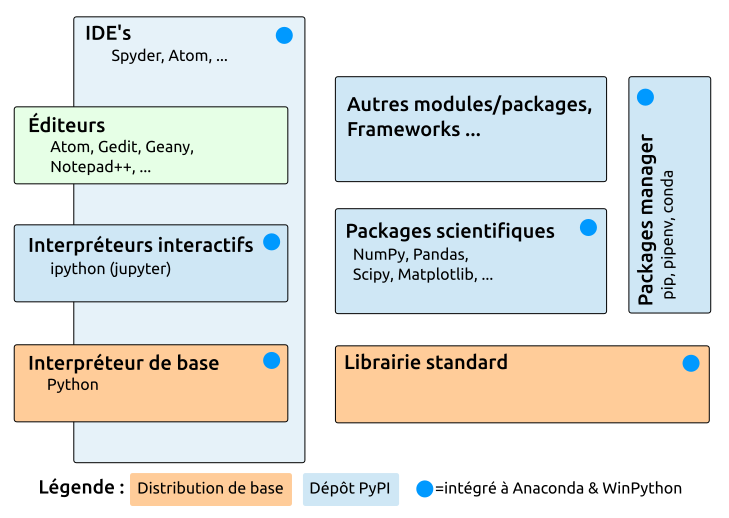
TP 0 : Mise en place de l’environnement de travail python

## Objectifs du TP :

Préparer votre environnement de travail pour les projets data science

* Installer le langage de programmation python
* Installer un environnement de développement
* Créer et utiliser des environnements virtuels python

L’utilisation du langage python et de ses nombreuses bibliothèques est incontournable pour la réalisation des projets du Data science. Pour bien démarrer, il est important de préparer un environnement python complet et fonctionnel. Un environnement de développement Python est constitué de différents outils et composants présentés dans la figure ci-dessous.



Source image : https://www.jdbonjour.ch/cours/python/introduction/outils-python.html

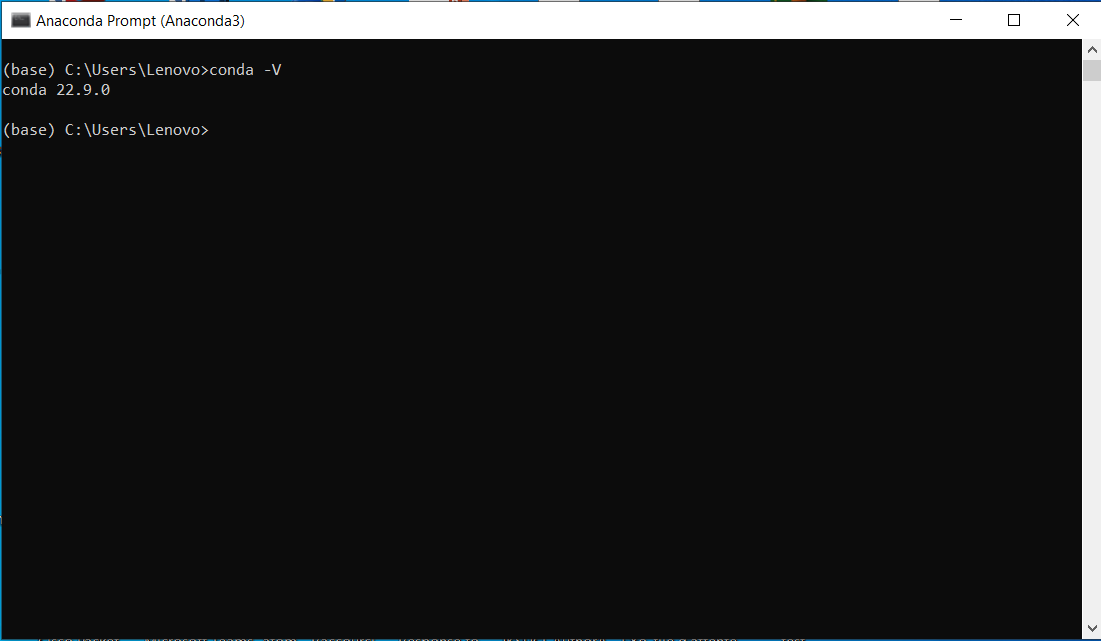
Anaconda est une solution open source destinée à la programmation python et R. L’utilisation de Anaconda est recommandée car il met à la disposition des développeurs tous l’écosystème de la data science et Machine Learning. Il permet l’installation de Python avec tous les paquets nécessaires, les notebook jupyter et Spyder.

## Installation d’Anaconda

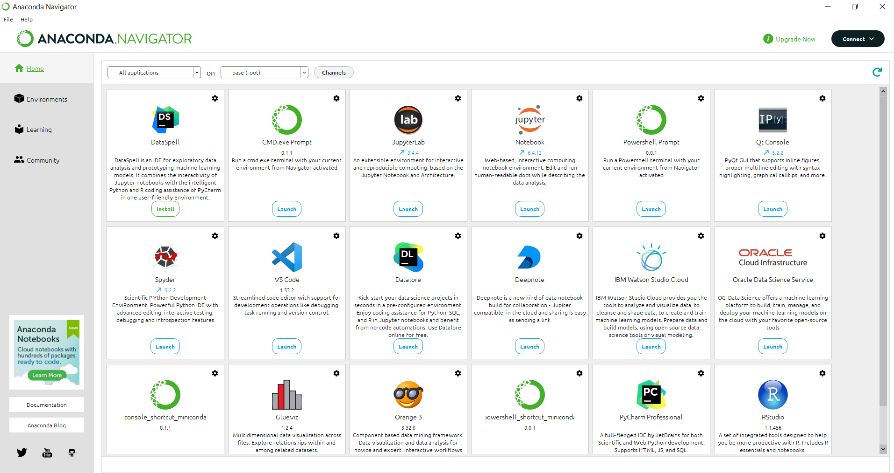
Les distributions Anaconda sont téléchargeables depuis le site officiel : <https://www.anaconda.com/>. L’installation est intuitive, vous pouvez suivre les étapes pour une installation par défaut.

La distribution Anaconda est livrée avec :

* L’outil de ligne de commande **Anaconda prompt**



* Une interface utilisateur graphique de bureau appelée **Anaconda Navigator**.



* Les packages de base couramment utilisés en data science. La liste complète est disponible sur <https://docs.anaconda.com/anaconda/packages/pkg-docs/>
* Plusieurs interfaces de développement, dont les plus utilisés sont :
  + Les *notebooks Jupyter* sont très utilisés en data science. Ils permettent de mêler du code, du texte et des sorties graphiques ou des tableaux sur la même interface. L’intérêt principal des notebooks est qu’ils permettent d’écrire, d’exécuter et de partager du code dans la même interface utilisateur.
  + JupyterLab est un IDE basé en principe sur les notebooks jupyter. Il permet de travailler avec plusieurs notebooks ou fichiers ouverts dans des onglets de la même fenêtre. C’est le successeur officiel de Jupyter Notebook, et doit à long terme le remplacer.

Pour lancer Jupyter, utiliser l’interface Anaconda Navigator ou les commandes suivantes avec Anaconda prompt :

* **Jupyter lab** pour lancer jupyterLab
* **Jupyter notebook** pour lancer jupyter notebook

## Création d’environnements conda

## De façon analogue à la technique virtualenv, il est possible sur anaconda de définir des environnements virtuels distincts avec leurs propres versions Python, leurs propres modules/packages.

## Quelques commandes principales pour la création des environnements via Anaconda prompt:

* Créer un environnement avec le nom env\_name :

**conda create --name env\_name**

* Créer l’environnement avec définition de la version Python :

**conda create -n env\_name python=3.8**

* Importer un environnement à partir d’un fichier

**conda env create -f environment.yml**

* Activer et utiliser l’environnement crée :

**conda activate env\_name**

* Quitter l’environnement :

**conda deactivate**

* Exporter un environnement :

**conda env export > environment.yml**

* Supprimer un environnement conda :

**conda env remove -n env\_name**

* Afficher les listes des environnements créé avec conda :

**conda env list**

**conda info –envs**

Plus de détails sur les commandes pour la gestion des environnements sont disponibles dans la documentation : [https://docs.conda.io/projects/conda/en/4.6.1/user-guide/tasks/manage-environments.html](https://docs.conda.io/projects/conda/en/4.6.1/user-guide/tasks/manage-environments.html%20)

La création est aussi possible via l’interface graphique Anaconda Navigator.

## Gestion des packages avec conda

Anaconda dispose de son propre gestionnaire de package que l'on invoque avec la commande **conda**. La différence entre *conda* et le gestionnaire de package *pip* réside dans la gestion des dépendances du package.

- Quand *pip* installe un package, il installe automatiquement tous les packages Python dépendants sans vérifier si ceux-ci entrent en conflit avec les packages précédemment installés.

- *conda* analyse l’environnement actuel, y compris tout ce qui est actuellement installé, et, avec toutes les limitations de version spécifiées, explique comment installer un ensemble compatible de dépendances, et affiche un avertissement si cela ne peut pas être fait.

Pour installer un package, utiliser la commande ci-dessous en remplaçant package par le nom du package.

**conda install package\_name**

## Installation des libraries python pour la data science

Python dispose de nombreuses librairies pour effectuer des tâches de calcul, de manipulation et d’exploration des données, et de la visualisation.

Les bibliothèques les plus utilisée en data science et machine learning sont :

* **numpy** : la bibiliothèque de base sur laquelle se base toutes les librairies de manipulation et d’analyse des données. Elle se caractérise par des capacités de calcul robustes, ce qui est très utile pour l’analyse des données massives.
* **pandas** : une bibliothèque qui facilite la manipulation des données, en utilisant les Dataframes.
* **matplotlib** et seaborn : des bibliothèques utilisées pour la visualisation des données.
* **scikit-learn** : la principale bibliothèque d'outils du machine learning en Python.
* **keras** : la bibliothèque du deep learning en Python.