REPUBLIQUE TUNISIENNE
Ministère de l'enseignement Supérieure, de la recherche scientifique

Institut Préparatoire aux Etudes d'Ingénieurs de Sfax

الجمهورية التونسية
وزارة التعليم العالي
والبحث العلمي

المعهد التحضيري للدراسات الهندسية
بصفاقس

Examen d'informatique

2ème semestre AU : 2022 – 2023

Date : 17 Mai 2023 ;	**Durée :** 2H ;	**section :** BG1 ;	**Nombre de page :** 4

Le langage de programmation sera obligatoirement ***Python****.*

Exercice 1 : Les montres multisport

On s'intéresse dans cet exercice aux montres multisport, en développement croissant depuis les dix dernières années. Ces montres permettent aux sportifs amateurs ou professionnels d'enregistrer diverses données physiques et physiologiques. Elles permettent maintenant de connaître sa position GPS, d'enregistrer un trajet et tous les paramètres associés, de suivre un trajet prédéfini.

Dans la montre, un module électronique capte les signaux GPS à l'aide d'une antenne et décode ces signaux pour générer plusieurs trames enregistrées dans un fichier de la forme suivante :

trames.txt

```
GPS 092828   4754.18982N 00154.69147E    2.61
GAL 092830   4784.38253N 00234.67847E    3.69
GPS 092928   5854.49014N 00154.69147E    7.31
GAL 093028   7433.58900N 00154.69147E    5.60
GPS 100223   7551.69988N 00154.69147E    1.69
GLO 100533   8100.62988N 00154.69147E    4.51
GPS 101059   8914.70988N 00154.69147E    0.01
```

Où chaque ligne représente une trame qui contient les informations suivantes :

Par exemple pour la trame1 = 'GPS 092828 4754.18982N 00154.69147E 2.61'

- GPS : Type de trame (GPS , GAL , GLO)
- 092828 : Horaire UTC - 09 h 28 min 28 s
- 4754.18982N Latitude : 47°54, 18982 , N : latitude Nord
- 00154.69147E Longitude 001°54,69147 , E : longitude Est
- 2.61 : Coefficient de précision

Ces informations sont séparées par une tabulation ' \t '.

Travail demandé :

1. Ecrire une fonction lireFichier(nomF) qui prend en paramètre un nom de fichier et qui renvoie une liste L où chaque élément est une ligne du fichier (sans \n)

2. Ecrire une fonction type_trame(trame) qui prend en paramètre une chaine représentant une trame et qui renvoie le type de trame.

Indication : utiliser la fonction ch.split(séparateur) qui permet de séparer une chaine ch autour d'un séparateur et le résultat est une liste des éléments séparés.

Exemple :

```
>>> trame = "GPS   092828   4754.18982N   00154.69147E   2.61"
>>> type = type_trame(trame)
>>> print('type de trame = ',type)
GPS
```

3. Ecrire une fonction test_precision(trame) qui prend en paramètre une chaine représentant une trame et qui renvoie True si la précision est inférieure à 5, False sinon.

Exemple :

```
>>> trame = "GPS   092828   4754.18982N   00154.69147E   2.61"
>>> test =  test_precision (trame)
>>> print(test)
True
```

Soit **d_trame** un dictionnaire dont les éléments sont des associations entre une clé qui est le type de trame (GPS , GAL , GOL, ...) et une valeur qui représente une liste des trames (chaine de caractère) de ce type de trame.

4. Ecrire une fonction dict_trame(L_trames) qui prend en paramètre une liste de trames et qui renvoie un dictionnaire **d_trame**.

5. Ecrire une fonction enregistrer_trames(d_trame) qui prend en paramètre un dictionnaire **d_trame** et qui crée pour chaque type de trame un fichier nommé le type de trame et enregistre les trames de même type dans les lignes de ce fichier.

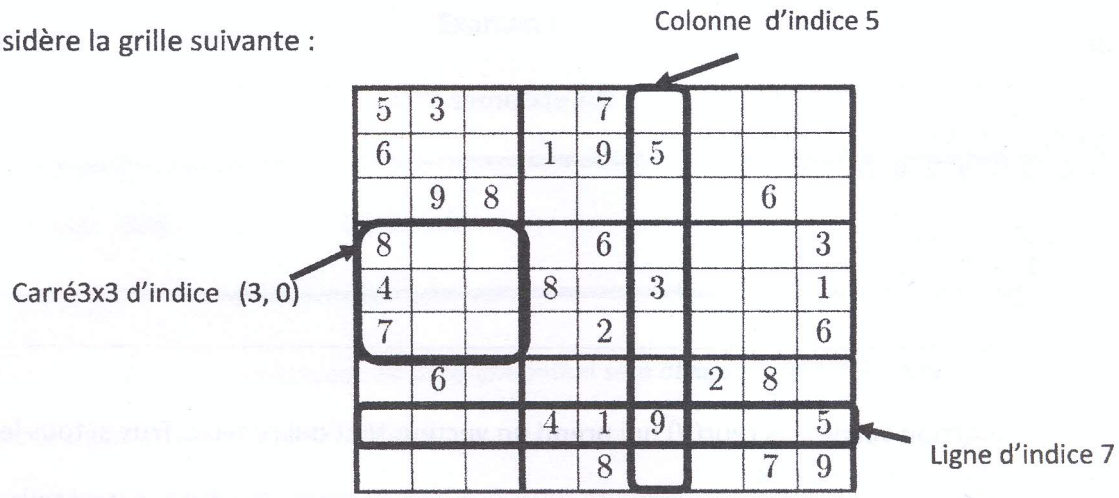
6. Ecrire un programme principal qui permet de :
   - remplir le fichier nommé trames.txt dans une liste L_trames.
   - remplir un dictionnaire d_trame à partir de la liste L_trames.
   - créer pour chaque type de trame un fichier contenant ses trames

Exercice 2 : Soduku

Le but de cet exercice est de vérifier si une grille de soduku classique ne contient aucune incohérence.

Rappelons qu'un sudoku est une grille de 9 lignes et de 9 colonnes, dont le but du jeu est de remplir cette grille avec une série de chiffres de 1 à 9, qui ne se trouvent jamais plus d'une fois sur une même ligne, dans une même colonne ou dans un même carré de 3 × 3. Quelques chiffres sont déjà disposés dans la grille, ce qui autorise une résolution progressive du problème complet.

On considère la grille suivante :



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

On peut voir une grille de sudoku comme un **tableau numpy** de 9 lignes et 9 colonnes où :

- Chaque carré de 3x3 est repéré par son premier élément le plus haut à gauche.
- Chaque case inconnue est remplacée par 0 :

Travail demandé :

1. Ecrire une fonction genererGrille(N) qui prend en paramètre un entier N et qui crée une matrice de 9 x 9 avec N entiers aléatoires compris entre 1 et 9 et le reste avec des zéros.

Indication : utiliser les fonctions suivantes :

- La fonction zeros((n,m)) du module numpy pour générer une matrice nulle de n x m zéro.
- La fonction randint(a,b) du module random pour générer un entier aléatoire entre a et b.

Exemple :

```
>>> G1 = genererGrille(30)

>>> print(G1)

[[0 0 0 0 0 7 0 0 0]
 [0 9 0 0 1 4 0 0 0]
 [0 5 0 0 0 0 0 7 0]
 [9 6 0 5 0 0 2 0 0]
 [0 0 0 9 0 0 0 0 0]
 [0 4 0 0 0 0 3 5 3]
 [0 8 0 0 2 0 0 0 3]
 [0 0 1 6 0 0 0 4 2]
 [0 7 0 0 0 0 0 0 0]]
```

2. Ecrire une fonction sansZero(V) qui prend en paramètre un vecteur V et qui renvoie un nouveau vecteur en supprimant tous les zéros de V.

Indication : utiliser les fonctions suivantes :

- La fonction list(V) qui convertit un vecteur en une liste
- La fonction np.array(L) qui convertit une liste en un vecteur

Exemple :

```
>>> V = G1[3,:]
>>> print("V = ",V)
V =  [9 6 0 5 0 0 2 0 0]

>>> V1 = sansZero(V)
>>> print("V1 = ",V1)
V1 =  [9 6 5 2]
```

3. Ecrire une fonction valide_vecteur(V) qui prend un vecteur V et qui renvoie True si tous les éléments de V (privé de 0) sont distincts.

Exemple :

```
>>> V =  [9 6 0 5 0 0 2 0 0]
>>> print( valide_vecteur(V)  )
True
```

4. Ecrire une fonction valide_carre(G , i ,j) qui prend en paramètre la grille G (une matrice 9x9) et deux entiers i , j représentant la position du carre dans G. Cette fonction renvoie True si tous les éléments du carré (privé de 0 sont distinct), False sinon.

Exemple :

```
>>> C1 = G_soduku[3 : 6 , 0 : 3]
>>> print(C1)
[[9 6 0]
 [0 0 0]
 [0 4 0]]

>>> print( valide_carre(G , 3 , 0 ) )
True
```

5. Ecrire une fonction valideTousLignes(G) qui prend en paramètre une grille G et renvoie True si toutes les lignes de G sont valides.
6. Ecrire une fonction valideTousColonnes(G) qui prend en paramètre une grille G et renvoie True si toutes les colonnes de G sont valides.
7. Ecrire une fonction valideTousCarres(G) qui prend en paramètre une grille G et renvoie True si tous les carrés de G sont valides.
8. Ecrire une fonction valideGrilleSoduku(G) qui prend en paramètres une grille G et renvoie True si la grille de soduku est valide. On rappelle qu'une grille est valide si toutes ses lignes sont valides et toutes ses colonnes sont valides et tous ses carrés sont valides.