Tableaux en compréhension
Tableaux de tableaux

APPLICATION

1) Écrire un programme qui affiche la liste des nombres pairs strictement inférieurs à 20
2) Écrire un programme qui affiche la liste du triple des nombres compris entre 1 et 10 inclus
3) Écrire un programme qui affiche dans un tableau les résultats de 20 tirages aléatoirement entre `P` pour "Pile" et `F` pour "Face" (Conseil : Utiliser la fonction `choice()`)
4) Reprendre les questions précédentes en construisant le tableau par compréhension

EXERCICES

Exercice 1

En utilisant la syntaxe des tableaux en compréhension :

1. Générer les entiers naturels de 1 à 100.
2. Générer les multiples de 5 inférieurs ou égaux à 100.
3. Générer une liste des entiers naturels de 1 à 100 dans laquelle les multiples de 5 seront remplacées par le caractère *

Exercice 2

En utilisant les fonctions `randint()` ou `choice()` du module `random` et la syntaxe des listes en compréhension :

1. Générer une liste de 20 entiers naturels aléatoires entre 0 et 255.
2. Générer 100 caractères au hasard parmi a,b,c

Exercice 3

Pour représenter des tableaux à double entrée (images, matrices...), on peut utiliser une liste de listes. On identifie ainsi un élément du tableau à l'aide de deux indexs.

Exemple :

```
tableau=[['A','b','c','d'],['E','f','g'],['I','j','k','m'],['N','o','p','q']]

print(tableau[0][0])
```

1. Quel est la lettre correspondant à `tableau[1][2]` ?
2. Quelle instruction permet d'accéder à la lettre `'m'` ?
3. Ajouter au tableau la ligne `['R','s','t','u']`.
4. Ajouter le caractère `h` à sa place.
5. Remplacer la caractère `N` par `n`.

Exercice 4
Générer en compréhension une liste de 10 listes contenant chacune 10 entiers binaires aléatoires (`0` ou `1`).

TD

Exercice 1 (en mode débranche)

Soit `m = [['a', 'b'], ['d', 'z']]`

a. Comment accède-t-on à la case en bas à gauche (celle qui vaut ici `'d'`) dans cette matrice `2x2` ? m=[1][0]

b. Compléter le code suivant pour qu'il affiche les 4 éléments de m : `'d a b z'`, dans cet ordre :

```
print(..., ..., ..., ...)
```

m[1][0] m[][] m[][] m[][]

c. Compléter le code suivant pour que la fonction affiche les dimension de `m`. Ces dimensions doivent être calculées à partir de `m` :

```
def affiche_dimensions(mat) :
    """ affiche les dimensions de mat"""
    print(f'{...} lignes et {...} colonnes')
# affiche : 2 lignes et 2 colonnes pour m
# 1 ligne et 3 colonnes pour [[2, 3, 5]]
```

len(mat) len(mat[0])

Exercice 2 (en mode branché)

Écrire un programme qui construise la matrice suivante :

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{pmatrix}$$

print ([[i for i in range (6)], [i for i in range(5,11)] ,[i for i in range(10,16)]])

Exercice 3 (en mode débranché type BAC)