Université Assane SECK de Ziguinchor

Unité de Formation et de Recherche des Sciences et Technologies

Département d'Informatique

Introduction à Django

Licence 2 en Ingénierie Informatique

Avril 2022

Papa-alioune.cisse@univ-zig.sn

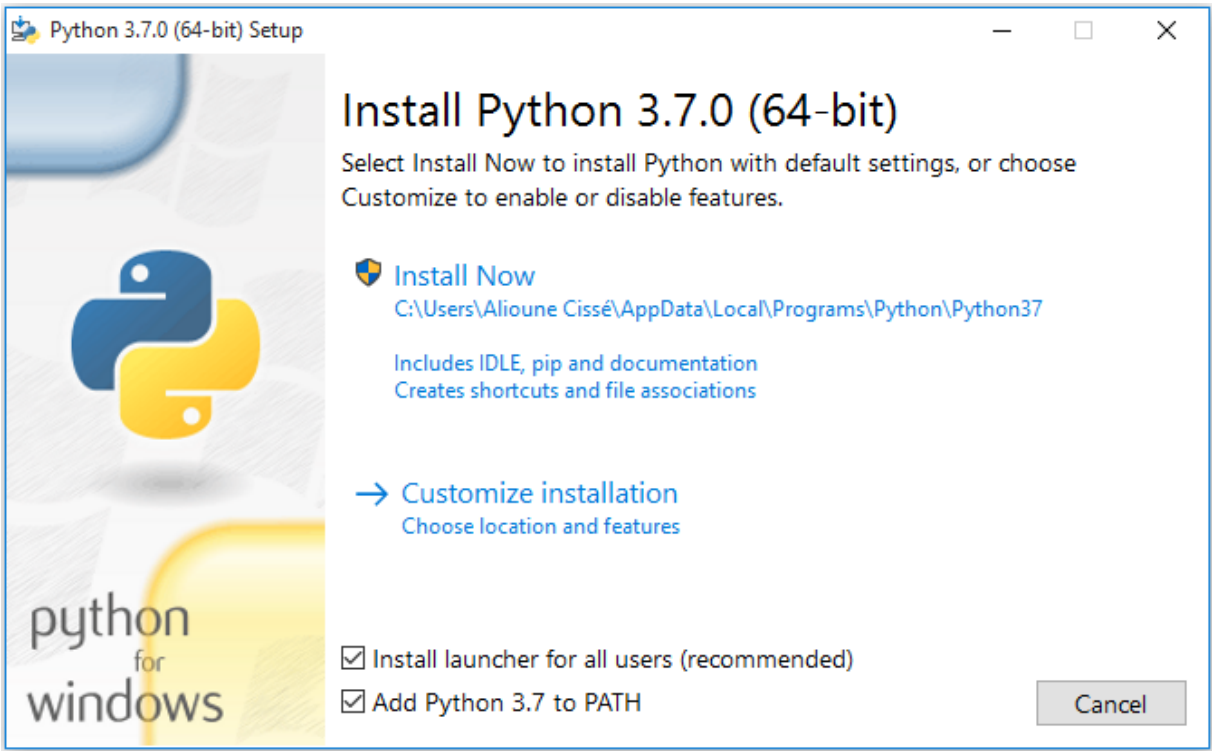
Résumé : À la différence du Développement Front-End, qui concerne l'aspect visuel et ergonomique d'un site web, le Développement Back-End se penche sur les aspects techniques et fonctionnels du développement d'un site web dynamique. Il s'agit du développement de l'ensemble des fonctionnalités "invisibles" d'un site web (le serveur, le code applicatif, la base de données, etc.) qui sont indispensables au bon fonctionnement d'un site. Ainsi, on peut dire que les Développeurs Back End travaillent sur la partie immergée de l'iceberg, alors que les Développeurs Front End se chargent essentiellement de la partie visible.
Ce chapitre introduit le développement web back-end avec le Framework Python nommé Django. Il aborde l'installation de l'environnement Python et de Django, la création d'environnements virtuels et la création de projets dans Django.

1 - Installation de Python sur Windows

1.1 - Installation

Lien de téléchargement de Python : https://www.python.org/downloads/windows/

N'oubliez pas de cocher la case « Add Python XX.X to Path » comme dans l'image suivante).



Si ce n'est pas fait au moment de l'installation (en oubliant de cocher la case « Add Python XX.X to Path » comme dans l'image précédente), ajouter Python au « Path Système ».

1.2 - Tester l'installation

Lancer la console Windows et taper la commande « python » pour ouvrir la "console Python".

Tester ensuite la commande « 2+5 » pour voir le résultat 7.

Taper la commande « exit() » pour sortir de la « console Python ».

2 - CRÉATION D'UN ENVIRONNEMENT VIRTUEL PYTHON

2.1 - Introduction

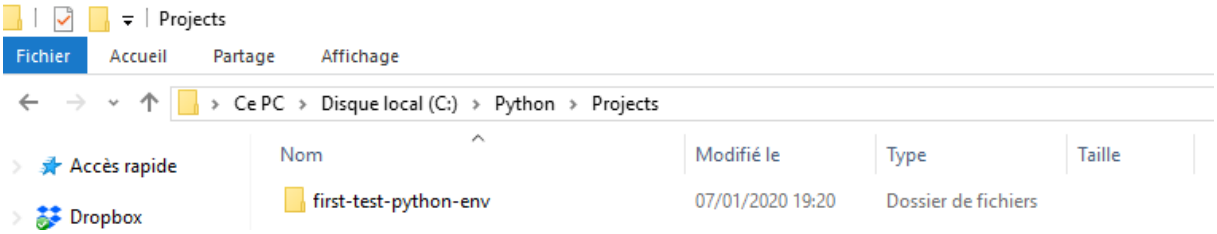
Un environnement virtuel est un environnement qui permet d'isoler vos dépendances de développement pour différents projets à différents endroits. Par exemple, les projets que vous développez peuvent dépendre de différentes versions de Python et utiliser des dépendances diverses. Avec un environnement virtuel, chacun de vos projets peut avoir sa propre version de Python, comporter ses dépendances et pourtant tous s'exécutent sur le même ordinateur. Il faut comprendre par là aussi, qu'il est possible d'installer sur une même machine plusieurs versions de Python.

C'est pourquoi, il est toujours recommandé d'utiliser un environnement virtuel lors du développement d'une application sur votre machine.

2.2 - Création d'un environnement virtuel

Il faut commencer par créer le répertoire de votre environnement comme dans cette image où mon environnement s'appelle « first-test-python-env »

Remarque : j'ai créé un dossier « Projects » dans le répertoire « C:\Python » qui contient d'ailleurs mes différentes versions (installations) de Python.

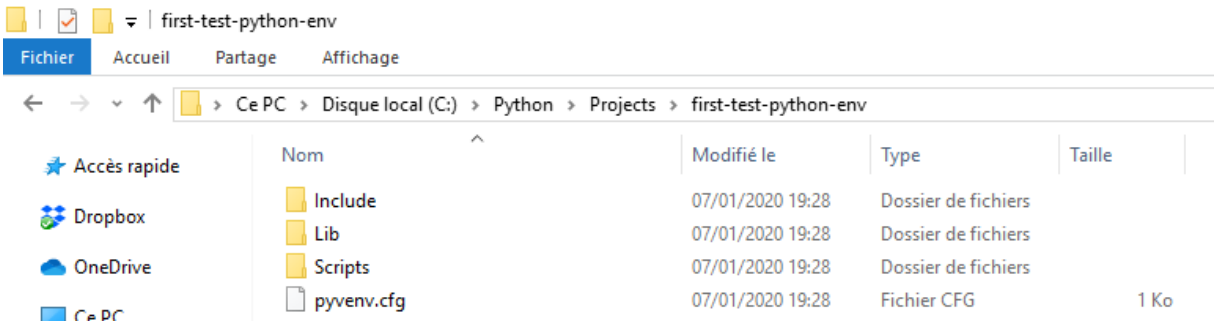


Copier ensuite le chemin de ce répertoire qui est pour moi : **C:\Python\Projects\first-test-python-env**

Ensuite, il faut ouvrir la console Windows et taper la commande suivante (en remplaçant mon chemin par le vôtre) :

```
python -m venv C:\Python\Projects\first-test-python-env
```

Après cette commande, vous verrez que votre environnement est créé avec quelques dossiers et fichiers ajoutés, comme dans :



2.3 - Activation d'un environnement virtuel

Pour activer un environnement virtuel et informer Python que vos prochains travaux vont se dérouler dans cet environnement, il faut :

- Se déplacer dans le dossier « Scripts » de votre environnement, en tapant par exemple dans la console la commande suivante (en remplaçant mon chemin par le vôtre) :

```
cd C:\Python\Projects\first-test-python-env\Scripts
```

- Taper simplement la commande suivante (qui va exécuter le fichier de commande Windows « activate » situé dans le dossier « Scripts ») pour activer l'environnement virtuel :

```
activate
```

- Pour savoir que la commande passe, la console doit ressembler à ceci (on voit bien qu'on se situe désormais dans l'environnement virtuel « first-test-python-env ») :

```
C:\Python\Projects\first-test-python-env>cd C:\Python\Projects\first-test-python-env\Scripts

C:\Python\Projects\first-test-python-env\Scripts>activate

(first-test-python-env) C:\Python\Projects\first-test-python-env\Scripts>
```

- N'oubliez pas de sortir du répertoire « Scripts » avant de continuer votre travail :

```
cd ..
```

2.4 - Désactivation d'un environnement virtuel

La désactivation d'un environnement se fait en se rendant toujours dans le répertoire « Scripts » de l'environnement, puis taper la commande « deactivate.bat », comme dans :

```
(first-test-python-env) C:\Python\Projects\first-test-python-env>cd Scripts

(first-test-python-env) C:\Python\Projects\first-test-python-env\Scripts>deactivate.bat
C:\Python\Projects\first-test-python-env\Scripts>
```

3 - Installation de Django et création de projet

3.1 - Installer l'installateur « pip »

Avant de le réinstaller, il faut être sûr qu'il n'est pas déjà installé, puisqu'il est automatiquement installé avec certaines versions de Python. Pour cela, ouvrez la console, activez un environnement virtuel et taper la commande suivante :

```
pip help
```

S'il répond, alors il est installé. Sinon, il faudra l'installer en suivant les étapes suivantes :

1. Télécharger le fichier « get-pip.py »

   Ouvrez la console et exécuter la commande suivante :

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

2. Installer pip

   Taper sur la console la commande suivante, en étant sûr d'être dans le répertoire où se trouve le fichier « get-pip.py » :

```
python get-pip.py
```

3. Ajouter pip dans les variables d'environnement Windows

   La dernière étape consiste à ajouter pip dans les variables d'environnement et de tester ensuite l'installation avec la commande :

```
pip help
```

3.2 - Installation de Django

Installer Django avec simplement la commande :

```
pip install django
```

Tester l'installation avec la commande :

```
django-admin --version
```

Ou

```
python -m django --version
```

3.3 - Création de projet Django

Créer un projet Django (ici le projet créé s'appelle "**mon_projet**") avec la commande :

```
django-admin startproject mon_projet
```

A la création du projet, rien ne se passe, mais vous pouvez au moins remarquer la création du dossier « ***mon_projet*** » avec comme contenu la structure suivante :

```
nom_projet/
  manage.py
  nom_projet/
    __init__.py
    settings.py
    urls.py
    wsgi.py
```

- Le script ***manage.py*** est un outil qui nous permettra d'exécuter des commandes utiles au sein de notre projet. Nous verrons très bientôt comment l'utiliser, par exemple pour créer les tables de notre base de données ou lancer le serveur de développement.
- Le sous-répertoire ***nom_projet*** contient les fichiers propres à notre projet :
  - ***settings.py*** contient la configuration globale du projet, par exemple les identifiants de connexion BDD, les chemins des différentes ressources, etc.
  - ***urls.py*** est le contrôleur frontal de notre projet : c'est le chef de gare qui orientera toutes les requêtes vers les bons contrôleurs.
  - ***wsgi.py*** est un fichier de configuration relatif au serveur qui exécutera notre projet, suivant l'interface WSGI (Web Server Gateway Interface). Nous n'avons pas à nous en occuper.

Ouvrez le fichier ***settings.py*** et lisez-le attentivement. Parmi les réglages qui y sont listés, j'aimerais attirer votre attention sur ceux-ci :

- ***DEBUG = True*** : si votre application rencontre une erreur, le navigateur n'affichera pas une page blanche mais des informations sur le bug rencontré. C'est extrêmement pratique en développement !
- ***LANGUAGE_CODE = "en-us"*** : La langue utilisée dans le projet, notamment dans l'interface d'administration. Remplacez-la dès maintenant par fr.
- ***TIME_ZONE = 'UTC'*** : le fuseau horaire que doit suivre votre projet.

Pour démarrer le serveur, déplacer vous dans le dossier du projet nouvellement créé (avec la commande « cd ») et taper la commande :

```
python manage.py runserver
```

Si le serveur est bien démarré, le message qui s'affiche sur la console vous indique clairement l'adresse de votre site : http://127.0.0.1:8000/

Pour changer le port par défaut (8000) par 8001 par exemple, la commande est :

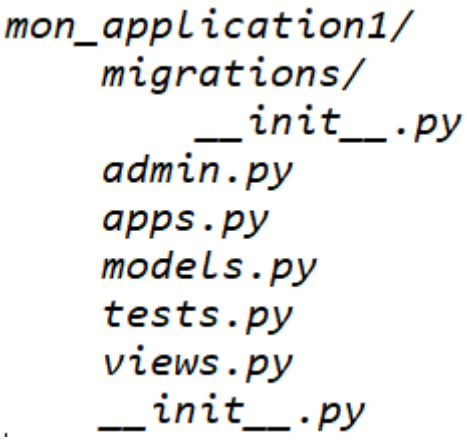
```
python manage.py runserver 8001
```

Pour arrêter le serveur depuis la console, il vous suffit de taper « ctrl + c ».

Vous pouvez à présent ajouter les applications (modules) de votre projet avec la commande (en se positionnant à la racine du projet nouvellement créé, dans le même répertoire que le fichier « manage.py ») :

```
django-admin startapp mon_application1
```

L'application créée « ***mon_application1*** » est au même niveau que le projet « ***mon_projet*** » et a la structure suivante :



- Le fichier ***models.py*** est destiné, comme son nom l'indique, à accueillir les modèles de notre application.
- Le fichier ***views.py*** est destiné, comme son nom ne l'indique pas, à accueillir les contrôleurs de notre application.
- Le fichier ***tests.py*** accueillera quant à lui les tests (notamment unitaires et d'intégration) de notre appli.

Chaque application nouvellement créée doit être ajoutée dans le fichier de configuration. Pour cela, ouvrez le fichier ***settings.py***, cherchez la variable ***INSTALLED_APPS***, et modifiez là pour qu'elle ressemble à ceci :

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
```

```
    'django.contrib.staticfiles',

    ' nom_application1'
)
```