En este programa vamos testear los máximos adyacentes comunes a dos cadenas de ADN.

Creamos una clase llamada Nucleótidos que se inicializa

con la cadena de "nomenclaturas" o iniciales de la tabla nucleótidos

tiene dos métodos:

1ª método RndAdn crea dos secuencias de ADN aleatorias de longitud 13

2ª método Nu_comparate compara los nucleótidos posibles

en las dos secuencias de ADN dando como resultado

los adyacentes o comunes entre sí de máxima longitud

def __init__(self):

Inicializamos una tabla con los nucleótidos y otra sus las nomenclaturas

con la cual trabajaremos para la comparación de las cadenas de ADNs

```
13
14      def __init__(self):
15          self.nucleotidos=["Adenina","Citosina","Guanina","Timina"]
16          self.nomcl=self.Iniciales()
17
18          #creamos una tabla con las nomenclaturas de la tabla nucleotidos
19          #con la cual trabajaremos para la comparacion de las cadenas de adns
20      def Iniciales(self):
21          nomcl=""
22          for i in range(len(self.nucleotidos)):
23              #llamada a la primera letra de  tabla nucleotidos
24              nom=next(self.Nomenclatura(self.nucleotidos[i]))
25              nomcl+=nom#acumulamos cada inicial
26          return(nomcl)
27
28      def Nomenclatura(self,nucl):
29          # devuelve la primera inicial de la lista nucleotidos
30          for n in nucl:
31                  yield from n
32
```

def Iniciales(self):

por cada campo de tabla nucleótidos llamamos a Nomenclatura(campo)

def Nomenclatura(self,nucl):

devuelve la primera inicial de cada nucleótido.

def RndAdn(self):

```
32
33      # funcion que crea las dos secuencias aleatorias de ADN de long 13
34      def RndAdn(self):
35          c=self.nomcl
36          adn=""
37          for n in range(13):#hasta rango 13
38              a=random.choice(c)#elige una letra cualquiera de c(self.nomencl)
39              adn+=a#acumula la letra de la var(a) creando secuencia en adn
40          #devolvemos una secuencia de adn rango 13
41          return(adn)
42
```

función que crea las dos secuencias aleatorias de ADN de Long 13

hasta rango 13 elige una letra cualquiera de c(self.nomencl)

acumulo la letra creando secuencia en ADN

devolvemos una cadena de secuencia de ADN rango 13

def Nu_comparate(self, adn1,adn2):

```
43      #comparacion de las dos cadenas de ADN
44      def Nu_comparate(self, adn1,adn2):
45          comunes = set(adn1).intersection(set(adn2))
```

Comparación de las dos cadenas de ADN:

Calculo los nucleótidos k hay en común en cada cadena de ADN ya que puede haber algún nucleótido k no se repite en ambas cadenas.

```
52              for c in comunes:#por cada nucleotido en comun:
53                  #print(c)
54                  l_ady1+=adn1.count(c)#num por nucleotidos comunes en adn1
55                  l_ady2+=adn2.count(c)#num por nucleotidos comunes en adn2
56                  l_ady=min(l_ady1,l_ady2)#long max de secuencia=minima en comun
57                  ady_l+=c#cadena letras de nucleotidos
58                  for l in range(l_ady):#en el rango de secuencia maxima en comun
59                  #hallamos las posibles combinaciones de nucleotidos (ady_l)
60                  #desde cont=2 parejas hasta que cont=l_ady (rango de secuencia maxima en comun)
61                      pos_comb_ady=list(itertools.product(ady_l, repeat=cont))
62                  cont+=1
```

Calculo la cual es la longitud mínima de nucleótidos comunes, repetidos en ambas cadenas de ADN.

No habrá una combinación mayor, que sea igual, a la mínima del contenido de nucleótidos comunes en ambos ADNs .

Por ejemplo:

Si los comunes son: GTA es decir, Citosina no es común en ambos ADNs

ADN1 tiene 6 totales de GTA

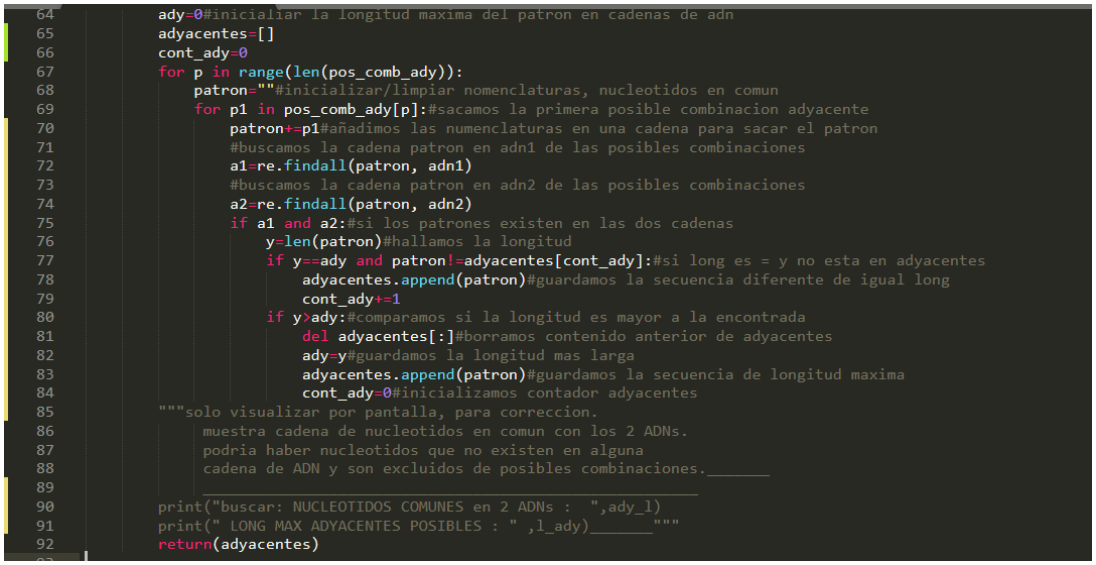
ADN2 tiene 4 totales de GTA

La mayor longitud de posibles combinaciones de nucleótido adyacentes en común nunca será superior a la longitud mínima de los nucleótidos en común de ambos

ADNs. En este caso 4

En el rango, este caso 4 guardo en posibles combinaciones de adyacentes, las combinaciones posibles desde cont= 2 por pares , hasta 4 de longitud.

Recorremos los campos de posibles combinaciones adyacentes y los guardamos en forma de cadena (patrón).



```
        ady=0#inicializar la longitud maxima del patron en cadenas de adn
        adyacentes=[]
        cont_ady=0
        for p in range(len(pos_comb_ady)):
            patron=""#inicializar/limpiar nomenclaturas, nucleotidos en comun
            for p1 in pos_comb_ady[p]:#sacamos la primera posible combinacion adyacente
                patron+=p1#añadimos las numenclaturas en una cadena para sacar el patron
                #buscamos la cadena patron en adn1 de las posibles combinaciones
                a1=re.findall(patron, adn1)
                #buscamos la cadena patron en adn2 de las posibles combinaciones
                a2=re.findall(patron, adn2)
                if a1 and a2:#si los patrones existen en las dos cadenas
                    y=len(patron)#hallamos la longitud
                    if y==ady and patron!=adyacentes[cont_ady]:#si long es = y no esta en adyacentes
                        adyacentes.append(patron)#guardamos la secuencia diferente de igual long
                        cont_ady+=1
                    if y>ady:#comparamos si la longitud es mayor a la encontrada
                        del adyacentes[:]#borramos contenido anterior de adyacentes
                        ady=y#guardamos la longitud mas larga
                        adyacentes.append(patron)#guardamos la secuencia de longitud maxima
                        cont_ady=0#inicializamos contador adyacentes
        """solo visualizar por pantalla, para correccion.
             muestra cadena de nucleotidos en comun con los 2 ADNs.
             podria haber nucleotidos que no existen en alguna
             cadena de ADN y son excluidos de posibles combinaciones._______
                                                                    _____
        print("buscar: NUCLEOTIDOS COMUNES en 2 ADNs :  ",ady_1)
        print(" LONG MAX ADYACENTES POSIBLES : " ,l_ady)_______"""
        return(adyacentes)
```

Buscamos el patrón en ambas combinaciones y si los patrones en ambas combinaciones son iguales, hallamos la longitud del patrón y comparamos con la longitud mayor encontrada.

If ADN1(patron) y ADN2(patron) son encontrados en los dos casos y no está en adyacentes:

> si hay varios de Long igual, guardamos las diferentes cadenas en adyacentes
>
> si es la long es mayor, inicializamos la cadena de adyacentes con la secuencia de nucleótidos de longitud máxima.

Devuelve la lista de cadenas de adyacentes