DS n° 01 – DS01

— 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 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 va consister à déterminer les besoins et les solutions pour l'arrosage automatique d'un balcon parisien sur la période juillet/août.

— Dans un premier temps, nous allons déterminer les besoins en arrosage en analysant les données des précipitations sur ces deux mois,
— Dans un second temps, nous allons proposer une solution d'arrosage automatique.

I Analyse des données météorologiques pour la région parisienne

I.1 Déterminer le poste de mesure météorologique le plus proche de Paris

Le fichier 'postesSynop.csv' présent dans le dossier de travail recense les principaux postes d'observation météorologiques en France. Le premier objectif de cette partie consiste à déterminer celui qui se trouve le plus proche de Paris.
Pour cela nous allons dans un premier temps lire ce fichier en utilisant le code suivant.

```
fichier_postes=open('/home/eleve/Ressources/PTSI/postesSynop.csv','r')
contenu_postes=fichier_postes.read()
fichier_postes.close()
lignes_postes=contenu_postes.split('\n')
entetes_postes=lignes_postes[0].split(';')
postes=lignes_postes[1:]
print(entetes_postes)
```

Question 1 **Recopier** et **exécuter** le code précédent dans votre script python. Il doit afficher la ligne suivante dans la console : `['ID', 'Nom', 'Latitude', 'Longitude', 'Altitude']` .

Solution 1.
Recopier le code

Comme l'indique le résultat du `print(entetes_postes)`, le fichier contient pour chaque poste, son **identifiant** ('ID'), son **nom**, sa **latitude**, sa **longitude** et son **altitude**.

Remarque : La compréhension du code suivant est nécessaire pour répondre à la Question 2.

```
l=[1,2,3,4] # création d'une liste l
print(l[2]) # affichage de l'élément d'index 2 de la liste
```

Question 2 En s'inspirant du code précédent, **afficher** l'élément d'index 3 de la liste `postes`.

Solution 2.

```
print(postes[3])
```

On donne la latitude et la longitude de la ville de Paris sous la forme d'une liste : `paris=[48.85,2.34]`. Une façon de comparer les distances entre les villes consiste à évaluer une image de la distance grâce à la fonction suivante :

$$\text{image distance} = \sqrt{(\text{latitude ville 1} - \text{latitude ville 2})^2 + (\text{longitude ville 1} - \text{longitude ville 2})^2} \quad (1)$$

On souhaite effectuer ce calcul sous python à l'aide d'une fonction `distance_ville`. Dans cette fonction, chaque ville (`ville1` et `ville2`) est définie comme une liste contenant 2 valeurs, sa latitude (ex `ville1[0]`) et sa longitude (ex `ville1[1]`). On rappelle que `**` est le symbole de la puissance en python. Ainsi, `3**2` renvoie 9. L'exemple ci-dessous ne renvoie pas le résultat escompté.

```
def distance_ville(ville1,ville2):
    return ville1[0]**(1/2)+ville2[0]+ville1[1]+ville2[1]**2
```

Question 3 **Modifier** la fonction `distance_ville` ci-dessus afin qu'elle renvoie l'image de la distance définie par l'équation 1.

Solution 3.

```
def distance_ville(ville1,ville2):
    return ((ville1[0]-ville2[0])**2+(ville1[1]-ville2[1])**2)**(1/2)
```

Le code suivant va permettre de déterminer le poste de mesure le plus proche de Paris.

```
min=distance_ville(paris,[0,0])
for poste in postes[:-1]:
    poste=poste.split(';')
    coord_ville=[float(poste[2]),float(poste[3])]
    if distance_ville(paris,coord_ville)<min:
        ville_plus_proche=poste[:2]
        min=distance_ville(paris,coord_ville)
```

Question 4 **Recopier** et **analyser** le code suivant, puis **ajouter** une ligne afin d'afficher le poste le plus près de paris.

Solution 4.

```
min=distance_ville(paris,[0,0])
for poste in postes[:-1]:
    poste=poste.split(';')
    coord_ville=[float(poste[2]),float(poste[3])]
    if distance_ville(paris,coord_ville)<min:
        ville_plus_proche=poste[:2]
        min=distance_ville(paris,coord_ville)
print(ville_plus_proche)
```

Question 5 S'inspirer du code précédent afin d'**écrire** le code permettant de déterminer le poste le plus loin. Pour information, la Base Dumont D'Urville se trouve sur l'Île des Pétrels en Antarctique.

Solution 5.

```
max=0
for poste in postes[:-1]:
    poste=poste.split(';')
    coord_ville=[float(poste[2]),float(poste[3])]
    if distance_ville(paris,coord_ville)>min:
        ville_plus_loin=poste[:2]
        max=distance_ville(paris,coord_ville)
print(ville_plus_loin)
```

Pour la suite, on considérera que l'index du poste le plus proche de Paris est 07149.

I.2 Précipitations sur juillet/août au poste 07149.

Les fichiers 'synop.20220701.csv' et 'synop.20220801.csv' contiennent les relevés météorologiques des postes du fichier 'postesSynop.csv' pour les mois de juillet et août 2022.
Le script suivant permet de lire les entêtes du fichier 'synop.20220701.csv'.

```
fichier=open('/home/eleve/Ressources/PTSI/synop.20220701.csv','r')
contenu=fichier.read()
lignes=contenu.split('\n')
entetes=lignes[0].split(';')
data=lignes[1:]
print(entetes)
print(entetes.index('rr24'))
fichier.close()
```

La commande `print(entetes)` affiche l'ensemble des titres des colonnes du tableau, elle commence comme suit :
`['numer_sta', 'date', 'pmer', 'tend', 'cod_tend', 'dd', 'ff', 't', 'td', 'u', 'vv',...`
La colonne qui nous intéresse est la colonne `'rr24'`, elle indique les précipitations pour les dernières 24h.
La commande `print(entetes.index('rr24'))` nous indique que l'index de cette colonne est 42.
Ainsi, `print(entetes[42])` affiche `'rr24'`.
Le code suivant lit les deux fichiers `'synop.20220701.csv'` et `'synop.20220801.csv'` et stocke les valeurs des précipitations dans une liste `precip`.

```
precip=[]
for mois in ['07','08']:
    fichier=open('/home/eleve/Ressources/PTSI/synop.2022'+mois+'01.csv','r')
    contenu=fichier.read()
    fichier.close()
    lignes=contenu.split('\n')
    data=lignes[1:]
    for idx,donnees_jour in enumerate(data):
        donnees_jour=donnees_jour.split(';')
        if donnees_jour[0]==07149:
            precip.append([donnees_jour[1][:8],donnees_jour[42]])
```

Ainsi, la liste `precip` commence comme suit :
`[['20220701', '12.900000'], ['20220702', '0.000000'], ['20220703', '0.200000'],...`
Ce qui s'interprète par le fait, qu'il a plu 12.9mm le 1er juillet, qu'il n'a pas plu le 2 et qu'il a plu 0.2mm le 3, etc...
On souhaite maintenant connaître les jours pendant lesquels il a plu ($rr24 > 0$) sur ces deux mois. Pour cela on va stocker dans une liste `precip2`, seulement les jours où il a plu.
Ainsi, la liste `precip2` commencerait comme suit :
`[['20220701', '12.900000'], ['20220703', '0.200000'], ['20220721', '10.000000'],...`

Question 6 **Écrire** le code qui permet à partir de la liste `precip` de générer la liste `precip2`.

Solution 6.

```
precip2=[]
for prec in precip:
    if prec[1]!='mq':
        if float(prec[1])>0:
            precip2.append(prec)
```

Question 7 **Écrire** le code qui permet de compter le nombre de jours de pluie sur ces deux mois à partir de la liste `precip2` (vous pourrez travailler à partir de `precip` si vous n'avez pas répondu à la question précédente).

Solution 7.

```
# Solution 1
print(len(precip2))

# Solution 2
jours_pluie=0
for prec in precip:
    if prec[1]!='mq':
        if float(prec[1])>0:
            jours_pluie+=1
print(jours_pluie)
```

II Mise en place d'un arrosage automatique.

On souhaite utiliser une carte électronique programmable en python.
Cette carte permet de :

- lire les relevés d'un capteur d'hydrométrie (ex : `hydro=40` signifie que le degré d'humidité dans l'air est de 40%),
- lire une horloge (ex : `clock=083554` à 8h35min54s),
- commander une pompe qui arrose le jardin (`pompe=T` (allume la pompe pendant T min)).

Question 8 **Créer** un script qui permet de suivre l'algorithme suivant :

- si l'hydrométrie est inférieure à `50` entre 18h et 18h01, on allume la pompe pendant 2 minutes,
- si l'hydrométrie est supérieure ou égale `50` entre 18h et 18h01, on allume la pompe pendant 1 minute,
- si l'hydrométrie est inférieure à `30` quelle que soit l'heure, on allume la pompe pendant 3 minutes.

Solution 8.

```
if (clock>180000 and clock<180100):
    if hydro<50:
        pompe=2
    else:
        pompe=1
elif hydro<30:
    pompe=3
```