Mathématiques supérieures — le Mardi 9 Décembre 2025

Correction du Concours Blanc d'Informatique n°1

Ce sujet est composé de deux exercices et un problème. Vous prendrez soin de bien justifier vos calculs.
Veillez à bien respecter l'indentation des programmes Python que vous écrirez sur votre copie.

Exercice 1.

Dans cet exercice, pour chaque fonction, l'argument `x` est un nombre et l'argument `L` est une liste de nombres.

1. Écrire une fonction `recherche(x,L)` qui renvoie `True` si la valeur `x` se trouve dans la liste `L` et `False` sinon.
2. Écrire une fonction `recherche_indice(x,L)` qui renvoie l'indice dans `L` de la première occurence de `x` si la valeur `x` se trouve dans la liste `L` et `False` sinon.
3. Écrire une fonction `recherche_tous_indices(x,L)` qui renvoie une liste contenant les indices dans `L` de toutes les occurences de `x` si la valeur `x` se trouve dans la liste `L` et une liste vide `[]` sinon.

Correction.

1.

```
def recherche(x,L):
    for v in L:
        if v == x:
            return True
    return False
```

2.

```
def recherche_indice(x,L):
    for i in range(len(L)):
        if L[i] == x:
            return i
    return False
```

3.

```python
def recherche_tous_indices(x,L):
    Ind = []
    for i in range(len(L)):
        if L[i] == x:
            Ind.append(i)
    return Ind
```

Exercice 2.

1. Comportement des listes Python

   Que renvoient les instructions suivantes :

   (a) `>>> [0,0,0]+[1,1,1]`

   (b) `>>> 3*[1,1,1]`

2. Programmation de fonctions

   (a) Écrire une fonction `produit(M,N)` d'arguments deux listes de nombres `M` et `N` de même longueur et qui renvoie **une nouvelle liste** constituée du produit terme à terme des deux listes. *Exemple de comportement attendu :*

   ```
   >>> produit([2,0,-2],[2,3,4])
   [4,0,-8]
   ```

   (b) Écrire une fonction `somme(L)` d'argument une liste non vide de nombres `L` et qui renvoie la somme des éléments de `L`. *Exemple de comportement attendu :*

   ```
   >>> somme([2,3,4])
   9
   ```

   (c) Écrire une fonction `moyenne_ponderee(Notes,Coeff)` d'arguments deux listes non vides de nombres `Notes` et `Coeff` de même longueur et qui renvoie la moyenne coefficientée des notes de la liste `Notes` dont les coefficients sont donnés par la liste `Coeff`. On suppose que la somme des éléments de `Coeff` n'est pas nulle - on n'aura pas à le vérifier dans le corps de la fonction.

   Par exemple, l'instruction `moyenne_ponderee([12,8,14],[2,3,1])` doit donner le résultat m du calcul :

   $$m = \frac{12 \times 2 + 8 \times 3 + 14 \times 1}{2 + 3 + 1}$$

   Ainsi, on aura :

```
>>> moyenne_ponderee([12,8,14],[2,3,1])
10.333333333333
```

On pourra avoir recours aux fonctions `produit` *et* `somme` *des deux questions précédentes - et ce même si on n'a pas réussi ces questions !*

3. Fonctions mystères

Le but des deux prochaines questions est de comprendre les algorithmes proposés.

(a) Dans la fonction suivante, l'argument `n` est un entier naturel.

```
def mystere(n):
    p = 1
    for i in range(1,n+1):
        p = p*i
    return p
```

i. Que renvoie `mystere(4)` ?

ii. Plus généralement, que renvoie `mystere(n)` pour `n` un entier naturel ?

(b) Dans la fonction suivante, l'argument `L` est une liste non vide de nombres.

```
def enigme(L):
    m = L[0]
    for i in range(1,len(L)):
        if L[i] > m:
            m = L[i]
    return m
```

i. Que renvoie `enigme([2,1,3,0])` ?

ii. Plus généralement, que renvoie `enigme(L)` pour `L` est une liste non vide de nombres ?

Correction.

1. Comportement des listes Python

Que renvoient les instructions suivantes :

(a)
```
>>> [0,0,0]+[1,1,1]
[0,0,0,1,1,1]
```

(b)

```
>>> 3*[1,1,1]
[1,1,1,1,1,1,1,1,1]
```

2. Programmation de fonctions

(a)

```python
def produit(M,N):
    P = []
    for i in range(len(M)):
        P[i].append(M[i]*N[i])
    return P
#ou avec la syntaxe des listes Python
def produit(M,N):
    return [M[i]*N[i] for i in range(len(M))]
```

(b)

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

(c)

```python
def moyenne_ponderee(Notes, Coeff):
    P = produit(Notes, Coeff) # liste de chaque note/coefficient
        correspondant
    n = somme(P) # numérateur de la moyenne
    s = somme(Coeff) # dénominateur de la moyenne
    return P/s # moyenne
```

3. Fonctions mystères

Le but des deux prochaines questions est de comprendre les algorithmes proposés.

(a) i. Elle renvoie 24

ii. Elle renvoie $n!$.

(b) i. Elle renvoie 3

ii. Elle renvoie le maximum de la liste `L`.

Exercice 3. *Parties d'un ensemble*

On considère l'ensemble $[\![0, 100]\!]$ des entiers de 0 à 100.
Dans cet exercice, on modélise un sous-ensemble de $[\![0, 100]\!]$ par une liste sans doublons dont les élements sont des entiers compris entre 0 et 100.
Par exemple, la liste `[8,21,44,99]` modélise le sous-ensemble $\{8, 21, 44, 99\}$ de $[\![0, 100]\!]$

1. Donner des instructions qui permettent de modéliser les sous-ensembles de $[\![0, 100]\!]$ suivants sous forme de liste :
   (a) $\{1, 7, 12\}$
   (b) $[\![0, 50]\!]$
   (c) $\{0, 2, 4, 6, 8, ..., 98, 100\}$.
2. Déterminer le sous-ensemble de $[\![0, 100]\!]$ modélisé par la liste :

```
[i**2 for i in range(21) if i%3==1]
```

3. Écrire une fonction `estdedans(L,k)` qui prend pour arguments une liste `L` de nombres entiers et un nombre entier `k` et qui retourne le booléen `True` si `k` est dans la liste `L` et qui retourne `False` sinon.
   Par exemple, l'instruction `estdedans([1,5,9],5)` doit renvoyer `True`.
   Dans la suite, on pourra se servir librement de la fonction `estdedans` qu'on ait réussi à la coder ou non.
4. Que fait la fonction `uni` ci-dessous pour deux listes `L` et `M` ? À quelle opération mathématique ensembliste correspond-elle pour des parties ?

```
def uni(L,M):
    N=[]
    for k in range(101):
        if estdedans(L,k) or estdedans(M,k):
            N.append(k)
    return N
```

5. Écrire une fonction `inter(L,M)` qui retourne la liste qui correspond à l'intersection des parties représentées par `L` et `M`.
6. Écrire une fonction `comp(L)` qui retourne la liste qui correspond au complémentaire dans $[\![0, 100]\!]$ de la partie représentée par `L`.

Correction.

1. (a) `[1,7,12]`
   (b) `[i for i in range(51)]` ou `list(range(51))`
   (c) `[2*i for i in range(51)]` ou `[i for i in range(101) if i%2==0]` ou `[i for i in range(0,101,2)]` ou `list(range(0,101,2))`.
2. Il s'agit des nombres de la forme i^2 avec i entier compris entre 0 et 20 vérifiant $i = 3k + 1$

avec k un entier donc l'ensemble recherché est :

$\{(3\times 0+1)^2,(3\times 1+1)^2,(3\times 2+1)^2,(3\times 3+1)^2,(3\times 4+1)^2,(3\times 5+1)^2,(3\times 6+1)^2\}$
$=$
$\{1,16,49,100,169,256,361\}$.

3.

```
def estdedans(L,k):
    for e in L:
        if e == k:
            return True
    return False
```

4. La fonction `uni(L,M)` renvoie une liste contenant les éléments qui sont dans `L` ou dans `M` sans doublons.
L'opération mathématique sous-jacente est l'union ($\cup$).

5.

```
def inter(L,M):
    N=[]
    for k in range(101):
        if estdedans(L,k) and estdedans(M,k):
            N.append(k)
    return N
```

6.

```
def comp(L):
    N=[]
    for k in range(101):
        if not estdedans(L,k):
            N.append(k)
    return N
```