[projet-1] Environnement de développement

Class	MDV2025

Environnement de développement

Pyenv

> Il existe de nombreuses versions de Python. Il est important d'avoir le contrôle sur la version utilisée dans un projet.
> pyenv est un outil qui permet d'installer différentes versions de Python dans un dossier dédié. À partir de chacune des versions installée, pyenv permet de créer des environnements virtuels dans cette version. Ainsi, il suffit d'installer une version que l'on souhaite utiliser, et ensuite de créer un environnement virtuel à partir de cette version.
> C'est à notre sens l'approche la plus simple et safe - et nous vous encourageons à toujours l'utiliser à moins que vous ne soyez plus à l'aise avec un autre outil.

Sur MacOS

On utilise le gestionnaire de package Homebrew pour installer pyenv

```
brew update
brew install pyenv
```

Sur Linux

Suivre ce tutoriel :

> Install pyenv
>
> https://ggkbase-help.berkeley.edu/how-to/install-pyenv/

Sur Windows

Clique droit : LANCER EN TANT QU'ADMINISTRATEUR SUR POWERSHELL

On utilise pyenv-win

> https://github.com/pyenv-win/pyenv-win

Installer Python 3.11.2

```
pyenv install 3.11.2
```

Créer un environnement virtuel

On crée un dossier qui contiendra le repo et on crée un environnement virtuel

```
mkdir projet_1
cd projet_1
~/.pyenv/versions/3.11.2/bin/python -m venv env
source env/bin/activate # activer l'environnement virtuel
# deactivate # désactiver l'environnement virtuel

# Vérifier sur Linux / Mac la version
# which python3

# Vérifier sur Windows
# where python3
```

Installer Black

> Black est un code formatter très *opiniated :* il permet donc d'avoir un code uniformisé lorsque plusieurs personnes travaillent sur la base code. Avec VS Code, il permet de "Format On Save".

Github

https://github.com/psf/black

Paramétrer VS Code pour utiliser Black

https://marketplace.visualstudio.com/items?itemName=ms-python.black-formatter

Setup packages

Installer un package

```
cd projet_1
source env/bin/activate
pip3 install requests==2.28.1 # installer requests

# Vérifier l'installation du package
# pip3 show requests
# En particulier, on veut voir que le package est bien installé ici :
# repos/dj/env/lib/python3.11/site-packages
```

Créer le projet

On préfère utiliser la structure suivante pour le projet :

```
##################
# dj
```

```
##################
projet_I
├── alerter.py
├── explications.txt
├── main.py
├── processor.py
├── requirements.txt
├── retriever.py
└── runtime.txt
```

Créer le fichier requirements.txt

```
###################
# dj
###################

touch requirements.txt
echo "requests==2.28.1" > requirements.txt
```

Créer un repository Git et push en remote

```
###################
# dj
###################
git init
touch .gitignore
echo "env" > .gitignore

git status
# On branch master

# No commits yet

# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#   .gitignore
#   requirements.txt
#   src/

git add .
git commit -m "first commit"
```

```
git remote add origin <your-git-remote-url>
git push --set-upstream origin master
```