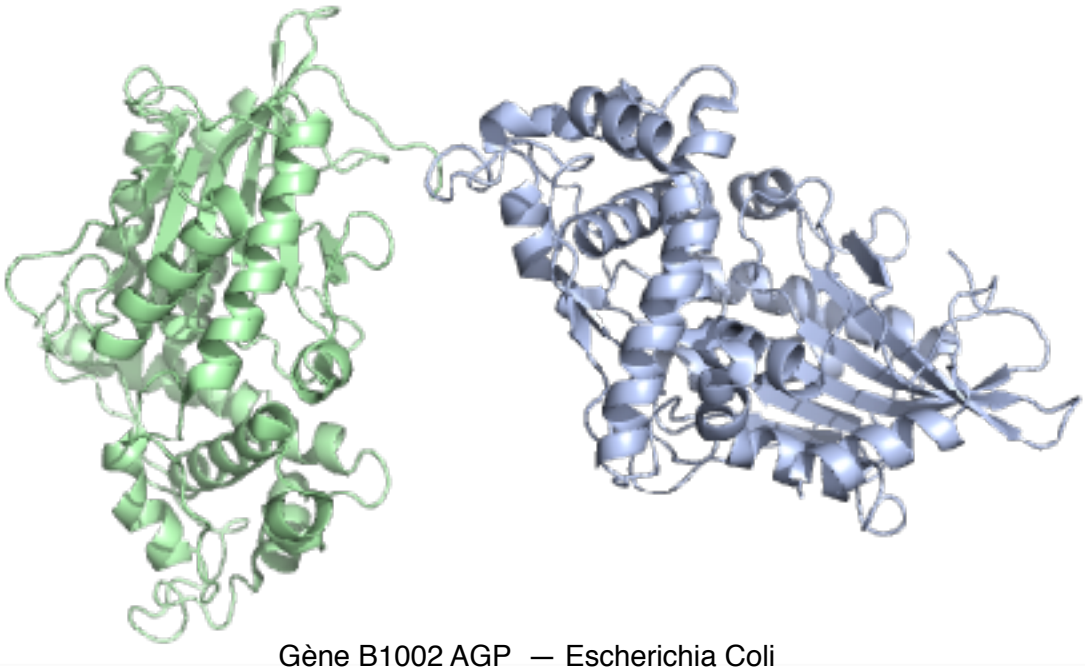
BIO - INFORMATIQUE

Le code génétique

L'objectif de ce TD est de mettre a profit les dictionnaires en python pour traduire une séquence de nucléotides en séquence d'acides aminés.



Gène B1002 AGP — Escherichia Coli

Préparation :

Relire les parties de cours sur les dictionnaires et les compréhension de liste en python, ainsi que la lecture et écriture dans un fichier.

I Récupération de données : la séquence ADN

La séquence ADN de la bactérie Escherichia Coli est donnée dans le fichier EscherichiaColi.txt

a - Écrire une fonction getSequence(monFichier) qui prend en argument l'adresse du fichier sous forme de chaine et qui renvoie la séquence de nucléotides « ACGTGTCAA » de la bactérie.
Rq : Prenez le temps de regarder dans le fichier .txt pour en comprendre la structure.
A vous aussi de voir à quel emplacement placer le fichier pour qu'il soit vu par python cette étape pose toujours problème car votre session est sur un ordinateur distant.

II Recherche de gènes

La séquence ADN de Escherichia Coli contient plus de 4 millions de nucléotides ACG ou T.
Le code génétique est un système de correspondance entre un **codon** [qui est une suite de 3 nucléotides] et **un acide aminé**. Seuls 22 acides aminés sont utilisés pour former des protéines dans la plupart des organismes.

Enfin toute la séquence n'est pas « codante » mais **la lecture peut commencer sur n'importe quel nucléotide**, et la séquence complémentaire s'exprime également car la double hélice d'ADN est ouverte lors de la lecture.
La lecture commence lorsque l'on rencontre un **codon start** : «**ATG**» «**TTG**» ou «**CTG**».
La lecture ne s'arrête que si l'on rencontre un **codon stop** : «**TAG**» «**TAA**» ou «**TGA**»

a - Combien de codons sont possibles ? Conclure sur le caractère «redondant» du code génétique.

b - Créer un dictionnaire de conversion A —> T, C —> G, G—> C et T—> A pour obtenir la séquence complémentaire. On le notera **convert**.
- Écrire une fonction **seqComp**(seq) qui renvoie la séquence complémentaire de celle passée en argument. Faire un test sur une séquence courte.

c - Créer une liste **starter** et une liste **stopper** contenant chacune les 3 codons (chaînes) de début et de fin. A l'aide de la commande « IN » de python on pourra rapidement savoir si un codon est un marqueur de début ou de fin de gène.

d - Nous voulons compter le nombre potentiel de gènes :
Écrire une fonction countStarter(seq) qui prend en argument une séquence et renvoie un tuple : (nATG, nGTG, nTTG) contenant les nombres de départ potentiels en ATG, GTG ou en TTG.
Penser à prendre en compte les deux brins d'ADN.

e - On compte en réalité moins de 10 000 gènes de protéines. Comment expliquer cet écart ?
Ce point sera important pour coder un algorithme de lecture de gène.

f - Écrire une fonction **geneFinder**(seq) qui prend en argument la séquence et qui renvoie les gènes sous la forme d'un tuple (mesGenesATG, mesGenesGTG, mesGenesTTG). Dans ce tuple, mesGenesATG etc.. seront des listes de gènes commençants par ATG etc.. et finissant par un codon stop quelconque. Chaque gène sera une liste de chaine de codon.

Exemple :
['**ATG**', 'TCT', 'CTG', 'TGT', 'GGA', 'TTA', 'AAA', 'AAA', 'GAG', 'TGT', 'CTG', 'ATA', 'GCA', 'GCT', 'TCT', 'GAA', 'CTG', 'GTT', 'ACC', 'TGC', 'CGT', 'GAG', '**TAA**']

Chaque gène conservera ainsi le codon start mais aussi le codon stop bien que ce dernier ne fasse pas partie de la protéine. Quels sont à présent les nombres de gènes ?

g - Écrire une fonction **geneSeq**(geneList) qui prend en argument un gène sous forme de liste de codon (comme ci-dessus) et qui renvoie la chaine de nucléotide (voir ci-dessous). On pourra mettre à profit la méthode **.join()** sur une chaine vide : « ».join(['T', 'A', 'T', 'A']) renvoie 'TATA'.

Écrire ensuite une fonction **geneList**(geneSeq) qui prend en argument la séquence du gène et renvoie la liste des chaines de ses codons. On renverra un message d'erreur lorsque la séquence ne permet pas de regrouper en codon de 3 nucléotides.

On donne le gène suivant : Gène agp B102 impliqué dit-on dans le métabolisme de la bactérie !

"ATGAACAAAACGCTAATCGCCGCAGCTGTGGCAGGGATAGTTTTACTCGCTTCAAACGCTCAGGCACAAACCGTACC
GGAAGGCTATCAGCTACAGCAAGTGCTCATGATGAGCCGCCATAACTTACGTGCGCCGCTGGCGAACAATGGCAGT
GTGCTGGAGCAGTCGACGCCGAATAAATGGCCAGAATGGGACGTCCCCGGTGGGCAACTCACCACCAAAGGTGGC
GTGCTCGAAGTGTATATGGGCCATTACATGCGTGAATGGCTGGCAGAGCAGGGGATGGTGAAATCGGGGGAATGCC
CGCCGCCGTACACCGTTTATGCCTATGCCAATAGTCTGCAACGTACCGTTGCGACCGCACAGTTCTTTATTACCGGCG
CATTCCCGGGGGTGTGATATTCCTGTGCATCACCAGGAAAAAATGGGCACTATGGACCCAACCTTTAACCCGGTGATCA
CCGATGATTCCGCCGCATTCAGTGAACAGGCGGTGGCGGCAATGGAGAAAGAGCTCAGCAAACTCCAGCTTACCGA
CAGCTACCAGCTACTGGAAAAAATCGTTAACTATAAAGATTCCCCTGCCTGTAAAGAGAAACAACAGTGTTCGCTGGT
GGATGGCAAAAATACCTTTAGCGCCAAGTATCAACAAGAACCAGGTGTTTCCGGGCCGCTGAAAGTCGGCAACTCGC
TGGTAGATGCGTTTACTTTGCAATATTACGAAGGTTTTCCGATGGATCAGGTGGCCTGGGGAGAAATCAAATCTGACC
AGCAGTGGAAGGTGTTGTCGAAGCTTAAAAACGGCTACCAGGACAGCCTGTTTACCTCACCGGAAGTGGCGCGCAA
TGTTGCGAAACCGCTGGTCAGTTATATCGACAAAGCTCTGGTCACCGATCGCACCAGCGCACCGAAAATTACAGTGT
TGGTTGGGCACGACTCCAACATTGCCTCTCTGTTAACGGCGCTGGATTTCAAACCGTATCAGTTGCATGACCAGAAC
GAACGCACGCCGATTGGCGGCAAAATCGTTTTCCAGCGTTGGCATGACAGCAAAGCCAATCGCGATTTGATGAAAAT
TGAATATGTGTATCAGAGTGCGGAACAGTTACGTAATGCCGATGCGTTAACCCTGCAGGCACCTGCGCAGCGTGTGA
CGCTGGAATTAAGCGGTTGCCCGATAGACGCTGATGGTTTCTGCCCGATGGATAAGTTTGATAGCGTGTTGAATGAAG
CGGTGAAATAA"

h - Vérifier en 1 ligne que ce gène est bien présent dans la séquence ou son complémentaire.

i - Écrire une fonction **genePosition**(gene, seq) qui prend en argument la chaine du gène et de la séquence et qui renvoie une liste contenant des tuples d'indice (début, fin). Exemple : **genePosition("TAC", "TACTICTACETAC")** renvoie **[(0, 2), (6, 8), (10, 12)]**.

III Traduction du gène

On donne le code génétique sous la forme d'un dictionnaire en python :
{clef : valeur} —> **{codon : acide aminé}** vous pouvez la copier-coller dans votre code.

codeGenetique = {
"TTT":"F","TTC":"F","TTA":"L","TTG":"L","TCT":"S","TCC":"S","TCA":"S","TCG":"S",
"TAT":"Y","TAC":"Y","TAA":"*","TAG":"*","TGT":"C","TGC":"C","TGA":"*","TGG":"W",

"CTT":"L","CTC":"L","CTA":"L","CTG":"L","CCT":"P","CCC":"P","CCA":"P","CCG":"P",
"CAT":"H","CAC":"H","CAA":"Q","CAG":"Q","CGT":"R","CGC":"R","CGA":"R","CGG":"R",

"ATT":"I","ATC":"I","ATA":"I","ATG":"M","ACT":"T","ACC":"T","ACA":"T","ACG":"T",
"AAT":"N","AAC":"N","AAA":"K","AAG":"K","AGT":"S","AGC":"S","AGA":"R","AGG":"R",

"GTT":"V","GTC":"V","GTA":"V","GTG":"V","GCT":"A","GCC":"A","GCA":"A","GCG":"A",
"GAT":"D","GAC":"D","GAA":"E","GAG":"E","GGT":"G","GGC":"G","GGA":"G","GGG":"G"}

a - A l'aide des fonctions conçues au II, écrire une fonction **aminoSeq**(gene), qui peux prendre en argument un gène sous une forme de liste de codon ou simplement de chaine, et qui renvoie la chaine de caractères associée aux acides aminés, conformément au code génétique ci-dessus. Dans le cas où le gène est passé à la fonction dans une forme incompatible la fonction renverra un message d'erreur. On ne vérifiera pas la validité biologique des données traitées.

b - Écrire un « bout de programme » [comprendre : ce n'est pas une fonction] qui construise un dictionnaire inversé du code génétique. On souhaite donc avoir un **dictionnaire** qui à un acide aminé fasse correspondre l'ensemble (au sens python —> **set**) des codons qui lui sont associés.

c - Consulter le **tableau inverse du code génétique** sur wikipédia :
—> https://fr.wikipedia.org/wiki/Code_génétique.

Certains acides aminés ne sont pas présents dans notre table pourquoi ??

d - On veut compter la fréquence des codons ainsi que des acides aminés dans l'ensemble des gènes de la bactérie Escherichia Coli. On fourni pour cela la structure de données ci-dessous.
De quoi s'agit-il ?

```
freqCodon = {
    "A":{"GCT":0,"GCC":0,"GCA":0,"GCG":0},
    "C":{"TGT":0,"TGC":0},
    "D":{"GAT":0,"GAC":0},
    "E":{"GAA":0,"GAG":0},
    "F":{"TTT":0,"TTC":0},
    "G":{"GGT":0,"GGC":0,"GGA":0,"GGG":0},
    "H":{"CAT":0,"CAC":0},
    "I":{"ATT":0,"ATC":0,"ATA":0},
    "K":{"AAA":0,"AAG":0},
    "L":{"CTT":0,"CTC":0,"CTA":0,"CTG":0,"TTA":0,"TTG":0},
    "M":{"ATG":0},
    "N":{"AAT":0,"AAC":0},
    "O":{"TAG":0},
    "P":{"CCT":0,"CCC":0,"CCA":0,"CCG":0},
    "Q":{"CAA":0,"CAG":0},
    "R":{"CGT":0,"CGC":0,"CGA":0,"CGG":0, "AGA":0,"AGG":0},
    "S":{"TCT":0,"TCC":0,"TCA":0,"TCG":0,"AGT":0,"AGC":0},
    "T":{"ACT":0,"ACC":0,"ACA":0,"ACG":0},
    "U":{"TGA":0},
    "V":{"GTT":0,"GTC":0,"GTA":0,"GTG":0},
    "W":{"TGG":0},
    "Y":{"TAT":0,"TAC":0},
    "*":{"TAG":0,"TAA":0,"TGA":0}}
```

Écrire une fonction **codonFreq**(mesGenes) qui prend en argument l'ensemble des gènes sous forme de liste de liste retourné par **geneFinder** au II-f, et qui remplit la structure précédente.

e - Écrire une fonction **acidAminFreq**(mesGenes) qui elle renvoie la fréquence des acides aminés en pourcent.

f - On souhaite tracer un diagramme en bâtons de la fréquence des acides aminés.
On donne le code ci-dessous. A vous de tester puis de commenter ce code :

```
from matplotlib import pyplot as plt;          import pylab

AA_Freq=acidAminFreq(mesGenes)

name = [k for k in AA_Freq]                #C1
x = range(len(name))                       #C2
y = [AA_Freq[k] for k in AA_Freq]          #C3
width=0.05                                 #C4

plt.bar(x, y, width)                       #C5
pylab.xticks(x, name, rotation=45)         #C6
plt.scatter([i+width/2.0 for i in x],y,color='k',s=40) #C7

plt.xlim(0,11); plt.ylim(0,14)         #C8
plt.grid()

plt.ylabel('Fréquence en %')       #C9
plt.title('Fréquence d''apparition des acides aminés dans les gènes d''EscherichiaColi')
plt.show()
```

FIN