Python Programming Best Practices: Conda and Virtual Environments

Agenda

Tutorial: Conda and Virtual Environments

- Virtual Environments
- Package and Environment Managers
- Installing Conda
- What is Anaconda?
- What is Miniconda?
- What to choose?
- What is a root environment?
- Creating a virtual environment
- Switching to an environment
- Installing packages to an environment
- Uninstalling packages from an environment
- Removing an environment
- Other commands
- Links

Virtual environments

- **What is a virtual environment in python?**
  - *A virtual environment is a named, isolated, working copy of Python that that maintains its own files, directories, and paths*
- **Why do we need one?**
  - *Virtual environments make it easy to cleanly separate different python projects and avoid problems with different dependencies and version requirements across components*
- *Note: We recommend to use different virtual environments for different projects*

Virtual environments (cont.)

- **Why do we need different environments?**
  - Changes in the package versions:
    - You have developed a nice application, which had worked before, but then it suddenly started to throw exceptions or to completely crash
    - The reason can be that you updated one of the packages you used for your application and it is not compatible to your application anymore (e.g. a signature of a class you used has changed)
    - By setting an environment for your application with Python and packages you can be sure that the application will still work
  - Deploying the application on a server, delivering it to the client (reproducibility):
    - You want to ensure that the application developed by can also be run at your client's machine
    - You can create an environment and also save all the packages' names and versions in a separate file

Virtual environments (cont.)

- **Why do we need different environments?**
  - Collaboration
    - If you are developing an application together with your colleagues, you want to be sure that the code written by them will also work for you and vice versa
  - Developing an app in a python version you usually do not use
    - E.g. you always use python 3.6, however, for one specific data science project you need a package that is compatible only with python 2. So you can create a Python 2 environment for this project

Package and environment managers

- PIP (Python package manager 'Pip Installs Packages') and virtualenv (PIP tool for creating environments)
- Conda (package and environment manager)
  - In this guide we will be using conda, due to its flexibility, clear structure and multipurpose application
  - Using package manager *conda* you can create, export, list, remove and update environments that have different versions of Python and/or packages installed in them

Installing Conda

- **We can install conda in three different ways:**
  - Anaconda
  - Miniconda
  - Anaconda Entreprise Platform (=commercial product and is irrelevant for us)

What is Anaconda?

- **Anaconda = enterprise-ready Python distribution for data analytics, processing, and scientific computing**
  - 150+ Python packages
  - Package manager conda
  - Anaconda Navigator (GUI for managing conda environments)
  - Requires about 3 GB space on your hard drive
- **By installing Anaconda you install the majority of packages to start working on your scientific project**

What is Miniconda?

- **Miniconda = minimalistic distribution of Anaconda containing conda package manager and few basic Python packages:**
  - Basic Python packages
  - Package manager conda
  - Requires about 400 MB space on your hard drive

What to choose?

- **Choose Anaconda:**
  - You like having 150+ Python packages preinstalled and do not want to install each package individually
  - You have sufficient space on your disk
- **Choose Miniconda:**
  - You do not mind installing packages yourself and want to avoid pre-installed packages that you will never use
  - Save disk space

Note: We strongly recommend Python 3 and choosing the corresponding ana/miniconda installer. If you need Python 2 for a different project, you are able to set up a new virtual environment with Python 2

What is a root environment?

- Root environment is the default environment.
- If you run your python code without activating any other environment before, it will run in your root environment
- Root environment contains:
  - Certain version of Python 2 or 3 (the one you installed through ana/miniconda installer)
  - Basic Python packages

Creating a virtual environment

- Open the command prompt (Windows) or terminal (OSX)
- Create a new virtual environment (see example on next slides)
  - To create new virtual environment you need to use the following command:
    - conda create –n *nameOfTheEnvironment [packages it should contain]*
    - *-n option means name of the environment you are working with*
    - *In the [] you can find the packages you want to have in your environment and the version of Python*
  - And press Enter
  - When asked if you want to proceed, type y (for “yes”) and hit enter

Creating a virtual environment (cont.)

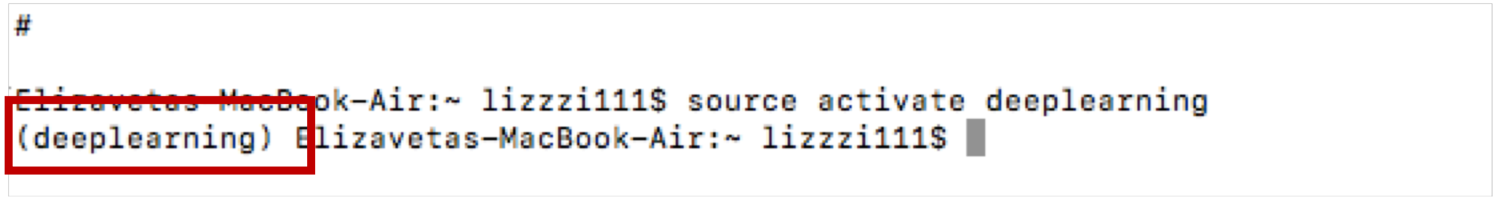
- **For example we are working with textual data and deep learning and we prefer Python 3.6**
  - We want to install packages “keras”, “scikit-learn”, “nltk”, “pandas”, “tensorflow” (the set of packages dpeneds on your application)
  - We also want jupyter lab to work with
- **Create a new virtual environment**
  - Choose an environment name, e.g. *deeplearning*
  - Type:
    - conda create –n *deeplearning python=3.6 keras scikit-learn nltk pandas tensorflow jupyterlab*
  - And press Enter
  - When asked if you want to proceed, type y and hit enter

Switching to an environment

- **Activate your newly created environment**
  - Type:

  Windows: activate *deeplearning*

  Linux/ MacOS: source activate *deeplearning*
  - And press Enter
- **This is what your console should look like currently**



```
#

Elizavetas-MacBook-Air:~ lizzzi111$ source activate deeplearning
(deeplearning) Elizavetas-MacBook-Air:~ lizzzi111$ 
```

- **Instead of the usual terminal prompt you will also see the name of the virtual environment**

Switching to an environment (cont.)

- **To deactivate the environment use:**
  - Type:

    Windows: deactivate

    Linux/ MacOS: source deactivate
  - And press Enter

Installing packages to an environment

- If you do not specify –n option (--name), in the *conda install* command the package will be installed to the active environment
  - If you have not activated any environment, command *conda install nameOfPackage* will install this package into your root environment.

- You can also install many packages in one line:

```
conda install scipy keras
```

- You can specify their versions if needed

```
conda install scipy=0.15.0
```

Installing packages to an environment (cont.)

- On the previous slide we created an environment and installed all the packages we need in one step
- To install packages to already existing environment
  - E.g. we created our environment by using:

    conda create –n deeplearning python=3.6
  - To install additional packages:

    conda install –n nameOfEnvYouWantToInstallPackageIn nameOfThePackage

    conda install –n deeplearning keras jupyterlab pandas numpy
  - If you do not specify –n option (--name), the package will be installed to the active environment
    - If you have not activated any environment, command *conda install nameOfPackage* will install this package into your root environment.

Uninstalling packages from an environment

- **To remove a packages from an environment:**
  - conda uninstall –n nameOfEnvYouWantToUninstallPackageFrom nameOfThePackage
  - E.g. conda uninstall –n deeplearning keras
  - And press Enter
  - Type y and hit enter when asked if you want to proceed

Removing an environment

- To remove an environment:
  - conda remove –n nameOfEnv --all
  - e.g. conda remove –n deeplearning --all
  - And press Enter
  - Type y and hit enter when asked if you want to proceed

Other commands

- **To list all your conda environments use:**
  - conda info --envs

  or
  - conda env list
- **To list all packages in a specific environment**
  - conda list –n nameOfTheEnv
- **To update packages:**
  - conda update –n nameOfTheEnv nameOfPackage

  Or to update all pacakges
  - conda update
- **To see if a specific package available for installation using conda:**
  - conda search nameOfPackage
  - conda search scipy
- **To get the version of Conda use:**
  - conda --version

Additional info

- Note: if you do not use –n or --name option in your conda command (update, install, remove) it will be applied to your root environment
- If you activated a conda environment you still can install packages using PIP for this environment:
  - e.g. pip install keras
- As mentioned before, we recommend to create separate environments for different projects!

Links

- Official User-Guide: https://conda.io/docs/user-guide/tasks/manage-environments.html
- Medium Blogpost: https://medium.freecodecamp.org/why-you-need-python-environments-and-how-to-manage-them-with-conda-85f155f4353c

(This guide is to a big extent inspired by this blog post, so all credits go to Gergely Szerovay)

- https://stackoverflow.com/questions/45421163/anaconda-vs-miniconda