Initiation à la programmation – TP 11 du 10 novembre

Objectif : Manipuler des grilles. Il pourra être utile de mettre dans un fichier particulier les fonctions sur les grilles qui pourront resservir plusieurs fois.

Exercice 1 : Premières grilles

1. Ecrire la fonction `cree_grille(n,p)` qui crée une grille à n lignes et p colonnes remplie de 0; elle doit créer et renvoyer la grille comme sur l'exemple ci-dessous :

```
>>>cree_grille(3,4)
[[0,0,0,0],[0,0,0,0],[0,0,0,0]]
```

2. Écrire la fonction `zero(magrille)` qui renvoie True s'il n'y a que des 0 dans la grille .
3. Écrire la fonction `affiche(magrille)` qui étant donnée une grille de taille quelconque affiche "joliment" cette grille (voir exemple d'affichage ci-dessous). Il ne doit plus y avoir ni virgules ni crochets dans l'affichage.
4. Écrire la fonction `compter(magrille ,val)` qui renvoie combien il y a d'éléments égaux à val dans la grille .
5. Écrire une fonction qui prend en paramètre une grille de nombres et renvoie la somme de tous ces nombres. On fera deux versions : une première version en utilisant un parcours avec une double boucle par indice et une version en utilisant un parcours des lignes.
6. On appellera "case" la donnée d'un couple de coordonnées `[i,j]` représenté par une liste à deux éléments correspondant aux coordonnées d'un élément de la grille (i ème ligne, j ème colonne). Par exemple l'élément en haut à gauche est la case `[0,0]` et celui en bas à droite d'une grille de taille n est `[n-1,n-1]`.
   Écrire la fonction `CasesZero(grille)` qui envoie la liste des cases données par leur couple de coordonnées qui contiennent un 0.

```
>>> ex=creeGrille(5,4)
>>> ex
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
>>> zero(ex)
True

>>> ex=[[5, 10, 4, 7], [7, 8, 2, 4], [0, 0, 9, 8], [1, 9, 6, 3], [8, 0, 10, 6]]
>>> affiche(ex)
5 10 4 7
7 8 2 4
0 0 9 8
1 9 6 3
8 0 10 6


>>> zero(ex)
```

```
False
>>>compter(ex,8)
3

>>>grille_ex
[[0,1,8,9,4],[9,6,5,8,0],[2,4,3,2,1]]
>>>somme(grille_ex)
62
>>> CasesZero(ex)
[[2, 0], [2, 1], [4, 1]]
```

Exercice 2 : Mots croisés

On veut travailler avec des « grilles » de mots croisés : une grille sera une liste de listes. Une case noire sera représentée par une étoile '*'.

La grille suivante pourra servir d'exemple :

```
grille = [ ['P','Y','T','H','O','N'] , ['R','*','A','U','M','A'],
 ['O','M','B','R','E','*'], ['G','A','L','E','T','S'], ['*','T','E','*','S','U']]
```

Grâce à la fonction `affiche` vue en TD, on peut voir que la grille précédente ressemble à :

```
P Y T H O N
R * A U M A
O M B R E *
G A L E T S
* T E * S U
```

1. Écrire une fonction `uneLettreParCase(magrille)` qui prend en paramètre une grille remplie de mots et qui renvoie `True` si chaque mot de la grille contient une seule lettre (c'est-à-dire est de longueur 1) ; `False` sinon.

```
>>> uneLettreParCase(grille)
True
>>> uneLettreParCase([['A','B','C'],['*','DEF','G']])
False
```

2. Écrire une fonction `ligne(magrille,i)` qui prend en paramètre une grille remplie de lettres et un numéro de ligne `i` ; et qui renvoie la `i`-ième ligne sous forme de chaîne de caractères, étoiles '*' compris.

```
>>> ligne(grille,2)
'OMBRE*'
>>> ligne(grille,4)
'*TE*SU'
```

3. Écrire une fonction `recupereLignes(magrille)` qui prend en paramètre une grille remplie de lettres ; et qui renvoie la liste de toutes les lignes, chacune d'entre elles étant une chaîne de caractères, étoiles '*' compris.

```
>>> recupereLignes(grille)
['PYTHON', 'R*AUMA', 'OMBRE*', 'GALETS', '*TE*SU']
```

4. Écrire une fonction `recupereColonnes(magrille)` qui est l'analogue de la question précédente, mais pour les colonnes : la fonction doit renvoyer la liste de toutes les colonnes, chacune d'entre elles étant une chaîne de caractères, étoiles * compris.

```
>>> recupereColonnes(grille)
['PROG*', 'Y*MAT', 'TABLE', 'HURE*', 'OMETS', 'NA*SU']
```

5. Écrire une fonction `enleveEtoiles(mot)` qui prend en paramètre une chaîne de caractères `mot` et qui renvoie la liste des sous-mots décomposés selon les étoiles `'*'` compris dans `mot`.

```
>>> enleveEtoiles("BONJOUR*CA*VA")
['BONJOUR', 'CA', 'VA']
>>> enleveEtoiles("*MOI*CA*VA*T")
['', 'MOI', 'CA', 'VA', 'T']
```

6. Écrire une fonction `ecreme(liste)` qui prend en paramètre une liste de mots et qui renvoie la même liste de mots mais dans laquelle nous avons enlevé tout mot de longueur 0 ou 1.

```
>>> ecreme(['UN','EXEMPLE','','A','TRAITER','!'])
['UN', 'EXEMPLE', 'TRAITER']
```

7. Écrire une fonction `listeMots(magrille)` qui affiche joliment tous les mots compris dans la liste de mots croisés.

   Un exemple :

```
>>> listeMots(grille)
Horizontalement :
1. PYTHON. 2. AUMA. 3. OMBRE. 4. GALETE. 5. TE. SU.
Verticalement :
1. PROG. 2. MAT. 3. TABLE. 4. HURE. 5. OMETS. 6. NA. SU.
```

8. Écrire une fonction `anagrammes(magrille)` qui affiche une grille vide de mots croisés, correspondant à la grille `magrille`, puis qui affiche des définitions des mots de la grille sous forme d'anagrammes. (Une anagramme d'un mot `m` est juste un mot différent de `m` obtenu en mélangeant les lettres de `m`).

   On pourra utiliser la fonction `shuffle` de la librairie `python` qui mélange les cases d'une liste.

```
>>> anagrammes(grille)
  123456

 _______
1|       |
2| #     |
3|      #|
4|       |
5|#  #   |

Horizontalement :
1. OYNTPH. 2. AUAM. 3. EMBRO. 4. EAGLET. 5. ET. US.
```

```
Verticalement :
1. RPOG. 2. TAM. 3. TLABE. 4. UREH. 5. EOTSM. 6. AN. US.
```

Exercice 3 : Des dessins

On veut travailler avec des « grilles » : une grille sera une liste de listes. Les grilles seront de tailles quelconques. On va utiliser des grilles pour représenter des dessins pixelisés.

La grille coeur pourra servir d'exemple avec coeur =[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0], [0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0], [0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0], [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0], [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 1, 0, 0, 0, 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, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

1. Écrire la fonction creGrille(n,p) qui renvoie une grille à n lignes et p colonnes contenant des 0 dans toutes les cases.

2. Écrire la fonction affiche(dessin) qui étant donné une grille représentant un dessin (c'est à dire une grille constituée de 0 et de 1) affiche "joliment" ce dessin en mettant des espaces pour les 0 et des * pour les 1.

```
>>> affiche(coeur)

      * *       * *
    *     *   *     *
  *         *         *
  *                   *
    *               *
      *           *
        *       *
          *   *
            *
```

3. Écrire la fonction negatif(dessin) qui inverse les 0 et les 1 dans la grille. Il faut modifier la grille ou en renvoyer une nouvelle(il ne suffit pas de modifier l'affichage, il faut travailler sur la grille, la fonction affiche n'est là que pour mieux voir ce qui se passe).

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

>>> affiche(negatif(coeur))
* * * * * * * * * * * * *
* * *     * * *     * * *
* *   * *   *   * *   * *
*   * * * *   * * * *   *
*   * * * * * * * * *   *
* *   * * * * * * *   * *
* * *   * * * * *   * * *
* * * *   * * *   * * * *
* * * * *   *   * * * * *
```

```
* * * * * *   * * * * * *
* * * * * * * * * * * * *
```

4. On veut effectuer une symétrie horizontale des dessins.
   (a) Écrire la fonction echange(g,i,j) qui échange les lignes i et j dans la grille .
   (b) Écrire la fonction symHorizontale(grille) qui effectue une symétrie horizontale sur la grille dessin (on échange première et dernière ligne, deuxième et avant-dernière ligne etc...). On utilisera la fonction précédente.

```
>>> affiche(symHorizontale(coeur))

          *
        *   *
      *       *
    *           *
  *               *
*                   *
*         *         *
  *     *   *     *
    * *       * *
```

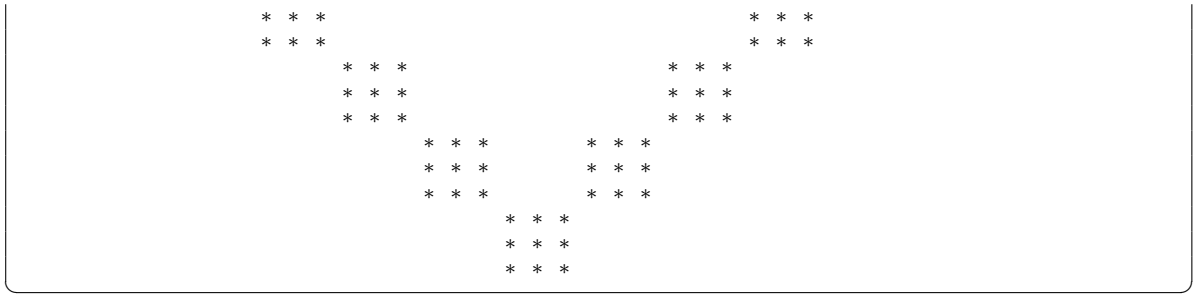
5. On veut "zoomer" les dessins
   (a) Ecrire la fonction zoomListe(l,k) qui a pour paramètres une liste et un entier et qui renvoie une liste où tous les éléments de l sont écrits k fois de suite.
   (b) Ecrire la fonction zoom(dessin,k) qui renvoie une grille où toutes les lignes sont zoomées k fois et répétées k fois.

```
>>> zoomListe(l,2)
[0, 0, 1, 1, 0, 0, 1, 1, 1, 1]
>>> grille=[[0,1,0,1,1],[0,1,0,1,1],[0,1,0,1,1]]
>>> grille=[[1,1],[0,0]]
>>> zoom(grille,3)
[[1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0,


>>> affiche(zoom (coeur, 3))



                * * * * * *                 * * * * * *
                * * * * * *                 * * * * * *
                * * * * * *                 * * * * * *
          * * *           * * *       * * *           * * *
          * * *           * * *       * * *           * * *
          * * *           * * *       * * *           * * *
    * * *                       * * *                       * * *
    * * *                       * * *                       * * *
    * * *                       * * *                       * * *
    * * *                                                   * * *
    * * *                                                   * * *
    * * *                                                   * * *
          * * *                                       * * *
          * * *                                       * * *
          * * *                                       * * *
                * * *                           * * *
```



Exercice 4 - Jeu de morpion

Écrire un petit jeu de morpion en mode console. Le jeu doit se dérouler de la manière suivante :

```
>>> jeu()
0 0 0
0 0 0
0 0 0
tour du joueur 2
ligne? 1
colonne? 1
0 0 0
0 2 0
0 0 0
tour du joueur 1
ligne? 2
colonne? 2
0 0 0
0 2 0
0 0 1
tour du joueur 2
ligne? 0
colonne? 1
0 2 0
0 2 0
0 0 1
tour du joueur 1
ligne? 2
colonne? 1
0 2 0
0 2 0
0 1 1
tour du joueur 2
ligne? 0
colonne? 2
0 2 2
0 2 0
0 1 1
tour du joueur 1
ligne? 2
colonne? 0
0 2 2
```

```
0 2 0
1 1 1
bravo joueur  1 vous gagnez!!
```

Les joueurs 1 et 2 jouent une case chacun leur tour en saisissant ligne et colonne ; leur numéro est mis alors à la bonne position dans la grille. Un joueur gagne s'il parvient à aligner son symbole 3 fois en ligne, colonne ou diagonale. Le jeu s'arrête au bout de 9 coups si personne n'a gagné (la grille est alors pleine).

Le jeu se joue avec une grille de 3 sur 3.
On commencera par une version simple et sans test sur les saisies puis on rajoutera des tests pour vérifier que le choix du joueur correspond bien à une case et que cette case est effectivement vide !

On conseille d'écrire quelques fonctions comme par exemple des fonctions de saisie et une fonction qui teste si une grille est gagnante.

On pourra ensuite ajouter les améliorations de son choix et/ou faire une version dans laquelle un joueur joue contre l'ordinateur !