EXPLICATION PROJET PYTHON :

Consigne pour les examinateurs :

- *Le fichier csv doit s'appeler EIVP_KM.csv.*
- *En première commande saisir seulement :* ***python Projet_Julien_Brian_Groupe_B_V12.py*** *dans la console.*
- *Il est conseillé d'afficher les graphiques en plein écran.*

Lien Github :

https://github.com/BDELAUNAY/ProjetJulienBrian.git

Sommaire :

DELAUNAY Brian
COTISSON Julien

1) Description des différentes versions :

V1 :

- Extraction des données du fichier CSV
- Convertisseur des dates en secondes par rapport au premier temps de chaque capteur
- Affichage avec des courbes des 5 données, avec titre, et axes. (Bruit, Température, Humidité, Luminosité, CO2)

⇒ **Problème 1 : Affichage du 6e capteur absent : seuls les 5 premiers sont affichés**

V2 :

- Tentative de résolution du problème 1 : Trouver autre solution fonctionnelle

V3 :

- Résolution du problème 1 : Le programme comparait un chiffre avec une ligne vide d'où la nécessité, pour la dernière opération d'affecter à la prochaine ligne le chiffre 7, pour que le programme comprenne qu'il a « changé » de capteur et qu'il peut imprimer le dernier capteur (6).

V4 :

- Optimisation : création d'une liste présente toutes les données, et chaînes de caractères destinées à afficher
- Fonction graphique qui affiche les courbes une par une ou toute en même temps suivant un booléen et la moyenne ou non aussi avec un booléen
  - ⇒ **Problème 2 : capteur 5 : ligne droite dans l'affichage des courbes : identifier d'où vient le problème.**
  - ⇒ **Problème 3 : capteur 1 : Décalage par rapport à tous.**

V5 :

- Identification du problème 2 : Le capteur 5 s'arrête et reprend plus tard, d'où le "saut"
- Identification du problème 3 : Le capteur 1 commence après tous les autres
- Résolution problème 2 : Ajout de None pour le capteur 5 à la place des valeurs absentes
- Résolution problème 3 : Ajout de None au début pour le capteur 1 et changement de la date initial du graphique : au lieu de faire démarrer tous les capteurs à leur propre date initiale, ils commencent tous à la date la plus antérieure, alias la première du capteur 5
  - ⇒ **Problème 4 : Optimisation à prévoir en vue du très grand nombre de données, et du grand nombre d'opérations.**

V6 / Statistiques :

- Correction du programme vis-à-vis de la nouvelle feuille de données
- Création de toutes les fonctions statistiques (maximum, minimum, somme, moyenne, écart-type, covariance, variance, étendue)

V7 / Statistiques 2 :

- Création de l'indice de la fonction indice de corrélation
  - ⇒ **Problème 5 : Problème de calcul de la moyenne du au capteur 5 qui s'arrête et manque de valeurs capteur 1 (début) et capteur 4 (fin) : provoque un décalage de la courbe de moyenne.**

V8 :

- Ajout de « None » sur les valeurs manquantes : capteur 1,5 et 4.
- Modification de la fonction moyenne pour y inclure une division en lien avec le nombre de valeur.
- Inclusion dans le programme des fonctions statistiques
- Optimisation du programme en incluant une boucle pour créer la variable k qui représente le nombre de caractères en moins d'une ligne par rapport à la plus grande ligne (52)

V9 :

- Modification de la fonction graph () pour qu'elle utilise : soit la liste complète, soit un intervalle donné.
- Ajout des inputs qui pose les questions relatives à l'étude des données.

V10 :

- Ajout de la fonction occupation : permet de savoir les débuts et fins des périodes occupés sur l'intervalle étudié.
- Changement de côtés des légendes pour une meilleur lisibilité (aucune solution pour la luminosité, c'est le meilleur emplacement qui a été choisi).

V11 :

- Création de la fonction start qui permet de lancer le programme après l'appel du fichier dans PowerShell.
- Création de flèches qui affiche le maximum et le minimum sur les courbes étudiés
- Modification du comptage des valeurs. Le nom d'un booléen était utiliser deux fois sur des fonctions non compatibles.
- Affinage de l'affichage des différentes courbes : placement des flèches maximum et minimum

V12 : VERSION FINAL !!!!

- Optimisation de la taille de la fonction graph () : réduite de moitié
- Affichage du tableau des valeurs statistiques
- Intégration de l'appel nommé « autre »
- Intégration des fonctions d'erreurs relatives à l'entrée des commandes
- Intégration de l'appel nommé « occupation » et création de toutes les fonctions liées à cela afin d'afficher le graphique de l'occupation des bureaux
- Amélioration de l'affichage et des entrées des commandes avec l'ajout d'explications optionnelles en amont du programme
- Test d'absolument toutes les possibilités
- Intégration de l'appel corrélation pour comparer deux variables
- Affichage des tableaux des valeurs statistiques dans l'appel « autre »
- Correction de problèmes d'optimisations récurrent tels que les variables inutiles ou l'appel systématique de la même fonction plutôt que d'enregistrer la fonction dans une variable
- Ajustement des sorties de quelques fonctions telles que maximum() pour avoir aussi les indices correspondants, et la moyenne, variance et écart type pour afficher la version logarithmique si le Bruit est analysé
- Ajustement de l'entrée des fonctions, les variables n'étant pas globale, le programme avait du mal à les retrouver, il a donc fallu les redéfinir toute avant chaque appel, cela ne change quasiment pas la complexité du programme puisque les opérations sont les mêmes qu'avant, c'est juste qu'elle sont faites désormais en amont des fonctions et non plus dedans.

2) Descriptions des commandes :

Chers examinateurs, bienvenue sur le Projet python de Julien et Brian

*** Les commandes à utiliser en premier argument sont les suivantes :
- <display> <variable> (<date_debut> <date_fin>) : Permet d'afficher le graphique d'une variable pour les six capteurs
- <displayStat> <variable> (<date_debut> <date_fin>) : Affiche un tableau des différentes données statistisques d'une variable pour les six capteurs
- <corrélation> <variable1> <variable2> (<date_debut> <date_fin>) : Affiche les deux variables sur un même graphique pour chaque capteur, avec l'indice de corrélation correspondant
- <autre> (<date_debut> <date_fin>) : Permet d'afficher le nombre de capteur que vous souhaitez ainsi que le nombre de variable à traiter, en vous laissant le choix sur les propriétés du graphique,
essayez tout ce que vous souhaitez !
- <occupation> (<date_debut> <date_fin>) : Affiche le graphique représentant les périodes d'occupation du bureau

Attention cependant : les dates sont optionnelles mais les deux bornes sont nécessaires si vous souhaitez les utiliser,
de plus si vous souhaitez afficher les données dans un intervalle de temps qui n'a pas été mesuré par les capteurs, les graphiques seront logiquement blanc, vous devrez alors recommencer avec de nouvelles dates

*** Les différentes variables sont : Bruit, Température, Humidité, Luminosité, CO2, Humidex

*** Si vous rencontrez des problèmes avec les dates sachez qu'ils faut les écrire sous format : AAAA-MM-JJ
Ces dates s'étendent du 2019-08-11 au 2019-08-25

3) Description des variables utilisées et des fonctions dans l'ordre d'apparition dans le programme :

Les variables et listes :

- sys.argv : liste créée par l'utilisateur au moment de l'envoie de la commande.
- nb_cap : nombre de capteurs étudiés.
- nb_donnee : nombre de données étudiées.
- liste_etude_cap : liste de 1 à 6 éléments correspondants aux id des capteurs (exemple 0 pour le capteur 1, 1 pour le capteur 2 …).
- liste_etude_donnee : liste de 1 à 6 éléments correspondants aux id des données.
- cap_date : liste de toutes les dates d'un capteur choisi.
- cap_date_inter : liste de toutes les dates d'un capteur choisi sur l'intervalle choisi.
- date1,date2 : respectivement, dates de début et de fin de l'intervalle si il est demandé.
- id_donnee : id d'une donnee (de 0 à 5).
- id_cap ou cap : id d'un capteur (de 0 à 5).

Les booléens :

- bol_info : Définit si l'on affiche les instructions du programme.
- bol_intervalle : Définit si l'on travaille sur un intervalle.
- bol_log : Définit si la donnée utilisée est le Bruit ou pas, et donc s'il faut utiliser les statistiques en versions logarithmiques ou non.
- bol_corr : Définit si l'on affiche l'indice de corrélation.
- bol_init_moy : Définit si l'on travaille avec la courbe moyenne.
- bol_stats : Définit si l'on affiche les valeurs statistiques.
- bol_diff : Définit si l'on affiche les courbes dans des graphiques séparés.
- bol_tps : Définit si l'on affiche en date ou en Demi-journées la liste des temps.
- bol_enregistrement : Définit si l'on met en attente un capteur le temps de rattraper les trous causée par l'absence de données (pour ne pas créer un décalage entre les capteurs).

Les fonctions :

- **Start :** Cette fonction ne prend pas d'argument car elle exécute les commandes fournies par l'utilisateurs, que sont <display>, <displayStat>, <corrélation>, <autre>, <occupation>, dont la description figure ci-dessus.
  Elle travaille avec une liste choisie parmi deux : la liste entière ou la liste des données de l'intervalle si cette dernière option est choisie.
  Il y a aussi la possibilité pour l'utilisateur de ne rien entrer en argument pour afficher les instructions possibles du programme. Ce à quoi nous l'invitons à faire pour se laisser guider.

- **indice variable :** Permet de renvoyer le numéro lié à la variable sous forme de chaîne de caractère en entrée

- **Les fonctions d'intervalles :**
  *** **indices_intervalle :** Permet d'identifier les indices de début et de fin de l'intervalle demandé entre les dates date1 et date2
  *** **liste_intervalle :** Permet de créer une liste avec uniquement les données correspondant à l'intervalle demandé entre les dates date1 et date2
  *** **date_intervalle :** Permet de créer une liste de toutes les dates correspondantes à l'intervalle demandé entre les dates date1 et date2.

- **Les fonctions liées aux tris**
  *** **fusion et trifusion :** Permettent de ranger les éléments d'une liste dans l'ordre croissant en utilisant la méthode du tri fusion
  *** **pop_None :** Permet de retirer les éléments de type None d'une liste.
  *** **tri :** Permet de retirer les éléments de type None d'une liste et de la trier par ordre croissant, elle utilise les trois fonctions précédentes. Elle renvoie une liste triée par ordre croissant.

- **Les fonctions de conversion (de chaîne de caractère et temporelle)**

*** **saut_first_col :** Permet de sauter la première colonne d'une ligne en csv, elle a été implémentée après l'arrivée de la seconde version du fichier csv, pour se ramener au premier cas. Elle renvoie donc une chaîne de caractère.
*** **dissociation_date :** Renvoie une date sous forme 'AAAA-MM-JJ HH:MM:SS' sous forme d'une liste d'entier : [AAAA,MM,JJ]
*** **convertisseur_HMS :** Permet de convertir en seconde écoulée dans la journée une heure sous format 'AAAA-MM-JJ HH:MM:SS', donc sans se soucier de l'année, du mois ou du jour.
*** **distance_temporelle :** Permet de renvoyer la distance temporelle en seconde d'une date par rapport à la date initial fixée, à condition que ces dates se situent la même année.

- **Les fonctions d'analyse statistiques :**
*** **maximum :** Permet de renvoyer le maximum d'une liste de nombre, ainsi que son indice associé dans la liste.
*** **minimum :** Permet de renvoyer le minimum d'une liste de nombre, ainsi que son indice associé dans la liste.
*** **etendue :** Permet de renvoyer l'étendue d'une liste de nombre.
*** **mediane :** Permet de renvoyer la valeur médiane d'une liste de nombre.
*** **count_None :** Permet de renvoyer le nombre de None dans une liste, pour ne pas les prendre en compte du calcul de la moyenne ou de la variance.
*** **somme :** Permet, suivant les booléens utilisés, de renvoyer la somme des valeurs d'une liste ou la somme des valeurs au carré d'une liste, et si la liste concerne des valeurs en décibels (bol_log = True), elle renvoie aussi ces sommes, avec chaque nombre convertis au préalable. (Disjonction nécessaire pour la moyenne logarithmique du Bruit)
*** **moyenne :** Permet de renvoyer la valeur moyenne d'une liste, et si la liste concerne des valeurs en décibels (bol_log = True), elle renvoie la moyenne logarithmique.
*** **variance :** Permet de renvoyer la variance d'une liste, , et si la liste concerne des valeurs en décibels (bol_log = True), elle fait les ajustements nécessaires.
*** **ecart_type :** Permet de renvoyer l'écart-type d'une liste, , et si la liste concerne des valeurs en décibels (bol_log = True), elle fait les ajustements nécessaires.
*** **covariance :** Permet de calculer la covariance de deux listes, si une des listes concerne des valeurs en décibels (bol_log = True) (bol_emplacement désignant son emplacement : L1 ou L2).
*** **indice_correlation :** Permet de calculer l'indice de corrélation entre deux listes, si une des listes concerne des valeurs en décibels (bol_log = True) (bol_emplacement désignant son emplacement : L1 ou L2).

- **Les fonctions humidex :**
*** **alpha :** Permet de calculer le coefficient alpha utilisé dans le calcul de l'indice humidex.
*** **humidex :** Permet de calculer l'indice humidex à partir d'une température, d'une humidité relative et du coefficient alpha.

- **Les fonctions graphiques :**
*** **fusion_str :** Créer une chaine de caractère à partir d'un mot et d'une valeur (nécessaire pour l'affichage du maximum par exemple).
*** **titre :** Créer un titre en fonction du nombre de capteur étudié, tous (bol_tous = True), ou bien soit un seul ou la moyenne des capteurs, ainsi que de la donnée étudiée.
*** **lab :** Renvoie le label associé à la donnée demandée, en l'occurrence, l'unité.
*** **occupation :** Permet de créer la liste représentant l'occupation du bureau (1 : occupé, 0 : inoccupé), en fonction du temps sous deux formes possibles : en dates (bol_tps = True) ou en demi-journées, sachant que la 2^{e} option est plus rapide.

***** graph :** Surement la deuxième fonction la plus importante de ce programme avec la fonction start, elle permet d'afficher les graphiques des données choisis, des capteurs choisis, de l'intervalle de temps choisis (bol_intervalle = True) sur toutes les données (bol_intervalle = False), sur des graphiques différents (bol_diff = True) ou non (bol_diff = False), avec l'apparition de la courbe moyenne (bol_init_moy = True), ou non (bol_init_moy = False), avec l'indice de corrélation si deux capteurs sont demandés (bol_corr = True), ou non (bol_corr = False).
***** graph_corr :** Similaire à graph, elle renvoie le graphique de corrélation entre deux données et non pas deux capteurs.

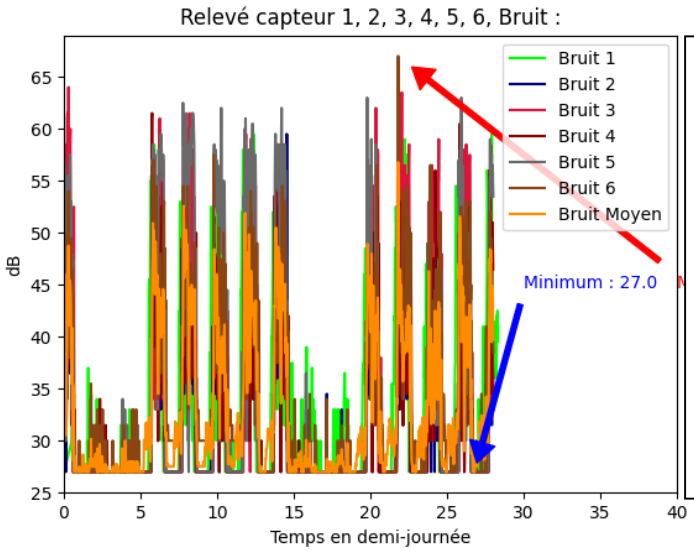
- **Fonction d'erreurs :**
  ***** erreur_date :** Vérifie si les dates d'intervalle d'étude données par l'utilisateur sont correctes, elle renvoie une erreur sinon avec les précisions.

4) Enregistrement des données :

- La partie with open() permet d'extraire les données du fichier EIVP_KM.csv, et de créer liste_cap, la liste comprenant toutes les données de tous les capteurs.
- La partie ajustement des données permet de rajouter des None dans les trous de certains capteurs pour que tous les capteurs aient la même taille, en effet, le capteur 1 commence en retard, le capteur 4 commence légèrement après les autres et fin légèrement avant, d'où l'absence d'une valeur, enfin le capteur 5 s'arrête de fonction pendant environ 1 jour et demi.
- La partie création de la courbe moyenne, comme son nom l'indique, crée la courbe moyenne des 6 capteurs.

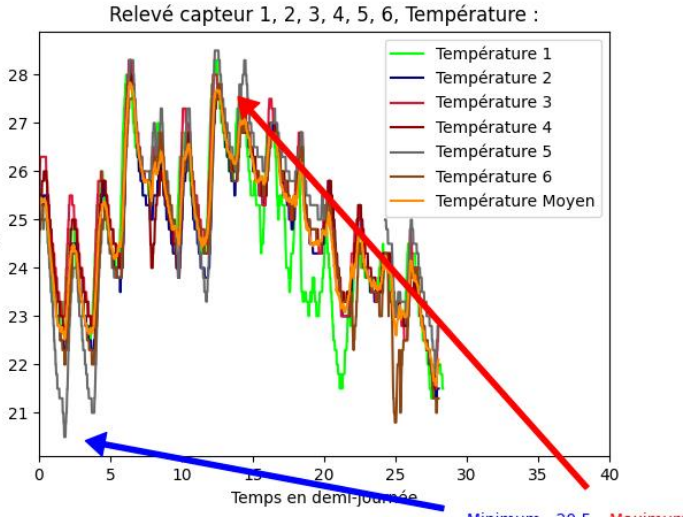
DELAUNAY Brian
COTISSON Julien

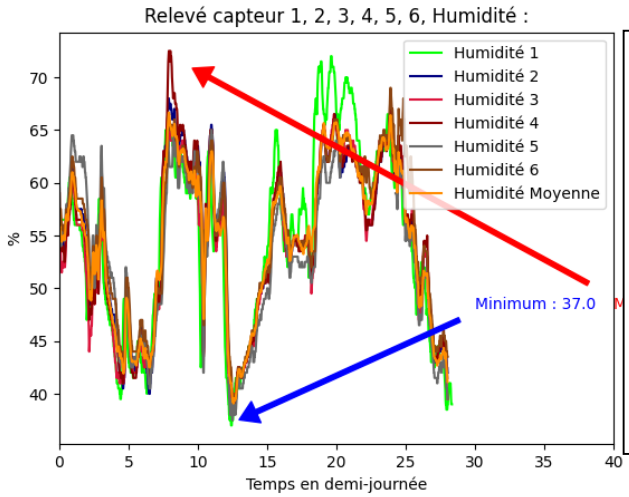
5) Compte rendu des similarités :



Les pics d'activité son sensiblement similaires et sur les périodes de temps.
On observe des baisses significatives de bruit qui correspondent au bruit ambiant, quand celui-ci passe en dessous de 35dB on en déduit que les locaux sont vides.
Aucun pic n'est significatif par rapport aux autres, ce qui reflète la similarité des capteurs.

On observe des extremums locaux importants vis-à-vis du capteur 1 et 5 et le capteur 6 au niveau de la 25ème demi-journée.
Hormis ces différences, les capteurs présentent une forte similarité, car la plupart sont proches de la courbe moyenne.
Les différences peuvent s'expliquer par un courant d'air ou par le fait que les capteurs soient proches d'une ouverture.

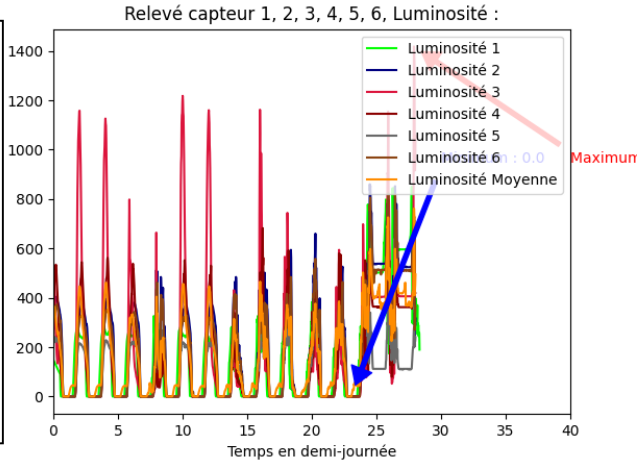


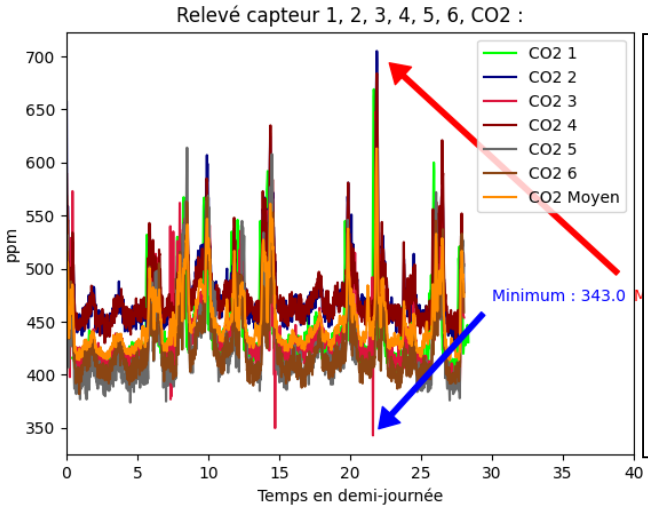


On observe une très forte similarité des capteurs. Mis à part 2 pics locaux des capteurs 1 et 4.
Les courbes sont proches de la moyenne et donc sont similaires.

DELAUNAY Brian
COTISSON Julien

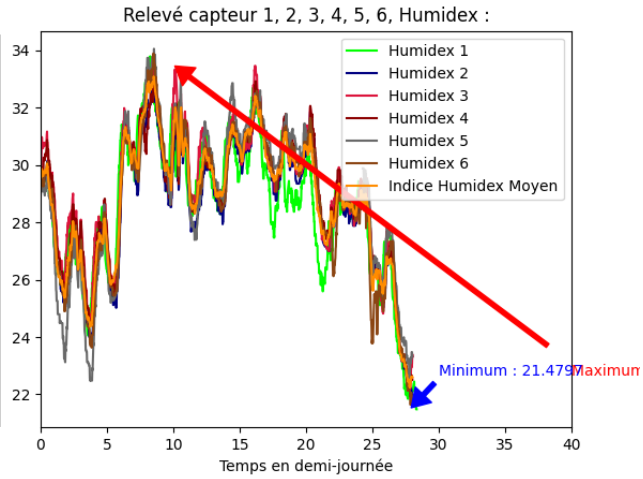
Le capteur 3 est différent des autres. Les pics atteints sont largement au-dessus de la moyenne, cependant, ils se produisent en même temps que les autres capteurs.
On peut en déduire un placement différent, notamment sous une lumière.
Tous les capteurs ont leurs valeurs qui grimpent en flèche à partir de la 25 demi-journée, on ne peut déduire pourquoi ils réagissent de cette manière.





Les capteurs 2,3 et 4 suivent la même courbe et sont légèrement au-dessus de la moyenne. Cependant on observe une similarité des capteurs 1, 5 et 6.
Les façons d'expliquer ces différences sont peut-être dues à un placement différent.

Les capteurs sont semblables, on observe une grande corrélation des mesures effectuées.



Lorsque l'on demande au programme des indices de corrélation, via la commande « autre », on se rend compte que des courbes sont très similaires notamment avec des indices de corrélation supérieurs à 0.95 sur la majorité des valeurs (exemple : 3 et 4 et 6 etc…).
Quelques-uns sont fortement différents au niveau de la corrélation avec des valeurs bien inférieures à 0.95.
L'indice de corrélation entre 2 variables, n'est pas un indicateur fiable du fait de la trop grande différence des variables (exemple : les décibels sont une fonction logarithmique…).
Observation : le capteur présente un arrêt de fonctionnement durant quelques demi-journées.

Bruit :

Numéro du capteur	Moyenne	Variance	Ecart-type	Minimum	Médiane	Maximum	Etendue
1	38.9577	-1429.3421	37.8066	27.0	30.0	59	32.0
2	33.4681	-1045.2399	32.3301	27.0	27.0	59	32.0
3	41.01	-1586.9134	39.836	27.0	27.0	64	37.0
4	36.1101	-1219.9834	34.9282	27.0	27.0	61	34.0
5	43.5787	-1799.8188	42.4242	27.0	27.0	63	36.0
6	37.1771	-1297.4272	36.0198	27.0	30.0	67	40.0

Température :

Numéro du capteur	Moyenne	Variance	Ecart-type	Minimum	Médiane	Maximum	Etendue
1	24.6609	2.5488	1.5964	21.3	24.5	28	6.7
2	24.7139	2.023	1.4223	21.3	24.8	28	6.7
3	25.0762	2.1623	1.4704	21.5	25.0	28	6.5
4	24.9127	1.87	1.3674	21.5	24.8	28	6.5
5	25.0863	3.2434	1.8009	20.5	25.3	28	7.5
6	24.6982	2.39	1.5459	20.8	24.8	27	6.1999

Humidité :

Numéro du capteur	Moyenne	Variance	Ecart-type	Minimum	Médiane	Maximum	Etendue
1	54.4434	77.3385	8.7942	37.0	55.5	72	35.0
2	54.5319	57.3367	7.5721	38.5	55.5	68	29.5
3	54.0598	58.7449	7.6645	39.0	54.5	67	28.0
4	54.6417	59.9332	7.7416	39.0	55.0	72	33.0
5	52.7978	54.4375	7.3781	37.5	53.0	65	27.5
6	55.3613	53.3714	7.3055	39.5	56.0	69	29.5

Luminosité :

Numéro du capteur	Moyenne	Variance	Ecart-type	Minimum	Médiane	Maximum	Etendue
1	162.9625	36825.472	191.8996	0.0	104.0	850	850.0
2	198.3836	44308.9137	210.4968	0.0	156.0	860	860.0
3	182.1204	74662.3576	273.2441	0.0	48.0	1418	1418.0
4	148.7559	32194.3191	179.4277	0.0	51.0	692	692.0
5	102.2042	10009.7384	100.0486	0.0	110.0	366	366.0
6	182.9918	38768.7996	196.8979	0.0	144.0	824	824.0

CO2 :

Numéro du capteur	Moyenne	Variance	Ecart-type	Minimum	Médiane	Maximum	Etendue
1	441.8914	1365.623	36.9543	390.0	430.0	669	279.0
2	476.6193	1097.3744	33.1266	429.0	467.0	705	276.0
3	442.6921	1564.083	39.5484	343.0	430.0	671	328.0
4	477.8943	1047.6585	32.3675	430.0	467.5	684	254.0
5	426.4669	1648.3488	40.5998	374.0	412.0	614	240.0
6	426.8899	1074.7166	32.7828	382.0	415.0	595	213.0

Humidex :

Numéro du capteur	Moyenne	Variance	Ecart-type	Minimum	Médiane	Maximum	Etendue
1	28.5257	5.9754	2.4444	21.4797	28.7778	33	11.5202
2	28.6369	5.2334	2.2876	21.9094	29.0077	33	11.0905
3	29.1216	5.3896	2.3215	22.1964	29.4361	33	10.8036
4	28.9699	5.0588	2.2491	22.4119	29.3933	32	9.5881
5	28.9215	7.6341	2.7629	22.481	29.6481	34	11.5189
6	28.7618	5.9568	2.4406	21.6438	29.1803	33	11.3562

Sur ce tableau des valeurs statistiques, les données de la température, du CO2, de l'humidité et humidex, sont toutes proches les unes aux autres pour tous les capteurs (exemple : température moyenne autour de 24-25°C, médiane humidex proche de 29 etc…).

Pour de plus amples détails sur les similarités des capteurs, amusez-vous avec les différentes possibilités de l'algorithme.

6) Retour d'expérience

Pour ce qui est de github, cet outil nous a frustré au départ car nous ne savions pas l'utiliser ni quelle était son utilité. Mais après avoir découvert Github Desktop qui est très facile à prendre en main, nous avons finalement pu nous échanger régulièrement nos travaux et tenir un regard sur le travail de chacun, tout en évoluant nos différentes versions, plus qu'un seul drive, github nous a permis une très belle organisation, et est si pratique pour apporter des modifications en quelques clics. C'est un logiciel que nous réutiliserons pour sûr.

Les fonctions implémentées sont toutes optimisées pour faire le moins d'opérations possibles, que cela soit pour créer des boucles dans des listes de plusieurs listes pour choisir celle voulue, ou utiliser des booléens pour créer tout un cheminement dans les fonctions. Aussi, le stockage et l'écrasement des données inutiles était primordiale pour ne pas surcharger la mémoire des ordinateurs, d'où l'écrasement des listes bruits, humidity, temp, … et de bien d'autres variables à l'intérieur des fonctions.

Notre but était de créer une seule fonction qui réalise tous les affichages de courbes (sauf pour la corrélation entre deux données qui était incompatible avec la première fonction graph), ainsi grâce aux booléens, les fonctions s'exécutent sans réaliser d'actions inutiles. Aussi leur construction en chaîne de règles avec les booléens rendait leur modification très simple, comme c'est le cas des fonctions moyennes, variances.. auxquelles l'on a rajouté la possibilité du calcul des valeurs logarithmiques aisément.
L'algorithme s'est articulé autour de la fonction graph, chaque fonction s'y ramène, que cela soit dans le traitement des données, ou l'affichage de texte ou de statistiques. Ainsi nous avons d'abord créé une esquisse de cette fonction et au fur et à mesure de la création des autres fonctions, nous avons rajouté des instructions. Pour finir vient la fonction start qui se charge d'orienter le programme grâce aux commandes de l'utilisateur.
Chaque fois qui nous trouvions des choses devenues inutiles, nous faisions en sorte de les supprimer ou de les contourner. Et cela nous donne un programme qui s'exécute en un rien de temps, pour réaliser un tas d'action.
Seul bémol, l'affichage de l'occupation avec les dates était obligatoire. Nous avions testé l'affichage avec des dates au tout début, mais nous nous sommes vite rendu compte que cela faisais beaucoup plus de choses à afficher pour le programme, d'où notre choix de travailler en distance temporelle en seconde par rapport à la date la plus ancienne, ce qui a aussi facilité notre démarche pour la recherche des intervalles.

Merci de nous avoir lu, à bientôt sur Python !

Julien&Brian