Lycée Jean Moulin **Draguignan**
NSI 1re Année .....

Les Tableaux

Représentation d'un tableau

Définition :
En informatique, un tableau est une structure de données représentant une séquence finie d'éléments auxquels on peut accéder efficacement par leur position, ou indice

Voici un tableau de nombre :

4	9	2
3	5	7
8	1	6

En Python il n'existe pas de structure de donnée particulière pour représenter un tableau, il faudra utiliser des listes de liste:

```
# représente le tableau ci-dessus
t=[[4,9,2],[3,5,7],[8,1,6]]
```

Question 1:
Comment accéder à la valeur 5 de ce tableau?
.....

Question 2:
Que retourne cette instruction : `len(t)==len(t[1])`?
.....

Exercice 1 :
Créer une fonction `cree_tableau(n,p)` qui renvoie un tableau de n lignes et p colonnes contenant des nombres entiers aléatoirement choisit entre 0 et 100

On utilisera la création de liste en compréhension : `t=[[valeur for...]for ...]`

```
from random import randint
def cree_tableau(n,p):



_
```

Exercice 2 :

En utilisant un tableau carré (nombre de lignes = nombre de colonnes), créé avec la fonction précédente, écrire une fonction `somme_ligne(t,k)` qui prend en paramètre le tableau et un n° de ligne et qui retourne la somme des valeurs de cette ligne

```
def somme_ligne(t,k):




-
```

Exercice 3 :

Même question mais pour les colonnes : `somme_colonne(t,k)`

```
def somme_colonne(t,k):




-
```

Exercice 4 :

Même question mais pour les deux diagonales de ce tableau : `somme_diagonales(t,k)`
le retour se fera sous forme d'un tuple

```
def somme_diagonales(t,k):




-
```

Question 3:

Que retourne ces fonctions pour ce tableau: .....

4	9	2
3	5	7
8	1	6

Les carrés magiques

Un carré magique d'ordre n est un tableau 2d, composé de n lignes et n colonnes, chaque case du tableau contenant un nombre strictement positif.
Les nombres sont disposés de sorte que :
• la somme des éléments d'une ligne est toujours égale à la constante magique S ;
• la somme des éléments d'une colonne est toujours égale à la constante magique S ;
• la somme des éléments d'une diagonale est toujours égale à la constante magique S.
De plus, un carré magique contenant n × n nombres est dit normal s'il contient tout les nombres de 1 à n^2.

On s'intéresse à une méthode pour générer des carrés magiques appelée **méthode Siamoise** (de Simon de la Loubère qui l'a découverte en 1688 alors qu'il revenait d'une ambassade au Siam). Cette méthode ne fonctionne que sur des carrés d'une certaine taille (n doit être impair..) et dans certaines conditions.

Description de la méthode:

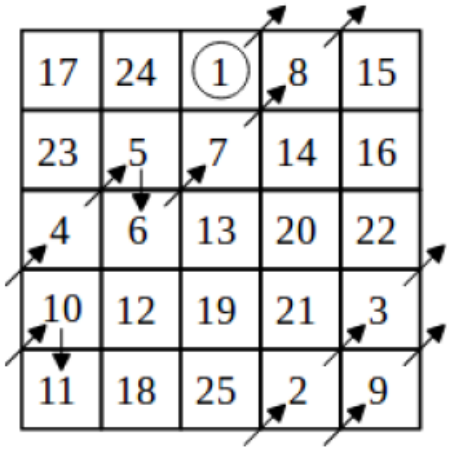
• On place le 1 au milieu de la 1ère ligne

• Se décaler d'une case en haut à droite, y placer le 2, et ainsi de suite pour le 3, 4, etc.

• Si on sort du carré on revient de l'autre côté.

• Si la prochaine case est occupée, on se déplace vers le bas à la place.

Illustration :



? Exercice 5 :

Expliquer ce que font ces fonctions, l'objectif final étant de réaliser un carré magique avec la méthode siamoise...
Écrire leur spécifications (type d'entrée et de sortie)

```
def cree_carre(n):
    """

    """
    return [[0 for i in range(n)] for j in range(n)]
```

.....
.....
.....
.....

```python
def affiche(t):
    """

    """
    for el in t:
        print(el)
```

.....
.....
.....
.....

```python
def suivant(k, n):
    """

    """
    if k>=n-1:
        return 0
    else:
        return k+1
```

.....
.....
.....
.....

```python
def precedent(k, n):
    """



    """
    if k==0:
        return n-1
    else:
        return k-1
```

.....
.....
.....
.....

```
def carre(n):
    """

    """
    assert(n%2!=0),"le carré doit être d\'ordre impair"
    t=cree_carre(n)
    i0=0
    j0=n//2
    t[i0][j0]=1
    m=n**2
    for valeur in range(2,m+1):
        isucc=precedent(i0,n)
        jsucc=suivant(j0,n)
        if t[isucc][jsucc]==0:
            i0,j0=isucc,jsucc
        else:
            i0=suivant(i0,n)
        t[i0][j0]=valeur
    affiche(t)
```

.....
.....
.....
.....
.....
.....
.....
.....

Exercice 6 :

Réaliser le programme et créer une nouvelle fonction `carre_verif(t)` qui vérifie que le carré est bien magique!!

```
def carre_verif(t):




-
```

Pour les courageux....

Vous trouverez sur le net d'autres façons de créer des carrés magiques; implémenter l'une d'entre elle en Python:

```
--
--
```