ISM MPSI/PCSI, Informatique
Année 2024/2025

TP Informatique 8

⚠ On rappelle qu'un script (fichier `*.py`) doit être enregistré et exécuté (touche `F5`) pour que les fonctions saisies dans le script soient utilisables dans la console.

Exercice 1

1. Écrire une fonction `detect(elt,L)` d'argument `elt` un entier et `L` une liste d'entiers qui renvoie `True` si `elt` est dans `L` et `False` sinon. On n'utilisera pas l'instruction `in` pour tester l'appartenance d'un élément à une liste.
2. Écrire une fonction `pos(elt,L)` d'argument `elt` un entier et `L` une liste d'entiers et qui renvoie la liste des indices de `elt` dans `L`. Si `elt` est absent de `L`, la fonction renvoie la liste vide. Par exemple, l'appel `pos(3,[3,1,2,3,0,-1,5,3])` renvoie `[0,3,7]`.

Exercice 2

Écrire une fonction `elt(L)` d'argument une liste `L` et qui renvoie une nouvelle liste constituée des éléments de `L` en supprimant tous les doublons. Par exemple, l'appel `elt([1,2,3,3,4,2])` renvoie `[1,2,3,4]`. La fonction `elt` pourra utiliser la fonction `detect` écrite dans l'exercice 1.

Exercice 3

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 4

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 5

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 6

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`.

Exercice 7

Compléter le code `preder(L,elt)` pour qu'il renvoie la première et la dernière occurrence de `elt` dans la liste `L`. Si `elt` ne possède qu'une occurrence dans `L`, la première et la dernière occurrence coïncident. Si `elt` est absent de `L`, le code doit renvoyer le tuple `(None, None)`. Une des contraintes de `preder` consiste à réaliser ce travail avec une seule boucle `for`.

```
def preder(L,elt):
    n=len(L)
    g,d=None,None
    for k in range(n):
        if ... :
            ...
        if ... :
            g=n-(k+1)
    return g,d
```