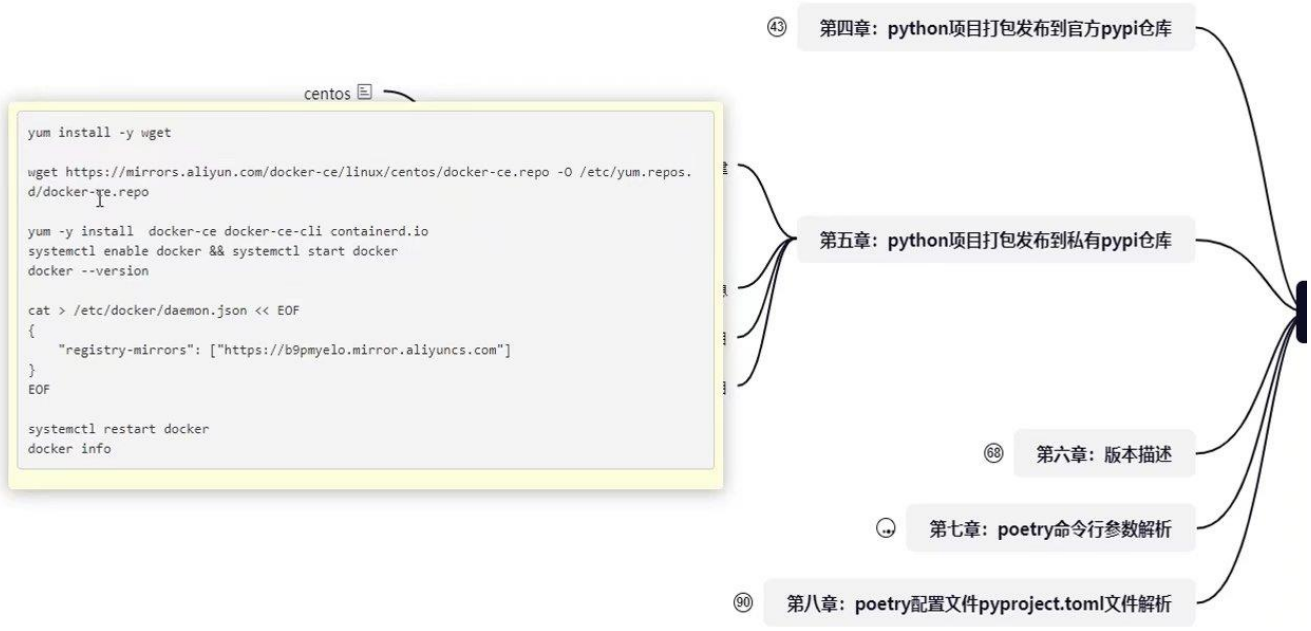

CHAR_PIC
char_pic
__init__.py
start.py
tests
poetry.lock
pyproject.toml
README.rst
pyproject.toml
start.py
char_pic > start.py > run
1 from cowpy import cow
2
3
4 def run(name):
5 cow.milk_random_cow(name)
问题 1 输出 终端 调试控制台
PS D:\PycharmProjects\char_pic> poetry add cowpy
Using version ^1.1.0 for cowpy
Updating dependencies
Resolving dependencies...
Writing lock file
Package operations: 1 install, 0 updates, 0 removals
• Installing cowpy (1.1.0)
PS D:\PycharmProjects\char_pic>

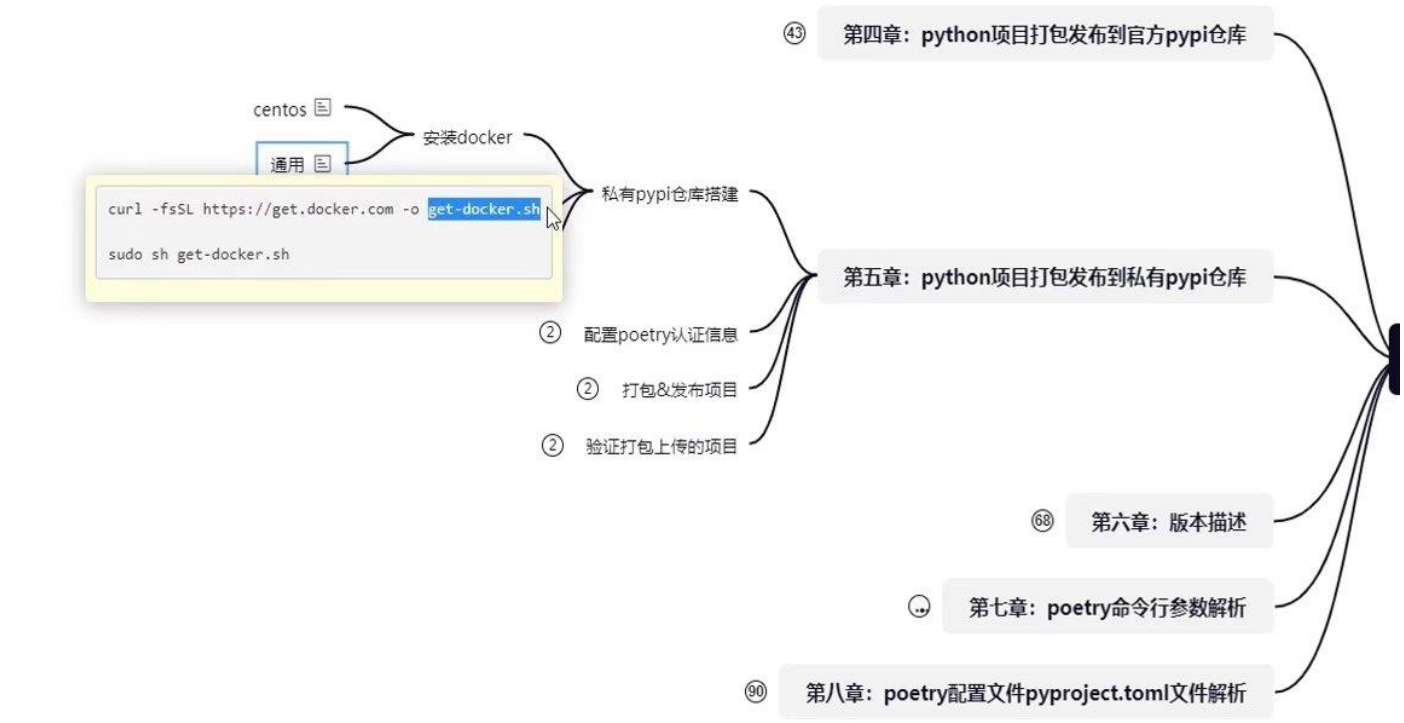

CHAR_PIC
char_pic
dist
char_pic-0.1.0-py3-none-any.whl
char_pic-0.1.0.tar.gz
tests
poetry.lock
pyproject.toml
README.rst
pyproject.toml
3 version = "0.1.0"
4 description = ""
5 authors = ["xiaodeng <xiaodengteacher@qq.com>"]
6
7 [tool.poetry.dependencies]
8 python = "^3.8"
9 cowpy = "^1.1.0"
10
11 [tool.poetry.dev-dependencies]
12 pytest = "^5.2"
13
14 [build-system]
15 requires = ["poetry-core>=1.0.0"]
16 build-backend = "poetry.core.masonry.api"
PS D:\PycharmProjects\char_pic> poetry config http-basic.pypi teacherxiaodeng
Password:
PS D:\PycharmProjects\char_pic> poetry build
Building char_pic (0.1.0)
- Building sdist
- Built char_pic-0.1.0.tar.gz
- Building wheel
- Built char_pic-0.1.0-py3-none-any.whl
PS D:\PycharmProjects\char_pic>

问题 输出 终端 调试控制台

```
  - Building wheel
  - Built char_pic-0.1.0-py3-none-any.whl
PS D:\PycharmProjects\char_pic> poetry publish

Publishing char_pic (0.1.0) to PyPI
 - Uploading char_pic-0.1.0-py3-none-any.whl 0%
 - Uploading char_pic-0.1.0-py3-none-any.whl 100%
 - Uploading char_pic-0.1.0-py3-none-any.whl 100%

  UploadError
```

1 poetry public --build




43 第四章：python项目打包发布到官方pypi仓库
centos
安装docker
通用
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
私有pypi仓库搭建
第五章：python项目打包发布到私有pypi仓库
② 配置poetry认证信息
② 打包&发布项目
② 验证打包上传的项目
68 第六章：版本描述
第七章：poetry命令行参数解析
90 第八章：poetry配置文件pyproject.toml文件解析

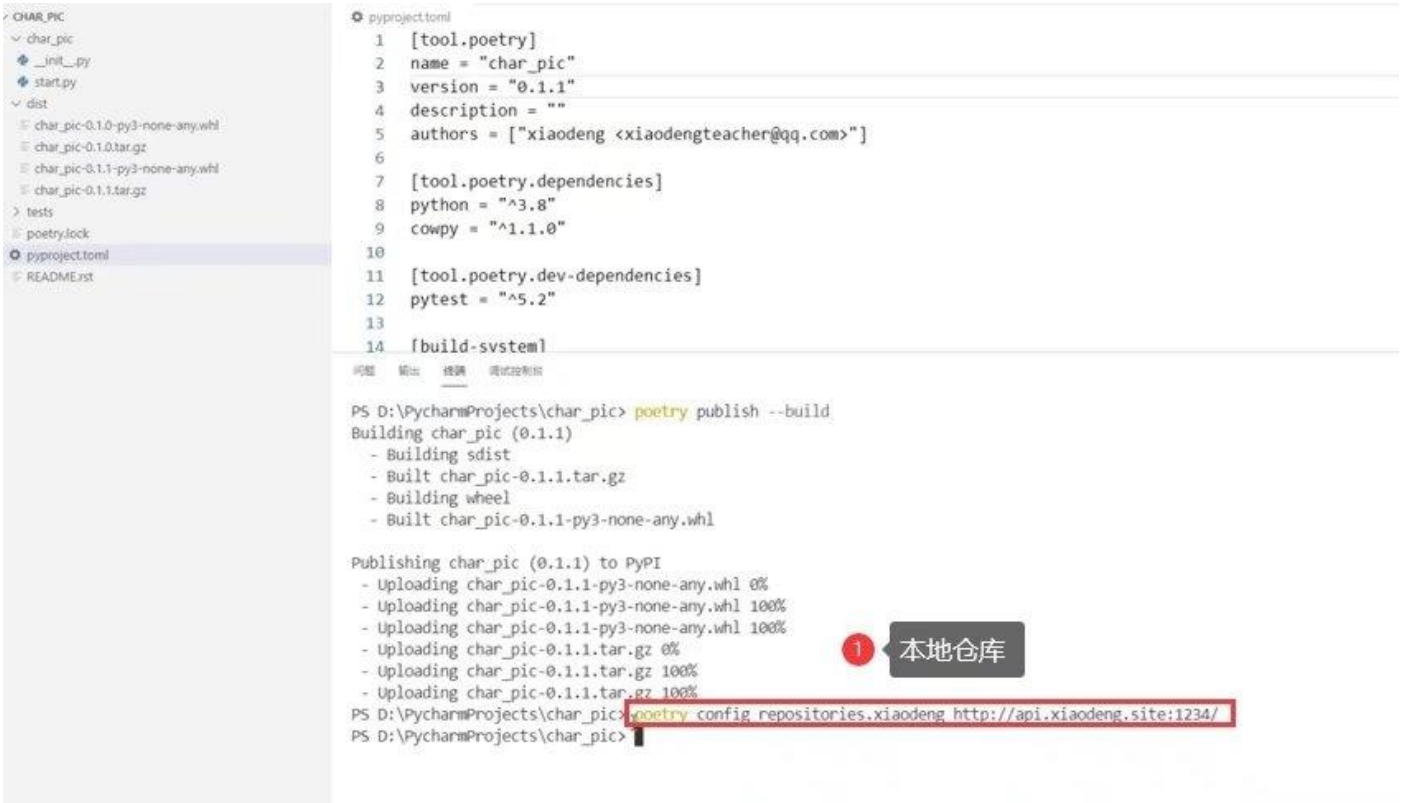

43 第四章：python项目打包发布到官方pypi仓库
centos
安装docker
通用
安装docker-compose
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose
-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
私有pypi仓库搭建
python项目打包发布到私有pypi仓库
② 打包&发布项目
② 验证打包上传的项目
68 第六章：版本描述
第七章：poetry命令行参数解析
90 第八章：poetry配置文件pyproject.toml文件解析

```
[root@instance-k0fsji48 pypiserver]# ls
auth  docker-compose.yml  README.md
[root@instance-k0fsji48 pypiserver]# ls
auth  docker-compose.yml  README.md
[root@instance-k0fsji48 pypiserver]# cat README.md
## 部署pypi server

### 使用docker-compose部署认证版本

- docker-compose up -d


### 参考文档

- https://github.com/pypiserver/pypiserver#quickstart-installation-and-usage
- https://hostingcanada.org/htpasswd-generator/
- https://github.com/pypiserver/pypiserver/blob/master/docker-compose.yml
[root@instance-k0fsji48 pypiserver]#
```

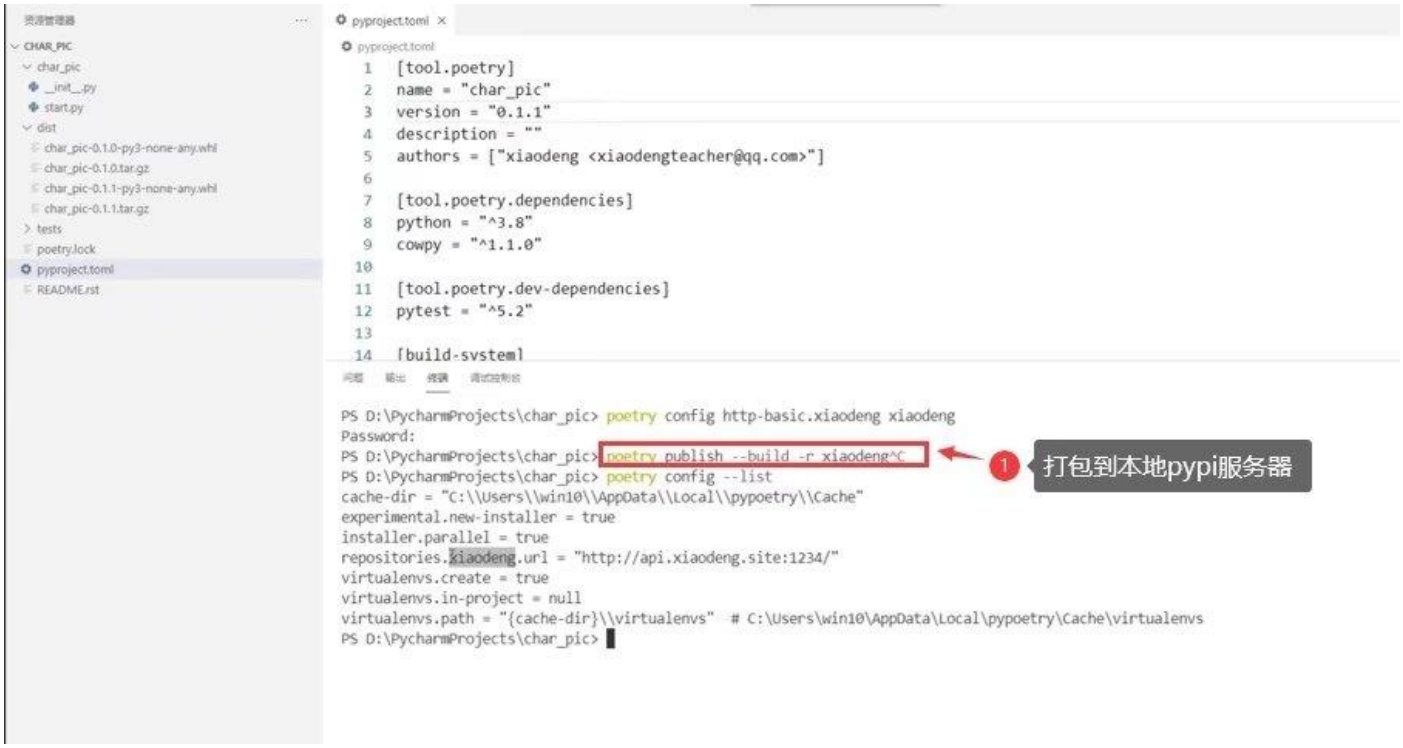
```
/github.com/pypiserver/pypiserver#quickstart-installation-and-usage
/hostingcanada.org/htpasswd-generator/
/github.com/pypiserver/pypiserver/blob/master/docker-compose.yml
tance-k0fsji48 pypiserver]# cat docker-compose.yml
"3.3"


erver-authenticated:
mage: pypiserver/pypiserver:latest
olumes:
    - type: bind
      source: ./auth
      target: /data/auth
    - type: volume
      source: pypi_packages
      target: /data/packages
ommand: -P /data/auth/.htpasswd -a update,download,list /data/packages
orts:
    - "1234:8080"


packages:

tance-k0fsji48 pypiserver]# cat
          docker-compose.yml  README.md
tance-k0fsji48 pypiserver]# cat auth/.htpasswd
$apr1$2c4fsv10$Zu6pyf1Ej5jyeCwNthSt1.
tance-k0fsji48 pypiserver]#
```

1 用户注册信息


43 第四章：python项目打包发布到官方pypi仓库
5 私有pypi仓库搭建
添加repository
poetry config repositories.xiaodeng http://api.xiaodeng.site:1234/
poetry config repositories.xiaodeng
第五章：python项目打包发布到私有pypi仓库
2 打包&发布项目
2 验证打包上传的项目
68 第六章：版本描述
第七章：poetry命令行参数解析
90 第八章：poetry配置文件pyproject.toml文件解析


CHAR_PIC
char_pic
__init__.py
start.py
dist
char_pic-0.1.0-py3-none-any.whl
char_pic-0.1.0.tar.gz
char_pic-0.1.1-py3-none-any.whl
char_pic-0.1.1.tar.gz
tests
poetry.lock
pyproject.toml
README.rst
pyproject.toml
[tool.poetry]
name = "char_pic"
version = "0.1.1"
description = ""
authors = ["xiaodeng <xiaodengteacher@qq.com>"]
[tool.poetry.dependencies]
python = "^3.8"
cowpy = "^1.1.0"
[tool.poetry.dev-dependencies]
pytest = "^5.2"
[build-system]
PS D:\PycharmProjects\char_pic> poetry publish --build
Building char_pic (0.1.1)
- Building sdist
- Built char_pic-0.1.1.tar.gz
- Building wheel
- Built char_pic-0.1.1-py3-none-any.whl
Publishing char_pic (0.1.1) to PyPI
- Uploading char_pic-0.1.1-py3-none-any.whl 0%
- Uploading char_pic-0.1.1-py3-none-any.whl 100%
- Uploading char_pic-0.1.1-py3-none-any.whl 100%
- Uploading char_pic-0.1.1.tar.gz 0%
- Uploading char_pic-0.1.1.tar.gz 100%
- Uploading char_pic-0.1.1.tar.gz 100%
1 本地仓库
PS D:\PycharmProjects\char_pic> poetry config repositories.xiaodeng http://api.xiaodeng.site:1234/
PS D:\PycharmProjects\char_pic>


CHAR_PIC
char_pic
__init__.py
start.py
dist
char_pic-0.1.0-py3-none-any.whl
char_pic-0.1.0.tar.gz
char_pic-0.1.1-py3-none-any.whl
char_pic-0.1.1.tar.gz
tests
poetry.lock
pyproject.toml
README.rst
pyproject.toml
[tool.poetry]
name = "char_pic"
version = "0.1.1"
description = ""
authors = ["xiaodeng <xiaodengteacher@qq.com>"]
[tool.poetry.dependencies]
python = "^3.8"
cowpy = "^1.1.0"
[tool.poetry.dev-dependencies]
pytest = "^5.2"
[build-system]
PS D:\PycharmProjects\char_pic> poetry config http-basic.xiaodeng xiaodeng
Password:
1 登录本地pypi服务器


CHAR_PIC
char_pic
__init__.py
start.py
dist
char_pic-0.1.0-py3-none-any.whl
char_pic-0.1.0.tar.gz
char_pic-0.1.1-py3-none-any.whl
char_pic-0.1.1.tar.gz
tests
poetry.lock
pyproject.toml
README.rst
pyproject.toml
[tool.poetry]
name = "char_pic"
version = "0.1.1"
description = ""
authors = ["xiaodeng <xiaodengteacher@qq.com>"]
[tool.poetry.dependencies]
python = "^3.8"
cowpy = "^1.1.0"
[tool.poetry.dev-dependencies]
pytest = "^5.2"
[build-system]
PS D:\PycharmProjects\char_pic> poetry config http-basic.xiaodeng xiaodeng
Password:
PS D:\PycharmProjects\char_pic> poetry publish --build -r xiaodeng^C
PS D:\PycharmProjects\char_pic> poetry config --list
cache-dir = "C:\\Users\\win10\\AppData\\Local\\pypoetry\\Cache"
experimental.new-installer = true
installer.parallel = true
repositories.xiaodeng.url = "http://api.xiaodeng.site:1234/"
virtualenvs.create = true
virtualenvs.in-project = null
virtualenvs.path = "{cache-dir}\\virtualenvs" # C:\Users\win10\AppData\Local\pypoetry\Cache\virtualenvs
PS D:\PycharmProjects\char_pic>
1 打包到本地pypi服务器


不安全 | api.xiaodeng.site:1234
Welcome to pypiserver!
This is a PyPI compatible package index serving 2 packages.
To use this server with pip, run the following command:
pip install --index-url http://api.xiaodeng.site:1234/simple/ PACKAGE [PACKAGE2...]
To use this server with easy_install, run the following command:
easy_install --index-url http://api.xiaodeng.site:1234/simple/ PACKAGE [PACKAGE2...]
The complete list of all packages can be found here or via the simple index.
This instance is running version 1.4.2 of the pypiserver software.


TWO_PROJECT
poetry.lock
pyproject.toml
pyproject.toml
[tool.poetry.dev-dependencies]
[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"
[[tool.poetry.source]]
name = "xiaodeng"
url = "http://api.xiaodeng.site:1234/simple/"
PS D:\PycharmProjects\two_project> poetry add char-pic
Using version ^0.1.1 for char-pic
Updating dependencies
Resolving dependencies...
1
从本地Pypi服务器下载安装

pyproject.toml

```
[tool.poetry.dev-dependencies]

[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"

[[tool.poetry.source]]
name = "xiaodeng"
url = "http://api.xiaodeng.site:1234/simple/"
```

```
PS D:\PycharmProjects\two_project> pip install -i http://api.xiaodeng.site:1234/simple/ char-pic
Looking in indexes: http://api.xiaodeng.site:1234/simple/
WARNING: The repository located at api.xiaodeng.site is not a trusted or secure host and is being ignored. If this repository is availab
 use HTTPS instead, otherwise you may silence this warning and allow it anyway with '--trusted-host api.xiaodeng.site'.
ERROR: Could not find a version that satisfies the requirement char-pic (from versions: none)
ERROR: No matching distribution found for char-pic
WARNING: The repository located at api.xiaodeng.site is not a trusted or secure host and is being ignored. If this repository is availab
 use HTTPS instead, otherwise you may silence this warning and allow it anyway with '--trusted-host api.xiaodeng.site'.
PS D:\PycharmProject 关注链接 (Ctrl + 单击) ip install -i http://api.xiaodeng.site:1234/simple/ char-pic --trusted-host api.xiaodeng.site
Looking in indexes: http://api.xiaodeng.site:1234/simple/
User for api.xiaodeng.site:1234:
```

pyproject.toml

```
[tool.poetry.dependencies]
python = "^3.8"
cowpy = "^1.1.0"
requests = "^1"

[tool.poetry.dev-dependencies]
pytest = "^5.2"

[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"

[[tool.poetry.source]]
name = "douban"
```

requests (2.26.0)

```
PS D:\PycharmProjects\char_pic> poetry update
Updating dependencies
Resolving dependencies...

Writing lock file

Package operations: 0 installs, 1 update, 4 removals

  • Removing certifi (2021.5.30)
  • Removing charset-normalizer (2.0.4)
  • Removing idna (3.2)
```

1 不能超过已固定的版本号


poetry new 项目路径
会以项目路径作为包名称
poetry new 项目路径 --name 包名称
poetry new char_pic_one --name xxx
包名称为小写
创建打包项目工程
③ 项目开发
⑪ 注册python官方pypi账号
② 配置poetry认证信息
⑧ 打包
⑦ 发布项目
验证打包上传的项目
第四章：python项目打包发布到官方pypi仓库
㉔ 第五章：python项目打包发布到私有pypi仓库
⑲ 第六章：版本描述
第七章：poetry命令行参数解析
⑩ 第八章：poetry配置文件pyproject.toml文件解析

pyproject.toml

```
[tool.poetry]
name = "char_pic"
version = "0.1.1"
```

```
USAGE
  poetry install [--no-dev] [--no-root] [--dry-run] [--remove-untracked] [-E <...>]

OPTIONS
  --no-dev              Do not install the development dependencies.
  --no-root             Do not install the root package (the current project).
PS D:\PycharmProjects\char_pic> poetry install --remove-untracked --ansi -vvv
Using virtualenv: C:\Users\win10\AppData\Local\pypoetry\Cache\virtualenvs\char-pic-LUmnN3Xf-py3.8
Installing dependencies from lock file

Finding the necessary packages for the current system

Package operations: 0 installs, 0 updates, 2 removals, 16 skipped

  • Removing et-xmlfile (1.1.0): Pending...
  • Removing et-xmlfile (1.1.0): Removing...
  • Removing et-xmlfile (1.1.0)
  • Removing openpyxl (3.0.7): Pending...
  • Removing openpyxl (3.0.7): Removing...
  • Removing openpyxl (3.0.7)
  • Installing pyparsing (2.4.7): Pending...
  • Installing pyparsing (2.4.7): Skipped for the following reason: Already installed
  • Installing atomicwrites (1.4.0): Pending...
  • Installing atomicwrites (1.4.0): Skipped for the following reason: Already installed
  • Installing attrs (21.2.0): Pending...
  • Installing attrs (21.2.0): Skipped for the following reason: Already installed
  • Installing certifi (2021.5.30): Pending...
  • Installing certifi (2021.5.30): Skipped for the following reason: Already installed
  • Installing charset-normalizer (2.0.4): Pending...
  • Installing charset-normalizer (2.0.4): Skipped for the following reason: Already installed
  • Installing colorama (0.4.4): Pending...
  • Installing colorama (0.4.4): Skipped for the following reason: Already installed
  • Installing idna (3.2): Pending...
  • Installing idna (3.2): Skipped for the following reason: Already installed
  • Installing more-itertools (8.8.0): Pending...
  • Installing more-itertools (8.8.0): Skipped for the following reason: Already installed
```

```
PS D:\PycharmProjects\char_pic> poetry config
PS D:\PycharmProjects\char_pic> poetry config  --list
cache-dir = "C:\\Users\\win10\\AppData\\Local\\pypoetry\\Cache"
experimental.new-installer = true
installer.parallel = true
repositories.xiaodeng.url = "http://api.xiaodeng.site:1234/"
virtualenvs.create = true
virtualenvs.in-project = null
virtualenvs.path = "{cache-dir}\\virtualenvs"  # C:\Users\win10\AppData\Local\pypoetry\Cache\virtualenvs
PS D:\PycharmProjects\char_pic> ls C:\\Users\\win10\\AppData\\Local\\pypoetry\\Cache


    目录: C:\Users\win10\AppData\Local\pypoetry\Cache


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        2021/8/24     20:21                artifacts
d-----        2021/8/14     16:42                cache
d-----        2021/8/24     18:34                virtualenvs


PS D:\PycharmProjects\char_pic> ls C:\\Users\\win10\\AppData\\Local\\pypoetry\\Cache\\cache
```

修改虚拟环境路径

接下来，可以按照自己的文件存放习惯，修改缓存目录，同时也修改了虚拟环境目录：

```
1  poetry config cache-dir E:\Documents\Library
```

```
E:\Documents\PycharmProjects\Test>poetry config --list
cache-dir = "E:\\Documents\\Library"
virtualenvs.create = true
virtualenvs.in-project = false
virtualenvs.path = "{cache-dir}\\virtualenvs"  # E:\Documents\Library\virtualenvs
```

```
`poetry --version`


D:\CODE\PYTHON\Project05>poetry -V
Poetry version 1.1.8

D:\CODE\PYTHON\Project05>poetry config --list
cache-dir = "C:\\Users\\Administrator\\AppData\\Local\\pypoetry\\Cache"
experimental.new-installer = true
installer.parallel = true
virtualenvs.create = true
virtualenvs.in-project = null
virtualenvs.path = "{cache-dir}\\virtualenvs"  # C:\Users\Administrator\AppData\Local\pypoetry\Cache\virtualenvs

D:\CODE\PYTHON\Project05>
```

```
PS D:\PycharmProjects\char_pic> poetry config installer.parallel false
PS D:\PycharmProjects\char_pic> poetry config
PS D:\PycharmProjects\char_pic> poetry config --list
cache-dir = "C:\\Users\\win10\\AppData\\Local\\pypoetry\\Cache"
experimental.new-installer = true
installer.parallel = false
repositories.xiaodeng.url = "http://api.xiaodeng.site:1234/"
virtualenvs.create = true
virtualenvs.in-project = null
virtualenvs.path = "{cache-dir}\\virtualenvs"  # C:\Users\win10\AppData\Local\pypoetry\Cache\virtualenvs
PS D:\PycharmProjects\char_pic> poetry config installer.parallel true
PS D:\PycharmProjects\char_pic> poetry config --list
cache-dir = "C:\\Users\\win10\\AppData\\Local\\pypoetry\\Cache"
experimental.new-installer = true
installer.parallel = true
repositories.xiaodeng.url = "http://api.xiaodeng.site:1234/"
virtualenvs.create = true
virtualenvs.in-project = null
virtualenvs.path = "{cache-dir}\\virtualenvs"  # C:\Users\win10\AppData\Local\pypoetry\Cache\virtualenvs
PS D:\PycharmProjects\char_pic> poetry config repositories.douban https://pypi.doubanio.com/
PS D:\PycharmProjects\char_pic> poetry config --list
cache-dir = "C:\\Users\\win10\\AppData\\Local\\pypoetry\\Cache"
experimental.new-installer = true
installer.parallel = true
repositories.douban.url = "https://pypi.doubanio.com/"
repositories.xiaodeng.url = "http://api.xiaodeng.site:1234/"
virtualenvs.create = true
virtualenvs.in-project = null
virtualenvs.path = "{cache-dir}\\virtualenvs"  # C:\Users\win10\AppData\Local\pypoetry\Cache\virtualenvs
PS D:\PycharmProjects\char_pic>
```

```
PS D:\PycharmProjects\char_pic_one> poetry config virtualenvs.in-project true --local
PS D:\PycharmProjects\char_pic_one> poetry config
PS D:\PycharmProjects\char_pic_one> poetry config --local
PS D:\PycharmProjects\char_pic_one>
```

1 --local 仅用于当前项目配置

```
PS D:\PycharmProjects\char_pic_one> poetry config virtualenvs.in-project true --local
PS D:\PycharmProjects\char_pic_one> poetry config
PS D:\PycharmProjects\char_pic_one> poetry config --local
PS D:\PycharmProjects\char_pic_one> poetry config --list
cache-dir = "C:\\Users\\win10\\AppData\\Local\\pypoetry\\Cache"
experimental.new-installer = true
installer.parallel = true
repositories.douban.url = "https://pypi.doubanio.com/"
repositories.xiaodeng.url = "http://api.xiaodeng.site:1234/"
virtualenvs.create = true
virtualenvs.in-project = true
virtualenvs.path = "{cache-dir}\\virtualenvs"  # C:\Users\win10\AppData\Local\pypoetry\Cache\virtualenvs
PS D:\PycharmProjects\char_pic_one> poetry config repositories.douban --unset
PS D:\PycharmProjects\char_pic_one> poetry config --list
cache-dir = "C:\\Users\\win10\\AppData\\Local\\pypoetry\\Cache"
experimental.new-installer = true
installer.parallel = true
repositories.xiaodeng.url = "http://api.xiaodeng.site:1234/"
virtualenvs.create = true
virtualenvs.in-project = true
virtualenvs.path = "{cache-dir}\\virtualenvs"  # C:\Users\win10\AppData\Local\pypoetry\Cache\virtualenvs
PS D:\PycharmProjects\char_pic_one>
```

1 删除配置

pyproject.toml

```toml
[tool.poetry]
name = "char_pic"
version = "0.1.3"
description = "generate char pic demo."
authors = ["xiaodeng <xiaodengteacher@qq.com>", "xiaodeng1 <xiaodengteacher1@qq.com>"]

license = "Apache-2.0"
#license = "Propriertary"

maintainer = ["xiaodeng2 <xiaodengteacher2@qq.com>", "xiaodeng3 <xiaodengteacher3@qq.com>", "xiaodeng4 <xiaodengteach

readme = "README.rst"

homepage = "http://api.xiaodeng.site:8888/"
repository = "https://github.com/psf/requests"
documentation = "http://api.xiaodeng.site:8888/"
keywords = ["char", "pic", "xioadeng"]
classifiers = ["Programming Language :: Python :: 3", "Topic :: Communications", "Development Status :: 3 - Alpha", "

[tool.poetry.dependencies]
python = "^3.8"
cowpy = "^1.1.0"
requests = "^2.26.0"

[tool.poetry.dev-dependencies]

[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"
```

pyproject.toml

```toml
authors = ["xiaodeng <xiaodengteacher@qq.com>", "xiaodeng1 <xiaodengteacher1@qq.com>"]

license = "Apache-2.0"
#license = "Propriertary"

maintainers = ["xiaodeng2 <xiaodengteacher2@qq.com>", "xiaodeng3 <xiaodengteacher3@qq.com>", "xiaodeng4 <xiaodengteacher4@qq.com>"]

readme = "README.rst"

homepage = "http://api.xiaodeng.site:8888/"
repository = "https://github.com/psf/requests"
documentation = "http://api.xiaodeng.site:8888/"
keywords = ["char", "pic", "xioadeng"]
classifiers = ["Programming Language :: Python :: 3", "Topic :: Communications", "Development Status :: 3 - Alpha", "License :: Freeware"]
packages = [{ include = "char_pic", format = "sdist" }]
include = ["README.md", "requirements.txt"]
exclude = ["1.txt"]

[tool.poetry.dependencies]
python = "^3.8"
```

```
S D:\PycharmProjects\char_pic> poetry publish --build
```

pyproject.toml × README.rst README.md

pyproject.toml

```
22
23  [tool.poetry.dependencies]
24  python = "^3.8"
25  cowpy = "^1.1.0"
26  requests = "^2.26.0"
27
28  mysqlclient = { version = "^2.0", optional = true }
29
30  [tool.poetry.dev-dependencies]
31  PyYAML = "^5.4"
32
33  [tool.poetry.extras]
34  mysql = ["mysqlclient"]
35
36  [build-system]
37  requires = ["poetry-core>=1.0.0"]
38  build-backend = "poetry.core.masonry.api"
39
40  [[tool.poetry.source]]
41  name = "douban"
```

1 可选包

问题 输出 终端 调试控制台

```
PS D:\PycharmProjects\char_pic> poetry show --all
certifi            2021.5.30 Python package for providing Mozilla's CA Bundle.
charset-normalizer 2.0.4     The Real First Universal Charset Detector. Open, modern and actively maintained alternative to Chardet.
cowpy              1.1.0     A cowsay clone for python in one file.
idna               3.2       Internationalized Domain Names in Applications (IDNA)
mysqlclient    (!) 2.0.3     Python interface to MySQL
pyyaml             5.4.1     YAML parser and emitter for Python
requests           2.26.0    Python HTTP for Humans.
urllib3            1.26.6    HTTP library with thread-safe connection pooling, file post, and more.
PS D:\PycharmProjects\char_pic>
```

pyproject.toml

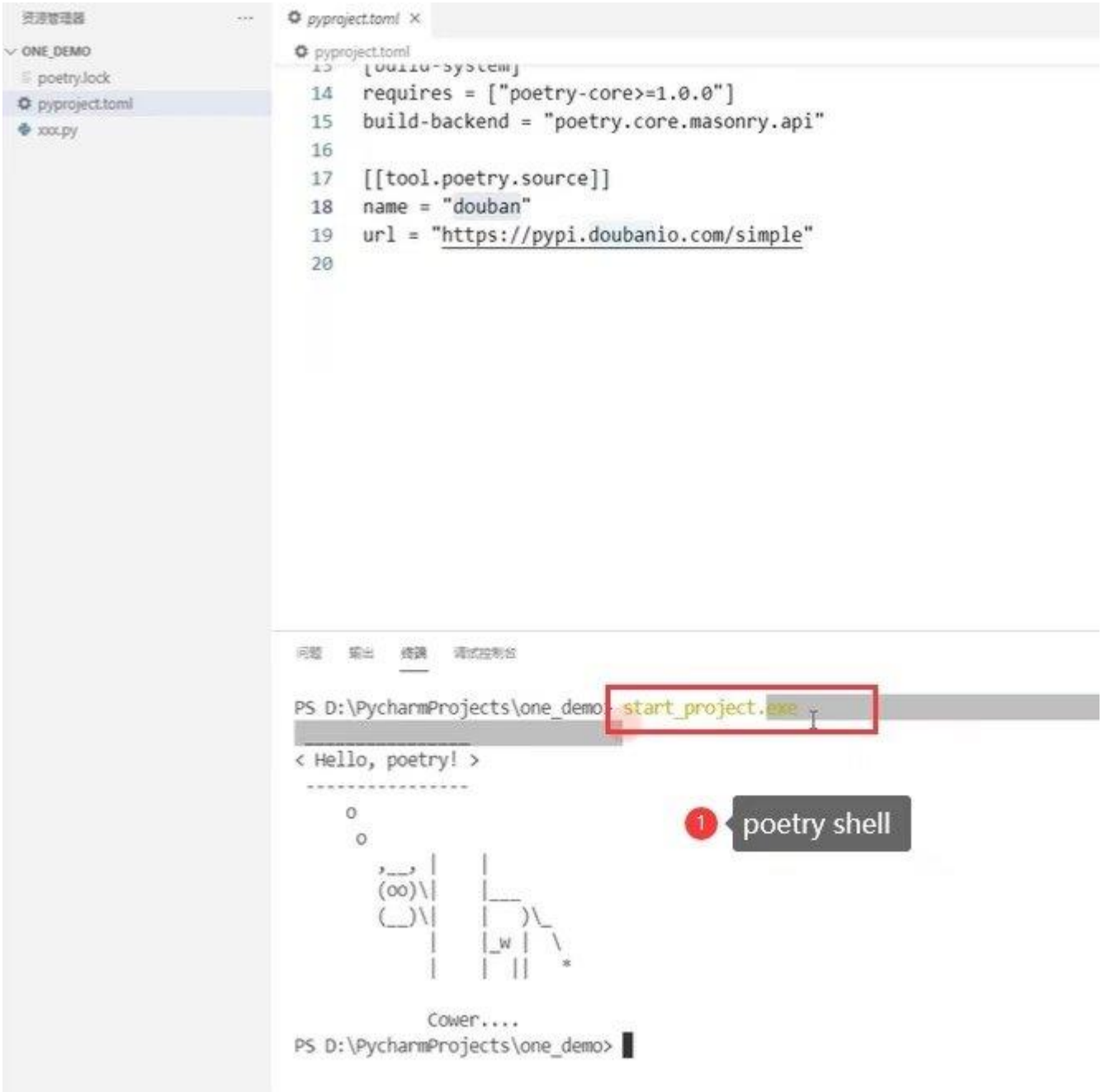
```
mysqlclient = { version = "^2.0", optional = true }

[tool.poetry.dev-dependencies]
PyYAML = "^5.4"

[tool.poetry.extras]
mysql = ["mysqlclient"]

[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"

[[tool.poetry.source]]
name = "douban"
url = "https://pypi.doubanio.com/simple"
```

```
PS D:\PycharmProjects\char_pic> poetry install -E mysql
Installing dependencies from lock file

Package operations: 1 install, 0 updates, 0 removals

  • Installing mysqlclient (2.0.3)

Installing the current project: char_pic (0.1.3)
PS D:\PycharmProjects\char_pic>
```

1 poetry update

pyproject.toml

```
[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"

[[tool.poetry.source]]
name = "douban"
url = "https://pypi.doubanio.com/simple"

[tool.poetry.urls]
hhh = "http://api.xiaodeng.site:8888/"


[tool.poetry.scripts]
# 可执行程序名称 = "包名.模块名:函数名"
start_project = "char_pic.start:run"

```


ONE_DEMO
poetry.lock
pyproject.toml
xxx.py
pyproject.toml
14 requires = ["poetry-core>=1.0.0"]
15 build-backend = "poetry.core.masonry.api"
16
17 [[tool.poetry.source]]
18 name = "douban"
19 url = "https://pypi.doubanio.com/simple"
20
PS D:\PycharmProjects\one_demo> start_project.exe
< Hello, poetry! >
1 poetry shell
Cower....
PS D:\PycharmProjects\one_demo>

//----------

pip 安装缓慢

在 C:\Users\Administrator\pip 目录中添加 pip.ini 文件或通过-i 参数设定国内安装源[mirrors.aliyun.com，清华或豆瓣]

pip.ini 内容如下：

[global]

index-url =https://pypi.tuna.tsinghua.edu.cn/simple

[install]

trusted-host=pypi.tuna.tsinghua.edu.cn

或者

[global]

index-url =https://pypi.douban.com/simple

[install]

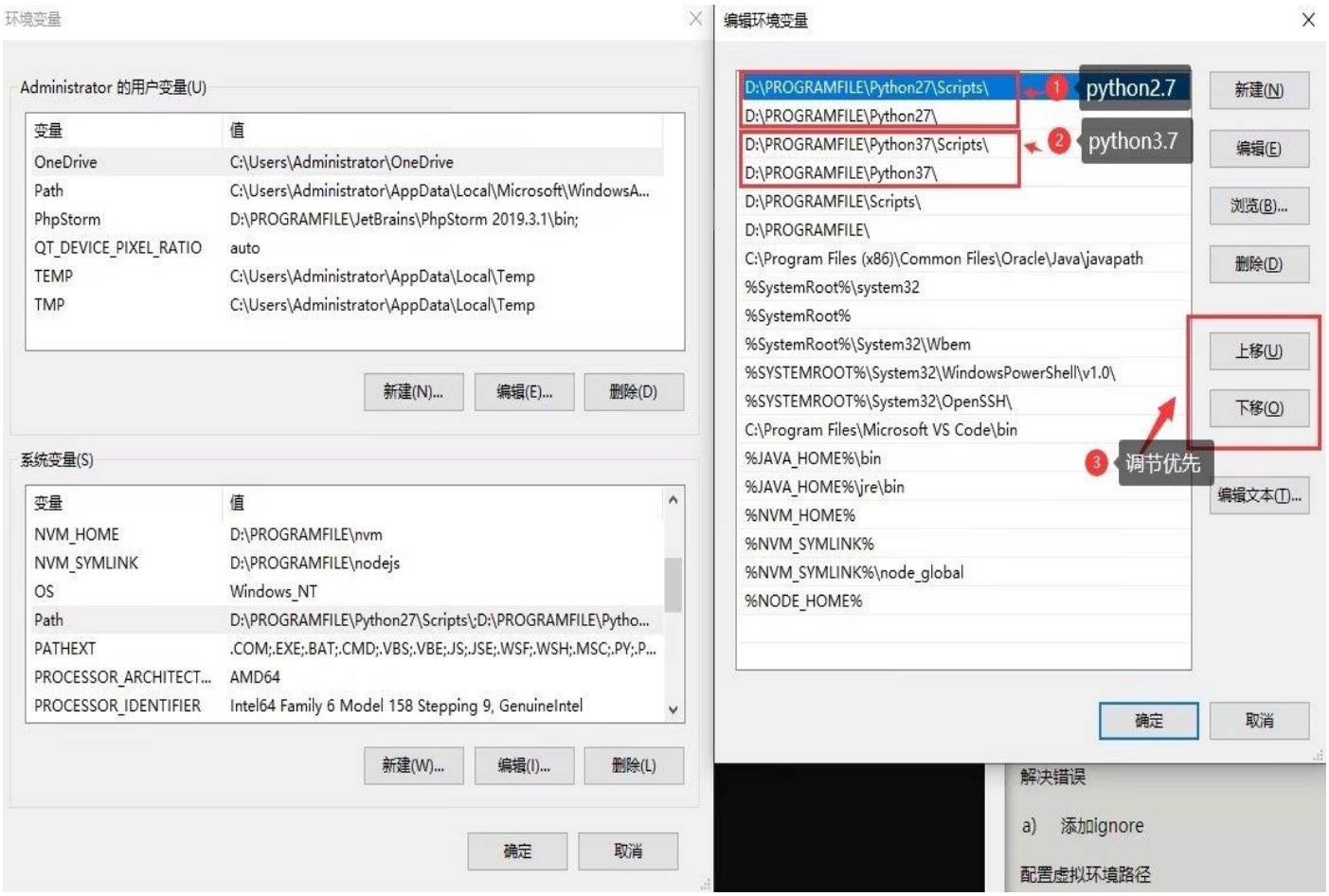
trusted-host=pypi.douban.com

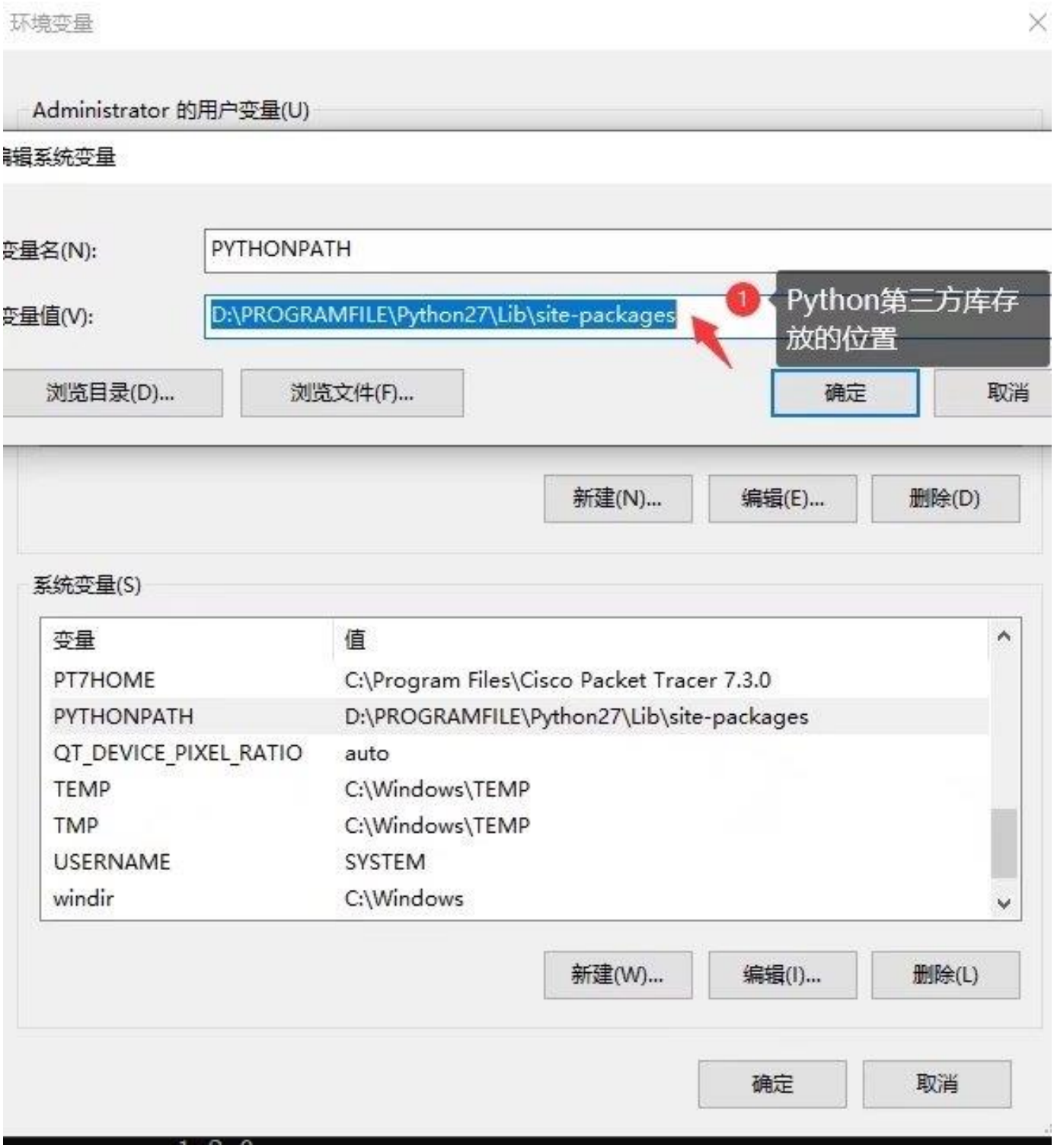
Linux在当前用户家目录.pip/pip.conf

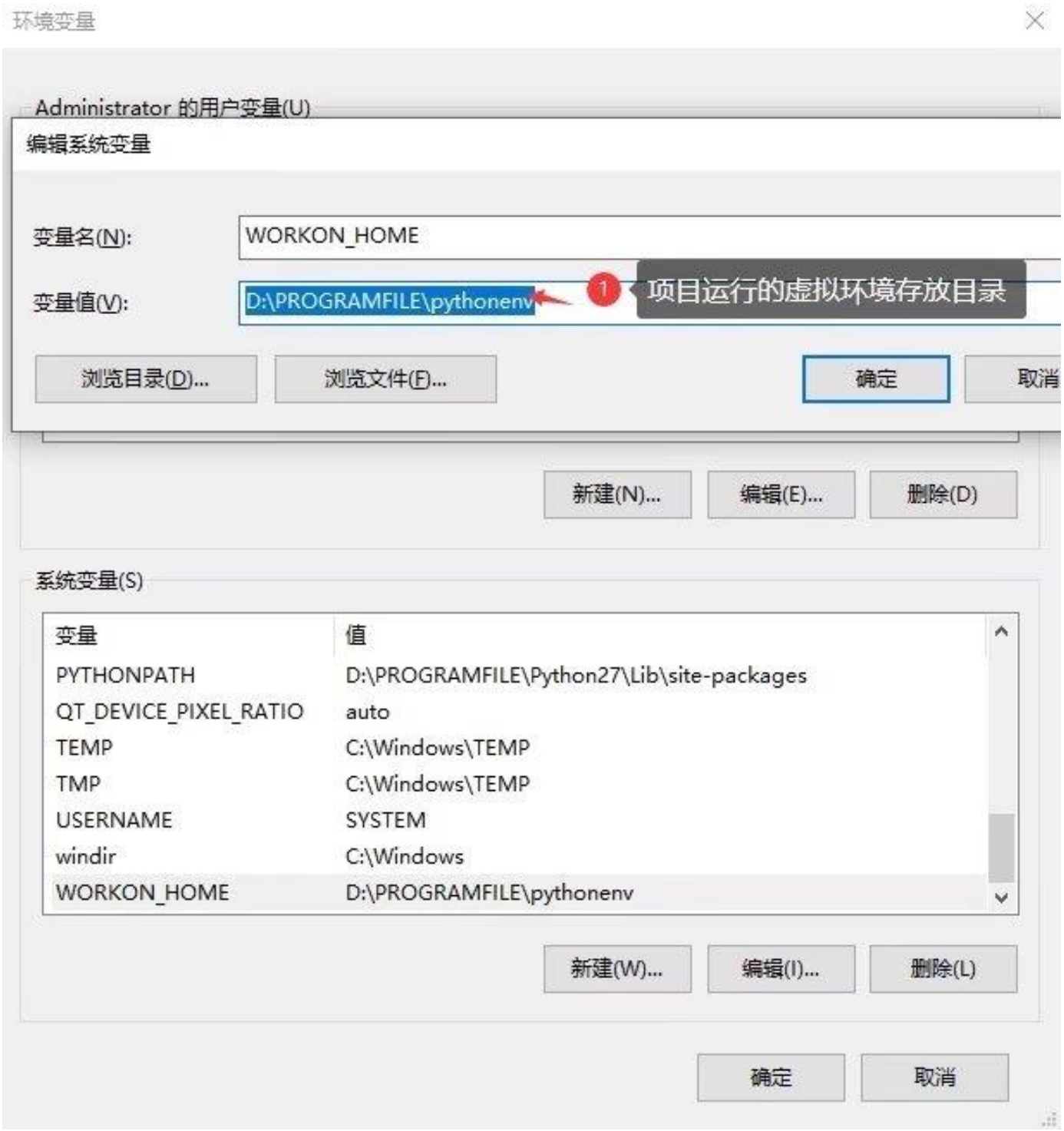
//----------

Python 第三方库路径

变量 PYTHONPATH 路径指向第三方库安装路径 C:\Python\Program\Lib\site-packages


环境变量
Administrator 的用户变量(U)
变量
值
OneDrive
C:\Users\Administrator\OneDrive
Path
C:\Users\Administrator\AppData\Local\Microsoft\WindowsA...
PhpStorm
D:\PROGRAMFILE\JetBrains\PhpStorm 2019.3.1\bin;
QT_DEVICE_PIXEL_RATIO
auto
TEMP
C:\Users\Administrator\AppData\Local\Temp
TMP
C:\Users\Administrator\AppData\Local\Temp
新建(N)...
编辑(E)...
删除(D)
系统变量(S)
变量
值
NVM_HOME
D:\PROGRAMFILE\nvm
NVM_SYMLINK
D:\PROGRAMFILE\nodejs
OS
Windows_NT
Path
D:\PROGRAMFILE\Python27\Scripts\;D:\PROGRAMFILE\Pytho...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.P...
PROCESSOR_ARCHITECT...
AMD64
PROCESSOR_IDENTIFIER
Intel64 Family 6 Model 158 Stepping 9, GenuineIntel
新建(W)...
编辑(I)...
删除(L)
确定
取消
编辑环境变量
D:\PROGRAMFILE\Python27\Scripts\
D:\PROGRAMFILE\Python27\
D:\PROGRAMFILE\Python37\Scripts\
D:\PROGRAMFILE\Python37\
D:\PROGRAMFILE\Scripts\
D:\PROGRAMFILE\
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Microsoft VS Code\bin
%JAVA_HOME%\bin
%JAVA_HOME%\jre\bin
%NVM_HOME%
%NVM_SYMLINK%
%NVM_SYMLINK%\node_global
%NODE_HOME%
1
python2.7
2
python3.7
3
调节优先
新建(N)
编辑(E)
浏览(B)...
删除(D)
上移(U)
下移(O)
编辑文本(T)...
确定
取消
解决错误
a) 添加ignore
配置虚拟环境路径


环境变量
Administrator 的用户变量(U)
编辑系统变量
变量名(N):
PYTHONPATH
变量值(V):
D:\PROGRAMFILE\Python27\Lib\site-packages
1
Python第三方库存放的位置
浏览目录(D)...
浏览文件(F)...
确定
取消
新建(N)...
编辑(E)...
删除(D)
系统变量(S)
变量
值
PT7HOME
C:\Program Files\Cisco Packet Tracer 7.3.0
PYTHONPATH
D:\PROGRAMFILE\Python27\Lib\site-packages
QT_DEVICE_PIXEL_RATIO
auto
TEMP
C:\Windows\TEMP
TMP
C:\Windows\TEMP
USERNAME
SYSTEM
windir
C:\Windows
新建(W)...
编辑(I)...
删除(L)
确定
取消



导出安装的第三方库到 c:\1.txt

d:\>pip freeze >"c:\1.txt"

静默删除第三方库

d:\>pip uninstall -r -y "c:\1.txt"

Linux 下安装，激活，离开虚拟环境

yum install python-virtualenv

virutalenv env

bin$source activate
deactivate

Python 的 Scripts 文件夹下没有 pip.exe

解决办法：

Windows 命令行，输入：python -m ensurepip，即可生成 pip3.exe

将 scripts 文件夹中的 pip3.exe 修改为 pip.exe

然后添加 Scripts 路径到环境变量 path 中即可。