**Introduction**

Bonjour le nouvel alternant !

Voici une liste des éléments à installer pour pouvoir programmer en Python à la Caisse d'Epargne.

Normalement, on installe Anaconda et le tour est joué ! Cependant, Anaconda n'est pas autorisé dans le groupe et il faut utiliser un autre gestionnaire Python. On va donc utiliser **Miniforge** qui est un équivalent open source. Cela fonctionne exactement comme Anaconda (toutes les commandes conda fonctionnement) mais nous n'avons pas d'interface. On utilisera uniquement l'invité de commande.

Notons qu'en raison des pares-feux de l'entreprise, nous sommes très limitée en termes de téléchargements et installations de logiciels. Ceci s'applique également à un certain nombre de librairies Python qu'il faudra peut-être installer à la main (là il faudra chercher sur internet comment faire) ou qui ne pourront pas être installées du tout.

Ainsi, si tu veux installer une librairie Python, essaye tout d'abord avec l**'invité de commande** en exécutant conda install ou pip install (voir plus bas pour le pip). Si cela ne fonctionne pas, essaye l'installation **à la main**. Je recommanderais d'utiliser le navigateur Chrome pour télécharger les librairies car Internet Explorer bloque davantage les téléchargements. Si tu ne peux pas télécharger ou que l'installation ne fonctionne pas, tu peux toujours faire une demande via le **service informatique SII**.

**Installation de Miniforge & Environnements conda**

Cette partie montre comment installer **Miniforge** (ton Python).

Pour cela, on utilise le tutoriel en ligne fournit par le groupe BPCE.

Cependant, j'ai rencontré quelques problèmes lors de l'installation. Les explications ci-dessous viennent compléter le tutoriel du groupe.

Je recommanderais de lire l'entièreté de chaque point avant de réaliser chaque étape d'installation car les informations importantes ne se trouvent toujours au début de mes phrases.

Voici le lien du tutoriel officiel pour l'installation de Miniforge dans le groupe : [Utilisation de Miniforge (sharepoint.com)](https://groupebpce.sharepoint.com/sites/CommunautDataScience/SitePages/Utilisation-de-Miniforge.aspx)

* 1. Ouvrir le tutoriel : [Utilisation de Miniforge (sharepoint.com)](https://groupebpce.sharepoint.com/sites/CommunautDataScience/SitePages/Utilisation-de-Miniforge.aspx) et installer comme indiqué Miniforge3 pour Windows. Ouvrir le lien pour installer Miniforge3 avec le moteur de recherche Chrome (Chrome permet d'éviter d'être bloqué pour le téléchargement). Le téléchargement sera sûrement placé dans le dossier C:\Users\AXXXXXXX\Downloads où AXXXXXXX est ton identifiant.

Avant de poursuivre le tutoriel en ligne, effectuer l'étape 2.

* 1. Cliquer sur l'exécutable Miniforge3-Windows-x86\_64.exe contenu dans C:\Users\AXXXXXXX\Downloads et installer Miniforge. **/.\** Lors de l'installation, cliquer sur **Add Anaconda to my PATH environment variable** (ignorer l'avertissement), ce qui permettra par exemple d'avoir accès à Miniforge lorsque l'on tape conda dans l'invité de commandes. Je pense que cette étape est indispensable et permettra d'éviter de nombreux problèmes. Miniforge est à présent installé dans le dossier

C:\Users\AXXXXXXX\AppData\Local\miniforge3 (dans *affichage*, cocher *Eléments masqués* pour faire apparaître le dossier AppData).

* 1. Revenir au tutoriel en ligne. Etant donné que nous avons ajouté Anaconda au PATH dans l'étape 2), nous n'avons pas besoin de mettre à jour la variable d'environnement PATH avec les 2 entrées spécifiées dans le tutoriel. D'ailleurs, pour mettre à jour le PATH comme indiqué dans le tuto, il faudrait avoir accès à System > About > Advanced system settings, ce qui n'est pas le cas avec l'ordinateur d'entreprise.

* 1. Section du tutoriel *2.2. Créer et mettre à jour le fichier de config* :

Le tutoriel ne me semble pas très clair à ce niveau. Voilà ce que j'ai fait:

* 1. Taper dans l'invité de commande cmd les lignes suivantes:

conda config

conda config --add channels <https://artifactory.f.bbg/artifactory/bpce-it-conda-forge/>

conda config --remove channels defaults

conda config --set ssl\_verify False

Un fichier .condarc a ainsi été créé à l'emplacement C:\Users\AXXXXXXX. On peut ouvrir ce fichier avec le bloc note par exemple. J'ai supprimé du .condarc la ligne de texte [conda-forge]. Ainsi, ce fichier .condarc doit contenir:

channels: [<https://artifactory.f.bbg/artifactory/bpce-it-conda-forge/>]

ssl\_verify: false

* 1. Il y a un **2e** fichier .condarc dans l'emplacement C:\Users\AXXXXXXX\AppData\Local\miniforge3. Ouvrir ce fichier (par ex avec le bloc note) et supprimer la ligne de texte [conda-forge]. Inscrire dans le 2e fichier .condarc le texte suivant:

channels:

  - <https://artifactory.f.bbg/artifactory/bpce-it-conda-forge/>

ssl\_verify: false

override\_channels\_enabled: true

default\_channels: {}

custom\_channels: {}

custom\_multichannels: {}

channel\_priority: strict

Rq: Faire attention à ne pas laisser d'espaces ni de saut de lignes après la ligne de texte "channel\_priority: strict" à la fin du .condarc sinon cela va créer des problèmes !

Tu peux à présent taper la commande conda info dans l'invité de commande cmd et tu devrais avoir quelque chose

d'à peu près similaire à ce qu'il y a dans le tutoriel en ligne dans la partie *2.1. Obtenir les informations de version de Conda, install, channels, config.*

Rq: En ayant configuré les fichiers .condarc de la sorte, je peux créer des environnements et installer des packages mais je ne trouve plus l'accès au Miniforge Prompt. Je pense que la raison est que j'ai supprimé des .condarc le channel conda-forge en supprimant la ligne de texte [conda-forge].

Le Miniforge prompt est simplement un invité de commande comme l'invité de commande Windows cmd où l'environnement base est activé par défaut. On peut retrouver ceci en tapant conda activate base (pour activer par exemple l'environnement base) depuis l'invité de commande cmd windows (ceci fonctionne car on ajouté Anaconda à la variable PATH dans l'installation de Miniforge…).

* 1. Poursuivre le tutoriel à partir de la partie *2.3. Création d'un environnement*

Pour rappel, on crée un environnement conda en tapant les lignes suivantes dans l'invité de commandes Windows :

conda create --name myenv (ou myenv est le nom de votre choix pour l'environnement conda créé)

On active l'environnement créé de la sorte :

conda activate myenv

Et on le désactive en exécutant

conda deactivate

*Voir* [*Managing environments — conda 23.7.5.dev97 documentation*](https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html) *pour plus d'information sur les environnement conda.*

**>>>** Il est important de **créer un nouvel environnement conda par proje**t, ceci pour plusieurs raisons :

* 1. Lors de manipulations/installations de nouvelles librairies, il peut arriver de "casser" l'environnement conda. Plus aucun code ne fonctionne et on doit tout recommencer. Au lieu de devoir réinstaller tout le Python (comme on l'a vu plus haut c'est un peu fastidieux), il suffit de supprimer uniquement l'environnement conda cassé et de le recréer ensuite, ce qui est bien plus rapide.

* 1. Cela évite aussi d'avoir des problèmes de dépendances entre les libraires. Admettons que l'on travaille sur deux projets A et B et que nous n'avons créé qu'un seul environnement conda pour ces 2 projets, par exemple nommé myenv (ou pire admettons que nous travaillons directement dans l'environnement base de conda).

Le projet A requiert la version Python 3.7.0 pour fonctionner et nous installons donc Python 3.7.0 sur myenv. Une fois le projet terminé, nous passons au projet B. Cependant, ce projet requiert la version Python 3.8.0 et nous mettons alors à jour le Python de l'environnement myenv en le faisant passer de la version 3.7.0 à la version 3.8.0. Le projet B peut à présent fonctionner. Cependant, si nous souhaitons à présent revenir sur le projet A, c'est impossible car la version de Python n'est plus correcte. Le projet A ne fonctionne plus. On peut alors essayer de rétrograder Python de 3.8.0 à 3.7.0 et c'est comme ça que l'on casse son Python (tiré d'une histoire vraie…)

La bonne pratique consiste donc à créer 2 environnements Pythons : l'environnement myenvA pour le projet A avec Python 3.7.0 et l'environnement myenvB pour le projet B avec Python 3.8.0.

De manière générale, il faut créer un environnement conda par projet.

Cet exemple est illustré avec des versions de Python mais cela vaut aussi pour toute autre librairie.

* 1. Pour connaître toutes les librairies disponibles pour le groupe BPCE, aller sur *:* [Index of bpce-it-conda-forge/](https://artifactory.f.bbg/artifactory/bpce-it-conda-forge/).

C'est de ce channel que proviennent les packages quand on lance un conda install package. Par défaut, il me semble que ça installe la version la plus récente.

**Installation de davantage de librairies** (**/!\ à faire**)

Certaines librairies dont on a besoin pour le machine learning ne sont pas sur le channel <https://artifactory.f.bbg/artifactory/bpce-it-conda-forge/> où conda va chercher les packages à installer.

Pour avoir davantage de librairies, créer à la main un fichier **pip.ini** dans le dossier de l'environnement de votre choix C:\Users\AXXXXXXX\AppData\Local\miniforge3\envs\myenv où myenv est le nom de l'environnement de votre choix (qui aura au préalable été créé selon les étapes ci-dessus).

Inscrire dans le fichier **pip.ini** les commandes suivantes (ne pas laisser d'espaces à la fin des lignes):

[global]

timeout = 60

index-url = <https://artifactory.f.bbg/artifactory/api/pypi/python-proxy-pypi/simple>

trusted-host = artifactory.f.bbg

[install]

index-url = <https://artifactory.f.bbg/artifactory/api/pypi/python-proxy-pypi/simple>

trusted-host = artifactory.f.bbg

A présent, pour installer une nouvelle librairie qui n'est pas dans conda, installer avec **pip install** au lieu de **conda install**. On a accès à beaucoup plus de librairies. On peut même utiliser **pip install** pour tout installer dès à présent.

Notons qu'il faut reproduire ces étapes pour chaque nouvel environnement conda car il faut un fichier **pip.ini** par environnement conda.

**Rq**: J'ai eu beaucoup de problèmes avec l'installation de nombreuses librairies avec le pip sur mon environnement sous **python 3.11**. Par exemple, pour l'installation de la librairie tsai, le message d'erreur me disait que je devais avoir une version de setuptools < 60. Lorsque j'ai installé setuptools 58, une autre erreur affichait que je devais avoir python >= 3.7 et < 3.11… Je n'ai pu résoudre cette erreur que en recréant un environnement avec python 3.10 et dans ce cas pip fonctionne très bien pour tsai et pour d'autres libraries.

**Conclusion: pip.ini fonctionne très bien avec des versions de python < 3.11**

**Installation de VS Code**

Maintenant que Python est installé avec Miniforge, il faut un éditeur de code. Je recommande **VS Code** puisqu'il est léger, simple et efficace avec de nombreuses extensions. VS Code contient notamment une extension Jupyter Notebook pour que l'on puisse éditer les notebooks.

Quelques notes supplémentaires sont apportées au tutoriel d'installation de VS Code que l'on peut trouver sur le sharepoint du groupe [Installation VSCODE - 1 (sharepoint.com](https://groupebpce.sharepoint.com/sites/CommunautDataScience/SitePages/INSTALLATION-VSCODE--.aspx)). Une fois encore, je recommande de lire toutes les étapes ci-dessous avant de se lancer dans l'installation :

* 1. Cliquer sur le lien [Installation VSCODE - 1 (sharepoint.com](https://groupebpce.sharepoint.com/sites/CommunautDataScience/SitePages/INSTALLATION-VSCODE--.aspx)) et suivre les étapes jusqu'à l'installation de VS Code. Une fois téléchargé, VS Code se situe dans le dossier: C:\Users\AXXXXXXX\AppData\Local\Programs\Microsoft VS Code

* 1. Quelques précisions sur la section Partie *3. Utilisation en DataScience > 3.1 Installation de python/MiniForge et configuration de Conda* du tutoriel en ligne :

* 1. Comme indiqué sur le tutoriel, ouvrir un terminal *windows powershell* et taper conda init powershell, ce qui permet d'activer un environnement conda depuis powershell. Redémarrer le terminal powershell. Cette manipulation n'est à effectuer qu'une seule fois.

* 1. Comme indiqué dans le tutoriel, trouver le fichier settings.json dans le dossier C:\Users\AXXXXXXX\AppData\Roaming\Code\User et ajouter les lignes indiquées dans le tuto:

"terminal.integrated.profiles.windows": {

"GitBash": {

"path": "C:\\Path\\to\\bash.exe",

},

"PowerShell -NoProfile": {

"source": "PowerShell",

"args": ["-NoProfile"]

}

}

*Explications de ce qui se trouve dans le fichier settings.json :*

*VSCode lie automatiquement les environnements conda dans le terminal intégré grâce à l'extension python avec la commande conda activate myenv. Cependant par défaut, VSCode utilise windows cmd.*

*Or, la plupart des scripts lourds sont réalisés à l'aide de powershell. Par défaut, l'extension python ne prend pas en charge powershell pour activer les environnements conda. Les deux commandes dans le fichier settings.json ci-dessus permettent d'activer les environnements conda automatiquement en utilisant powershell au lieu de cmd.*

*[Source:* [Automatically activate conda evironment in Powershell for VSCode | Krishan’s Tech Blog (krishansubudhi.github.io)](https://krishansubudhi.github.io/vscode/2020/09/17/vscod-conda-powershell.html)*]*

* 1. Installation de l'extension Jupyter sur VS Code :

On peut utiliser Jupyter sur VS Code, mais je préfère personnellement utiliser directement Jupyter Notebook indépendamment de VS Code car je trouve que Jupyter Notebook est mieux construit. Pour l'installation de Jupyter Notebook indépendamment de Jupyter, se référer à la section suivante.

Je donne cependant les étapes pour installer le Jupyter de VS Code au cas où tu préfèrerais ce Jupyter :

Après l'installation de l'extension Jupyter sur VS Code, il faut taper les commandes suivantes dans un invité de commande (invité de commandes Winfows cmd par ex) pour que l'on puisse utiliser Python depuis le Jupyter de VS Code :

conda activate myenv (activer l'environnement dans lequel on veut installer Jupyter)

conda install ipykernel

ipython kernel install --user --name=myenv (on met le même nom --name=myenv que l'environnement que l'on vient d'activer pour éviter les confusions par la suite).

* 1. Avec VS Code, tu peux avoir besoin des extensions suivantes :
     + Python
     + Jupyter
     + Jupyter Keymap
     + Jupyter Notebook Renderers
     + Pylance

D'autres extensions très utiles :

* 1. autoDocstring, si tu fais 3 guillements ("""), on te suggère de créer un docstring, c’est-à-dire une documentation.
  2. Better comments : tu peux avoir des commentaires de couleurs différentes sur ton VS Code, tu pourras lire l'extension.
  3. CodeSnap : prend une belle capture d'écran de ton code et tu peux sauvegarder l'image. Très pratique pour les rapports.

**Installation de Jupyter Notebook**

**Etapes pour l'installation de Jupyter Notebook :**

Bien que Jupyter soit installé dans VS Code, c'est moins agréable que **Jupyter Notebook**.

Voici les étapes pour installer Jupyter Notebook:

conda activate myenv (installe Jupyter notebook dans l'environnement de ton choix myenv)

conda install jupyter

Rq: On vient d'installer Jupyter notebook dans l'environnement conda myenv.

Si on veut changer d'environnement conda, il semble qu'il faille réinstaller Jupyter Notebook dans ce nouvel environnement. En effet, j'ai eu des problèmes de dépendances de librairies en utilisant sur un environnement conda un Jupyter Notebook qui avait été installé sur un autre environnement conda.

Si ce n'est pas déjà fait, il faudra installer **ipykernel** en tapant les commandes suivantes dans l'invite de commandes Windows cmd :

conda activate myenv (activer l'environnement dans lequel on veut installer le kernel)

ipython kernel install --user --name=myenv (on met le même nom --name=myenv que l'environnement que l'on vient d'activer pour éviter les confusions par la suite).

On peut à présent lancer jupyter en tapant dans l'invité de commande Windows cmd:

conda activate myenv (activer l'environnement dans lequel oest installé jupyter)

jupyter notebook (lancer jupyter : une page web s'ouvre)

**Possible problème rencontré :**

Il se peut que Jupyter Notebook ne reconnaisse pas le kernel **ipykernel**. Ceci signifie que Jupyter Notebook ne va pas chercher le python.exe dans le bon dossier. Voici une solution qui a fonctionné pour moi:

Taper dans l'invité de commande Windows :

conda activate myenv

jupyter kernelspec list

J'obtiens:

base C:\Users\AXXXXXXX\AppData\Roaming\jupyter\kernels\base (<- pour info, ceci a été créé avec la commande tapée + haut: python kernel install --user --name=myenv où myenv = base)

python3 C:\Users\AXXXXXXX\AppData\Local\miniforge3\share\jupyter\kernels\python3

Aller à l'emplacement du fichier indiqué à la suite de python3 : C:\Users\AXXXXXXX\AppData\Local\miniforge3\share\jupyter\kernels\python3 et ouvrir le fichier kernel.json. Dans ce fichier, il se peut que tu doives modifier l'emplacement du python dans "argv" avec l'emplacement de ton python. Par exemple, j'ai mis:

"argv": [

   "C:\\Users\\AXXXXXXX\\AppData\\Local\\miniforge3\\python.exe", <- ligne modifiée (en ajoutant le double "\\" pour que ce soit lisible sur Windows)

  "-m",

  "ipykernel\_launcher",

  "-f",

  "{connection\_file}"

 ],

 "display\_name": "Python 3 (ipykernel)",

 "language": "python",

 "metadata": {

  "debugger": true

 }

}

Mon problème était dû au fait que je n'avais qu'un seul symbole slash "\" au lieu de deux "\\" pour le chemin de mon Python. En effet, Python a du mal avec le simple "\" sur Windows et préfère le chemins avec des "/" ou "\\".

**Facultatif mais très pratique :**

Pour avoir davantage de fonctionnalités dans Jupyter Notebook, installer jupyter\_contrib\_nbextensions :

conda install jupyter\_contrib\_nbextensions

jupyter contrib nbextension install --user

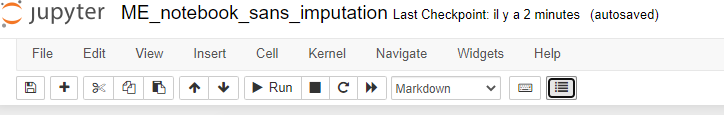
Maintenant, un nouvel onglet Nbextensions est présent à l'ouverture de Jupyter Notebook:



Aller dans l'onglet Nbextensions , et cocher Table of Contents qui est un outil pratique pour avoir une table des matières et structurer les Notebooks.

Ouvrir un notebook. Pour voir apparaître la table of contents, cliquer sur le logo :





Rq : si Nbextension n'apparaît pas, essayer d'installer une ancienne version de Jupyter :

conda install notebook==6.1.5 (fonctionne avec python==3.7 par exemple)

*([Jupyter nbextensions does not appear - Stack Overflow](https://stackoverflow.com/questions/49647705/jupyter-nbextensions-does-not-appear))*

**Environnements conda sous Jupyter :**

Comme évoqué plus haut, une bonne pratique consiste à créer un nouvel environnement conda pour chaque nouveau projet afin de ne pas avoir de problèmes de versions de librairies et ne pas avoir à réinstaller tout Python en cas de problème.

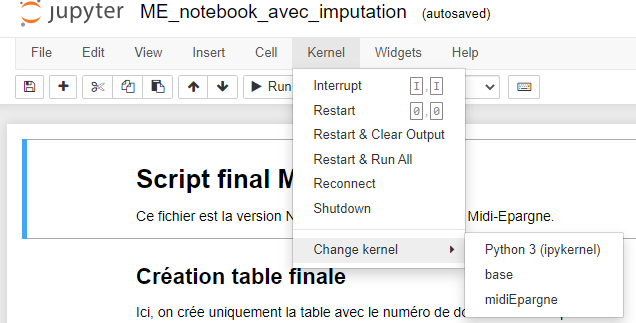
Pour chaque nouveau projet, on crée donc un nouvel environnement conda, par exemple nommé myenv, et on doit donc réinstaller dans ce nouvel environnement Jupyter Notebook et son ipykernel (voir les étapes ci-dessus).

On peut maintenant lancer Jupyter Notebook depuis l'environnement myenv.

Lorsque l'on est sur un Notebook, on sélectionne le kernel lié à l'environnement myenv en allant dans :

Kernel > Change Kernel > et sélectionne ton environnement virtuel myenv

Dans l'image ci-dessous, j'ai créé un environnement appelé midiEpargne



**Supprimer un environnement Conda**

Pour supprimer un environnement conda, on peut taper la commande:

conda env remove -n myenv (où myenv est le nom de l'environnement que l'on veut supprimer).

Si cela ne fonctionne pas, on peut aussi supprimer l'environnement "à la main".

Pour cela, aller dans le fichier C:\Users\AXXXXXXX\AppData\Local\miniforge3\envs et supprimer le répertoire portant le nom de l'environnement .

Penser aussi à supprimer le ipykernel. Pour cela, ouvrir une fenêtre de commande et taper :

jupyter kernelspec list

S'affiche alors la localisation du ipykernel créé, par exemple:

C:\Users\AXXXXXXX\AppData\Roaming\jupyter\kernels\myenv

Aller dans l'emplacement indiqué et supprimer le répertoire myenv.

**Installation de Git**

(au final, je ne l'ai jamais utilisé, mais ça peut être utile)

On a accès à Git uniquement en local mais ça peut être utile pour gérer les différentes versions d'un même fichier.

Pour installer Git, aller sur le lien [Git - Downloading Package (git-scm.com)](https://git-scm.com/download/win) et installer 64-bit Git for Windows Setup. Le fichier téléchargé est dans C:\Users\AXXXXXXX\Downloads.

J'utilise Vim comme éditeur par défaut pour Git.

Une fois installé, Git se situera dans le dossier: C:\Users\AXXXXXXX\AppData\Local\Programs\Git.

Git peut être utilisé dans VS Code directement avec l'onglet Source Control (Ctrl + Shift + G).

**Récapitulatif des logiciels dont tu peux avoir besoin**

* 1. Alteryx
  2. SQL developer
  3. Microsoft Power BI
  4. Keepass (stockage de mots de passe)
  5. Un IDE de code, personnellement je suggère VS Code
  6. Miniforge (ton python)
  7. R n'est pas autorisé. Tu peux essayer de faire la demande mais ce sera sans doute très long à obtenir. Cependant, plus tu le demandes tôt, plus tu as des chances de l'obtenir peut-être.
  8. Attention, Git n'est pas autorisé mais tu peux l'avoir en local
  9. Tu n'as pas accès à Gmail et tu ne peux pas mettre de clé USB dans l'ordi.

Le seul moyen que connais pour de transférer des documents et d'utiliser ta **boîte mail INSA**.