D-CLIC, FORMEZ-VOUS AU NUMÉRIQUE AVEC L'**OIF**

Installation et prise en main des environnements : Python, Jupyter Notebook, Google Colab

1) Choisir ton chemin (recommandé)

- **Le plus simple (zéro installation)** : **Google Colab**
  → Idéal pour débuter, machines modestes, travail en groupe, démos rapides.

- **Le plus robuste (local)** : **Anaconda + Jupyter Notebook**
  → Idéal pour travailler hors-ligne, gérer des projets propres, installer des libs.

Astuce : commence par **Colab** pour tester, puis installe **Anaconda** si tu veux travailler sérieusement en local.

2) Installer Python en local (méthode Anaconda – recommandé)

Anaconda, c'est une **distribution Python** spécialement conçue pour la **data science**, l'**IA** et l'**analyse de données**.

De façon globale :

- **Tout-en-un** : il installe Python + plus de 200 bibliothèques utiles (*pandas*, *numpy*, *matplotlib*…) dès le départ.

- **Gestion d'environnements** : avec la commande conda, tu peux créer plusieurs environnements séparés pour éviter les conflits de versions entre projets.

- **Outils inclus** : Jupyter Notebook, Spyder, et d'autres interfaces prêtes à l'emploi.

- **Gain de temps** : pas besoin d'installer chaque bibliothèque une par une, tout est déjà prêt après installation.

Anaconda est donc comme une **boîte à outils déjà remplie** pour commencer à coder en Python dans le domaine des données et de l'IA.

2.1. Télécharger et installer

1. Va sur **anaconda.com/download**
2. Télécharge **Anaconda Distribution** (Python 3.x) pour ton OS.
3. Installe (suivant → suivant…).
    - Sur Windows : coche "Add Anaconda to my PATH" **si proposé** (sinon, utilise l'Anaconda Prompt).
    - Sur macOS : ouvre le .pkg et suis l'assistant.
    - Sur Linux : **bash Anaconda3-*.sh** puis accepte les options par défaut.

2.2. Vérifier l'installation

Ouvre **Anaconda Prompt** (Windows) ou un **Terminal** (macOS/Linux) et tape :

conda --version
Cette commande affiche la **version de conda** installée.
conda = l'outil de gestion d'environnements et de packages inclus avec Anaconda.

python --version

Vérifie que Python est bien installé et utilisable depuis le terminal.
Affiche la version de Python utilisée par défaut

2.3. Créer un environnement propre

Les environnements évitent les conflits de versions.

conda create -n ia101 python=3.11 -y

- **conda create** → demande à Anaconda de créer un nouvel environnement.
- **-n ia101** → donne un **nom** à l'environnement (ia101 dans cet exemple). Tu peux choisir un autre nom.
- **python=3.11** → précise la version de Python que tu veux utiliser dans cet environnement.
- **-y** → répond "yes" automatiquement aux questions pendant la création.

Résultat :
Un dossier séparé contenant son propre Python + ses propres bibliothèques, indépendant des autres projets.

conda activate ia101

- **conda activate** → "entre" dans l'environnement ia101.
- À partir de là, toutes les commandes Python, pip, etc., utiliseront **ce Python-là** et **ses bibliothèques**, pas celles d'un autre environnement.

Résultat :
Ton terminal affiche quelque chose comme :

Pourquoi tout ceci est utile ?

- Évite les **conflits de versions** (par exemple, un projet a besoin de pandas 1.5, un autre de pandas 2.0).

Tu as un projet “Analyse ventes” : il marche seulement avec **pandas 1.5**.

Tu as ensuite un Projet “IA images” qui marche seulement avec **pandas 2.0**.

Si tu n’as qu’un seul Python, installer pandas 2.0 écrasera la version 1.5 et cassera ton premier projet.

Avec deux environnements séparés → les deux projets cohabitent sans problème. Cela te permet de **tester des librairies** sans casser ton installation globale.

2.4. Installer des bibliothèques utiles

conda install jupyter pandas numpy matplotlib -y # Ou via pip si tu préfères :

pip install jupyter pandas numpy matplotlib

3) Jupyter Notebook (local)

3.1. Lancer Jupyter

Dans ton terminal (après conda activate ia101) :

jupyter notebook

Un onglet s’ouvre dans le navigateur. Clique **New → Python 3 (ipykernel)**.

3.2. Prendre en main (5 gestes)

- **Exécuter une cellule** : Shift + Enter
- **Ajouter une cellule** : A (au-dessus), B (en dessous)
- **Changer le type** : Code ↔ Markdown (menu déroulant ou M pour Markdown / Y pour Code)

- **Sauvegarder** : Ctrl/Cmd + S
- **Redémarrer le noyau** (Kernel) : Kernel → Restart

3.3. Mini-exemple

Crée deux cellules : **Cellule**

1 (Python)

```
import pandas as pd

df = pd.DataFrame({
   "pays": ["Togo", "Bénin", "Sénégal"],
   "PIB_mds_USD": [8.4, 17.1, 28.0]
}) df
```

Cellule 2 (graphique simple)

```
import matplotlib.pyplot as plt

df.plot(kind="bar", x="pays", y="PIB_mds_USD", title="PIB (approx.)") plt.show()
```

3.4. Markdown (pour les notes)

Créer une cellule **Markdown** et coller :

```
# Mon premier notebook
- Objectif : tester Python + pandas + graphiques
- Données : PIB (exemple jouet)
```

3.5. Fermer proprement

- File → Save and Checkpoint

- File → Close and Shutdown

- Dans le terminal : Ctrl + C pour arrêter Jupyter.

4) Alternative locale légère (sans Anaconda)

Si tu préfères **Python "pur"** + pip :

1. Installe Python depuis **python.org/downloads** (coche *Add Python to PATH* sur Windows).

2. Vérifie :

python --version
pip --version

3. Crée un environnement virtuel :

python -m venv .venv #
Activer l'environnement :
Windows
.venv\Scripts\activate #
macOS/Linux ***source***
.venv/bin/activate

4. Installe Jupyter et bibliothèques :

pip install jupyter pandas numpy matplotlib jupyter notebook

5) Google Colab (zéro installation)

Google Colab (*Google Colaboratory*) est un **service gratuit en ligne** de Google qui permet d'écrire et d'exécuter du code **Python** directement dans ton navigateur, **sans rien installer** sur ton ordinateur.

- **Coder tout de suite** : pas besoin d'installer Python ou Jupyter, tout est déjà prêt.
- **Stockage et partage faciles** : les notebooks sont sauvegardés dans ton **Google Drive**.
- **Travail collaboratif** : plusieurs personnes peuvent éditer le même notebook, comme un Google Docs.
- **Puissance gratuite** : tu peux utiliser un processeur puissant, un GPU ou même un TPU pour accélérer tes calculs (utile pour le deep learning).
- **Assistance Gemini intégrée** : tu peux utiliser **Gemini**, l'IA générative de Google, directement dans Colab pour t'aider à écrire, corriger ou optimiser ton code Python, expliquer des erreurs et proposer des solutions.

5.1. Démarrer

- Va sur **colab.research.google.com** (compte Google requis).
- **New Notebook** → tu écris du Python immédiatement.

5.2. Charger un fichier (CSV)

- **Files** (icône dossier à gauche) → **Upload**
- Ou bibliothèque files :

```
from google.colab import files
uploaded = files.upload()  # sélectionner ton CSV
```

Puis :

import pandas as pd df =

pd.read_csv("ton_fichier.csv")

df.head()

5.3. Sauvegarder

- Menu **File → Save a copy in Drive** (sauvegarde dans Google Drive).
- Tu peux aussi **télécharger** le notebook : File → Download → .ipynb.

5.4. Utiliser GPU/TPU (si besoin)

- Runtime → Change runtime type → Hardware accelerator → GPU/TPU

Utile pour deep learning, pas nécessaire pour les bases.

5.5. Importer depuis Google Drive

from google.colab import drive
drive.mount('/content/drive')
Tes fichiers sont accessibles sous /content/drive/MyDrive/...

6) Bonnes pratiques (dès le début)

- **Un projet = un dossier** (et un environnement virtuel si local).
- **Notebooks clairs** : sections Markdown (# Titre, ## Sous-titre), peu de magie "cachée".
- **Versionner** (Git) : garde l'historique de ton code.
- **Données** : un sous-dossier data/ ; ne mets pas de données sensibles en clair.
- **Reproductibilité** : exporte les dépendances :
    - conda : conda env export > environment.yml
    - pip : pip freeze > requirements.txt

7) Dépannage rapide (FAQ)

- **jupyter: command not found**
  → Active l'environnement (ex. conda activate ia101) ou pip install jupyter.
- **Notebook ne se lance pas**
  → Ferme tous les terminaux, relance jupyter notebook; vérifie le parefeu/antivirus.
- **Problème de versions (pandas, numpy)**
  → Mets à jour : pip install -U pandas numpy (ou conda update --all).
- **"Kernel died / redémarre en boucle"**
  → Trop de mémoire utilisée : ferme d'autres notebooks, échantillonne tes données, redémarre le noyau.

8) Check-list “Prêt à coder”

- Python installé (Anaconda **ou** Python+pip)
- Environnement créé et **activé** (conda activate ia101 ou .venv activé)
- Jupyter opérationnel (jupyter notebook) **ou** Colab ouvert
- pandas, numpy, matplotlib installés
- Un notebook testé avec un petit DataFrame + un graphique

9) Mini-TP “Hello Data!”

9.1. Colab ou Jupyter — cellule 1

```
import pandas as pd

data = {
   "ville": ["Lomé", "Cotonou", "Accra", "Abuja"],
   "tempC": [30.5, 29.8, 31.2, 28.7]
}
df = pd.DataFrame(data)
df
```

9.2. Cellule 2 import

```
matplotlib.pyplot as plt


df.plot(kind="bar", x="ville", y="tempC", title="Températures moyennes (exemple)")
plt.xlabel("Ville") plt.ylabel("°C")
plt.show()
```

Objectif : vérifier que ton environnement sait **lire des données** et **afficher un graphe**.