INUC et ISM

Mardi 28 février 2014

Contrôle Programmation Python n°2 (durée 1h30)

Le cours est autorisé. La correction des exercices de TP n'est pas autorisée. Vous devez rendre toutes vos solutions dans un unique fichier, le fichier joint `nom_prenom.py` *et en plaçant votre nom et prénom dans le nom du fichier (uniquement des lettres et le caractère* `_` *et sans placer d'espaces ou d'accents). L'énoncé comporte 3 exercices.*

1. Sous-listes d'une liste (7 points)

On donne une liste `L` d'entiers entre 0 et 9, par exemple

$$\texttt{L} = [5, 5, 0, 8, 1, 4, 2, 3, 7, 8, 1, 4, 2, 3, 3]$$

On observe que cette liste contient exactement à deux reprises la sous-liste `M = [8, 1, 4, 2]` : en effet, les chiffres 8, 1, 4, 2 apparaissent les uns à la suite des autres dans `L` en exactement deux endroits.

Ecrire une fonction `nb_sous_listes(L, M)` qui renvoie le nombre de sous-listes identiques à `M` et qui sont présentes dans `L`. Avec l'exemple ci-dessus, la fonction doit renvoyer 2.

2. Tourner une matrice d'un quart de tour *(7 points)*

1)a) Ecrire une fonction `rotate(M)` qui étant donné une matrice carrée `M` de taille $n \times n$ renvoie la matrice `N` déduite de `M` par rotation d'un quart de tour dans le sens des aiguilles d'une montre.

Par exemple, voici une matrice `M` et la matrice `N = rotate(M)` :

$$M = \begin{pmatrix} 23 & 51 & 75 & 50 & 56 & 94 \\ 52 & 93 & 97 & 17 & 99 & 79 \\ 21 & 33 & 61 & 97 & 82 & 10 \\ 42 & 69 & 20 & 18 & 80 & 69 \\ 54 & 58 & 54 & 86 & 20 & 44 \\ 23 & 12 & 41 & 53 & 27 & 78 \end{pmatrix} \quad N = \begin{pmatrix} 23 & 54 & 42 & 21 & 52 & 23 \\ 12 & 58 & 69 & 33 & 93 & 51 \\ 41 & 54 & 20 & 61 & 97 & 75 \\ 53 & 86 & 18 & 97 & 17 & 50 \\ 27 & 20 & 80 & 82 & 99 & 56 \\ 78 & 44 & 69 & 10 & 79 & 94 \end{pmatrix}$$

b) En déduire une fonction `antiRotate(M)` qui renvoie la matrice déduite de `M` par rotation d'un quart de tour dans le sens *inverse* des aiguilles d'une montre.

2) On se propose d'obtenir `N` par une autre méthode. On rappelle que la *transposée* d'une matrice A est la matrice dont les lignes sont exactement les colonnes de A.

On considère par ailleurs la matrice J de taille $n \times n$ dont tous les termes valent 0 sauf les termes de la diagonale secondaire qui eux valent tous 1. Par exemple, si $n = 3$ alors $J = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$. La diagonale secondaire est la diagonale qui joint le coin inférieur gauche au coin supérieur droit.

a) Ecrire une fonction `matriceJ(n)` qui renvoie la matrice `J` de taille `n x n`. On utilisera les fonctions `matriceNulle` et `afficher` du fichier `matrices.py`.

b) On admettra alors que si `T` est la transposée de la matrice `M` alors le produit de matrices de matrices `T x J` est la matrice `N` déduite de `M` par rotation d'un quart de tour dans le sens des aiguilles d'une montre.

Ecrire une fonction `rotation(M)` qui renvoie la matrice `N` en utilisant la méthode que l'on vient de décrire. On utilisera les fonctions `dimensions`, `transpose`, `produit` et `afficher` du fichier `matrices.py`.

3. Le plus petit et le plus grand *(6 points)*

Soient une liste L d'entiers, de longueur $n > 0$ et k un entier tel que $1 \leq k \leq n$. Ecrire une fonction récursive `min_max(L, k)` qui renvoie la liste de deux éléments `[m, M]` constituée

- du plus petit élément `m`
- du plus grand élément `M`

choisis parmi les k premiers éléments de la liste `L`.

Voici quelques exemples de comportements de la fonction :

```
L = [42, 81, 31, 81, 12, 99, 81], k = 4   -->   [31, 81]
L = [42, 81, 31, 81, 12, 99, 81], k = 2   -->   [42, 81]
L = [42], k = 1   -->   [42, 42]
L = [42, 42, 42, 42], k = 3   -->   [42, 42]
```

Explication du premier exemple. Comme $k = 4$, on recherche le plus petit et le plus grand élément des 4 premiers éléments de la liste `L` autrement dit, le plus petit et le plus grand élément de la liste `[42, 81, 31, 81]`. Le plus petit élément est bien 31 et le plus grand est bien 81 d'où la réponse `[31, 81]`.