Rapport Python

Le but du projet Python était de nous faire travailler le design d’algorithme, leur implémentation et l’utilisation d’outil de gestion de code informatique. Notre sujet consistait à trouver des anomalies dans les données fournies, issues d’une campagne de mesure au sein d’un bâtiment de bureau. Les données sont proposées par Kandu et sont composées entre autres de la mesure de la température ambiante (°C), de l’humidité relative (%), du niveau sonore (dBA), du niveau lumineux (lux), de la quantité de CO2 (ppm).

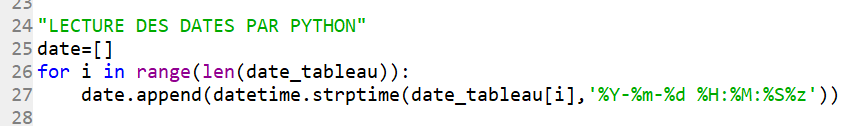


Dans la première partie de l’algorithme, le but était de lire les données CSV. Pour cela, nous avons utilisé la bibliothèque panda (l.8) en utilisant la fonction « pd.read » (l.15) et ouvert notre fichier CSV nommé : EIVP\_KM.csv. Chaque colonne du tableau étant séparé d’un « ; » nous avons décidé de créer une séparation des colonnes du tableau par ce signe. Dans un premier temps, nous avions créé une séparation à partir du nom des colonnes, mais l’algorithme a rencontré des difficultés à reconnaitre le nom de nos variables.

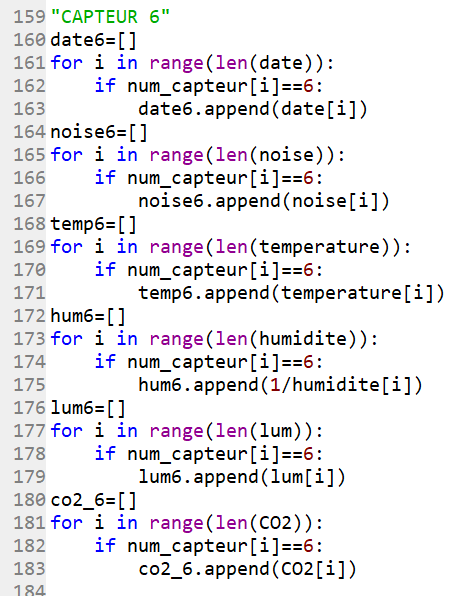
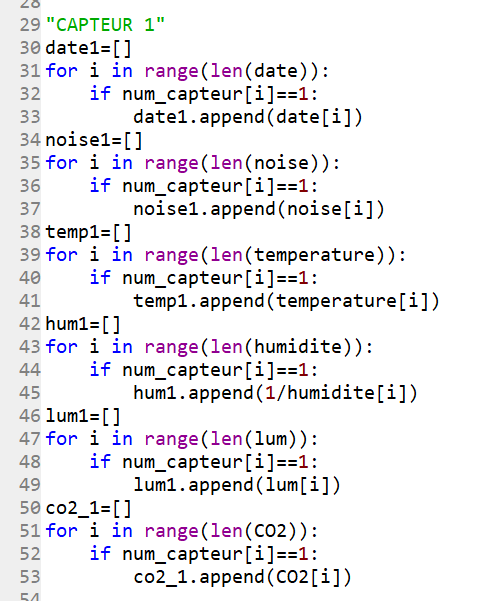
Pour créer notre graphe, nous devions obtenir des listes composées des différentes valeurs de chacune des colonnes. Dans un premier temps, nous avons voulu utiliser la fonction Dataframe de la bibliothèque. Finalement, nous avons opté pour l’utilisation de la fonction tolist (l.16 à l.22), car cette dernière nous semblait plus intuitive ; ainsi dans chaque liste se trouvent les données de tous les capteurs. Nous avons donc dû remédier à deux problèmes : le premier est le fait que les dates se trouvant dans la colonne, n’étaient pas reconnaissables par Python ; le deuxième est le fait qu’aucune des listes n’étaient séparées par capteur.

Par conséquent, le graphe créé reprenait les valeurs de chaque capteur ce qui le rendait illisible.

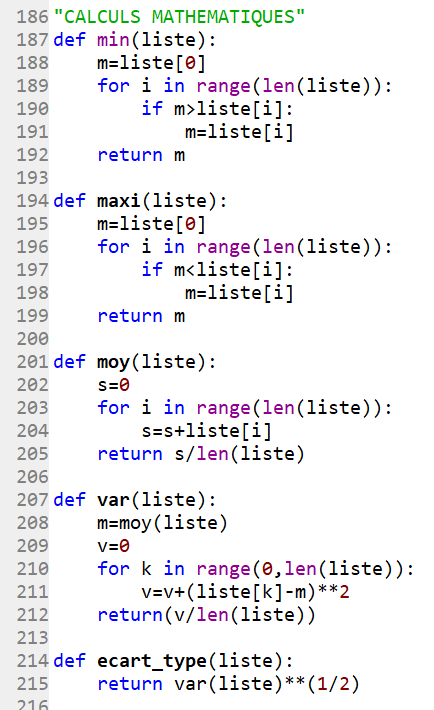
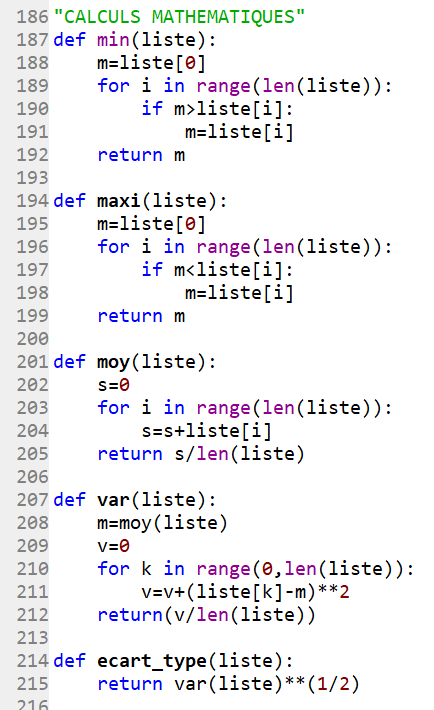


Pour remédier au problème de la date, nous avons importé la bibliothèque datetime (l.9) et créé une liste vide. Ainsi, nous avons transformé, les données de la colonne send\_at du tableau CSV (l.17) en des données pouvant être lus et traitées correctement par Python. Puis nous les avons ajoutées à la liste vide pour obtenir une liste de date fonctionnelle (l.25 à l.27).

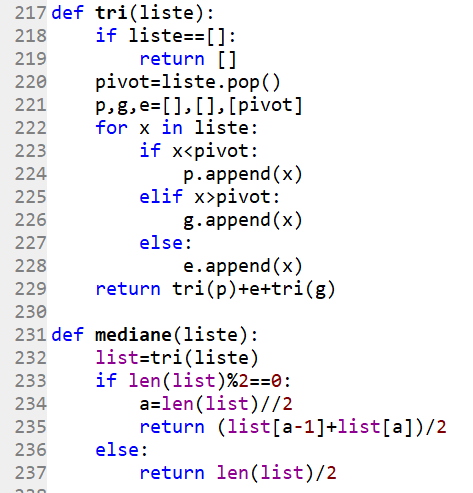
Ensuite, nous avons voulu remédier au fait que dans chaque liste (température, humidité, etc…) les valeurs de tous les capteurs étaient présents. Puisque le graphe n’était alors pas exploitable car illisible, nous avons créé de nouvelles listes, chacune associée à un numéro de capteur, contenant les valeurs d’une grandeur choisie. Ci-dessous, nous avons l’inventaire des listes des grandeurs qu’ont mesurées les capteurs 1 et 6. Enfin il était nécessaire de le faire pour chaque liste car toutes doivent posséder le même nombre de valeurs.

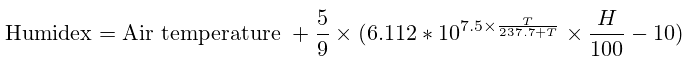


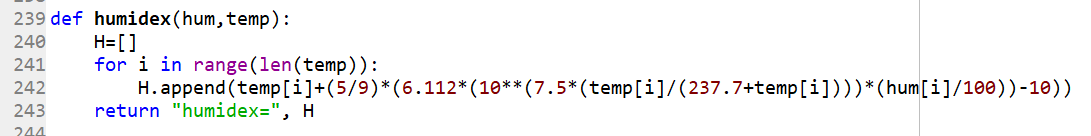
Ensuite, nous avons créé des fonctions permettant de déterminer certaines valeurs caractéristiques pour chaque liste dans le but de les reporter ensuite sur le graphe associé. Ainsi, les fonctions créées permettent de calculer dans l’ordre : le minimum, le maximum, la moyenne, la variance et l’écart-type.



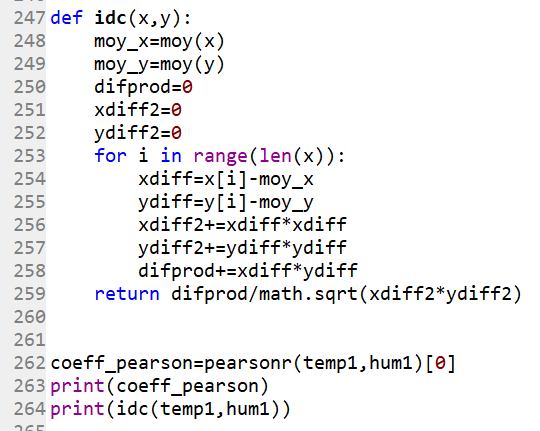
Pour déterminer la médiane d’une liste, il faut que celle-ci soit triée préalablement. Ainsi, nous avons utilisé un algorithme de tri (l.217) pour pouvoir créer ensuite une fonction médiane valable (l.231).



Par la suite, il est demandé de calculer l’indice humidex. Il se calcule à l’aide de cette formule :

Avec T la température et H l’humidité relative. Ainsi, nous avons créé une fonction qui calcule pour chaque ligne du tableau CSV de départ l’indice humidex correspondant.

Par la suite, il était demandé de calculer l’indice de corrélation.

Pour vérifier les résultats de notre fonction, nous avons importé la fonction pearsonr depuis la bibliothèque scipy.stats. Cette fonction permet de calculer le coefficient de corrélation de Pearson.

Utilisation de GitHub :