Créer et utiliser un **Environnement Python dédié**

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 :
– indépendant des autres environnements Python susceptibles de co-exister sur la même machine,
– indépendant des mises à jour de l'ordinateur (qu'il soit sous GNU/Linux, Mac 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 dédiée associée à cet environnement, et nulle part ailleurs.

Dans le cadre de l'utilisation de Python pour un projet en intelligence artificielle, il est très important de pouvoir figer un environnement Python dédié (version de python et des modules utilisés) le temps du travail sur le projet. Les modules Pythons comme Keras ou tensorflow doivent être impérativement être utilisés dans des versions très stables qui ne sont en général pas les « dernières versions », ce qui nécessite de maîtriser l'environnement Python dédié à l'utilisation de ces modules.

Création d'un environnement Python

Parmi toutes les méthodes permettant de créer un environnement Python, citons :
– la commande `pyenv` du module python `pyenv` ;
– la commande `conda`, disponible si tu as installé Python sur ta machine avec la distribution complète *`Anaconda`*, ou si tu as juste installé la distribution minimale *`miniconda`*.

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 encore installé Python sur ton PC, ou si tu ne l'as pas installé avec *`Anaconda`*, tu peux installer simplement *`miniconda`* (voir https://docs.conda.io/en/latest/miniconda.html) pour continuer.

Pour travailler avec les modules Keras et tensorflow sur OS X, GNU/Linux ou Windows, il est conseillé d'utiliser la version 3.6 de Python.

Créer un environnement Python 3.6 dédié

Toutes les commandes qui suivent (cadre avec fond jaune pâle) sont à taper :
– dans un terminal pour GNU/Linux ou Mac OS ;
– dans une fenêtre *prompt Anaconda* pour Windows.

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

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

– `conda` créé un dossier `pyml` dans le dossier `..quelque_part../Anaconda3_ou_miniconda3/envs` ,

– puis `conda` télécharge et installe les dossiers et les fichiers nécessaires dans le dossier `..quelque_part../Anaconda3_ou_miniconda3/envs/pyml`.

Pour rendre `jupyter notebook` compatible avec les environnements Python dédiés, il faut tout de suite installer le paquet `nb_conda_kernels` (ceci va également installer d'autres paquets Python requis) :

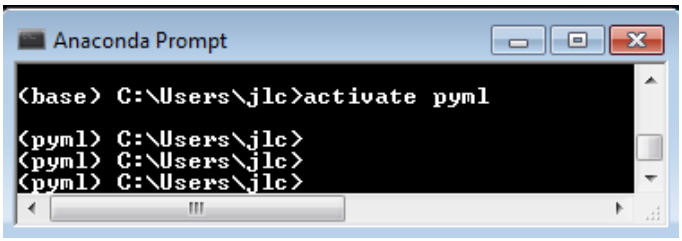
```
conda install -n pyml nb_conda_kernels
```

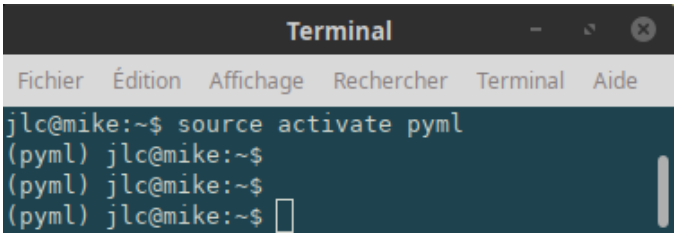
Activer un environnement Python dédié

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



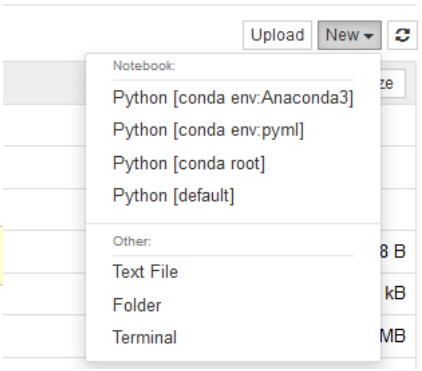


Utiliser jupyter-notebook dans un environnement Python dédié

Une fois dans l'environnement Python dédié marqué par lepréfixage du prompt par **(pyml)**, tu peux lancer **jupyter notebook** en tapant la commande `jupyter notebook`.
`Windows` : Si tu es avec un PC sous Windows, tu peux, si besoin ajouter un argument donnant le **chemin du dossier de travail** de jupyter en tapant comme commande :

```
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 et le choix `[conda env:pyml]` permet de travailler dans l'environnement dédié Python `pyml` (*cf* figure ci-dessus).

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

```
import sys
sys.version
```

```
'3.6.5 |Anaconda, Inc.| (default, Apr 29 2018, 16:14:56) \n[GCC 7.2.0]'
```

► **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 ».**

Calculer avec jupyter-notebook sous Python 3.6 :

Une fois retrouvé le terminal (ou la fenêtre *prompt Anaconda)* avec **l'environnement `pyml` activé**, installer avec `conda` les modules nécessaires au calcul scientifique :

```
(pyml) jlc@mikde conda install numpy scipy matplotlib
```

Installer ensuite les autres modules avec la commande pip :
par exemple pour les calculs de Machine Learning avec OpenAI/gym :

```
(pyml) jlc@mikde pip install gym tensorflow keras pydot h5py pyhamcrest
```

ou les calculs avec Kera et tensorflow :

```
(pyml) jlc@mikde pip install gym tensorflow keras pydot h5py pyhamcrest
```

« conda install » ou « pip install » ?

La stratégie est simple :

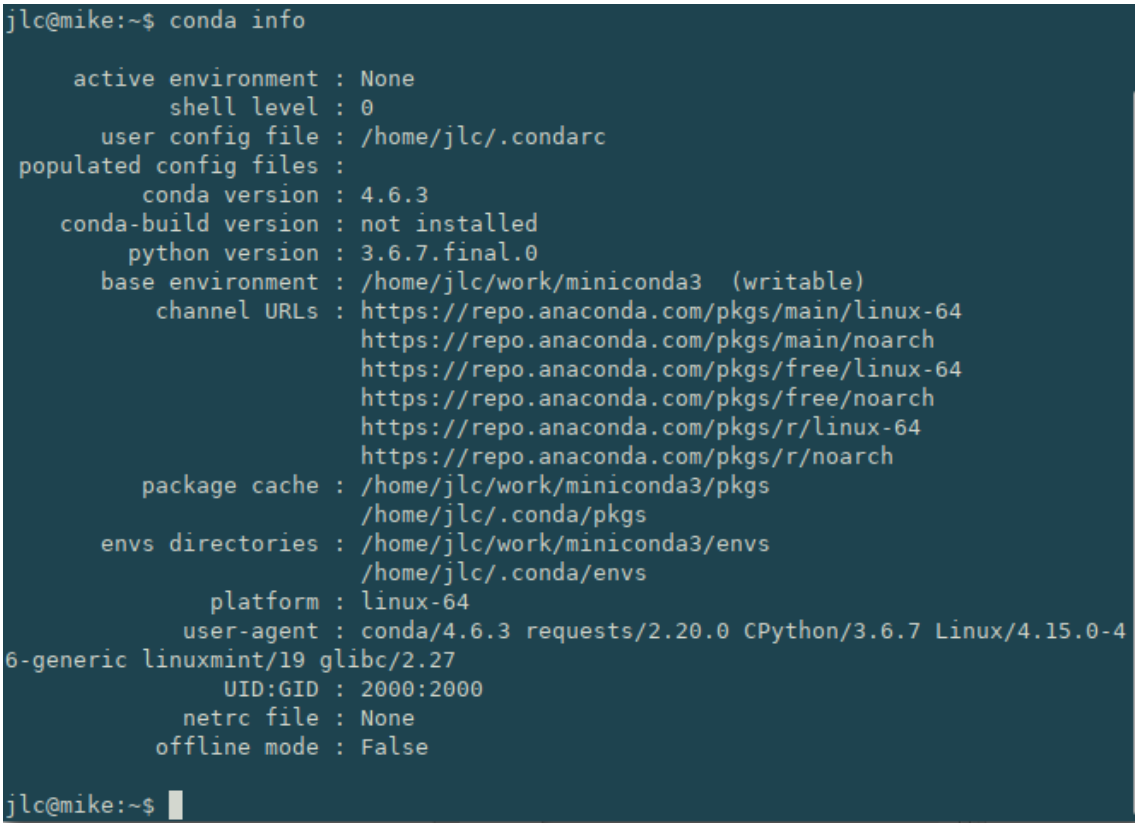
1/ **Essayer toujours en premier « `conda install ...` »** : par exemple « conda `install numpy` » permet d'installer le module numpy avec la **bibliothèque hyper optimisée MKL d'INTEL**, alors que « `pip install numpy` » installe ces bibliothèques d'algèbre linéaire dans des versions moins optimisées.

2/ **Si le module n'est pas installable avec conda, essayer l'installation avec pip**.

How To…

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

```
conda info
```



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
source deactivate   # GNU/Linux ou Mac 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 → source deactivate
```

puis le supprimer pour de bon :

```
conda remove -n pyml --all
```

et y'a plus qu'à tout recommencer ;-)