Correction TP2 Python ECG2

Exercice 1

1)L est formée des termes 2^k pour k allant successivement de 1 à 5.
Donc L=[2,4,8,16,32].

2)M est formée des entiers k pairs avec k allant successivement de 0 à 19.
Donc M=[0,2,4,6,8,10,12,14,16,18].

3)N est formée des entiers de la forme ij où i est un entier entre 1 et 3 et j un entier entre 1 et 4. Une liste étant un ensemble ordonné, il faut être vigilant sur l'ordre des éléments qu'on écrit.
On prend i=1, puis j=1,2,3,4, ce qui donne par produit : 1,2,3,4
Ensuite, on prend i=2, puis j=1,2,3,4, ce qui donne par produit : 2,4,6,8
Enfin, on prend i=3, puis j=1,2,3,4, ce qui donne par produit : 3,6,9,12.
On obtient finalement : N =[1, 2, 3, 4, 2, 4, 6, 8, 3, 6, 9, 12].

Exercice 2

1)L est formée de tous les entiers impairs de 1 à 143.
L=[k for k in range(1,143) if k%2==1] ou bien L=[2*k+1 for k in range(72)].

2)M est formée de tous les carrés parfaits entre 1 et 100.
M=[k**2 for k in range(1,11)]

3)N=[(k+1)/k for k in range(1,100)]

Exercice 3

1)(i)L est formée des nombres de forme 3k, k$\in$ [[0, 99] et k multiple de 5.
On a donc k$\in$ {5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95}.
D'où L=[0,15,30,45,60,75,90,105,120,135,150,165,180,195,210,225,240,255,270,285].

(ii)Le plus rapide est l'instruction : `M=L[5:len(L)-5]`

Mais, on peut aussi utiliser la commande `pop` dans une boucle :

```
for i in range(5):
    del L[0] # supprime le premier élément de la liste
    del L[-1] # supprime le dernier élément de la liste
M=L
```

On obtient au final : M=[75, 90, 105, 120, 135, 150, 165, 180, 195, 210]

2)I=[1/k for k in range(1,101)]

Exercice 4

1)La fonction crée une liste intitulée `indices`, vide initialement.
Dans la boucle, elle regarde si la valeur d'index k de la liste L vaut 2.
Si c'est le cas, elle ajoute cette valeur à la liste `indices`.
Le programme renvoie donc finalement la liste de tous les indices des éléments de L dont la valeur de l'indice vaut 2.

2)Avec L=[1,2,8,7,4,5,6,7,2], on obtient indices=[1,8].
Avec L=[5,4,6,3,1,8], on a indices=[ ] puisqu'il n'y a pas de 2 dans L.

Exercice 5

On parcourt la liste à l'aide de ses éléments ou des indices des éléments :

```
def nb_occurrences(a, L):
    n=0
    for x in L:
        if x ==a:
            n=n+1
    return n
```

```
def nb_occurrences(a, L):
    n=0
    for k in range(len(L)):
        if L[k]==a:
            n=n+1
    return n
```

Il est possible aussi d'utiliser une compréhension de liste (avec modération !)

```
def nb_occurences(a,L):
    doublons=[x for x in L if x==a]
    return len(doublons)
```

```
def nb_occurences(a,L):
    indices=[k for k in range(len(L)) if L[k]==a]
    return len(indices)
```

Remarque
La fonction intégrée `count` retourne directement le nombre d'occurrences de a dans L par l'instruction `L.count(a)`, mais elle doit être redonnée au concours.

Exercice 6

Comme on demande le rang de la première occurence de a, il vaut mieux parcourir la liste L à l'aide des indices des éléments.

```
def rang(a, L):
    for k in range(len(L)):
        if L[k]==a:
            return k
    return -1
```

Remarque
La commande `index` retourne directement le rang de la première occurrences de a dans L par l'instruction `L.index(a)`, mais elle doit être redonnée au concours.

Exercice 7

1)• On peut parcourir directement les éléments de la liste :

```
def somme(L):
    s=0
    for x in L:
        s=s+x
    return s
```

• Ou les parcourir à l'aide de leurs indices :

```
def somme(L):
    s=0
    for k in range(len(L)):
        s=s+L[k]
    return s
```

2)On utilise la fonction `somme` précédente.

```
def moyenne(L):
    return somme(L)/len(L)
```

Remarques

Les commandes `sum` et `mean` renvoient immédiatement la somme et la moyenne des éléments de L par les instructions `sum(L)` et `numpy.mean(L)`, elles sont au programme de la classe, on les verra dans un prochain TP.

Exercice 8

• On peut parcourir directement les éléments de la liste :

```
def minimum(L):
    m=L[0]
    for x in L:
        if x<m:
            m=x
    return m
```

• Ou les parcourir à l'aide de leurs indices :

```
def minimum(L):
    m=L[0]
    for k in range(len(L)):
        if L[k]<m:
            m=L[k]
    return m
```

Pour obtenir le plus grand élément de L, il suffit de changer dans le test $<$ par $>$.

Remarque Les commandes `min` et `max` renvoient respectivement le plus petit et le plus grand élément de L par l'instruction `min(L)` et `max(L)`, elles sont au programme de la classe, mais devraient être redonnée au concours.

Exercice 9

1)Programme :

```
def supprime(a, L):
    M=[]
    for k in range(len(L)):
        if L[k]!=a:
            M.append(L[k])
    return M
```

2)Programme :

```
def supprime(a, L):
    M=[x for x in L if x != a]
    return M
```

Exercice 10

```
def envers(L):
    k=len(L)-1
    M=[]
    while k>=0:
        M.append(L[k])
        k=k-1
    return M
```

Remarque

La commande `reverse` fait le job, par l'instruction `L.reverse()`, mais elle n'est pas au programme de la classe !

Exercice 11

La fonction `indicemini` s'inspire de l'exercice 8. Il faut juste prendre soin de repérer à chaque étape l'indice i où se produit le minimum.

```
def indicemini(L):
    m=L[0]
    i=0
    for k in range(len(L)):
        if L[k]<m:
            i=k
            m=L[k]
    return i
```

Puis, on complète cette fonction avec la fonction `tri` en construisant une liste notée "croissante".

```
def tri(L):
    croissante=[]
    while len(L)>0:
        i=indicemini(L)
        croissante.append(L[i])
        L.pop(i)
    return croissante
```

Une autre possibilité est de travailler avec des sous-listes (plus délicat) :

```
def tri(L):
    for i in range(len(L)):
        imin=indicemini(L[i:len(L)])+i
        L[i],L[imin]=L[imin],L[i]
    return L
```

> Remarque
> La commande `sort` trie la liste dans l'ordre croissant par l'instruction `L.sort()`, mais elle n'est pas au programme de la classe !