Environnements Virtuels Python

Qu’est-ce qu’un environnement virtuel Python et pourquoi l’utiliser ?

L’un des grands changements lors du passage de Python 2 à Python 3 est l’utilisation des environnements virtuels (venv).
L’idée est simple : dans un environnement virtuel, vous pouvez installer un ensemble de bibliothèques exclusives à cet environnement.
Les venvs évitent les problèmes de compatibilité (moins de soucis avec Python) et facilitent énormément les opérations de support (résolution plus rapide des problèmes).

Diviser pour régner

Lorsque vous travaillez dans un venv, vous limitez le nombre de paquets utilisés. Lister (`pip list`) et vérifier (`pip check`) est bien plus facile et rapide avec un nombre réduit de paquets.

Pour le support, vous pouvez créer une “carte d’identité” de votre environnement Python, qui est simplement la liste des paquets installés. Plus cette liste est petite, plus le support pourra rapidement recréer exactement le même état que vous, même sur une autre machine.

Comment installer un environnement virtuel ?

Prérequis

Commencez par installer Python 3, vous trouverez comment faire sur python.org/downloads.
Vérifiez que vous utilisez la bonne version de Python avec la commande `which python`.

python -m venv

La commande pour installer un environnement virtuel est :

```
python -m venv -awesome_new_environement-
```

Exemple concret

Voici un exemple avec le venv **totoro** créé dans un dossier **python_venvs** à la racine :

```
Copier le code
mylogin@mymachine:~>python -m venv ~/python_venvs/totoro
mylogin@mymachine:~>ls totoro
bin/    include/    lib/    pyvenv.cfg
```

Cette commande crée un dossier `totoro` contenant trois sous-dossiers et un fichier : `bin/`, `include/`, `lib/` et `pyvenv.cfg`.

Maintenant que vous avez créé votre environnement, vous devez l'activer avant d'installer les bibliothèques nécessaires.

Pour activer votre environnement virtuel, utilisez le script `activate` dans le venv :

```
Copier le code
mylogin@mymachine:~>source ~/totoro/bin/activate
(totoro) mylogin@mymachine:~>
```

Vous devriez maintenant voir le nom de votre environnement virtuel au début de la ligne de commande.

Vous pouvez installer toutes les bibliothèques nécessaires :

```
Copier le code
(totoro) mylogin@mymachine:~>pip install numpy
(totoro) mylogin@mymachine:~>pip install scipy
```

Vous pouvez quitter votre environnement virtuel et revenir à l'environnement par défaut avec **deactivate** :

```
Copier le code
(totoro) mylogin@mymachine:~>deactivate
mylogin@mymachine:~>
```

Les paquets disponibles dans **totoro** ne sont alors plus visibles. Essayez `pip list` pour vérifier.

Problèmes courants

Les environnements virtuels peuvent rencontrer des soucis.
Voici quelques bonnes pratiques.

Dès la création de votre environnement virtuel, **mettez à jour pip et setuptools** :

```
Copier le code
(totoro) mylogin@mymachine:~>pip install --upgrade pip
(totoro) mylogin@mymachine:~>pip install --upgrade setuptools
```

Ces deux bibliothèques évoluent rapidement, gardez-les à jour pour limiter les problèmes.

N’utilisez jamais l’option `--user` de pip. Celle-ci place certaines dépendances dans un emplacement utilisateur. En cas de support, ce “truc” peut vous faire perdre des heures.

Faites attention aux actions tierces qui modifient votre `$PYTHONPATH`. Cette variable doit être gérée uniquement par votre venv.
Les sources de problèmes fréquentes sont :

1. `module load ****` : les modules modifient souvent les variables d’environnement. Un `module load python/3.8.2` exécuté dans un venv est une balle empoisonnée.
2. Les scripts shell, en particulier ceux liés au profil, modifient souvent vos variables d’environnement. Si une action touche aux outils Python, il est probable qu’elle mette à jour `$PYTHONPATH`.

Installer correctement un paquet

Une fois votre environnement virtuel en place, vous pourriez avoir besoin d'installer des paquets accessibles via git, par exemple, et vouloir les installer ainsi que leurs dépendances.
La bonne pratique est d'éviter `python setup.py install` et `python setup.py develop`, ces deux commandes peuvent mener à des dépendances non installées dans votre environnement virtuel.
Une méthode plus sûre consiste à utiliser directement `pip install` dans le dossier du dépôt.

Pour cela, assurez-vous d'être dans le même dossier que le setup.py du paquet à installer et utilisez l'une de ces deux commandes :

`pip install .` si vous voulez installer le paquet Python en mode installation. Cette commande va d'abord chercher les dépendances, récupérer les fichiers sur pypi, puis installer le tout pour avoir un paquet fonctionnel. Ce n'est qu'après cette étape que la commande python setup.py install est exécutée ;

`pip install -e .` si vous voulez installer le paquet Python en mode développement.

Une collection de tutoriels et motivations sur les environnements virtuels

L'environnement virtuel n'est pas une invention de COOP, mais "la manière recommandée de maîtriser votre framework Python".
Il peut être rapproché de vos premiers contacts avec les variables d'environnement UNIX (`.bashrc`, `module load`, etc.).
Voici quelques liens pour des explications non-COOP :

- python.org : la référence absolue sur les venvs.
- realpython : un tutoriel plus accessible.
- geeksforgeeks : une présentation encore plus courte des venvs.
- towardsdatascience : un article non technique expliquant POURQUOI utiliser un venv.

Créer un alias pour activer automatiquement votre venv

Si vous utilisez régulièrement le venv totoro, vous pouvez créer un alias dans votre fichier `.bashrc` afin de l'activer plus rapidement.

Pour cela, on ajoute cette ligne à la fin de notre fichier `~/.bashrc` :

```
alias totoro="source ~/python_venvs/totoro/bin/activate"
```

Ensuite, on recharge votre configuration :

```
source ~/.bashrc
```

À partir de maintenant, il suffit de taper :

```
totoro
```

Pour activer automatiquement l'environnement virtuel **totoro** et la commande