Python - Gérer plusieurs versions de Python sous Windows avec pyenv-win

[Mise à jour le : 19/2/2025]

- **Sources**
  - **Github** : pyenv-win
- **Ressources**
  - **Real Python** : Managing Multiple Python Versions With pyenv

---

1. Pourquoi utiliser pyenv-win ?

pyenv-win est la version Windows de pyenv. pyenv est un excellent outil pour **gérer plusieurs versions de Python**. Même si Python est déjà installé sur votre système, il est interressant d'installer pyenv afin de pouvoir facilement tester de nouvelles fonctionnalités du langage ou contribuer à un projet qui utilise une version différente de Python. L'utilisation de pyenv est également un excellent moyen d' installer des versions préliminaires de Python afin de pouvoir les tester pour détecter les bugs.

2. Installer pyenv-win

> **POWERSHELL**
> Utiliser **powershell** en administrateur.

1. Vérifier que powershell autorise l'exécution des scripts.

   *.powershell

   ```
    Get-ExecutionPolicy
   # Si restricted faire
     Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope
   CurrentUser
   ```

2. Installez pyenv-win

   *.powershell

```
Invoke-WebRequest -UseBasicParsing -Uri
"https://raw.githubusercontent.com/pyenv-win/pyenv-win/master/pyen
v-win/install-pyenv-win.ps1" -OutFile "./install-pyenv-win.ps1";
&"./install-pyenv-win.ps1"
```

3. Rouvrir PowerShell

4. Exécutez **pyenv --version** pour vérifier si l'installation a réussi.

*.powershell

```
pyenv --version
# Exemple de résultat : pyenv 3.1.1
```

3. Installer plusieurs versions de Python

1. Exécutez **pyenv install -l** pour obtenir la liste des versions de Python prises en charge par pyenv-win.

*.powershell

```
PS C:\Users\phili> pyenv install -l
:: [Info] ::  Mirror: https://www.python.org/ftp/python
:: [Info] ::  Mirror:
https://downloads.python.org/pypy/versions.json
:: [Info] ::  Mirror:
https://api.github.com/repos/oracle/graalpython/releases
2.4-win32
2.4.1-win32
2.4.2-win32
...
```

2. Exécutez **pyenv install** <version> pour installer la version prise en charge.

*.powershell

```
pyenv install 3.10.11
# Résultat ->
# :: [Info] ::  Mirror: https://www.python.org/ftp/python
# :: [Info] ::  Mirror:
https://downloads.python.org/pypy/versions.json
# :: [Info] ::  Mirror:
https://api.github.com/repos/oracle/graalpython/releases
# :: [Downloading] ::  3.10.11 ...
# :: [Downloading] ::  From
```

```
https://www.python.org/ftp/python/3.10.11/python-3.10.11-amd64.exe
# :: [Downloading] ::  To   C:\Users\phili\.pyenv\pyenv-win\install_cache\python-3.10.11-amd64.exe
# :: [Installing] ::  3.10.11 ...
```

4. Version globale, version locale

> **GLOBAL**
> Avec pyenv, la version globale de Python est celle qui sera utilisée par défaut pour tous les projets, sauf si une version locale ou spécifique est définie.

- **Définir une version globale**
  Exécutez **pyenv global** <version> pour définir une version Python comme version globale.

  *.powershell

  ```
  # Exemple
  pyenv global 3.12.7
  ```

- **Définir une version locale**
  Exécutez **pyenv local** <version> pour définir une version locale de Python pour un projet spécifique.

  Dans le répertoire **du projet**, exécutez :

  *.powershell

  ```
  # Exemple
  pyenv local 3.10.11
  ```

- **Vérifications**
  1. Vérifiez quelle version de Python vous utilisez et son chemin

     *.powershell

     ```
     pyenv version
     # Résultat -> 3.12.7 (set by C:\Users\phili\.pyenv\pyenv-win\version)
     ```

  2. Vérifiez que Python fonctionne

*.powershell

```
python -c "import sys; print(sys.executable)"
# Résultat -> C:\Users\phili\.pyenv\pyenv-win\versions\3.12.7\python.exe
```

1. Restreindre l'exécution des scripts.

*.powershell

```
Set-ExecutionPolicy Restricted
```

3. Les bibliothèques

- **Position dans l'arborescence des répertoires**

> Si l'environnement virtuel est dans ***C:\Users\NomUtilisateur\.pyenv\pyenv-win\versions\3.x.x***, alors les bibliothèques installées avec pip seront dans :
>
> - ***C:\Users\NomUtilisateur\.pyenv\pyenv-win\versions\3.x.x\Lib\site-packages***

4. Liste des commandes

> **Préfixe**
> Sous Windows, les commandes sont utilisées dans **powershell** et préfixée par **pyenv**.

Exemple

*.powershell

```
pyenv commands # pour lister les commandes
```

- **commands** : liste toutes les commandes pyenv disponibles
- duplicate
- **exec** : Exécute un exécutable en préparant d'abord PATH afin que le répertoire `bin' de la version Python sélectionnée soit au début
- export

- **global** : définit ou affiche la version globale de Python
- **help** : affiche l'aide pour une commande
- **install** : installe une ou plusieurs versions de Python
- **local** : définit ou affiche la version locale de Python spécifique à l'application
- **rehash** : réorganise les cales pyenv (exécuter cette opération après avoir changé de version Python)
- **shell** : définit ou affiche la version de Python spécifique au shell
- **shims** :
- **uninstall** : désinstalle une ou plusieurs versions de Python
- **update** : met à jour la base de données de versions en cache
- **version-name** : affiche la version Python actuelle
- **version** : affiche la version Python actuelle et son origine
- **versions** : Liste toutes les versions Python disponibles pour pyenv
- **vname** : affiche la version Python actuelle
- **whence** : liste toutes les versions Python qui contiennent l'exécutable donné
- **which** : affiche le chemin complet vers un exécutable

5. pyenv et VSCODE

- Si VSCode s'exécute dans un autre compte que celui de l'administrateur, voir IDE VSCode - Généralités