

DEVOIR DE SYNTHESE SEMESTRE 2**Matière : INFORMATIQUE****Classes : MP1, PC1, PT1 - Durée : 2 h**

Les fonctions demandées doivent être écrites en langage **Python**. On sera très attentif à la rédaction et notamment à l'indentation du code.

Exercice 1

Un triangle **magique** est un arrangement de nombres entiers de 1 à n sur les côtés d'un triangle avec le même nombre d'entiers de chaque côté, appelé l'ordre du triangle, de sorte que la somme des nombres entiers sur chaque côté est une constante.

Deux exemples de triangles magiques :

$L = [1, 5, 3, 4, 2, 6]$, Ordre = 3 avec $n = 6$	$L = [2, 5, 9, 1, 6, 7, 3, 4, 8]$, Ordre = 4 avec $n = 9$
<pre> 1 6 5 2 4 3 </pre>	<pre> 2 8 5 4 9 3 7 6 1 </pre>
La somme des nombres entiers sur chaque côté du triangle est une constante :	
Côté 1 : $1 + 5 + 3 = 9$ Côté 2 : $3 + 4 + 2 = 9$ Côté 3 : $2 + 6 + 1 = 9$	Côté 1 : $2 + 5 + 9 + 1 = 17$ Côté 2 : $1 + 6 + 7 + 3 = 17$ Côté 3 : $3 + 4 + 8 + 2 = 17$

Nota : le début de la liste L constitue le côté 1 du triangle.

Ecrire une fonction **magique(L)** qui prend en argument une liste L de nombres et renvoie :

- **False** si le nombre d'éléments de L n'est pas un multiple de 3,
- **False** si les éléments de L n'appartiennent pas à l'intervalle $[1, \text{len}(L)]$,
- **True** si L forme un triangle magique.

Exercice 2

Le **tableau de Polybe** est une technique ancienne de chiffrement de mots qui consiste à ordonner les lettres de l'alphabet en ordre alphabétique dans un tableau dont chaque ligne et chaque colonne sont numérotées conformément au tableau suivant :

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I	J
3	K	L	M	N	O
4	P	Q	R	S	T
5	U	V	W	X	Y
6	Z				

Pour chiffrer un mot, il faut trouver la paire de numéros correspondants à chaque lettre. Le premier chiffre est le numéro de la ligne et le second celui de la colonne. Par exemple, le mot « **IPEIS** » est ainsi chiffré : **2441152444**. Pour déchiffrer un mot, il suffit d'effectuer la méthode inverse.

Soit **D** un dictionnaire composé par 26 clés qui correspondent aux 26 lettres majuscules de l'alphabet, tel que pour chaque clé est associée la valeur entière correspondante aux numéros de ligne et de colonne :

$$D = \{ 'A' : 11, 'B' : 12, 'C' : 13, \quad \dots \quad 'Y' : 55, 'Z' : 61 \}$$

1) Ecrire une fonction **cle()** qui retourne le dictionnaire **D**.

Nota : il ne faut pas définir **D** manuellement, plutôt le créer par programmation. On peut utiliser la commande **chr(n)** qui permet de générer le caractère correspondant au code ASCII **n**, ainsi : chr(65) = 'A' ; chr(66) = 'B' ; ... chr(90) = 'Z'.

Les codes ASCII des 26 lettres majuscules de l'alphabet appartiennent à l'intervalle [65, 90].

2) Ecrire une fonction **codage(mot, D)** qui prend en argument un mot (composé uniquement par des caractères majuscules) et le dictionnaire **D** et retourne l'entier correspondant au mot.

3) Ecrire une fonction **decodage(n, D)** qui prend en argument un entier **n** et le dictionnaire **D** et retourne le mot correspondant.

Exercice 3

En mathématiques, les **nombre**s de **Catalan** forment une suite d'entiers naturels utilisée dans divers problèmes de dénombrement.

Dans sa définition **ré**cur

sive le nombre de Catalan d'indice **n** est défini par :

$$(1) \quad C(n) = \begin{cases} 1 & \text{si } n = 0 \\ \sum_{p=0}^{n-1} C(p) \times C(n-1-p) & \text{si } n > 0 \end{cases}$$

Les sept premiers nombres de Catalan (pour **n** de 0 à 6) sont :

<i>n</i>	0	1	2	3	4	5	6
<i>C(n)</i>	1	1	2	5	14	42	132

1) Ecrire une fonction récur

sive **Catalan(n)** qui prend en argument un entier **n** et retourne le nombre de Catalan **C(n)**.

2) On définit la matrice carrée de **Hankel** **H** d'ordre **n** dont l'élément **H_{i,j}** est le nombre de Catalan **C(i + j)** , ainsi pour **n = 4**, on a :

$$H = \begin{pmatrix} 1 & 1 & 2 & 5 \\ 1 & 2 & 5 & 14 \\ 2 & 5 & 14 & 42 \\ 5 & 14 & 42 & 132 \end{pmatrix}$$

Ecrire une fonction **Hankel(n)** qui prend en argument un entier **n** et retourne la matrice de Hankel d'ordre **n** sous forme de tableau **numpy**.

3) Les éléments de la matrice de Hankel H sont constants le long des diagonales ascendantes, c'est-à-dire dont les indices vérifient la relation : $H_{i,j} = H_{i-1,j+1}$.

Ecrire une fonction **Test(H)** qui prend en argument une matrice carrée H et renvoie **True** si H est une matrice de Hankel et **False** sinon.

Pour les grandes valeurs de n , la complexité temporelle pour calculer $C(n)$ par l'équation (1) n'est pas efficace. Un moyen plus rapide pour calculer les nombres de Catalan est la définition suivante :

$$(2) \quad C(n) = \begin{cases} 1 & \text{si } n = 0 \\ \frac{2(2n-1)}{n+1} C(n-1) & \text{si } n > 0 \end{cases}$$

4) Ecrire une fonction réursive **Catalan2(n)** qui prend en argument un entier n et retourne le nombre de Catalan $C(n)$ calculé par l'équation (2).

5) On se propose de tracer la courbe des nombres **$C(i)mod1000$** en fonction des entiers i , où *mod* est l'opérateur modulo.

Ecrire une fonction **graphe(n)** qui prend en argument un entier n et permet de tracer cette courbe en définissant les listes suivantes :

- X composée par les entiers i variant de 0 à n ,
- Y contenant les nombres $C(i)mod1000$ et ce pour i variant de 0 à n .

6) On veut enregistrer les entiers i variant de 0 à n ainsi que les nombres de Catalan correspondants $C(i)$ dans un fichier texte tel que, chaque ligne du fichier contient l'entier i et $C(i)$ séparés par une virgule, comme le montre l'extrait suivant :

0,1
1,1
2,2
3,5
4,14
...

Ecrire une fonction **stockage(fichier, n)** qui prend en argument un nom de fichier et un entier n et enregistre les entiers i ainsi que $C(i)$ (i variant de 0 à n) conformément à l'extrait ci-dessus.

Exercice 4

Le $n^{ième}$ terme de la suite de **nombres de triangles** est donné par $t_n = n(n+1)/2$, $n \geq 1$; les dix premiers nombres de triangles sont : 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, ...

L'alphabet comptant 26 lettres majuscules, on peut repérer chaque lettre par un nombre qui correspond à sa position alphabétique conformément au tableau suivant :

A	B	C	...	Y	Z
1	2	3		25	26

La **valeur numérique** d'un mot (chaîne de caractères majuscules) est définie par la somme des positions alphabétiques de toutes les lettres qui le composent. Par exemple, la valeur

numérique du mot **SKY** est : $19 + 11 + 25 = 55 = t_{10}$. Si la valeur du mot est un nombre triangulaire, le mot est appelé un **mot triangulaire**.

Soit un fichier texte contenant des mots majuscules séparés par le caractère " : ", comme le montre un extrait de l'exemple suivant :

Ligne 1	CODE:COURS:IPEIS:	...	:PYTHON
Ligne 2	MOT:MATH:SKY:	...	:EXAMEN
...	...		

On se propose de calculer le nombre de mots triangulaires dans ce fichier.

Ecrire une fonction **triangulaire(fichier)** qui prend en argument un fichier texte et retourne le nombre de mots triangulaires dans le fichier en effectuant les calculs suivants :

- générer la liste L_words contenant tous les mots du fichier,
- générer la liste T des nombres de triangles t_n où :
 - n varie de 1 à $\left\lfloor \sqrt{\max(P) * 26} \right\rfloor$
 - P est la liste contenant la longueur de tous les mots du fichier
 - $\lfloor x \rfloor$ est le symbole de la partie entière du réel x

Nota : pour calculer la position alphabétique d'un caractère c , on peut utiliser la commande Python **ord(c)** qui permet d'obtenir le code ASCII correspondant à c . Les caractères majuscules de l'alphabet correspondent aux codes ASCII consécutifs compris entre 65 (pour le A) et 90 (pour le Z) ; comme $\text{ord}("B") = 66$, alors la position alphabétique de B est égale : $\text{ord}("B") - \text{ord}("A") + 1 = 66 - 65 + 1 = 2$.