**MACHINE LEARNING WITH PYTHON**

Liste des libreries les plus utilisées dans le machine learning en python

* Numpy
* Pandas
* Scikit-Learn

Lorsque nous travaillons sur un projet de machine Learning, nous utilisons un environnement appelé Jupyter pour écrire le code,

Pour télécharger Jupyter, nous utilisons anaconda

DJANGO

Pour créer un nouveau projet (Après avoir installé Django), on fait :

Django-admin startproject nom\_du\_projet .

Note\_\_\_ Cette commande prend plusieurs paramètres :

* startproject : Qui signifie que nous voulons commencer un nouveau projet
* Nom\_du\_projet : Qui représente le nom de notre projet
* . qui représente le répertoire courant (Donc celui dans lequel nous voulons créer notre projet)

Après avoir créer le projet Django, nous devons lancer notre serveur :

**python manage.py runserver**

**Note\_\_\_** Nous utilisons le module manage.py car c’est lui qui va gérer notre projet… Ce sera le fichier principal

**Note Importante\_\_\_** Sachant qu’une application est un ensemble de plusieurs mini applis qui fonctionnent ensemble, pour notre projet (Cas d’un supermarché pour nous) Nous allons créer tout d’abord une appli pour la gestion des produits du supermarché :

Pour créer une nouvelle appli on utilise la commande :

**python manage.py startapp products**

L’appli que nous voulons créer s’appelle **products**

1. View FUNCTIONS

Les View Functions sont des fonctions qui sont appelées lorsque l’utilisateur navigue sur une page spécifique.

Par exemple, lorsque l’utilisateur tape quelque chose dans la barre de recherche du navigateur, le navigateur envoie une requête http au serveur Django qui va va prendre cette url/adresse et vérifier à quelle view function elle correspond.

Et le rôle de cette fonction est de renvoyer une réponse au navigateur ou au client. Donc la fonction retourne du contenu HTML au navigateur afin que celui-ci puisse afficher une réponse au client

* Création de la vue d’une app, Dans le fichier vue d’une app :
* On importe ceci :

*from* django.shortcuts *import* render

* On créer une fonction index() qui est la fonction principale des views de toutes nos applications, ce paramètre est la requête http envoyée par le client/Navigateur

1. URL MAPPING

Le URL MAPPING permet de relier des view functions à des urls

Pour le faire, on suit les étapes suivantes :

1. On crée un fichier urls.py dans l’application actuelle

Une fois dans le fichier urls.py crée, on fait :

*from* django.urls *import* path

* Ceci nous permet d’avoir un objet qui va permettre de relier l’url à la fonction view correspondante
* *from* . *import* views

On importe le module views de notre application afin de pouvoir accéder aux VIEW FUNCTIONS qu’il contient

urlpatterns =\  
 [  
 path('', views.index, name='index'),  
 ]

* On créer une liste urlpatterns qui va contenir les liaisons
* Lorsque nous laissons le premier paramètre vide, ça veut dire que notre route est le répertoire actuel dans lequel nous sommes

Note\_\_\_ L’objet path() prend en paramètre :

1. **Route (chemin)** : La route est la partie de l'URL après le nom de domaine (par exemple, **https://example.com/chemin**). La fonction **path()** utilise la route pour déterminer quelle vue sera appelée lorsqu'une URL correspondante est reçue.

Exemple : **path('produits/', views.liste\_produits, name='liste\_produits')**

Dans cet exemple, la route est **'produits/'**. Cela signifie que lorsque l'utilisateur accède à l'URL **https://example.com/produits/**, la vue **views.liste\_produits** sera appelée.

1. **view (vue)** : La vue est une fonction Python qui reçoit la requête HTTP entrante en tant qu'argument et renvoie une réponse HTTP. C'est la vue qui sera appelée lorsque l'URL correspondante est atteinte.
   1. Exemple : **views.liste\_produits**
   2. Dans cet exemple, la vue associée est **views.liste\_produits**, ce qui signifie que lorsque l'utilisateur accède à l'URL correspondante, la fonction **liste\_produits** dans le module **views** sera exécutée pour générer la réponse.
2. **kwargs (arguments nommés)** : Vous pouvez spécifier des arguments nommés supplémentaires dans la fonction **path()** pour fournir des informations supplémentaires à la vue.
   1. Exemple : **path('produits/<int:id>/', views.detail\_produit, name='detail\_produit')**
   2. Dans cet exemple, **<int:id>** est un argument nommé qui permet de capturer un entier (d'où le **<int>**) à partir de l'URL et de le passer à la vue. Cela permet d'identifier de manière unique un produit spécifique dans cet exemple.
3. **name (nom)** : Le nom est un identifiant unique donné à une URL. Il est utilisé pour référencer l'URL de manière dynamique dans les templates ou dans le code Python.
   1. Exemple : **name='liste\_produits'**
   2. Dans cet exemple, l'URL est nommée **'liste\_produits'**, ce qui permet de l'utiliser plus tard pour générer des URL de manière dynamique à l'aide de la fonction **reverse()** ou de l'attribut **url** dans les templates.

* On va faire les même configuration dans le fichier urls.py du logiciel du projet (pas de l’application )