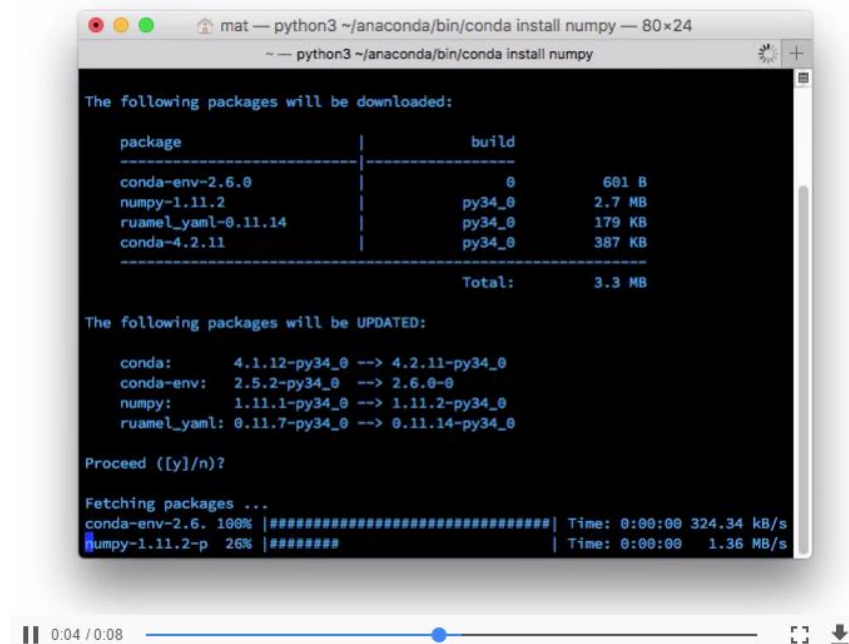


管理包

安装了 Anaconda 之后，管理包是相当简单的。要安装包，请在终端中键入 `conda install package_name`。例如，要安装 `numpy`，请键入 `conda install numpy`。



你还可以同时安装多个包。类似 `conda install numpy scipy pandas` 的命令会同时安装所有这些包。还可以通过添加版本号（例如 `conda install numpy=1.10`）来指定所需的包版本。

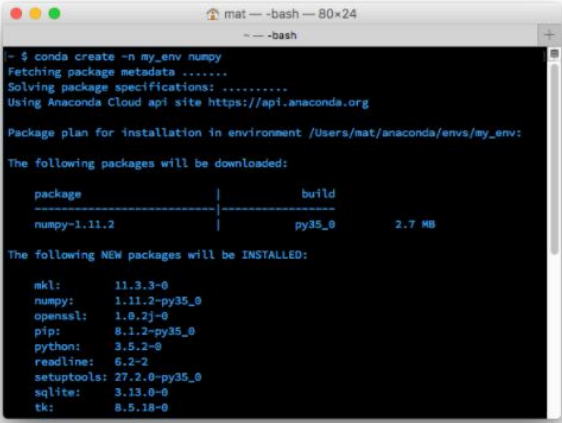
Conda 还会自动为你安装依赖项。例如，`scipy` 依赖于 `numpy`，因为它使用并需要 `numpy`。如果你只安装 `scipy` (`conda install scipy`)，则 conda 还会安装 `numpy`（如果尚未安装的话）。

大多数命令都是很直观的。要卸载包，请使用 `conda remove package_name`。要更新包，请使用 `conda update package_name`。如果想更新环境中的所有包（这样做常常很有用），请使用 `conda update --all`。最后，要列出已安装的包，请使用前面提过的 `conda list`。

如果不知道要找的包的确切名称，可以尝试使用 `conda search search_term` 进行搜索。例如，我知道我想安装 `Beautiful Soup`，但我不清楚确切的包名称。因此，我尝试执行 `conda search beautifulsoup`。

管理环境

如前所述，你可以使用 conda 创建环境以隔离项目。要创建环境，请在终端中使用 `conda create -n env_name list of packages`。在这里，`-n env_name` 设置环境的名称（`-n` 是指名称），而 `list of packages` 是要安装在环境中的包的列表。例如，要创建名为 `my_env` 的环境并在其中安装 `numpy`，请键入 `conda create -n my_env numpy`。



```
mat ~ -bash -- 80x24
-- -bash

~ $ conda create -n my_env numpy
Fetching package metadata .....
Solving package specifications: .....
Using Anaconda Cloud api site https://api.anaconda.org

Package plan for installation in environment /Users/mat/anaconda/envs/my_env:

The following packages will be downloaded:

package-----|-----build-----
numpy-1.11.2    |-----py35_0-----    2.7 MB

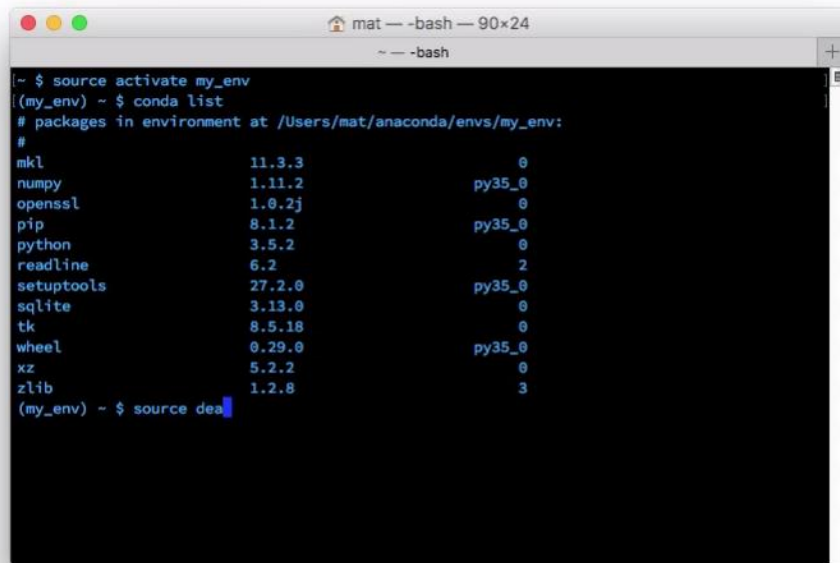
The following NEW packages will be INSTALLED:

mkl:           11.3.3-0
numpy:         1.11.2-py35_0
openssl:       1.0.2j-0
pip:           8.1.2-py35_0
python:        3.5.2-0
readline:      6.2-2
setuptools:    27.2.0-py35_0
sqlite:        3.13.0-0
tk:            8.5.18-0
```

创建环境时，可以指定要安装在环境中的 Python 版本。这在你同时使用 Python 2.x 和 Python 3.x 中的代码时很有用。要创建具有特定 Python 版本的环境，请键入类似于 `conda create -n py3 python=3` 或 `conda create -n py2 python=2` 的命令。实际上，我在我的个人计算机上创建了这两个环境。我将它们用作与任何特定项目均无关的通用环境，以处理普通的工作（可轻松使用每个 Python 版本）。这些命令将分别安装 Python 3 和 Python 2 的最新版本。要安装特定版本（例如 Python 3.3），请使用 `conda create -n py python=3.3`。

进入环境

创建了环境后，在 OSX/Linux 上使用 `source activate my_env` 进入环境。在 Windows 上，请使用 `activate my_env`。

A terminal window titled 'mat - bash - 90x24' showing the process of activating a conda environment. The user runs 'source activate my_env', and the prompt changes to '(my_env) ~ \$'. Then, they run 'conda list', which displays a table of installed packages. The table lists packages like mkl, numpy, openssl, pip, python, readline, setuptools, sqlite, tk, wheel, xz, and zlib with their versions and channels. The user then starts typing 'source dea' at the prompt.

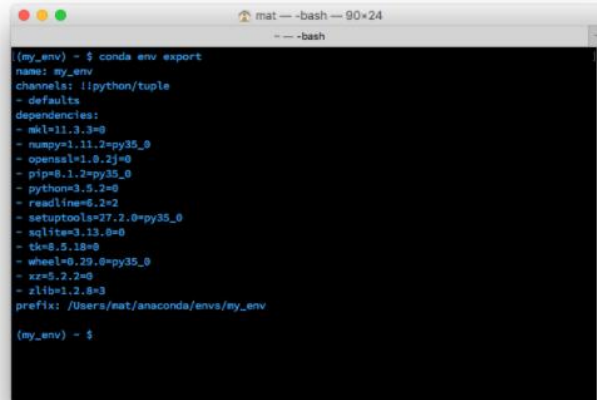
```
mat - bash - 90x24
~ - bash

~ $ source activate my_env
(my_env) ~ $ conda list
# packages in environment at /Users/mat/anaconda/envs/my_env:
#
mkl                    11.3.3                0
numpy                  1.11.2                py35_0
openssl                1.0.2j                0
pip                    8.1.2                py35_0
python                 3.5.2                0
readline               6.2                   2
setuptools             27.2.0               py35_0
sqlite                 3.13.0                0
tk                     8.5.18                0
wheel                  0.29.0               py35_0
xz                     5.2.2                0
zlib                   1.2.8                 3
(my_env) ~ $ source dea
```

进入环境后，你会在终端提示符中看到环境名称，它类似于 `(my_env) ~ $`。环境中只安装了几个默认的包，以及你在创建它时安装的包。你可以使用 `conda list` 检查这一点。在环境中安装包的命令与前面一样：`conda install package_name`。不过，这次你安装的特定包仅在你进入环境后才可用。要离开环境，请键入 `source deactivate`（在 OSX/Linux 上）。在 Windows 上，请使用 `deactivate`。

保存和加载环境

共享环境这项功能确实很有用，它能让其他人安装你的代码中使用的所有包，并确保这些包的版本正确。你可以使用 `conda env export > environment.yaml` 将包保存为 **YAML**。命令的第一部分 `conda env export` 用于输出环境中的所有包的名称（包括 Python 版本）。

A terminal window titled 'mat - bash' with a 90x24 resolution. The prompt is '(my_env) - \$'. The command 'conda env export' has been executed, and the output is a YAML file content. The output lists the environment name 'my_env', channels 'lpython/tuple' and 'defaults', and a list of dependencies with their versions: 'skl=11.3.3=0', 'numpy=1.11.2=py35_0', 'openal=1.9.2j=0', 'pip=8.1.2=py35_0', 'python=3.5.2=0', 'readline=6.2=2', 'setuptools=27.2.0=py35_0', 'sqlite=3.13.0=0', 'tk=8.5.18=0', 'wheel=0.29.0=py35_0', 'xz=5.2.2=0', and 'zlib=1.2.8=3'. The prefix is '/Users/mat/anaconda/envs/my_env'. The prompt changes to '(my_env) - \$'.

将导出的环境输出到终端中

上图中，你可以看到环境的名称和所有依赖项及其版本。导出命令的第二部分 `> environment.yaml` 将导出的文本写入到 YAML 文件 `environment.yaml` 中。现在可以共享此文件，而且其他人能够用于创建和你项目相同的环境。

要通过环境文件创建环境，请使用 `conda env create -f environment.yaml`。这会创建一个新环境，而且它具有同样的在 `environment.yaml` 中列出的库。



列出环境

如果忘记了环境的名称（我有时会这样），可以使用 `conda env list` 列出你创建的所有环境。你会看到环境的列表，而且你当前所在环境的旁边会有一个星号。默认的环境（即当你不在选定环境中时使用的环境）名为 `root`。

删除环境

如果你不再使用某些环境，可以使用 `conda env remove -n env_name` 删除指定的环境（在这里名为 `env_name`）。

TFlearn 安装

首先，我们需要设置 TFlearn。你需要创建一个叫做 `tflearn` (也可以设定任何其他名称) 的新环境。

```
conda create -n tflearn python=3.5
```

然后，进入该环境：

```
# on macOS or Linux
source activate tflearn

# on Windows
activate tflearn
```

我还会提供一个 notebook 文件作为练习，所以你需要安装 numpy、pandas、matplotlib 和 Jupyter notebook。

```
conda install numpy pandas jupyter notebook matplotlib
```

然后，安装 TFlearn 和其依赖项。

警告：下面的说明可能不适用于 Windows。如果你遇到任何错误，请尝试页面底部的解决方案。

```
conda install scipy h5py
pip install tensorflow
pip install TFlearn
```

现在应该已经准备好了。

用于解决 Windows 安装问题的潜在方案：

如果你遇到 `ImportError: No module named '_curses'` 这一错误，请尝试执行以下步骤：

转到[此链接](#)，找到与你的 Windows 系统版本对应的 Curses 文件并下载该文件。下载完毕后，运行 `pip install FILENAME`，其中 `FILENAME` 是你刚刚下载的文件名称。

如果不能解决问题，或者出现其他安装问题，请尝试：

学员 Andreas Wienzek 发现从源文件中安装 TFlearn 解决了问题。如果你已经用 pip 安装了 TFlearn，请用 `pip uninstall tflearn` 卸载该 TFlearn。然后，你可以[在此处下载源代码](#)并解压。转到相关目录下，然后运行 `python setup.py install` 即可构建和安装该程序包。

希望上述两个提示能够帮助你解决问题，如果不行，请告诉我们。

你可以在下面的链接中找到 ZIP 格式的 notebook 文件和数据。请下载该 ZIP 文件并解压，然后在你的 TFlearn 环境中运行 notebook。请自己编写代码，然后在下一个视频里观看我是如何编写解决方案代码的。

辅助材料

 [Sentiment Analysis with TFlearn](#)