**Proposition d’un contexte pour la programmation en python d’un traducteur ARN (ou ADN) / protéine en Première spécialité SVT.**

L. Roux

La démarche proposée ici cherche à utiliser la logique de programmation pour faire comprendre le fonctionnement d’un ribosome (concept de décalage du cadre de lecture). En effet, comme le ribosome, un programme traduisant une séquence AUGC en une suite d’acides aminés devra scanner la chaine de caractère nucléotidique. Le développeur devra forcément définir le pas de ce scan. C’est autour de la valeur de ce paramètre que l’on souhaite engager une discussion avec les élèves. En effet, le programme proposé devra permettre de tester des modèles de lectures chevauchants (pas de 1, 2..) ou non chevauchants (3 ou multiple de 3).

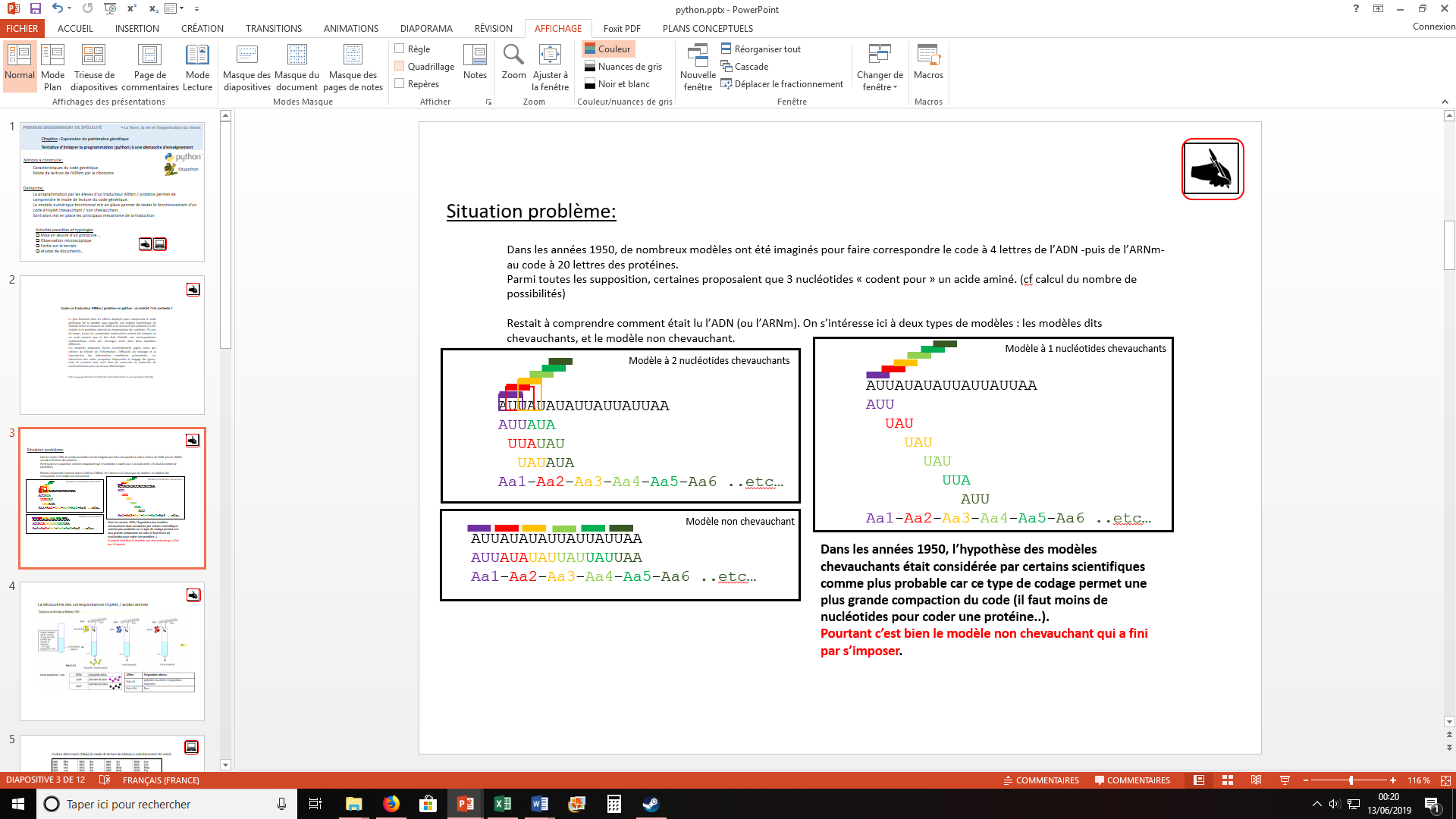
Point de départ : un peu d’Histoire…

Extrait d’un article de Pour La science, « L’invention du code génétique »  (https://www.pourlascience.fr/sd/histoire-sciences/linvention-du-code-genetique-5672.php):

*Le plus fascinant dans les efforts déployés pour comprendre le code génétique fut la rapidité avec laquelle une énigme biochimique (la relation entre la structure de l’ADN et la structure des protéines) a été réduite à un problème abstrait de manipulation des symboles. En peu de temps, toutes les complexités moléculaires avaient été balayées et on avait compris que le but était d’établir une correspondance mathématique entre des messages écrits dans deux alphabets différents.*

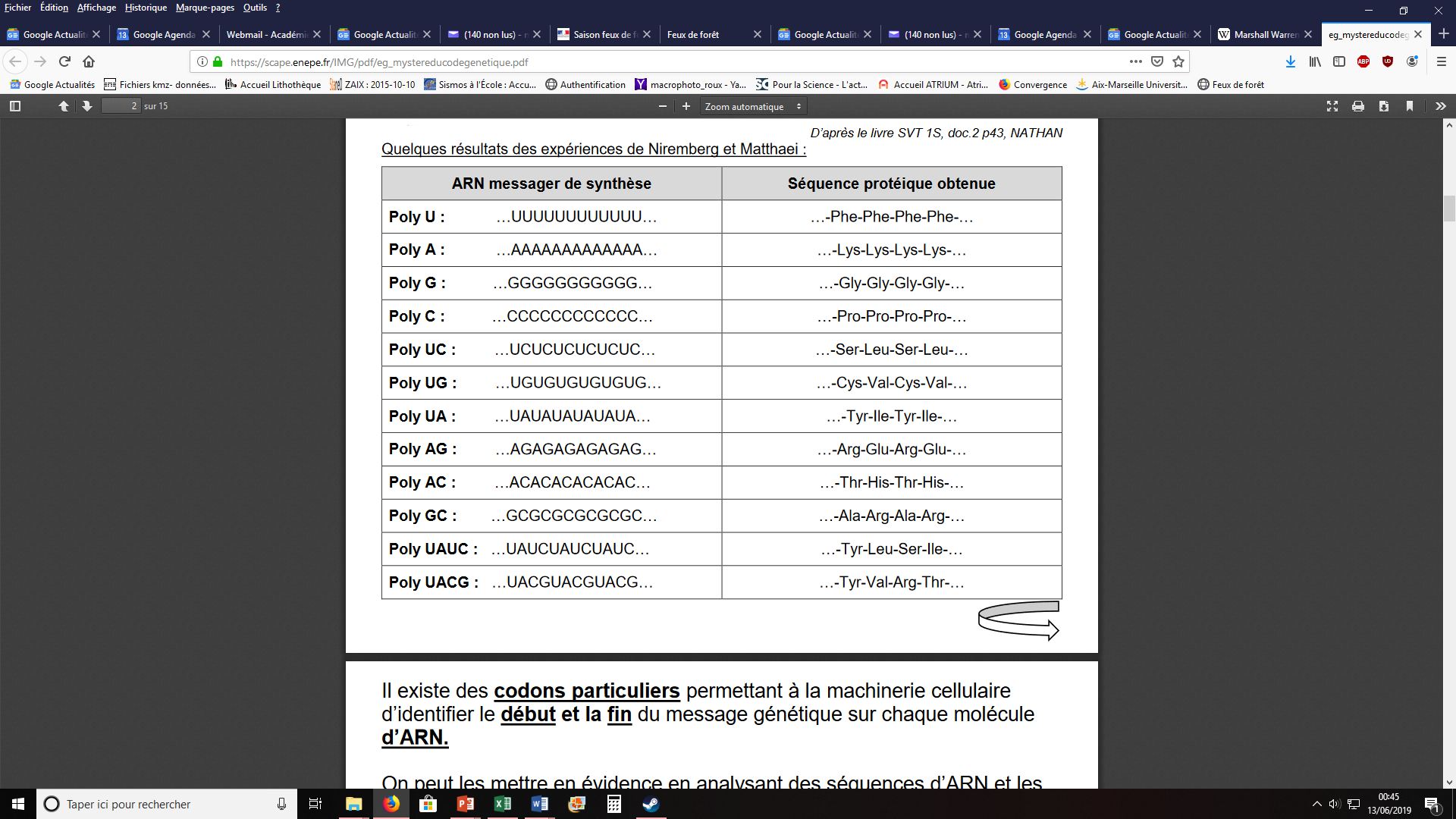
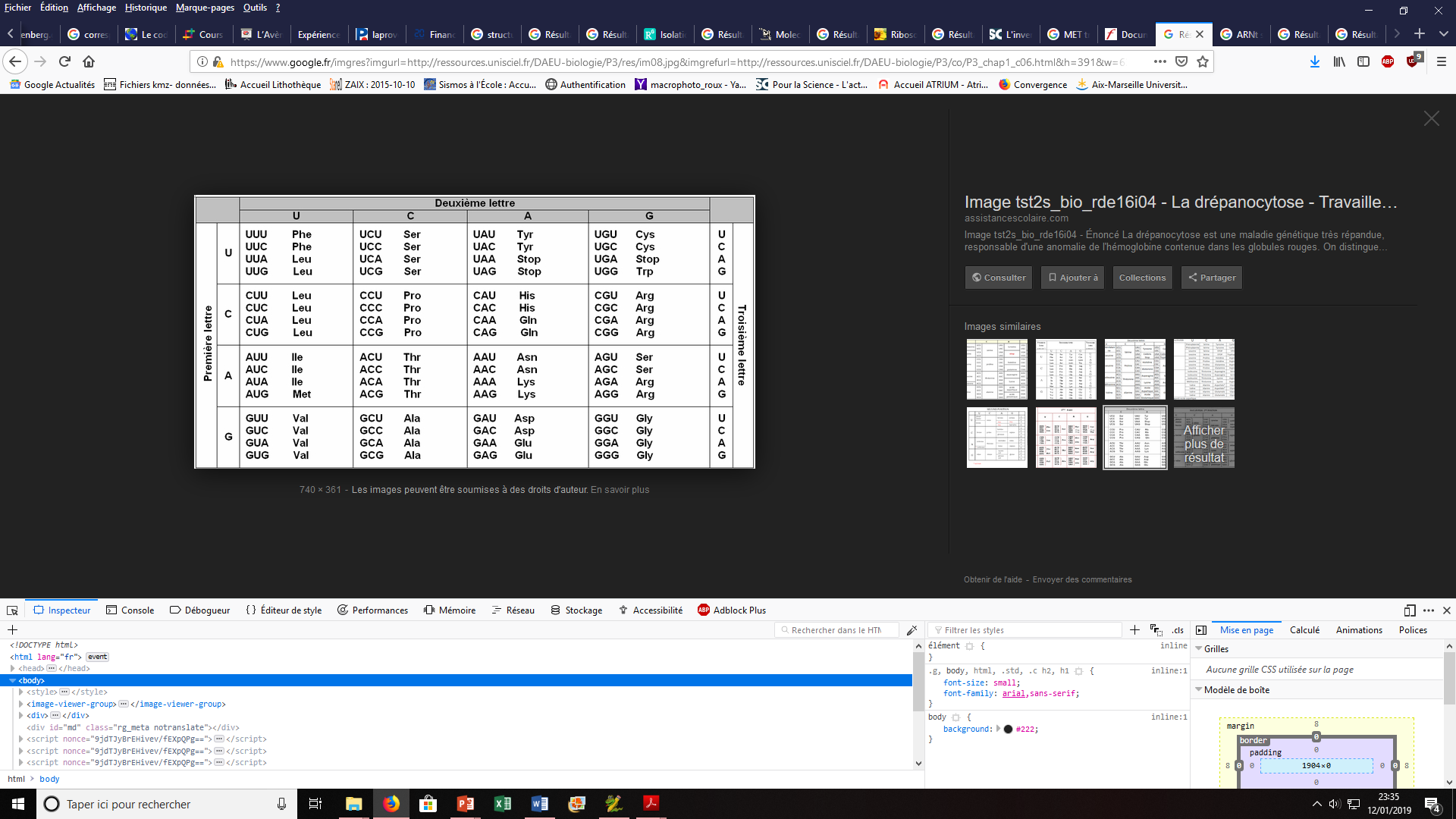
*Les solutions proposées furent essentiellement jugées selon des critères de théorie de l’information. L’efficacité du stockage et la transmission des informations semblaient primordiales. Les théoriciens des codes essayaient d’apprendre le langage des gènes, mais ils auraient tout aussi bien pu concevoir un protocole de communications pour un réseau informatique.*

Dans les années 1950, de nombreux modèles ont été imaginés pour faire correspondre le code à 4 lettres de l’ADN -puis de l’ARNm- au code à 20 lettres des protéines. Parmi toutes les suppositions, certaines proposaient que 3 nucléotides « codent pour » un acide aminé. (cf calcul du nombre de possibilités). LE cadre de lecture est donc constitué de 3 positions de nucléotides. Restait à comprendre comment étaient lus ces cadres sur l’ADN (ou l’ARNm). On s’intéresse ici à deux types de modèles : les modèles dits chevauchants, et le modèle non chevauchant.



On peut imaginer des décalages du cadre de lecture avec des n > 3 qui laisserait des nucléotides. Certains scientifiques ont pensé que de tels codes étaient probables, les nucléotides délaissés jouant alors le rôle d’intercalaire entre triplets codants…

Sont fournis aux élèves, de façon classique les documents simplifiés relatant les expériences de Niremberg et Matthaei (1961) ainsi que leur résultat : la correspondance triplet / acide aminé.

Dans le tableau du code génétique à droite ci dessus, le mode de lecture (intitulé des colonnes et des lignes) a volontairement été retiré.

Pour pouvoir tester aisément différents décalages du cadre de lecture, on propose d’élaborer un outil : un programme traduisant les triplets d’une séquence ARN en une suite d’acides aminés. (il est en effet fastidieux de lire les séquences ARN en faisant varier les règles de lecture …).

La programmation se fera sous Python à partir de données du tableau ci-dessus à droite. Avec les instructions suivantes :

var=input("sequence :")

print (var)

var=len(chain)

var=chain [index:index+n]

While

If

ATTENTION : Mode d’indexation d’une chaine de caractère sous Python : le premier caractère a pour index 0

Indication facultative : Pour des questions d’optimisation (ou de beauté du code, comme on veut), on cherchera à minimiser pour un cycle de lecture les conditions If parcourus par l’exécution du programme (Il est peu judicieux de mettre autant de conditions que de ligne dans le tableau)

Nota : si les élèves l’ont vu dans un autre cours (mathématiques ? Sciences du numérique ?), on peut aussi proposer d’utiliser la fonction « liste » qui permet d’obtenir des codes plus compacts.

On peut aussi imaginer faire travailler les élèves sur une ligne voire sur une seule case du tableau

Enfin si les élèves ne programment pas sous python dans d’autres cours, on peut imaginer faire de l’analyse de code plutôt que du développement.

On notera que les différentes options (développement de tout ou partie, analyse de code …) ne sont pas exclusives et peuvent être choisie dans un même groupe classe dans le but de différencier les attendus en fonction du niveau des élèves en développement informatique….

Un exemple de code attendu (pour des raisons de place, c’est le code le plus compact qui a été recopié ici)

from lycee import \*

from tkinter import \*

import tkinter.font as tkFont

Invite = Tk()

Invite.configure(padx=20, pady=20, background="orange")

Invite.title("Séquence à traduire")

message=Label(Invite, text="Entrez ci dessous votre séquence d'ARNm")

message.configure(background="orange")

message.pack()

seq=StringVar()

saisie=Entry(Invite, textvariable=seq, width=30)

saisie.pack()

framebouton=Frame(Invite)

framebouton.configure(background="orange", padx=20, pady=10)

framebouton.pack()

bouton\_quitter = Button(framebouton, text="ENVOYER", command=Invite.destroy)

bouton\_quitter.pack()

Invite.mainloop()

seq=seq.get()

print (seq)

long=len(seq)

print("longueur ARN = ")

print (long)

longabs=floor(long/3)

print("longueur prévue de la protéine (si aucun codon STOP) = ")

print (longabs)

index=0

prot="Protéine : -"

codARN="ARNm : -"

aa=""

while index<long :

indexliste=0

codon=seq[index:index+3]

if len(codon)==3 :

codARN = codARN + codon + "-"

if codon[0:1]=="U" :

indexliste=indexliste+0

if codon[0:1]=="C" :

indexliste=indexliste+16

if codon[0:1]=="A" :

indexliste=indexliste+32

if codon[0:1]=="G" :

indexliste=indexliste+48

if codon[1:2]=="U" :

indexliste=indexliste+0

if codon[1:2]=="C" :

indexliste=indexliste+4

if codon[1:2]=="A" :

indexliste=indexliste+8

if codon[1:2]=="G" :

indexliste=indexliste+12

if codon[2:3]=="U" :

indexliste=indexliste+0

if codon[2:3]=="C" :

indexliste=indexliste+1

if codon[2:3]=="A" :

indexliste=indexliste+2

if codon[2:3]=="G" :

indexliste=indexliste+3

aa=liste[indexliste]

print (indexliste)

if aa!="STOP":

prot = prot + aa + "-"

longprot=longprot+1

**index=index+3**

else :

prot = prot + " " + " "

index=long

else :

index=long

print (codARN)

print (prot)

longprot=0

liste=["Phe","Phe","Leu","Leu","Ser","Ser","Ser","Ser","Tyr","Tyr","STOP","STOP","Cys","Cys","STOP","Trp","Leu","Leu","Leu","Leu","Pro","Pro","Pro","Pro","His","His","Gln","Gln","Arg","Arg","Arg","Arg","Ile","Ile","Ile","Met","Thr","Thr","Thr","Thr","Asn","Asn","Lys","Lys","Ser","Ser","Arg","Arg","Val","Val","Val","Val","Ala","Ala","Ala","Ala","Asp","Asp","Glu","Glu","Gly","Gly","Gly","Gly"

]

Resultat = Tk()

Resultat.title("Résultats")

Resultat.configure(background="black", padx=10, pady=10)

my\_font = "{courier new} 10 bold"

scroll=Scrollbar(Resultat, orient = HORIZONTAL)

cadre=Canvas(Resultat, background="orange", xscrollcommand = scroll.set)

cadre.pack()

scroll.pack()

scroll.config(command = cadre.xview)

codons\_ARN=Label(cadre, text=codARN, font=my\_font, padx=20, pady=10)

codons\_ARN.configure(background="#C0C0C0")

codons\_ARN.pack()

seq\_prot=Label(cadre, text=prot, font=my\_font, padx=20, pady=10)

seq\_prot.configure(background="#C0C0C0")

seq\_prot.pack()

framebouton=Frame(Resultat)

framebouton.config(background="black", padx=20, pady=10)

framebouton.pack()

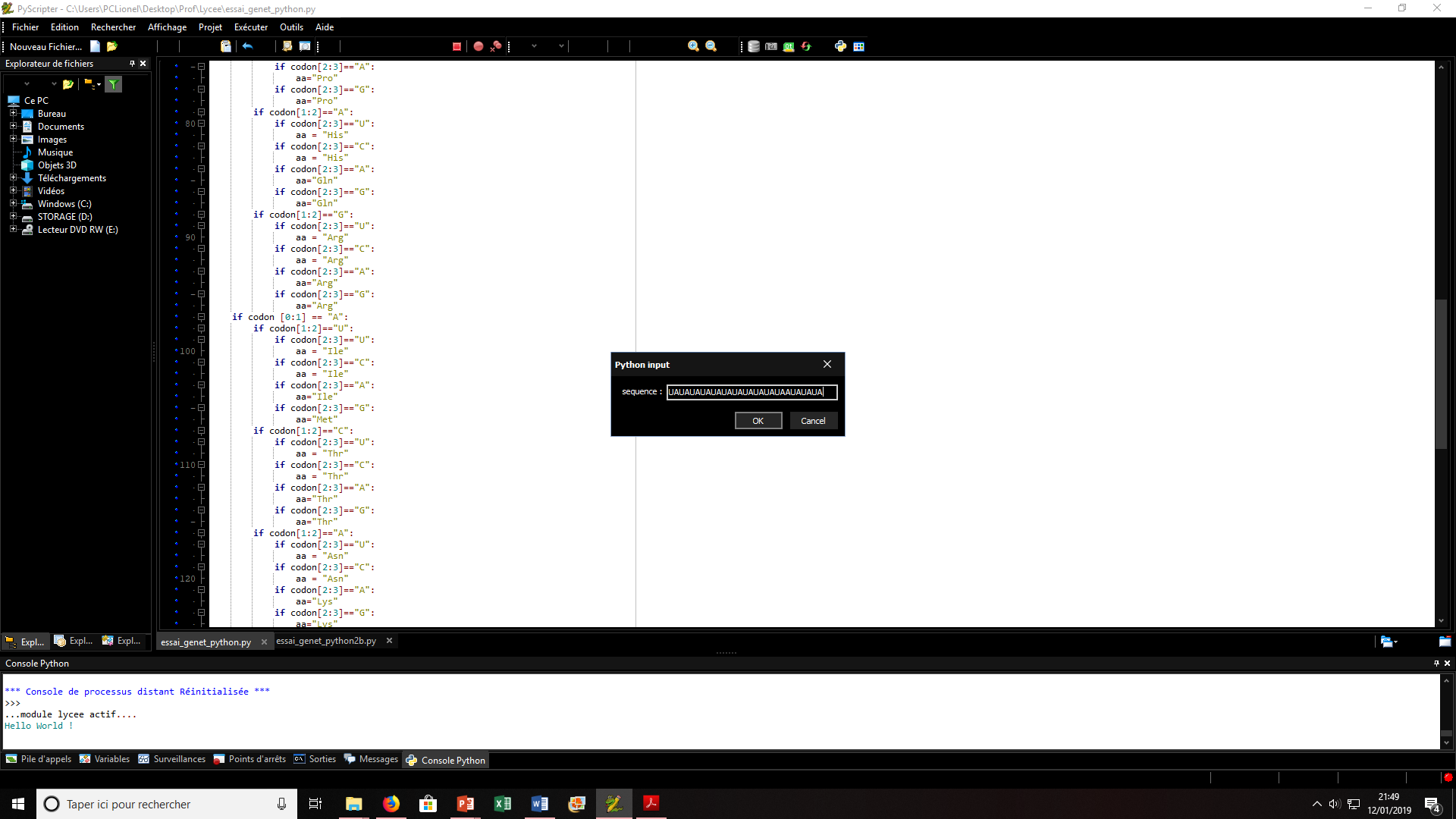
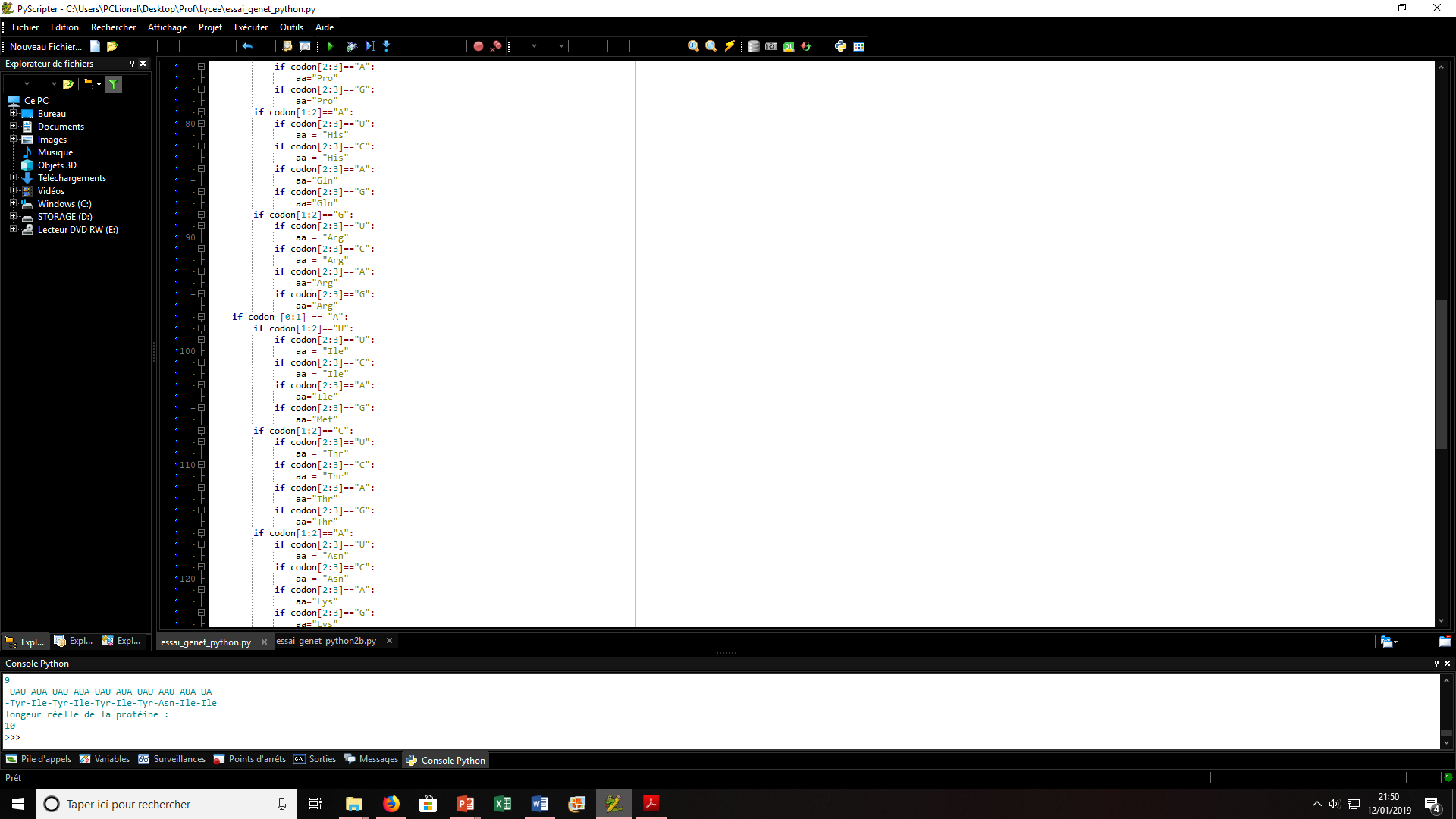
bouton\_quitter = Button(framebouton, text="QUITTER", command=Resultat.destroy)

bouton\_quitter.pack()

Resultat.mainloop()

Quel que soit le code il faut faire varier la valeur index (ligne encadrée) pour décaler la lecture sur la chaine de caractère « ARN ». on peut tester différentes solution (index + 1, +2, +3, + 11…)

Exemple d’affichage sous edu python : (saisie à gauche, résultat dans la console en bas à droite) –le code utilisé dans ces captures d’écran est une solution différente – mais fonctionnelle – que celle proposée plus haut)

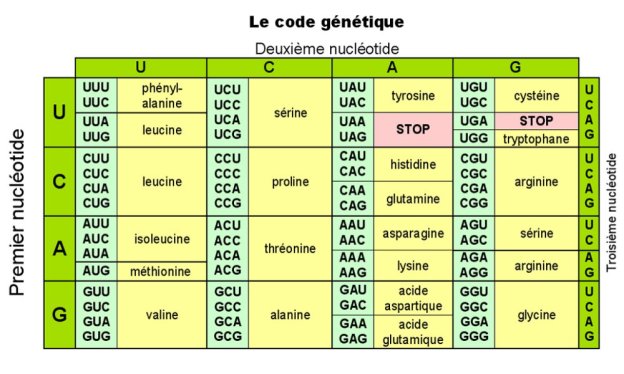
En faisant varier l’incrémentation de l’index dans un programme que le professeur aura vérifié fonctionnel, l’élève pourra confronter ses résultats à ceux des expériences de Niremberg et Matthaei.

L’élève remarquera que les codes chevauchant donnent des chaines d’acides aminés mais ce ne sont pas les bons …

Les codes non chevauchants avec intercalaires (index variant avec un n>3) ne rendent pas compte des résultats sur les poly UC par exemple.

La seule solution pour respecter les dernières lignes du tableau est de faire varier l’index de 3. L’ARNm est donc lue avec un cadre de lecture de 3 nucléotides, de façon continue et non chevauchante.

Lors du codage, on se rend compte de la redondance du code génétique et de la présence de codons stop. De plus un codage correct optimisé consiste finalement à retrouver le principe de lecture du code génétique (si première lettre du triplet = , alors ….).



Enfin, pour retourner à la biologie, on peut fournir aux élèves des documents complémentaires permettant de confirmer que le cadre de lecture se décale de 3 nucléotides :

|  |  |
| --- | --- |
| MET montrant les ribosomes qui scannent un ARNm (de la même façon que notre programme scanne une chaine de caractère) | LA structure des ARNt autorisent un cadre de lecture à 3 triplets (notion d’anticodon) |
| Enfin, la structure des ribosomes permet l’accrochage de 2 ARNT contigus sur l’ARNm (donc de deux cadres de lecture juxtaposés non chevauchants).  Modèle 3 D de la structure d’un ribosome associé à des ARNt et à un aRNm | |

On essaie donc ici que la programmation ne soit pas une fin en soi mais que l’élaboration du programme soit un moyen d’approcher certains aspects du fonctionnement de la traduction qui sont souvent trop implicites.

Toutefois cette démarche ne peut être couronnée de succès que si les élèves possèdent au préalable des notions de programmation. La conception du programme doit apparaitre comme un aide, une autre façon de raisonner sur la notion de correspondance entre deux séquences et non comme un obstacle supplémentaire. Dans le cas d’élèves n’utilisant pas ou très peu python dans d’autres cours une telle démarche serait par ailleurs bien trop chronophage.

Il serait en effet souhaitable que l’ensemble des activités proposées tiennent en un TP de 2h + une mise en commun d’une à deux heures maximum…