Support de formation complet

Au-delà de la formation, pour avoir plus de connaissances

Youssef MOURCHID

youssef.mourchid@isen-ouest.yncrea.fr

Lancer Anaconda Prompt :

- Via l'icône Windows
- Ou par le `Terminal` :

```
C:\>%windir%\System32\cmd.exe "/K"
            C:\Users\<Nom_Utilisateur>\anaconda3\Scripts\activate.bat
            C:\Users\<Nom_Utilisateur>\anaconda3
```

Ce qui permet d'obtenir l'Anaconda Prompt avec l'activation de l'environnement de base `anaconda3`

```
(anaconda3) C:\>
```

Celui-ci indique l'environnement de "base" d'anaconda.

Remarque

- le répertoire d'installation d'Anaconda est accessible via le terminal grâce à `echo %CONDA_PREFIX%`
- depuis un fichier `.BAT` on lancera `call C:\Users\<Nom_Utilisateur>\anaconda3\Scripts\activate.bat C:\Users\<Nom_Utilisateur>\anaconda3`

Connaître la version de conda et les chemins d'installation

Depuis `Anaconda Prompt` :

```
(anaconda3) C:\> conda -V
conda 4.8.3

(anaconda3) C:\> where conda
C:\Users\isen\anaconda3\condabin\conda.bat
C:\Users\isen\anaconda3\Scripts\conda.exe

(anaconda3) C:\> where python
D:\Recherche\lsl_team_project\Environment\environment_LSL\python.exe
C:\Users\isen\anaconda3\python.exe
```

Lister les commandes disponibles :

conda : {clean, config, create, info, list...}

```
(base) C:\> conda --help      # ou conda -h
```

conda env : {create, export, list, remove, update, config...}

```
(base) C:\> conda env -h
```

Obtenir la liste des environnements virtuels :

```
(base) C:\> conda env list
```

L'environnement estampillé "*" est l'environnement actif :

```
# conda environments:
#
base                  *  C:\Users\isen\anaconda3
envTEST                  C:\Users\isen\anaconda3\envs\envTEST
```

Créer un environnement vide :

```
(base) C:\> conda create -n Nom_Env
```

L'environnement virtuel est alors placé (emplacement par défaut) dans :

```
C:\Users\ [NomUtilisateur] \Anaconda3\envs\Nom_Env
```

Charger (activer) un environnement

```
(base) C:\> conda activate Nom_Env
```

Par défaut, conda chargera l'environnement situé dans :

```
C:\Users\ [NomUtilisateur] \Anaconda3\Nom_Env
```

L'invite de commandes du terminal affichera alors le nom de l'environnement virtuel actif :

```
(Nom_Env) C:\>
```

Remarque l'activation d'un environnement revient à placer cinq répertoires relatifs à cet environnement en tête dans le path. Depuis l'anaconda prompt :

```
(anaconda3) C:\> path
PATH = C:\Users\isen\anaconda3; C:\Users\isen\anaconda3\Library\mingw-w64\bin;
C:\Users\isen\anaconda3\Library\usr\bin; C:\Users\isen\anaconda3\Library\bin;
C:\Users\isen\anaconda3\Scripts;C:\Users\isen\anaconda3\bin; C:\Users\isen\anaconda3
\condabin; C:\WINDOWS\system32; C:\WINDOWS; C:\WINDOWS\System32\Wbem;
C:\WINDOWS\System32\WindowsPowerShell\v1.0; C:\WINDOWS\System32\OpenSSH;
C:\Program Files\Git\cmd;

(anaconda3) C:\> conda activate Nom_Env

(Nom_Env) C:\> path
PATH = C:\Users\isen\anaconda3\envs\Nom_Env; C:\Users\isen\anaconda3
\envs\Nom_Env\Library\mingw-w64\bin; C:\Users\isen\anaconda3
\envs\Nom_Env\Library\usr\bin; C:\Users\isen\anaconda3\envs\Nom_Env\Library\bin;
C:\Users\isen\anaconda3\envs\Nom_Env\Scripts; C:\Users\isen\anaconda3
\envs\Nom_Env\bin; C:\Users\isen\anaconda3\condabin; C:\WINDOWS\system32;
C:\WINDOWS; C:\WINDOWS\System32\Wbem; C:\WINDOWS\System32
\WindowsPowerShell\v1.0; C:\WINDOWS\System32\OpenSSH; C:\Program Files\Git\cmd;
```

Désactiver un environnement

```
(Nom_Env) C:\> conda deactivate
```

Par défaut, conda chargera l'environnement de base :

```
(base) C:\>
```

Créer un nouvel environnement et installer des packages

```
(base) C:\> conda create -n Nom_Env python=3.7.6 numpy=1.18.5
```

On remarque qu'il est possible de spécifier la version exacte que l'on souhaite. Si tout se passe bien, l'installation se termine en affichant :

```
# To activate this environment, use
#     $ conda activate Nom_Env
# To deactivate an active environment, use
#     $ conda deactivate
```

Installer des packages à un environnement existant

Activer l'environnement choisi, puis installer les packages correspondants :

```
(base) C:\> conda activate Nom_Env
```

```
(Nom_Env) C:\> conda install scipy
```

On aurait également pu spécifier la version et le channel souhaité pour le téléchargement (un channel désigne un emplacement (URL) où sont stockées les packages. Par défaut, `-c defaults`, packages issus de `https://repo.anaconda.com/pkgs/`, tandis que `conda-forge` désigne un channel communautaire, doté de milliers de contributeurs) :

```
(Nom_Env) C:\> conda install -c conda-forge scipy=1.4.1
```

Il est aussi possible de préciser la version et le numéro de build (précompilation) souhaité :

```
(Nom_Env) C:\> conda install -c conda-forge numpy=1.18.5=py37h6530119_0
```

Lister les packages installés dans l'environnement actif:

```
(Nom_Env) C:\> conda list
```

Lister les packages disponibles dans un repository distant :

```
(Nom_Env) C:\> conda search numpy
```

En précisant dans quel channel supplémentaire aux channels par défaut on veut rechercher :

```
(Nom_Env) C:\> conda search -c conda-forge numpy
```

En précisant sur quelle type de machine on travaille (par défaut, plateforme courante):

```
(Nom_Env) C:\> conda search -c conda-forge --platform linux-64 numpy   #win-32, osx-64,linux-32,w
```

Plus d'options :

```
(Nom_Env) C:\> conda search -h
```

Supprimer un environnement

Une fois désactivé, on peut suppprimer un environnement. Cela reviendra à supprimer :

- le répertoire contenant l'environnement : par défaut, dans `C:\Users\<Nom_Utilisateur>\Anaconda3\envs\Nom_Env`, ou bien le répertoire personnalisé qui a été spécifié (option `--prefix` ou `-p`)
- la ligne correspondante dans le fichier texte qui liste les environnements conda accessibles. Il se situe dans : `C:\Users\<Nom_Utilisateur>.conda\environments.txt`

```
(base) C:\> conda env list

# conda environments:
#
base                  *  C:\Users\isen\anaconda3
Nom_Env                  C:\Users\isen\anaconda3\envs\Nom_Env

(base) C:\> conda env remove -n Nom_Env

(base) C:\> conda env list

# conda environments:
#
base                  *  C:\Users\isen\anaconda3
```

Répliquer (cloner) un environnement

Il n'est pas possible de renommer un environnement, mais on peut le cloner :

```
(Nom_Env) C:\> conda env list

# conda environments:
#
base                     C:\Users\isen\anaconda3
Nom_Env               *  C:\Users\isen\anaconda3\envs\Nom_Env

(Nom_Env) C:\> conda create -n Nom_Env_Copie --clone Nom_Env
Source:      C:\Users\isen\anaconda3\envs\Nom_Env
Destination: C:\Users\isen\anaconda3\envs\Nom_Env_Copie
Executing transaction: done
# To activate this environment, use
#     $ conda activate Nom_Env_Copie
# To deactivate an active environment, use
#     $ conda deactivate

(Nom_Env) C:\> conda env list

# conda environments:
#
base                     C:\Users\isen\anaconda3
Nom_Env               *  C:\Users\isen\anaconda3\envs\Nom_Env
Nom_Env_Copie            C:\Users\isen\anaconda3\envs\Nom_Env_Copie
```

Partager un environnement : fichier YAML

- **Objectif**: pouvoir partager un environnement, pouvoir reproduire un environnement sur une autre machine, avec les mêmes packages dans des versions identiques.
- **Export d'un fichier YAML décrivant les packages installés sur une machine donnée (dépendant de l'OS):**

  https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#sharing-an-environment

  Activer l'environnement à partager, et créer un fichier YAML contenant la description des librairies et des dépendances de cet environnement :

```
(Nom_Env) C:\> conda env export --file Description_Nom_Env.yml
```

  Le fichier `Description_Nom_Env.yml` possède la structure suivante :

```
name: Nom_Env                    # Nom
channels:
  - conda-forge                   # We added a third party channel
  - defaults
dependencies:
  - numpy=1.16.3=py37h926163e_0   # nom = version = build
  - pandas=0.24.2=py37h0a44026_0
  - pip=19.1.1=py37_0
  - vs2015_runtime=14.16.27012=hf0eaf9b_3
  - pip:                            # Packages installed from PyPI
    - requests==2.21.0
prefix: C:\Users\<Nom_Utilisateur>\anaconda3\envs\Nom_Env #chemin d'installation
```

- Reconstruction d'un environnement à partir d'un fichier YAML (sur une machine de même OS que celle qui a généré le .yml):

```
(base) C:\> conda env create -n New_Env --file Description_Nom_Env.yml

Solving environment: done
# To activate this environment, use
#     $ conda activate New_Env
# To deactivate an active environment, use
#     $ conda deactivate
```

- Mise à jour d'un environnement à partir d'un fichier YAML mis à jour:

```
(base) C:\> conda env create -n New_Env --file Description_Nom_Env.yml

(base) C:\> conda env update -n New_Env --file Description_Nom_Env_update.yml
```

- **Fichier YAML et gestion des dépendances : une solution pour la gestion des environnements multi-plateformes**

  Depuis l'environnement actif, la commande `conda env export --file Description_Environnement.yml` permet de recenser toutes les bibliothèques qui ont été installées dans cet environnement. Très souvent, cela comprend bien sûr les bibliothèques dont on a besoin, mais aussi de nombreuses bibliothèques additionnelles, requises sur la machine sur laquelle on travaille. En fait, ces dépendances dépendent de l'architecture (32 ou 64 bits), ainsi que de l'OS (macOS, Windows, Linux).

  Pour pouvoir être complètement portable et compatible entre les plateformes, il est impératif de ne pas inclure les dépendances liées à un OS, et de laisser le solveur conda résoudre et lister les dépendances nécessaires pour que les packages nécessaires puissent fonctionner. La commande `conda env export --from-history --file Description_Environnement.yml` permet de réaliser ceci.

Exemple :

Supposons que l'on crée un environnement Env_SIMPLE dans lequel on installe python=3.7 et numpy :

```
(base) C:\> conda env create -n Env_SIMPLE python=3.7 numpy

(base) C:\> conda activate Env_SIMPLE

(Env_SIMPLE) C:\> conda env export >env.yml

(Env_SIMPLE) C:\> type env.yml
 name: Env_SIMPLE
 channels:
   - defaults
 dependencies:
   - intel-openmp=2020.1=216
   - [...]
   - numpy=1.18.5=py37h6530119_0
   - numpy-base=1.18.5=py37hc3f5095_0
   - pip=20.1.1=py37_1
   - python=3.7.7=h81c818b_4
   - [...]
   - vs2015_runtime=14.16.27012=hf0eaf9b_3
 prefix: C:\Users\isen\anaconda3\envs\Env_SIMPLE
```

De nombreuses librairies additionnelles sont installées : ainsi `vs2015_runtime` est-elle spécifique aux OS windows, tandis que les autres librairies sont des versions précompilées pour windows (numéro de build).

Au contraire, avec l'option `--from-history`, le fichier YAML ne recense que les besoins initiaux, et sera déployable sur toute plateforme :

```
(Env_SIMPLE) C:\> conda env export --from-history >envFH.yml

(Env_SIMPLE) C:\> type envFH.yml
 name: Env_SIMPLE
 channels:
   - defaults
 dependencies:
   - python=3.7
   - numpy
 prefix: C:\Users\isen\anaconda3\envs\Env_SIMPLE
```

Inconvénient : les librairies installées avec `pip` ne sont pas prises en compte

- **Autre solution:** toujours écrire le fichier YAML à la main, et laisser le solveur conda résoudre les dépendances en fonction de la plateforme. Ceci est d'ailleurs une recommandation (https://github.com/conda/conda-env/#environment-file-example).

```yaml
name: D:\Programmes\Environments\env_LSL_Team_Project
channels:
  - conda-forge
  - defaults
dependencies:
  - python          = 3.7
  - numpy           = 1.18.5
  - scipy           = 1.3.1
  - pandas          = 1.0.5
  - gdal            = 3.0.4
  - geopandas       = 0.8.0
  - keras           = 2.3.1
  - scikit-learn    = 0.23.1
  - shapely         = 1.7.0
  - py-opencv       = 4.3.0
  - matplotlib      = 3.2.1
  - jupyter         = 1.0.0
  - spyder          = 4.1.3
  - pip
  - pip:
    - streamlit == 0.62.1
prefix: D:\Programmes\Environments\env_LSL_Team_Project
```

librairies scientifiques

IA

graphes

IDE

Pour cela, l' option `--name Nom_Environnement` ou `-n Nom_Environnement` doit être remplacée par `--prefix D:\Repertoire\Nom_Environnement` ou `-p D:\Repertoire\Nom_Environnement` .

- Création `conda env create -p D:\Repertoire\Nom_Environnement`
- Activation `conda activate D:\Repertoire\Nom_Environnement`
- Export de YAML `conda env export --from-history -p D:\Repertoire\Nom_Environnement -f fichier_env.yml`
- Installation de librairie lors de la création `conda env create -p D:\Repertoire\Nom_Environnement python=3.7`
- Mise à jour d'environnement `conda env update -p D:\Repertoire\Nom_Environnement -f Description_Nom_Env_update.yml`
- Supprimer un environnement `conda env remove -p D:\Repertoire\Nom_Environnement`

conda et pip

https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#using-pip-in-an-environment

Si l'utilisation de pip dans un environnement conda est requise, il est recommandé :

- de n'utiliser pip qu'après conda
- si des modifications (mise à jour ou ajout de librairies conda) sont nécessaires après une installation avec pip, recréer un nouvel environnement dans lequel on terminera par les installations pip. En effet, une fois que pip est utilisé, conda ne pourra en tracer les changements.

Chronologie des environnements.

La notion de **revision** permet de remonter dans l'historique des installations effectuées, et éventuellement, de retourner à un état antérieur :

```
(Env_SIMPLE) C:\> conda list --revisions
```

```
2020-06-10 19:20:37  (rev 10)
    +affine-2.0.0.post1
    +click-6.6
    +click-plugins-1.0.3
    +cligj-0.4.0
    +rasterio-0.35.1
    +snuggs-1.3.1

2020-06-10 20:10:19  (rev 11)
     libpng  {1.6.17 -> 1.6.22}

2020-06-10 07:25:49  (rev 12)
    -gdal-2.1.0
```

La sortie permet de voir une liste de revisions de cet environnement, la date de création, et les différences constatées (installations signalées par un "+", désinstallations par un "-", mises à jour par "->").

Il est possible de revenir à un état antérieur grâce à la commande `conda install --revision N` (où N désigne la revision choisie).

En réalité, revenir à un état antérieur nécessite les opérations opposées des installations qui ont eu lieu depuis lors.

Exemple

```
(Env_SIMPLE) C:\> conda list --revisions

     2020-06-14 11:12:34  (rev 1)
         +mkl-11.3.3
         +numpy-1.17.0
         +pandas-0.18.1
         +python-dateutil-2.5.3
         +pytz-2016.4
         +six-1.10.0

     2020-06-14 07:13:08  (rev 2)
         +cycler-0.10.0
         +freetype-2.6.3
         +libpng-1.6.22
         +matplotlib-1.6.1
         +pyparsing-2.1.4

(Env_SIMPLE) C:\> conda install --revision 1

[...]
```

```
(Env_SIMPLE) C:\> conda list --revisions

     2020-06-14 07:13:08  (rev 2)
         +cycler-0.10.0
         +freetype-2.6.3
         +libpng-1.6.22
         +matplotlib-1.6.1
         +pyparsing-2.1.4

     2020-06-14 21:15:45 (rev 3)
         -cycler-0.10.0
         -freetype-2.6.3
         -libpng-1.6.22
         -matplotlib-1.6.1
         -pyparsing-2.1.4
```

Maîtriser la taille sur le disque

Nettoyer l'espace disque

```
(base) C:\> conda clean --all
```

Remove index cache, lock files, unused cache packages, and tarballs.

Automatiser l'activation d'environnement et le chargement d'un IDE

Ressource : https://medium.com/@divjyotsinghlearn/automate-conda-commands-ec551684f4f2

Pour cela, il suffit de créer un fichier `.BAT` contenant :

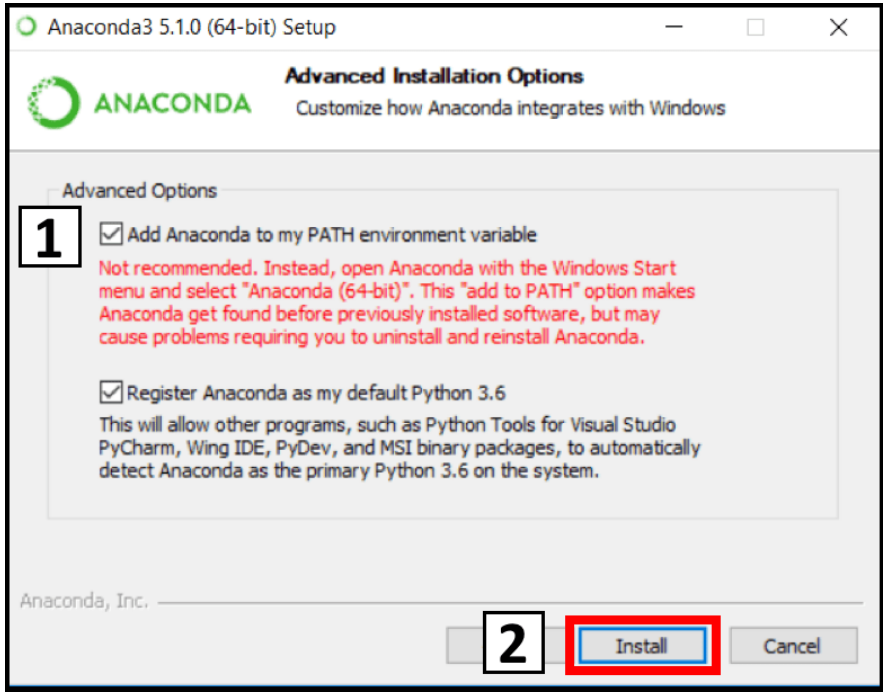
```
call C:\Users\%USERNAME%\anaconda3\Scripts\activate.bat D:\<...>\Nom_Env_A_Activer

call spyder -w D:\<...>\Repertoire_De_Travail
REM ou call jupyter notebook
```

Annexes

Installation de conda - PATH

- https://www.datacamp.com/community/tutorials/installing-anaconda-windows

- Si vous avez déjà une version de Python d'installée, Conda a dû vous demander lors de l'installation s'il fallait mettre la version de Conda par défaut.
- Cela consiste à redéfinir le PATH pour y ajouter le dossier bin de Conda.



Source : https://www.datacamp.com/community/tutorials/installing-anaconda-windows

Ajout manuel dans le PATH

- Ressource : https://medium.com/@GalarnykMichael/install-python-on-windows-anaconda-c63c7c3d1444
- Il s'agit d'ajouter les répertoires contenant le python système et conda.exe .

  Pour les connaître, ouvrir **`Anaconda Prompt`** et taper :

  - **`where python`**

    `C:\Users\ <NomUser> \anaconda3`
  - **`where conda`**

    `C:\Users\ <NomUser> \anaconda3\Scripts`

Option 1 : modification des variables d'environnement

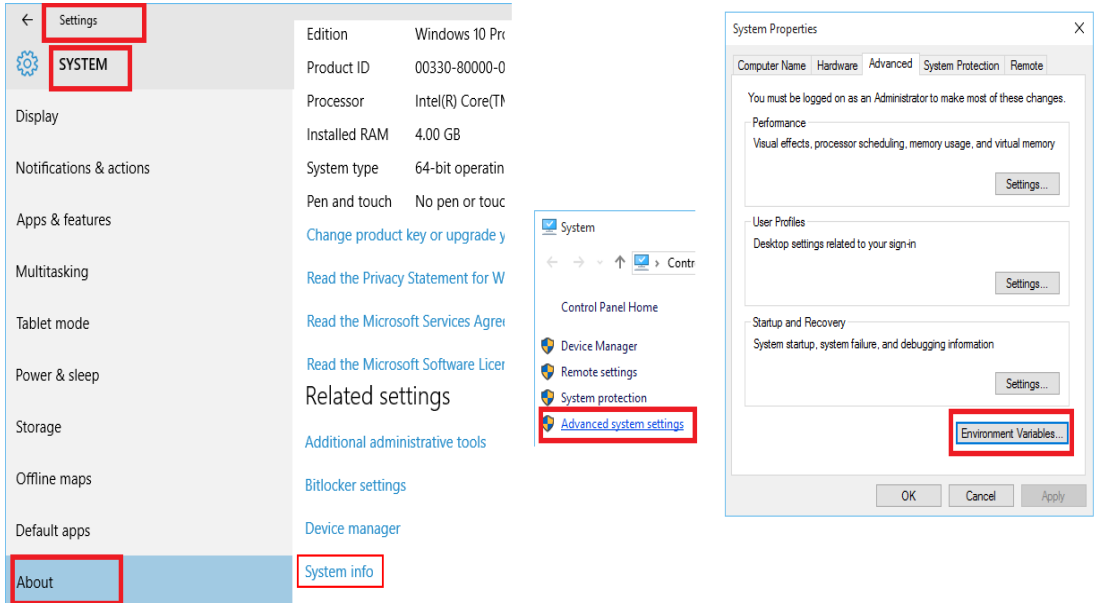
Paramètres>Informations Système>Paramètres Associés>Informations système>Paramètres systèmes avancés>Variables d'environnement

Edit Path et ajouter les deux répertoires à la fin (séparés par des ;)

Option 2 : depuis le **`command prompt`**, taper :

`C:\>SETX PATH "%PATH%";C:\Users\ <NomUser> \anaconda3\Scripts;C:\Users\ <NomUser> \anaconda3`

Fermer le **`prompt`**, et en ouvrir un autre. Vérifier en tapant :
`PATH`



Ressources

Ressources

- The Definitive Guide to Conda Environments
- Getting started with Python environments (using Conda)
- A Quick and Easy Guide to Managing Conda Environments

- Why You Need Python Environments and How to Manage Them with Conda
- Save the environment with conda (and how to let others run your programs)

- Sharing an environment
- How to share conda environments across platforms
- Combining conda environment.yml with pip requirements.txt

- Automate AnaConda Environment setup — using Scripts

- Conda vs. pip vs. virtualenv commands
- Conda: Myths and Misconceptions