Nom	
Prénom	
Groupe	

Note	16/20

Algorithmique
Matrices

SUP S2 EPITA

Examen B3

12 mars 2024

4,5/6

3,5/4

Consignes (à lire) :

- ☐ Vous devez répondre directement **sur ce sujet**.
  - — Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées**.
  - — Aucune réponse au crayon de papier ne sera corrigée.
- ☐ La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
- ☐ **Code :**
  - — Tout code doit être écrit dans le langage `Python` (pas de C, CAML, ALGO ou autre).
  - — **Tout code `Python` non indenté ne sera pas corrigé.**
  - — Tout ce dont vous avez besoin (types, fonctions, méthodes) est indiqué en **annexe**.
  - — Vos fonctions doivent impérativement respecter les exemples d'applications donnés.
  - — Vous pouvez également écrire vos propres fonctions, dans ce cas elles doivent être documentées (on doit savoir ce qu'elles font).
    Dans tous les cas, la dernière fonction écrite doit être celle qui répond à la question.
  - — Comme d'habitude l'optimisation est notée. Si vous écrivez des fonctions non optimisées, vous serez notés sur moins de points. [1]
- ☐ Durée : 45min

Annexes

Fonctions et méthodes autorisées

Vous pouvez utiliser la méthode **`append`**, la fonction **`len`** sur les listes ainsi que la fonction **`range`** :

```
>>> L = []

>>> for i in range(5):
        L.append(i)

>>> L
[0, 1, 2, 3, 4]

>>> len(L)
5

>>> for i in range(5, 10):
    L.append(i)
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Aucun opérateur n'est autorisé sur les listes (`+`, `*`, `==` ...).

1. Des fois, il vaut mieux moins de points que pas de points.

Exercice 1 (Double – *6 points*)

Écrire la fonction `check_mat(`L`)` avec `L` une liste de listes supposée non vide, cette fonction vérifie :

- que `L` est une matrice
- que chaque ligne est la "double" de la précédente (chaque élément est le double de celui à la même place de la ligne précédente)

Exemples d'applications :

```
>>> check_mat([[1, -1, 120, 10, -7], [2, -2, 240, 20, -14], [4, -4, 480, 40, -28]])
True
>>> check_mat([[1, -1, 120, 10, -7], [4, -4, 480, 40, -28], [2, -2, 240, 20, -14]])
False
>>> check_mat([[1, -1, 120, 10, -7], [2, -2, 240, 20, -14], [4, -4, 480]])
False
```

```
def check_mat(L):
    len0 = len(L)
    len1 = len(L[0])
    i = 1
    while len(L[i]) == len1 and i < len0:
        j = 0
        while L[i][j] == 2 * L[i-1][j] and j < len1:
            j += 1
        i += 1
    return i == len0 and j == len1
```

out of range
[-1,5]
dommage

Exercice 2 (Rotation – *4 points*)

Écrire la fonction `build_rotation(L, k)` avec :
- `L` une liste d'entiers de longueur $n > 0$
- `k` un entier tel que $0 < k < n$

qui construit et retourne une matrice telle que :
- la première ligne est la liste `L`
- chaque autre ligne contient les éléments de la ligne précédente décalés de `k` positions vers la droite
- la matrice est carrée

Par exemple avec la liste `L = [1, 2, 3, 4]`, les matrices résultats pour différentes valeurs de `k` sont :

1	2	3	4
4	1	2	3
3	4	1	2
2	3	4	1

FIGURE 1 – `k = 1`

1	2	3	4
3	4	1	2
1	2	3	4
3	4	1	2

FIGURE 2 – `k = 2`

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

FIGURE 3 – `k = 3`

```
def build_rotation(L,k):
    M = []
    lenL = len(L)
    M.append(L)
    for i in range(lenL-1):
        l = []
        for j in range(lenL):
            cur = M[i][(j-k)%lenL]
            l.append(cur)
        M.append(l)
    return M
```