Guide détaillé
Traitement des données avec Python

Prérequis :

- Avoir effectué au moins une nuit de mesure avec le NINOX ;
- Fiche pratique ou guide détaillé « Comment récupérer les mesures NINOX ? » ;
- Fiche pratique ou guide détaillé « Traitement des données NINOX ».

Matériel nécessaire :

- Ordinateur avec les droits pour installer l'environnement de programmation Python
- En première lecture, une archive contenant un ensemble de mesures est disponible en téléchargement sur la page web du projet

Maintenant que le système NINOX est installé et qu'au moins une nuit de mesure a été effectué, la prochaine étape consiste à manipuler, traiter et analyser les données afin d'étudier le phénomène de pollution lumineuse autour du lieu d'observation.

Dans la fiche pratique et dans le guide détaillé intitulés « **Traitement des données NINOX** », plusieurs solutions ont été présentées : utiliser un tableur ou bien passer par la programmation Python.

Si vous lisez ce guide détaillé, cela signifie surement que vous avez choisi d'utiliser le langage de programmation Python pour analyser les données NINOX. Sinon, cela peut être l'occasion de découvrir ce qu'il est possible de faire avec cette solution.

Dans les sections suivantes, une explication des fichiers présents dans l'archive contenant toutes les mesures et téléchargée en suivante les indications de la fiche pratique « Comment récupérer les données NINOX ? » sera donnée puis les avantages (et quelques inconvénients) de la programmation Python pour le traitement de données seront présentés afin de confirmer ce choix de solution.

1. Présentation de l'archive contenant les mesures NINOX

A la fin de la fiche pratique « Comment récupérer les mesures NINOX ? », une archive contenant les mesures acquises par le système NINOX a été téléchargée. Cette archive contient plusieurs fichiers dont le nom commence toujours par le nom du système NINOX utilisé (ninox<numéro de série>) puis la date de téléchargement et fini par le nom spécifique de chaque fichier.

Tous ces fichiers sont au format CSV (pour *Comma-Separated Values)* qui peuvent être ouverts avec un éditeur de texte comme le bloc-notes.

Voici une description de chaque fichier :

Nom du fichier	Description
location.csv	Contient les coordonnées de chaque lieu d'observation
measure_full.csv	Contient toutes les mesures et leurs informations complémentaires (cf. tableau suivant)
ninox.csv	Contient les détails techniques du NINOX utilisé
nss.csv	Contient les différentes valeurs de NSS (*Night Sky Stability*) calculées durant les nuits de mesure. Pour plus d'informations, se référer au manuel NINOX.
sqm.csv	Contient les détails techniques du capteur SQM (*Sky Quality Meter*) utilisé dans le NINOX

En pratique et dans la suite de ce projet, seul le fichier *measure_full.csv* contient des informations intéressantes pour l'étude de la pollution lumineuse. En outre, ce fichier centralise toutes les informations contenues dans les autres fichiers de l'archive.

Explicitions donc le contenu de ce fichier.

Le fichier *measure_full.csv* contient autant de lignes que le système NINOX a effectué de mesures. Il contient également 22 colonnes, chacune contenant une information particulière :

Nom de la colonne	Description
measure_id	Identifiant de la mesure de NSB
ninox_id	Identifiant du système NINOX (contenu dans ninox.csv)
sqm_id	Identifiant du capteur SQM (contenu dans sqm.csv)
loc_id	Identifiant du site d'observation (contenu dans location.csv)
jd_utc	Date et heure de la mesure en Jours Julien UTC
az	Azimut de visée (en degré) du SQM, **par défaut 0**
alt	Altitude de visée (en degré) du SQM, **par défaut 90**
temp_sensor	Température du SQM (x100, en °C)
temp_ambiant	Température ambiant (x100, en °C), **capteur absent, -10000 par défaut**
humidity	Humidité relative (en %), **capteur absent, -100 par défaut**
pressure	Pression (en hPa), **capteur absent, -100.0 par défaut**
cloud_cover	Couverture nuageuse (en %), **capteur absent, -100 par défaut**
wind_speed	Vitesse du vent (x10, en km/h), **capteur absent, -100 par défaut**
counts	Paramètre « count » du SQM
frequency	Paramètre « frequency » du SQM
sqm_mag	Luminosité du ciel (NSB) mesuré, en mag/arcsec²
sun_alt	Altitude du Soleil (x10, en degré)
moon_alt	Altitude de la Lune (x10, en degré)
moon_phase	Phase de la Lune (x10, en degré)
gal_lon	Longitude galactique du point de visée du SQM (x10, en degré)
gal_lat	Latitude galactique du point de visée du SQM (x10, en degré)
flag_sent	Indique si la mesure à déjà été exportée (1 si oui, 0 sinon)

Ce fichier contient donc de nombreuses informations qui complètent la valeur de NSB mesurée toute les minutes.

Cependant, en réalité, certaines colonnes ne contiennent pas d'informations exploitables et le début du traitement des données NINOX passera par la simplification des mesures téléchargées en supprimant certaines colonnes inutiles dans notre cas.

Ainsi, une fois ces données récupérées, l'utilisateur peut maintenant passer au traitement de ces données pour les analyser par la suite.

2. Pourquoi choisir le traitement des données avec Python ?

Comme rappelées dans le paragraphe introductif de ce guide détaillé, il existe plusieurs solutions pour analyser les données mesurées par NINOX. Chacune possède leurs propres avantages et inconvénients.

Concernant Python, l'avantage principal est la **réutilisation des ressources développées** par la suite. En effet, il suffira de modifier quelques lignes de code pour adapter le programme à de nouveaux fichiers de mesures. En comparaison, en utilisant un tableur, chaque fichier est d'autant de feuilles de calcul différentes sur lesquelles il faut réappliquer les mêmes manipulations.

Ainsi, la programmation Python représente un **réel avantage** et un **gain de temps important sur le long terme**.

De plus, le système NINOX mesure la brillance du ciel nocturne automatique avec une fréquence d'une observation par minute. Si une nuit dure 10 heures, cela représente 600 points de mesure par nuit, nombre à multiplier par le nombre total de nuits d'observations. Au final, le fichier de stockage des valeurs relevées comporte plusieurs milliers de lignes de mesure.

Avec un tableur, ce grand nombre de mesures peut vite devenir trop lourd à gérer. En utilisant la programmation Python, le programme, plus léger, s'exécute facilement et indépendamment du nombre de valeurs.

Le traitement des données NINOX avec Python est donc **plus souple** car **indépendant du nombre de mesures effectuées** au préalable.

Enfin, l'environnement Python est un **langage sous licence libre**, **entièrement gratuit** et qui ne demande que **peu d'effort lors de l'installation**.

Néanmoins, cette solution peut présenter quelques inconvénients. Cela **demande des connaissances de base en langage Python** avant d'arriver à des résultats exploitables, ce qui peut demander du temps d'apprentissage.

Cela est à relativiser avec le fait que ce **langage est très répandu** dans la très grande majorité de toutes les industries. **Les connaissances acquises sont donc très facilement réutilisables dans de nombreux contextes différents**.

Si la lecture de cette section conforte votre choix de la programmation Python comme solution de traitement des données Python, ou si vous êtes toujours curieux.euse des manipulations possibles, la prochaine section détaillera le choix de l'environnement de programmation Python choisi ainsi que son installation.

3. Installation de l'environnement de programmation

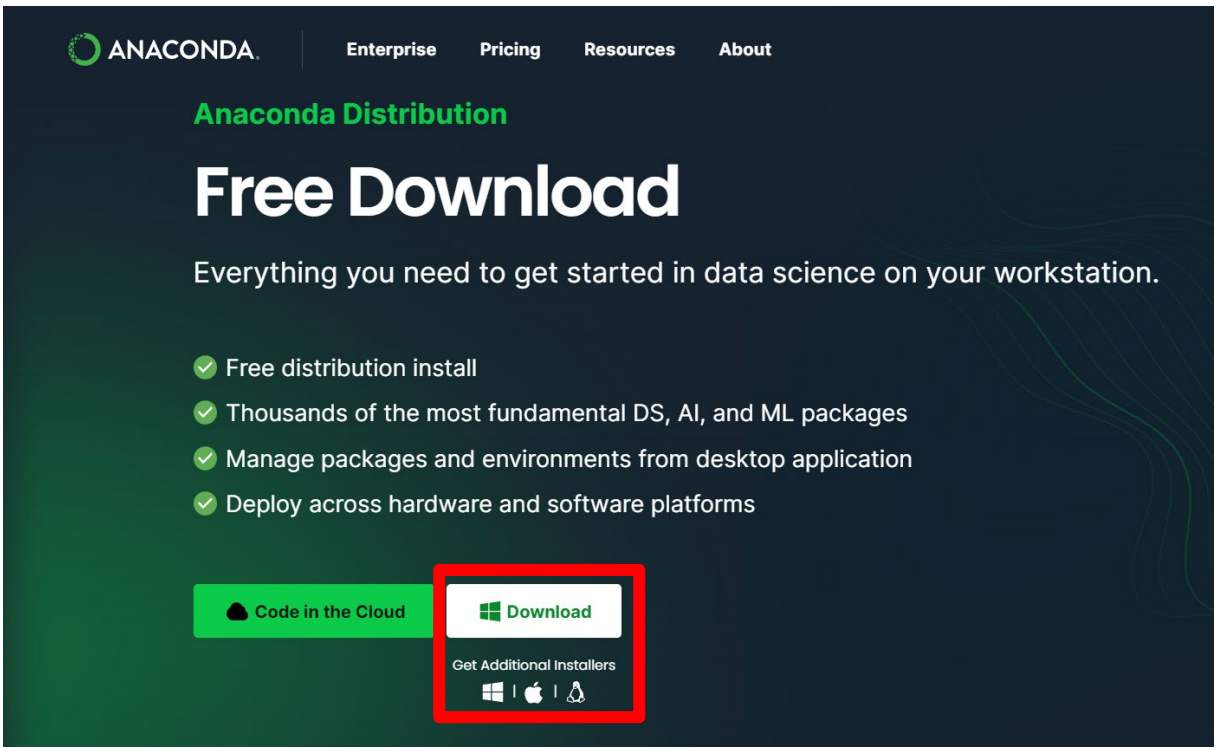
Dans la suite de ce projet, l'environnement de programmation Python qui sera utilisé est **Anaconda**.

Cet environnement est **open-source et gratuit**. L'intérêt est aussi qu'avec une seule installation, **toutes les bibliothèques de fonctions Python utilisées dans ce projet seront aussi installées sans qu'aucune manipulation supplémentaire** ne soit nécessaire. Cela simplifie grandement la suite des opérations.

Pour installer cet environnement, rendez-vous sur la page de téléchargement associée :

https://www.anaconda.com/download

Puis cliquez sur le bouton de téléchargement associé :



Le fichier d'installation se télécharge alors. La taille de ce dernier étant plutôt conséquente, plusieurs centaines de mégaoctets, le téléchargement peut prendre plusieurs minutes selon le débit de la connexion internet.

Une fois téléchargé, exécutez le fichier (il devrait se nommer *Anaconda<version_téléchargée>.exe*).

Acceptez le contrat d'utilisation et installez l'environnement Anaconda pour tous les utilisateurs si vous possédez les droits sur votre ordinateur.

Choisissez enfin le dossier d'installation de l'environnement (vous pouvez très bien laisser celui recommandé par l'installateur) puis cliquez sur le bouton *Install*.

L'installation s'effectue alors. Cette étape peut prendre quelques minutes.

Une fois l'installation terminée, Anaconda se lance et une fenêtre s'ouvre. Si rien ne se passe, lancez le programme *Anaconda navigator*.

Cela terminer cette section sur l'installation de l'environnement de programmation Python qui sera utilisé dans la suite.

La prochaine section présentera l'environnement de programmation qui vient d'être installé et plus précisément les composantes qui seront utilisées.

4. Présentation de l'environnement de programmation

L'environnement de programmation Anaconda contient de nombreuses applications utilisées dans le développement Python.

La fenêtre *Anaconda Navigator* présente toutes celles qui sont installées en même temps que l'environnement Python.

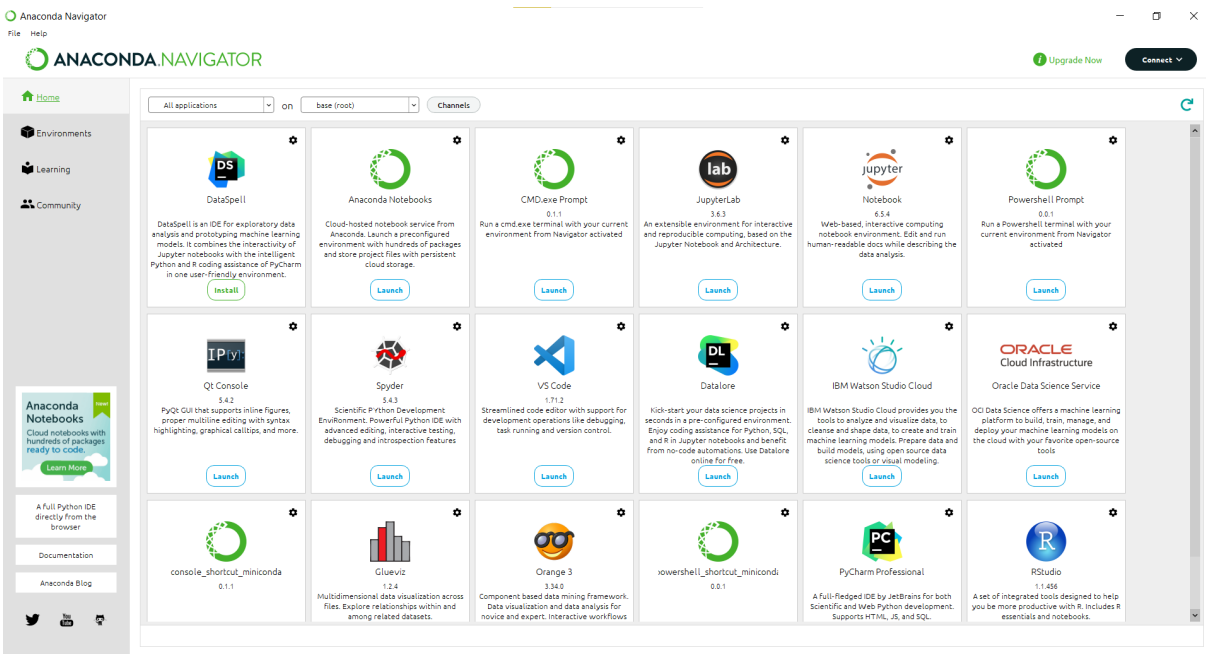
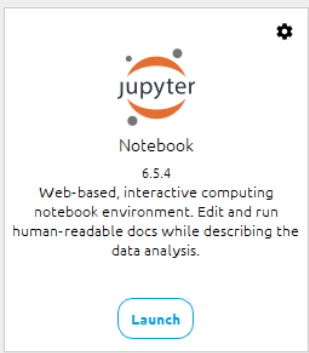


Figure 1 : Fenêtre du navigateur Anaconda

Citons par exemple *Spyder*, un éditeur de code Python (ou IDE en anglais) ou encore *RStudio* utilisé pour les études statistiques.

Dans la suite de ce guide, l'application qui sera utilisée pour le traitement et l'analyse des données NINOX est *Jupyter Notebook :*



4.1. Présentation de Jupyter Notebook

Jupyter Notebook est un programme gratuit qui utilise le navigateur internet de son choix pour programmer en Python. Comme son nom l'indique, les fichiers manipulés sont appelés *Notebook*.

Ces *Notebooks* permettent une mise en forme approfondie des fichiers Python.

En effet, un *Notebook* est composé d'une succession de cellules qui peuvent contenir du code Python ou bien du texte et des images.

Chaque cellule peut être exécutée de façon indépendante mais utilise également les données provenant des cellules précédentes, situées au-dessus de celle exécutée.

Ainsi, il faut voir un *Notebook* comme un livre ou un document dans lequel du code Python peut être exécuté au fil de la lecture.

Dans ce projet, l'utilisation de *Notebook* est avantageuse du point de vue pédagogique : avant et après chaque cellule de code, des explications seront fournies afin de comprendre l'objectif de chaque ligne de code pour faciliter l'assimilation et la réutilisation de ces connaissances.

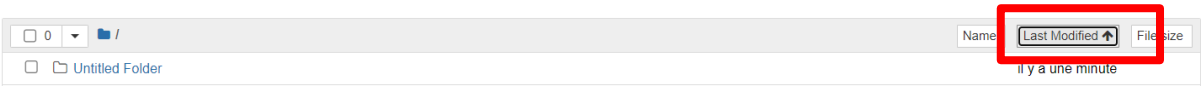
Lançons maintenant Jupyter Notebook pour découvrir l'environnement de développement Python utilisé. Pour cela, il suffit de cliquer sur le bouton *Launch* ou *Lancer* selon la langue d'installation d'Anaconda, situé sous le logo de l'application *Jupyter Notebook*.

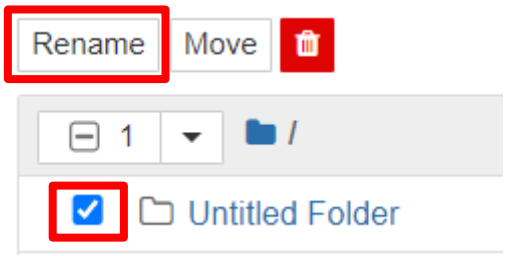
Une fenêtre du navigateur internet s'ouvre, contenant de nombreux fichiers et répertoires.

Commencez par créer un nouveau répertoire dans lequel seront sauvegardés tous les Notebooks et fichiers créés par la suite :

- Cliquez sur *New*
- Cliquer sur *Folder*

Un nouveau répertoire est créé avec le nom *Untitled Folder*. Pour le trouver facilement, vous pouvez trier les documents et répertoires par ordre de modification décroissant en cliquant sur le bouton *Last Modified* deux fois :





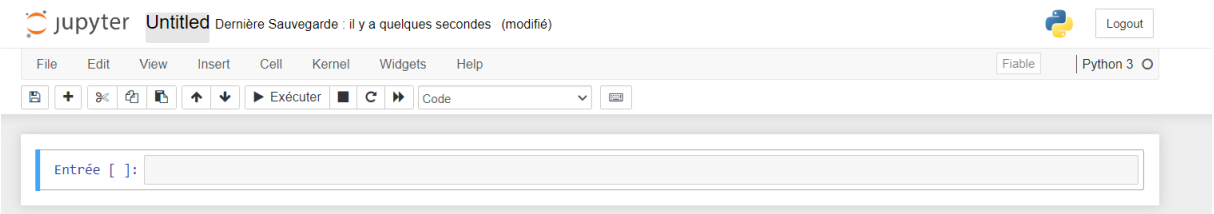
En cochant la case située à gauche du nom du répertoire puis en cliquant sur *Rename* en haut de la fenêtre, vous pouvez renommer le répertoire.

Dans la suite, ce répertoire sera nommé *Traitement donnees NINOX*.

En cliquant sur le nom du répertoire, vous vous trouvez maintenant à l'intérieur de ce dernier.

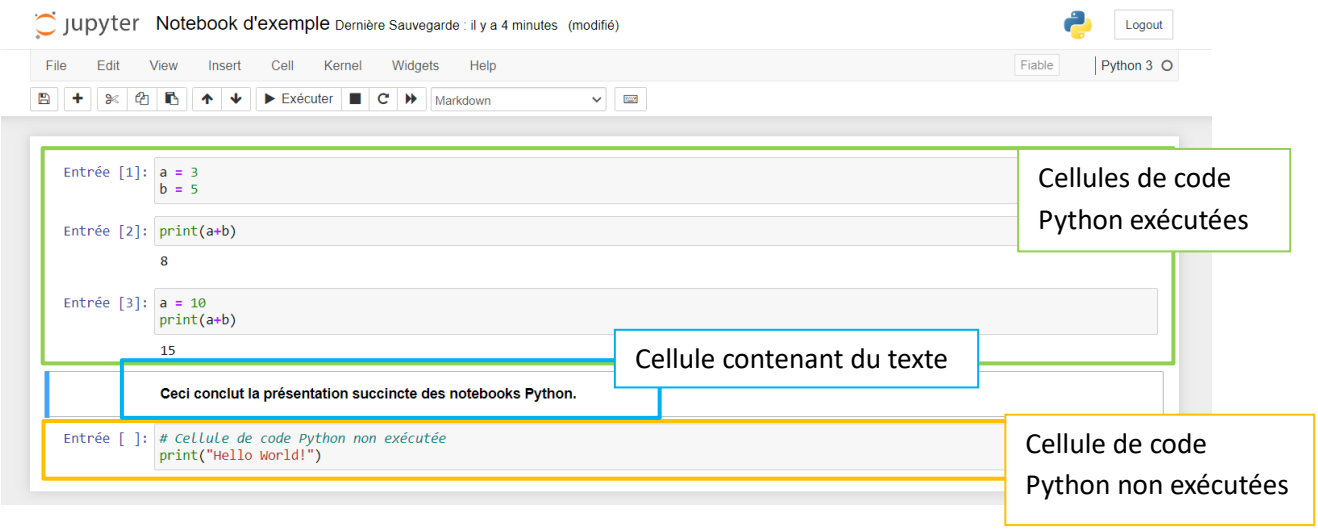
En cliquant ensuite sur *New* puis *Python 3*, vous créez un nouveau *Notebook*, ouvrant par la même occasion un nouvel onglet.

Vous pouvez renommer ce notebook en cliquant sur son nom en haut de la fenêtre :



Nous allons finir cette section en montrant une mise en forme classique d'un notebook afin que vous puissiez le manipulez facilement dans la suite de ce guide détaillé.

Voici un exemple très simple de notebook :



Encadrées en vert se trouvent des cellules de code Python qui ont été exécutées. Cela se remarque par la présence d'un nombre entre les [] à gauche de la cellule. Ces nombres représentent l'ordre chronologique d'exécution des cellules. Lorsqu'aucun nombre n'est présent entre les crochets, cela signifie que la cellule de code n'a pas encore été exécutée, comme celle encadrée en orange dans l'image précédente.

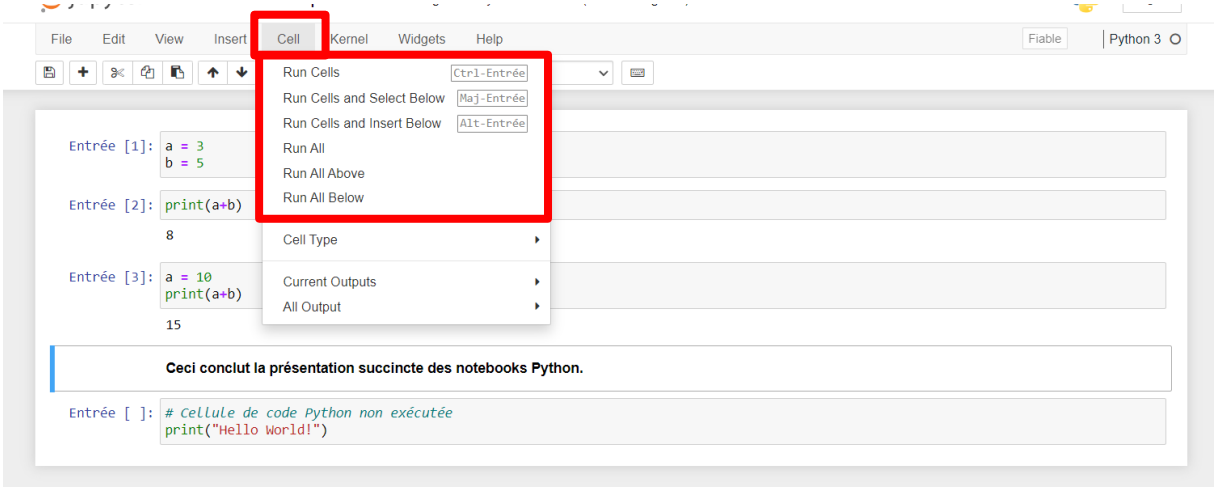
Dans les cellules exécutées, nous pouvons voir qu'elles peuvent être exécutées indépendamment mais que l'ordre d'exécution peut avoir de l'importance.

En effet, si la deuxième cellule est exécutée avant la première, cela génère une erreur dans le programme. Ainsi, dans la suite, il faudra être vigilant quant à l'ordre d'exécution des cellules.

Pour exécuter une cellule à la fois, il suffit de cliquer dans celle que l'on souhaite exécuter puis de cliquer sur le bouton 

.

Pour accéder à d'autres options d'exécution (toutes les cellules du notebook, toutes les situées au-dessus ou en-dessous d'une cellule, …), vous pouvez cliquer sur le bouton *Cell* puis choisir l'option désirée :



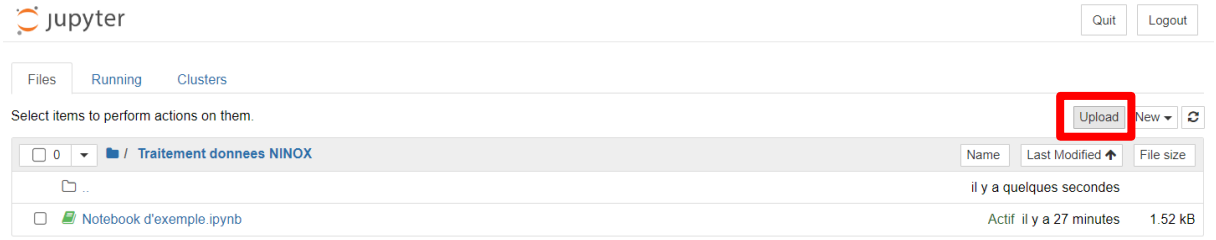
Maintenant qu'une présentation basique de l'environnement de développement Python qui sera utilisé dans le traitement et l'analyse des données NINOX a été donnée, la prochaine partie présentera les différentes manipulations qui seront réalisées dans le notebook tutoriel disponible sur la page web du projet.

(Pour celles et ceux qui ne souhaitent pas utiliser Jupyter Notebook, il est tout à fait possible d'utiliser un éditeur de code classique. Néanmoins, le guide n'est pas disponible dans ce format. Une façon de faire est d'ouvrir le notebook tutoriel dans Jupyter Notebook et de reporter cellule après cellule, le code dans un fichier Python classique.)

5. Présentation des manipulations dans le notebook Python

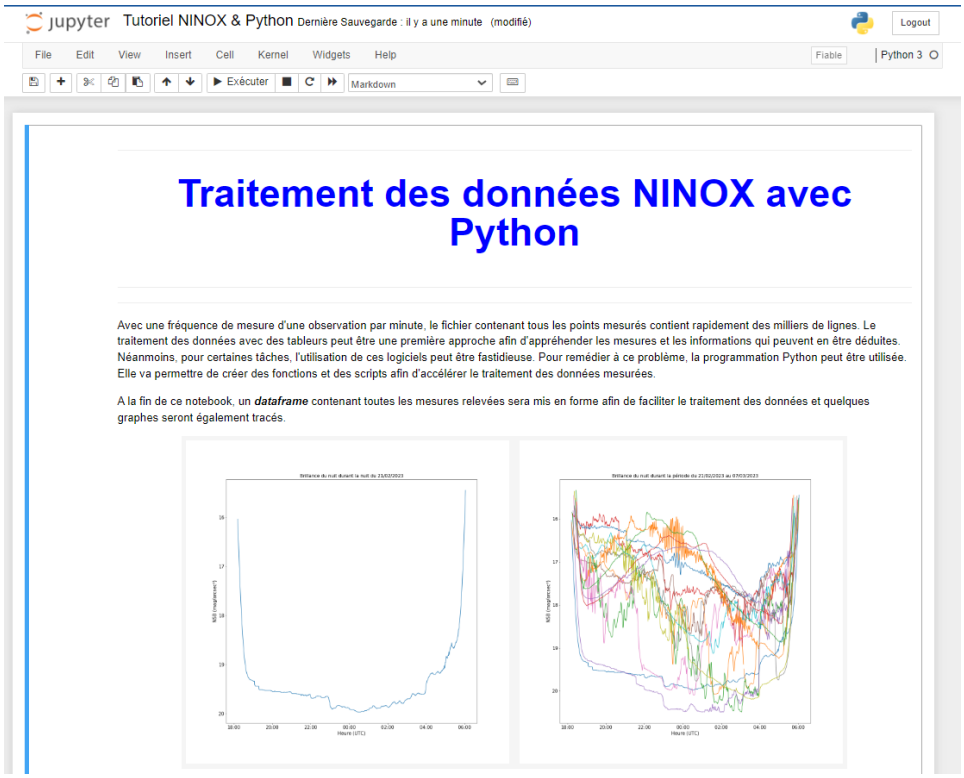
Toutes les manipulations de traitement et d'analyse des données NINOX sont détaillées dans le notebook intitulé *Tutoriel NINOX & Python* disponible en téléchargement sur la page web du projet.

Pour l'ouvrir dans Jupyter Notebook, il suffit de cliquer sur le bouton *Upload* présent dans la fenêtre Jupyter Notebook puis de naviguer dans vos répertoires afin de choisir le bon fichier :



Puis de cliquer sur le bouton *Téléverser* :

En cliquant ensuite sur le nom du notebook, ce dernier s'ouvre dans un nouvel onglet.



Ce notebook est construit de façon chronologique, dans l'ordre de lecture. Ainsi, chaque manipulation dépend des précédentes. Il faut donc exécuter une à une les cellules de code en lisant avec attention chaque cellule de texte.

En effet, **initialement, le notebook n'est pas exécutable**. Comme détaillé dans ce notebook, l'utilisateur doit modifier quelques paramètres comme le chemin d'accès au fichier contenant toutes les données NINOX ou encore les chemins de sauvegarde de fichiers et de graphiques créés par le notebook lors de son exécution.

C'est pour cela qu'une exécution cellule par cellule est nécessaire, surtout lors que la première ouverture de ce notebook.

En suivant étape par étape le notebook *Tutoriel NINOX & Python*, l'utilisateur sera amené à découvrir plusieurs bibliothèques Python :

- ***Pandas***, dédiée à la manipulation, au traitement et à l'analyse d'un grand nombre de données.
- ***Matplotlib***, pour la création et l'affichage de graphique
- ***Numpy***, pour l'utilisation de fonctions mathématiques
- ***…***

Voici un résumé chronologique des manipulations présentées dans le notebook *Tutoriel NINOX & Python :*

- Les valeurs mesurées par le système NINOX seront importées à partir du fichier de sauvegarde puis converties en *dataframe*, une structure de donnée propre à Pandas, semblable aux feuilles de calcul d'un tableur
- Les données seront pour certaines converties (pour afficher la date par exemple)
- Plusieurs types de graphiques seront tracés pour commencer l'analyse des mesures.

6. Et maintenant ?

Après la lecture de ce guide détaillé sur le traitement des données NINOX avec Python, vous pouvez :

- Télécharger le notebook *Tutoriel NINOX & Python* disponible sur la page web de ce projet pour lire les instructions

- Télécharger une archive contenant des mesures d'exemple (celles utilisées pour la rédaction du notebook tutoriel) pour essayer le traitement des données en Python avant l'acquisition d'un NINOX par exemple

- Exécuter étape par étape les instructions du notebook tutoriel et les appliquer aux valeurs d'exemple

- Appliquer les manipulations présentées dans le notebook tutoriel à vos propres valeurs mesurées.

- Approfondir le traitement et l'analyse des données NINOX grâce à la programmation Python