Evaluation N°3 : Tuples - Listes – Dictionnaires - Grilles

Créer un fichier python que vous nommerez « ***eval3_NOM_prenom.py*** » à enregistrer dans le dossier « ***NSI_Première*** » dans « ***restitution de devoirs*** », ou à renvoyer sur « **Teams »** à la fin de l'évaluation.

Exercice 1 : Tuples

« **Rien ne se perd**, **rien ne se crée**, **tout se transforme.** »
Ecrire une fonction « **ordre(Tuple)** » qui prend en argument un tuple et qui remet cette célèbre phrase d'**Antoine Laurent de Lavoisier** (1743-1794) dans le bon ordre.
Les mots sont dans le tuple « **mots** » ci-dessous.

mots = ("rien", "Rien", "ne", "ne", "se", "se", "se", "perd", "crée", ",", ",", ".", "tout", "transforme")

```
*** Remote Interpreter Reinitialized ***
>>> ordre(mots)
Rien ne se perd, rien ne crée, tout se transforme.
```

Exercice 2 : Pizzas

Ces listes contiennent des « ingrédients » de pizzas....

liste1= ['jambon', 'mozzarella', 'sauce_tomate', 'champignons', 'poivrons']
liste2= ['chèvre', 'lardons', 'crème_fraîche', 'mozzarella', 'tomates', 'sauce_tomate', 'oignons']

1. Écrire la fonction « **nombreIngredients (listeA)** » qui renvoie le nombre d'ingrédient de la listeA.

```
>>>
*** Console de processus distant Réinitialisée ***
>>> nombreIngredient(liste1)
5
>>>
```

2. Écrire la fonction « **sommeIngredient(listeA,listeB)**» qui renvoie une liste nommée « liste3 » qui contient tous les ingrédients des listes 1 et 2.

```
>>>
*** Console de processus distant Réinitialisée ***
>>> sommeIngredient(liste1,liste2)
['jambon', 'mozzarella', 'sauce_tomate', 'champignons', 'poivrons', 'chèvre', 'lardons', 'crème_fraîche', 'mozzarella', 'tomates', 'sauce_tomate', 'oignons']
>>>
```

3. Écrire la fonction « **unionSansdoublon(listeA,listeB)** » qui renvoie une liste nommée « liste4» qui contient les ingrédients des listes A et B, mais qui ne contient plus le même ingrédient plusieurs fois.
   Aide : il faut chercher dans la liste B si l'ingrédient n'est pas dans la liste A et l'ajouter alors dans la liste4. (**if** ingredient **not in** listeA:)

```
>>>
*** Console de processus distant Réinitialisée ***
>>> unionSansDoublon(liste1,liste2)
['jambon', 'mozzarella', 'sauce_tomate', 'champignons', 'poivrons', 'chèvre', 'lardons', 'crème_fraîche', 'tomates', 'oignons']
>>>
```

Exercice 3 : Dictionnaires

Ecrire une fonction « prix_pizza(dico, pizza) » qui prend en argument un dictionnaire qui donne le prix des ingrédients et une liste d'ingrédients. Cette fonction doit renvoyer le prix de la pizza.

dico1={"jambon": 2.2, "mozzarella": 1.6, "sauce_tomate": 1.5, "champignons": 1.7, "poivrons": 1.5, "chèvre": 1.6, "lardons": 2.0, "crème_fraîche": 1.6, "tomates": 1.5, "oignons": 1.5}

pizza1= ["jambon", "sauce_tomate", "lardons", "crème_fraîche", "oignons"]

Exercice 4 : Grilles

On donne la grille suivante :
grille1=[[1,2,3,0],[4,1,2,0],[3,3,1,0],[0,0,0,0]]

1. Ecrire la fonction « **sommeLigne(grille)** » qui renvoie la grille après l'avoir modifiée de la manière suivante : le dernier élément de chaque ligne est la somme des éléments précédents qui composent cette ligne.

```
>>> sommeLigne(grille1)
[[1, 2, 3, 6], [4, 1, 2, 7], [3, 3, 1, 7], [0, 0, 0, 0]]
```

2. Ecrire la fonction « **sommeColonne(grille)** » qui renvoie la grille après l'avoir modifiée de la manière suivante : le dernier élément de chaque colonne est la somme des éléments précédents qui composent cette colonne.

```
>>> sommeColonne(grille1)
[[1, 2, 3, 6], [4, 1, 2, 7], [3, 3, 1, 7], [8, 6, 6, 0]]
```

3. Ecrire la fonction « **afficheGrilleFinale(grille)** » qui permet d'afficher la grille dans laquelle la dernière colonne est la somme des précédentes et la dernière ligne est également la somme des précédentes de manière lisible.

```
>>> afficheGrilleFinale(grille1)
1 2 3 6
4 1 2 7
3 3 1 7
8 6 6 20
```

Exercice 5 : Négatif

On donne la grille suivante :
grille2=[[0,0,0,0,0,0,1,0,0,0,0,0,0], [0,0,0,0,0,1,0,1,0,0,0,0,0], [0,0,0,0,1,0,0,0,1,0,0,0,0], [1,1,1,1,1,1,1,1,1,1,1,1,1], [0,0,1,0,0,0,0,0,0,0,1,0,0], [0,1,0,0,0,0,0,0,0,0,0,1,0]]

1. Ecrire la fonction **« afficheGrille(grille) »** qui permet d'afficher la grille de manière lisible sans espace entre les éléments d'une même ligne.

```
>>> afficheGrille(grille2)
0000001000000
0000010100000
0000100010000
0001111111000
0010000000100
0100000000010
```

2. Ecrire la fonction « negatif(grille) » qui renvoie la grille après avoir inversé les 0 et les 1.

```
>>> negatif(grille2)
[[1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1],
 [1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1],
 [1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1],
 [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1],
 [1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1],
 [1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1]]
```

3. Tester cette dernière fonction avec la fonction afficheGrille(grille).

```
>>> afficheGrille(negatif(grille2))
1111110111111
1111101011111
1111011101111
1110000000111
1101111111011
1011111111101
```