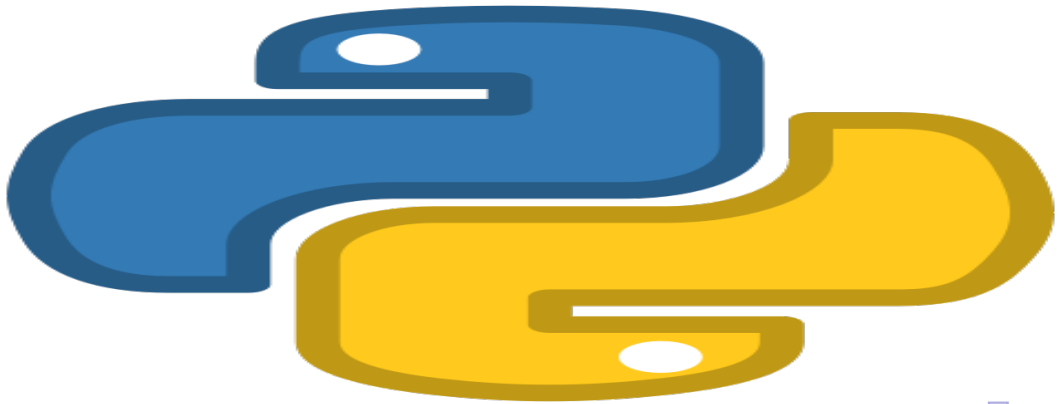
Python : Pip, pyenv et venv

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com



Plan

Python

PyPI : **Py**thon **P**ackage **I**ndex

- Dépôt de packages, officiel et principal, pour le langage **Python**.
- Contenant une grande variété de packages open-source pour divers usages
- Lien vers le dépôt : `https://pypi.org/`
- Tous les packages **PyPI** sont gérés par **PIP**

Python

PIP : **P**ackage **i**nstaller for **P**ython

- Gestionnaire de packages/modules par défaut pour **Python**
- Utilisable en ligne de commandes
- Utilisé pour installer, mettre à jour et gérer les packages **Python**

Python

Et **easy_install** ?

- Ancien gestionnaires de paquets pour **Python** fourni avec **setuptools**
- Utilise également **PyPI**
- A été officiellement marqué comme obsolète et ne doit plus être utilisé

Python

Pour connaître la version de `pip`

```
pip --version
```

Python

Pour connaître la version de `pip`

```
pip --version
```

Toutes les commandes `PIP` peuvent être exécutées de la manière suivante

```
python -m pip [suite]
```

Python

Pour lister les packages installés (en global avec Python)

```
pip list
```

Python

Pour lister les packages installés (en global avec Python)

```
pip list
```

Pour avoir plus de détails sur un package installé

```
pip show nom_package
```

Python

Pour installer un package

```
pip install nom_package
```

Python

Pour installer un package

```
pip install nom_package
```

Pour installer plusieurs packages

```
pip install p1 p2 p3
```

Python

Pour installer un package

```
pip install nom_package
```

Pour installer plusieurs packages

```
pip install p1 p2 p3
```

Pour désinstaller un package

```
pip uninstall nom_package
```

Python

Pour chercher un package

```
pip search nom_package
```

Python

Pour chercher un package

```
pip search nom_package
```

Pour lister les packages installés

```
pip list
```

Python

Deux types de package que l'on peut installer avec `pip`

- **Bibliothèques et Frameworks** : destinés à être importés et utilisés dans les propres projets Python. Ils peuvent offrir des fonctionnalités allant des
  - traitements et opérations (comme **NumPy**)
  - aux frameworks web complets (comme **Django**, **Fast API** ou **Flask**)
- **Outils de développement et Utilitaires** : fournissent des fonctionnalités qui aident au développement, au déploiement, ou à la maintenance de projets **Python**, mais ne sont généralement pas importés dans le code source.

Python

Exemple : installons `numpy`

```
pip install numpy
```

Python

Exemple : installons `numpy`

```
pip install numpy
```

Pour l'utiliser, commençons par l'importer

```
import numpy as np
```

Python

Exemple : installons `numpy`

```
pip install numpy
```

Pour l'utiliser, commençons par l'importer

```
import numpy as np
```


Utilisons le pour calculer et afficher la somme des éléments du tableau

```
arr = np.array([1, 2, 3, 4, 5])

sum_result = np.sum(arr)
print("Somme des éléments du tableau :", sum_result)
```

Python

Pour générer un exécutable de notre application

- Installer le module `pip install pyinstaller`
- Lancer la commande `pyinstaller.exe --onefile main.py`
- Un fichier `main.exe` a été généré dans le répertoire `dist`.

Python

Exemple : installons `mypy`

```
pip install mypy
```

Python

Exemple : installons `mypy`

```
pip install mypy
```

Considérons un fichier `tester_types.py` **ayant le contenu suivant**

```
def somme(a: int, b: int) -> int:
    return a + b

x: str = '2'
y: int = '3'

print(somme(x, y))
```

Python

Exemple : installons `mypy`

```
pip install mypy
```

Considérons un fichier `tester_types.py` ayant le contenu suivant

```
def somme(a: int, b: int) -> int:
    return a + b

x: str = '2'
y: int = '3'

print(somme(x, y))
```

Lancez le code et vérifiez le résultat suivant

```
23
```

Python

Lancez la commande suivante pour vérifier les incohérences de type

```
mypy tester_types.py
```

Résultat

```
tester_types.py:5: error: Incompatible types in assignment
(expression has type "str", variable has type "int")  [assignment]
tester_types.py:7: error: Argument 1 to "somme" has incompatible type "
   str";
expected "int"  [arg-type]
Found 2 errors in 1 file (checked 1 source file)
```

Python

Pour analyser tous les fichiers du répertoire courant et ses sous-répertoires avec `mypy`

```
mypy .
```

Python

Remarque

Avant de commencer, désinstaller **Python** globalement (depuis **Applications et fonctionnalités** sous **Windows**).

Python

pyenv : **P**ython **env**ironments

- Gestionnaire de versions pour **Python** : permettant de travailler avec plusieurs versions de **Python**
- Disponible pour **Windows**, **Mac OS** et **Linux**
- Conçu initialement pour **Mac OS** et **Linux** ensuite pour **Windows**
- Deux dépôts **GitHub** différents :
  - Pour **Mac OS** et **Linux** : `https://github.com/pyenv/pyenv`
  - Pour **Windows** : `https://github.com/pyenv-win/pyenv-win`

Python

Pour l'installer sous Windows depuis PowerShell

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

Python

Pour l'installer sous Windows depuis PowerShell

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

En cas de problème avec la commande précédente, exécutez la commande suivante depuis PowerShell pour permettre à l'utilisateur actuel d'exécuter des scripts locaux sans restriction

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

Python

Depuis Mac ou Linux

```
curl https://pyenv.run | bash
```

Python

Depuis Mac ou Linux

```
curl https://pyenv.run | bash
```

Pour vérifier la version de pyenv

```
pyenv --version
```

Python

Depuis Mac ou Linux

```
curl https://pyenv.run | bash
```

Pour vérifier la version de pyenv

```
pyenv --version
```

Pour mettre à jour la version de pyenv

```
pyenv update
```

Python

Pour lister les versions Python disponibles pour pyenv

```
pyenv install -l
```

Python

Pour lister les versions Python disponibles pour pyenv

```
pyenv install -l
```

Pour installer une version particulière de Python (3.8.1 par exemple)

```
pyenv install 3.8.1
```

Python

Pour lister les versions Python disponibles pour pyenv

```
pyenv install -l
```

Pour installer une version particulière de Python (3.8.1 par exemple)

```
pyenv install 3.8.1
```


Pour installer plusieurs versions

```
pyenv install 3.12.3 3.13.0 3.5.1
```

Python

Pour lister les versions installées avec pyenv

```
pyenv versions
```

Python

Pour lister les versions installées avec pyenv

```
pyenv versions
```

Pour afficher la version courante de Python utilisée par pyenv

```
pyenv version
```

Python

Pour définir une version de Python comme étant globale (3.8.1 par exemple)

```
pyenv global 3.8.1
```

Python

Pour définir une version de Python comme étant globale (3.8.1 par exemple)

```
pyenv global 3.8.1
```

Pour désinstaller une version Python installée avec pyenv

```
pyenv uninstall 3.8.1
```

Python

Pour définir une version de Python comme étant locale (3.8.1 par exemple)

```
pyenv local 3.8.1
```

Python

Pour définir une version de Python comme étant locale (3.8.1 par exemple)

```
pyenv local 3.8.1
```

Explication

- `pyenv` permet de gérer et de basculer entre plusieurs versions de **Python**.
- Il ne crée cependant pas un environnement isolé pour les paquets.
- Il ne fait que définir la version de **Python** utilisée.

Python

Remarque

Pour créer un environnement virtuel isolé pour un projet et installer des dépendances sans les mélanger avec celles d'autres projets, on utilise **venv**.

Python

venv : **v**irtual **env**ironments

- Outil utilisé pour créer des environnements virtuels en **Python**
- Inclus par défaut avec **Python** à partir de la version 3.3
- Fournissant les fonctionnalités de base nécessaires pour créer et gérer des environnements virtuels
- Si vous deviez utiliser une version < 3.3, alors vous devriez installer **virtualenv**

Python

Question

Quelle version de **Python** utilise `venv` lors de la création d'un environnement privé ?

Python

Question

Quelle version de **Python** utilise `venv` lors de la création d'un environnement privé ?

Réponse

- **Version locale** (`pyenv local`) : Elle s'applique dans le répertoire courant et ses sous-répertoires. Elle est prioritaire sur la version globale.
- **Version globale** (`pyenv global`) : Elle s'applique en l'absence de configuration locale.

Python

Pensez à vérifier la version de Python que `pyenv` met à disposition avant de créer l'environnement virtuel en exécutant

```
python -m -version
```

Python

Pensez à vérifier la version de Python que `pyenv` met à disposition avant de créer l'environnement virtuel en exécutant

```
python -m -version
```

Pour forcer une version spécifique (3.8.1 par exemple), même si une locale est définie, exécutez

```
pyenv shell 3.8.1
```

Python

Pour créer un environnement virtuel (appelé `my_env`)

```
python -m venv chemin/vers/my_env
```

Python

Pour créer un environnement virtuel (appelé `my_env`)

```
python -m venv chemin/vers/my_env
```

Pour activer l'environnement virtuel

```
my_env/Scripts/activate
```

Python

Pour créer un environnement virtuel (appelé `my_env`)

```
python -m venv chemin/vers/my_env
```

Pour activer l'environnement virtuel

```
my_env/Scripts/activate
```


Résultat

```
(my_env) Chemin/vers/répertoire/courant>
```

Python

Remarques

- Maintenant, toute installation de package via `pip` sera limitée à cet environnement virtuel, et vous utiliserez la version de **Python** spécifiée pour ce projet.
- L'environnement peut être utilisé dans n'importe quel répertoire sue votre disque dur
- L'environnement peut être utilisé dans plusieurs projets différents

Python

Vérifions les packages installés dans notre environnement

```
pip list
```

Python

Vérifions les packages installés dans notre environnement

```
pip list
```

Installons le package `password-maker` dans notre environnement

```
pip install password-maker
```

Python

Vérifions les packages installés dans notre environnement

```
pip list
```

Installons le package `password-maker` dans notre environnement

```
pip install password-maker
```


Désactivons l'environnement virtuel

```
my_env/Scripts/deactivate
```

Python

Remarque

Vérifiez que `password-maker` n'est pas disponible dans l'environnement global.

Python

Pour lister les dépendances d'un projet dans un fichier `requirements.txt`

```
pip freeze > requirements.txt
```

Python

Pour lister les dépendances d'un projet dans un fichier `requirements.txt`

```
pip freeze > requirements.txt
```

Pour installer les dépendances d'un projet spécifiées dans un fichier `requirements.txt`

```
pip install -r requirements.txt
```

Python

Développement Web

- **Django** : Framework web complet.
- **Flask** : Micro-framework léger.
- **FastAPI** : API performante.
- **Requests** : Effectuer des requêtes **HTTP** de manière simple.

Python

Bases de Données

- **SQLAlchemy** : **ORM** pour bases de données **SQL**.
- **Psycopg2** : connecteur **PostgreSQL**.
- **mysql-connector-python** : pilote officiel **MySQL**.
- **pymongo** : pilote officiel de **MongoDB**.

Python

Science des Données et IA

- **NumPy** : Tableaux multidimensionnels et calculs numériques.
- **pandas** : Manipulation et analyse de données.
- **Matplotlib** / **Seaborn** : Visualisation de données.
- **scikit-learn** : Algorithmes de machine learning.
- **TensorFlow** / **PyTorch** : Deep learning.

Python

Science des Données et IA

- **NumPy** : Tableaux multidimensionnels et calculs numériques.
- **pandas** : Manipulation et analyse de données.
- **Matplotlib** / **Seaborn** : Visualisation de données.
- **scikit-learn** : Algorithmes de machine learning.
- **TensorFlow** / **PyTorch** : Deep learning.

Sécurité et Cryptographie

- **cryptography** : fonctions cryptographiques.
- **PyJWT** : gestion des **JWT** (**J**SON **W**eb **T**okens).

Python

Réseaux et Protocoles

- **scapy** : analyse de paquets réseau.
- **paramiko** : connexion **SSH**.
- **socket.io** : communication en temps réel.