DM n° 1

Informatique tronc commun

A rendre le 16/09
Facultatif

NOM/Prénom

Consignes : *Le seul langage de programmation autorisé dans cette épreuve est Python. On fera particulièrement attention à l'indentation et on n'hésitera pas à agrémenter les codes de quelques commentaires (si besoin).*
Merci de rendre directement le sujet complété.

Problème 1 : Applications directes du cours et des TP

1. Définir une fonction `f` qui à x associe 4x+7.

2. Écrire un programme qui demande à l'utilisateur d'entrer un nombre de secondes et qui l'affiche sous la forme d'heures/minutes/secondes. Par exemple, si l'utilisateur entre `25000`, le code devra renvoyer : `6 heures 56 minutes 40 secondes`

3. Écrire une fonction `factorielle` qui à un entier $n > 0$ associe $n!$ (n factorielle), c'est-à-dire $n \times (n-1) \times ... \times 2 \times 1$. (Indication : utiliser une boucle for)

4. Écrire un programme qui, à l'aide d'une simple boucle for, affiche tous les entiers pairs de 100 à 0 par ordre décroissant.

5. Écrire une fonction `moyenne` renvoyant la moyenne des éléments d'une liste de nombres (sans utiliser la fonction prédéfinie `mean`). Si la liste est vide, on renverra `0`.

6. Écrire une fonction `prix_total` qui prend en argument deux listes (`prix_au_kg` et `masse_voulue`) représentant respectivement les prix au kilogramme de chaque ingrédient et la masse (en kg) nécessaire à la fabrication du goûter, rangés bien sûr dans le même ordre pour qu'ils se correspondent. Cette fonction doit renvoyer le prix total à payer à la caisse.

7. On veut résoudre le même problème, mais cette fois-ci `prix_au_kg` et `masse_voulue` sont des *dictionnaires* qui n'ont pas forcément le même nombre de clés (`prix_au_kg` contient l'ensemble des articles proposés à la vente alors que les clés de `masse_voulue` concernent seulement les articles nécessaires à la fabrication du goûter). Écrire alors une fonction `prix_total_avec_dico` qui permette de renvoyer le prix total à payer à la caisse.

Problème 2 : Étude d'une séquence ADN.

L'ADN de tout être humain est composé d'unités structurales appelées nucléotides. Chaque sorte de nucléotides est formée de trois unités : une base azotée, un sucre et un groupe phosphate. Il existe quatre sortes de nucléotides formant l'ADN : l'adénine (A), la guanine (G), la thymine (T) et la cytosine (C). Les deux brins d'ADN sont reliés entre eux par les nucléotides qui forment des paires complémentaires : l'adénine avec la thymine (A-T ou T-A) et la guanine avec la cytosine (G-C ou C-G).

1. On considère une courte séquence d'ADN, représentée par une chaîne de caractères composée des lettres. Exemple de séquence : `sequence = "ATGCGATAGGCTAGCTA"`
Écrire une fonction `compte(sequence)` qui prend en argument la chaîne de caractères de la séquence ADN et qui renvoie un dictionnaire donnant le nombre d'occurrences de chaque nucléotide. (La clé est donc une chaîne de caractères correspondant au nucléotide et la valeur est le nombre de fois que ce nucléotide est présent dans la séquence).

2. Écrire une fonction `majoritaire(sequence)` qui renvoie le nucléotide (la lettre) le plus fréquent de la séquence.

3. Le taux de GC (ou pourcentage de GC, ou coefficient de Chargaff) d'une séquence d'ADN est défini comme la proportion de bases nucléiques de cette séquence étant soit une guanine (G), soit une cytosine (C). Écrire une fonction `tauxGC(sequence)` qui renvoie ce taux (un flottant compris entre 0 et 1).

4. Un codon est une suite de trois nucléotides consécutifs. Le premier codon est donc la suite des trois premières lettres de la chaîne de caractères, le deuxième codon est la suite des trois lettres suivantes, etc.
Écrire une fonction `codons(sequence)` qui renvoie la liste des codons contenus dans la séquence et dans l'ordre. On ignore les nucléotides en fin de séquence s'ils ne forment pas un codon complet.

5. Pour éviter les erreurs et pouvoir appliquer ces fonctions à n'importe quelle chaîne de caractères, on va écrire une fonction permettant d'éliminer les chaînes ne contenant pas que des nucléotides.
Écrire une fonction `valide(sequence)` qui renvoie `True` si la séquence ne contient que des lettres A, T, G, C et `False` sinon.

6. Écrire une fonction `frequences(sequence)` qui renvoie un dictionnaire donnant la fréquence de chaque nucléotide. La fréquence est ici un flottant entre 0 et 1 (ou le pourcentage).

7. Écrire une fonction `complementaire(sequence)` qui renvoie la chaîne de caractères correspondant à la chaîne complémentaire. On pourra s'aider d'un dictionnaire avec les correspondances.

8. Afin de pouvoir retrouver un codon dans la séquence, il faut pouvoir retrouver un motif dans la chaîne de caractères.
Écrire une fonction `position(sequence,motif)` qui renvoie la liste des indices où le motif apparaît dans la séquence.