DS n° 02 – DS02

- Faire tous les exercices dans un même fichier NomPrenom.py à sauvegarder,
- Mettre en commentaire l'exercice et la question traités (ex : # Exercice 1),
- Ne pas oublier pas de commenter ce qui est fait dans votre code (ex : # Je crée une fonction pour calculer la racine d'un nombre),
- Il est possible de demander un déblocage pour certaines questions, mais celle-ci seront notées 0,
- Il faut vérifier avant de partir que le code peut s'exécuter et qu'il affiche les résultats que vous attendez. Les lignes de code qui doivent s'exécuter sont décommentées.

Introduction

Le but de ce travail est d'analyser un jeu de données des comptages horaires de vélos par compteur et localisation des sites de comptage. On se limitera pour des raisons de taille de fichier à la date du 11/11/2022.

Un compteur est un système placé sous la route comme le montre la figure 1 qui détecte le passage des vélos.

FIGURE 1 – Compteur installé sur la piste cyclable en bordure de l'avenue Daumesnil (75012).

Il peut être équipé d'un compteur dans le cas d'un aménagement cyclable unidirectionnel ou de deux compteurs dans le cas d'un aménagement cyclable bi-directionnel.

La Ville de Paris déploie depuis plusieurs années des compteurs vélo permanents pour évaluer le développement de la pratique cycliste.

Les compteurs sont situés sur des pistes cyclables et dans certains couloirs bus ouverts aux vélos. Les autres véhicules (ex : trottinettes,...) ne sont pas comptés.

Vous retrouverez ici :

- Identifiant du compteur,

— Nom du compteur,
— Identifiant du site de comptage,
— Nom du site de comptage,
— Comptage horaire,
— Date et heure de comptage,
— La date d'installation du compteur,
— Les coordonnées GPS du compteur,
— La référence du compteur.

Les données sont disponibles sur la page suivante :

https://www.data.gouv.fr/fr/datasets/comptage-velo-donnees-compteurs/

1 Analyse des données de comptage des passage de vélos dans les rues de Paris

1.1 Lecture du fichier de données

Le fichier `comptage-velo-donnees-compteurs_light.csv` présent dans le dossier « /home/eleves/Ressources/PTSI/ » recense les données de comptage de l'ensemble des compteurs de la ville de Paris.

Pour cela nous allons dans un premier temps lire ce fichier en utilisant le code suivant.

Question 1 **Écrire** et **exécuter** un script permettant de lire le fichier de données et de découper son contenu sous la forme d'une liste `lignes` dont chaque élément correspondra à une ligne du contenu sous la forme d'une chaîne de caractères (string).

Remarque : Cette réponse étant nécessaire pour traiter la suite, il est autorisé de demander la correction à votre enseignant. La note à cette question sera alors de 0.

Ainsi, `ligne[0]` renvoi :

```
id_compteur;nom_compteur;id;name;sum_counts;date;installation_date;coordinates;counter;
```

Et `ligne[1]` renvoi :

```
100003096-353242251;97 avenue Denfert Rochereau SO-NE;100003096;97 avenue Denfert
 Rochereau;6.0;2022-11-11T00:00:00+01:00;2012-02-22;48.83504,2.33314;X2H20012081;
```

Cela signifie que le compteur dont l'`id` est `100003096-353242251`, situé au `97 avenue Denfert Roche` qui capte les vélos circulant dans le sens `SO-NE` (sud-ouest vers nord-est), dont les coordonnées GPS sont `48.83504,2.33314` (latitude, longitude) a vu passer `6.0` vélos le `2022-11-11` (11 novembre 2022) à `T00:00:00+01:00` (entre 0h et 1h).

Solution 1

```
file_in=open('comptage-velo-donnees-compteurs_light.csv','r')
contenu=file_in.read()
file_in.close()
lignes=contenu.split('\n')
```

1.2 Pic de circulation

On souhaite savoir à quel moment de la journée ont lieu le plus de trajets en vélo. Pour cela, nous allons sommer pour chaque créneau horaire les données de l'ensemble des compteurs de la ville et enregistrer ces valeurs dans une liste `creneaux`.

Question 2 **Recopier** et **compléter** le code suivant afin de générer la liste `creneaux`.

```
creneaux=[0]*24

for ligne in lignes[1:-1]:
    data=ligne.split(';')
    creneaux[int(data[5][__:__])]+=float(_______)
print(creneaux)
```

Ainsi, `creneaux[0]` donne `5631.0`, ce qui signifie qu'il y a 5631 passages de vélos sur les compteurs de toute la ville de Paris entre 0h et 1h du matin le 11/11/2022.

Solution 2

```
creneaux=[0]*24

for ligne in lignes[1:-1]:
    data=ligne.split(';')
    creneaux[int(data[5][11:13])]+=float(data[4])
print(creneaux)
```

Le code suivant permet de tracer les valeurs de `creneaux` en fonction de l'heure.

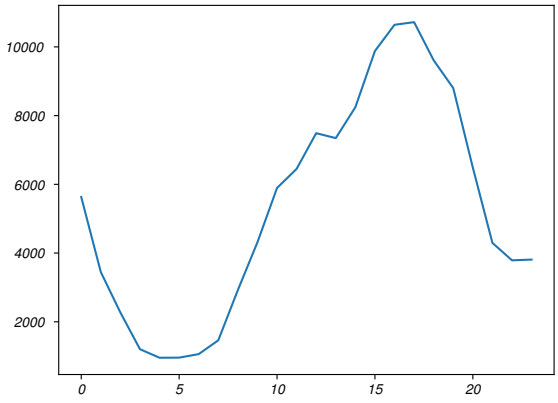
```
plt.plot(creneaux)
plt.show()
```

Question 3 Grâce à cette commande, **déterminer** et **afficher** à l'aide d'un commentaire dans un `print` l'heure à laquelle il y a eu un maximum de passages en vélo.

Solution 3

```
plt.plot(creneaux)
plt.show()
print('Le max est de 10720 à 17h.')
```

```
[5631.0, 3447.0, 2270.0, 1202.0, 954.0, 957.0, 1060.0, 1461.0, 2925.0, 4315.0, 5894.0
 6450.0, 7489.0, 7345.0, 8244.0, 9880.0, 10643.0, 10720.0, 9610.0, 8801.0, 6487.0, 4297.
 3789.0, 3810.0]
```


10000
8000
6000
4000
2000
0
5
10
15
20

1.3 Créer la liste des compteurs

Le code suivant permet de créer une liste `compteurs` qui contient les données des compteurs qui nous intéressent, c'est à dire :

— l'identifiant,
— le nom,
— les coordonnées GPS,
— le nombre de passage pour la journée (il sera pour le moment initialisé à 0).

```
compteurs=[]
for ligne in lignes[1:-1]:
    data=ligne.split(';')
    data_compteur=[data[0],data[1],[float(d) for d in data[7].split(',')],0]
    if data_compteur not in compteurs:
        compteurs.append(data_compteur)
```

Question 4 **Écrire** le code permettant de déterminer et d'**afficher** le nombre de compteurs de la ville de Paris (un compteur bi-directionnel sera compté comme 2 compteurs). Valider votre code en vérifiant qu'il y en a plus de 90.

Solution 4

```
print(len(compteurs))
```

Il y en a 2185.

On souhaite maintenant modifier la valeur du nombre de passage pour chaque compteur de la liste `compteurs` (pour le moment égale à 0).

Pour cela, il est conseillé :

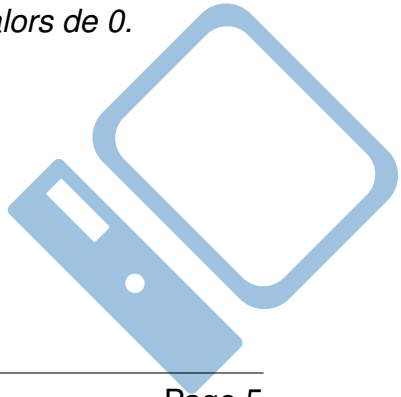
1 De parcourir la liste `lignes`,
2 De découper chaque ligne à l'aide du séparateur ';',
3 D'identifier pour chaque ligne, l'`id` du compteur et de chercher le compteur correspondant dans la liste `compteurs`,
4 D'ajouter à la valeur du nombre de passage de chaque compteur de la liste `compteurs`, la valeur du nombre de passage correspondant à la ligne de `lignes`.

Question 5 **Écrire** le code correspondant à ce pseudo code afin de modifier la liste `compteurs` en conséquence.

Remarque : Cette réponse étant nécessaire pour traiter la suite, il est autorisé de demander la correction à votre enseignant. La note à cette question sera alors de 0.

Solution 5

```
for ligne in lignes[1:-1]:
    data=ligne.split(';')
    for compteur in compteurs:
        if compteur[0]==data[0]:
            compteur[3]+=float(data[4])
```



2 Calcul de la distance entre deux compteurs

On donne la fonction suivante qui permet de déterminer la distance (attention celle-ci n'est pas dans une unité SI) entre deux lieux dont les positions relatives sont caractérisées par deux listes `lieu1` et `lieu2` qui contiennent leur $latitude$ et $longitude$.

```
def distance(lieu1, lieu2):
    return ((lieu1[0]-lieu2[0])**2+(lieu1[1]-lieu2[1])**2)**(1/2)
```

Afin de rechercher les compteurs proches du lycée, celui-ci sera modélisé comme s'il s'agissait d'un compteur.

Ainsi, on écrira :

```
dorian=['0','74 Av. Philippe Auguste',[48.854576520866964, 2.3926308459105954],0]
```

Question 6 **Écrire** le code permettant de déterminer et d'**afficher** la distance entre le lycée et le premier compteur de la liste `compteurs`. Valider votre code en vérifiant que celle-ci est comprise entre 0.05 et 0.1.

Solution 6

```
dorian=['0','74 Av. Philippe Auguste',[48.854576520866964, 2.3926308459105954],0]
print(distance(dorian[2],compteurs[0][2]))
```

Le résultat est 0.06261658242625116

3 Recherche du plus gros hub à proximité

3.1 Recherche des 5 compteurs les plus proches

Cette partie va consister en la recherche des 5 compteurs les plus proches d'un lieu.
Pour cela, on propose le pseudo-code suivant :

- On crée une fonction `get_five_more_close(depart)`,
- On part d'une distance `d` nulle,
- On créer une liste `five_more_close` vide,
- Tant que cette liste ne compte pas 5 éléments,
- On augmente la distance `d` de 0.0001,
- On cherche dans la liste `compteurs` si un compteur est à une distance inférieure à `d`.
- Si on en trouve un ET que ce n'est pas le `depart` (`!=`) ET qu'il n'est pas déjà dans la liste (`not in`), on l'ajoute à `five_more_close`
- On retourne `five_more_close`

Question 7 **Écrire** le code correspondant à ce pseudo code. **Utiliser** cette fonction afin de **déterminer** et d'**afficher** les 5 plus proches éléments du lycée Dorian.

Remarque : Cette réponse étant nécessaire pour traiter la suite, il est autorisé de demander la correction à votre enseignant. La note à cette question sera alors de 0.

Solution 7

```
def get_five_more_close(depart):
    d=0
    five_more_close=[]
    while len(five_more_close)<5:
        d+=0.0001
        for compteur in compteurs:
            if distance(compteur[2],depart[2])<d and compteur not in five_more_close and
                five_more_close.append(compteur)
    return five_more_close

print(get_five_more_close(dorian))
```

3.2 Recherche du plus gros hub

On appelle « plus gros hub » le compteur qui compte le plus de passage.

Question 8 **Écrire** une fonction getmax(compteurs) qui permet à partir d'une liste de compteurs de retourner celui qui compte le plus de passages.

Ainsi, `print(get_max(compteurs))` renvoie :

```
['100057445-104057445', 'Totem 73 boulevard de Sébastopol S-N', [48.86377, 2.35096], 649
```

Solution 8

```
def getmax(compteurs):
    max=0
    for compteur in compteurs:
        if compteur[3]>max:
            max=compteur[3]
            plus_gros_hub=compteur
    return plus_gros_hub

print(get_max(compteurs))
```

Question 9 **Écrire** une commande permettant de déterminer le plus gros hub parmi les 5 plus proches du lycée Dorian.

Solution 9

```
print(get_max(get_five_more_close(dorian)))
```

4 Suivi d'un parcours de gros hub en gros hub

Afin d'estimer l'itinéraire le plus emprunté à partir de Dorian, la suite va consister à répéter `n` fois les opérations suivantes, à partir d'un point de départ :

1 sélectionner le compteur avec le plus de passage parmi les 5 plus proches,

2 prendre ce compteur comme nouveau point de départ et ré-effectuer l'étape 1,

Question 10 Écrire un algorithme qui effectuerait les opérations précédentes, avec `n=10`.

Solution 10

```
parcours=[dorian]
for i in range(10):
    parcours.append(get_max(get_five_more_close(parcours[i])))
print(parcours)
```

On constate que le parcours ainsi établit retombe plusieurs fois sur le même compteur (il faut « demi-tour »).

Question 11 Modifier la fonction de la question 7 et l'algorithme de la question 10 afin de ne pas retomber deux fois sur le même compteur.

Solution 11

```
def get_five_more_close_v2(hub,exclude):
    d=0
    five_more_close=[]
    while len(five_more_close)<5:
        d+=0.0001
        for compteur in compteurs:
            if distance(compteur[2],hub[2])<d and compteur not in five_more_close and co
                five_more_close.append(compteur)
    return five_more_close

parcours=[dorian]
for i in range(10):
    parcours.append(get_max(get_five_more_close_v2(parcours[i],parcours)))
print(parcours)
```