

xuanhun / PythonHackingBook1

[Code](#)[Issues 11](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)[Insights](#)[master](#)

...

PythonHackingBook1 / 1.1 python简介和环境搭建 /



yangwenhai@xueleyun.com

...

on 7 Jan 2019



..

[code](#)

2 years ago

[img](#)

2 years ago

[Readme.md](#)

2 years ago

[Readme.md](#)

Python简介和环境搭建

于20世纪80年代末，Guido van Rossum发明了Python，初衷据说是为打发圣诞节的无趣。1991年首次发布，是ABC语言的继承，同时也是一种脚本语言。取名时，Guido van Rossum认为它应该“短小，独特，还有一点神秘感”，他是英国著名剧团Monty Python的忠实粉丝，所以就取名Python了。



图1 Monty Python剧团

牛人的世界我们无法企及，随便玩玩就玩出门流行的语言来。

编程语言众多，Python按照分类来讲，首先是动态语言，无需编译，然后是脚本语言。当然脚本语言这个特性在逐渐淡化，Python可以在Web、桌面、后台服务各种应用类型中占有一席之地。

◆ Python从◆创建之后，到2000年发布了2.0版本，到2008年发布了3.0版本。到目前为止，2.*版本仍然是使用最广泛的版本，有一部分原因要归咎于3.0版本对2.*版本的不兼容。本系列教程会尽量使用最新的3.*版本来开发示例。

Python本身是开源的，我们可以从<https://www.python.org/downloads/source/> 下载Python从2.0.1开始到当前最新版（3.7.0）各个版本的源码。

The screenshot shows a web browser window with the URL <https://www.python.org/downloads/source/>. The page title is "Python Source Releases". Below the title, there is a list of Python releases with download links for XZ compressed source tarballs and Gzipped source tarballs.

- [Latest Python 3 Release - Python 3.7.0](#)
- [Latest Python 2 Release - Python 2.7.15](#)
- [Python 3.5.6 - 2018-08-02](#)
 - Download [XZ compressed source tarball](#)
 - Download [Gzipped source tarball](#)
- [Python 3.4.9 - 2018-08-02](#)
 - Download [XZ compressed source tarball](#)
 - Download [Gzipped source tarball](#)
- [Python 3.4.9rc1 - 2018-07-20](#)
 - Download [XZ compressed source tarball](#)
 - Download [Gzipped source tarball](#)
- [Python 3.5.6rc1 - 2018-07-20](#)
 - Download [XZ compressed source tarball](#)
 - Download [Gzipped source tarball](#)
- [Python 3.7.0 - 2018-06-27](#)
 - Download [XZ compressed source tarball](#)
 - Download [Gzipped source tarball](#)
- [Python 3.6.6 - 2018-06-27](#)
 - Download [XZ compressed source tarball](#)

图2 ◆Python源码◆下载

Python的跨平台特性，可以让开发者放心的选择自己喜欢的操作系统和开发工具来开发代码。Python 不仅可以在Windows、Mac OS X、Linux/Unix运行，也可以运行在移动设备上，后面我们会有单独的章节来具体介绍。

Python有很多值得我们称道的特性，这里就先不展开介绍了，“绝知此事要躬行”，我们先搭建出开发环境来，实现我们的第一个“Hello world”。

1.1.1 安装Python

我们可以到 <https://www.python.org/downloads/windows/> 下载windows下的Phthon安装文件，我们下载独立的安装文件进行安装。

Python Releases for Windows

- [Latest Python 3 Release - Python 3.7.0](#)
- [Latest Python 2 Release - Python 2.7.15](#)
- [Python 3.7.0 - 2018-06-27](#)
 - Download [Windows x86 web-based installer](#)
 - Download [Windows x86 executable installer](#)
 - Download [Windows x86 embeddable zip file](#)
 - Download [Windows x86-64 web-based installer](#)
 - Download [Windows x86-64 executable installer](#)
 - Download [Windows x86-64 embeddable zip file](#)
 - Download [Windows help file](#)

Python安装文件下载（windows）

到<https://www.python.org/downloads/mac-osx/>下载Mac OS X的安装文件,

Python Releases for Mac OS X

- [Latest Python 3 Release - Python 3.7.0](#)
- [Latest Python 2 Release - Python 2.7.15](#)
- [Python 3.7.0 - 2018-06-27](#)
 - Download [macOS 64-bit installer](#)
 - Download [macOS 64-bit/32-bit installer](#)
- [Python 3.6.6 - 2018-06-27](#)
 - Download [macOS 64-bit installer](#)
 - Download [macOS 64-bit/32-bit installer](#)
- [Python 3.6.6rc1 - 2018-06-12](#)
 - Download [macOS 64-bit installer](#)
 - Download [macOS 64-bit/32-bit installer](#)
- [Python 3.7.0rc1 - 2018-06-11](#)
 - Download [macOS 64-bit installer](#)

Python安装文件下载 (Mac OS X)

到<https://www.python.org/download/other/>下载◆其他平台的安装文件。

Download Python for Other Platforms

Python has been ported to a number of specialized and/or older platforms, listed below in alphabetical order. Note that these ports often lag well behind the latest Python release.

Python for AIX

Python 3 and Python 2 are available for free for AIX in `installp` format at AIXTOOLS. See the [AIXTOOLS website](#) for general info. Currently, there are no additional pre-requisites other than a recent version of `openssl.base` for Python3-3.6.5.

The download page for Python 3 is at <http://www.aixtools.net/index.php/python3> and the download page for Python 2 is at <http://www.aixtools.net/index.php/python2>.

Python for IBM i (formerly AS/400, iSeries, System i)

Both Python 2.7 and Python 3.4 are available for free for IBM i via the 5733OPS licensed program offering. These binaries require IBM i 7.1 or newer. For more information about 5733OPS, see [the offering's landing page on IBM developerWorks](#). Alternatively, <http://www.iseriespython.com> hosts an IBM i port of Python 2.7, ported by Per Gummedal, which can also be run on older versions of the operating system.

Python for iOS

[Pythonista](#) is a complete development environment for writing Python scripts on your iPad or iPhone.

Python for OS/390 and z/OS

Python安装文件下载（其他平台）

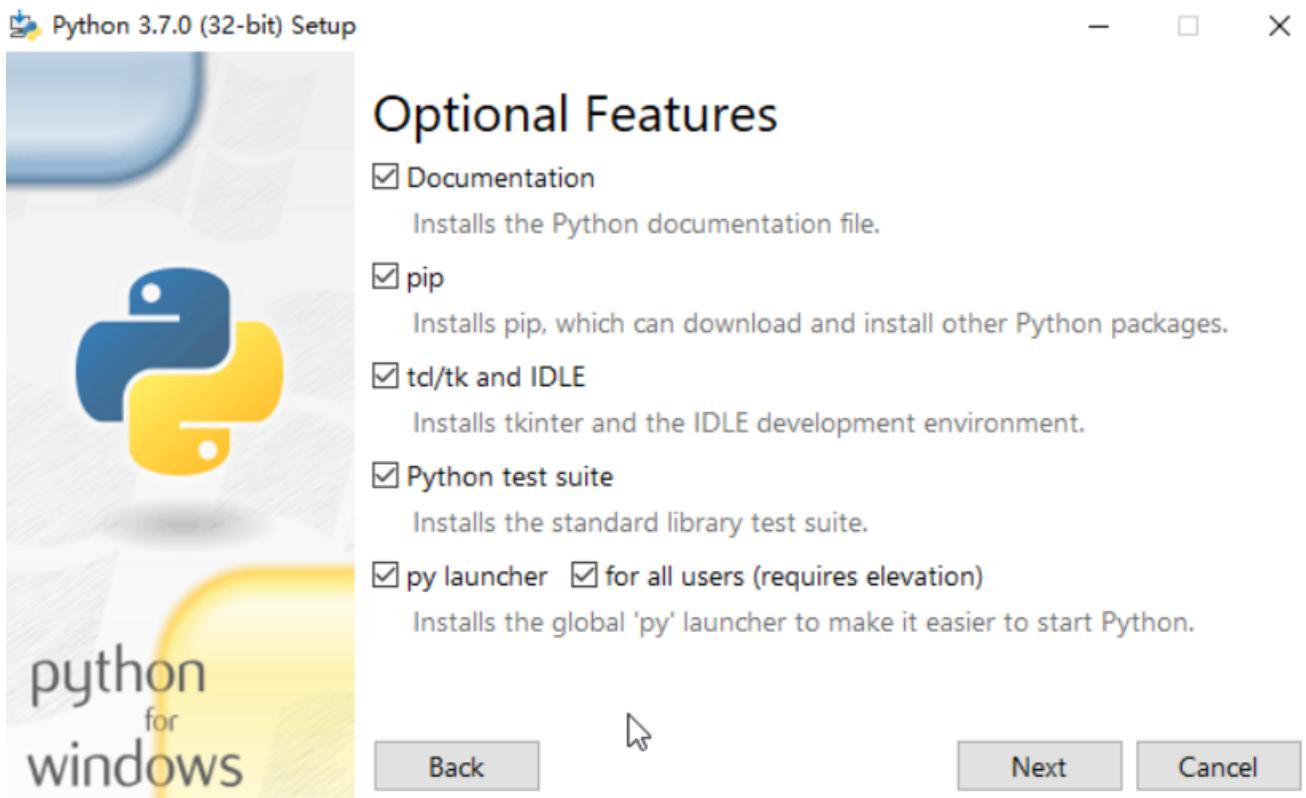
Linux 下安装Python◆一般采用◆源码或则软件包管理器安装的方式。下面分别介绍在不同操作系统下Python的安装方法。

1.1.1.1 Windows下安装Python

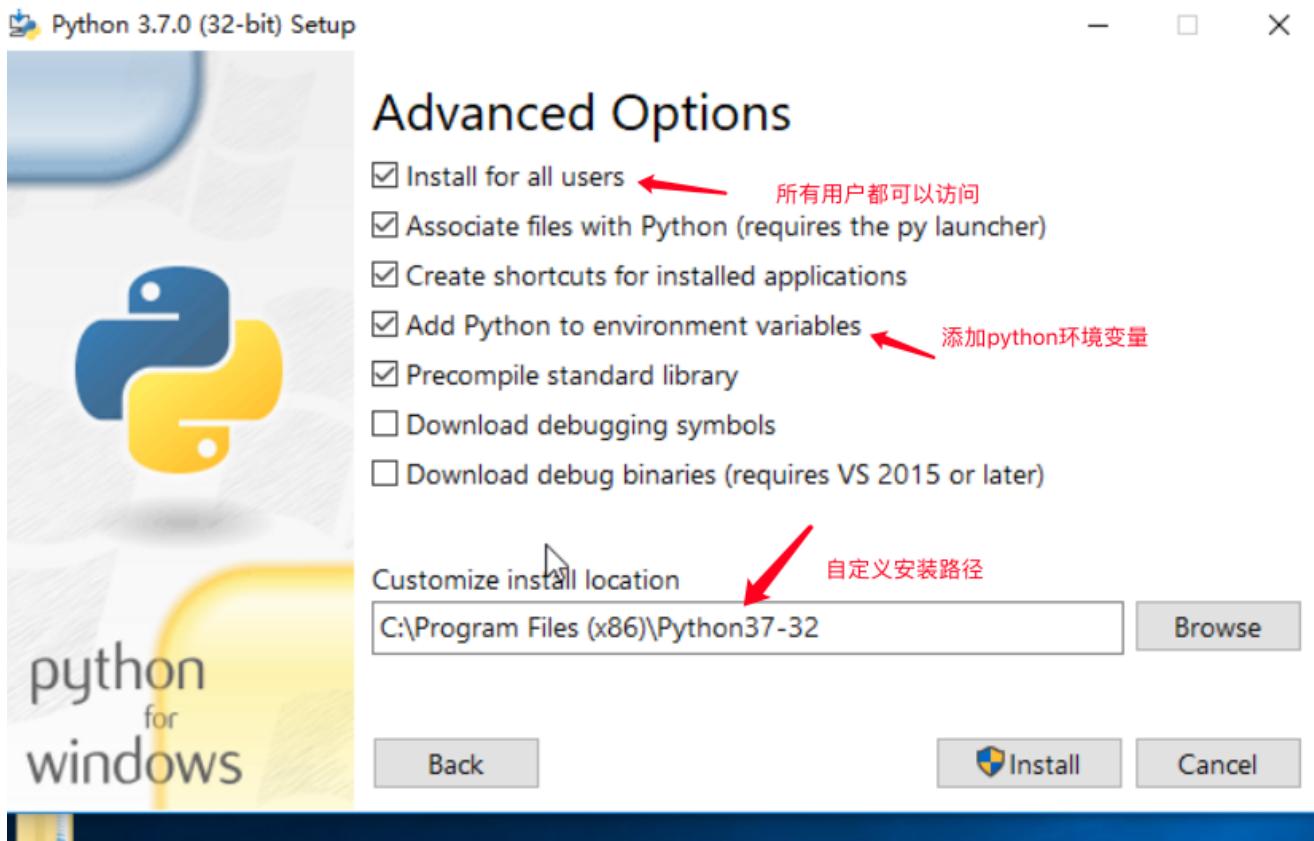
我们从下载安装文件之后，打开exe文件。



注意上图中的标记的选项。通常我们会把Python安装的系统的PATH系统变量中，这样我们就可以在任何地方直接调用Python。当然也可以不安装，需要到Python的安装目录里调用 `python.exe`。选择自定义安装。



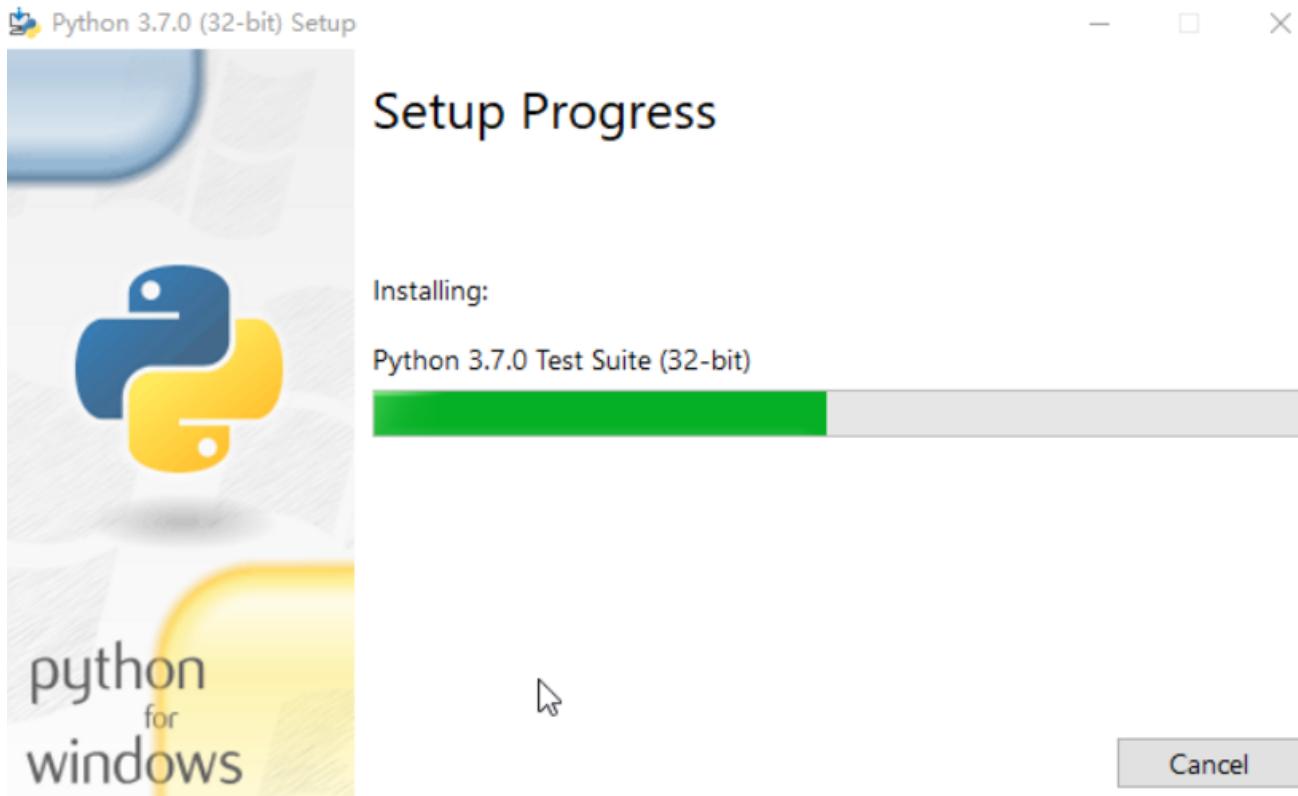
这里注意pip一定要勾选上，这是默认的Python包管理工具，后面我们会继续介绍。点击下一步。



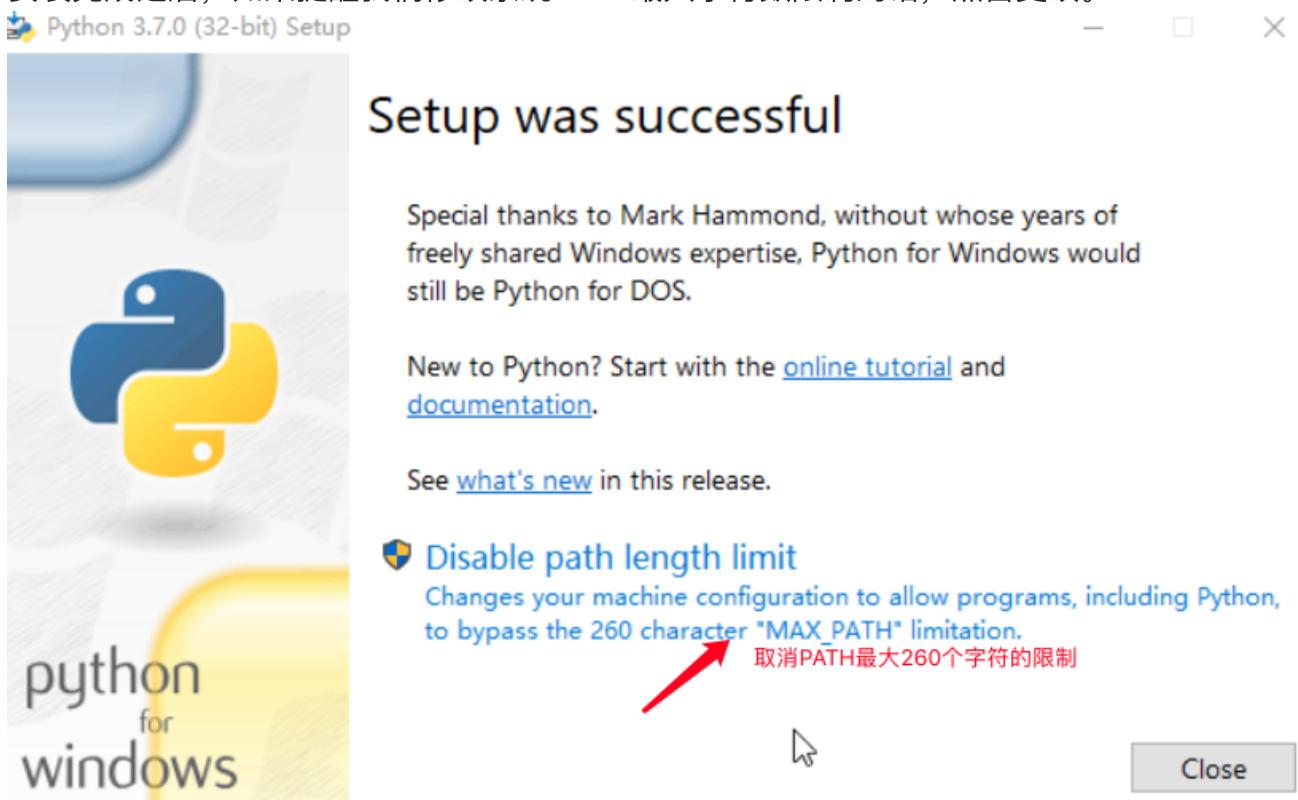
这里我们可以自定义安装目录，添加环境变量等选项。点击安装。



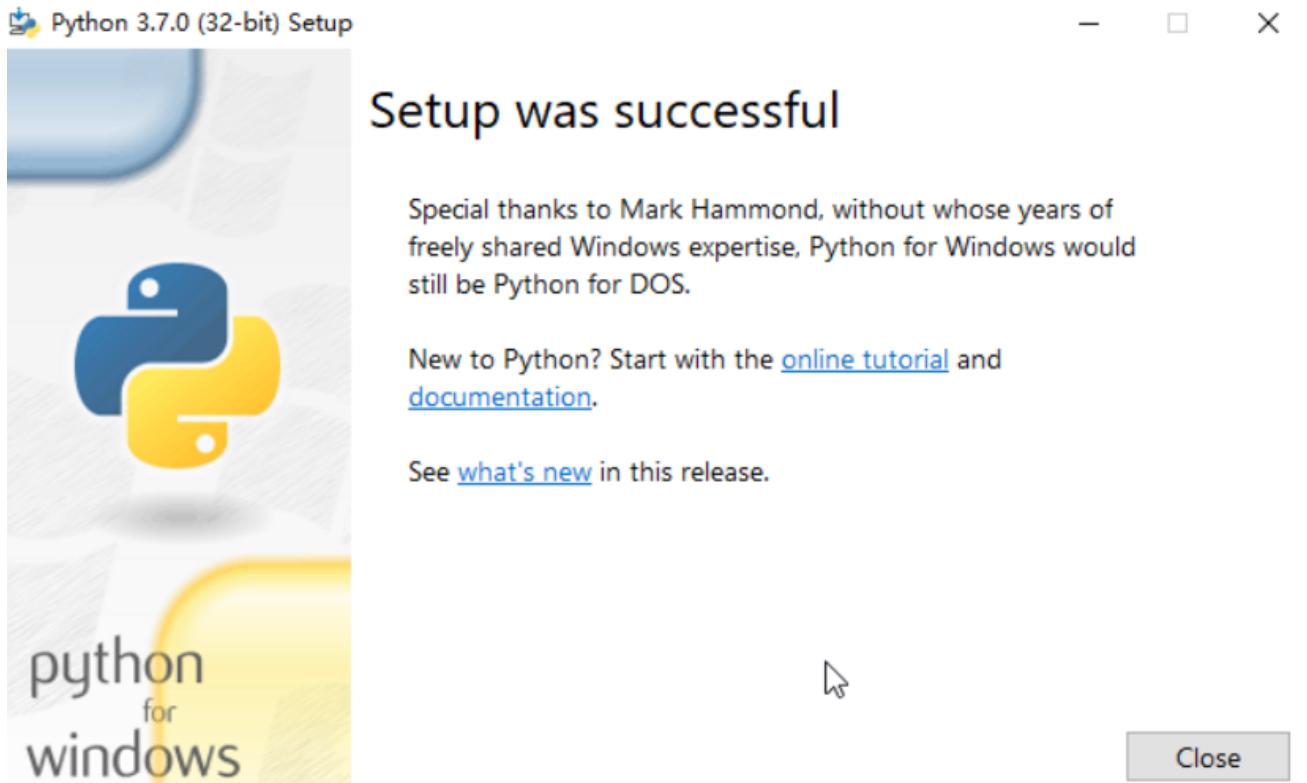
允许安装。



安装完成之后，如果提醒我们修改系统PATH最大字符数限制的话，点击更改。



安装成功后，**关闭对话框。**



现在我们可以进行**简单的测试了**。如果你选择了将Python安装到PATH，直接命令行输入Python启动运行时，否则到Python的安装目录启动python.exe。

A screenshot of a Windows Command Prompt window titled "命令提示符 - python". The window shows the following text:

```
Microsoft Windows [版本 10.0.16299.371]
(c) 2017 Microsoft Corporation. 保留所有权利。

C:\Users\xuanh>python
Python 3.7.0 (v3.7.0:bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> -
```

输入简单的语句测试下。

```

C:\Users\xuanh>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('hello')
hello
>>> a='ddddd'
>>> a
'dddddd'
>>> print(a)
ddddd
>>> 1+2
3
>>> exit()

C:\Users\xuanh>

```

到此为止，Python在widonws上的安装完成了，下面我们看看在Mac OS X上如何安装。

1.1.1.2 Mac OS X 安装Python

目前Mac OS X 默认安装Python2.7，我们下载最新的安装包。



点击继续。



安装“Python”

重要信息

- 介绍
- 请先阅读
- 许可
- 目的宗卷
- 安装类型
- 安装
- 摘要

This package will install Python 3.7.0 for macOS 10.9 or later for the following architecture(s): x86_64.

Which installer variant should I use?

For Python 3.7, python.org currently provides two installer variants for download: one that installs a *64-bit-only* Python capable of running on *macOS 10.9 (Mavericks)* or later; and one that installs a *64-bit/32-bit* Intel Python capable of running on *macOS 10.6 (Snow Leopard)* or later. (This ReadMe was installed with the *10.9 or later* variant.) If you are running on macOS 10.9 or later and if you have no need for compatibility with older systems, use the 10.9 variant. Use the 10.6 variant if you are running on macOS 10.6 through 10.8 or if you want to produce standalone applications that can run on systems from 10.6. The Pythons installed by these installers are built with private copies of some third-party libraries not included with or newer than those in macOS itself. The list of these libraries varies by installer variant and is included at the end of the License.rtf file.

Certificate verification and OpenSSL

This variant of Python 3.7 includes its own private copy of OpenSSL 1.1.0. The deprecated Apple-supplied OpenSSL libraries are no longer

打印... 存储... 返回 继续



安装“Python”

软件许可协议

HISTORY AND LICENSE

HISTORY OF THE SOFTWARE

Python was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum (CWI, see <http://www.cwi.nl>) in the Netherlands as a successor of a language called ABC. Guido remains Python's principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for National Research Initiatives (CNRI, see <http://www.cnri.reston.va.us>) in Reston, Virginia where he released several versions of the software.

In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same year, the PythonLabs team moved to Digital Creations (now Zope Corporation, see <http://www.zope.com>). In 2001, the Python Software Foundation (PSF, see <http://www.python.org/psf/>) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation is a sponsoring member of the PSF.

All Python releases are Open Source (see <http://www.opensource.org> for the Open Source Definition). Historically most, but not all, Python

打印... 存储... 返回 继续



点击“安装”。



安装成功之后，如果要求我们安装SSL根证书的话，需要点击链接跳转到下载页面下载安装。其实，也可以直接通过pip来安装。

在使用pip之前，我们先对pip进行更新。

```
pip3 install --upgrade pip
```

```
xuanhundeMBP:~ xuanhun$ pip3 install --upgrade pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/5f/25/e52d3f31441505a5f3a
41213346e5b6c221c9e086a166f3703d2ddaf940/pip-18.0-py2.py3-none-any.whl (1.3MB)
    100% |████████████████████████████████| 1.3MB 66kB/s
Installing collected packages: pip
  Found existing installation: pip 10.0.1
    Uninstalling pip-10.0.1:
      Successfully uninstalled pip-10.0.1
Successfully installed pip-18.0
```

之后执行命令安装certifi。

```
pip install certifi
```

下面我们来验证新安装的Python。

直接输入python，使用的还是Python2.7.

```
[xuanhundeMBP:~ xuanhun$ python
Python 2.7.10 (default, Oct  6 2017, 22:29:07)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> exit()
xuanhundeMBP:~ xuanhun$ ]
```

任务

默认的环境变量，我们不去动，这里使用python3来启动。

```
[xuanhundeMBP:~ xuanhun$ python3
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> print('hello world')
hello world
>>> ]
```

任务

1.1.1.3 Linux下安装Python

安装python

在Linux下安装Python，一般采用源码安装的方式。首先下载源码。

```
$ sudo mkdir /usr/local/python3 # 创建安装目录  
# 下载 Python 源文件  
$ wget --no-check-certificate https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz  
# 注意: wget获取https的时候要加上: --no-check-certificate  
$ tar -xzvf Python-3.7.0.tgz # 解压缩包  
$ cd Python-3.7.0 # 进入解压目录
```

第二步是编译安装。

```
$ sudo ./configure --prefix=/usr/local/python3 # 指定创建的目录  
$ sudo make  
$ sudo make install
```

第三步是配置链接。◆因为很多Linux系统自带Python2.7，所以这里要配置◆一下软连接，使得我们可以使用python3 命令来启动Python3.7.

```
$ sudo ln -s /usr/local/python3/bin/python3 /usr/bin/python3
```

如果使用的是apt软件包管理器的系统，可以直接使用如下命令来安装：

```
apt-get install python3
```

◆安装pip

可以使用命令直接安装pip。

```
sudo yum install python-pip
```

◆或者

```
sudo apt-get install python-pip
```

简单测试同上。

1.1.2 包管理

在实际开发过程中，不可能所有功能都基于Python原生API来进行开发，很多第三方软件包为我们提供了很好的便利条件。安装和管理软件包分为手动和工具两种方式，下面我们简单介绍这两种方式。软件包管理工具为大家介绍pip和easy_install。

1.1.2.1 ◆手动安装

第一种方法是手动下载软件包，运行安装程序来安装。

比如我们想要使用Python-nmap组件去解析nmap的扫描结果，我们先手动下载安装包。

```
wget http://xael.org/pages/python-nmap-0.6.0.tar.gz
```

```
root@kali: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@kali: ~# wget http://xael.org/pages/python-nmap-0.6.0.tar.gz
-- 2016-05-04 12:43:09 -- http://xael.org/pages/python-nmap-0.6.0.tar.gz
正在解析主机 xael.org (xael.org)... 194.36.166.10
正在连接 xael.org (xael.org) |194.36.166.10|:80... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度 : 41415 (40K) [application/x-gzip]
正在保存至: "python-nmap-0.6.0.tar.gz"

python-nmap-0.6.0.t 100%[=====] 40.44K 22.7KB/s 用时 1.8s
2016-05-04 12:43:14 (22.7 KB/s) - 已保存 “python-nmap-0.6.0.tar.gz” [41415/41415]
])
root@kali: ~#
```

解压：

```
root@kali: ~# tar xzf python-nmap-0.6.0.tar.gz
root@kali: ~# cd python-nmap-0.6.0/
root@kali: ~/python-nmap-0.6.0# ls
root@kali: ~/python-nmap-0.6.0#
```

```
paros      python-nmap-0.6.0.tar.gz 模板 图片 下载 桌面
phpmyadmin.sh 公共 视频 文档 音乐
root@kali: ~# tar xzf python-nmap-0.6.0.tar.gz
root@kali: ~#
paros      python-nmap-0.6.0 公共 视频 文档 音乐
phpmyadmin.sh python-nmap-0.6.0.tar.gz 模板 图片 下载 桌面
root@kali: ~# cd python-nmap-0.6.0/
root@kali: ~/python-nmap-0.6.0# ls
CHANGELOG  gpl-3.0.txt  MANIFEST.in  nmap.html  README.txt      setup.py
example.py  Makefile     nmap        PKG-INFO    requirements.txt
root@kali: ~/python-nmap-0.6.0#
```

安装：

```
python setup.py install
```

```
root@kali: ~/python-nmap-0.6.0
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
running build
running build_py
creating build
creating build/lib.linux-x86_64-2.7
creating build/lib.linux-x86_64-2.7/nmap
copying nmap/__init__.py -> build/lib.linux-x86_64-2.7/nmap
copying nmap/test_nmap.py -> build/lib.linux-x86_64-2.7/nmap
copying nmap/nmap.py -> build/lib.linux-x86_64-2.7/nmap
running install_lib
creating /usr/local/lib/python2.7/dist-packages/nmap
copying build/lib.linux-x86_64-2.7/nmap/__init__.py -> /usr/local/lib/python2.7/
dist-packages/nmap
copying build/lib.linux-x86_64-2.7/nmap/test_nmap.py -> /usr/local/lib/python2.7/
dist-packages/nmap
copying build/lib.linux-x86_64-2.7/nmap/nmap.py -> /usr/local/lib/python2.7/dist-
packages/nmap
byte-compiling /usr/local/lib/python2.7/dist-packages/nmap/__init__.py to __init__
.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/nmap/test_nmap.py to test_
nmap.pyc
byte-compiling /usr/local/lib/python2.7/dist-packages/nmap/nmap.py to nmap.pyc
running install_egg_info
Writing /usr/local/lib/python2.7/dist-packages/python_nmap-0.6.0.egg-info
root@kali: ~/python-nmap-0.6.0#
```

1.1.2.2 pip

上面安装Python的讲解中已经讲解了如何安装pip。pip是Python安装包默认的包管理工具，下面举例说明基本用法。

通过 pip 来安装github3模块：

```
pip install github3.py
```

```
root@kali: ~/python-nmap-0.6.0# pip install github3.py
Downloading/unpacking github3.py
  Downloading github3.py-0.9.5-py2.py3-none-any.whl (109kB): 109kB downloaded
Downloading/unpacking uritemplate>=0.2.0 (from github3.py)
  Downloading uritemplate-0.3.0.tar.gz
  Running setup.py (path:/tmp/pip-build-LRgVNm/uritemplate.py/setup.py) egg_info
for package uritemplate.py

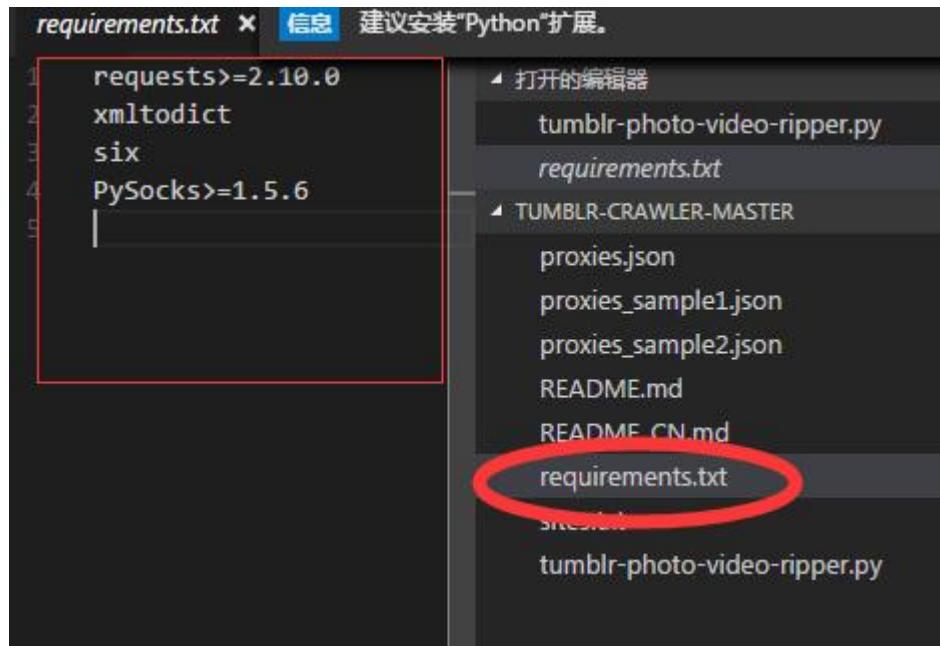
Requirement already satisfied (use --upgrade to upgrade): requests>=2.0 in /usr/
lib/python2.7/dist-packages (from github3.py)
Installing collected packages: github3.py, uritemplate.py
  Running setup.py install for uritemplate.py

Successfully installed github3.py uritemplate.py
Cleaning up...
root@kali: ~/python-nmap-0.6.0#
```

如果要安装特定版本的package，通过使用==, >=, <=, >, <来指定一个版本号。例如：

```
pip install 'Markdown<2.0'  
pip install 'Markdown>2.0,<2.0.3'
```

我们也可以将所有的依赖放到一个requirement文件中，一次性安装。例如新建内容如下的requirements.txt文件：



执行命令：

```
pip install -r requirements.txt
```

卸载软件包，使用uninstall选项：

```
pip uninstall SomePackage
```

更新软件包：

```
pip install --upgrade SomePackage
```

显示已经安装的文件：

```
pip show --files SomePackage
```

显示过期的安装包：

```
pip list --outdated
```

1.1.2.3 easy_install

easy_install 是Python setuptools系列工具的中的一个工具，可以用来自动查找、下载、安装、升级依赖包。

在Kali Linux中Python setuptools默认已经被安装，其他Linux系统中使用apt-get或者yum都可以安装。

apt-get 安装命令为：

```
sudo apt-get install python-setuptools
```

yum 安装命令为：

```
yum install setuptool
```

这里再介绍一个通用的方法，适合所有操作系统。首先下载ez_setup.py (https://bootstrap.pypa.io/ez_setup.py) 文件，然后执行下面的命令即可：

```
python ez_setup.py
```

下面我们使用easy_install 来安装Python的一个模块，可以用来对pdf进行解析和安全测试的pyPdf。

```
easy_install pyPdf
```

```
root@kali: ~/python-nmap-0.6.0# easy_install pyPdf
Searching for pyPdf
Best match: pyPdf 1.13
Adding pyPdf 1.13 to easy-install.pth file

Using /usr/lib/pymodules/python2.7
Processing dependencies for pyPdf
Finished processing dependencies for pyPdf
root@kali: ~/python-nmap-0.6.0#
```

easy_install当然也提供了卸载模块/包的功能。但是必须要注意的是，该模块/包必须要在easy-install.pth有相关信息，换句话说，也就是要使用easy_install安装的，才可进行卸载。比如命令：

```
easy_install -m redis
```

这样就会将redis模块卸载。

1.1.3 virtualenv

virtualenv是可以用来创建独立Python环境的工具, 来解决依赖、版本以及间接权限问题. 比如一个应用依赖Python 2.7 而另一个应用依赖Python 3.0, 或者一个项目依赖Django1.3 而当前全局开发环境为Django1.7, 版本跨度过大, 导致不兼容使项目无法正在运行, 使用virtualenv可以解决这些问题. 基本原理为:

virtualenv创建一个拥有自己安装目录的环境, 这个环境不与其他虚拟环境共享库, 能够方便的管理python版本和管理python库.

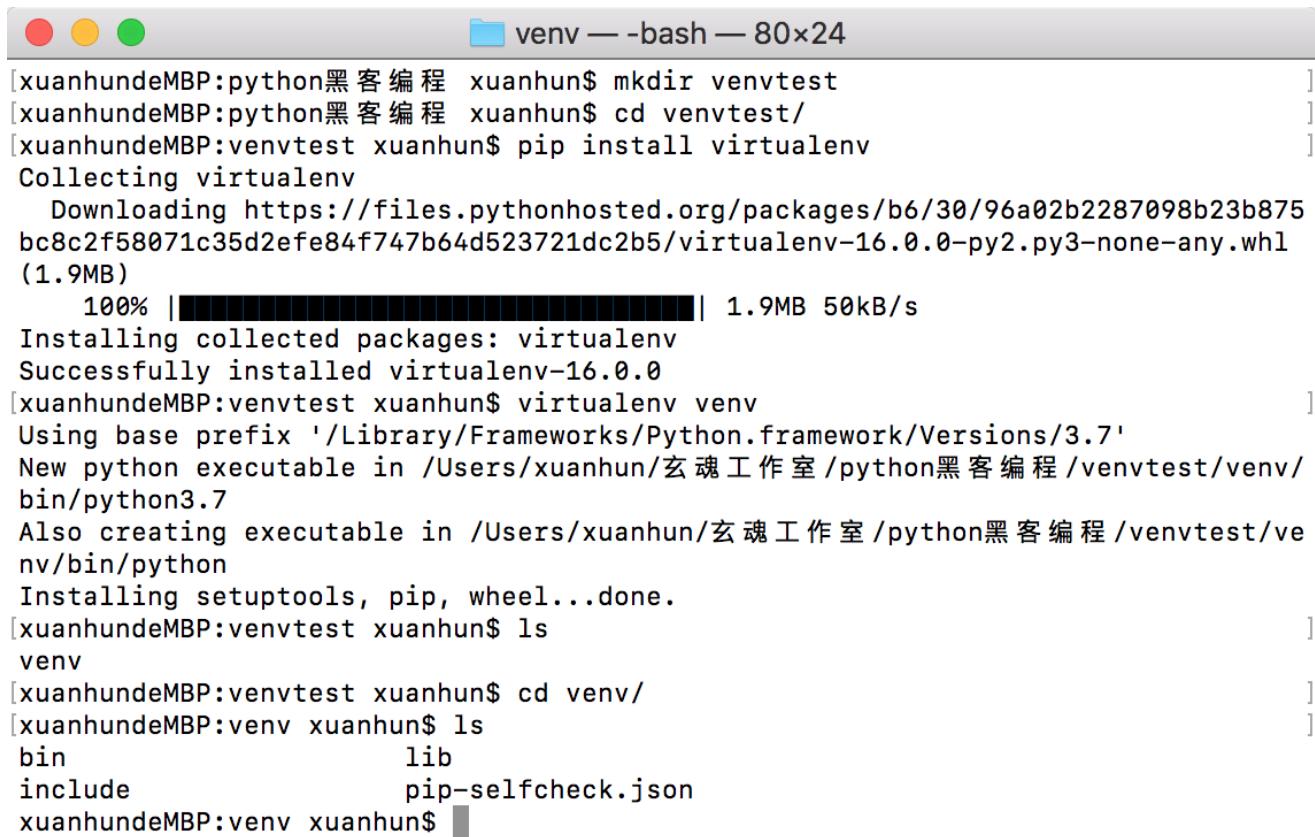
1. 安装Virtualenv

```
sudo pip install virtualenv
```

2. 创建项目的虚拟环境

```
mkdir testvenv #名字随便取
cd testvenv
virtualenv venv # venv 可替换为别的虚拟环境名称
```

执行后, 在本地会生成一个与虚拟环境同名的文件夹, 包含 Python 可执行文件和 pip 库的拷贝, 可用于安装其他包。但是默认情况下, 虚拟环境中不会包含也无法使用系统环境的global site-packages。比如系统环境里安装了 requests 模块, 在虚拟环境里 import requests 会提示 ImportError。如果想使用系统环境的第三方软件包, 可以在创建虚拟环境时使用参数--system-site-packages。



```
[xuanhundeMBP:python黑客编程 xuanhun$ mkdir venvtest
[xuanhundeMBP:python黑客编程 xuanhun$ cd venvtest/
[xuanhundeMBP:venvtest xuanhun$ pip install virtualenv
Collecting virtualenv
  Downloading https://files.pythonhosted.org/packages/b6/30/96a02b2287098b23b875bc8c2f58071c35d2efe84f747b64d523721dc2b5/virtualenv-16.0.0-py2.py3-none-any.whl
(1.9MB)
  100% |██████████| 1.9MB 50kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-16.0.0
[xuanhundeMBP:venvtest xuanhun$ virtualenv venv
Using base prefix '/Library/Frameworks/Python.framework/Versions/3.7'
New python executable in /Users/xuanhun/玄魂工作室/python黑客编程/venvtest/venv/bin/python3.7
Also creating executable in /Users/xuanhun/玄魂工作室/python黑客编程/venvtest/venv/bin/python
Installing setuptools, pip, wheel...done.
[xuanhundeMBP:venvtest xuanhun$ ls
venv
[xuanhundeMBP:venvtest xuanhun$ cd venv/
[xuanhundeMBP:venv xuanhun$ ls
bin                  lib
include              pip-selfcheck.json
xuanhundeMBP:venv xuanhun$ ]
```

```
virtualenv --system-site-packages venv
```

另外，你还可以自己指定虚拟环境所使用的 Python 版本，但前提是系统中已经安装了该版本：

```
virtualenv -p /usr/bin/python2.7 venv
```

3. 使用虚拟环境

进入虚拟环境目录，启动虚拟环境：

```
cd venv  
source bin/activate # Windows 系统下运行 Scripts\  
python -V
```

如果未对命令行进行个性化，此时命令行前面应该会多出一个括号，括号里为虚拟环境的名称。启动虚拟环境后安装的所有模块都会安装到该虚拟环境目录里。

退出虚拟环境：

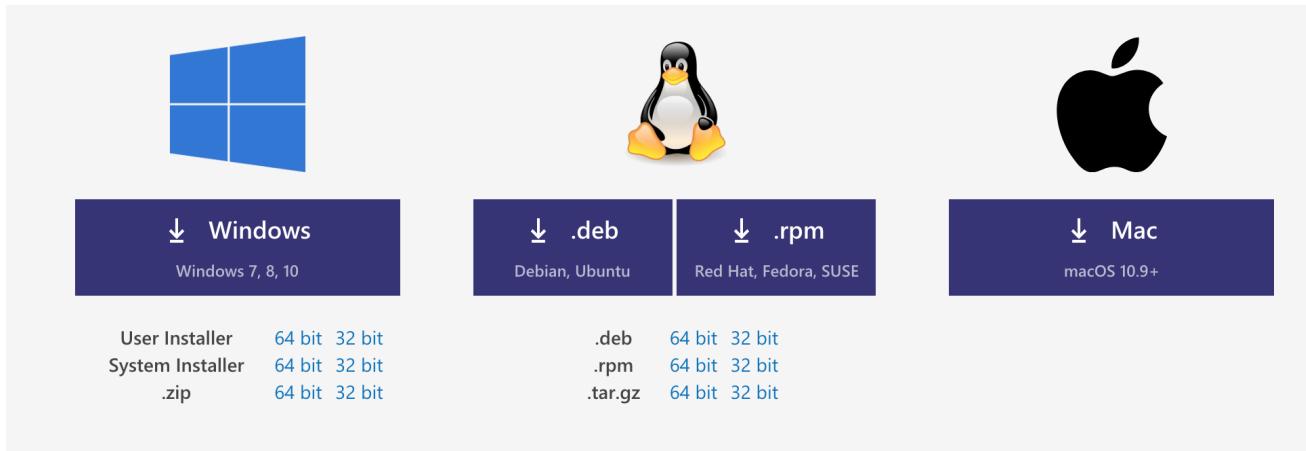
```
deactivate
```

```
[xuanhundeMBP:venv xuanhun$ source bin/activate  
[(venv) xuanhundeMBP:venv xuanhun$ python  
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24)  
[Clang 6.0 (clang-600.0.57)] on darwin  
Type "help", "copyright", "credits" or "license" for more information.  
'hello'  
[(venv) xuanhundeMBP:venv xuanhun$ deactivate  
xuanhundeMBP:venv xuanhun$ ]]
```

1.1.4 基于VS Code 搭建开发环境

可以用来开发Python的开发工具很多，选择 Visual Studio Code是因为它轻量，开源，跨平台，方便我们在不同的操作系统上开发和调试代码。

在 官方网站 <https://code.visualstudio.com/#alt-downloads>，可以知道它提供的各种系统的安装包。



windows 和 Mac下的安装都很简单，下面简单介绍Linux下的安装方法。

基于Debian/Ubuntu发行版的Linux，我们下载.deb安装包，然后执行以下命令：

```
sudo dpkg -i <file>.deb
sudo apt-get install -f # Install dependencies
```

基于RHEL, Fedora 和 CentOS 发行版的Linux，下载.rpm安装包进行安装。

以64位安装包为例，首先❸需要注册更新yum仓库。

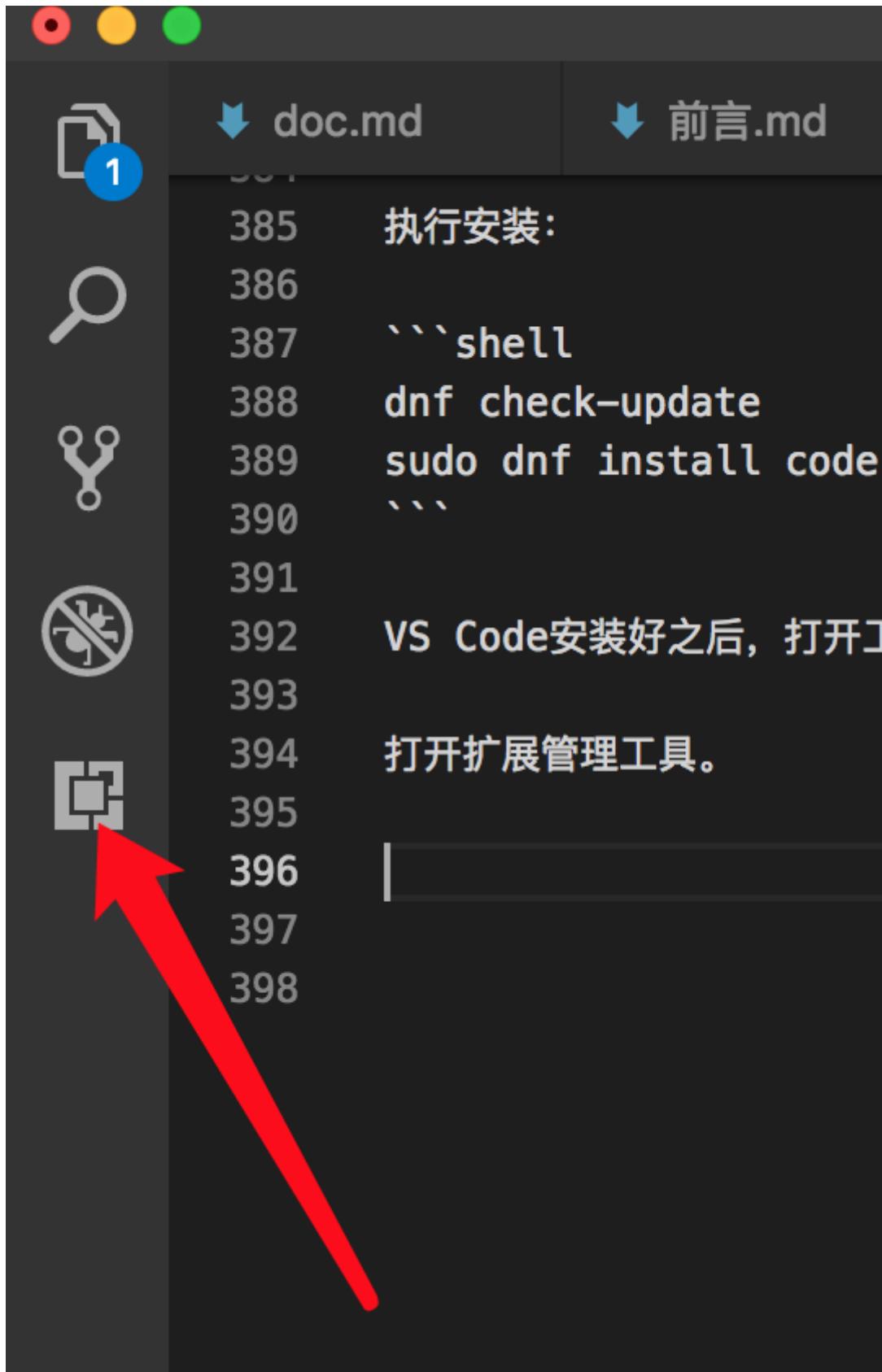
```
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
sudo sh -c 'echo -e "[code]\nname=Visual Studio Code\nbaseurl=https://packages
```

执行安装：

```
dnf check-update
sudo dnf install code
```

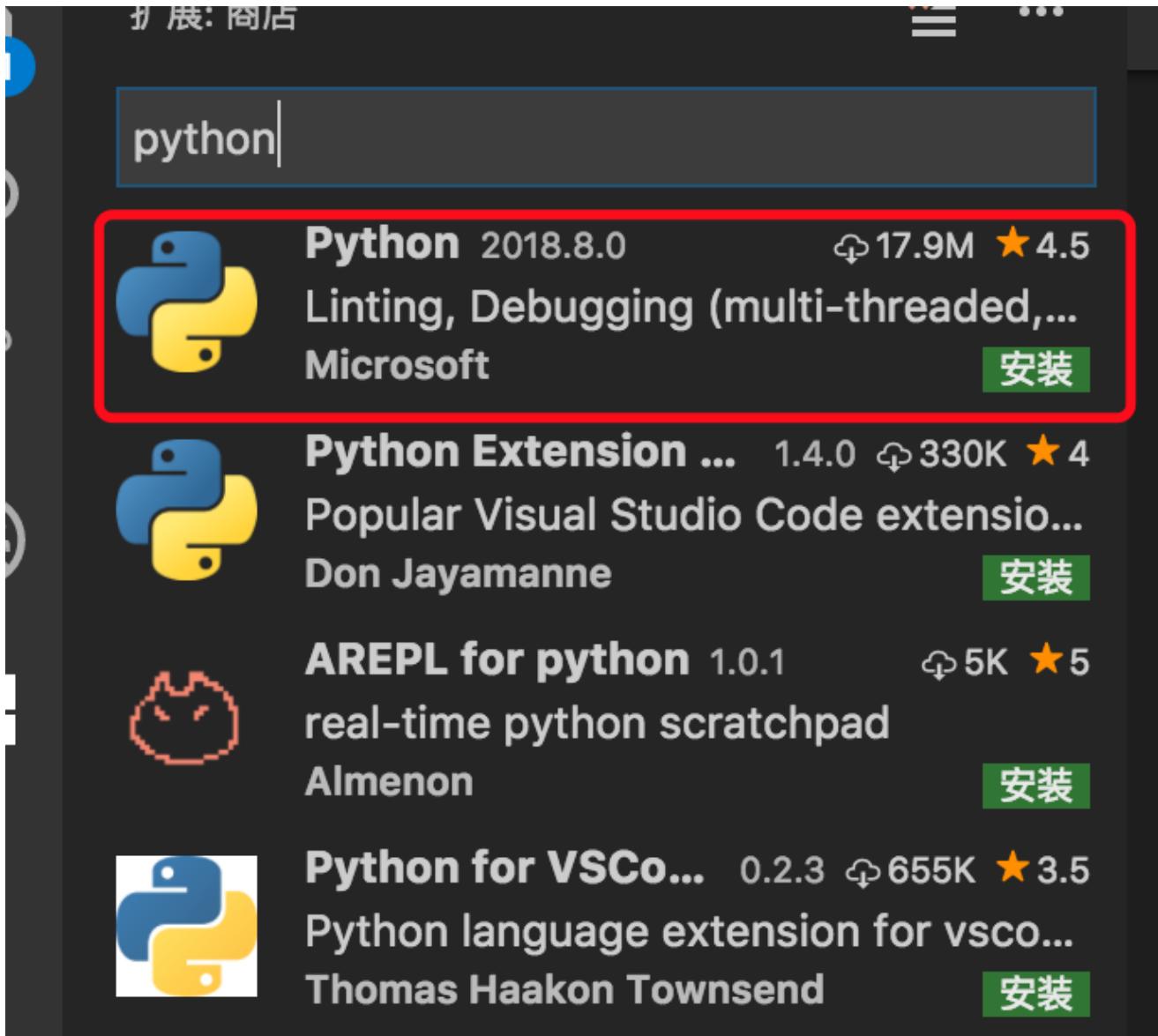
VS Code安装好之后，打开工具，安装扩展以支持Python开发。

打开扩展管理工具。

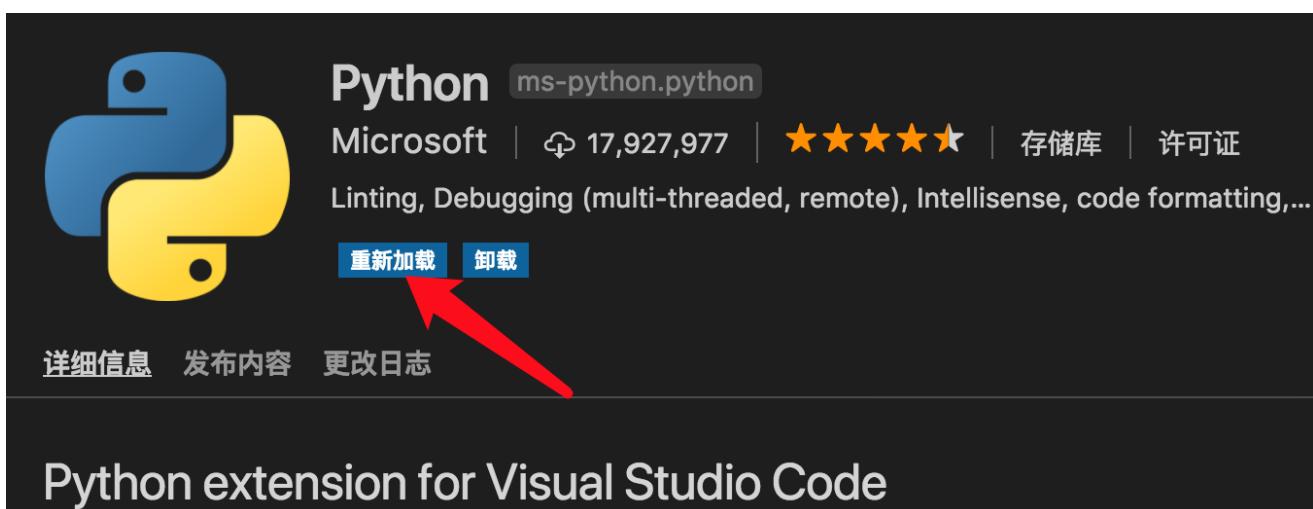


```
doc.md 前言.md
385 执行安装:
386
387 ````shell
388 dnf check-update
389 sudo dnf install code
390 `````
391
392 VS Code安装好之后，打开工
393
394 打开扩展管理工具。
395
396
397
398
```

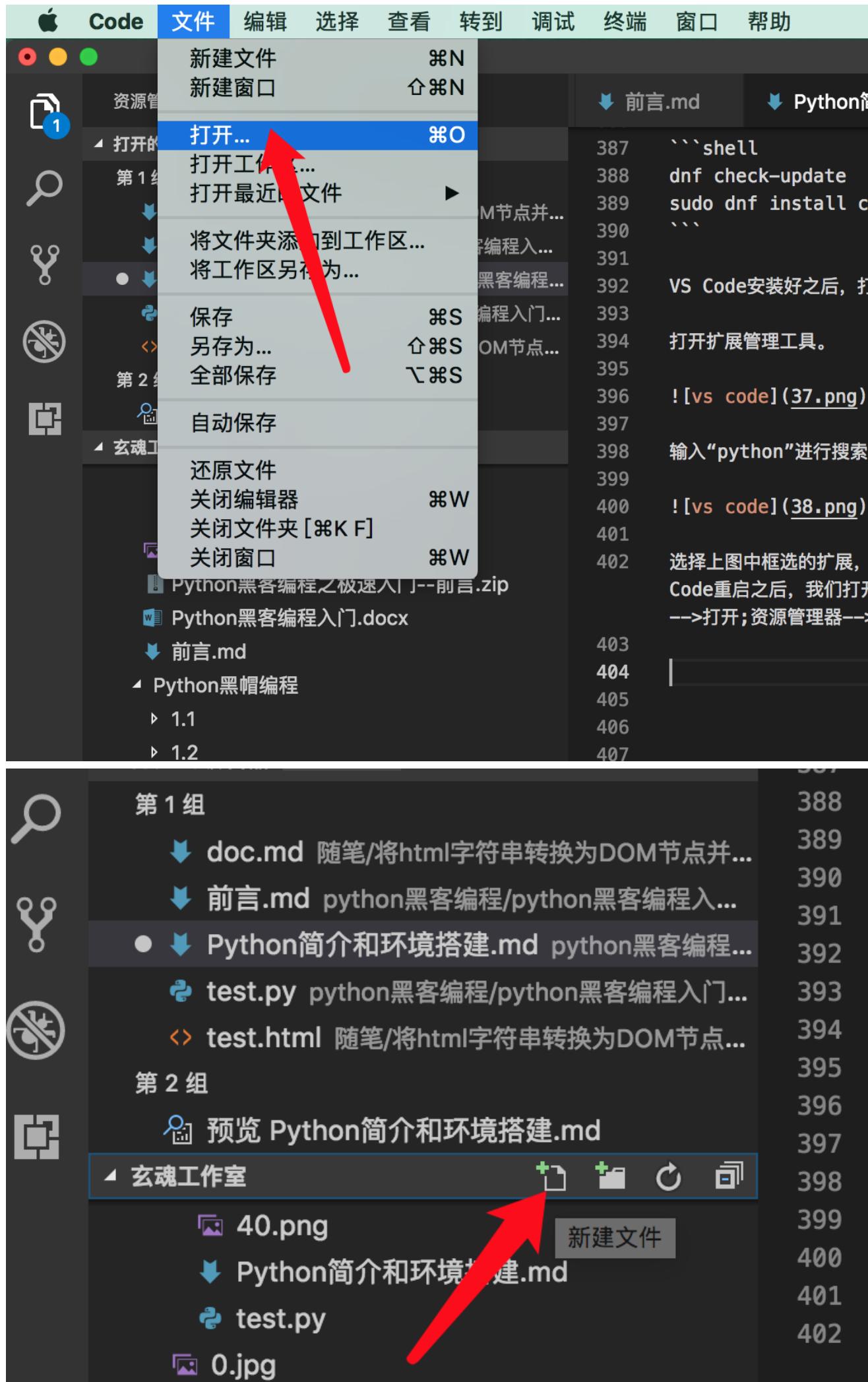
输入“?python”进行搜索。



选择上图中框选的扩展，点击“安装”按钮。安装完成后，点击“重新加载”。

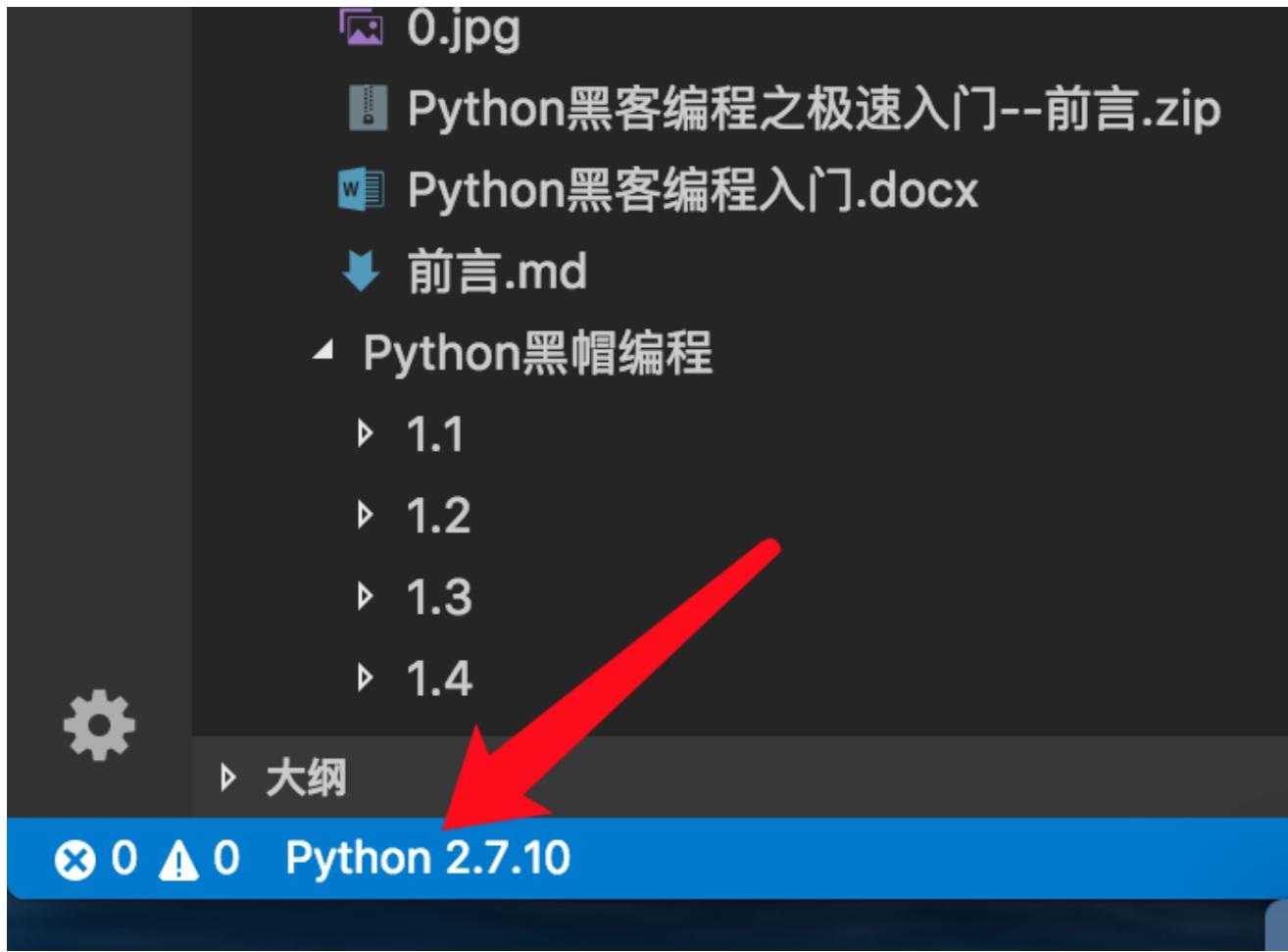


VS Code重启之后，我们打开一个文件夹，新建一个test.py文件（使用菜单-->文件-->打开；资源管理器-->新建文件）。



■ Python黑客编程之极速入门--前言.zip	
Word Python黑客编程入门.docx	403
↓ 前言.md	404
◀ Python黑帽编程	405
▷ 1.1	406
▷ 1.2	407
	408

新建 "*.py"文件后，在资源管理器左下角，可以切换Python的版本。



```

↓ 前言 current: python
396 Python 2.7.10 64-bit
397 /usr/bin/python2.7
398
399 Python 3.6.4 32-bit
400 /usr/local/bin/python3-32
401 Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24) [Clang 6.0 (clang-600.0.57)]
402 /Library/Frameworks/Python.framework/Versions/3.7/bin/python3
403
404 Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24) [Clang 6.0 (clang-600.0.57)] 6
405 /usr/local/bin/python3
406 VS Code重启之后，我们打开一个文件夹，新建一个test.py文件（使用菜单-->文件-->打开；资源管理器-->新建文件）。
407
408 ! [vs code] (40.png)
409 ! [vs code] (41.png)
410
411 新建 “*.py”文件后，在资源管理器左下角，可以切换Python的版本。
412
413
414
415
416
417

```

选择合适的版本之后，VS Code会提醒我们安装pylint，点击安装即可。我们来安装pylint
(Pylint是一个Python代码分析工具，它分析Python代码中的错误，查找不符合编码风格的代码)。

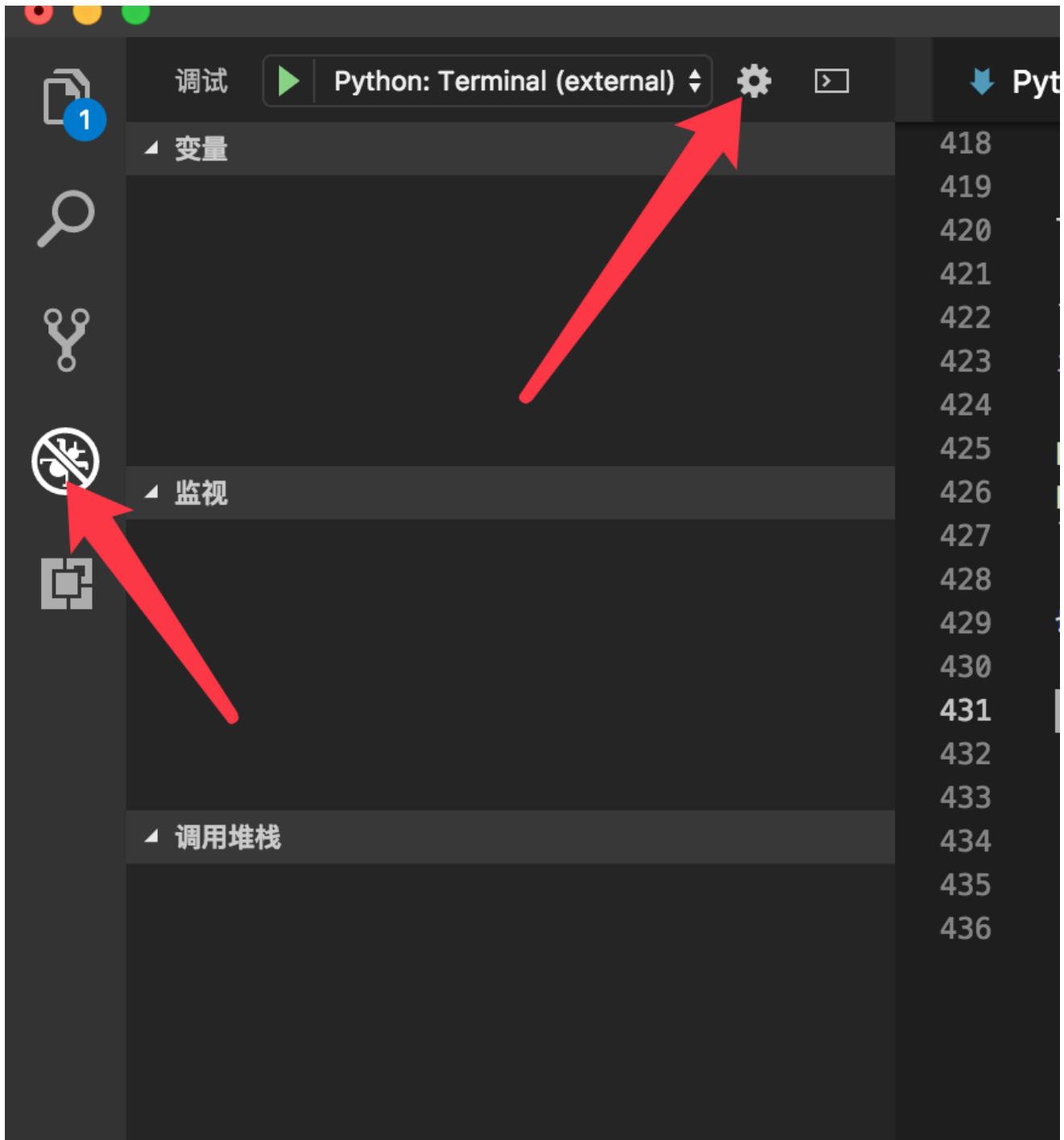
```
xuanhundeMacBook-Pro:玄魂工作室 xuanhun$ /usr/local/bin/python3 -m pip install -U pylint --user
Collecting pylint
  Downloading https://files.pythonhosted.org/packages/6e/c2/1e97c238877b6a86562d32297eb33a6670b6220e8ec0ca85f67
  whl (737kB)
    100% |██████████| 747kB 11.1MB/s
Collecting mccabe (from pylint)
  Downloading https://files.pythonhosted.org/packages/87/89/479dc97e18549e21354893e4ee4ef36db1d237534982482c368
  any.whl
Collecting astroid>=2.0.0 (from pylint)
  Downloading https://files.pythonhosted.org/packages/19/92/6f6d3591c429dbdb31c18d8476ba1af08d5973d7cc09f663461
  .whl (172kB)
```

下面我们回到test.py文件，随便加◆几行代码。

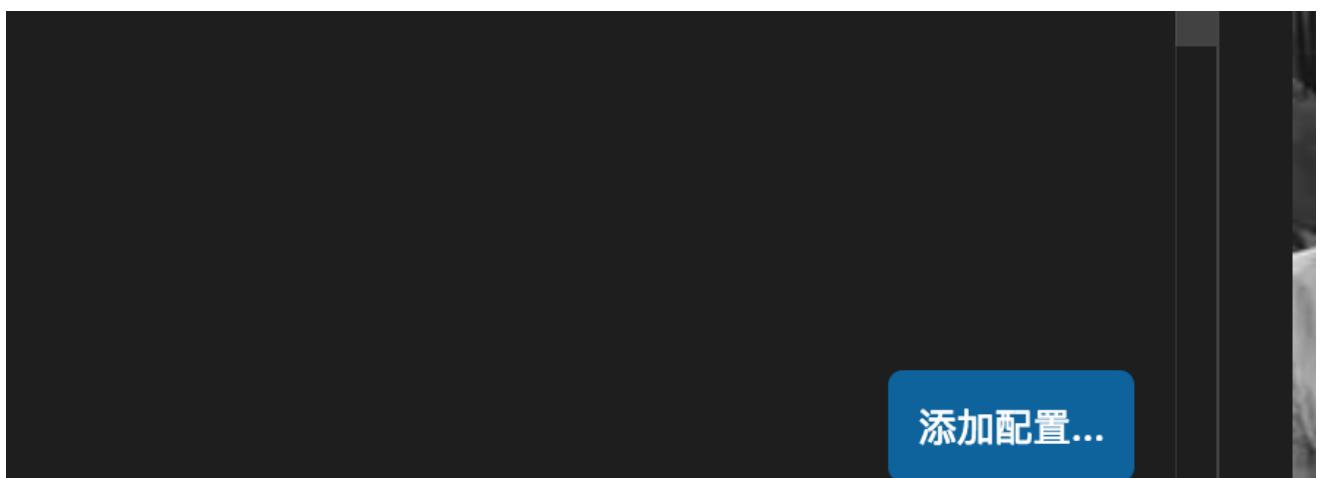
```
import socket

print("hello\n")
print("world!")
```

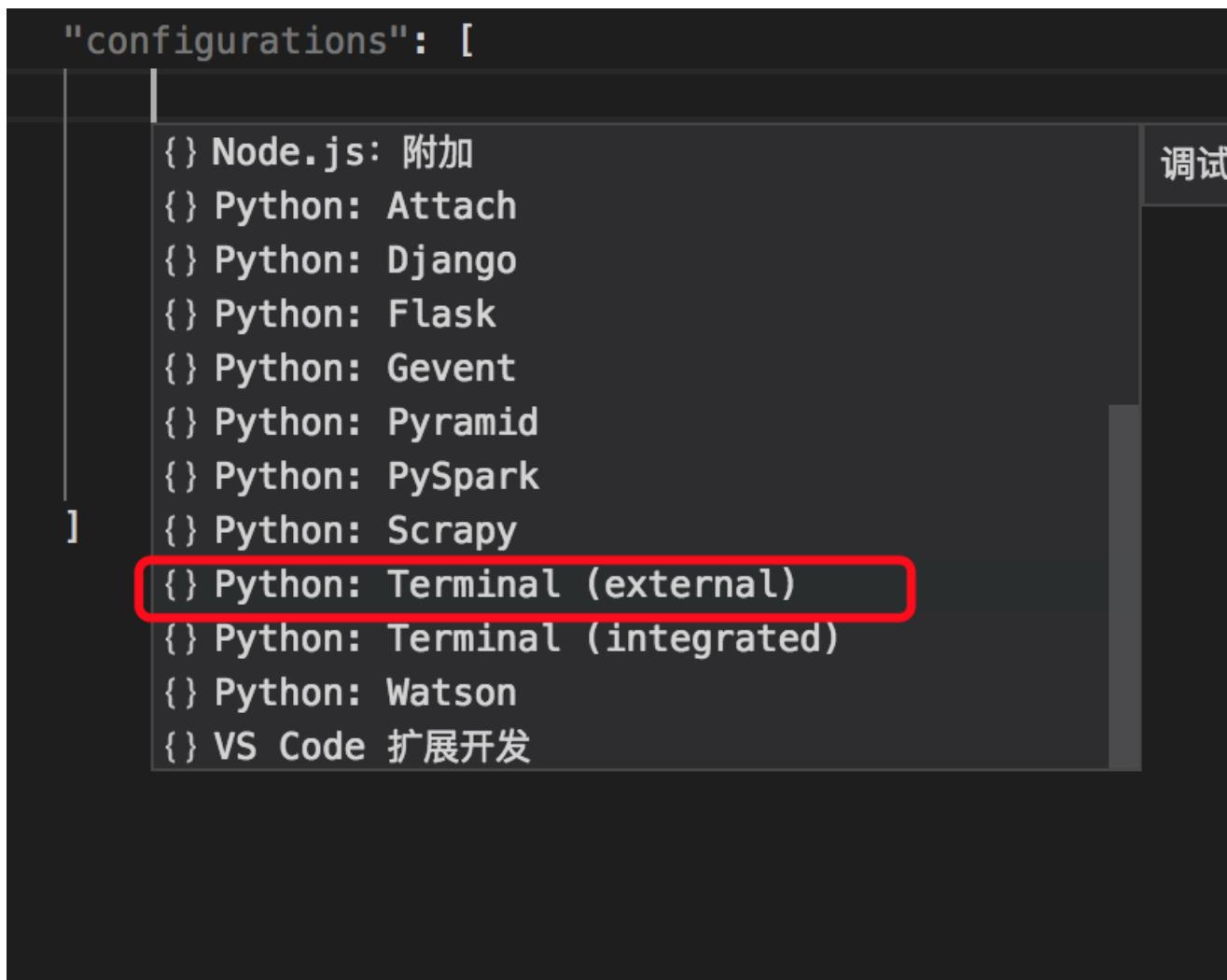
切换到◆调试菜单，点击“配置”按钮。



编辑器会打开launch.json文件，点击下方的添加配置。



选择使用外部终端的Python调试器。



```
"configurations": [
    {
        "name": "Node.js: 附加",
        "type": "node"
    },
    {
        "name": "Python: Attach",
        "type": "python"
    },
    {
        "name": "Python: Django",
        "type": "python"
    },
    {
        "name": "Python: Flask",
        "type": "python"
    },
    {
        "name": "Python: Gevent",
        "type": "python"
    },
    {
        "name": "Python: Pyramid",
        "type": "python"
    },
    {
        "name": "Python: PySpark",
        "type": "python"
    },
    {
        "name": "Python: Scrapy",
        "type": "python"
    },
    {
        "name": "Python: Terminal (external)", // 当前激活的文件作为启动项
        "type": "python"
    },
    {
        "name": "Python: Terminal (integrated)",
        "type": "python"
    },
    {
        "name": "Python: Watson",
        "type": "python"
    },
    {
        "name": "VS Code 扩展开发",
        "type": "extension"
    }
]
```

添加的配置如下：

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Python: Terminal (external)", // 当前激活的文件作为启动项
      "type": "python",
      "request": "launch",
      "program": "${file}", // 当前激活的文件作为启动项
      "console": "terminal.external.osxExec" // 启用外部终端
    }
  ]
}
```

保存配置之后我们切回test.py，在代码行数标识的左侧单击可以添加断点，方便我们逐行或者逐过程调试代码。

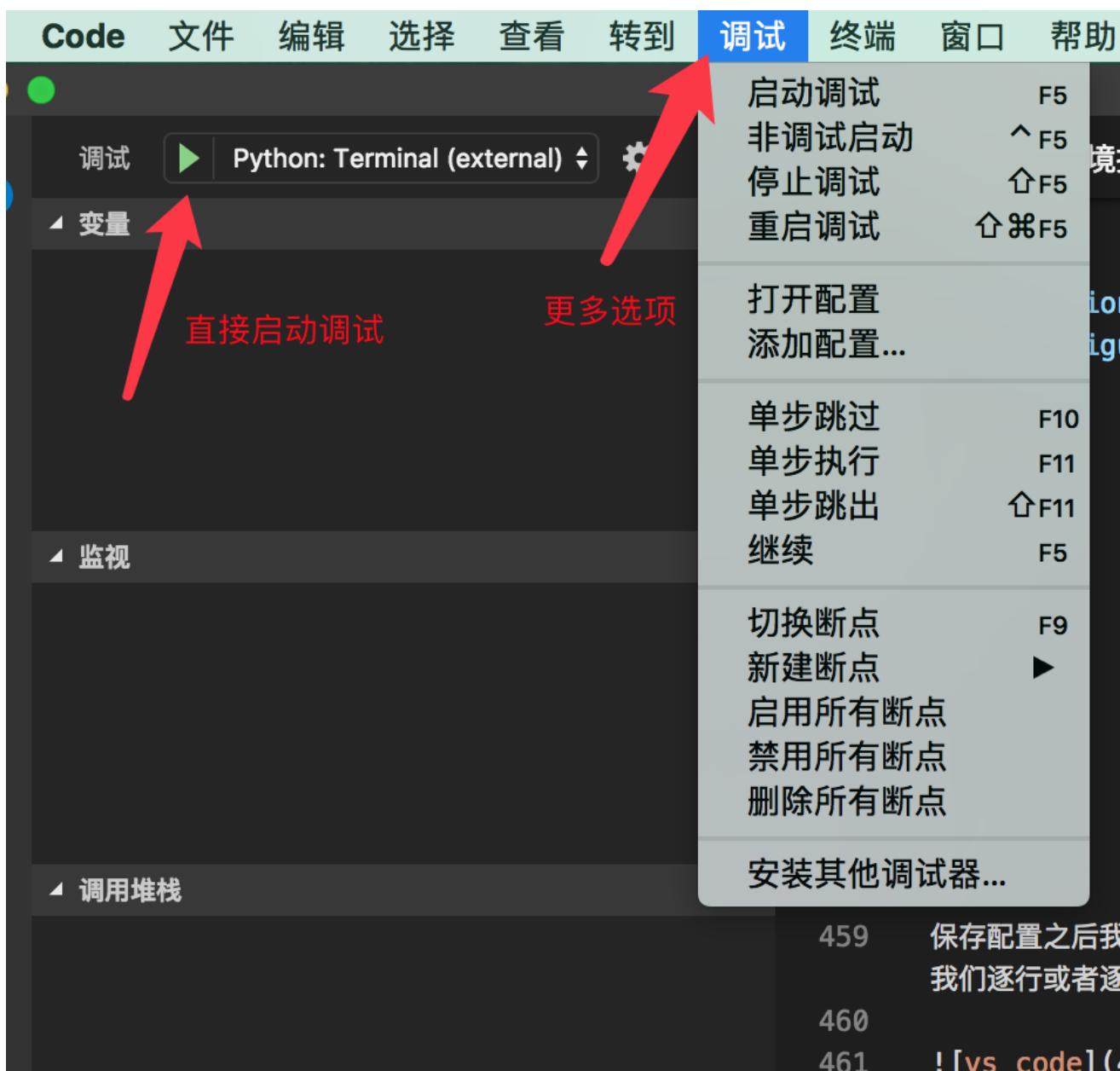


```

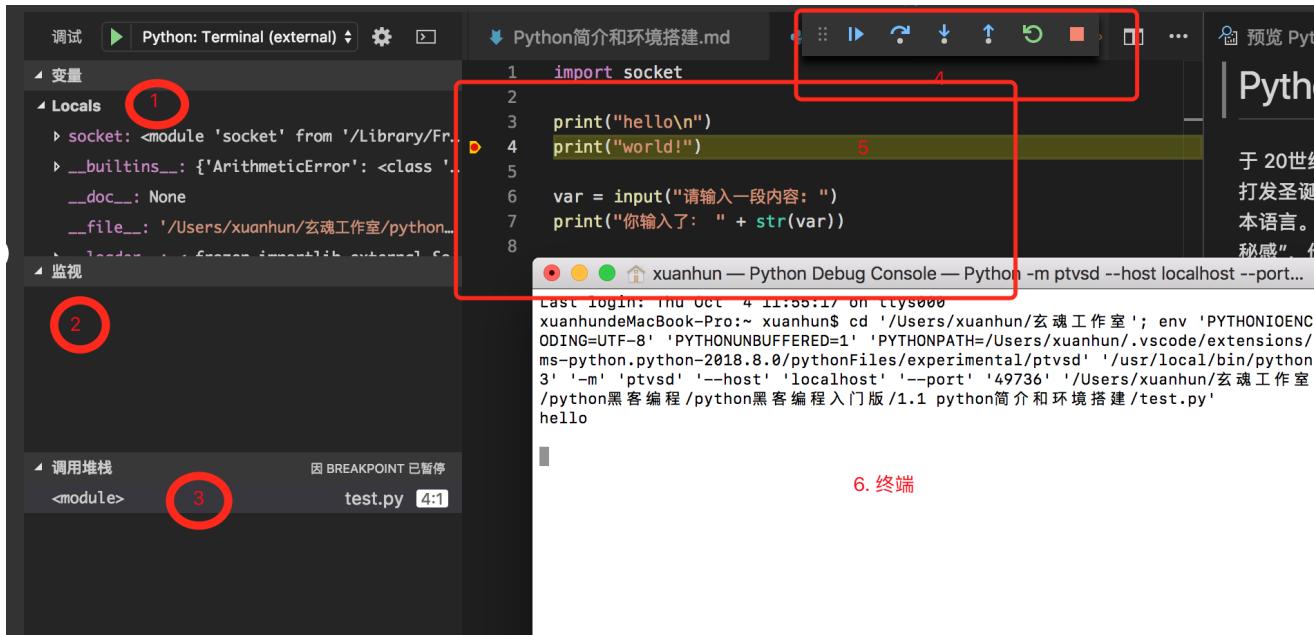
1 import socket
2
3 ● print("hello\n")
4 print("world!")
5
6 var = input("请输入一段内容：")
7 print("你输入了：" + str(var))
8

```

启动调试可以直接点击启动按钮，更多的启动选项可以在调试菜单中看到。



下面我们直接启动调试，程序首先会打开新的终端，然后会在我们的断点处断住。



如上图是启动调试，代码运行到断点处，左侧是调试信息区域，这里我们可以看到变量和堆栈信息。右侧4区域是调试控制按钮，鼠标移上去会有功能提示。5区域是代码区，此时我们可以增加和去掉断点，查看变量。6区域是打开的终端，test.py的测试代码接收用户的输入并打印，我们可以输入任意内容回车查看结果。同时在编辑器下方的调试控制台也可以显示输出内容。

VS Code 编辑器的详细使用细节这里我们就不展开了，后面教程中涉及到的应用点会单独补充。建议读者尽可能全面的阅读官方文档（<https://code.visualstudio.com/docs>）。

1.1.5 小结

本节课我们完成了Python运行环境和开发环境的搭建，完成了第一个“hello world”程序的编写和运行。为了尽可能简单的跨过第一个门槛，我省略了很多细节和原理性的讲述。不过不用担心，后面的章节会逐步将这些内容补充进来。

接下来的章节，我们会快速的过一遍Python编程基础。实践出真知，这里给几点建议：

1. 创建一个独立的文件夹，每次练习的时候使用VS Code “文件菜单-->打开” 打开该文件夹。在该文件夹下，每一个小节再独立一个文件夹存放代码和心得。
 2. 将你的代码和心得上传到github上，方便我们后面的共享和交流。

最后是本节的练习题目：

1. 完成Windows/Mac OS /Linux 至少2种操作系统的Python环境安装和测试
 2. 完成Windows/Mac OS /Linux 至少2种操作系统的VS Code安装和测试
 3. 完成“hello world”程序编写，进行调试启动、非调试启动、断点单步调试、断点逐过程调试、中断程序运行、调试过程中重启

本系列教程全部内容在星球空间内发布，并提供答疑和辅导。



星主：程序员－玄魂



星球：玄魂工作室-安全圈



○ 知识星球

长按扫码预览社群内容
和星主关系更近一步

欢迎到关注微信订阅号，交流学习中的问题和心得

