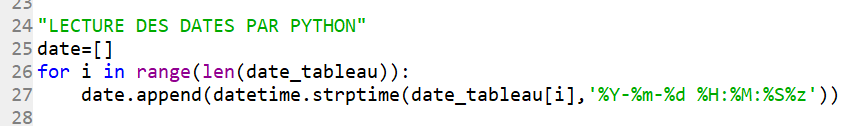
Rapport Python :

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).

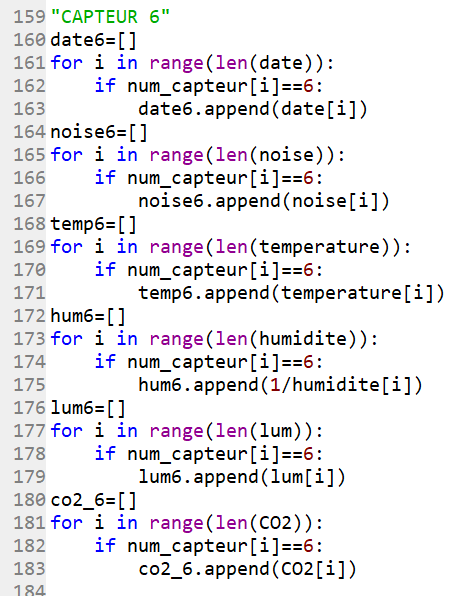
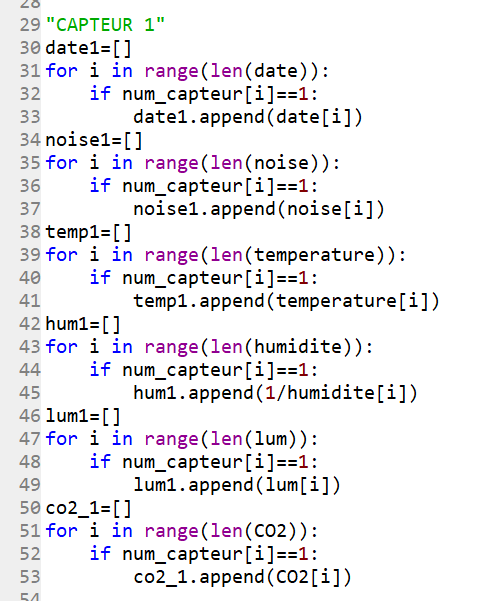


Dans la première partie de l’algorithme, le but était de lire les données CSV. Pour cela, nous avons utilisé la bibliothèque panda en utilisant la fonction « pd.read » et ouvert notre fichier CSV nommé : EIVP\_KM.csv. Chaque colonne du tableau étant séparé d’un « ; » nous avons décidé de créer une séparation 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.

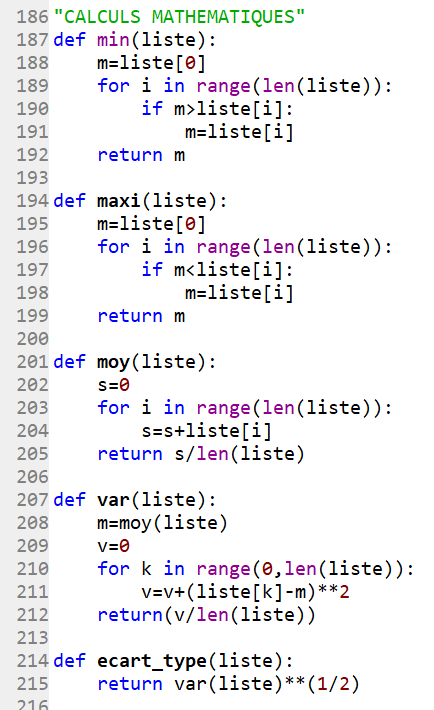
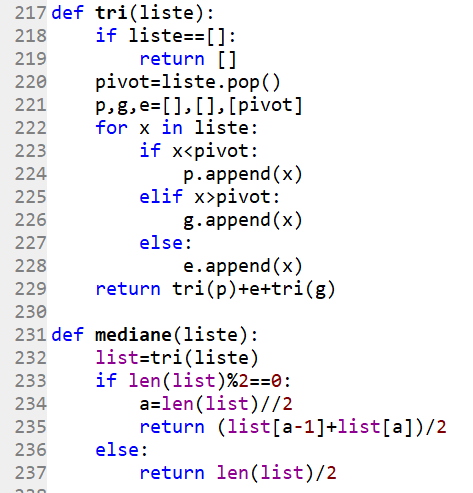
Pour créer notre graphe, il nous fallait alors obtenir des listes composées des différentes valeurs des différentes colonnes. Dans un premier temps, nous avons voulu utiliser la fonction Dataframe de la bibliothèque. Finalement, nous avons opté pour l’utilisation de la fonction tolist, ainsi dans chaque liste se trouve 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’était pas reconnaissable par Python et le deuxième est le fait qu’aucune des listes ne soient séparées par capteur. Ainsi, le graphe créé reprenait les valeurs de chaque capteur et donc était faux.



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

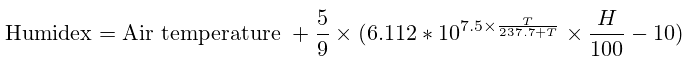


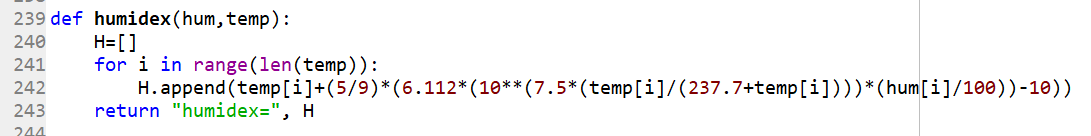
Ensuite, nous avons voulu remédier au fait que dans chaque liste de valeur (température, humidité, …) les résultats de tous les capteurs étaient présents. En effet, lorsque l’on affichait un graphe avec par exemple les dates en abscisse et la température en ordonnée, le graphe n’était alors pas exploitable. Pour cela, nous avons créé de nouvelles listes vides avec le numéro du capteur correspondant à la fin de chacune. Par exemple, ici, dès que le numéro de capteur est 1 dans la liste id, les valeurs associées dans les listes : date, humidite, lum, co2 et temperature sont implémentées dans les nouvelles listes vides respectives créées. Il nous a été obligatoire de le faire pour chaque liste afin que chaque liste possède le même nombre d’argument. Ainsi, chaque capteur possède une liste de donné par informations fournies par ce capteur.

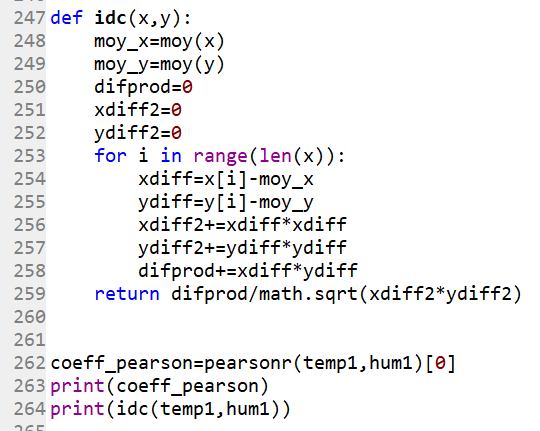


Ensuite, nous avons créer des fonctions pour déterminer les valeurs caractéristiques des différentes listes et ainsi, pouvoir indiquer le maximum de celle-ci sur un graphe. Ainsi, les fonctions créées permettent de calculer dans l’ordre : le minimum, le maximum, la moyenne, la variance et l’écart-type.

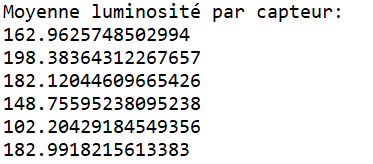
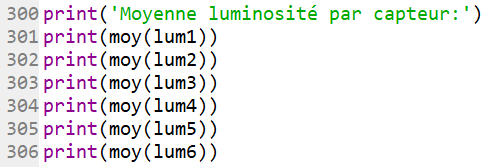
Pour déterminer une médiane d’une liste, il faut que celle-ci soit triée préalablement. Ainsi, nous avons utilisé un algorithme de tri pour pouvoir créer ensuite une fonction médiane valable.

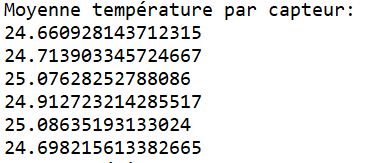
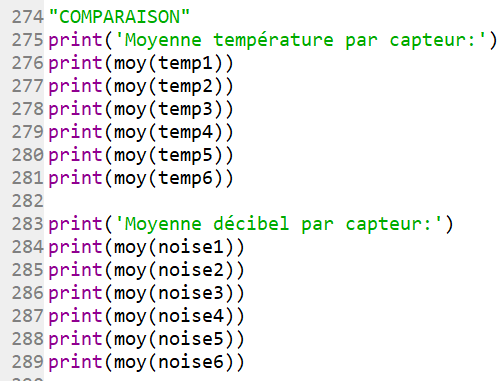
Par la suite, il est demandé de calculer l’indice humidex. Il se calcule à l’aide de cette formule :

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.

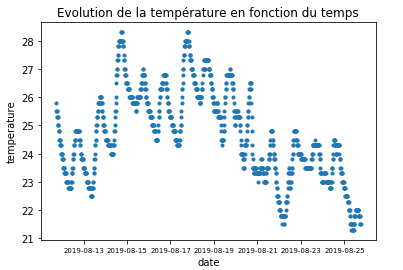
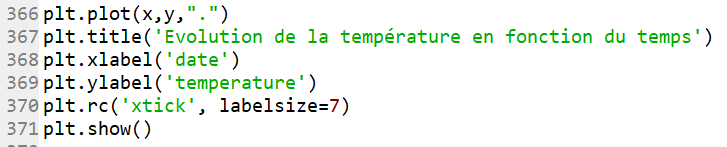
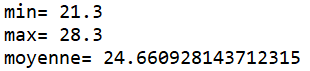
/Pour vérifier les résultats de notre fonction créée, nous avons utilisé la bibliothèque scipy.stats et la fonction correspondant à l’indice de corrélation déjà implémentée dans Python. Les résultats obtenus étaient similaires à ceux fournis par notre fonction, d’où la bonne vérification des résultats.





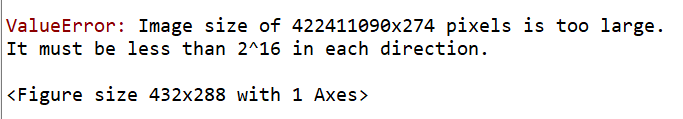
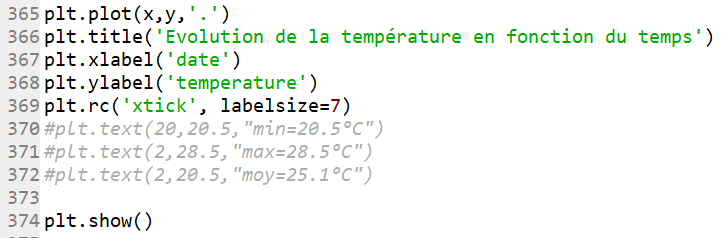
Afin de faire apparaître les anomalies, nous avons voulu comparer les différents capteurs entre eux pour vérifier qu’aucun ne soit défectueux. Ainsi, nous avons calculé différentes moyennes comme celle de la luminosité ou de la température. Il est alors ressorti que le capteur 5 avait tendance à obtenir une moyenne différente des cinq autres, par exemple environ 102 lux (l.280) alors que tous les autres obtiennent une moyenne autour de 150-190 lux. Cependant, la différence n’était pas assez significative pour que l’on puisse en déduire quoi que se soit.

Il était aussi demandé d’afficher des courbes montrant l’évolution d’une variable en fonction du temps. Pour se faire nous avons utilisé la bibliothèque mathplotlib.pyplot (ligne 10).



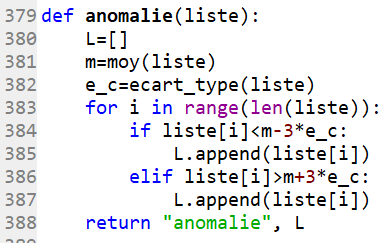
Par exemple, ici, se trouve le graphe de la température en fonction du temps du capteur 1. Nous avons préféré travailler avec des points sur le graphique plutôt que des lignes pour plus de clarté. Le x de la légende représente la liste du temps et le y la liste des différentes températures.

Nous avons ensuite essayé d’afficher les valeurs caractéristiques sur le graphe.



En voulant insérer différentes valeurs comme le minimum, le maximum et la moyenne sur la courbe, par des lignes de codes (ici décommentées), nous nous sommes retrouvés devant une erreur inconnue. Ainsi, il nous a été impossible d’afficher les valeurs caractéristiques sur la courbe.

Enfin, nous avons voulu créer une fonction permettant de déterminer les anomalies des différents capteurs et des différentes données.



Pour se faire nous avons mis dans une liste toutes les données supérieures à la moyenne plus trois fois l’écart type ou inférieures à la moyenne moins trois fois l’écart type. Or, cette fonction nous renvoyait uniquement des listes vides. Ainsi, l’

Utilisation de GitHub :