

[Toggle navigation](#) [Rex's BLOG](#)

- [Archives](#)
- [Categories](#)
- [Tags](#)
- [About](#)

# windows下安装不同版本的python

## 什么叫不同版本的python？

一般对于开发人员来说，工具的版本并不陌生。

python 大家或者熟悉或者听说过，用的人自然懂，这里也就不过多介绍了。

不过，对于linux的开发人员来说，安装不同版本的python轻而易举，经常和命令行打交道的他们可以分分钟完成不同版本的配置。

对于windows开发人员也不例外，他们一样熟悉着自己的工作环境，并且熟练的掌握着吃饭的家伙(偷笑)。

本文主要是给那些不太熟悉windows操作系统，且想要开发的用户，用以来解决一些在windows开发环境上的疑惑。

python发展历史我不想追究，随便使用个搜索引擎都能轻松了解到，这里只告诉您，python有两个重要的版本：

1. python2 历史版本，至今还有很多大项目在用，所以还没有废弃，python社区还在做兼容性维护
2. python3 革新版本，从python2跃迁而来，添加了很多新的特性，有些基础内容已经从根本上改变

## 在windows上安装不同版本python的方法

其实在windows上，安装不同版本python，笔者感觉主要有三种方法：

1. 通过[python 官网](#)，下载不同版本的python，然后通过安装配置，最后做一些手脚就可以顺利完成配置。
2. 通过 [anaconda](#)，配置你想要的版本，它自带版本管理系统，可以方便帮助隔离不同的python版本环境。
3. 通过python第三方版本管理库，完成python版本的隔离控制

本文主要介绍在windows上使用前两种方法。

## 官网下载，安装自定义

### 下载python的不同版本

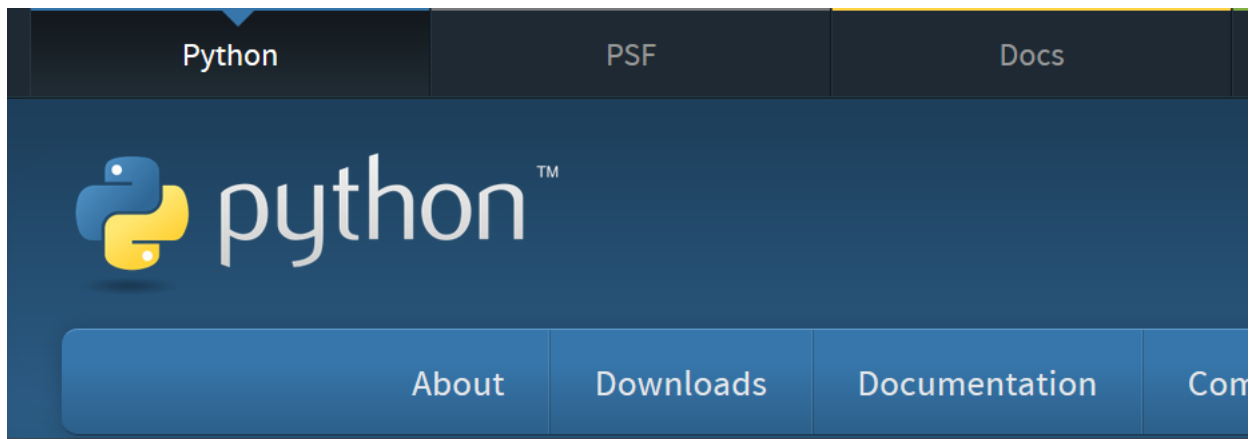
[python 官网](#)中有两个版本的下载地址，仔细找找很容易找到。

本文实例以图示版本为例，如果安装其他版本，你应该知道怎么找。

这里windows用户喜欢下载安装包，方便快捷的下载安装。

步骤如图：

step.1 找到官网下载地址



Python >>> Downloads >>> Windows

## Python Releases for Windows

- [Latest Python 3 Release - Python 3.7.4](#)
- [Latest Python 2 Release - Python 2.7.16](#)

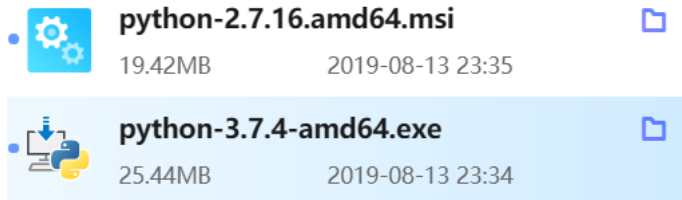
step.2选择你需要的安装包下载它

## Files

Version	Operating System	Description
<a href="#">Gzipped source tarball</a>	Source release	
<a href="#">XZ compressed source tarball</a>	Source release	
<a href="#">macOS 64-bit/32-bit installer</a>	Mac OS X	for Mac OS X 10.6 and later
<a href="#">macOS 64-bit installer</a>	Mac OS X	for OS X 10.9 and later
<a href="#">Windows help file</a>	Windows	
<a href="#">Windows x86-64 embeddable zip file</a>	Windows	for AMD64/EM64T/x64
<a href="#">Windows x86-64 executable installer</a>	Windows	for AMD64/EM64T/x64
<a href="#">Windows x86-64 web-based installer</a>	Windows	for AMD64/EM64T/x64
<a href="#">Windows x86 embeddable zip file</a>	Windows	
<a href="#">Windows x86 executable installer</a>	Windows	
<a href="#">Windows x86 web-based installer</a>	Windows	

看看自己的操作系统和电脑配置，选一个适合你的

step.3双击安装，你懂的

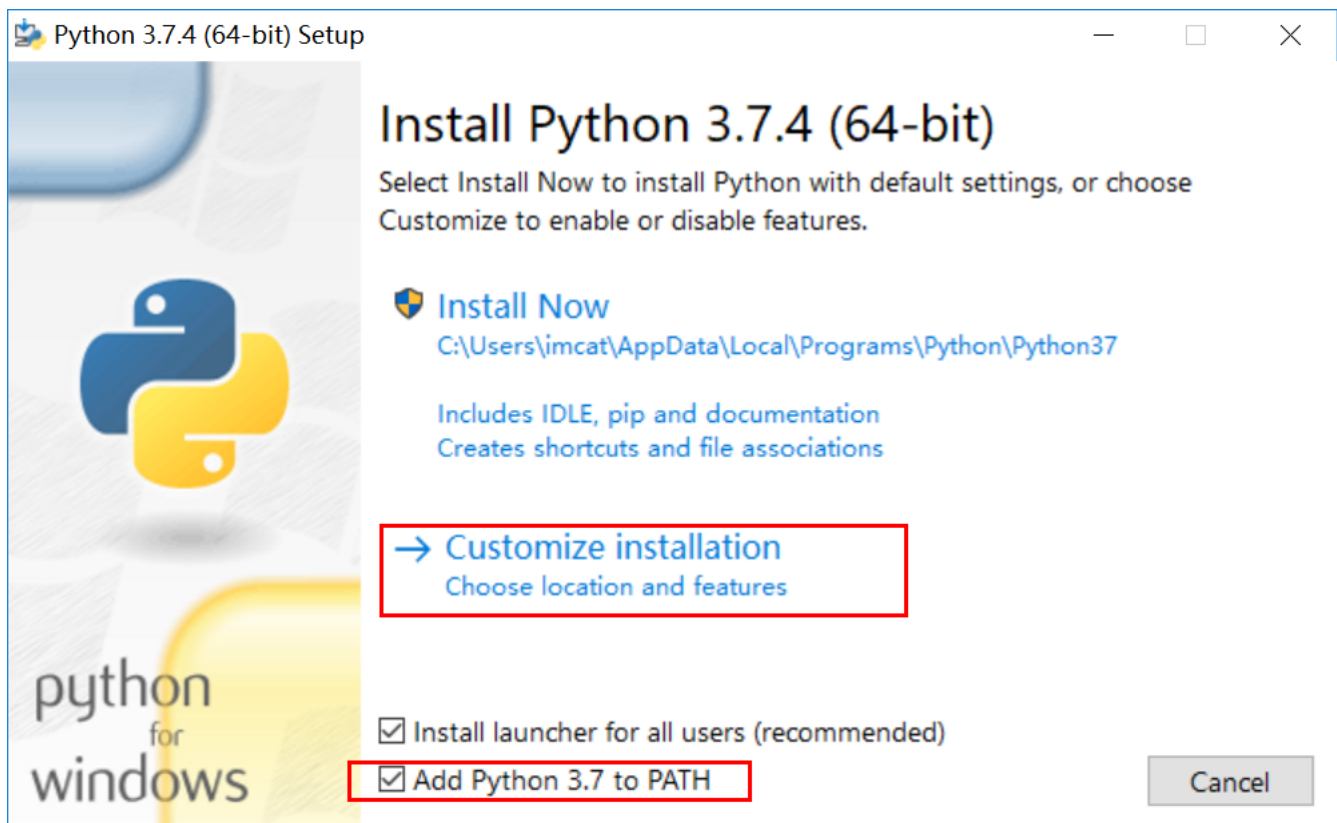


### 定制化自己的python版本环境并安装

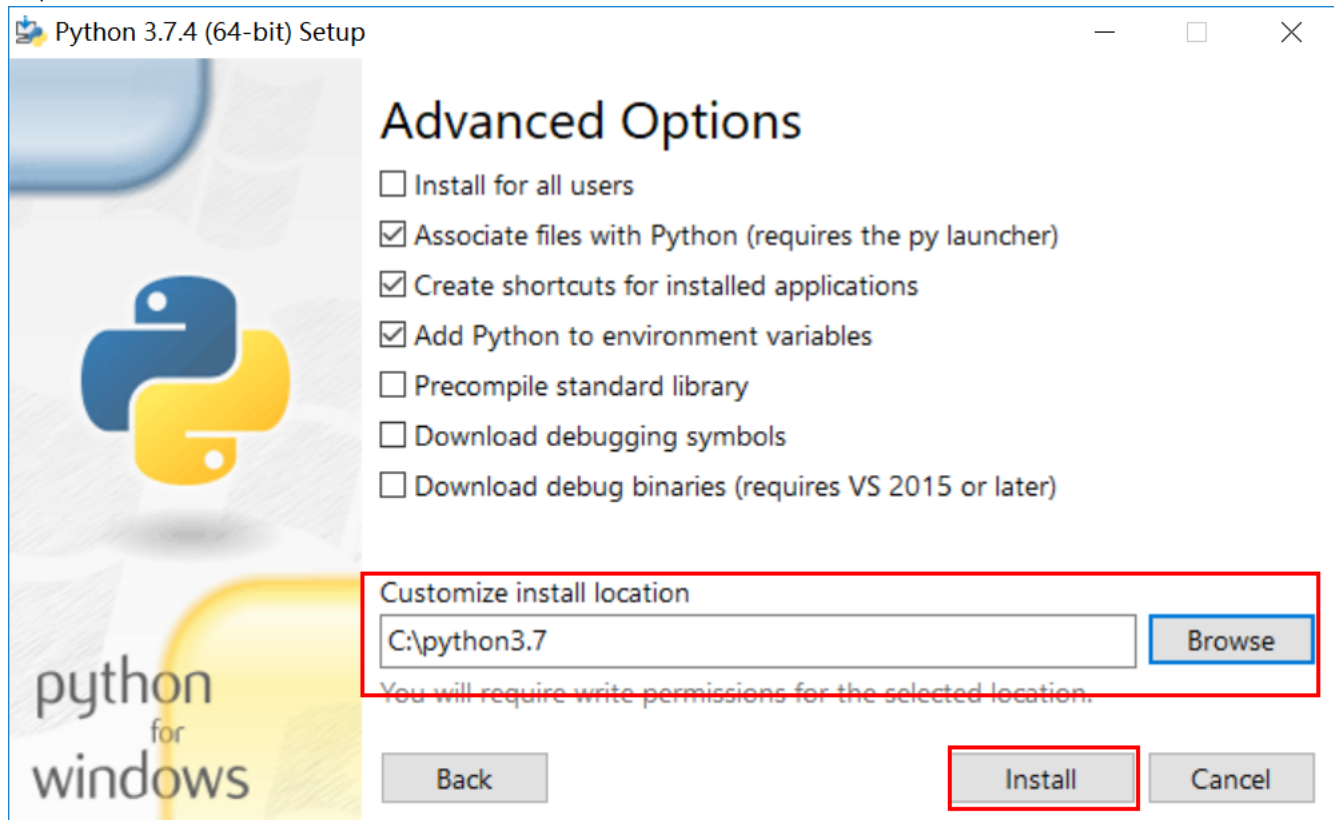
本人比较喜欢干净的工作环境，所以一般安装的python也是自己使用，并不影响其他开发者。你需要对自己安装的版本及路径了如指掌，后续做一些修改的时候也方便一些。

以python3安装为例，python2雷同。  
图示步骤如下：

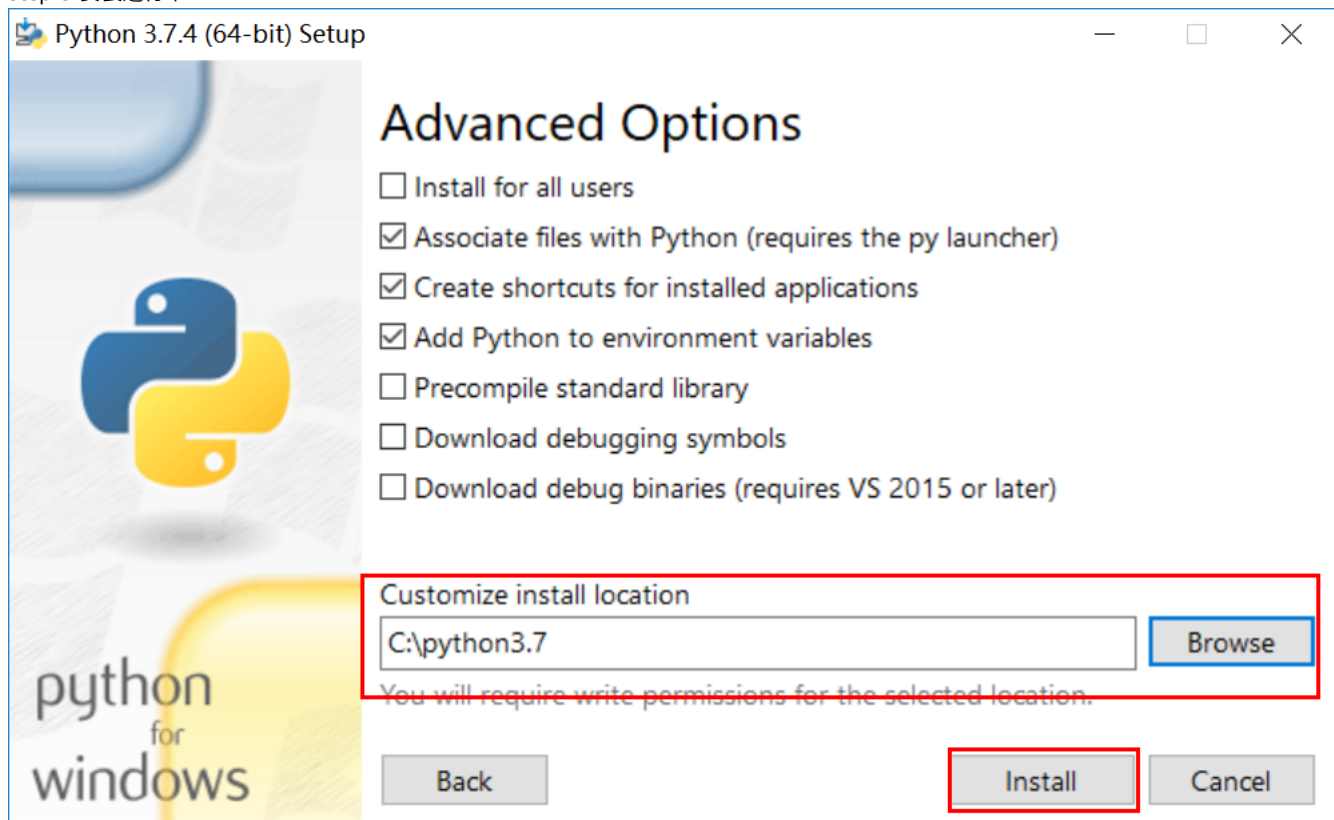
step.1 选择工具自动添加环境



step.2 手动选择你想要安装的目录

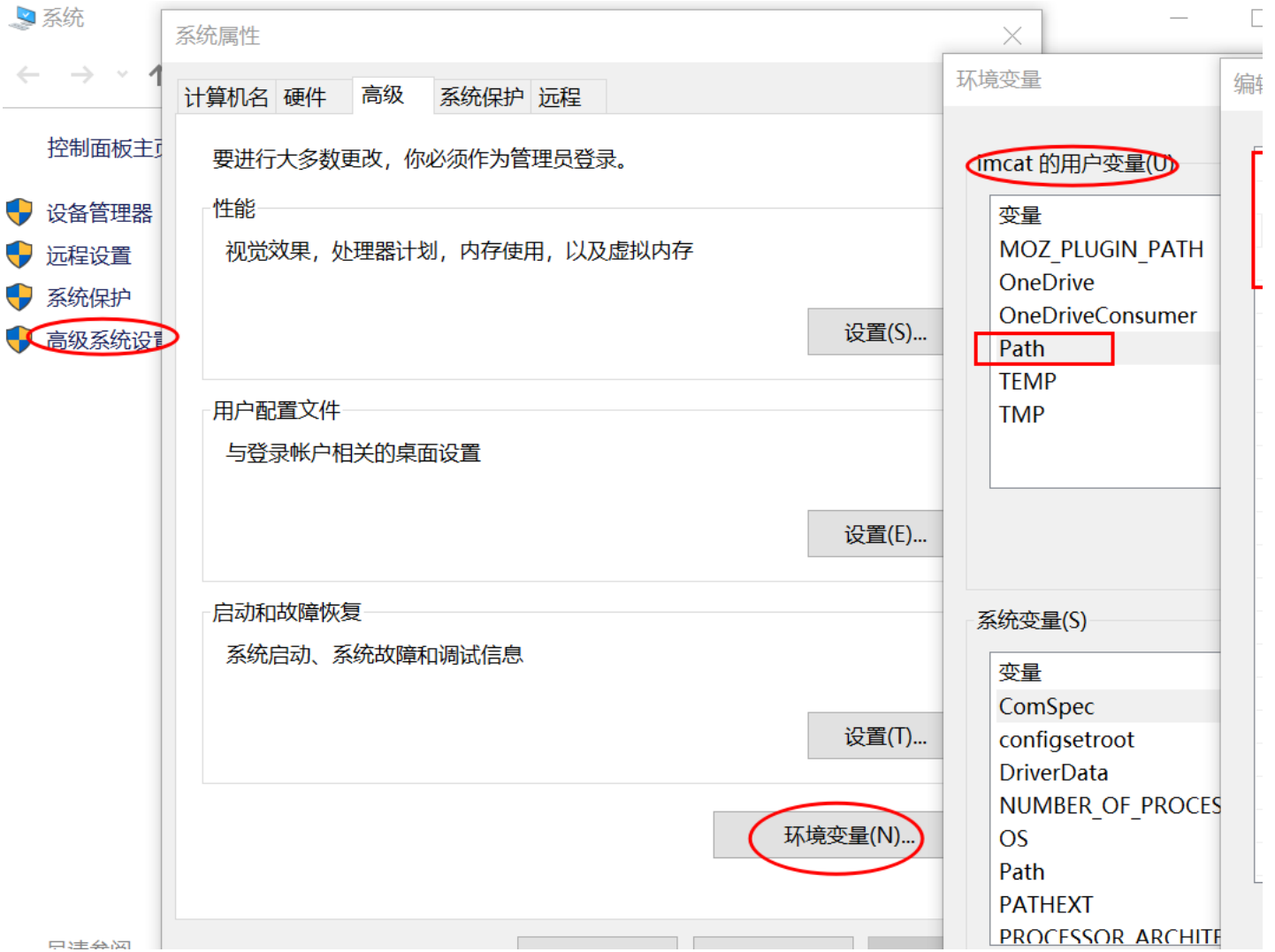


step.3 安装进行中



安装过程中，可能会提示兼容以往的 DOS 最大路径字符长度限制，请 enable。

安装完成后配置环境路径，如图：



### 为了使用命令行，修改以做兼容

修改python2安装路径下的执行文件为：python2.exe

修改python3安装路径下的执行文件为：python3.exe

修改执行入口程序的名字，操作类似如下：

此电脑 > Windows-SSD (C:) > python3.7 >

名称	修改日期	类型
DLLs	2019/8/13 23:41	文件夹
Doc	2019/8/13 23:41	文件夹
include	2019/8/13 23:41	文件夹
Lib	2019/8/13 23:41	文件夹
libs	2019/8/13 23:41	文件夹
Scripts	2019/8/13 23:41	文件夹
tcl	2019/8/13 23:41	文件夹
Tools	2019/8/13 23:41	文件夹
LICENSE.txt	2019/7/8 20:38	文本文档
NEWS.txt	2019/7/8 20:38	文本文档
python3.dll	2019/7/8 20:35	应用程序扩展
<b>python3.exe</b>	2019/7/8 20:36	应用程序
python37.dll	2019/7/8 20:35	应用程序扩展
pythonw.exe	2019/7/8 20:36	应用程序


从python.exe修改而来

当然，这里你可以复制原来的可执行文件然后修改名字也是可以的，不过python这命令在cmd中按照PATH配置顺序执行的，所以最好指定python版本。

这里有一个小技巧，如果你想要python命令，默认使用哪一个版本，那就保留该版本的可执行文件，比如：对于python3来说，保留python.exe和python3.exe，但是python2中的可执行文件只保留python2.exe。这样，我们得到了三个命令，你可以自由切换：

1. python – 表示python3
2. python3 – 表示python3
3. python2 – 表示python2

重启命令行，操作验证：

 命令提示符

```
Microsoft Windows [版本 10.0.17134.885]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\imcat>python2 --version
Python 2.7.16

C:\Users\imcat>python3 --version
Python 3.7.4

C:\Users\imcat>python --version
Python 3.7.4

C:\Users\imcat>
```

## 不同版本的pip管理器的使用

正常来说，pip在笔者下载的这两个版本中都有默认支持了。

对于老版本的python2如果没有，可以简单的在搜索引擎上找到安装python2的pip的解决方法。


由于之前我们修改了python.exe以支持想要的版本区分，不过也使得对应版本的pip.exe无法使用了。

这里提供简单的操作方法：

```
C:\Users\imcat>python2 -m pip --version
pip 18.1 from C:\Python2.7\lib\site-packages\pip (python 2.7)

C:\Users\imcat>python3 -m pip --version
pip 19.0.3 from C:\python3.7\lib\site-packages\pip (python 3.7)
```

结果类似这样：

 命令提示符

```
C:\Users\imcat>python2 -m pip --version
pip 18.1 from C:\Python2.7\lib\site-packages\pip (python 2.7)

C:\Users\imcat>python3 -m pip --version
pip 19.0.3 from C:\python3.7\lib\site-packages\pip (python 3.7)

C:\Users\imcat>_
```

这样，你就可以使用pip安装你想要的库到指定版本路径中。

## 使用anaconda自带的版本控制

anaconda 是一个比较有意思的开源项目，它对于python的版本控制非常到位，并且内置了很多有用的功能。

由于它的强大，省去了我们的配置时间，接下来简单看看如何使用 anaconda。

### anaconda 的下载与安装

去[anaconda 的官网](https://anaconda.org/)下载下来python3版本，其实我们只需要下载python3版本，也就是比较前卫的版本。之所以这样，是因为conda管理器可以帮助我们自动下载python2的内容，而且也方便切换。

下载安装步骤如图：

step.1 找到官网下载地址并下载



Windows



macOS



Linux

## Anaconda 2019.07 for Windows Installer

### Python 3.7 version

[Download](#)

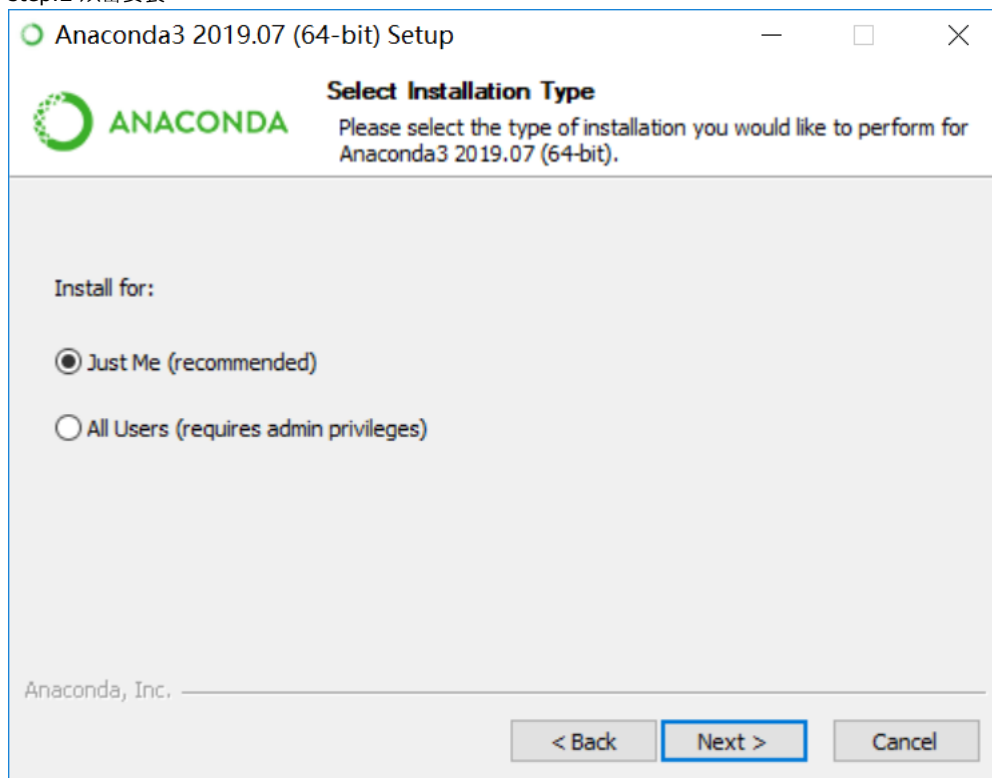
64-Bit Graphical Installer (486 MB)  
32-Bit Graphical Installer (418 MB)

### Python 2.7 version

[Download](#)

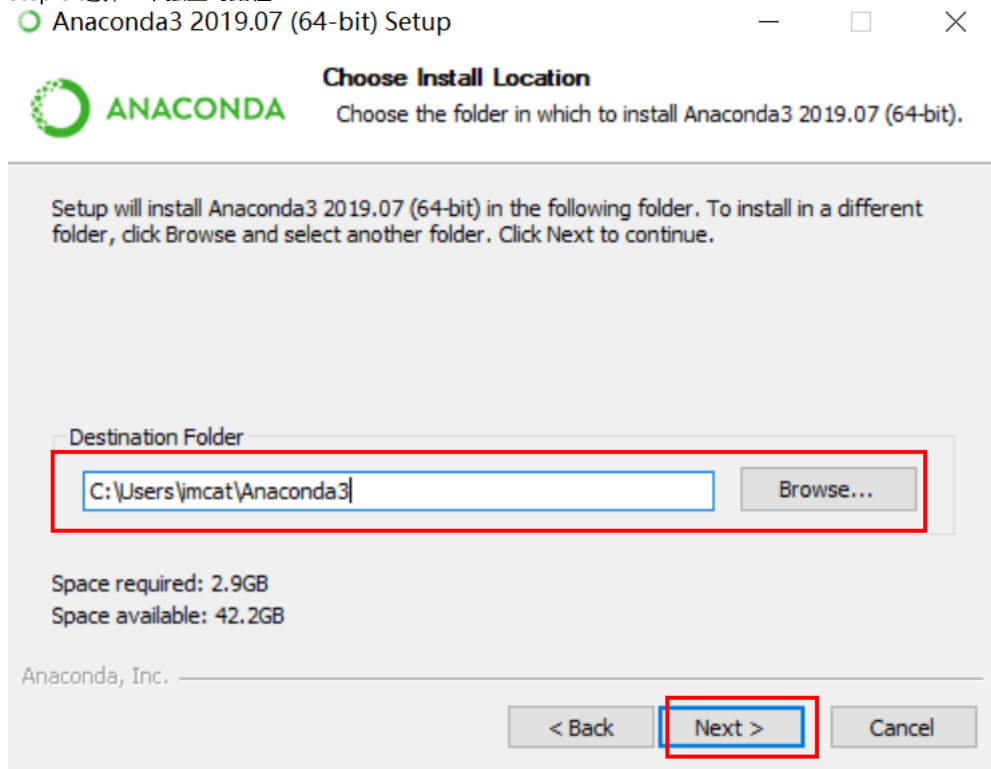
64-Bit Graphical Installer (486 MB)  
32-Bit Graphical Installer (418 MB)

step.2 双击安装

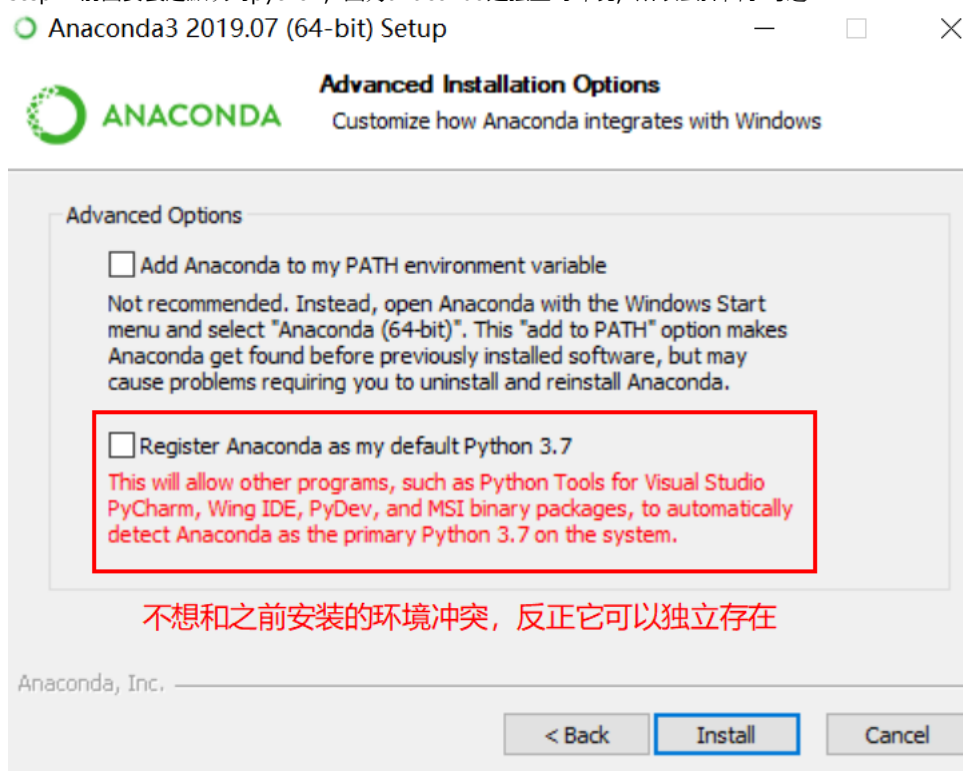




step.3 选择一个独立的路径

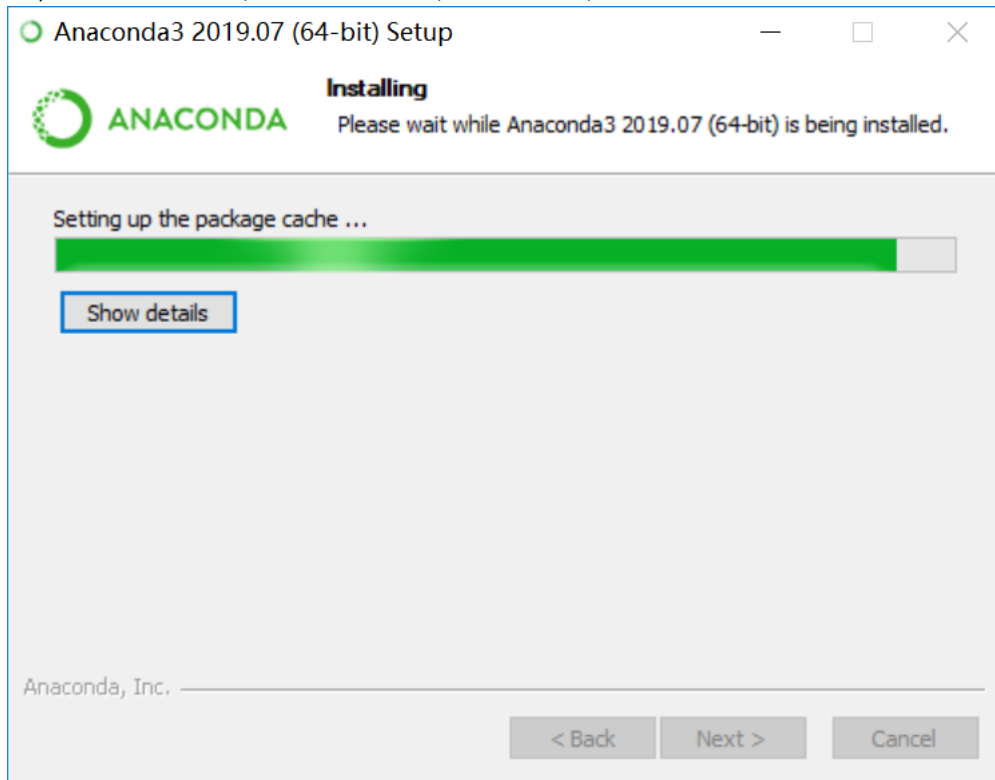


step.4 前面安装过默认的python, 因为anaconda是独立的环境, 所以去掉图示勾选



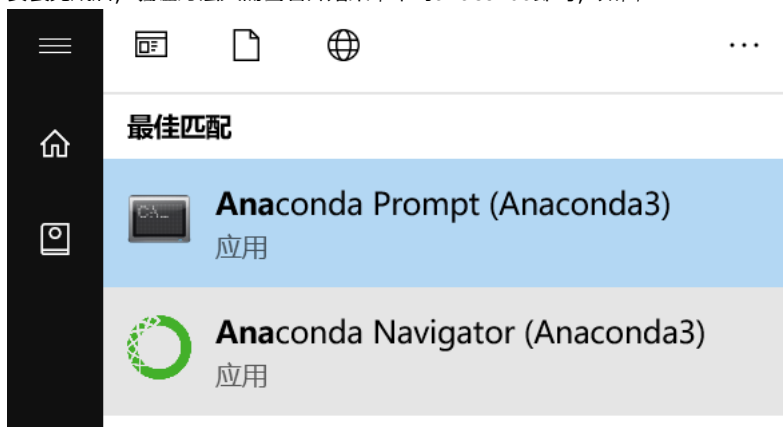


step.5安装过程有些枯燥，由于不是mini版本，需要一些时间，去喝点水吧



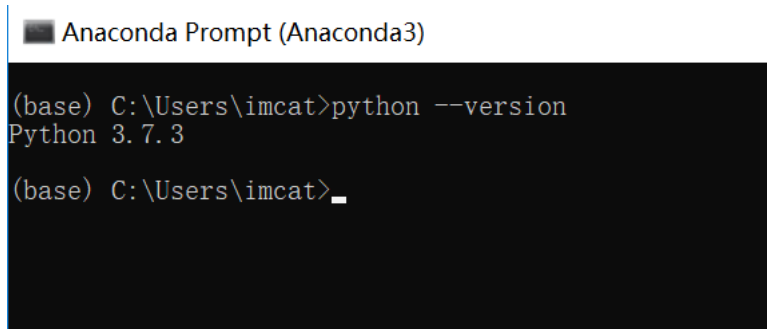
由于默认安装了一些应用，所以安装时间稍微长一些，当然你可以下载它的mini版本，这里不做赘述了。

安装完成后，验证方法只需查看开始菜单中的anaconda即可，如图：



如果你喜欢gui，那就启动gui，笔者一般喜欢使用命令行，所以使用shell接口就足够了，本文也主要介绍shell接口的相关配置，也就是anaconda prompt。

运行效果如图：



## conda 命令的简单使用

anaconda 的管理器接口是 conda 应用，从名字看就很容易理解。

这里列出一些常用的 conda 管理器的常用命令，如果想要更详细的内容，可以自行搜索官网文档。

以下命令都是在anaconda的shell中完成的，也就是anaconda prompt

```
1 # 帮助:
2 conda -h
3 # 查看conda版本:
4 conda -V
5 # 查看虚拟环境列表:
6 conda env list 或 conda info -e 或 conda info --envs
7 # 创建python虚拟环境:
8 conda create -n your_env_name python=X.X (2.7、3.6等)
9 your_env_name >> 就是你的环境名字, 识别号而已
10 python=X.X >> 就是你要安装的虚拟环境使用的真实环境版本
12 # 向指定虚拟环境中安装额外的库:
13 conda install -n your_env_name [package]
14 其实在虚拟环境中, 通过正常的安装流程也是一样可以办到的, 当然需要你切到虚拟环境下。
15 # 开启虚拟环境:
16 activate your_env_name
17 # 关闭虚拟环境:
18 deactivate
19 # 删除虚拟环境:
20 conda remove -n your_env_name (虚拟环境名称) --all
# 删除环境中的某个包:
conda remove --name your_env_name package_name
```

## 简单操作与验证

### 安装python2

```
conda create -n python27 python=2.7
```

### 查看安装好的虚拟环境

```
conda env list
```

### 切换到指定的虚拟环境

```
activate python27
```

### 验证版本

```
python --version
```

### 安装卸载包

```
python -m pip install openpyxl
python -m pip uninstall openpyxl
```

### 退出环境

```
deactive
```

简单、直接、粗暴, 很符合有强迫症的人士。

## 两种方法的利与弊

其实就方法而言, 并没有什么高低之分, 只不过针对应用场景来说, 有的方法稍微有点别手而已。

比如, 笔者在工作中就遇到的情况, 这里简单描述下:

项目同时使用python2和python3的内容, 因为一早就有的工具, 需要你去继承使用;

当然有些同学会说, 自己可以将python2的项目修改成python3兼容的, 不过这需要花费很多成本, 一般在项目中, 除非逼不得已, 一般都不会这么做, 而是一起使用python3和python2, 这个时候, 貌似纯粹的虚拟环境就有点劣势, 所以配置干净的系统环境, 也就是使用第一种方法, 就成了符合当前策略的主要方案。

而一般一个项目的开发可能支持不同的python版本, 设计之初, 就可能考虑到不同版本的向后兼容性, 需要不停的切换版本验证, 这种时候, 虚拟环境的特长就显露出来了。

当然, 这也是笔者个人所思, 百家想法, 各有观点, 这里就不过多的讨论了。

好了, 终于在windows上, 不断的卸载和安装, 完成了这篇文章。

^\_^ 祝贺下(拍手)。

[Prev](#) [Home](#) [Next](#)

## Comments

2019-08-12

- 
- [python2](#)
- [install\\_python1](#)

