Procédure d'installation de « Python » et utilisation sous « Windows »

Auteurs : François Bertin et Amal Chabli
Date : 07/12/2021

Sommaire

I. Matériels considérés et préparation

Le descriptif ci-après concerne :

- Les PC sous Windows 10.#.# personnels ;
- Les PC sous Windows 10.#.# sous gestion centralisée CEA .

Néanmoins, tout ou partie des étapes de la procédure dépend du type de matériel. Les variantes sont précisées pour chaque type.

La préparation consiste à identifier les dossiers qui vont être utilisés pour l'installation. La procédure est basée sur l'arborescence de la figure 1.

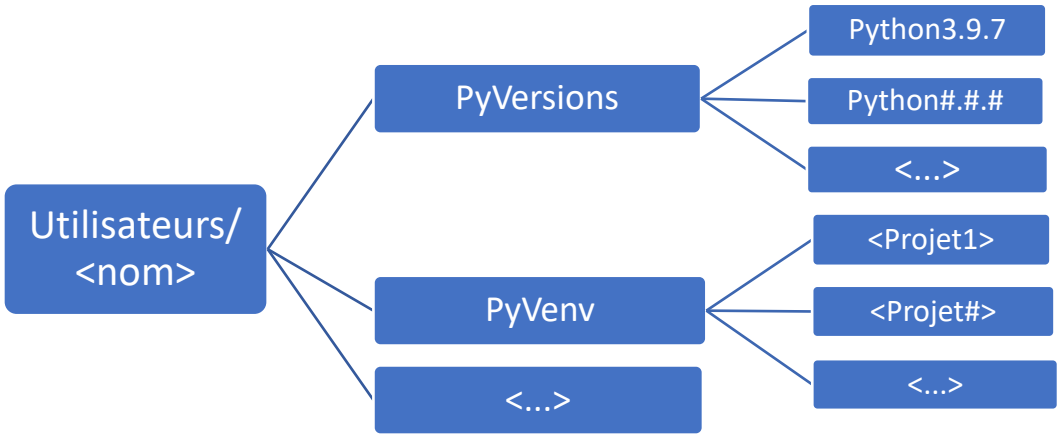

Figure 1 : Arborescence requise pour l'installation d'une version de Python.

Description des dossiers

- **Utilisateurs/<nom>** : Dossier racine de l'utilisateur ex : C:\Users\ac112303
  - **PyVersions** : Dossier à créer qui va contenir les dossiers de différentes versions de Python pouvant être utilisées
    - **Python3.9.7** : Dossier à créer qui va contenir l'ensemble des fichiers résultant de l'installation de la version python3.9.7
    - **Python#.#.#** : Dossier créé contenant l'ensemble des fichiers résultant de l'installation de la version python#.#.#
    - **<…>** : indique la présence d'autres dossiers préexistants du même type que **Python#.#.#**
  - **PyVenv** : Dossier à créer qui va contenir les dossiers de projets déposés sur Git Hub utilisant des environnements virtuels (venv)
    - **<Project1>** ou **<Project#>** : dossier à créer qui va accueillir les éléments (*jupyter notebooks*, dossiers de *packages,* fichiers spécifiques à un dossier *Git Hub*) issus du clonage du dossier de dépôt *GitHub* (GitHub *repository)* ainsi qu'un dossier **venv** que nous décrivons dans la suite (§.IV)
    - **<…>** : indique la présence d'autres dossiers préexistants du même type que **<Project#>**
  - **<…>** : indique la présence d'autres fichiers et dossiers préexistants dont l'installation de Python est indépendante

II. Accès à une version spécifique de Python

La dernière version de python disponible en téléchargement (onglet « **Downloads** ») en date du **1er septembre 2021** sur le site officiel de Python :

https://www.python.org/

La version **python3.9.7** est celle qui est généralement visée par ce document.

Toutes les versions antérieures restent également disponibles. Il existe deux versions majeures de Python : **python2** et **python3**. Ces deux versions ne sont pas compatibles entre elles. La compatibilité descendante est assurée pour les versions python3.#.#. Autrement dit un programme développé sous python3.6.# tournera pour toutes les versions ultérieures (python3.7.#, python3.8.# …). Mais à l'inverse **un programme développé sous python3.9.# ne tournera pas forcément sous des versions antérieures**. C'est au programmeur qu'incombe de gérer les limites de compatibilité.

L'installation n'est possible via les exécutables d'installation proposé par le site que sur les PCs sous Windows hors gestion centralisée CEA. Pour les autres types de matériel, la procédure s'appuie sur la disponibilité, sur dispositif de stockage externe ou mis à disposition via FileSender (si possible compatible MacOS et Windows), d'un dossier contenant l'ensemble des fichiers résultant de l'installation effectuée sur un PC sous Windows hors gestion centralisée CEA.

1. Installation sur PC sous Windows 10.x.x personnels

1. Télécharger l'exécutable d'installation de la version ad-hoc **python3.#.#** depuis le site python **python.org**
2. Lancer l'exécutable et suivre les instructions en choisissant l'option **« installation personnalisée »** ;
3. Sélectionner le dossier pour le stockage de la version téléchargée, soit :

   Utilisateurs\<nom>\PyVersions\python3.#.#

4. Terminer l'installation : un ensemble important de fichiers et dossiers sont maintenant présents dans le dossier ci-dessus.

2. Installation sur PC sous Windows 10.x.x sous gestion centralisée CEA

1. Copier le dossier **Python3.#.#** du dispositif de stockage externe ou mis à disposition via FileSender.
2. Coller ce dossier **Python3.#.#** dans le dossier :

   Utilisateurs\<nom>\PyVersions

III. Clone d'un projet python disponible sur GitHub

Pour mettre en place un clone de projet disponible sur GitHub, il est nécessaire :

- d'installer l'outil Git qui permet de travailler en local sur les dossiers de GitHub ;
- de créer un compte sur GitHub (double authentification nécessaire).

De plus, la disponibilité d'un projet sur GitHub doit être connue au préalable. Il y a plusieurs possibilités dont :

- La recherche sur le réseau via des mots clefs concernant le sujet d'intérêt
- La communication directe du nom de projet par l'auteur ou un utilisateur

1. Mise en place de l'accès à GitHub

1. Télécharger l'exécutable d'installation de Git depuis le site suivant :

   https://git-scm.com/downloads

2. Fermer toutes les fenêtres de commandes éventuellement ouvertes
3. Lancer l'exécutable et suivre les instructions d'installation (vérifier que le dossier d'installation est bien sous **C:\Users\<nom>** pour les PC sous infogérance CEA).
4. Créer un compte sur GitHub (*Accompagnement nécessaire)* :

   https://github.com/

2. Clone du projet <Project#>

1. Ouvrir une fenêtre de commande via :
   - le menu « démarrer » ;
   - la barre de raccourcis si le raccourci a été mis en place ;
   - sous jupyter (sélectionner « **Terminal** » dans le menu « **New** »)

   Le « prompt » dans la fenêtre de commande est alors :

   C:\Users\<nom>>

1. Se positionner dans le dossier d'accueil des projets clonés avec environnement virtuel en exécutant la commande :

   >cd PyVenv

   Le « prompt » dans la fenêtre de commande est alors :

   C:\Users\<nom>\PyVenv>

2. Se loguer sous son compte GitHub (via **Sign in**)
3. Aller sur le dossier de dépôt portant le nom **<Project#>**
4. Dans l'onglet « **Code** » copier l'url **<url-du-projet>**
5. Dans la fenêtre de commande exécuter la commande :

   >git clone <url-du-projet>

   Le dossier **<Project#>** du projet est alors disponible en local selon l'arborescence décrite au paragraphe I.

IV. Création d'un environnement virtuel de travail d'un projet python

Pour un fonctionnement pérenne et transportable d'un utilisateur à l'autre, chaque projet python **<Project#>** doit comporter la définition d'un environnement virtuel dit **venv** qui lui est propre. Cet environnement se présente sous la forme d'un dossier qui contient l'ensemble des versions des modules python qui sont nécessaires au projet. Il convient de le créer avant de le configurer.

La procédure pour la création d'un environnement virtuel de travail du projet python **<Project#>** est la suivante **:**

1. Ouvrir une fenêtre de commande (via le menu « démarrer » ou la barre de raccourcis si le raccourci a été mis en place)

   Le « prompt » dans la fenêtre de commande est alors :

   C:\users\<nom>>

2. Cloner le dossier de dépôt voulu depuis GitHub dans le dossier **<Project#>** (cf § 0)

3. Créer le dossier d'environnement virtuel **venv** dans le dossier projet **<Project#>** avec utilisation de la version **python3.#.#** en exécutant la commande :

   >C:\users\<nom>\PyVersions\python3.#.#\python -m venv C:\users\<nom>\PyVenv\<Project#>\venv

4. Se placer dans le dossier **C:\users\<nom>\PyVenv\<Project#>\venv** en exécutant la commande :

   >cd PyVenv\<Project#>\venv

   Le « prompt » dans la fenêtre de commande devient :

   C:\users\<nom>\PyVenv\<Project#>\venv>

5. Activer l'utilisation de cet environnement virtuel en exécutant la commande :

   >Scripts\activate

   Le « prompt » dans la fenêtre de commande devient :

   (venv) C:\users\<nom>\PyVenv\<Project#>\venv>

6. Examiner la liste des modules installés en éditant le contenu du dossier d'environnement **venv** du dossier **<Project#>** en exécutant la commande :

   >pip freeze

   La liste est vide si aucune installation de module n'a encore été faite volontairement.

V. Installation des modules requis pour un projet python

L'environnement virtuel d'un projet python est spécifique à chaque utilisateur, il convient de l'ignorer lors des interactions sur GitHub. Par contre une version de son contenu validé peut-être partagé (en particulier sur GitHub) par l'intermédiaire d'un fichier texte communément nommé **requirements.txt**.

La procédure pour installer les modules requis pour la bonne exécution des applications du projet python **<Project#>** est la suivante **(Attention l'étape d'activation de l'environnement virtuel est incontournable)** :

1. Ouvrir une fenêtre de commande (via le menu « démarrer » ou la barre de raccourcis si le raccourci a été mis en place)

   Le « prompt » dans la fenêtre de commande est alors :

   C:\users\<nom>>

2. Se placer dans le dossier **PyVenv\<Project#>\venv** en exécutant la commande :

   >cd PyVenv\<Project#>\venv

   Le « prompt » dans la fenêtre de commande devient :

   C:\users\<nom>\PyVenv\<Project#>\venv>

3. Activer l'utilisation de cet environnement virtuel en exécutant la commande :

   >Scripts\activate

   Le « prompt » dans la fenêtre de commande devient :

   (venv) C:\users\<nom>\PyVenv\<Project#>\venv>

   Il est conseillé de nommer le noyau du même nom que le projet **<Project#>**.

4. Si un fichier **requirements.txt** est déjà associé au projet **<Project#>** (par exemple sur GitHub), le copier dans le dossier du projet **<Project#>,** soit ici :

   C:\users\<nom>\PyVenv\<Project#>

5. Importer le contenu de **requirements.txt** dans l'environnement virtuel local en exécutant la commande :

   >pip install -r C:\users\<nom>\PyVenv\<Project#>\requirements.txt

6. Installer le module **ipykernel** qui gère la liste des noyaux reconnus par jupyter de façon globale en exécutant la commande :

   >pip install ipykernel

   Cette installation peut déjà avoir été prévue dans le fichier **requirements.txt**. Alors **pip** informe que le module est déjà disponible (pas d'impact).

7. Ajouter le noyau, nommé **<kernel_name>** associé à l'environnement virtuel créé, à la liste des noyaux reconnus par jupyter de façon globale en exécutant la commande :

   >python -m ipykernel install --name=<kernel_name>

8. Installer le module **notebook** qui permet de lancer des environnements de travail jupyter en exécutant la commande :

   >pip install notebook

   Cette installation peut déjà avoir été prévue dans le fichier **requirements.txt**. Alors **pip** informe que le module est déjà disponible (pas d'impact).

   On peut également choisir un environnement de travail du type **jupyter lab** en exécutant :

   >pip install jupyterlab

Cette installation peut déjà avoir été prévue dans le fichier **requirements.txt**. Alors **pip** informe que le module est déjà disponible (pas d'impact).

9. Ouvrir une fenêtre **jupyter** ou **jupyter lab** depuis une fenêtre de commande en exécutant l'une des instructions :

   >jupyter notebook

   >jupyter-lab

10. Sélectionner le dossier du projet **<Project#>** dans la fenêtre jupyter qui s'est ouverte dans le navigateur.
11. Sélectionner dans le menu « **New** » (en haut à droite de la fenêtre) l'environnement virtuel portant le nom **<kernel_name>** dans lequel les applications python du projet **<Project#>** vont s'exécuter sous jupyter.

    Le nom de l'environnement sera indiqué dans l'entête de la fenêtre **jupyter notebook** qui s'ouvre pour éditer une application (fichier avec extension « **.ipynb** ») du projet.

12. Identifier les modules manquants en testant l'exécution de toutes les applications python du projet **<Project#>**.
13. Installer les modules requis en exécutant dans la fenêtre de commande de l'environnement virtuel du projet **<Project#>** la commande suivante pour chaque module **<module>** de la liste des modules identifiés :

    >pip install <module>

14. Répéter les étapes 10 et 11 tant que l'exécution des applications python du projet **<Project#>** indique des modules manquants.
15. Sauvegarder la configuration de l'environnement virtuel dans le fichier **requirements.txt** local en exécutant la commande :

    >pip freeze > C:\users\<nom>\PyVenv\<Project#>\requirements.txt

16. En fin de travail sur le projet, il est conseillé de désactiver l'environnement virtuel du projet **<Project#>** en exécutant la commande :

    >deactivate

VI. Installation d'Anaconda (pour mémoire, à éviter)

Différents environnements de développement en python sont possibles.
La procédure ci-après concerne **Anaconda**, une application open source utilisée entre autres pour :

- Lancer des environnements de développement utiles pour construire, tester, gérer et utiliser des applications en python dont :
    - L'application **Jupyter** qui fonctionne en utilisant un navigateur web sur un serveur local (localhost, l'ordinateur se parle à lui-même, pas de faille de sécurité) ;
    - L'environnement de développement **Spyder** (similaire à celui de Matlab).
- D'ouvrir des fenêtres de commande (Powershell et Anaconda)

Procédure d'installation

1. Télécharger l'exécutable d'installation d'**Anaconda** depuis le site suivant :

   https://www.anaconda.com

   La version pour Windows au 1er septembre 2021 est la suivante :

   64-Bit Graphical Installer (477 MB)

   L'exécutable est stocké dans le répertoire **Téléchargement**.

2. Lancer l'exécutable (en double cliquant sur son icone)

   Une interface graphique s'active. Un répertoire d'installation vous sera proposé par défaut. En principe sur Windows ce répertoire est :

   C:\Utilisateurs\<nom>

   Une fois l'installation terminée **Anaconda** apparaît dans la liste de vos programmes. Un raccourci d'accès peut être créé.

Remarques

- Il existe aussi une version « 32 bit » d'installation d'Anaconda. Pour vérifier le système d'exploitation de l'ordinateur, allez sur **Paramètres/Système/Informations système**.
- L'installation d'Anaconda comporte également l'installation de Python et **la version proposée n'est pas forcément la dernière ni celle qui est destinée aux projets d'applications python à développer ou utiliser**. La gestion des versions de python est décrite dans la suite.

ØØØ