Autour de données météorologiques

Problématique

Le service des prévisions et d'observations météorologiques de météo-France relève toute l'année les conditions météorologiques dans chaque ville de France.
Cela permet de déterminer les évolutions de climat et d'aider les prévisions saisonnières.
Étudier le climat passé permet de mieux comprendre le fonctionnement du système climatique, clé pour anticiper ses évolutions futures. Pour cela, les climatologues doivent disposer de séries d'observations sur la période la plus longue possible.

Pour assurer sa mission de conservation de la mémoire du climat, Météo-France assure la collecte, le contrôle et l'archivage des données climatiques dans une base nationale.
Cette base de données contient les données de métropole, d'outre-mer et des TAAF (Terres australes et antarctiques françaises) observées au sol, en mer ou en altitude.
Les principales informations recueillies concernent la température, les précipitations, l'humidité, la pression atmosphérique, le vent et le rayonnement.

L'objectif de ce sujet est

- Partie 1 : de créer quelques outils (autour du minimum) pour le traitement de données météorologiques.
- Partie 2 : d'exploiter les mesures de température.
- Partie 3 : d'extraire des données de la base de données de météo-France.
- Partie 4 : de manipuler des données particulières récupérer dans un fichier.
- Partie 5 : d'étudier une modélisation physique des variations annuelles de température.

Remarque : chaque partie présente des questions de difficultés variées.

Partie 1 : quelques outils pour le traitement des données

Dans toutes cette partie, on s'interdit l'usage de la fonction **min** préprogrammée en Python et permettant d'obtenir directement le minimum d'une liste donnée en argument. En revanche, on se donne la fonction **mini** suivante, écrite en Python :

```
def mini(t):
    '''Calcule le minimum d'un tableau d'entiers ou de flottants.'''
    if len(t) == 0:
        return None
    p = t[0]
    for i in range(1,len(t)):
        if t[i] <= p:
            p = t[i]
    return p
```

Question 1 :
Expliquer le déroulement pas à pas (évolution de la valeur des variables) lors de l'appel **mini([6,2,15,2,15])**, puis donner la valeur renvoyée.

Question 2 :
Évaluer la complexité temporelle de l'appel **mini(t)** en fonction du nombre n d'éléments de `t`.
Vous calculerez pour cela le nombre de comparaisons et d'opérations (addition, soustraction, division, multiplication) réalisées.

Question 3 :
Proposer une modification de la fonction **mini** pour que la valeur renvoyée soit le *maximum* et non le *minimum*.

Question 4 :
Définir, du point de vue de la programmation, ce qu'est une fonction récursive. Donner les étapes de l'écriture d'une fonction récursive en donnant comme exemple la recherche du minimum d'une liste.

Question 5 :
Définir une fonction **mini_recursive** qui a pour argument une liste de valeurs entières ou flottants et qui renvoie par une

méthode récursive le *minimum* de cette liste.

Question 6 :
Écrire une fonction **position_mini** d'argument une liste et renvoyant **une** position du minimum de cette liste.
Dans l'exemple vu plus haut, il y a deux positions pour lesquelles le minimum est atteint : 1 et 3.

On souhaite maintenant déterminer la valeur minimale d'un tableau bidimensionnel d'entiers ou de flottants (c'est une liste de listes).
Par exemple le minimum de $[[10, 3, 15], [5, 13, 10]]$ est 3

Question 7 :
Écrire une fonction **min2D** d'argument un tableau **t** d'entiers ou flottants et renvoyant le minimum de **t**.
On utilisera la fonction **mini**.
Évaluer la complexité temporelle de cette fonction.
Vous calculerez pour cela le nombre de comparaisons et d'opérations (addition, soustraction, division, multiplication) réalisées.

On souhaite, partant d'une liste constituée de couples (chaîne, entier), déterminer la (les) chaîne(s) pour laquelle(s) l'entier/le flottant associé est minimal, voici un exemple d'utilisation de cette fonction :

```
>>> chaine_mini([['Tokyo',7000],['Paris',6000],['Londres',8000]])

'Paris'
```

Question 8 :
Écrire une fonction **chaine_mini** réalisant effectivement cette opération.

On souhaite enfin, à partir d'une liste **L** d'entiers (ou flottants) et d'un entier (ou flottant) appelé `seuil` obtenir le nombre d'éléments de **L** majorés (au sens strict) par `seuil` , voici un exemple d'utilisation de cette fonction :

```
>>> majores_par([12,-5,10,9],10)

2
```

Question 9 :
Écrire enfin une fonction **majores_par** réalisant cette opération.

Question 10 :
Modifier la fonction précédente pour obtenir une fonction **elements_majores_par** retournant la liste des éléments majorés par le seuil.

Partie 2 : Exploitation des mesures de température

La problématique de Météo France est de fournir des valeurs mesurées les plus justes possibles. Les écarts sont essentiellement dus :

- à la configuration de l'installation de la station météo qui doit respecter les normes MF et OMM (à une distance minimum de 2 fois la hauteur des obstacles dans un secteur le plus ensoleillé possible même en hiver, positionné à 1,5m d'un sol herbeux, à, au moins, 100m de sources de chaleur artificielles ou réfléchissantes et d'étendues d'eau). Ces conditions permettent de limiter les erreurs de mesures dues à l'environnement. Ces dernières pouvant facilement atteindre un écart de 50% dans des configurations défavorables.
- à la qualité des appareils de mesure.

Météo-France utilise, dans ses stations météos automatiques, des thermomètres conçus sur la variation de résistance nommés Pt100 (fil de platine dont la résistance vaut 100Ohms à 0°C). Ces sondes possèdent une sensibilité égale à $0,39\Omega.^{\circ}C^{-1}$.
Les sondes de platine utilisées possèdent les caractéristiques suivantes :

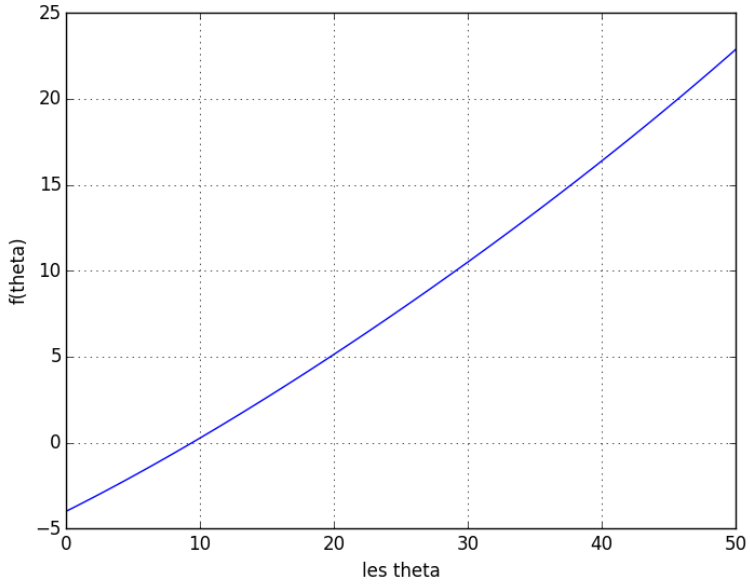
- pour $\theta = 0^{\circ}$, $R(0^{\circ}C) = 100\Omega$
- pour θ température ambiante en (°) et R en Ohms, $R(\theta) = R(0^{\circ}C) \times (1 + a\theta + b\theta^2 + c\theta^3 \times (\theta - 100))$

Les coefficients `a`, `b` et `c` sont donnés par une norme internationale EIT90.

On désire connaitre la température mesurée lorsque la résistance de la sonde est de 104Ω.
On souhaite donc résoudre l'équation $\mathbf{g(\theta,R)=0}$ d'inconnue θ pour une valeur de résistance R connue.

Question 11 :
Définir la fonction $\mathbf{g(\theta,R)}$.

Question 12 :
Ecrire les instructions python permettant de tracer $\mathbf{g(\theta,R)}$ pour $\theta \in [0°, 50°]$ et $R = 104\Omega$ par pas de 0.5°.



Question 13 :
Ecrire une fonction **dichotomie** d'arguments les bornes d'étude `a` et `b`, la fonction `g` ainsi que la précision `epsilon` et qui renvoie la valeur de θ à epsilon près solution de $\mathbf{g(\theta,R)=0}$.

Remarque : `b` et `c` étant très faibles, on utilise généralement la relation approchée suivante :
— $R(\theta) = R(0°C) \times (1 + a\theta)$ avec $a = 3,90802.10^{-3}°C^{-1}$

Le réseau RADOME fournit aux abonnés les données "en temps réel" mesurées aux stations automatiques du réseau français au pas de temps infrahoraire (6mn).
Ces données stockées dans un tableur sont obtenues après traitement des données brutes des différents appareils de mesure. Pour les mesures de température, deux listes sont générées : la liste des temps `les_t` et la liste des températures `les_theta` telles que `les_theta`[i] correspond à la mesure de température à l'instant `les_t`[i].
`les_t=[220.254,220.279,223.254,223.279,.....1012.368]`
`les_theta=[26,26,25.9,25.9,25.8,.......25.5]`

Question 14 :
Ecrire une fonction **tempInfrahoraire** ayant pour arguments une liste des temps en seconde sur un intervalle de 6 minutes et la liste des températures en degré celsius correspondantes et qui renvoie deux valeurs : la moyenne des températures sur cet intervalle de temps et l'heure en seconde correspondant au relevé de température. Les pas de temps ne sont pas constants dans la liste des temps.

Question 15 :
Ecrire les instructions python pour construire les deux listes `les_t_6min` et `les_theta_6min` à partir des deux listes des relevés pour une journée. Vous utiliserez la fonction **tempInfrahoraire**.

Partie 3 : Recherche dans les archives de Météo-France

La base de données de Météo-France concernant l'année écoulée est constituée de deux tables :

— la table **villes** stocke les informations sur chaque ville de France et contient les colonnes (attributs) suivantes :

insee : Chaîne de caractères représentant le numéro INSEE (caractéristique : deux villes différentes ont toujours deux numéros INSEE différents).

nom : Chaîne de caractères représentant le nom de la ville.

dpt : Chaîne de caractères représentant le département de la ville.
Ce n'est pas un simple entier du fait des départements corses numérotés 2A et 2B.

lat : Flottant représentant la latitude de la ville en degrés (positif si la ville est dans l'hémisphère nord).
On prendra 47 degre de latitude Nord comme étant le milieu de la France.

lon : Flottant représentant la longitude de la ville en degrés (positif si la ville est à l'Est du méridien de Greenwich).
On prendra 2 degre de longitude Est comme étant le milieu de la France.

pop : Entier représentant la population de la ville.

Données

Extrait de la table **villes**

insee	nom	dpt	lat	lon	pop
2*B*033	Bastia	2B	42.7	9.45	42912
23067	Saint Denis	23	45.7	2.25	766
25056	Besançon	25	47.23	6.02	115879
46175	Saint Denis	46	44.62	1.98	940
67482	Strasbourg	67	48.58	7.75	761042
93066	Saint Denis	93	48.93	2.35	107762
97411	Saint Denis	97	-20.87	55.43	145347
...					

— la table **mesures** enregistre l'évolution des températures au cours de l'année et contient les colonnes (attributs) suivantes :

ville : Chaîne de caractères représentant le numéro INSEE identifiant la ville.

jour : Entier (de valeur comprise entre 1 et 365 pour l'année 2013) représentant le jour j de l'année où est faite la mesure (1 correspond au premier janvier alors que 365 correspond au 31 décembre).

Tmin : Flottant représentant la température minimale mesurée le jour j en degrés celsius.

Tmax : Flottant représentant la température maximale mesurée le jour j en degrés celsius.

Données

Extrait de la table **mesures**

ville	jour	Tmin	Tmax
25056	110	5.2	9.7
97411	110	25.4	36.8
25056	216	16.7	33.9
67482	363	-2.7	3.3
...			

Question 16 :
Qu'appelle-t-on **clé primaire** pour une table ? Peut-on facilement en définir une pour la table **villes** ? Pour la table **mesures** ?

Question 17 :
Écrire une requête en langage SQL qui récupère depuis la table **mesures** les relevés dont la température moyenne définie par $T_{moy} = \dfrac{T_{min} + T_{max}}{2}$ est strictement inférieure à 0 degré celsius (avec toutes les colonnes disponibles).

Question 18 :
Écrire une requête en langage SQL qui récupère depuis la table **villes** le numéro INSEE, le nom et le département de toutes les villes du quart nord-est de la France, c'est-à-dire celles dont la latitude est supérieure à 47 degrés et dont la longitude est supérieure à 2 degrés.

Partie 4 : Manipulation des données

Depuis la base de données, on a récupéré les relevés météorologiques concernant la ville de Saint-Étienne pour l'année 2016 dans un fichier nommé **SaintEtienne_2016.txt** .
Ce fichier contient 365 lignes (une pour chaque mesure et donc jour de l'année) du type

Données

Extrait du fichier **SaintEtienne_2016.txt**

```
# Jour;Tmin;Tmax
1;4.5;12
```

Données

```
2;5.1;10.2
3;2.5;8.8
...
168;11.8;22.3
169;12.3;21.1
...
366;-4.7;4.7
```

NB : les trois petits points ne sont bien sûr pas présents dans le fichier mais signalent juste que l'on a omis de recopier un certain nombre de lignes.

Sur chaque ligne, la chaîne de caractère correspond à trois champs séparés par des point-virgules, à savoir
- le numéro correspondant au jour de la mesure (entier naturel) ;
- la température minimale mesurée ce jour (en degrés celsius, flottant) ;
- la température maximale mesurée ce jour (en degrés celsius, flottant).

Pour lire des fichiers de ce type, on a écrit :

```
def lecture_fichier(fichier):
    f = open(fichier,mode='r')
    liste=f.readlines()
    f.close()

    jours,Tmin,Tmax = [ ],[ ],[ ]
    for ligne in liste:
        if ligne[0] != '\#':
            t,T1,T2 = ligne.split(';')
            jours.append(int(t))
            Tmin.append(float(T1))
            Tmax.append(float(T2))

    return jours,Tmin,Tmax

jours,Tmin,Tmax = lecture_fichier('SaintEtienne_2016.txt')
```

On admet que l'appel de la fonction a permis de récupérer trois listes : la liste **jours** contenant le numéros du jour et les deux listes **Tmin** et **Tmax** contenant les températures minimales et maximales concernant le jour correspondant.

Question 19 :
Proposer un ordre de grandeur du nombre d'octets utiles du fichier **saintEtienne_2016.txt**.

Question 20 :
Écrire une fonction **moyenne** qui prend en entrée deux listes **a** et **b** de même taille (condition qui ne doit **pas** être vérifiée) et renvoie une liste de même taille contenant dans la case d'indice **i** la valeur moyenne des valeurs des flottants stockés dans les deux listes **a** et **b** à l'indice **i**.

Question 21 :
En appliquant la fonction précédente, écrire l'instruction Python qui stocke dans la variable **Tmoy** la liste des températures moyennes journalières à partir des données stockées dans les listes **Tmin** et **Tmax**.

Question 22 :
On considère qu'il est nécessaire de couper les arrivées d'eau extérieures pour risque de gel quand la température moyenne sur la journée est strictement inférieure à 0 degré celsius. En utilisant une des fonctions programmées dans la première partie, stocker dans la variable **nb_jours_gel** le nombre de jours où il a fallu couper l'eau des conduites extérieures pour la ville de Saint Étienne.

Question 23 :
On s'intéresse ici aux écarts entre la température minimale et maximale au cours d'une même journée, écrire les lignes de commande qui permettent de créer une liste **ecart** contenant les écarts $Tmax - Tmin$ pour chaque jour et de créer la variable mini_ecart contenant le minimum des écarts pour l'année 2016.

Question 24 :
On souhaite étudier les variations d'un jour sur l'autre de la température maximale.
Donner les lignes de commandes permettant d'obtenir, **sans créer de nouvelles listes**, la plus grande variation de la température maximale entre deux jours consécutifs.

Partie 5 : une modélisation physique

On souhaite modéliser les variations annuelles de température en les couplant aux variations saisonnières d'ensoleillement.
On admet que l'équation régissant l'évolution de la température (notée θ exprimée en degrés Celsius) s'écrit

$$\frac{d\theta}{dt} + \alpha\left(\theta(t) + T_0\right)^4 = \mu\left(1 + \varepsilon\cos(\omega t)\right)$$

où α, T_0, μ, ε et ω sont des constantes définies de manières globales et accessibles dans toute fonction.

L'équation différentielle vérifiée par θ peut se mettre sous la forme : $\dfrac{d\theta}{dt} = f(\theta(t), t)$ où f est une fonction.
On souhaite résoudre cette équation différentielle par la méthode d'Euler.
On travaille sur l'intervalle de temps $[0, t\_max]$ avec le pas de temps dt.
On pose $t_0 = 0$, $t_1 = t_0 + dt$, $\cdots$, $t_N = t_{N-1} + dt$ jusqu'à ce que $t_N + dt > t\_max$ et, θ_k une valeur approchée de la température au temps t_k obtenu par la méthode d'Euler.

Question 25 :
Écrire la fonction **f** d'arguments deux flottants t et θ et retournant $f(\theta, t)$.

Question 26 :
Justifier que pour tout k tel que $1 \leqslant k \leqslant N$, on a $\theta_k = \theta_{k-1} + f(\theta_{k-1}, t_{k-1}) \times dt$.

Question 27 :
Écrire la fonction **euler** d'argument trois flottants $theta0$, dt et $tmax$ qui retourne deux listes : **temps** et **theta** contenant respectivement les valeurs t_k et θ_k pour $0 \leqslant k \leqslant N$.