Python Installation and package managing using Conda

Alexandre Neto

2022-12-29

There are several ways to install Python on your system. Some operating systems even bring python installed by default. Nevertheless, to accelerate the development, working with python often leads you to install extra python packages (which are collections of python modules with useful functionality). Besides, some projects may need very specific versions of a certain package to work. Therefore, installing packages system wide is probably not a good idea. There are two main package managers for python, pip and conda. We will use the later.

Conda is a package managing system that allows you to create isolated environments by project, where you can choose a specific Python base version and select which packages to include and their versions. This not only allows you to isolate a projects environment, but makes it easy for you to replicate it on other machines, if needed. Conda also allows you to install full programs, like QGIS or jupyterlab.

There are several ways to install conda (Anaconda, miniconda, miniforge, mambaforge,etc…). My favorite is **Miniforge**. It's a community installer that uses conda-forge as it's main channel (a channel is a place where packages are available). It has the minimum set of necessary tools to work with conda, keeping all the functionality of Anaconda, but without installing too many packages that you will probably never use. That's why I suggest to use it.

This document shows how to install and use miniforge.

Install conda using miniforge

1. Go to https://github.com/conda-forge/miniforge and download the latest installer for your specific platform (probably Miniforge Windows 64-bit).
2. Once downloaded, for windows, double-click the .exe installer and use the default suggested directory. Select the **Just me** option and use the remaining default settings.

Note: In case your username has spaces or non-english characters, it's highly adviced to install miniconda in another folder such as `c:\miniconda` or it may cause problems.

For Linux or Mac, if needed, make the .sh file executable and execute the bash script:

```
bash Miniforge3-Linux-x86_64.sh
```

During the installation, go with all the default options

Create a new environment

As said before, for each project, it's recomended to create an isolated environment.

1. On windows, open the anaconda command prompt. On Linux or Mac use your default terminal. Then type:

   ```
   conda create -n name_of_the_environment
   ```

2. Activate the environment

   ```
   conda activate name_of_the_environment
   ```

When a environment is activate the prompt will show the name of the environment. Something like this:

```
(name_of_the_environment) C:\users\
```

3. Once you have finished using that environment, you can deactivate it.

   ```
   conda deactivate
   ```

Install new packages

With the project environment activated, you can install python packages (and actually other software like QGIS) in it, without affecting other installations. Conda allows you to use packages from many different channels, but the default of miniforge is conda-forge which we will use exlusively (not a got ideia to use more than one channel in each environment).

1. Try installing the `jupytelab` package

   ```
   conda install jupyterlab
   ```

   If you are not using miniforge or mambaforge, you may need to explicitly choose the conda-forge channel for correct package installation.

   ```
   conda install -c conda-forge jupyterlab
   ```

2. You can install more than one package at a time:

   ```
   conda install pandas geopandas
   ```

3. You can also expecify the version of a certain package that you want to use.

   ```
   conda install  flask=2.2.2
   ```

Manage installed packages

1. To show all the package that have been install in your environment:

   ```
   conda list
   ```

2. If you know that there is a new version of a package available, you can get the most recent version by doing the following:

   ```
   conda update flask
   ```

   Note: that may emply updating related packages too

3. You can remove one or more packages

   ```
   conda remove scipy
   ```

Save and load a conda environment

If you want to save or share you environment with others, you can create a yaml file with it.

```
conda activate name_of_the_environment
conda env export -f environment.yml
```

It's a good practice to keep an `environment.yml` file at the root of your projects. If someone else (ot the future you) wants to replicate your environment all he needs to do is:

```
conda env create -n new_env_name --file env.yml
```

There is much more you can do with conda, including compiling your own packages, save and share environments, and so on. For more information about it, check the documentation https://docs.conda.io/projects/conda/en/latest/user-guide/index.html