Guide de référence pour ISD

Contents

Qui est quoi ? Hein ?

Descriptions rapides des outils principaux

Anaconda, Conda et Jupyter sont des outils essentiels dans le monde de Python pour la science des données :

Anaconda

- Anaconda est une plateforme open-source populaire pour la science des données et l'apprentissage automatique.
- Il inclut une distribution de Python ainsi qu'un ensemble complet de bibliothèques et d'outils préinstallés, ce qui facilite le démarrage des projets de science des données.
- Anaconda fournit également un gestionnaire de paquets appelé Conda, qui aide à gérer les **paquets Python et les environnements**.

Conda

- Conda est un gestionnaire de paquets et un gestionnaire d'environnements multiplateforme.
- Il permet aux data scientists de créer des environnements Python isolés, chacun avec son propre ensemble de paquets, de dépendances et de versions de Python.
- Conda simplifie le processus d'installation, de mise à jour et de gestion des paquets et des bibliothèques, garantissant la compatibilité entre différents environnements de projet.

Jupyter

- Jupyter est un environnement de calcul interactif qui vous permet de créer et de partager des documents contenant du code en direct, des équations, des visualisations et du texte narratif.
- Le cahier Jupyter (Jupyter Notebook) est une interface populaire au sein de l'écosystème Jupyter couramment utilisée en science des données.
- Les data scientists utilisent les cahiers Jupyter pour écrire et exécuter du code Python de manière interactive, ce qui facilite l'exploration des données, le prototypage d'algorithmes et la création de rapports basés sur les données.

En résumé, Anaconda fournit une distribution Python pratique pour la science des données, Conda aide à gérer les paquets et les environnements, et Jupyter offre un environnement interactif et visuel pour l'analyse et l'exploration des données. Ces outils sont essentiels pour tout data scientist travaillant avec Python.

Conda vous permet aussi de créer rapidement et facilement des environnements virtuels avec la version de Python que vous désirez, ce qui permet d’isoler vos différents projets qui n’utiliseraient pas les mêmes librairies ou versions.

Procédure d'installation : Anaconda et Jupyter notebook

Intro et installation d'Anaconda

Dans cette partie vous allez :

1. Télécharger et installer Anaconda ici : https://www.anaconda.com/products/distribution. Suivez les instructions et gardez les options par défaut. **Ne définissez pas Anaconda comme version de Python par défaut et n'ajoutez rien au PATH**. Nous utiliserons la console « *Conda prompt* » fournie (Windows). Cette étape peut prendre du temps (**10-20 minutes** selon votre configuration et utilisation). Sous MacOS et Linux, Anaconda (et conda) sont accessible normalement via le terminal.
2. En suite vous allez créer un environnement virtuel pour les TPs et l'ajouter en tant que Kernel Jupyter (page suivante).

   Les étapes sont décrites dans ce document mais si besoin, la doc vous donnera plein d'infos plus précises, et pour faire court : Cette vidéo vous explique en moins de 10 minutes comment utiliser Anaconda et Jupyter notebook, ainsi que comment pouvoir utiliser le bon kernel dans vos nouveaux notebooks, car Jupyter ne va pas ajouter chaque environnement virtuel automatiquement aux Kernels disponibles.

Lancer le **Anaconda Prompt** qui a été installé avec Anaconda. C'est depuis cette console que vous aurez accès à Anaconda (pas depuis CMD, Powershell ou autre, en tout cas dans le contexte de ce cours et avec Windows).

1 : Environnements virtuels

Créer un environnement virtuel. Il s'agit simplement d'un répertoire créé automatiquement (dans *~Anaconda3/envs*) dans lequel il y aura un exécutable python et les libraires installées. Pour chaque environnement, un nouveau répertoire est créé. Nommez cet environnement « isd ».

```
(base) C:\Users\...> conda create --name isd python=3.9
```

Activer cet environnement dans la console Anaconda Prompt va sélectionner cet exécutable et donc permettre d'installer et d'utiliser les librairies qui y sont présente.

```
(base)  C:\Users\...>  conda activate isd
```

L'environnement actif apparait maintenant entre parenthèse dans la console :

```
(isd) C:\Users\...>
```

Jupyter est installé dans l'environnement de base mais pas encore dans celui-ci. En effet vous venez de créer un environnement vide ! Pour pouvoir lancer l'application, faites :

```
(isd) C:\Users\...> conda install notebook
```

Vous pourrez ainsi démarrer Jupyter et lancer les notebooks depuis votre environnement de base, ou depuis l'environnement isd.

Note : Peu importe d'où vous lancez le notebook, vous pourrez de toute manière sélectionner l'environnement désiré si celui-ci a été « inscrit » dans la liste des kernels comme ci-après.

Juste pour tester : Lancez Jupyter avec la commande ci-dessous. Jupyter s'ouvre normalement automatiquement dans le navigateur, sinon copiez le lien donné dans la console.

```
(isd) C:\Users\...>  jupyter notebook
```

L'interface web va automatiquement s'ouvrir là d'où vous avez lancé la commande. Vous pouvez vous déplacer avant depuis la console en utilisant la commande « cd *répertoire*» ou plus tard en utilisant l'interface web.

Lorsque vous aurez lancé le notebook, vous pouvez naviguer dans vos fichiers et créer un notebook avec "New" dans le répertoire courant. A cette étape, vous avez uniquement "Python3" dans les kernels proposés et ceci seulement si vous avez un exécutable Python dans votre path.

Pour avoir accès à votre environnement virtuel créé ci-dessus, **vous devez l'ajouter en tant que Kernel dans Jupyter**. Quittez le notebook, soit en faisant *ctrl+c* dans la console soit en appuyant sur « Quit » en haut à droite de la page web puis suivez les étapes de la page suivante.

2 : Kernel Jupyter

Le kernel Jupyter est le processus qui va exécuter le code du notebook. Dans notre cas, il s'agira de l'exécutable python provenant de l'environnement virtuel créé plus tôt.

Toujours dans **l'Anaconda Prompt dans l'environnement virtuel isd (ou autre nom)**

```
(isd) C:\Users\...>  pip install ipykernel
```

Une fois ipykernel installé, on peut ajouter notre environnement en tant que kernel jupyter. Ici, le kernel jupyter aura le même nom que notre environnement.

```
(isd) C:\Users\...>  python -m ipykernel install --user --name isd
```

Vous pouvez relancer Jupyter depuis la console (Anaconda prompt) et créer votre notebook où bon vous semble. L'environnement "isd" devrait maintenant être disponible.

```
(isd) C:\Users\...>  jupyter notebook
```

Dans la liste des kernels disponibles, on voit maintenant apparaitre notre environnement virtuel. Tout ce qui sera installé à l'intérieur de cet environnement, sera dorénavant accessible dans le notebook. Pour installer un nouveau package (par exemple numpy, pandas et matplotlib) dans notre environnement **isd**, il suffit dans un terminal de faire :

```
(isd) C:\Users\...>  conda install numpy
(isd) C:\Users\...>  conda install matplotlib
(isd) C:\Users\...>  conda install pandas
```

Vous pouvez aussi directement installer les packages suivants dont vous aurez besoin plus tard :

```
(isd) C:\Users\...>  conda install scikit-learn
(isd) C:\Users\...>  conda install seaborn
```

Alternativement on peut aussi installer des packages avec **pip**, selon leurs disponibilités, car pas tous les packages sont dans les dépots de Conda.

Pour quitter/stopper Jupyter

- Depuis l'interface graphique, appuyez sur "quit" en haut à droite.
- Depuis la console faites *ctrl+c* et confirmez si nécessaire.

Maintenant que votre environnement est présent dans la liste des kernels Jupyter, vous n'êtes pas obligé d'activer l'environnement « isd » à chaque fois si vous souhaitez simplement lancer le notebook. Vous pouvez simplement le lancer depuis votre environnement de base :

```
(base) C:\Users\...>  jupyter notebook
```

Suppression et réinstallation rapide d'un environnement

Il peut arriver que l'installation d'un package cause des conflits dans un environnement et que plus rien ne fonctionne. En cas de soucis, vous pouvez réinstaller rapidement le tout à l'aide des commandes ci-dessous. Les noms d'environnement et de kernel sont ceux utilisé dans la partie ci-dessus.

1. Supprimer entièrement l'environnement virtuel Anaconda

```
(base)  C:\Users\... > conda remove --name isd --all
```

2. Vérifier la liste des kernels puis supprimer le kernel Jupyter associé à l'environnement

```
(base)  C:\Users\... > jupyter kernelspec list
```

```
(base)  C:\Users\... > jupyter kernelspec uninstall isd
```

3. Recréer l'environnement virtuel

```
 base) C:\Users\...> conda create --name isd python=3.11
(base) C:\Users\...> conda activate isd
```

4. Installer les packages demandés

```
(isd) C:\Users\...>  pip install ipykernel
(isd) C:\Users\...>  pip install gapminder
(isd) C:\Users\...>  conda install numpy
(isd) C:\Users\...>  conda install pandas
(isd) C:\Users\...>  conda install matplotlib
(isd) C:\Users\...>  conda install scikit-learn
```

5. Ajouter l'environnement en tant que kernel Jupyter

```
(isd) C:\Users\...>  python -m ipykernel install --user --name isd
```

6. Désactiver l'environnement une fois que tout est terminé et relancez votre serveur jupyter

```
(isd) C:\Users\...> conda deactivate
```

```
(base) C:\Users\...> jupyter notebook
```