EXAMEN SEMESTRE 2

Matière : INFORMATIQUE
Classes : 1ère Année **Durée :** 2 h
Préparation : MP, PC, PT et BG

EXERCICE 1

On considère un tableau carré de **n** lignes et **n** colonnes.

Un carré magique de taille **n** est un arrangement en carré de n^2 valeurs. Ces nombres sont disposés de manière à ce que leurs sommes sur chaque rangée, sur chaque colonne et sur chaque diagonale soient égales. Un carré magique est dit normal s'il est rempli avec les nombres entiers compris entre **1** et n^2 (inclus). Le dessin suivant représente un carré magique de taille 5 :

15	8	1	24	17
16	14	7	5	23
22	20	13	6	4
3	21	19	12	10
9	2	25	18	11

On donne $n * S = \sum_{i=1}^{n^2} i \quad \Rightarrow S = \frac{n(n^2+1)}{2}$

1) Ecrire une fonction python **saisie()** qui permet de saisir un entier **n** impair.
2) Écrire une fonction python **afficherCarreMagique(T,n)** qui permet de saisir et d'afficher un tableau carré d'ordre **n**
3) Écrire une fonction python **sommeLigne(T,n,l)** qui calcule et retourne la somme des valeurs contenues dans la **l**ème ligne d'un carré.
4) Écrire une fonction python **sommeColonne(T,n,col)** qui calcule et retourne la somme des valeurs contenues dans la **col**ème colonne d'un carré.
5) Écrire une fonction python **sommeDiag1(T,n)** qui calcule et retourne la somme des valeurs contenues dans la première diagonale d'un carré.
6) Écrire une fonction python **sommeDiag2(T,n)** qui calcule et retourne la somme des valeurs contenues dans la deuxième diagonale d'un carré.

7) Écrire une fonction python **testCarreMagique(T,n)** qui teste si un carré est bien un **carré magique normal**.

EXERCICE 2

Ecrire une fonction Python nommée **sequence**, ayant comme paramètres un tableau **T** (à une dimension) et sa taille **N**, permettant de :

- déterminer **Nbs**, le nombre de séquences strictement croissantes de ce tableau,
- déterminer le nombre d'éléments **Ta**, de la séquence strictement croissante contenant le plus grand nombre d'éléments,
- placer, à partir de **T**, tous les éléments positifs ou nuls au début d'un tableau **TR** et tous les éléments négatifs à sa fin,
- retourner une liste contenant **Nbs**, **Ta** et **TR**.

Exemple : Soit T un tableau de 15 éléments :

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*
T =	1	2	5	3	12	25	13	-8	4	7	24	28	32	-14	-11

Les séquences strictement croissantes sont : < 1; 2; 5 >; < 3; 12; 25 >; < 13 >; < -8 ; 4; 7; 24; 28; 32 >;< -14; -11 >.

Le nombre de séquence est : Nbs=5

La séquence contenant le plus grand nombre d'éléments est : < -8 ;4; 7; 24; 28; 32 >.

Son nombre d'éléments : Ta=6

Le tableau (TR) est :

TR =	1	2	5	3	12	25	13	4	7	24	28	32	-8	-14	-11

EXERCICE 3

Notre but est d'écrire un programme Python qui lit un fichier **"données.txt"** contenant une suite de **n** entiers appartenant à l'intervalle **[0,20]**, les triés en utilisant une méthode de tri, et affiche le résultat du tri.

1) Ecrire une fonction python qui ouvre le fichier **"données.txt"**, lit les valeurs qu'il contient et les écrits dans un tableau **T**. Cette fonction renvoie le nombre **n** d'éléments écrits dans **T**
2) Ecrire une fonction **trier_tableau** qui effectue le tri par ordre croissant des **n** éléments

de **T** et les écrit dans un tableau **T1** selon un l'algorithme de tri par **sélection**.

3) Ecrire une fonction **saisie_tableau** permettant de sauvegarder les éléments triés du tableau **T1** dans un nouveau fichier **"données_tirées.txt"**

Rappel:

*Sur un **tableau** de n éléments (numérotés de 1 à n), le principe du tri par sélection est le suivant :*

- *rechercher **le plus petit élément** du tableau, et l'échanger avec l'élément d'indice 1 ;*
- *rechercher le second plus petit élément du tableau, et l'échanger avec l'élément d'indice 2 ;*
- *continuer de cette façon jusqu'à ce que le tableau soit entièrement trié.*

EXERCICE 4

Il s'agit d'écrire un programme permettant d'obtenir des figures illustrant la construction graphique des termes d'une suite récurrente définie par : $\begin{cases} u_0 \\ u_{n+1} = f(u_n) \end{cases}$

La fonction f est définie sur **[0;1]** par :

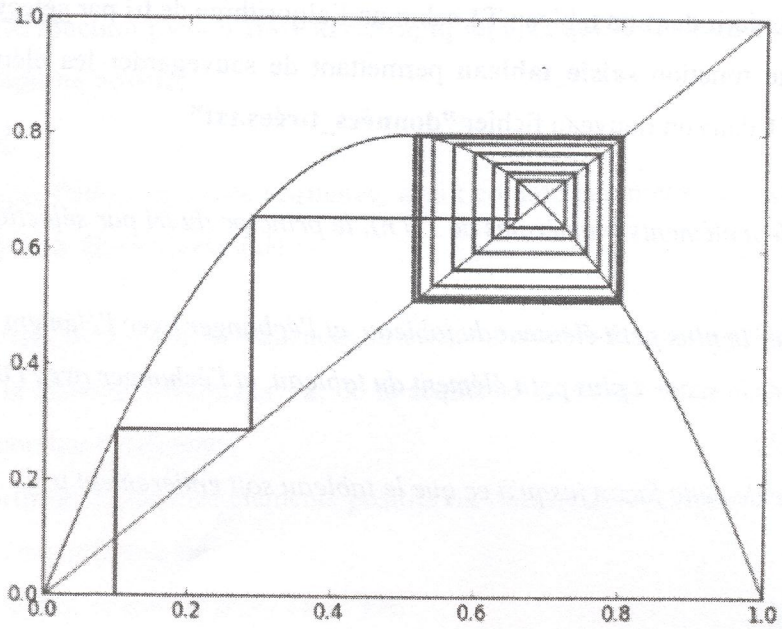
$$f(x) = a.x.(1-x)$$ avec **a** $\in$**]0;4]** et **u0** $\in$**]0;1[**

1) Ecrire une fonction python saisie qui demande à l'utilisateur les valeurs de **a**, de **u0**, et le nombre maximal d'itérations noté **imax**(compris entre 1 et 20).
2) Ecrire la fonction **f** qui prend en argument **x** et renvoie la valeur de **f(x)**.
3) Construire la liste ***x*** des abscisses variant de 0 à 1 avec un pas de 0.02 puis la liste **y** des ordonnées correspondantes $y = f(x)$.
4) Tracer sur une même figure la courbe représentant **f** sur l'intervalle **[0 ; 1]** et la bissectrice (droite d'équation **y = x**).
5) Pour le tracé des itérations, on crée deux nouvelles listes **x** et **y** contenant respectivement les abscisses et les ordonnées des points à placer. Initialement, les listes contiennent les coordonnées du premier point **(u0;0)**, puis, à l'aide d'une boucle, on complétera les deux listes avec les coordonnées des points **(x;y)** **et** **(y;y)** avec **y = f(x).**

   Construire les listes **x** et **y** et tracer ainsi la courbe des itérations

   Indication : Vous pouvez utiliser l'indiçage négatif.

On obtiendra des figures du type suivant : (exemple pour a=3.2, u0=0.1 et imax=10)



6) Ecrire dans un fichier **"data.txt"** les coordonnées x et y des deux listes précédentes. Chaque ligne du fichier contiendra deux nombres x et y séparés par une tabulation ("\t").