

TP Informatique 07

Évaluation automatique de l'exercice 1 (15 minutes)

Télécharger le fichier TP07_EX01 depuis la rubrique Informatique/TP/TP07 MAX-MIN, l'ouvrir et le compléter. Pour cela, supprimer l'instruction `pass` et compléter les fonctions. Au bout de 15 minutes maximum, se connecter au site de la classe et téléverser le fichier dans la rubrique Informatique/Transfert de documents. **Le fichier déposé doit pouvoir être exécuté sans message d'erreur et sans effet de bord (pas d'appel de la fonction `print`).**

Exercice 1

Vous respecterez l'utilisation d'une boucle conditionnelle `while` ou inconditionnelle `for` imposée dans chaque question.

1. On souhaite écrire une fonction `somme(n)` renvoyant la valeur de la somme $\sum_{k=0}^n k$. On propose :

```
def somme(n):  
    """n : entier"""  
    k,res=0,0  
    while k<n:  
        res+=k  
        k+1  
    return res
```

Identifier la (ou les) erreur(s) de ce code et écrire une version corrigée de la fonction `somme`. Cette fonction utilisera nécessairement une boucle `while`.

2. En vous inspirant du code précédent (toujours avec une boucle `while`), écrire une fonction `fact(n)` renvoyant $n!$.
3. Écrire une fonction `somme1(n)`, avec boucle conditionnelle, d'argument `n` un entier et qui renvoie $\sum_{0 \leq 2k+1 \leq n} (2k+1)^2$.
4. Écrire une fonction `geom(n,q)`, avec boucle inconditionnelle, d'arguments `n` un entier et `q` un flottant et qui renvoie la valeur de $\sum_{k=0}^n q^k$. On veillera à calculer efficacement les puissances successives de q^k en s'appuyant sur la valeur de q^{k-1} déjà calculée.

 Dans ce qui suit, l'utilisation des fonctions natives `max`, `min` n'est pas autorisée.

Exercice 2

1. Écrire une fonction `mini(L)` d'argument `L` une liste non vide de nombres et qui renvoie le minimum de cette liste.

2. Écrire une fonction `mini_pos(L)` d'argument `L` une liste non vide de nombres et qui renvoie le minimum de cette liste ainsi que l'indice de sa première occurrence.
3. Modifier la fonction précédente pour avoir non plus la première occurrence du minimum mais la dernière.

Exercice 3

1. Écrire une fonction `maxi_mini(L)` d'argument `L` une liste non vide de nombres et qui renvoie le minimum et le maximum de cette liste. On effectuera une seule boucle `for`.
2. Écrire une fonction `maxi_mini_pos(L)` d'argument `L` une liste de nombres et qui renvoie le maximum et le minimum de la liste ainsi que les indices des premières occurrences de ce maximum et minimum. On effectuera une seule boucle `for`.

Exercice 4

1. Compléter le code `max2(L)` d'argument `L` une liste contenant au moins deux nombres pour qu'il renvoie les deux plus grands éléments de `L`.
Par exemple, l'appel `max2([1,3,4,2])` renvoie `(4,3)` et l'appel `max2([1,3,2,3])` renvoie `(3,3)`.

```
def max2(L):
    """max2(liste) : détermine 2 plus grands éléments de liste"""
    a,b=L[0],L[1]
    if a<b:
        a,b=b,a
    for k in range(2,len(L)):
        c=L[k]
        if c>a:
            ...
        elif ... :
            ...
    return a,b          # a=max(liste) et b=max(liste\{a})
```

2. Écrire une nouvelle version `max2_ind(L)` qui renvoie les valeurs et des indices des occurrences des deux plus grands éléments (on n'impose pas qu'il s'agisse des premières ou dernières occurrences).

Exercice 5

Dans cet exercice, on ne fera pas appel à une fonction extérieure calculant le maximum d'une liste de nombres.

1. Écrire une fonction `maxi_occ1(L)` d'argument `L` une liste non vide de nombres et qui renvoie la liste des indices de toutes les occurrences du maximum de `L`. Par exemple, l'appel `maxi_occ1([1,3,2,0,3])` renvoie `[1,4]`. On autorise l'utilisation de deux boucles `for` consécutives.
2. Écrire une nouvelle version `maxi_occ2(L)` qui renvoie le même résultat que `maxi_occ1(L)` mais avec une seule boucle `for`.