

DEVOIR DE SYNTHESE DE SEMESTRE 2**Matière : INFORMATIQUE****Filière : PB1 - Durée : 2 h**

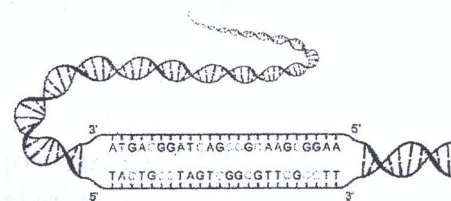
Toutes les réponses aux questions doivent être rédigées en **Python** et non pas en syntaxe algorithmique.

A ne pas oublier d'importer les modules nécessaires pour chaque exercice.

Exercice1 :

L'ADN est généralement constitué de deux brins enroulés en double hélice. Chaque brin est composé d'une succession de nucléotides formé d'une base nucléique, ou base azotée : l'adenine (A), la thymine (T), la cytosine (C), ou la guanine (G).

Chaque nucléotide d'un brin trouve son nucléotide correspondant dans l'autre brin : l'adénine A s'associant à la thymine T et la cytosine C à la guanine G. Ainsi, les deux brins d'ADN sont dits complémentaires.



La transcription est un mécanisme qui permet de "recopier" l'ADN dans le noyau de la cellule pour former un ARN (acide ribonucléique) en remplaçant la base thymine (T) par l'uracile (U).

Voici un exemple de séquence représentant un brin d'ADN.

ADN1 = "TGAGTGAACGTATATATCGGCGCA"

Travail demandé :

1. Ecrire une fonction **nature(s)** qui prend en paramètre une chaîne de caractère s et retourne :
 - la chaîne de caractères "ADN" si s ne contient que des caractères A, T, G ou C,
 - la chaîne de caractères "ARN" si s ne contient que des A, U, G ou C
 - la chaîne de caractères "Erreur" si s n'est ni ADN ni ARN

L'appel à la fonction `nature(ADN1)` retourne "ADN"

2. Ecrire une fonction **transcrit(s)** qui prend en paramètre une chaîne de caractère s et renvoie l'ADN transcrit en ARN (une chaîne de caractère).

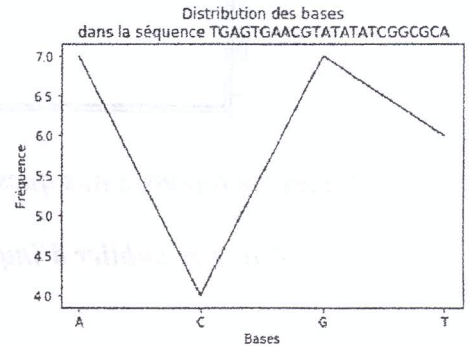
L'appel à la fonction `transcrit(ADN1)` retourne "UGAGUGAACGUAUAUAUCGGCGCA"

3. Ecrire la fonction **brincomplémentaire(s)** qui prend en paramètre une chaîne de caractères *s* représentant un brin d'ADN et qui renvoie son brin complémentaire en utilisant le dictionnaire de complémentarité `comp={'A': 'T', 'T': 'A', 'G': 'C', 'C': 'G'}`.

L'appel à la fonction `brincomplémentaire(ADN1)` retourne "ACTCACTTGCATATATAGCCGCGT"

4. Représenter graphiquement la distribution des différentes bases dans un brin d'ADN où les abscisses sont les bases d'ADN : ['A', 'C', 'G', 'T'] et les ordonnées sont le nombre d'occurrence de chaque base dans le brin ADN.

NB : il faut ajouter le titre et les labels des axes du graphe suivant.



Soit le fichier texte nommé 'BrinsADN.txt', où chaque ligne représente un brin d'ADN, comme suit :

```
TGAGTGAACGTATATATCGGCGCA
DGAAUGAACGUARATATCGTAACA
AACGTGAACGTCAATATCGGCGCA
```

'BrinsADN.txt'

5. Ecrire un script python permettant de :
 - a. Lire le fichier 'BrinsADN.txt'.
 - b. créer un fichier 'ADNs.txt'.
 - c. Pour chaque brin ADN lu du fichier 'BrinsADN.txt'
 - i. chercher son complémentaire s'il est ADN
 - ii. écrire le brin ADN et son complémentaire séparé par ';' dans une nouvelle ligne du fichier 'ADNs.txt'.
 - d. fermer les deux fichiers.

Exercice 2 :

Tous les matériaux qui nous entourent sont formés d'atomes. les atomes constituant la matière (gaz ou liquide) sont disposés de façon essentiellement aléatoire dans l'espace.

L'objectif de cet exercice est de calculer la distance inter-atomique.

Un atome est considéré comme un point 3D dans l'espace défini par (x,y,z).

Pour cela, on dispose d'un fichier nommé 'Atomes.txt', les coordonnées de chaque atome sont sauvegardées dans une ligne de la forme `x ; y ; z`

Voici un extrait du fichier 'Atomes.txt' :

```
0.5;1.0;1.0
4.5;1.0;3.0
4.5;1.5;1.5
0.0;2.5;4.5
4.0;4.0;1.5
1.0;1.5;2.5
3.0;3.0;3.5
2.0;3.5;1.5
2.0;1.5;0.0
4.0;3.0;0.0
```

Atomes.txt

Travail demandé :

1. Ecrire une fonction `distance_atomes(x1,y1,z1,x2,y2,z2)` qui renvoie la distance euclidienne entre les deux atomes dont leurs coordonnées sont passées en paramètre. On rappelle que la distance euclidienne d entre deux atomes de coordonnées cartésiennes respectives $(x1, y1, z1)$ et $(x2, y2, z2)$ se calcule comme suit :

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2 + (z1 - z2)^2}$$

NB : utiliser la fonction `sqrt` du module `math`.

2. Ecrire une fonction `chargerFichier()` qui ouvre, lit le fichier 'Atome.txt' et renvoie une liste de tuples où chaque tuple représente les coordonnées d'un atome.

L'appel de la fonction `chargerFichier()` retourne
[(2.5, 1.0, 2.0), (3.5, 0.0, 0.5), (2.5, 1.5, 0.5), (3.0, 4.5, 1.5),
(5.0, 4.0, 3.0), (2.0, 1.5, 2.0), (2.0, 4.5, 0.0), (1.5, 5.0, 4.0),
(4.0, 4.5, 2.0), (5.0, 2.0, 2.0)]

L'atome de coordonnées (2.5, 1.0, 2.0) est d'indice 0 dans la liste.

3. Ecrire une fonction `RempMatrDist()` qui calcule et renvoie une matrice de distance de type tableau2D numpy en suivant les étapes suivantes :

- a. Charger le fichier dans une liste de tuples `L`.
- b. Initialiser une matrice carrée `A` d'ordre `n` (avec `n` le nombre d'atomes dans la liste `L`)
- c. Calculer A_{ij} , tel que A_{ij} est égale à la distance entre l'atome d'indice `i` et l'atome d'indice `j` dans la liste `L`.

L'appel de la fonction `RempMatrDist('Atoms.txt')` retourne `A=`
`array([[0. , 2.06, 1.58, 3.57, 4.03, 0.71, 4.06, 4.58, 3.81, 2.69],`
`[2.06, 0. , 1.8 , 4.64, 4.95, 2.6 , 4.77, 6.42, 4.77, 2.92],`
`[1.58, 1.8 , 0. , 3.2 , 4.33, 1.58, 3.08, 5.05, 3.67, 2.96],`
`[3.57, 4.64, 3.2 , 0. , 2.55, 3.2 , 1.8 , 2.96, 1.12, 3.24],`
`[4.03, 4.95, 4.33, 2.55, 0. , 4.03, 4.27, 3.77, 1.5 , 2.24],`
`[0.71, 2.6 , 1.58, 3.2 , 4.03, 0. , 3.61, 4.06, 3.61, 3.04],`
`[4.06, 4.77, 3.08, 1.8 , 4.27, 3.61, 0. , 4.06, 2.83, 4.39],`
`[4.58, 6.42, 5.05, 2.96, 3.77, 4.06, 4.06, 0. , 3.24, 5.02],`
`[3.81, 4.77, 3.67, 1.12, 1.5 , 3.61, 2.83, 3.24, 0. , 2.69],`
`[2.69, 2.92, 2.96, 3.24, 2.24, 3.04, 4.39, 5.02, 2.69, 0.]])`

La distance entre l'atome d'indice 0 et l'atome d'indice 1 est égale à 2.06