Lestang Emma Promo 62

Elfassi Charlène

# **Compte rendu Projet Algorithme et programmation Python**

Utilisation de GIT HUB :

Il s’agit d’un service d’hébergement de codes sources qui permet un contrôle d'accès pour un projet partagé (entre plusieurs utilisateurs). Par ailleurs, il propose des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités, la gestion de tâches et un wiki (page web) pour chaque projet. Il s’apparente à une forme de réseau social avec ses flux, sa possibilité de suivre des personnes ou des projets ainsi que le fait de pouvoir créer des graphes sur ses dépôts (repository).

Il a été utile et efficace de travailler avec Git Hub puisque cela nous a permis d’échanger avec facilité et fluidité nos programmes. On a pu se répartir les différentes parties du code et ainsi compléter au fur et à mesure l’ensemble du code.

Pour l’installation de Git Hub, on a dû procéder avec le terminal de nos ordinateurs respectifs (Macintosh). Après s’être inscrites sur Git Hub, on a dû copier le lien de notre identifiant dans le terminal de notre ordinateur et suivre les étapes en répondant aux demandes du terminal.

Après l’installation complète, nous avons souhaité vérifier le bon fonctionnement de cette plateforme en créant un fichier commun pour pouvoir se partager le code.

Nous avons ensuite utilisé les commandes classiques de git, afin de synchroniser nos fichiers sur le dépôt distant Git Hub :

* “git add” pour ajouter nos fichiers à l’index
* “git clone” pour récupérer les derniers fichiers sur le serveur distant (dans notre cas GitHub)
* “git commit -m” pour valider les modifications sur les fichiers, tout en leur donnant un nom
* “git push” pour envoyer des fichiers sur Git Hub
* “git remote” pour se connecter au dépôt distant sur Git Hub

Programmation python :

Nous avons choisi d’importer la bibliothèque Pandas, puisqu’elle permet de manipuler facilement des données à analyser.

Tout d’abord, nous avons essayé de manipuler les tableaux de données avec la bibliothèque Pandas en utilisant la méthode des Data Frame. Les colonnes sont des variables et les lignes des individus par exemple. On peut alors facilement lire et écrire ces types de tableaux à partir ou vers un fichier tabulé et par la suite tracer des graphiques à l’aide de Matplotlib.

Nous avons donc converti le fichier en Data Frame, puis nous avions réussi à l’aide de MatPlotLib à tracer les courbes demandées. Cependant, cette méthode nous étant auparavant inconnue, nous n’avons pas réussi à l’utiliser pour la suite du sujet, comme pour l’utilisation de fonctions tel que la moyenne ou encore l’écart type.

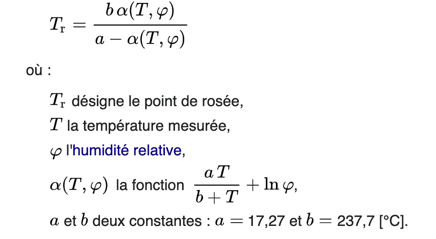
Finalement, nous avons décidé de convertir chaque colonne du fichier csv en liste, toujours en utilisant le module Pandas. Ainsi nous pouvions travailler sur un format qui nous est familier.

Pour répondre aux attentes du sujet, nous avons créé et utiliser plusieurs programmes :

* “conv()” : Ce programme permet de convertir le fichier csv choisi en liste. En effet, chaque colonne est nommée et devient une liste.
* “tri(l\_n,l\_t,l\_h,l\_l,l\_c,l\_date)” : Cette fonction prend en argument les listes créées dans la fonction conv(). On utilise la méthode du Bubble sort. On compare plusieurs fois les éléments consécutifs d’une liste et on les permute lorsqu’ils sont mal triés.
* “choixdonnée()” : Cette fonction permet de choisir quelle donnée tracer en fonction du temps lors de la construction des courbes.
* “choixcapteur()” : Cette fonction permet de choisir avec quel capteur on veut travailler.
* “listeparcapteur()” : Cette fonction permet de créer les listes des données demandées mesurées par le capteur demandé.
* “graph()” : Cette fonction permet de tracer les données en fonction du temps et du capteur choisi. On utilise le module matplotlib.
* “minimum(liste)”, “maximum(liste)”, “moyenne(liste)”, ”ecart\_type(liste)” : Ces fonctions ont toutes pour paramètre une liste. Pour la réalisation du projet, on utilise les listes créées par la fonction listeparcapteur(). On utilise pour chaque liste une boucle for, ce qu’on a l’habitude d’effectuer.
* “tri\_croiss(list)”, “mediane(list)” : Pour pouvoir calculer la médiane de valeur d’une liste, nous avons introduit la fonction tri\_croiss qui permet de trier chaque liste dans l’ordre croissant. Puis nous calculons la médiane, avec la fonction mediane, tout en prenant en compte la parité de la longueur de la liste.
* “rosee()”, “humidex()”, “humidexbis()” : Afin de calculer l’indice humidex, nous avons cherché sur internet une définition et avons choisi d’utiliser la formule de Wikipedia suivante :



Puis pour calculer Trosee, l’indice de rosée, nous avons utilisé la formule suivante :



Pour cela, nous avons créé la fonction rosee() qui nous donne la liste de valeur de l’indice de rosée en fonction de la température. Puis nous avons créé la fonction humidex, pour obtenir la liste des indices humidex en fonction de la température. Malheureusement, ce programme est trop complexe et le temps d’attente de résultat est trop long. Nous avons donc créé la fonction humidexbis qui prend comme argument la température et l’indice de rosée et qui, en sortit, renvoie l’indice humidex correspondant.

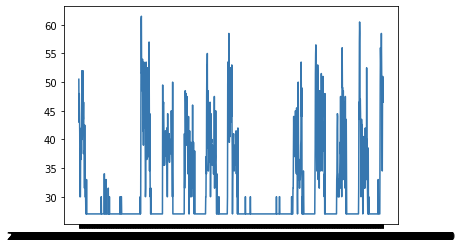
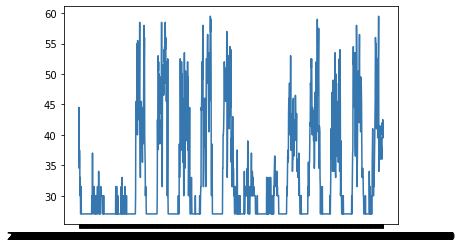
* “indcorrelation()” : Cette fonction prend pour paramètre deux listes. Nous avons effectué une boucle for pour exprimer la formule tout en réutilisant les fonctions moyenne et ecart-type. La fonction renvoie une liste de tous les indices de corrélation.

Nous avons remarqué que la plupart des programmes mettent un certain temps à s’exécuter. On peut l’expliquer par leur complexité. En effet, la fonction tri, nécessaire à l’obtention des courbes à une complexité de O(l!) avec l la longueur de la liste. Pour décomplexer les algorithmes, nous avions pensé au fait de procéder en deux temps. D'abord, d’ouvrir et de lire le fichier csv afin de convertir le tableur en tableau. Puis, dans un second temps, de récupérer les éléments des listes du tableau. C’est cette dernière étape que nous n’avons pas réussi à faire, notamment à cause du manque de temps. C’est pourquoi nous avons conservé notre code.

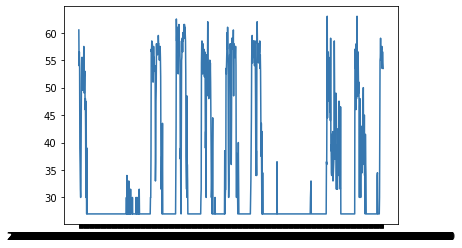
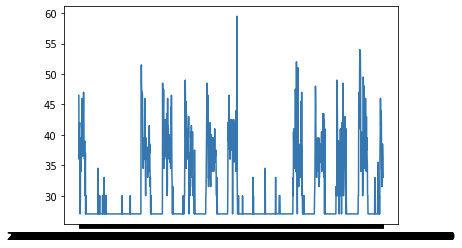
Réponse au sujet :

Nous avons tracé les courbes avec les données que chaque capteur a récupéré, afin de les comparer.

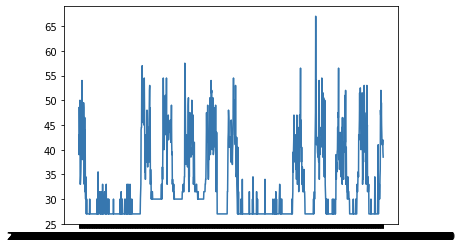
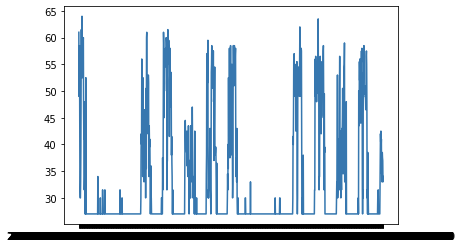
Par exemple, voici les courbes du bruit en fonction du temps mesuré par les 6 capteurs :



Capteur 1 Capteur 4



Capteur 2 Capteur 5



Capteur 3 Capteur 6

On remarque que les courbes sont très similaires mais que certains capteurs sont plus sensibles que d’autres. En effet, le capteur 2 semble moins sensible que les autres.

Afin de comparer les courbes, nous avons créé la fonction comparaisonliste qui prend en argument les listes qui correspondent aux données récupérées par un capteur et qui permet, en prenant un intervalle de confiance de 5%, de mesurer la similarité des données. La fonction renvoie le nombre de données similaires donc plus ce nombre est élevé, plus les capteurs sont similaires.