Log in Create Free Account

Olivia Smith
March 5th, 2020

PYTHON

Virtual Environment in Python

In this tutorial, you'll learn about the Virtual Environment and two different ways of creating it.

GAIN THE CAREER-BUILDING PYTHON SKILLS YOU NEED WITH DATACAMP

Start Learning Now

Virtual Environment in Python

Learn about the Virtual Environment and two different ways for creating it: Pipenv is mostly used by web developers and Anaconda distributions for data scientists where Virtual Environment is created from 'conda' through 'Anaconda Prompt'. It also has an alternative option to create from Anaconda Navigator.

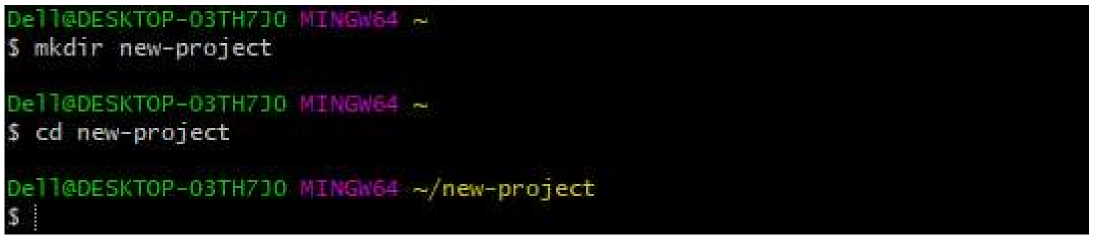
Virtual Environment is used to create a container or isolated environment where the Python-related dependencies are installed for a specific project. One can work with many different versions of Python with its various packages. Data scientists tend to use Anaconda distribution, which comes with many useful pre-installed packages, which are easy to install and manage. A web developer who uses Django, Flask, and other Python-related frameworks can use Pipenv as their Virtual Environment.

Pipenv

Through Pipenv, 'pip' and 'virtualenv' can be used together to create a Virtual Environment, Pipfile works as the replacement of the 'requirement.txt.' which tracks the package version according to the given project.

This tutorial uses Git Bash as the terminal in the Windows Operating System. However, for Mac users, Homebrew could be used, which is the package management tool, and similarly, LinuxBrew could be used for Linux users.

Let's create a project using the 'mkdir project-name' command, which stands for making a directory with the project name as 'new-project' and move to the newly created directory by using 'cd' command.



```
Dell@DESKTOP-03TH7J0 MINGW64 ~
$ mkdir new-project

Dell@DESKTOP-03TH7J0 MINGW64 ~
$ cd new-project

Dell@DESKTOP-03TH7J0 MINGW64 ~/new-project
$
```

Windows users can use the following command to install 'pipenv'.

```
pip install pipenv
```

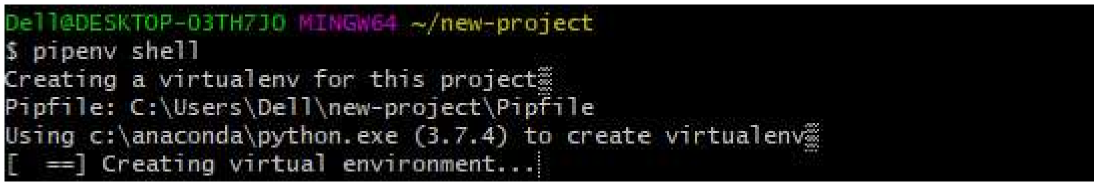
Linux/Mac Users can use the following command to install 'pipenv' after installing LinuxBrew.

```
brew install pipenv
```

Creating a virtualenv for the project

For creating the 'virtualenv' for the project, use the following command.

`pipenv shell` The command 'pipenv' creates a new 'virtualenv' for the project along with Pipfile side by side.

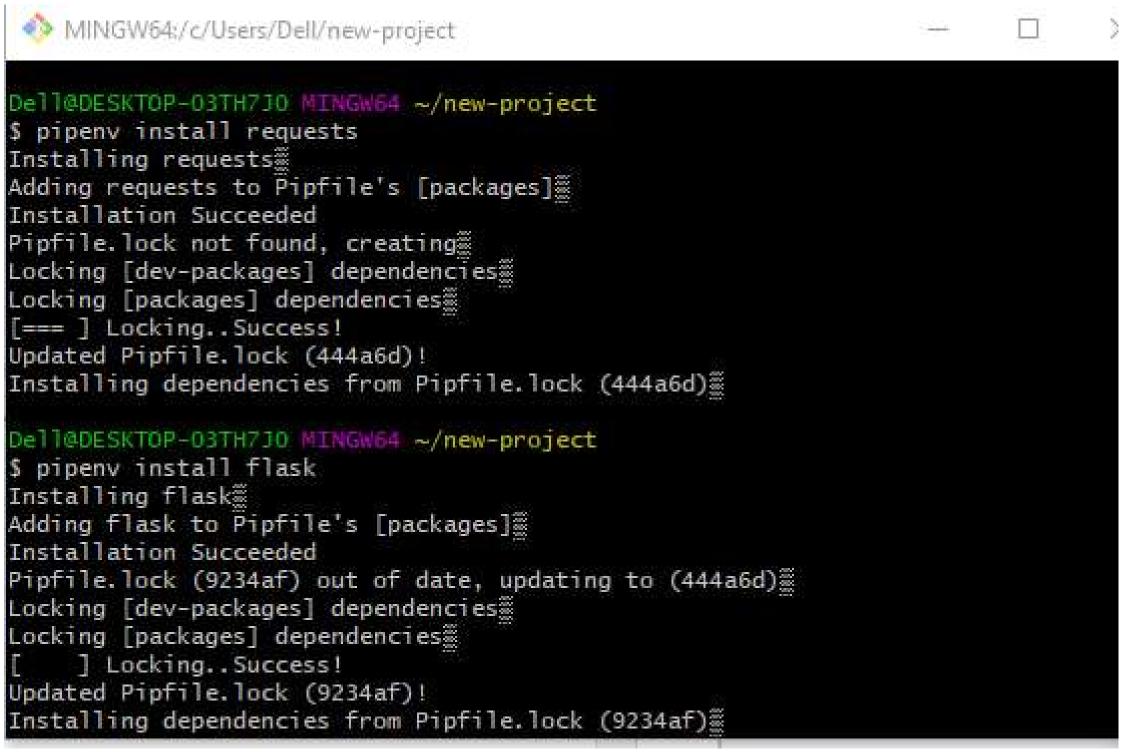


```
Dell@DESKTOP-03TH7J0 MINGW64 ~/new-project
$ pipenv shell
Creating a virtualenv for this project
Pipfile: C:\Users\Dell\new-project\Pipfile
Using c:\anaconda\python.exe (3.7.4) to create virtualenv
[  ==] Creating virtual environment...
```

You'll install the two package named 'requests' and 'flask' in your Virtual Environment using the following command.

```
pipenv install requests
pipenv install flask
```

The following changes are seen after the installation of the two packages.


MINGW64:/c/Users/Dell/new-project

```
Dell@DESKTOP-03TH7J0 MINGW64 ~/new-project
$ pipenv install requests
Installing requests▒
Adding requests to Pipfile's [packages]▒
Installation Succeeded
Pipfile.lock not found, creating▒
Locking [dev-packages] dependencies▒
Locking [packages] dependencies▒
[=== ] Locking..Success!
Updated Pipfile.lock (444a6d)!
Installing dependencies from Pipfile.lock (444a6d)▒

Dell@DESKTOP-03TH7J0 MINGW64 ~/new-project
$ pipenv install flask
Installing flask▒
Adding flask to Pipfile's [packages]▒
Installation Succeeded
Pipfile.lock (9234af) out of date, updating to (444a6d)▒
Locking [dev-packages] dependencies▒
Locking [packages] dependencies▒
[    ] Locking..Success!
Updated Pipfile.lock (9234af)!
Installing dependencies from Pipfile.lock (9234af)▒
```

For the uninstallation of the package, use the following command.

```
pipenv uninstall flask
```

Exiting from the Virtual Environment

You can use the following command to deactivate form the current environment. `exit`

Anaconda

Anaconda is the most popular platform used by data scientists and machine learning engineers. It consists of 'Conda', which helps in managing the environment, libraries, and dependencies.

Installing Anaconda for Python Virtual Environment Manager

- Linux/Mac Users then head over to the following tutorial to set up the Anaconda distribution.
  Install Anaconda Mac/Linux

Use the Anaconda version of 4.6 or newer. There are slight changes in the command for the previous version. If you are running older versions, use the following command for an update.
`conda update conda` One of two ways can be used in the creation of the Virtual Environment, which is shown below.

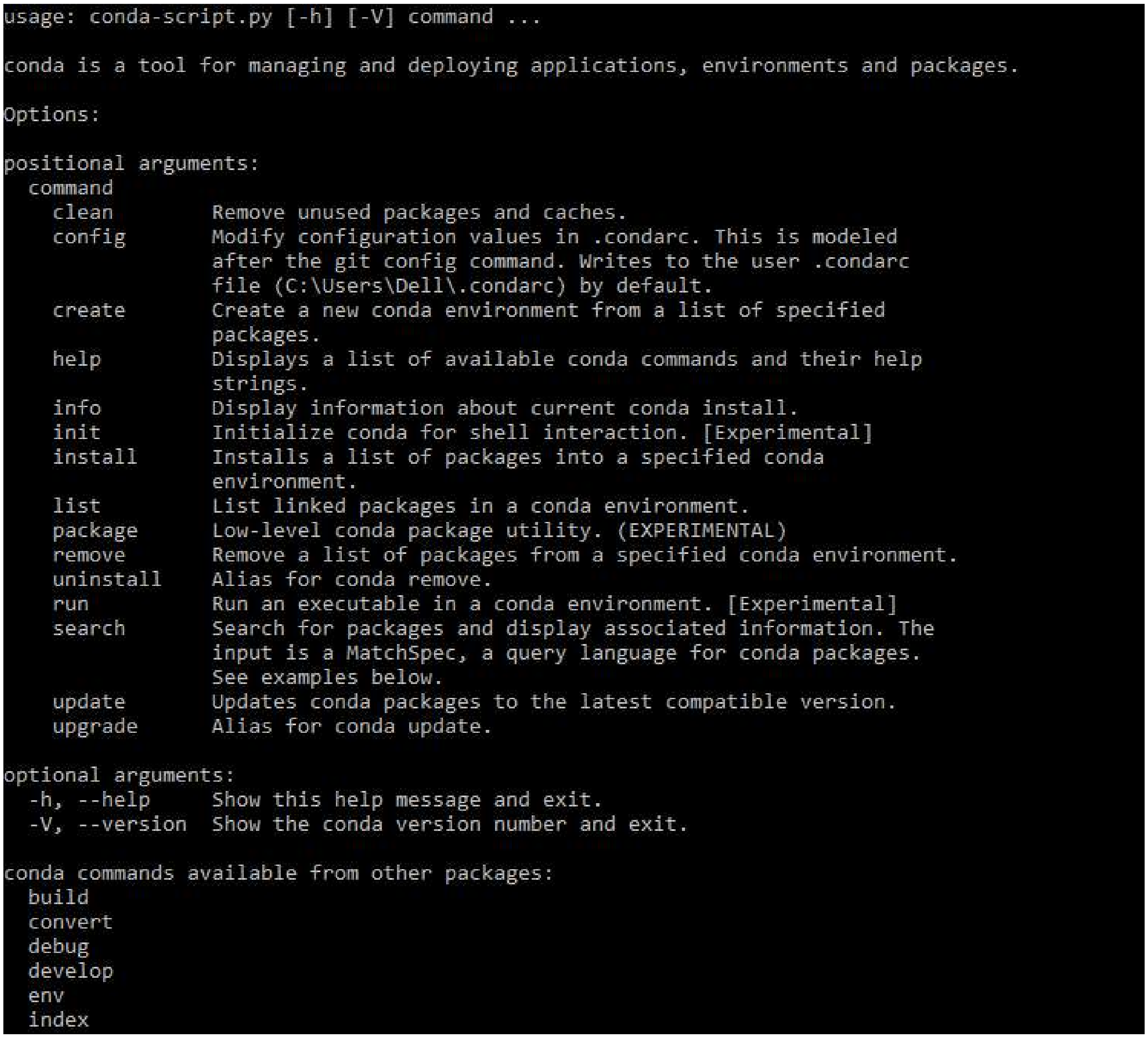
- Anaconda Prompt-It is a command-line tool that comes after the installation of Anaconda distribution.

- Anaconda Navigator-It is a Graphical User Interface that serves as an alternative tool to launch and manage packages in Anaconda.

  This tutorial uses Windows Operating System but works with any Operating System. There might be some minor changes in the Anaconda command, but the overall creation process for Virtual Environment is the same. You can open "Terminal" in Mac/Linux to achieve the following result. According to the requirements needed for the projects, the following guides serve more detail in the creation of a Virtual Environment.
  Managing environments

Anaconda Prompt

Press the "Windows" icon in the lower corner of the screen to open the "Search" box. Type "Anaconda Prompt" and then hit "Enter" to open it. Use `conda` to check Anaconda has been successfully installed in your system; the following changes could be seen.



```
usage: conda-script.py [-h] [-V] command ...

conda is a tool for managing and deploying applications, environments and packages.

Options:

positional arguments:
  command
    clean        Remove unused packages and caches.
    config       Modify configuration values in .condarc. This is modeled
                 after the git config command. Writes to the user .condarc
                 file (C:\Users\Dell\.condarc) by default.
    create       Create a new conda environment from a list of specified
                 packages.
    help         Displays a list of available conda commands and their help
                 strings.
    info         Display information about current conda install.
    init         Initialize conda for shell interaction. [Experimental]
    install      Installs a list of packages into a specified conda
                 environment.
    list         List linked packages in a conda environment.
    package      Low-level conda package utility. (EXPERIMENTAL)
    remove       Remove a list of packages from a specified conda environment.
    uninstall    Alias for conda remove.
    run          Run an executable in a conda environment. [Experimental]
    search       Search for packages and display associated information. The
                 input is a MatchSpec, a query language for conda packages.
                 See examples below.
    update       Updates conda packages to the latest compatible version.
    upgrade      Alias for conda update.

optional arguments:
  -h, --help     Show this help message and exit.
  -V, --version  Show the conda version number and exit.

conda commands available from other packages:
  build
  convert
  debug
  develop
  env
  index
```

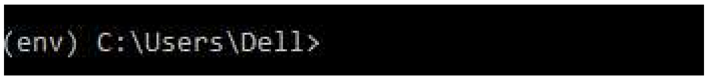
1. **Creating a new Virtual Environment.**

   The following command takes '-n' as a flag, which is for creating a new environment with its name as 'env' and the specific Python version of '3.7'.

   ```
   conda create -n env python = 3.7
   ```

2. **Activating the Virtual Environment.**

   The command below activates the Virtual Environment, which changes the prompt



```
(env) C:\Users\Dell>
```

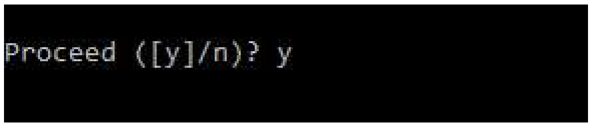
3. **Install the required package.**

   For example, the 'numpy' package is installed where 'env' is the specific Virtual Environment.

   `conda install -n env numpy`

   OR

   Also, Python Package manager could be used to install 'numpy'. `pip install numpy`

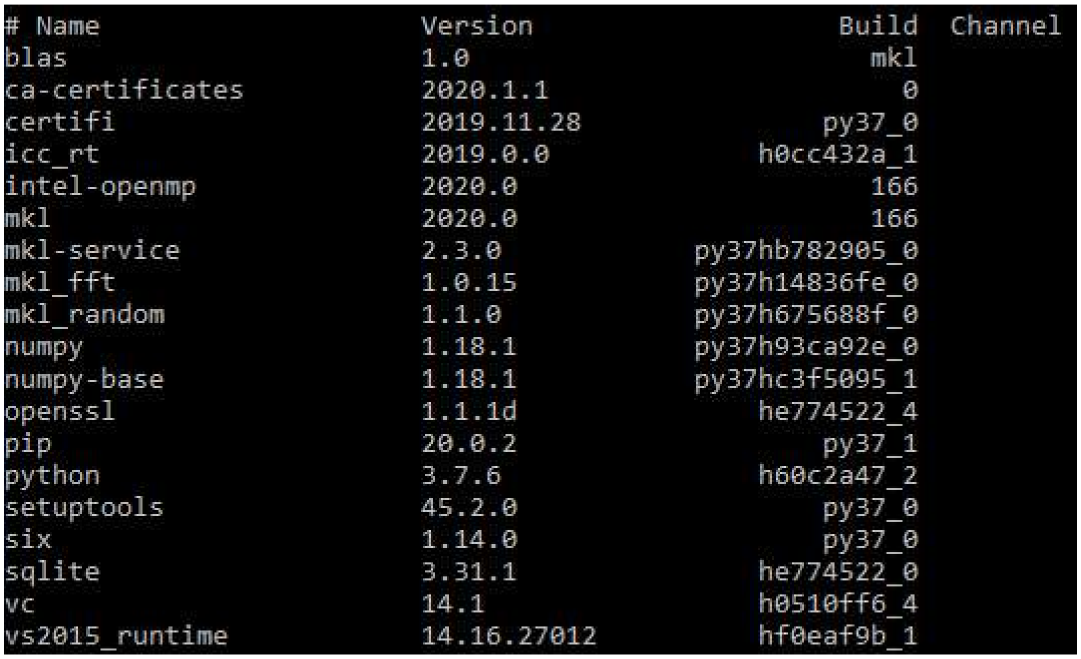


```
Proceed ([y]/n)? y
```

4. **Listing all of the installed packages inside a Virtual Environment.**

   The following command can list the package specific to the Virtual Environment.

   `conda list`

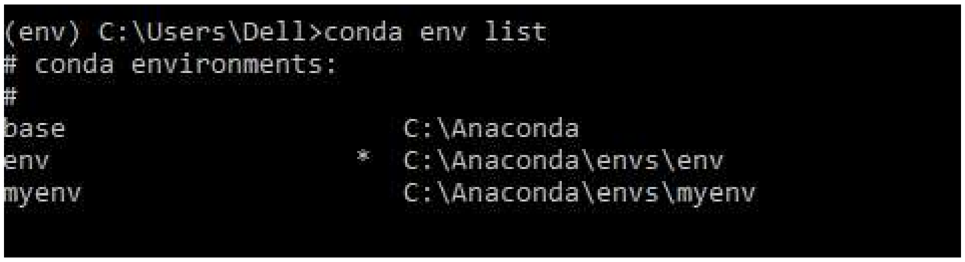


```
# Name                    Version                   Build  Channel
blas                      1.0                         mkl
ca-certificates           2020.1.1                      0
certifi                   2019.11.28                py37_0
icc_rt                    2019.0.0               h0cc432a_1
intel-openmp              2020.0                      166
mkl                       2020.0                      166
mkl-service               2.3.0            py37hb782905_0
mkl_fft                   1.0.15           py37h14836fe_0
mkl_random                1.1.0            py37h675688f_0
numpy                     1.18.1           py37h93ca92e_0
numpy-base                1.18.1           py37hc3f5095_1
openssl                   1.1.1d               he774522_4
pip                       20.0.2                    py37_1
python                    3.7.6                h60c2a47_2
setuptools                45.2.0                    py37_0
six                       1.14.0                    py37_0
sqlite                    3.31.1               he774522_0
vc                        14.1                 h0510ff6_4
vs2015_runtime            14.16.27012          hf0eaf9b_1
```

5. **Listing out all of the created Virtual Environment.**

   All of the environments created will be listed by the following command.

   `conda env list`



```
(env) C:\Users\Dell>conda env list
# conda environments:
#
base                     C:\Anaconda
env                   *  C:\Anaconda\envs\env
myenv                    C:\Anaconda\envs\myenv
```

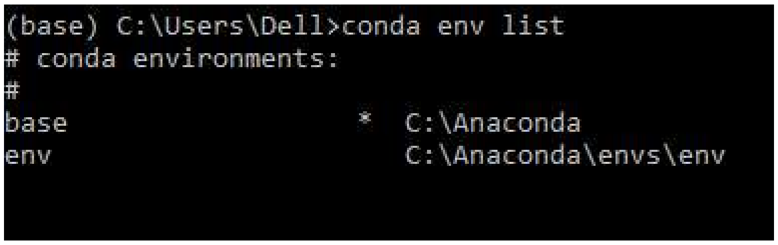
6. **Deactivating the Virtual Environment.**

   The following command will deactivate the current environment 'env' and will change to 'base'. `conda deactivate`

1. **Removing the Virtual Environment.**

   The following command removes the 'myenv' Virtual Environment with all its packages at the same time. `conda env remove -n myenv`

As you can see after listing with `'conda env list'`, only two Virtual Environments are shown.

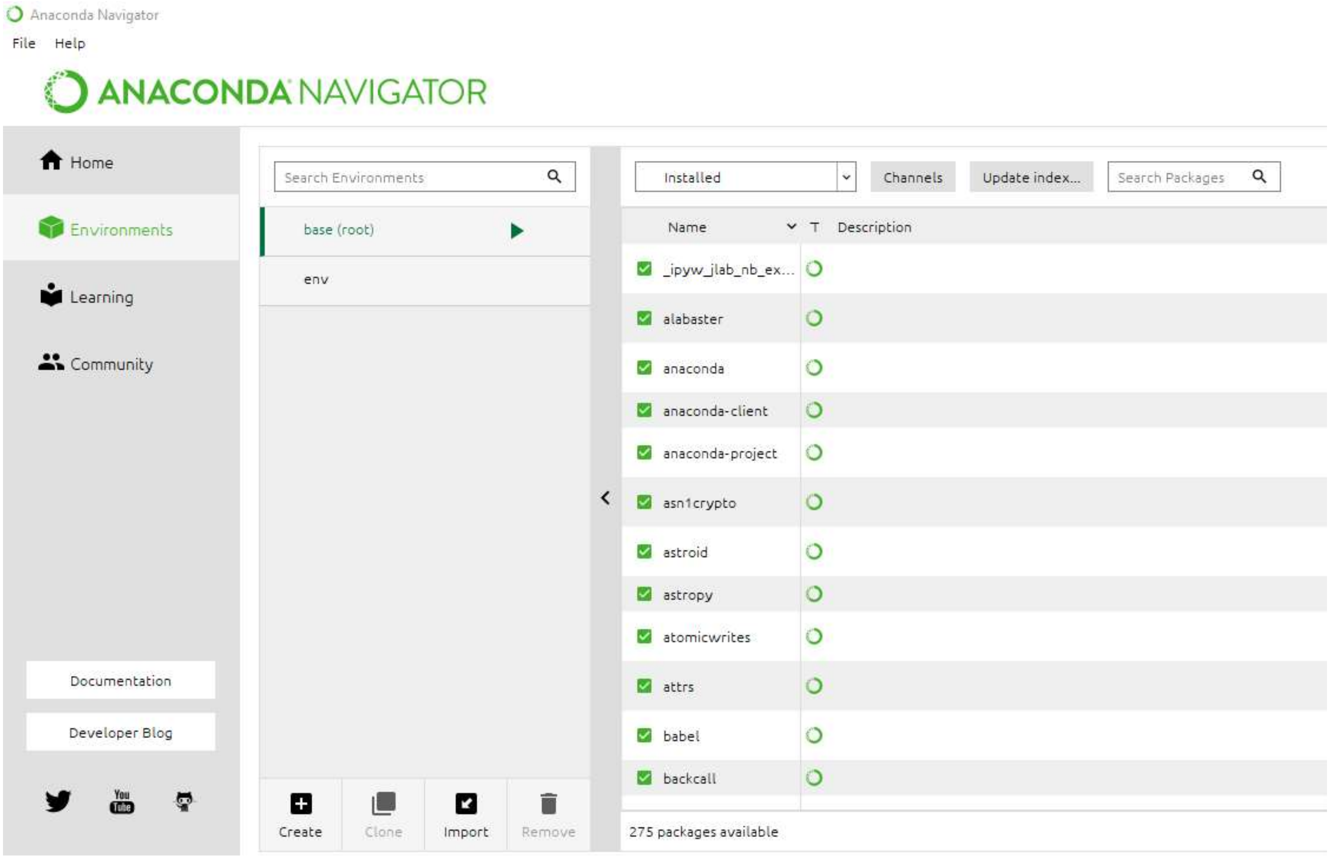


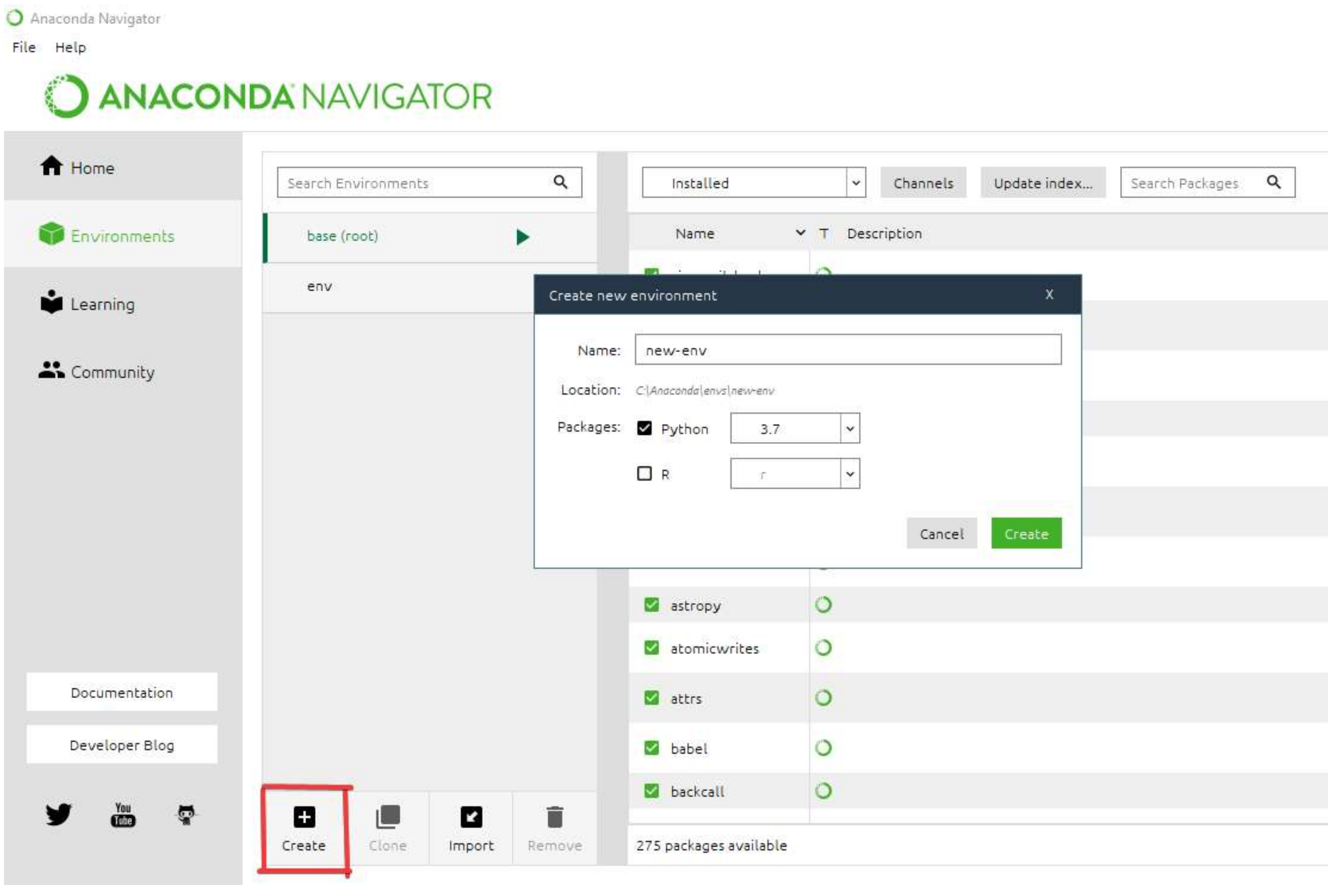
```
(base) C:\Users\Dell>conda env list
# conda environments:
#
base                  *  C:\Anaconda
env                      C:\Anaconda\envs\env
```

Anaconda Navigator

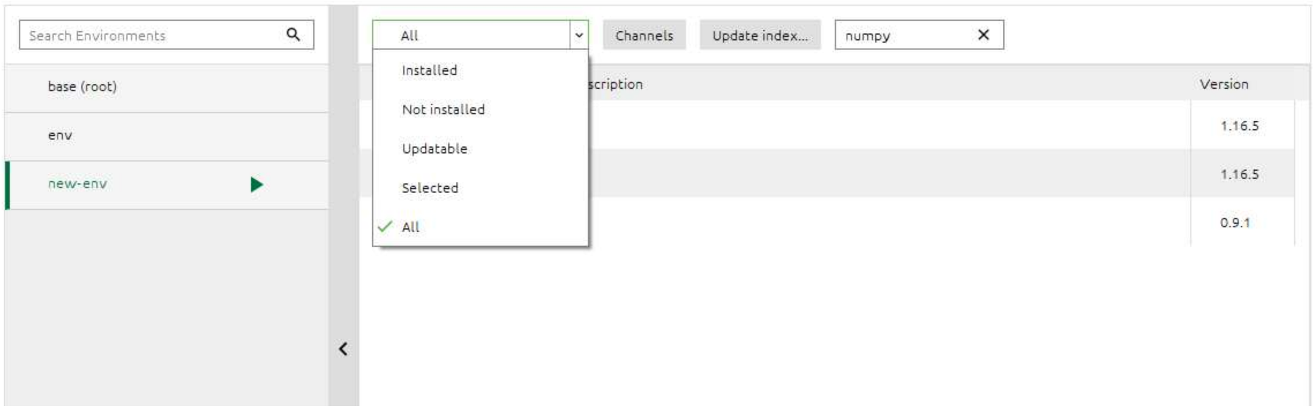
1. Press the "Windows" icon in the lower corner of the screen to open the "Search" box. Type "Anaconda Navigator" and then hit "Enter" to open it.

2. Move to "Environments," where two virtual environments are shown.' base' is the default, whereas 'env' is the previously created Virtual Environment.

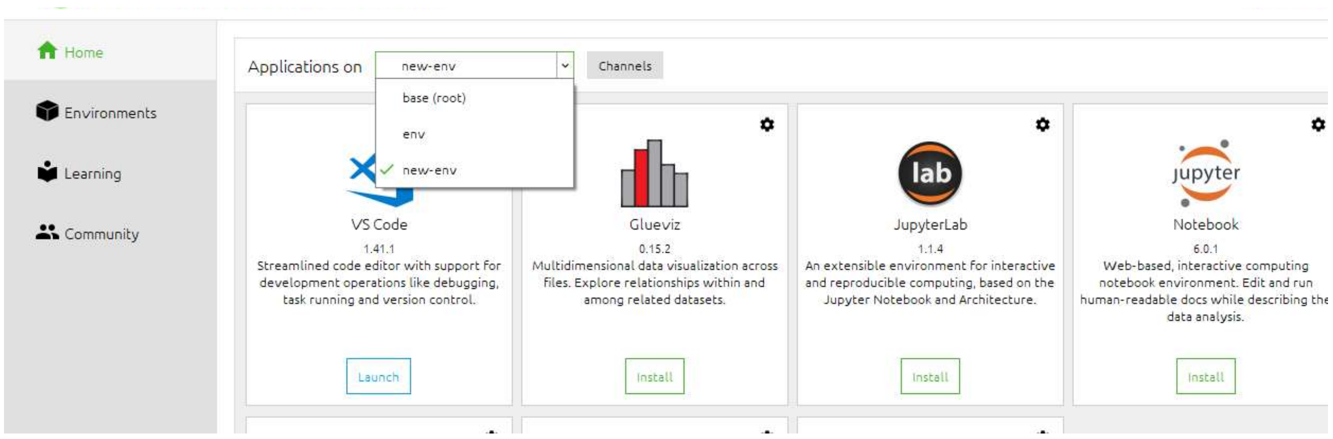




4. Move to the "new-env" and select "All" to install the required package, i.e., numpy in our case.



5. Select 'new-env' at the home directory so that launching an application will be specific to that particular virtual environment.


Home
Environments
Learning
Community
Applications on
new-env
base (root)
env
new-env
Channels
VS Code
1.41.1
Streamlined code editor with support for development operations like debugging, task running and version control.
Launch
Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install
JupyterLab
1.1.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Install
Notebook
6.0.1
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Install

Conclusion

Congratulations on finishing the tutorial!

You've successfully learned about the Virtual Environment and creation through Pipenv and Anaconda.

You can look over to the following courses created by Anaconda in DataCamp platform to learn more:

- Conda for Building & Distributing Packages

- Conda Essentials

References:
Installing pipenv

13

Subscribe to RSS

About Terms Privacy