

Créer et utiliser un Environnement Python dédié pour utiliser le module tensorflow 1

synthèse de <https://conda.io/docs/user-guide/tasks/manage-environments.html>,
voir cette URL pour plus de détails

Intérêt

Un **environnement Python dédié** (aussi appelé *environnement virtuel*) est un environnement informatique étanche contenant une installation de Python :

- indépendante des autres environnements Python susceptibles de coexister sur la même machine,
- indépendante des mises à jour de l'ordinateur (qu'il soit sous GNU/Linux, OS X ou Windows).

Un **environnement Python dédié** repose, entre autre, sur la création d'un **espace disque dédié** (une partie de l'arborescence disque) où seront installés la version de Python et des modules dont tu as besoin pour ton projet. Quand tu actives un environnement python dédié, la variable d'environnement **PATH** est modifiée de sorte que l'interpréteur Python et tous les modules sont recherchés dans l'arborescence associée à cet environnement, et nulle part ailleurs.

Pour travailler un projet d'IA avec Python, il est important de figer un environnement Python dédié au projet (version de python et des modules utilisés). Les modules Python Keras ou tensorflow doivent impérativement être utilisés dans des versions maîtrisées, ce qui nécessite de maîtriser l'environnement Python dédié à l'utilisation de ces modules.

Création d'un environnement Python

Parmi les différentes méthodes permettant de créer un environnement Python, citons :

- la commande `conda`, disponible si tu as installé Python sur ta machine avec la distribution *Anaconda*, ou si tu as juste installé la distribution minimale *miniconda*.
- le module `venv` qui permet de créer un environnement virtuel (cf docs.python.org/3/library/venv.html) ;

Comme nous utilisons *Anaconda* pour les activités pédagogiques à l'ENSAM, je vais continuer l'explication avec la commande `conda`, disponible avec la distribution *Anaconda*.

Si tu n'as pas installé Python sur ton PC, ou si tu ne l'as pas installé avec *Anaconda*, le plus simple est d'installer simplement *miniconda* (voir <https://docs.conda.io/en/latest/miniconda.html>) pour continuer.

Pour travailler ce TP avec les modules Keras et tensorflow nous allons utiliser la version 3.6 de Python.

Créer un environnement Python 3.6 dédié

Dans les pages qui suivent, les commandes dans un cadre avec fond jaune pâle sont à taper :

- dans un terminal, pour GNU/Linux ou OS X ;
- dans une fenêtre *prompt Anaconda*, pour Windows.

Pour créer un environnement **Python 3.6** nommé `pym1` par exemple (*Python for machine learning*), taper simplement :

```
conda create -n pym1 python=3.6
```

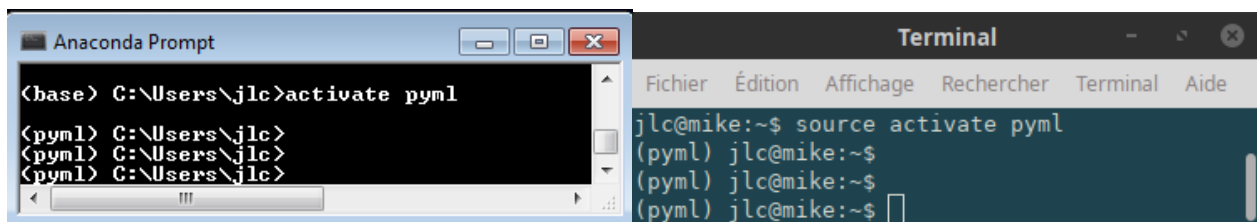
- conda crée un dossier `pyml` dans le sous-répertoire `envs` du répertoire d'installation `Anaconda3` (ou `miniconda3`).
- Puis conda télécharge et installe les dossiers et fichiers nécessaires dans le répertoire `../Anaconda3_ou_miniconda3/envs/pyml`.

Activer un environnement Python dédié

Pour activer l'environnement `pyml` il suffit d'utiliser la commande `activate`. Deux syntaxes existent, en fonction du système d'exploitation :

```
activate pyml          # → Windows
source activate pyml   # → GNU/Linux ou Mac Os X
```

- L'activation de l'environnement Python se traduit par un préfixe de la forme **(nom_environnement)** qui est ajouté dans le terminal (la fenêtre *prompt Anaconda*) en début du prompt :



Finaliser l'installation des paquets Python dans l'environnement dédié

Attention : bien vérifier que l'environnement `pyml` est activé [pré-fixage du prompt avec **(pyml)**] !

```
(pyml) jlc@mike conda install numpy=1.16 scipy matplotlib jupyter
```

Installer le module open-CV, utile pour toutes les manipulation de vidéos ou d'images :

```
(pyml) jlc@mike conda install -c anaconda opencv
```

Installer ensuite les autres modules nécessaires à la programmation des algorithmes de *machine learning* :

- installer le module tensorflow en **version 1.12**

```
(pyml) jlc@mike conda install tensorflow=1.13
```

- installer le module keras en **version 2.2.4**

```
(pyml) jlc@mike y
```

« conda install » ou « pip install » ?

La stratégie est simple :

1/ Essayer toujours en premier « `conda install ...` » :

`conda install numpy` permet d'installer le module numpy avec la **bibliothèque hyper optimisée MKL d'INTEL**, alors que `pip install numpy` installe le module avec des versions moins optimisées.

2/ Si le module n'est pas connu de conda, l'installer avec pip ...

Utiliser jupyter-notebook dans un environnement Python dédié

Une fois dans l'environnement Python dédié [noter le pré-fixage du prompt avec **(pym)**] tu peux lancer l'éditeur de cahiers IPython en tapant la commande :

```
(pym) jlc@mike jupyter notebook
```

Windows : Si tu utilises un PC sous Windows, tu peux si besoin ajouter un argument indiquant le **chemin du dossier de travail** de jupyter :

```
(pym) C:\Users\jlc> jupyter notebook "C:\...dossier_de_travail..."
```

→ le navigateur web par défaut se lance et charge la page d'accueil jupyter qui montre l'arborescence du dossier de travail. Le bouton New permet de créer un cahier IPython dans l'environnement dédié Python **pym**.

Il suffit ensuite de taper et d'exécuter dans le nouveau cahier IPython les 2 lignes ci-dessous, pour vérifier que la version de python est bien 3.6.x :

```
Entrée [1]: import sys  
            sys.version
```

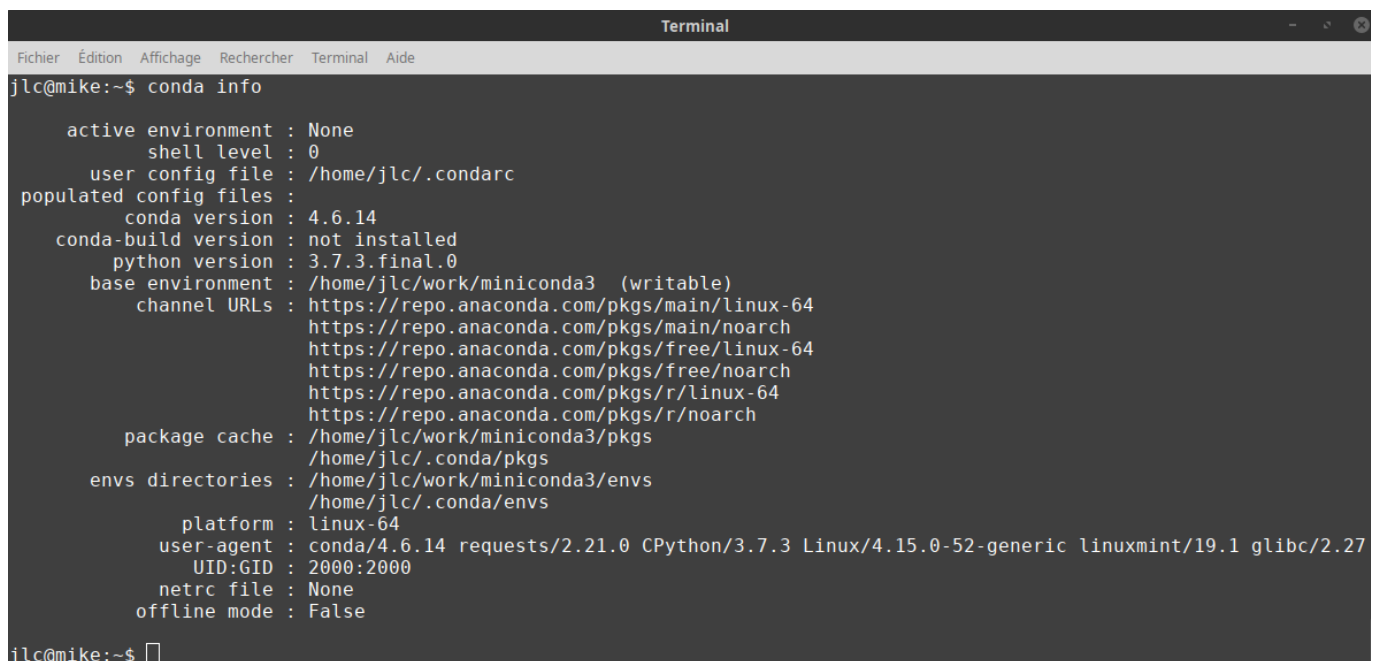
```
Out[1]: '3.6.8 |Anaconda, Inc.| (default, Dec 30 2018, 01:22:34) \n[GCC 7.3.0]'
```

► **Quitter Jupyter notebook** Après avoir quitté jupyter notebook, il faut taper 2 fois de suite **CTRL-C** dans le terminal ou la « fenêtre prompt Anaconda » pour « reprendre la main ».

How To...

Obtenir des informations sur la version et la configuration de conda installée :

```
conda info
```



```
Terminal
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
jlc@mike:~$ conda info

  active environment : None
    shell level      : 0
   user config file   : /home/jlc/.condarc
populated config files:
  conda version      : 4.6.14
conda-build version   : not installed
  python version     : 3.7.3.final.0
  base environment    : /home/jlc/work/miniconda3 (writable)
   channel URLs       : https://repo.anaconda.com/pkgs/main/linux-64
                        https://repo.anaconda.com/pkgs/main/noarch
                        https://repo.anaconda.com/pkgs/free/linux-64
                        https://repo.anaconda.com/pkgs/free/noarch
                        https://repo.anaconda.com/pkgs/r/linux-64
                        https://repo.anaconda.com/pkgs/r/noarch
   package cache      : /home/jlc/work/miniconda3/pkgs
                        /home/jlc/.conda/pkgs
  envs directories    : /home/jlc/work/miniconda3/envs
                        /home/jlc/.conda/envs
    platform         : linux-64
   user-agent        : conda/4.6.14 requests/2.21.0 CPython/3.7.3 Linux/4.15.0-52-generic linuxmint/19.1 glibc/2.27
      UID:GID        : 2000:2000
   netrc file        : None
  offline mode       : False

jlc@mike:~$
```

Lister les environnements installés et leur localisation :

```
conda env list
```

affiche les environnements installés et leurs dossiers d'installation :

- l'environnement actif est marqué avec un * ;
- l'environnement d'installation original d'Anaconda est identifié <base>.

Lister les modules d'un environnement :

Pour lister les module de l'environnement actif :

```
conda list
```

Pour lister les module d'un environnement particulier, utiliser l'option -n suivie du nom de l'environnement :

```
conda list -n pyml  
conda list -n base
```

Activer l'environnement :

```
activate pyml          # Windows  
source activate pyml # GNU/Linux ou Mac Os X
```

Quand un environnement est activé toutes les commandes tapées dans le terminal ou la « fenêtre prompt Anaconda » sont exécutées dans cet environnement : c'est le but recherché !

Si tu as besoin de désactiver l'environnement actif :

```
deactivate             # Windows  
conda deactivate       # GNU/Linux ou OS X
```

Si tout va mal...

Si tout va mal, le plus simple est de supprimer l'environnement et de le refaire...

Pour supprimer un environnement, le désactiver :

```
Windows → deactivate  
Lx & Mac → conda deactivate
```

puis le supprimer pour de bon :

```
conda env remove -n pyml
```

et il n'y a plus qu'à tout recommencer ;-)