|  |  |
| --- | --- |
|  | Il vous est demandé de rendre un rapport expliquant votre travail et soulignant:    - Votre maitrise de python (par exemple, en expliquant vos choix d'implémenter vos fonctions, le choix des moyennes, etc..)    - Votre maitrise de l'algorithme et votre capacité de répondre à un problème à l'aide d'un programme Python (par exemple, en expliquant vos algorithmes, leurs complexités, pourquoi les avoirs choisis, etc.) |

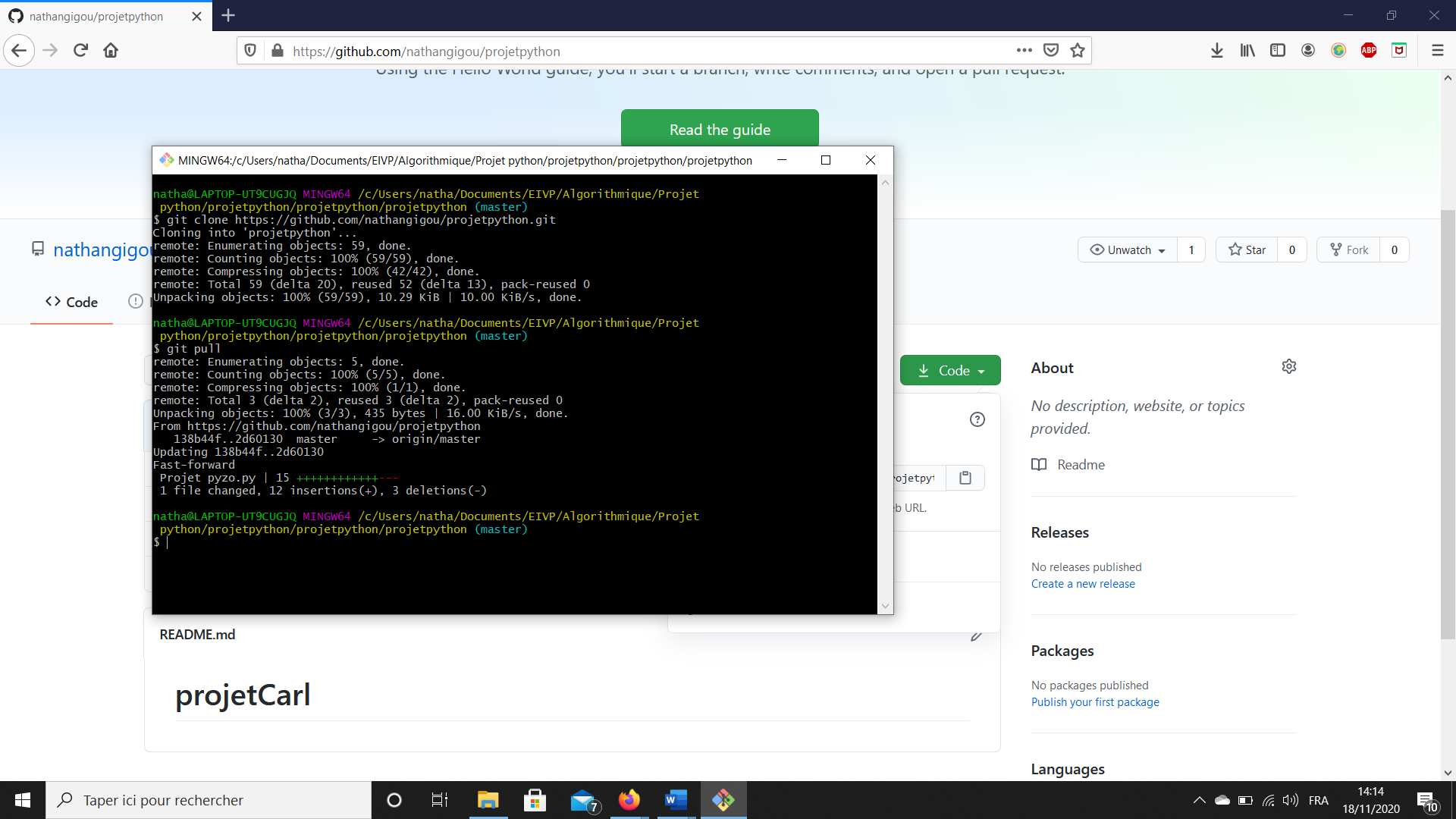
**Rapport Projet Programmation Python IVP1**

1. **Github/Git**

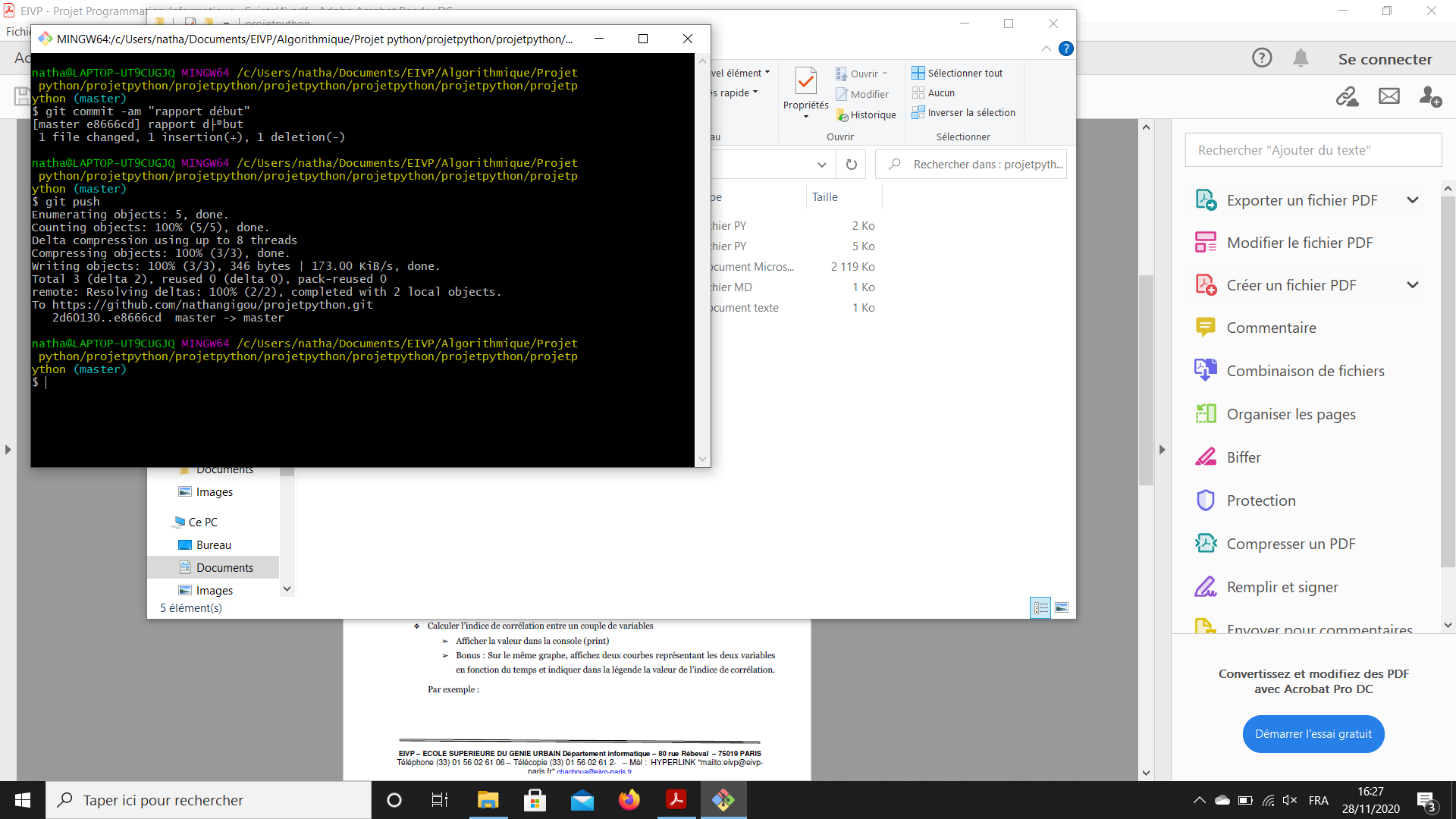
Après plusieurs jours, nous avons compris comment appréhender l’interface de partage qu’est Github et nous sommes familiarisé avec les commades de base de Git. En effet le plus complexe était de bien initialiser la liaison entre notre dossier de travail personnel et Github. De plus, une mise en main efficace de ce logiciel demandait une bonne compréhension des différentes branches de modifications.

Nous avons donc créé un répertoire sur Github accessible à tous les deux. Même si la prise en main n’est pas tout de suite intuitive, le résultat est à la hauteur car il simplifie grandement l’interaction d’écriture du programme et garantit non seulement un accès, mais aussi une sauvegarde et un historique.

En effet, nous utilisions la console Git Bash : pour récupérer les modifications apportées par le binôme nous utilisons*git clone/git pull*



Pour envoyer nos propres modifications au serveur, nous utilisons *git commit -am/git push*



Enfin pour ajouter un document nous utilisons *git status/git add*

*Screen*

Github nous permet ensuite de visualiser rapidement les modifications et les commentaires associés.

1. **Compréhension du sujet**

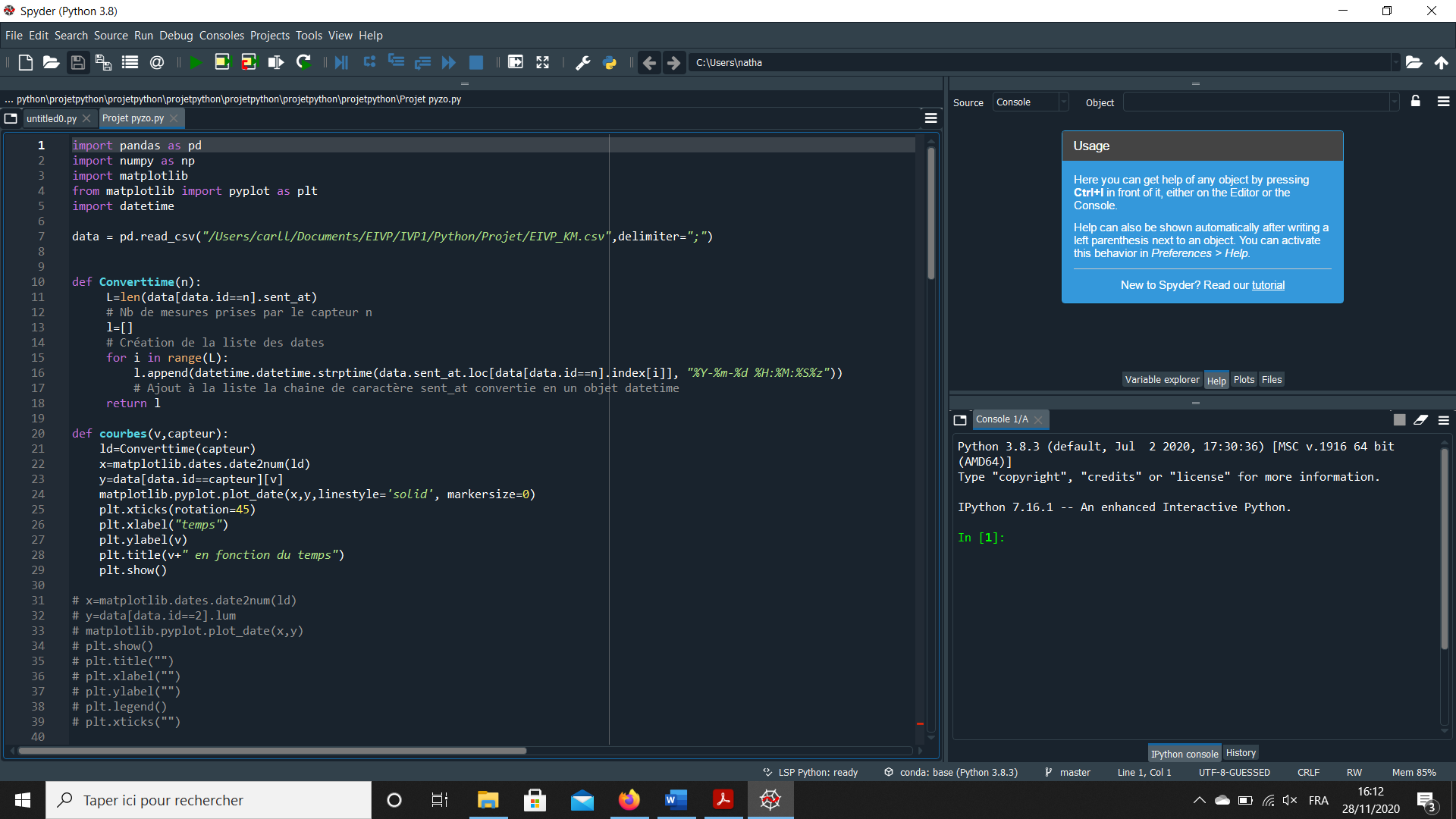
Le but du projet est d’appréhender la gestion/ le traitement de données, ici à partir d’un fichier CSV.

On va donc modéliser des représentations graphiques, des points/valeurs clés et évaluer les liens entre les sous-ensembles de données.

En effet, notre tableau initial comprend des relevés de 6 différents capteurs, des paramètres lumière-co2-bruit-humidité-température, relevés recueillis à différents horaires quasi-périodiquement.

1. **Ecriture du programme**

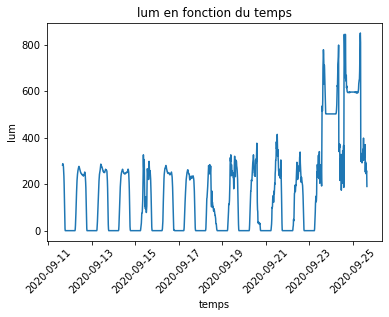
La première étape est tout simplement de lire le fichier CSV grâce à Python et pour mieux appréhender le format horaire de l’heure des relevés, nous séparons le fuseau horaire, le mois, le jour, l’heure, la minute et la seconde grâce au module datetime. Cela nous facilite l’utilisation de cette donnée.

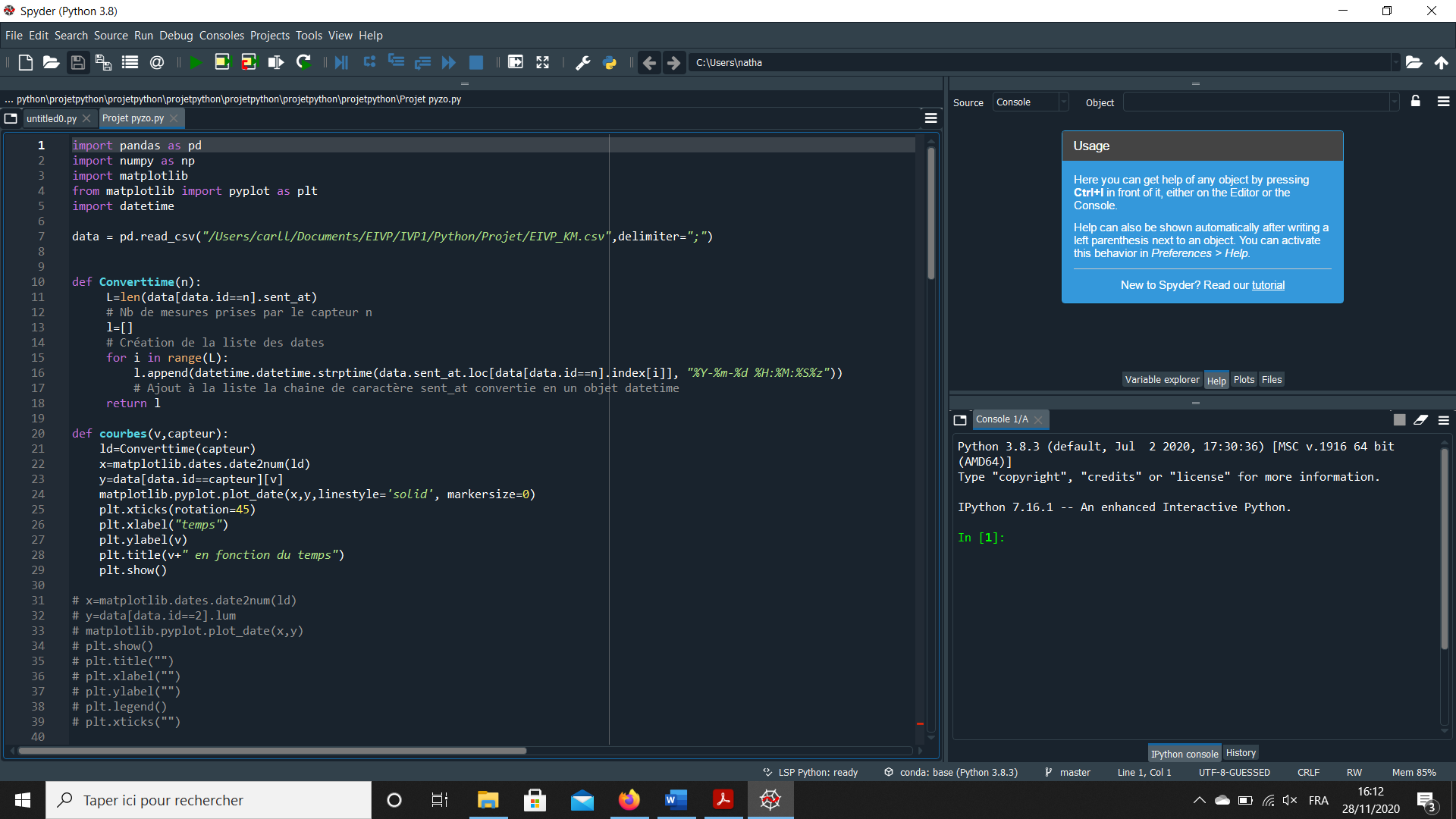


* Affichage des courbes d’un paramètre particulier, d’un capteur particulier en fonction du temps

Il s’agit ici de sélectionner les plages de données qui nous intéressent et d’être capable de sélectionner un capteur et un paramètre à partir du fichier brut.

Par exemple, on veut afficher la luminosité du capteur 1 :



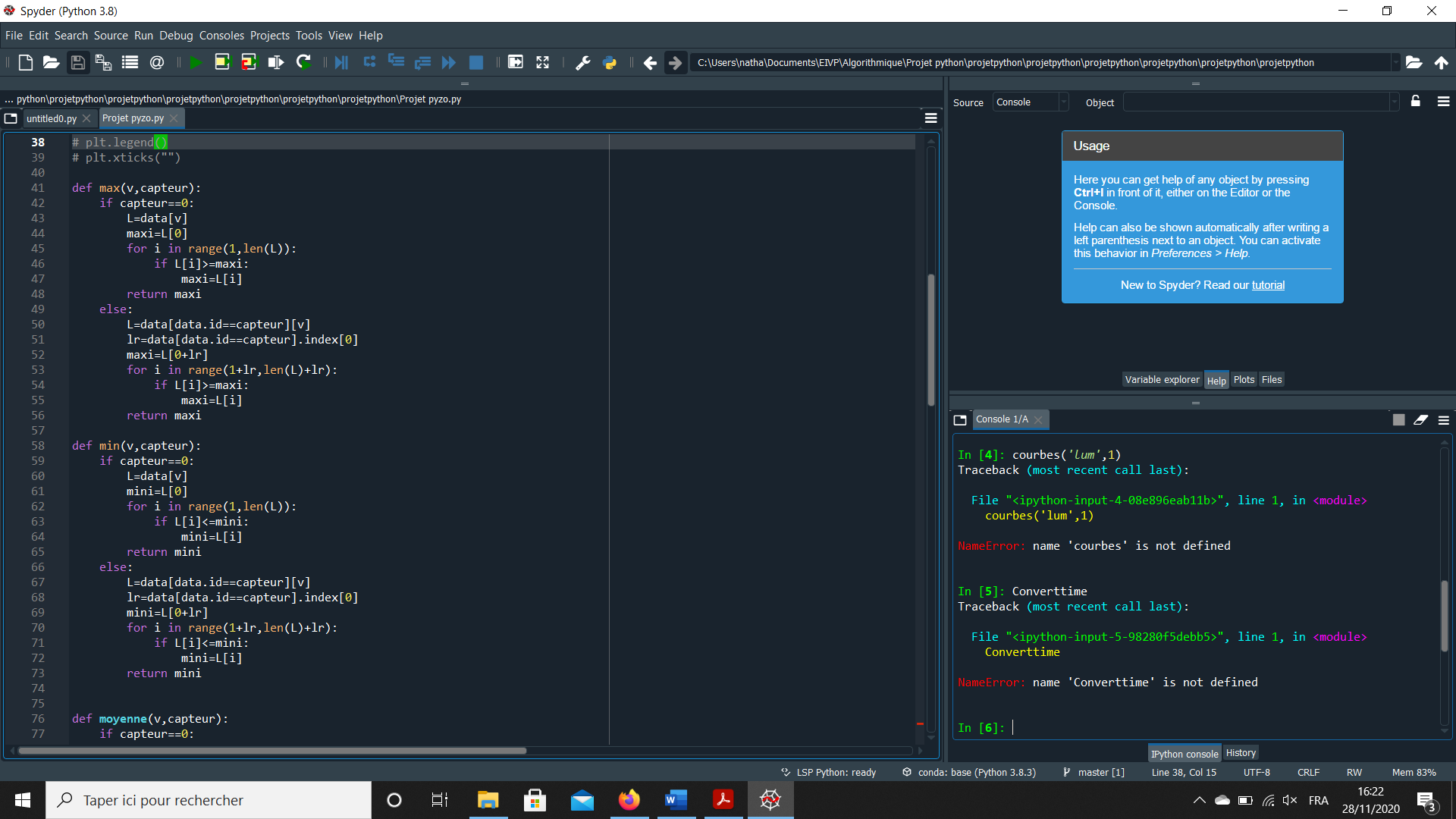


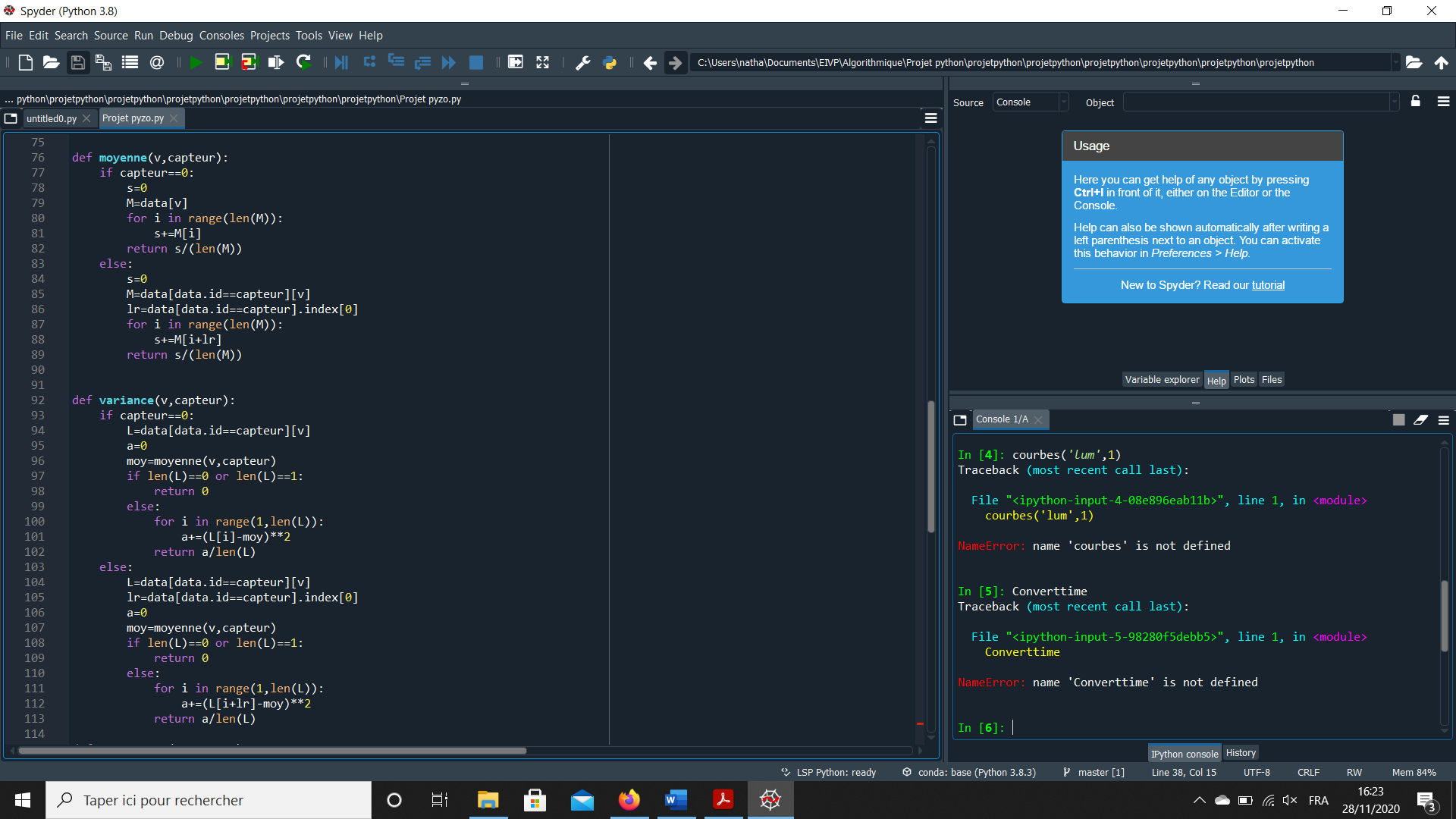
On affilie à l’axe des abscisses la donnée temporelle convertie préalablement, l’axe des ordonnées dépend des deux paramètres de la fonction à savoir le numéro du capteur et v l’entité qu’on veut afficher (luminosité,température…).

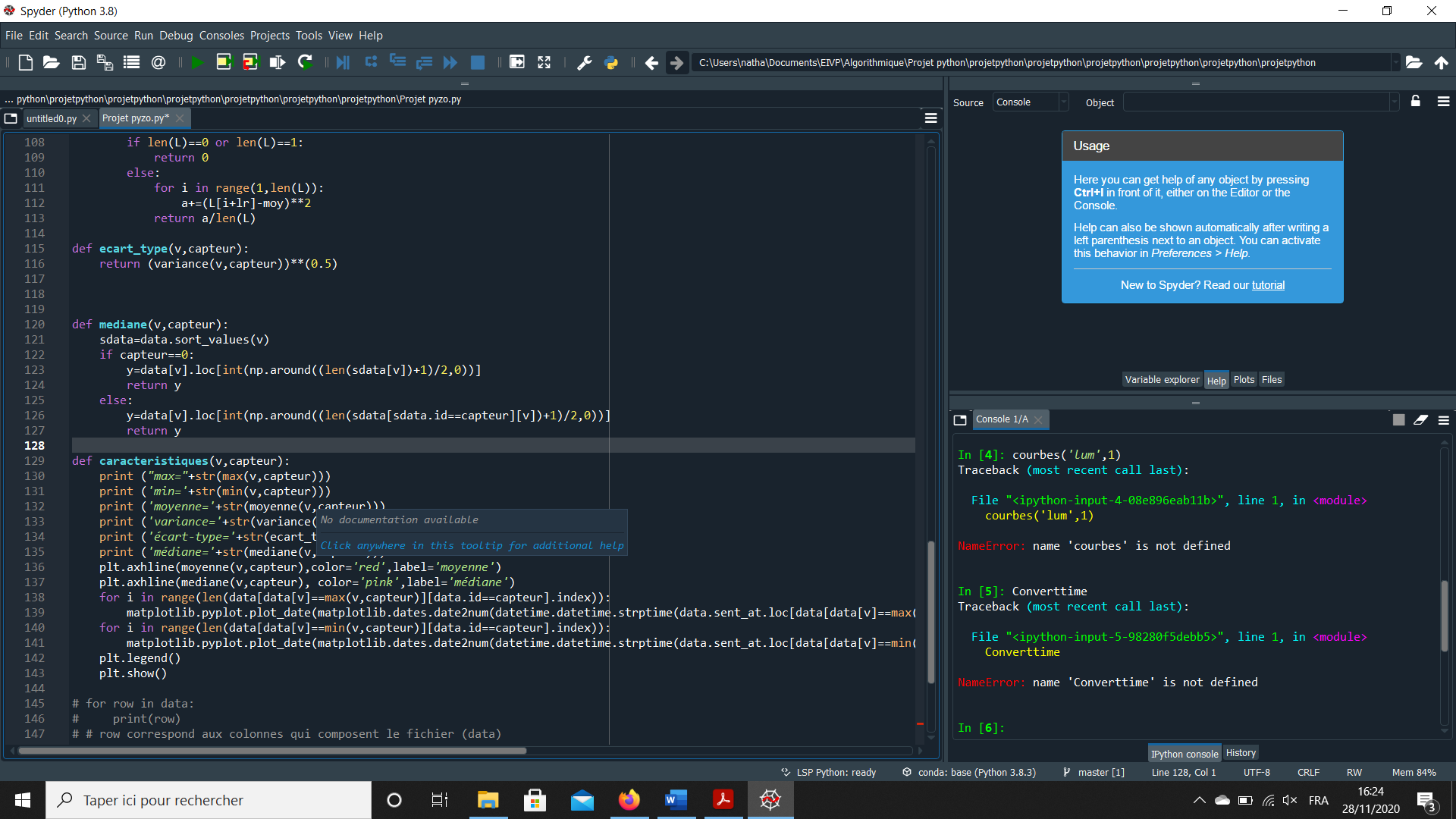
On utilise ensuite la fonction plot pour afficher, le reste des commandes est de la mise en forme.

* Affichage des valeurs statistiques (min, max…) sur la courbe

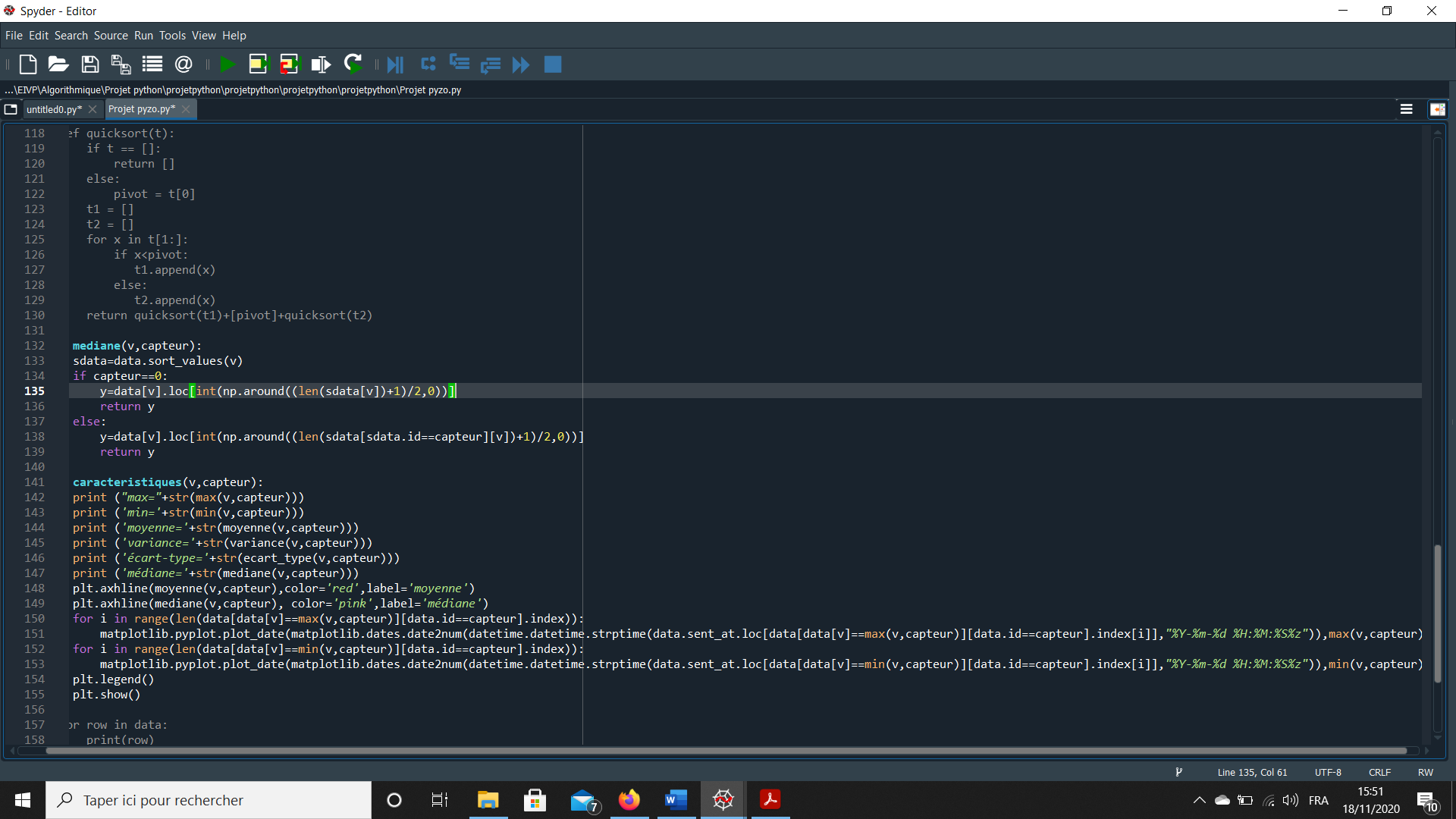
D’abord on écrit chaque sous fonctions :







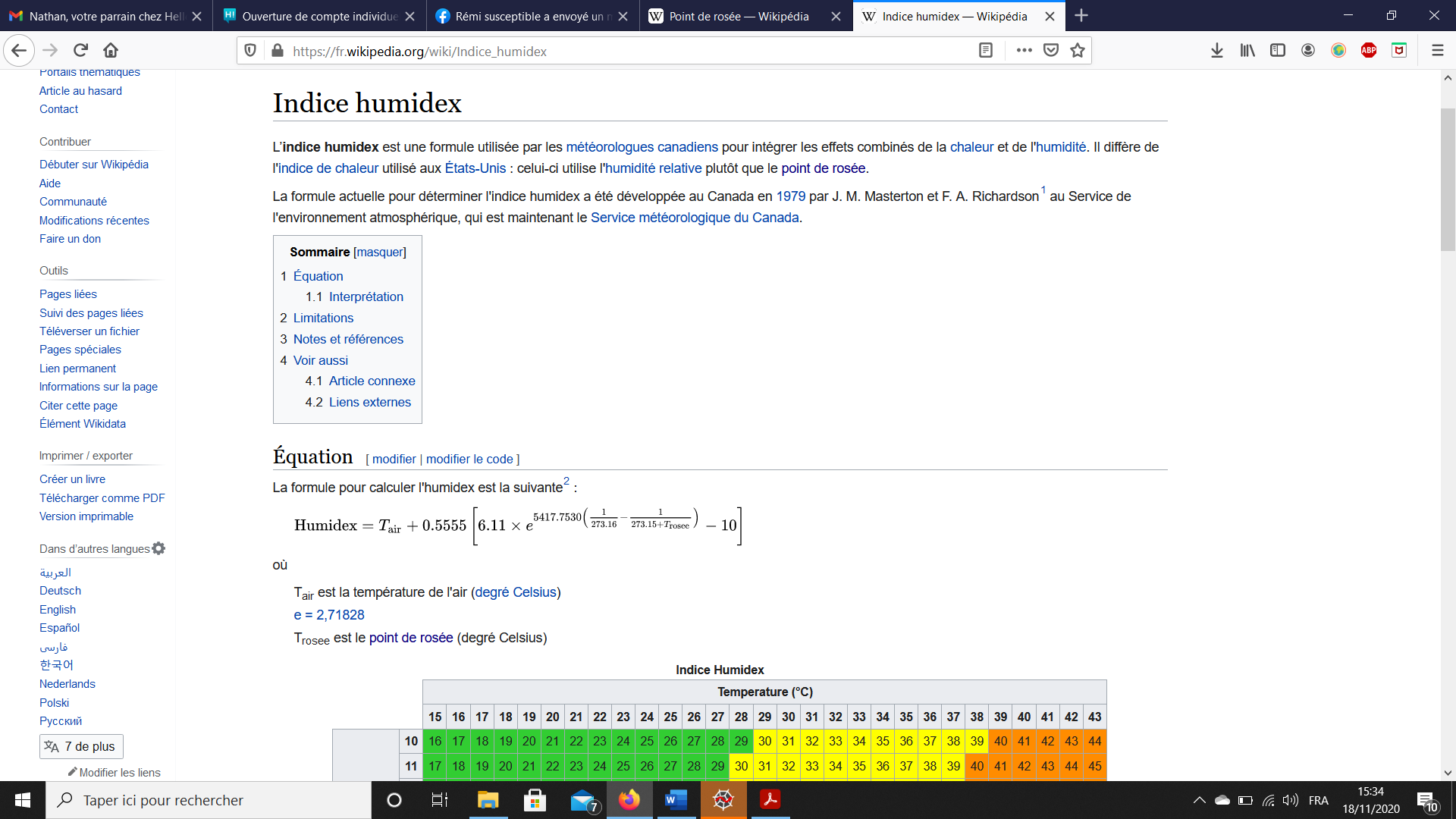
Puis la fonction d’affichage qui reprend les fonctions précédentes :



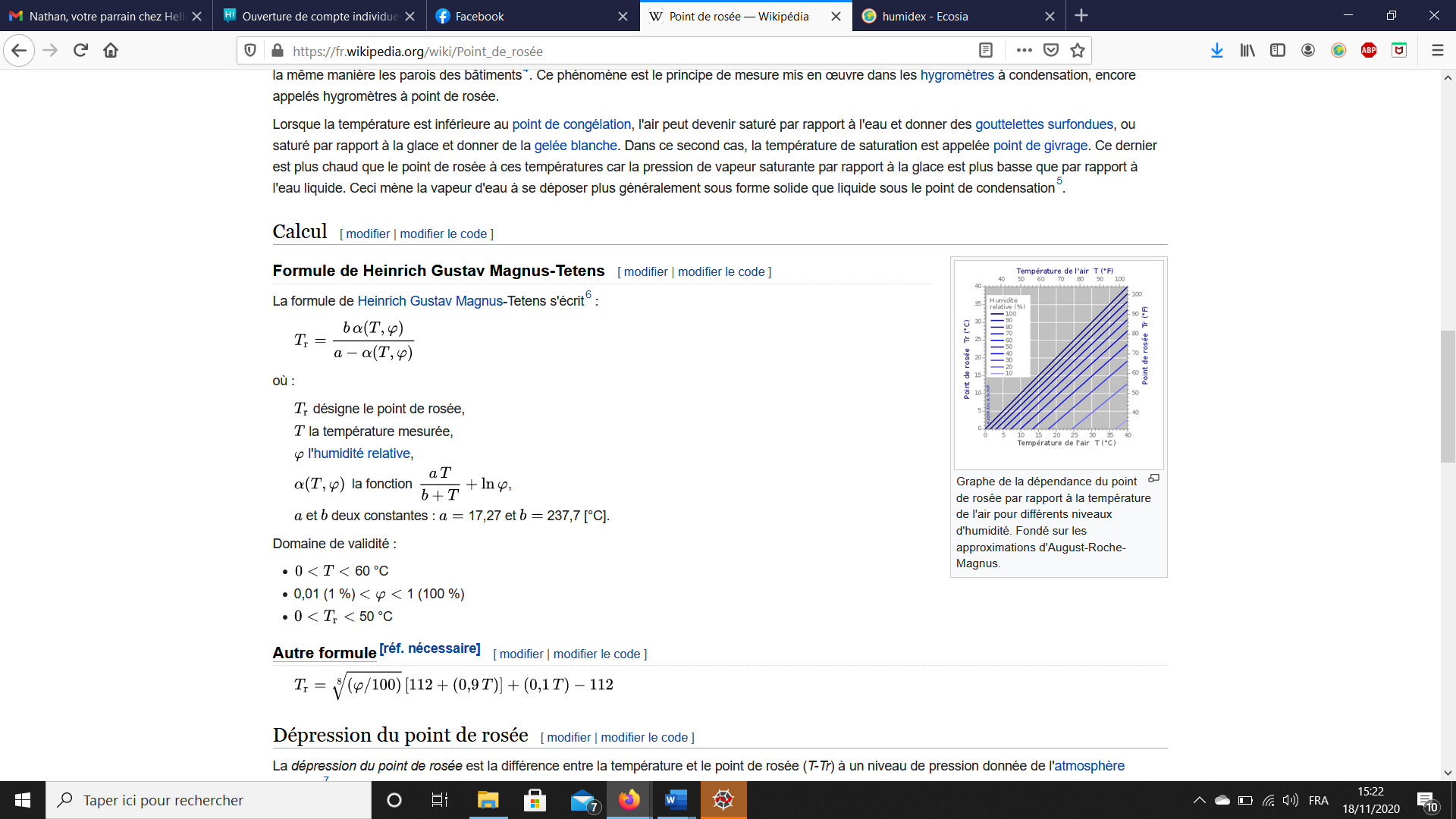
Screen courbe avec points particuliers

* Humidex

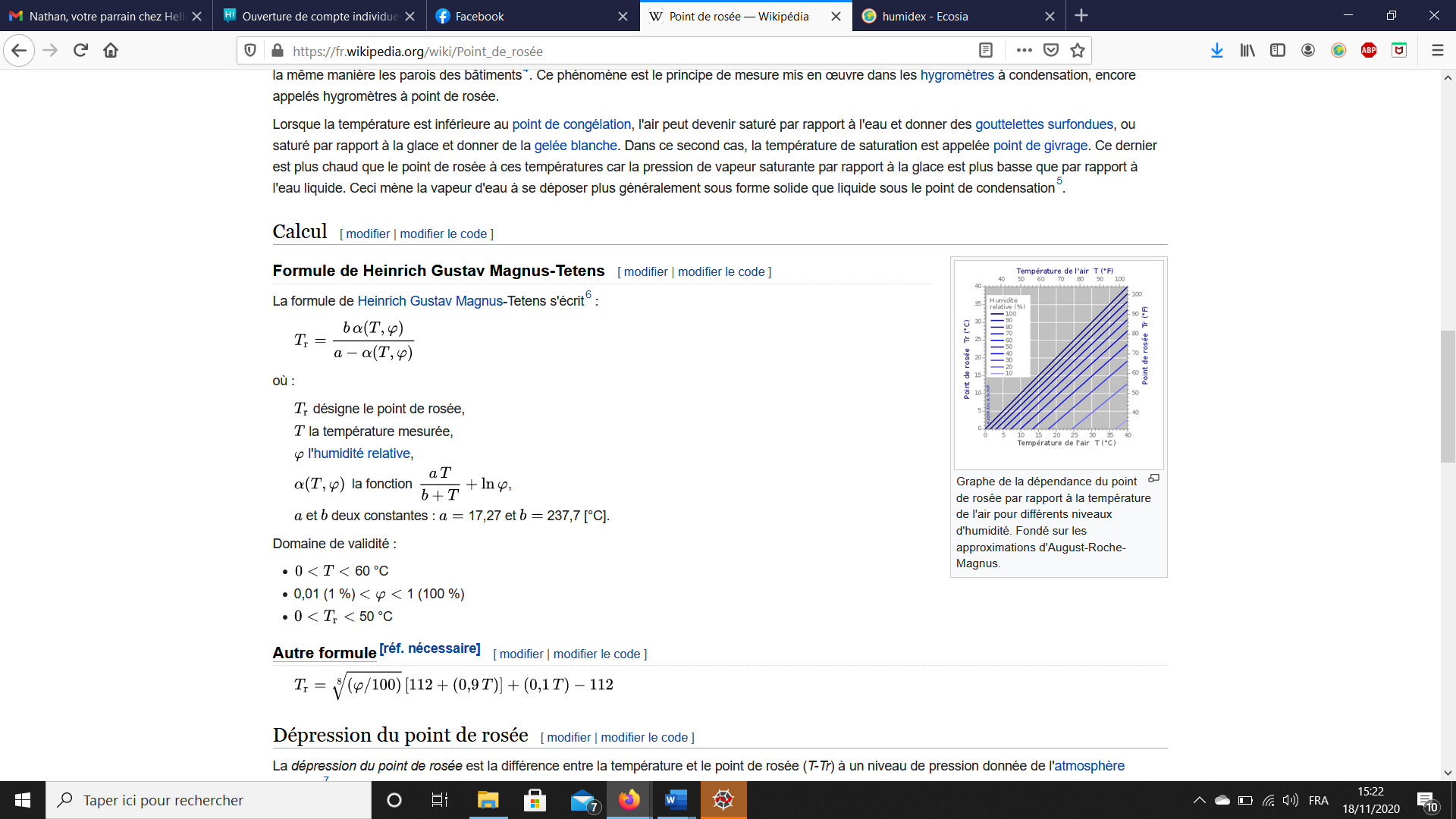
Après de rapides recherches nous avons trouvé la formule de l’humidex (indice permettant d’intégrer les effets combinés de l’humidité et la chaleur)



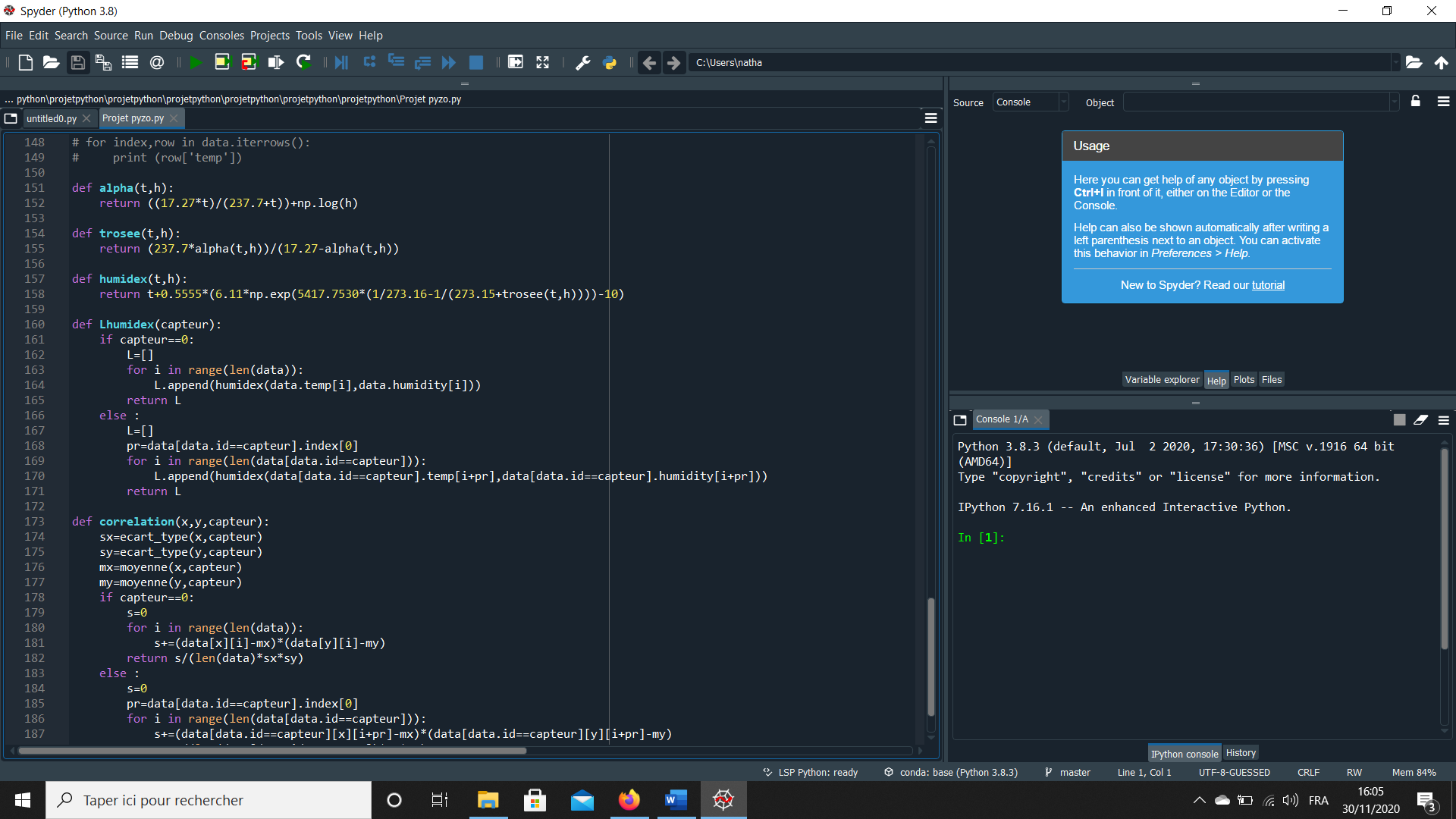
Nous avons besoin du point de rosée dont la formule apparaît ci-dessous :



Sachant que cette formule demande une dernière formule alpha :



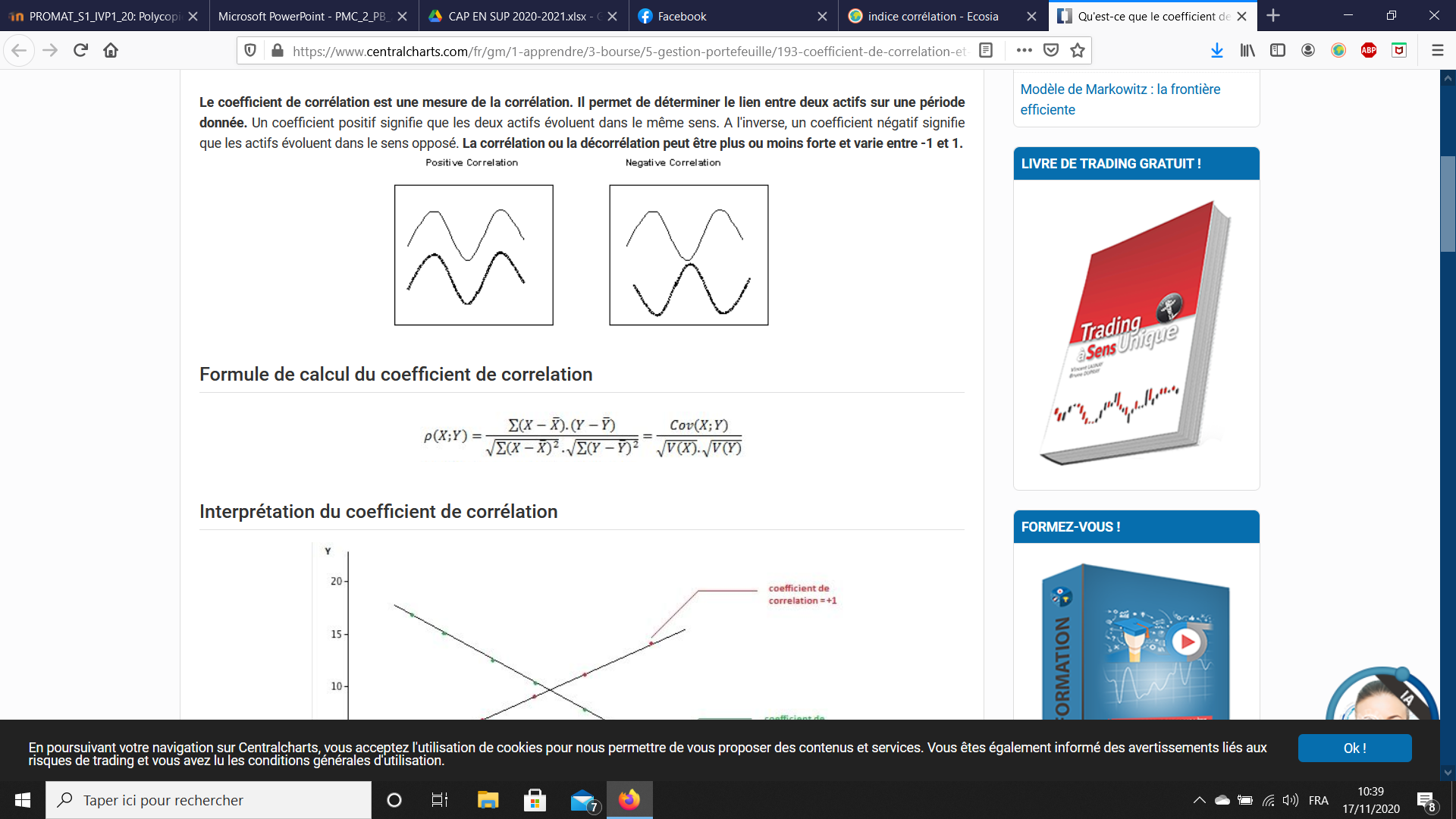
Ce qui se traduit au niveau du code par :

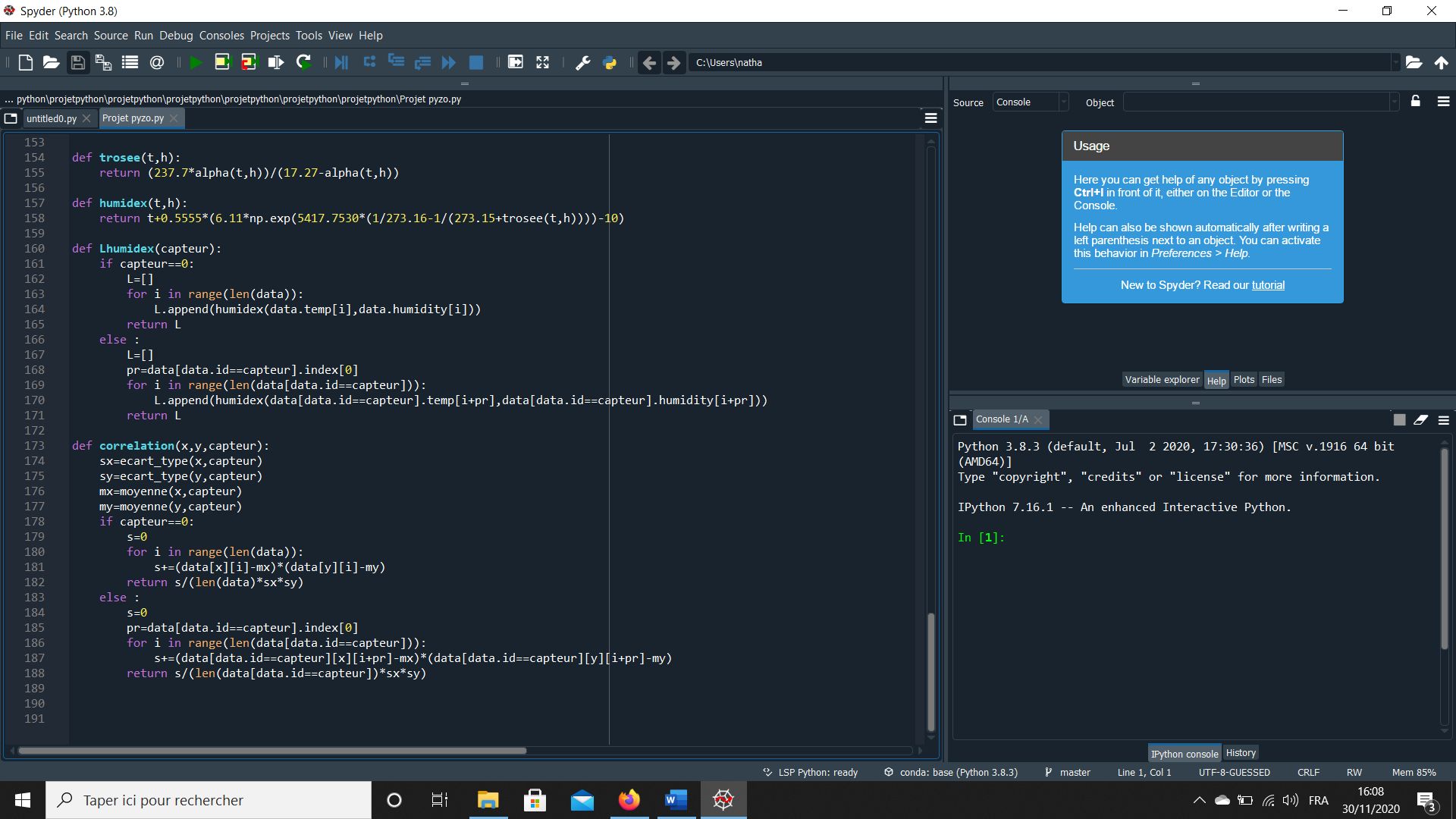


Notre fonction, grâce à la boucle retourne tous les humidex du capteur choisi.

* Indice de corrélation

Ci-dessous la formule





La somme est réalisée grâce à la boucle et on utilise les fonctions moyennes et écart-type déjà écrites.

* Recherche des similarités

On souhaite être capable de savoir si les comportements des capteurs sont proches, s’ils sont posés à des endroits proches ou non, on va donc s’intéresser aux écarts points par points et admettre un pourcentage d’écart considéré comme indicateur de l’équivalence (- de 5% = similaire).

Seulement pour réaliser cette comparaison point par point, il faut des listes de relevés de même taille et dont les dates de prises de relevés sont coïncidentes. On remarque que les relevés sont effectués environ toutes les 15 minutes pour chaque capteurs. Les relevés sont légèrement décalés entre chaque capteur mais grâce à un intervalle de 8 minutes on peut retrouver les valeurs liés à une même plage horaire :

Screen excel avec 6 capteurs

Le soucis est alors d’avoir des tableaux de même longueurs mais 3 problèmes viennent à jour :

* Certains capteurs commencent les relevés plus tôt ou finissent plus tard, il faut donc tronquer les listes pour qu’elles se coordonnent sur une même plage horaire commune

Screen fonction

* Il peut y avoir des valeurs de relevés inexistantes
* Il peut y avoir des valeurs de relevés dans une intervalle très proche, bien inférieure aux 15min régulières

Screen fonction

Une fois ces soucis réglés, on peut grâce à une double boucle parcourir les données des capteurs point par point et comparer s’ils sont similaires.

Screen fonction

On représentera ensuite par un point de couleur les points similaires, ce qui permettra de discerner à quel endroit de la courbe les données sont similaires ou non.

1. **Mise en forme de l’écriture sur console**

On va utiliser la banque cis et écrire une fonction globale reprenant nos fonctions déjà crées.

*Screen fonction*

**Lien Github:**