Traduction ADN-ARN-protéine

Avec l'interface Tk. Voir aussi le programme Traduction de l'ADN en séquence d'acides aminés (protéine) : utilisation d'un dictionnaire (type Python)

<sxh python; title : dictionaries_adn_arn_protein.py> #!/usr/bin/env python # -*- coding: utf-8 -*- “”“ Traduction de codes ADN en ARN et protéine. Basé sur le travail de CVDD, ba2 chimie 2013-2014 ”“” from Tkinter import * # permet d'importer et d'utiliser toutes les fonctions de la librairie Tkinter

nous allons définir la fonction “traduction en AM1” def traduction_en_AM1():

```
  gencode1 = {
  'ATA':'Y', 'ATC':'STOP', 'ATT':'STOP', 'ATG':'Y',
  'ACA':'C', 'ACC':'W', 'ACG':'C', 'ACT':'STOP',
  'AAC':'L', 'AAT':'L', 'AAA':'F', 'AAG':'F',
  'AGC':'S', 'AGT':'S', 'AGA':'S', 'AGG':'S',
  'CTA':'D', 'CTC':'E', 'CTG':'D', 'CTT':'E',
  'CCA':'G', 'CCC':'G', 'CCG':'G', 'CCT':'G',
  'CAC':'V', 'CAT':'V', 'CAA':'V', 'CAG':'V',
  'CGA':'A', 'CGC':'A', 'CGG':'A', 'CGT':'A',
  'GTA':'H', 'GTC':'Q', 'GTG':'H', 'GTT':'Q',
  'GCA':'R', 'GCC':'R', 'GCG':'R', 'GCT':'R',
  'GAC':'L', 'GAT':'L', 'GAA':'L', 'GAG':'L',
  'GGA':'P', 'GGC':'P', 'GGG':'P', 'GGT':'P',
  'TCA':'S', 'TCC':'R', 'TCG':'S', 'TCT':'R',
  'TTC':'K', 'TTT':'K', 'TTA':'N', 'TTG':'N',
  'TAC':'M', 'TAT':'I', 'TAA':'I', 'TAG':'I',
  'TGC':'T', 'TGT':'T', 'TGA':'T', 'TGG':'T',
  } # création d'un dictionnaire pr convertir chaque codon en acide aminé
  cha1 = entr1.get() # "cha1" va nous permettre d'exploiter les données
(séquence d'ADN) entrées dans l'entrée "entr1"
  tex2='' # "tex2" correspond à la réponse correspondant à la commande
"traduction en AM1"
  for n in range(0,len(cha1),3): # pour les entiers de la chaîne allant de 1
à 3 de la chaine "cha1"
      if gencode1.has_key(cha1[n:n+3]) == True: # utilisation du
dictionnaire gencode1, pour chaque codon (3 lettres), traduire en tex2
          tex2 += gencode1[cha1[n:n+3]]
  text2.configure( text= "Résultat : "+tex2,fg='blue') # pour déterminer ce
qui apparaitre dans la réponse
  return tex2 # l'instruction return définit ce que doit être la valeur
renvoyée par la fonction
```

def traduction_en_ARNm():

```
  gencode2 = {
  'ATA':'UAU', 'ATC':'UAG', 'ATT':'UAA', 'ATG':'UAC',
  'ACA':'UGU', 'ACC':'UGG', 'ACG':'UGC', 'ACT':'UGA',
  'AAC':'UUG', 'AAT':'UUA', 'AAA':'UUU', 'AAG':'UUC',
```

```
'AGC':'UCG', 'AGT':'UCA', 'AGA':'UCU', 'AGG':'UCC',
'CTA':'GAU', 'CTC':'GAG', 'CTG':'GAC', 'CTT':'GAA',
'CCA':'GGU', 'CCC':'GGG', 'CCG':'GGC', 'CCT':'GGA',
'CAC':'GUG', 'CAT':'GUA', 'CAA':'GUU', 'CAG':'GUC',
'CGA':'GCU', 'CGC':'GCG', 'CGG':'GCC', 'CGT':'GCA',
'GTA':'CAU', 'GTC':'CAG', 'GTG':'CAC', 'GTT':'CAA',
'GCA':'CGU', 'GCC':'CGG', 'GCG':'CGC', 'GCT':'CGA',
'GAC':'CUG', 'GAT':'CUA', 'GAA':'CUU', 'GAG':'CUC',
'GGA':'CCU', 'GGC':'CCG', 'GGG':'CCC', 'GGT':'CCA',
'TCA':'AGU', 'TCC':'AGG', 'TCG':'AGC', 'TCT':'AGA',
'TTC':'AAG', 'TTT':'AAA', 'TTA':'AAU', 'TTG':'AAC',
'TAC':'AUG', 'TAT':'AUA', 'TAA':'AUU', 'TAG':'AUC',
'TGC':'ACG', 'TGT':'ACA', 'TGA':'ACU', 'TGG':'ACC',
}
cha1 = entr1.get()
tex2=''
for n in range(0,len(cha1),3):
    if gencode2.has_key(cha1[n:n+3]) == True:
        tex2 += gencode2[cha1[n:n+3]]
text4.configure( text= "Résultat : "+tex2,fg='blue')
```

def traduction_en_AM2():

```
gencode3 = {
'ATA':'TYR-', 'ATC':'STOP-', 'ATT':'STOP-', 'ATG':'TYR-',
'ACA':'CYS-', 'ACC':'TRP-', 'ACG':'CYS-', 'ACT':'STOP-',
'AAC':'LEU-', 'AAT':'LEU-', 'AAA':'PHE-', 'AAG':'PHE-',
'AGC':'SER-', 'AGT':'SER-', 'AGA':'SER-', 'AGG':'SER-',
'CTA':'ASP-', 'CTC':'GLU-', 'CTG':'ASP-', 'CTT':'GLU-',
'CCA':'GLY-', 'CCC':'GLY-', 'CCG':'GLY-', 'CCT':'GLY-',
'CAC':'VAL-', 'CAT':'VAL-', 'CAA':'VAL-', 'CAG':'VAL-',
'CGA':'ALA-', 'CGC':'ALA-', 'CGG':'ALA-', 'CGT':'ALA-',
'GTA':'HIS-', 'GTC':'GLN-', 'GTG':'HIS-', 'GTT':'GLN-',
'GCA':'ARG-', 'GCC':'ARG-', 'GCG':'ARG-', 'GCT':'ARG-',
'GAC':'LEU-', 'GAT':'LEU-', 'GAA':'LEU-', 'GAG':'LEU-',
'GGA':'PRO-', 'GGC':'PRO-', 'GGG':'PRO-', 'GGT':'PRO-',
'TCA':'SER-', 'TCC':'ARG-', 'TCG':'SER-', 'TCT':'ARG-',
'TTC':'LYS-', 'TTT':'LYS-', 'TTA':'ASN-', 'TTG':'ASN-',
'TAC':'MET-', 'TAT':'ILE-', 'TAA':'ILE-', 'TAG':'ILE-',
'TGC':'THR-', 'TGT':'THR-', 'TGA':'THR-', 'TGG':'THR-',
}
cha1 = entr1.get()
tex2=''
for n in range(0,len(cha1),3):
    if gencode3.has_key(cha1[n:n+3]) == True:
        tex2 += gencode3[cha1[n:n+3]]
text3.configure( text= "Résultat : "+tex2,fg='blue')
```

fen1 = Tk() # pour créer et nommer la fenêtre fen1.title(“Traduction d'une séquenced'ADN en acides

aminés") fen1.geometry('800×450') fen1.configure(bg='white')

text1 = Label (fen1, text = "Veuillez introduire la séquence d'ADN :",fg='red', background='White') text2 = Label (fen1, text = "",background='white') text3 = Label (fen1, text = "",background='White') text4 = Label (fen1, text = "",background='White') text01 = Label (fen1, text = "",background='White') text02 = Label (fen1, text = "",background='White') text03 = Label (fen1, text = "",background='White') text04 = Label (fen1, text = "",background='White') text05 = Label (fen1, text = "",background='White') text5 = Label (fen1, text = " Vous pouvez consulter La DataBank regroupant toutes les protéines afin de voir si votre protéine est déja répertoriée : ", background='white',fg='green') text6 = Label (fen1, text = "http://www.rcsb.org/pdb/home/home.do",background='white',fg='green') entr1 = Entry(fen1, background='grey') button1 = Button(fen1, text='Structure primaire de la protéine', command =traduction_en_AM1) button2 = Button(fen1, text='Structure primaine de la protéine (code 3 lettres)', command =traduction_en_AM2) button3 = Button(fen1, text='Traduire ADN en ARNm', command =traduction_en_ARNm) button4 = Button(fen1, text='Quitter', command=fen1.destroy) text1.grid(row=2, column=2) text01.grid(row=3, column=2) text02.grid(row=5, column=2) text03.grid(row=8, column=2) text04.grid(row=11, column=2) text05.grid(row=14, column=2) text2.grid(row=13, column=2) text3.grid(row=10, column=2) text4.grid(row=7, column=2) text5.grid(row=15, column=2) text6.grid(row=16, column=2) entr1.grid(row=4, column=2) button1.grid(row=12, column=2) button2.grid(row=9, column=2) button3.grid(row=6, column=2) button4.grid(row=17, column=2)

fen1.mainloop() # lance la boucle principale </sxh>