Mercredi 8 janvier 2025 – *Durée : 1 heure*

Devoir Surveillé n°2 (a) – *Corrigé*

Exercice n°1 Comme un air de (presque) déjà vu

R1. Écrire une fonction `imini(L:list)->int` qui prend en argument une liste `L` et qui renvoie la position de la première occurrence (=apparition) du minimum.

Solution:

```
def imini(L):
    rang_min=0
    for i in range(1,len(L)):
        if L[i]<L[rang_min]:
            rang_min=i
    return rang_min
```

R2. Écrire une fonction d'en-tête `dim(L:list[list])->(int,int)` qui prend en argument une liste de listes `L` qui représente une matrice et qui renvoie le couple d'entiers `(n,m)` où `n` est la hauteur (nombre de lignes) et `m` la largeur (nombre de colonnes).

Solution:

```
def dim(L):
    n = len(L) # compte le nombre de sous-liste dans la liste, soit le nombre de lignes
    m = len(img[0]) # compte le nombre d'éléments dans la sous-liste img[0], soit le nombre de colonnes (toutes les lignes ont le même nombre d'éléments)
    return n,m
# OU
def dim(img):
    return (len(img),len(img[0]))
```

R3. Écrire une fonction d'en-tête `nulle(n:int,m:int)->list[list]` qui renvoie une liste de listes qui correspond à la matrice nulle de `n` lignes et `m` colonnes. *L'utilisation des fonctions de* ***numpy*** *est interdite.*

Solution:

```
def nulle(n,m):
    return [[0 for j in range(m)] for i in range(n)]
# OU
def nulle(n,m):
    return [[0]*m for i in range(n)]
# OU
def nulle (n,m):
    L=[]
    for i in range(n): # les lignes
```

```
        L_ligne=[] # liste vide d'initialisation d'une ligne
        for j in range(m): # on rempli la ligne i
            L_ligne.append(0)
    L.append(L_ligne)
    return L
```

R4. Écrire une fonction d'en-tête `copie(L:list[list])->list[list]` qui prend en argument une matrice `L` (liste de listes) et qui renvoie une copie de la matrice `L` totalement indépendante de `L`. Il est interdit d'utiliser le module `copy` et les fonctions `deepcopy` ou `deepcopy`, par contre, vous êtes invités à utiliser les deux fonctions écrites précédemment. *On réutilisera avantageusement des fonctions précédentes.*

Solution:

```
def copie(L):
    h,w=dim(img)
    L2=nulle(h,w) # initialisation de l'image à 0
    for i in range(h):
        for j in range(w):
            L2[i][j]=L[i][j]
    return L2
```

R5. Écrire une fonction `fois_trois(L:list[list])->list[list]` qui prend en argument une matrice `L` et qui renvoie une nouvelle matrice `L3` qui correspond à `L` dont tous les coefficients ont été multipliés par trois.
On pourra réutiliser les fonctions précédentes.

Solution: question tout à fait accessible !

```
def fois_trois(L):
    h,w=dim(L)
    L3=nulle(h,w) # tableau qui recevra l'image réduite
    for i in range(h): # parcours de img
        for j in range(w):
            L3[i][j]=L[i][j]*3
    return L3
```

R6. Écrire une fonction `reduc_tier(L:list[list])->list[list]` qui prend en argument une matrice `L` et qui renvoie une matrice trois fois moins large en ne gardant que les colonnes de rang multiple de 3.
On pourra réutiliser les fonctions précédentes.

Solution: question tout à fait accessible !

```
def reduc_moitie(L):
    h,w=dim(L)
    img_red=nulle(h,w//3) # tableau qui recevra l'image réduite
    for i in range(h): # parcours de img
        for j in range(w//3):
            img_red[i][j]=L[i][3*j]
    return img_red
```

Exercice n°2 Base de données

Nous utilisons une base de données nommée `world` qui comporte trois tables et dont le schéma relationnel est le suivant :

`city (Id, Name, CountryCode, Population)`

`country (Code, Name, Continent, SurfaceArea, Population, Capital)`

`countrylanguage (Id, CountryCode, Language, IsOfficial, Percentage)`

Les clés primaires sont les champs nommés `Id` pour les tables `city` et `countrylanguage`, et `Code` pour la table `country`.

Les champs nommés `CountryCode` sont des clés étrangères en références à la clé primaire `Code` de la table `country`. Le champ `Capital` est une clé étrangère en référence à la clé primaire `Id` de `city`.

R7. Écrire la requête SQL qui permet d'obtenir la capitale du Portugal.

Solution:

```
SELECT city.Name
FROM city
JOIN country
ON Capital=Id
WHERE country.Name='Portugal'
```

R8. Écrire la requête SQL qui permet d'obtenir la liste des langues parlées à Bruxelles.

Solution:

```
SELECT Language
FROM countrylanguage AS CL
JOIN country
JOIN city AS C
ON C.CountryCode=Code AND CL.CountryCode=Code
WHERE C.Name='Bruxelles'
```

R9. Écrire la requête SQL qui permet d'obtenir le nombre de pays par continent.

Solution:

```
SELECT Continent,COUNT(Code)
FROM country
GROUP BY Continent
```

R10. Écrire la requête SQL qui permet d'obtenir la liste des continents de plus de 1 milliard d'habitants.

Solution:

```
SELECT Continent
FROM country
GROUP BY Continent
HAVING SUM(population)>1e9
```

Mercredi 8 janvier 2025 – *Durée : 1 heure*

Devoir Surveillé n°2 (b) – *Corrigé*

Exercice n°1 Base de données

Nous utilisons une base de données nommée `world` qui comporte trois tables :
`city (Id, Name, CountryCode, Population)`
`country (Code, Name, Continent, Surface, Population, Capital)`
`countrylanguage (Id, CountryCode, Language, IsOfficial, Percentage)`

Les clés primaires sont les champs nommés `Id` pour les tables `city` et `countrylanguage`, et `Code` pour la table `country`. Les champs nommés `CountryCode` sont des clés étrangères en références à la clé primaire `Code` de la table `country`. Le champ `Capital` est une clé étrangère en référence à la clé primaire `Id` de `city`.

R1. Écrire la requête SQL qui permet d'obtenir le nombre de pays par continent.

Solution:

```
SELECT Continent,COUNT(Code)
FROM country
GROUP BY Continent
```

R2. Écrire la requête SQL qui permet d'obtenir la liste des continents de surface totale inférieure à 10 millions de km^2.

Solution:

```
SELECT Continent
FROM country
GROUP BY Continent
HAVING SUM(SurfaceArea)<10e6
```

R3. Écrire la requête SQL qui permet d'obtenir la capitale de la Norvège.

Solution:

```
SELECT city.Name
FROM city
JOIN country
ON Capital=Id
WHERE country.Name='Norvège'
```

R4. Écrire la requête SQL qui permet d'obtenir la liste des langues parlées à Lausanne.

Solution:

```
SELECT Language
FROM countrylanguage AS CL
JOIN country
JOIN city AS C
ON C.CountryCode=Code AND CL.CountryCode=Code
WHERE C.Name='Lausanne'
```

Exercice n°2 Comme un air de (presque) déjà vu

R5. Écrire une fonction d'en-tête `dim(L:list[list])->(int,int)` qui prend en argument une liste de listes `L` qui représente une matrice et qui renvoie le couple d'entiers `(m,n)` où `m` est la hauteur (nombre de lignes) et `n` la largeur (nombre de colonnes).

Solution:

```
def dim(L):
    m = len(L) # compte le nombre de sous-liste dans la liste, soit le nombre de lignes
    n = len(L[0]) # compte le nombre d'éléments dans la sous-liste L[0], soit le nombre de colonnes (toutes les lignes ont le même nombre d'éléments)
    return m,n
# OU
def dim(L):
    return (len(L),len(L[0]))
```

R6. Écrire une fonction d'en-tête `nulle(m:int,n:int)->list[list]` qui renvoie une liste de listes qui correspond à la matrice nulle de `m` lignes et `n` colonnes. *L'utilisation des fonctions de* ***numpy*** *est interdite.*

Solution:

```
def nulle(m,n):
    return [[0 for j in range(n)] for i in range(m)]
# OU
def nulle(m,n):
    return [[0]*n for i in range(m)]
# OU
def nulle (m,n):
    L=[]
    for i in range(m): # les lignes
        L_ligne=[] # liste vide d'initialisation d'une ligne
        for j in range(n): # on rempli la ligne i
            L_ligne.append(0)
    L.append(L_ligne)
    return L
```

R7. Écrire une fonction d'en-tête `copie(L:list[list])->list[list]` qui prend en argument une matrice `L` (liste de listes) et qui renvoie une copie de la matrice `L` totalement indépendante de `L`. Il est interdit d'utiliser le module `copy` et les fonctions `deepcopy` ou `deepcopy`, par contre, vous êtes invités à utiliser les deux fonctions écrites précédemment.

On réutilisera avantageusement des fonctions précédentes.

Solution:

```
def copie(L):
    h,w=dim(img)
    L2=nulle(h,w) # initialisation de l'image à 0
    for i in range(h):
```

```
        for j in range(w):
            L2[i][j]=L[i][j]
    return L2
```

R8. Écrire une fonction `imaxi(L:list)->int` qui prend en argument une liste `L` et qui renvoie la position de la première occurrence (=apparition) du maximum.

Solution:

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

R9. Écrire une fonction `fois_quatre(L:list[list])->list[list]` qui prend en argument une matrice `L` et qui renvoie une nouvelle matrice `L4` qui correspond à `L` dont tous les coefficients ont été multipliés par quatre.

On pourra réutiliser les fonctions précédentes.

Solution: question tout à fait accessible !

```
def fois_quatre(L):
    h,w=dim(L)
    L4=nulle(h,w) # tableau qui recevra l'image réduite
    for i in range(h): # parcours de img
        for j in range(w):
            L4[i][j]=L[i][j]*4
    return L4
```

R10. Écrire une fonction `reduc_quart(L:list[list])->list[list]` qui prend en argument une matrice `L` et qui renvoie une matrice quatre fois moins large en ne gardant que les colonnes de rang multiple de 4.

On pourra réutiliser les fonctions précédentes.

Solution: question tout à fait accessible !

```
def reduc_quart(L):
    h,w=dim(L)
    img_red=nulle(h,w//4) # tableau qui recevra l'image réduite
    for i in range(h): # parcours de img
        for j in range(w//4):
            img_red[i][j]=L[i][4*j]
    return img_red
```