CONCOURS BLANC : INFORMATIQUE

AUTOUR DE DONNÉES MÉTÉOROLOGIQUES

1 Variations autour du minimum

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.

```python
def mini(t):                                                    1
    """Calcule le minimum d'un tableau d'entiers ou de flottants."""  2
    if len(t) == 0:                                             3
        return None                                             4
    p = t[0]                                                    5
    for i in range(len(t)):                                     6
        if t[i] <= p:                                           7
            p = t[i]                                            8
    return p                                                    9
```

Question 1 *On appelle m = mini([6,2,15,2,15]). Donner la valeur de len(t) puis l'ensemble des valeurs qui seront prises par i. Expliquer le déroulement pas à pas (évolution de la valeur des variables) lors du déroulement de la boucle for. Donner enfin la valeur retournée.*

Question 2 *Prouver que lorsque t est une liste non vide d'entiers ou de flottants, mini(t) renvoie la valeur minimale des éléments de t . On pourra exhiber un invariant de boucle précis.*

Remarque

On pourra par exemple proposer la propriété d'invariance suivante : *« à chaque itération de boucle i, p contient le minimum de la liste t[0 :i] », i inclus.*

On rappelle qu'il faut alors :

1. montrer la terminaison du programme ;
2. définir les préconditions (état des variables avant d'entrer dans la boucle) ;
3. montrer que la propriété est vraie en entrant dans la boucle ;
4. montrer que la propriété est vraie à chaque itération ;
5. montrer qu'en sortie de boucle, la condition reste vraie.

Question 3 *Évaluer la complexité temporelle de l'appel mini(t) en fonction du nombre n d'éléments de t.*

Question 4 *Proposer une modification de la fonction mini pour que la valeur renvoyée soit le maximum et non le minimum. On pourra utiliser la numérotation des lignes pour préciser le lieu d'éventuelles modifications et ainsi éviter de réécrire toute la fonction.*

On souhaite récupérer non plus le minimum d'une liste mais la position (ou une des positions) dans le tableau où le minimum est atteint. Dans l'exemple vu plus haut, il y a deux positions où ce minimum est atteint : 1 et 3.

Question 5 *Expliquer le principe d'une fonction réalisant cette opération et en particulier le rôle des variables manipulées. Écrire alors une fonction position_mini réalisant effectivement cette opération*

Question 6 *Préciser l'indice renvoyé si le minimum est présent plusieurs fois dans la liste. Proposer une modification permettant de changer ce comportement.*

On souhaite maintenant déterminer la valeur minimale d'un tableau bidimensionnel d'entiers/de flottants. Un appel de cette fonction pourrait être :

```
>>> mini2D([[10,3,15],[5,13,10]])                    1
        3                                            2
```

Question 7 *Expliquer le principe d'une fonction réalisant ce travail.*

Question 8 *Programmer effectivement cette fonction (on supposera les listes internes de taille non nulle) en utilisant la fonction mini.*

Question 9 *Évaluer la complexité temporelle de cette fonction.*

On souhaite, partant d'une liste constituée de couples (chaîne, entier), déterminer la/une chaîne pour laquelle l'entier/le flottant associé est minimal :

```
>>> chaine_mini([['Tokyo',7000],['Paris',6000],['Londres',8000]])    1
        'Paris'                                                      2
```

Question 10 *Écrire une fonction chaine_mini réalisant effectivement cette opération.*

Question 11 *Écrire enfin une fonction majores_par prenant en entrée une liste t d'entiers/de flottants ainsi qu'un entier/flottant seuil et renvoyant le nombre d'éléments de t majorés (au sens strict) par seuil :*

```
>>> majores_par([12,-5,10,9],10)                     1
        2                                            2
```

2 Manipulation de données

Vous avez à présent fini votre séjour à Météo-France mais vous avez pu sauvegarder précieusement dans un fichier nommé besancon_2013.txt les relevés météos concernant la ville de Besançon pour l'année 2013. Ce fichier contient 365 lignes (une pour chaque mesure et donc jour de l'année) du type suivant :

Extrait du fichier besancon_2013.txt

```
# Jour ;Tmin ;Tmax
1 ;2.1 ;7.6
2 ;2.3 ;4.9
3 ;-1.9 ;5.7
...
168 ;16.7 ;32.3
169 ;18.8 ;32.
...
365 ;-3.2 ;1.9
```

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, vous écrivez la procédure suivante :

```python
def lecture_fichier ( fichier ):
    f = open(fichier ,mode='r')
    jours ,Tmin,Tmax = [ ],[ ],[ ]
    for ligne in f:
        if ligne [0] != '#':
            t,T1,T2 = ligne. split (';')
            jours .append(int(t))
            Tmin.append(float(T1))
            Tmax.append(float(T2))
    f. close ()
    return jours ,Tmin,Tmax

jours ,Tmin,Tmax = lecture_fichier('besancon_2013.txt')
```

On rappelle que append rajoute l'élément donné en argument à la fin de la liste sur laquelle on l'applique et voici ci-dessous le descriptif de l'aide Python concernant l'action de split sur une chaîne de caractères :

```python
split (...)
    S. split (sep=None, maxsplit=-1) -> list of strings
    Return a list of the words in S, using sep as the delimiter string .
    If maxsplit is given, at most maxsplit splits are done.
    If sep is not specified or is None, any whitespace string is a
    separator and empty strings are removed from the result .
```

Question 12 *Proposer un ordre de grandeur du nombre d'octets utiles du fichier besancon.txt.*

Question 13 *Décrire à quoi servent les lignes 5 et 6 du programme précédent. Donner le type des variables jours, Tmin et Tmax. Donner ensuite, si cela a un sens, le type des objets contenus dans jours, Tmin et Tmax.*

Question 14 *Expliquer pourquoi il est préférable d'utiliser jours.append(int(t)) (ligne 7) plutôt qu'une concaténation du type jours = jours + [int(t)].*

Question 15 *Écrire une fonction moyenne qui prend en entrée deux listes a et b de mêmes tailles (condition que la fonction devra vérifier préalablement) 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 16 *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 17 *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°C. 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 Besançon.*

Question 18 *Donner la suite d'instruction permettant de tracer sur un même graphe, la courbe des températures moyennes, minimales et maximales.*