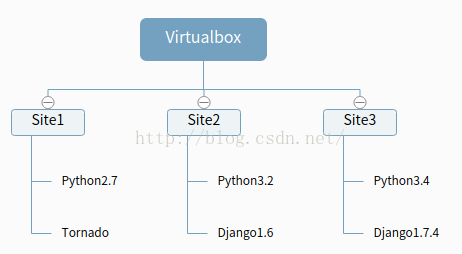
VirtualEnv用于在一台机器上创建多个独立的[**Python**](http://lib.csdn.net/base/python)运行环境，VirtualEnvWrapper为前者提供了一些便利的命令行上的封装。

Virtualenv是一个非常好的virtual python environment builder，他最大的好处是，可以让每一个python项目单独使用一个环境，而不会影响python系统环境，也不会影响其他项目的环境。

Virtualenv可用于创建独立的Python环境，在这些环境里面可以选择不同的Python版本或者不同的Packages，并且可以在没有root权限的情况下在环境里安装新套件，互相不会产生任何的影响。



**为什么要用virtualenv**

- 隔离项目之间的第三方包依赖，如A项目依赖django1.2.5，B项目依赖django1.3。

- 为部署应用提供方便，把开发环境的虚拟环境打包到生产环境即可,不需要在服务器上再折腾一翻。在服务器上都不用安装virtualenv，直接将virtualenv创建的目录拷贝到服务器，修改路径，进行虚拟环境迁移就可以用了。

- 还可以用在没有root权限的python环境配置上，如果没有root权限，可以先自己搞一个virtualenv，再在virtualenv中使用pip安装。（系统中没有pip，并且也没有root权限使用sudo apt-get安装）

**安装的库的位置**

env/Lib/site-packages/目录里，而不是在系统的python的Lib/site-packages目录里，这样你就知道为什么虚拟环境是分开的了吧。

**Note:**virtualenv 创建的虚拟环境与主机的 Python 环境完全无关，你主机配置的库不能在 virtualenv 中直接使用。你需要在虚拟环境中利用 pip install 再次安装配置后才能使用。

[皮皮Blog](http://blog.csdn.net/pipisorry)

**Virtualenv的安装**

**pip安装virtualenv**

pip install virtualenv   #py2安装

pip3 install virtualenv   #py3安装，这样用virtualenv创建的virtualenv默认python版本是py3

这时python3.4.2\Lib\site-packages目录下会出现两个新子目录virtualenv\_support、virtualenv-13.1.0.dist-info

[[pip](https://pip.pypa.io/en/stable/)]

**安装Virtualenvwrapper**

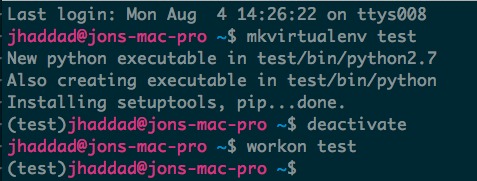
[可能需要掌握一些常见的virtualenvwrapper的命令，方便管理Python的版本，链接在：   
<http://www.doughellmann.com/docs/virtualenvwrapper/command_ref.html>]

安装完以后，创建一个虚拟环境，然后在安装virtualenvwrapper  
**virtualenv ENV**#ENV 为环境的名字，可以任意设置，其实就是一个文件夹，在home下的用户名文件夹下可以找到。  
**source ENV/bin/activate**    #这样进进入了virtualenv的虚拟开发环境。  
进入虚拟环境以后命令行最开始的地方会显示（ENV），代表已经进入了这个环境，然后就可以安装virtualenvwrapper和Django了

输入命令行**pip install virtualenvwrapper**  
这里可以不用sudo，因为在virtualenv里，不用管理权限也算是很方便的设计之一。

virtualenvwrapper安装后，它会把virtualenv列为依赖包，所以会自动安装。

打开一个新的shell，输入mkvirtualenv test 。如果你打开另外一个shell，则你就不在这个virtualenv中了，你可以通过workon test 来启动。如果你的工作完成了，可以使用deactivate 来停用。



[皮皮Blog](http://blog.csdn.net/pipisorry)

**创建虚拟环境并进入使用**

**使用virtualenv默认python版本创建虚拟环境**

**virtualenv --no-site-packages ubuntu\_env**

就可以在当前目录创建一个env目录(虚拟环境名称，这个文件夹就是保存 Python 虚拟环境)，你会注意到，virtualenv会把python，setuptools和pip给你安装上。

**自定义python版本创建虚拟环境**

1. 安装需要版本的python  
2. 指定virtualenv中的python版本  
virtualenv --no-site-packages --python=python2.7 env

**Note**:

1. 创建virtualenv虚拟环境之前，系统中必须要安装有对应版本的python，并且卸载之后当前虚拟环境就无效了。系统中可以同时存在python2和python3，通过环境变量中的系统变量path（不是用户变量）控制cmd或者系统中使用哪个版本的python，哪个版本的路径在前面就优先使用哪个版本。

2. –no-site-packages表示不包括系统全局的Python安装包，这样会更令环境更干净

2. –python=python2.7指定Python的版本未系统已经安装了的Python2.7

3. env是建立的虚拟环境名称

4 .没有安装python2.7或者使用命令virtualenv --no-site-packages --python=python2.7 env会出现The executable python does notexist 错误

**进入虚拟环境并激活**

[**Linux**](http://lib.csdn.net/base/linux)**(mac os):**

**. ubuntu\_env/bin/activate**

pika:/media/pika/files/mine/python\_workspace/NLP$d env   
Going to /media/pika/files/mine/ENV  
pika:/media/pika/files/mine/ENV$. ubuntu\_env/bin/activate  
(ubuntu\_env) pika:/media/pika/files/mine/ENV$d nlp   
Going to /media/pika/files/mine/python\_workspace/NLP  
(ubuntu\_env) pika:/media/pika/files/mine/python\_workspace/NLP$python  
Python 3.4.3 (default, Oct 14 2015, 20:28:29)

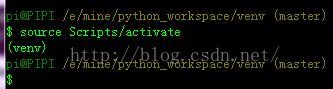
**windows:**

env > Script\activate  (or Script\activate.bat)

**http://img.blog.csdn.net/20150723191144999?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center**

**mingw(**[**Git**](http://lib.csdn.net/base/git)**):**

venv >source Scripts/activate

****

这时会发现，在命令行前面，会多出(env)出来，表示你已经进入了虚拟机了。现在你可以使用pip install xxx来安装你想要的库了。

Note: activate.bat脚本会设置控制台环境变量，使得该控制台以后调用的python命令会执行虚拟环境中的python。virtual env激活后，在任意目录输入python, ipython(如果安装了)都会调用virtual env中的命令。

**退出虚拟环境**

(ubuntu\_env) pika:/media/pika/files/mine/ENV/ubuntu\_env$deactivate   
pika:/media/pika/files/mine/ENV/ubuntu\_env$

Note: deactivate命令可能在virtualenv安装目录下。

直接在该环境中使用deactivate命令即可退出

http://img.blog.csdn.net/20150723191252433?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

**Note**: deactivate.bat是还原控制台环境变量设置

**删除虚拟环境**

rm -r venv

直接删除虚拟环境所在的文件夹venv就删除了我们创建的venv虚拟环境

**linux下virtualenv其它命令**

列出虚拟环境列表

workon

也可以使用lsvirtualenv

新建虚拟环境

mkvirtualenv [虚拟环境名称]

启动/切换虚拟环境

workon [虚拟环境名称]

删除虚拟环境

rmvirtualenv [虚拟环境名称]

[皮皮Blog](http://blog.csdn.net/pipisorry)

**虚拟环境迁移**

当需要将虚拟环境env转移到同一台电脑的另一个目录下时

或者当需要将虚拟环境env1迁移或复制到另一个虚拟环境（可能不在同一台机器上）env2时，首先仍然需要在目的机器上安装pip和virtualenv，然后采用以下方法之一安装其他的package：

1.直接将env1里的文件全部复制到env2里，然后修改涉及路径的文件。此种方法可能正常使用，但显然不是好办法(不过网络不好的时候lz就是用的这个)。

修改的路径主要是虚拟环境ubuntu\_env/bin目录下的文件执行程序的路径：如active, pip, ipython等等（几乎所有文件的路径，不过有的命令你不用的可以不改）。

如将虚拟环境ubuntu\_env/bin/active中的路径修改一下：

其中的VIRTUAL\_ENV路径其实最好修改成现在的绝对路径VIRTUAL\_ENV="/media/pika/files/mine/ENV/ubuntu\_env"；

将虚拟环境ubuntu\_env/bin/pip和pip3和pip3.4中的路径修改成绝对路径：#!/home/pipi/ENV/ubuntu\_env/bin/python3

注意最好不要将路径修改成相对路径。如果修改的ipython执行路径为相对路径#!./bin/python3，是相对这个ipython.py执行的路径，激活virtual env后如果在另一个目录下输入python就会出错找不到或者直接调用系统中的python了，而不是virtual env中的python，而原始的默认路径使用绝对路径就可以在virtual env激活后在任意目录都可以执行ipython而不出错！

2. 使用requirements.txt

进入原虚拟环境env1，然后执行pip freeze > requirements.txt将包依赖信息保存在requirements.txt文件中。

最好手动调整一下顺序，比如numpy和scipy要在matplotlib前面安装；另外如果想安装最新版本的，再将后面的版本号==1.9.1什么的删除。

然后进入目的虚拟环境env2，执行pip install -r requirements.txt，pip就会自动从网上下载并安装所有包。

虚拟环境env2如果是env1的拷贝，最好先pip uninstall -ry requirements.txt，再pip install -r requirements.txt

3.pip默认会从pypi服务器（http://pypi.python.org/simple）下载包的安装文件，如果目的机器无法连外网，则可以采用以下办法：  
3.1搭建自己的pypi服务器。专业的，可以使用第三方的软件包来搭建一个完整的pypi镜像服务器，参考http://www.worldhello[**.NET**](http://lib.csdn.net/base/dotnet)/2011/03/14/2357.html。更快速的方法只需要一条命令python -m SimpleHTTPServer即可完成搭建服务器，具体的目录结构可参考原pypi服务器，简而言之，就是把安装文件打包放入目录即可。搭建好服务器之后，在目的虚拟环境中，就可以使用pip来安装了，命令如：pip install -i http://127.0.0.1:8000/ -r requirements.txt  
3.2如果你实在不想搭建pypi服务器，也有办法。首先将所有包的安装文件下载下来，可以手动下载，也可以使用pip，如pip install -d /path/to/save/ -r requirements.txt，然后自己修改requirements.txt文件，将每一行改成对应的包的安装文件的路径。最后在目的虚拟环境中使用pip安装，如pip install -r requirements.txt即可。

3.3还有一种途径，就是pip提供的bundle选项。首先执行pip bundle MyEnv.pybundle -r requirements.txt，将生成一个MyEnv.pybundle文件，该文件夹包含所有包的安装文件（注意必须后缀名必须是.pybundle），默认是重新从pypi服务器下载安装文件的，如果愿意，也可以利用3.1中的方法，指定本地的pypi服务器。然后在目的虚拟环境中执行pip install MyEnv.pybundle即可

所以说，lz还是更喜欢[**Docker**](http://lib.csdn.net/base/docker)中配置的python开发环境。

**虚拟环境迁移出错**

**方法1虚拟环境迁移后包引入出错**

from scipy import stats

ImportError: liblapack.so.3: cannot open shared object

这时只需要pip install scipy重新安装一下就ok了。

ImportError: libBLT.2.4.so.8.6: cannot open shared object file: No such file or directory, please install the python3-tk package

也要重新安装matplotlib

所以最好使用方法2迁移。

[皮皮Blog](http://blog.csdn.net/pipisorry)

**Virtualenv中安装python拓展包**

pip安装时，使用命令pip, pip3, pip3.4都可以，因为virtualenv中安装的如果是py3则pip对应的就是pip3。

注意，虚拟环境中安装python拓展包需要先如上进入虚拟环境，不然就安装在系统的python路径下了。

**virtualenv从requirements.txt中安装python库**

进入原虚拟环境env1，然后执行pip freeze > requirements.txt将包依赖信息保存在requirements.txt文件中。然后进入目的虚拟环境env2，执行pip install -r requirements.txt，pip就会自动从网上下载并安装所有包。

**requirements.txt:**

numpy

pandas

**virtualenv中安装numpy**

通过pip install numpy直接安装numpy！

**virtualenv中安装scipy和matplotlib**

You may have to install some extra non-Python dependencies using apt-get.在安装scipy前，要先在系统中安装scipy的非python依赖，好像是fortran的。ubuntu16.04出来时已经不用了？

Building Matplotlib requires libpng (and freetype, as well) which isn't a python library, so pipdoesn't handle installing it (or freetype).

sudo apt-get build-dep python-scipy

sudo apt-get -y build-dep matplotlib

(virtualenv) **pip install scipy**  
(virtualenv) **pip install matplotlib**

[[Install python-scipy in a virtualenv](http://www.codegur.net/27732819/install-python-scipy-in-a-virtualenv)]

[[How to install matplotlib using virtualenv on Ubuntu](http://blog.pangyanhan.com/posts/2015-07-25-how-to-install-matplotlib-using-virtualenv-on-ubuntu.html)]

[[Installing](http://matplotlib.org/users/installing.html#build-requirements)]

**windows下virtualenv中安装pywin32**

virtualenv中不能直接用pip install pywin32，否则会出错：Couldnot find any downloads that satisfy the requirement.  pywin32 No distributions at all foundfor pywin32.

**Solution1**

在[http://sourceforge.net/projects/pywin32/files/](http://sourceforge.net/projects/pywin32/files/pywin32/) 找到你要的exe文件下载地址

激活virtualenv

运行easy\_install http://PATH.TO/EXE/DOWNLOAD

**Note**: easy\_install 太老的话可能出错：error: c:\users\blah\appdata\local\temp\easy\_install-ibkzv7\pywin32-214.win32-py2.6.exe is not a valid distutils Windows .exe

**Solution2**

如果下载不下来也可以这样

[http://sourceforge.net/projects/pywin32/files/](http://sourceforge.net/projects/pywin32/files/pywin32/) 中下载exe文件到本地

激活virtualenv

运行easy\_install DOWNLOADED\_FILE.exe   e.g. easy\_install D:\Downloads\Programming\Python\python3\pywin32-219.win-amd64-py3.4.exe

**Solution3（亲测有效）**

系统中安装pywin32后，将pywin32.pth复制到virtualenv site-packages并编辑文件指定相关路径。[[PyWin32 and virtualenv](http://www.activestate.com/blog/2010/12/tips-activepython-pywin32-and-virtualenv)]

> copy D:\python3.4.2\Lib\site-packages\pywin32.pth E:\mine\python\_workspace\CrawlerEnv\Lib\site-packages\pywin32.pth

查处文件内容more E:\mine\python\_workspace\CrawlerEnv\Lib\site-packages\pywin32.pth'

# .pth file for the PyWin32 extensions  
win32  
win32\lib  
Pythonwin

将其自改为：

D:\python3.4.2\Lib\site-packages\win32  
D:\python3.4.2\Lib\site-packages\win32\lib  
D:\python3.4.2\Lib\site-packages\Pythonwin

这样就可以成功引入，并且可以在pycharm中使用：



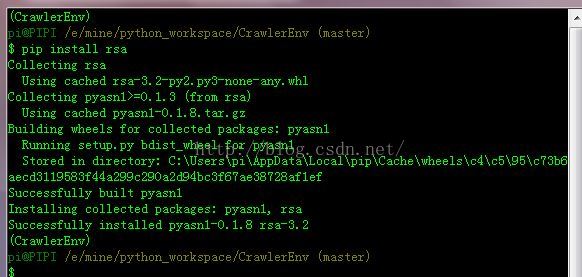
[[How to install win32com module in a virtualenv?](http://stackoverflow.com/questions/14913607/how-to-install-win32com-module-in-a-virtualenv)]

[[How can I use pywin32 with a virtualenv without having to include the host environment's site-packages folder?](http://stackoverflow.com/questions/1830304/how-can-i-use-pywin32-with-a-virtualenv-without-having-to-include-the-host-envir)]

[[How to install pywin32 in virtualenv with python 2.7 64 bit?](http://stackoverflow.com/questions/8078216/how-to-install-pywin32-in-virtualenv-with-python-2-7-64-bit)]

**virtualenv中安装rsae**

python版本是python3



**Note**: lz不知道为嘛要先重启电脑才能安装成功，否则报错could not find the version that satisfies the requirement rsa

**查看virtualenv中已经安装的python包列表**

(ubuntu\_env) pika:/media/pika/files/mine/python\_workspace/ubuntu\_env$**pip list**  
numpy (1.10.4)  
pip (8.1.1)  
py4j (0.9.2)  
setuptools (20.3.1)  
sh (1.11)  
wheel (0.29.0)  
(ubuntu\_env) pika:/media/pika/files/mine/python\_workspace/ubuntu\_env$**pip freeze**  
numpy==1.10.4  
py4j==0.9.2  
scipy==0.17.0  
sh==1.11

**Note**: virtualenv中的pip应该就是pip3默认的。两种方式还是有一丁丁不同的。

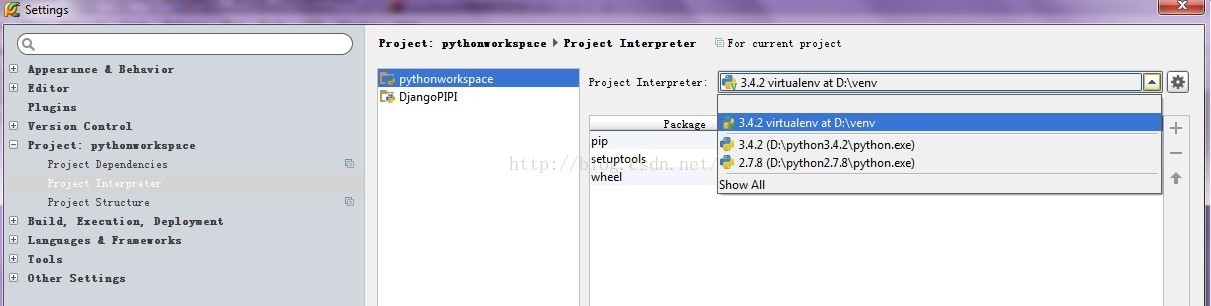
[皮皮Blog](http://blog.csdn.net/pipisorry)

**pycharm项目配置虚拟环境**

在pycharm中设置项目在virtualenv中运行

首先virtualenv venv创建一个虚拟环境并安装好相关运行环境

打开pycharm



下拉框中可以选择需要的运行环境

如果找不到创建的虚拟环境，可以在右边设置按钮中添加一个新的python interpreter， add local选择创建的虚拟环境venv中的python解释器（如D:\venv\Scripts\python.exe）就可以了

[皮皮Blog](http://blog.csdn.net/pipisorry)

**多个不同python版本共存**

我们可以用许多方法让不同的 Python 版本在系统上共存，例如在 OS X 上，如果使用官方提供的 DMG 版本安装，那么自带的 python2 和新安装的 python3 是可以共存的。python3 可以使用 python3 来调用，甚至 pip 都可以使用 pip3 来调用。

**1. 自己配置**

同时安装python2和python3，然后在系统变量中修改两个的顺序就可以设置系统中优先使用哪个版本的python

如果是用anaconda安装的python2，再安装python3就必须在python3安装程序中设置不安装pip，否则安装程序会出错，无法安装成功（所以小编从来都是需要什么拓展包才安装，从不用anaconda一次安装那么多没用的东西）。安装python3成功后，再下载运行[get-pip.py](http://download.csdn.net/detail/pipisorry/8996713)文件安装pip就ok了

**2. pyenv**

如果有很多小版本需要共存，[pyenv](https://github.com/yyuu/pyenv) 用来解决这类问题。它可以安装、卸载、编译、管理多个 python 版本，并随时将其中一个设置为工作环境。But pyenv 不支持 Windows 系统。[[Python多版本共存之pyenv](http://seisman.info/python-pyenv.html)]

**3. pywin**

Windows 上有一个 pyenv 的替代品，是 [pywin](https://github.com/davidmarble/pywin) 。它用来在多个安装的 Python 版本之间进行切换（就是在windowns中已经安装了多个版本的python），也支持 [MSYS/MINGW32](http://zengrong.net/post/1557.htm) 。

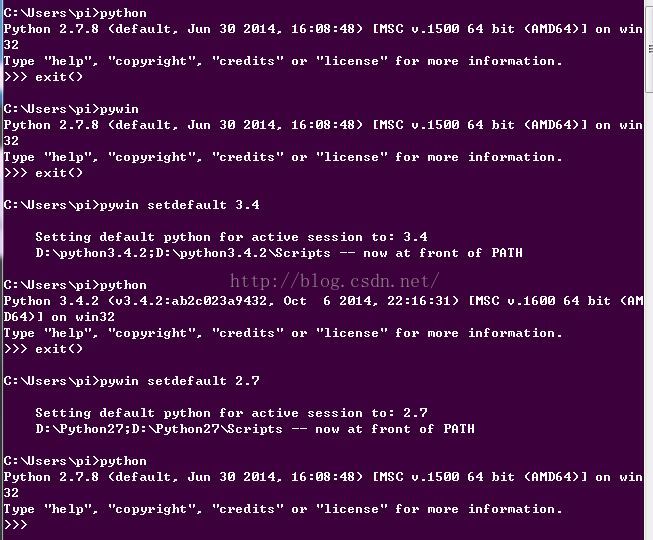
**安装和使用：**

pip install pywin

C:\>**pywin genlaunchers**  
Search for python installations and create batch files in the same directory where pywin is installed.

C:\>**pywin**# launch first python found in PATH  
C:\>**pywin -2.7**# launch python 2.7  
C:\>**pywin setdefault 3.4**  
    Setting default python for active session to: 3.4  
    C:\Python34;C:\Python34\Scripts -- now at front of PATH   #相当于1中的自己配置

C:\>**pywin -3.2 test.py**  # launch test.py with python 3.2  
  C:\Python32



**pywin启动python版本的顺序：**Any version specified after a #! in the first 2 lines of the source. The interpreter will be invoked with any additional parameters.examples:  
#! python3.3  
#! /usr/bin/python2.7 -v  
If the environment variable VIRTUAL\_ENV is set, use that virtualenv's python.exe.  
If the environment variable PYTHONHOME is set, use its python.exe.  
If none of the above, fall back to the first python.exe found on the path.  
**4. Python Launcher for Windows**

Python 从3.3版本开始（又是3.3？），在 Windows 系统中自带了一个 [py.exe](https://docs.python.org/3/using/windows.html#launcher) 启动工具。如果你是使用 Python.org 官网下载的安装包安装的 Python 3.3（或更新版本）环境，那么可以直接在命令提示符中使用这个工具。py 可以打开默认的 python 提示符； py -2.7 和 py -3 打开对应的 Python 版本。

[[Python 虚拟环境](http://zengrong.net/post/2167.htm)]

[[windows下多个python版本共存](http://jingyan.baidu.com/article/b87fe19e94ca95521935686e.html)]

[皮皮Blog](http://blog.csdn.net/pipisorry)

**另两种解决方案**

**Anaconda**

Conda 是 Continuum 公司发布的 Anaconda 里边配备的一个包管理器。Conda 让你更加方便地安装和管理各种扩展包和运行环境，同时支持 Windows，Mac OS X 以及 Linux。

**安装**

下载[anaconda](https://www.continuum.io/downloads#_unix)Python 3版本

~/ENV$ wget http://repo.continuum.io/archive/Anaconda3-4.1.1-Linux-x86\_64.sh

bash Anaconda3-4.1.1-Linux-x86\_64.sh

运行安装时注意不要一直enter，要设置一下anaconda安装目录及是否加入用户默认.bashrc中。

#PREFIX=~/opt/anaconda3

Do you wish the installer to prepend the Anaconda3 install location to PATH in your /home/pipi/.bashrc ? [yes|no]  
[no] >>> yes

重启terminal就可以使用conda安装了

这样安装的python就是用户独立的！linux下不同用户输入python都是对应自己安装的anaconda，pip也是。

**使用**

**使用conda安装python拓展包**

Conda支持多种配置选项。修改这些选项最简单的方法就是使用conda config命令。

Conda 通过使用 SAT 求解器加上一个伪布尔约束，来解决包之间的依赖关系。当 Conda 安装扩展包时，它会尝试查找和这个包结合在一起能够使用的那些包的最新版本。更新全部包，就是尝试安装每个包，让 SAT 求解器找到最新可用的版本。conda update –all 可以很容易的实现这一功能。例如，如果你现在安装了 Python 2.7.4, Nunpy 1.8.0, 和 SciPy 0.14.0, conda update –all 就和 conda install “python>=2.7.4, <3” “numpy>=1.8.0” “scipy>=0.14.0” 的功能一样（除此之外还包括一些Python的依赖关系，比如 readline 和 OpenSSL）。值得注意的是 conda update –all 不会把 Python 2 升级到 Python 3 。

如果你想把某一个包升级到比 anaconda 元包指定的版本，你可以通过 conda remove anaconda 来移除它。（这会移除元包，里面不包含任何代码。）

**conda list –export 和 conda create –file**

使用 Conda 可以很简单地复制环境。

conda list --export conda\_list.txt 可以导出所有你已经安装好的包，包括版本和编译字符。你可以把这些保存在文件里。

同时使用 conda install --file conda\_list.txt 或者 conda create --file 来安装同样的包。

**conda clean**

使用一段时间之后， Conda 会占用很多硬盘空间，这是因为它不会自动删除一些没用的包。

你可以通过 conda clean -p 来删除这些没用的包。这个命令会检查哪些包没有在包缓存中被硬链接到其他任何地方，并删除它们。注意，如果你是通过 symlink 等方式或通过一个单独的文件系统安装的包，你就没有办法使用这个命令删除它们，因为它检测不到它们的存在。

Conda 也会保存所有下载下来的 tar 包。它们只是为了缓存才保存下来的，是可以被删除的。你可以通过 conda clean -t删除它们。

**钉包（Pinning Packages）**

默认情况下，Conda 会在环境中安装一个包的最新版本。但是，有时候你可能会想保留某一个旧版本的包，哪怕你之后安装的包要依赖这个包的新版本（Conda 默认会升级你已经安装的包的依赖包）。

例如，假设你在你的环境里已经安装了 SciPy 0.13.3， 但是你现在还不想升级到 0.14.0（文章发表时的最新版本），虽然你安装了其他依赖于 SciPy 的包，比如 Pandas。

为了达到目的，可以在你的环境中的 conda-meta 目录下创建一个叫 pinned 的文件。例如，如果你有一个叫做 scipy-0.13 的环境，你可以这么写：

$ echo "scipy 0.13.3" > ~/anaconda/envs/scipy-0.13/conda-meta/pinned

pinned 文件中的每一行都应符合 conda 匹配规则。这就允许一些通用的事情，比如说指定 scipy<0.14。其中以‘#’号开头的行会被忽略。

它的工作原理是，每次 conda 在你的环境里安装扩展包时，conda 会把 pinned 文件里的每一行内容都附带发送给 SAT 求解器，这样就阻止了那些你不想要的升级。

忽视 pinned 文件，可以使用conda install --no-pin 。

[[学习 Conda 的高级特性（上）](http://www.10tiao.com/html/160/201607/2649639290/1.html)]

**pyenv**

python3自带了pyvenv

from:[http://blog.csdn.net/pipisorry/article/details/47008981](http://blog.csdn.net/pipisorry/article/details/39998317)

ref:[使用VitrualEnvWrapper隔离python项目的库依赖](http://www.jeffkit.info/2011/08/1012/)

[一种部署 Python 代码的新方法dh-virtualenv](http://python.jobbole.com/85057/)