Desafío 3

Diego Rodríguez Soto

Ing. Civil. Informática, Universidad de Aysén

Taller de Ingeniería

Prof. Gabriel Núñez

19 de diciembre

Objetivos

- Aprender manipulaciones de Archivos como lectura remplazos, etc.
- Uso de diccionarios para poder darle un valor u otro a una secuencia o variable
- Conocer manipulación de Strings y listas

En el siguiente informe veremos el tercer y último desafío de programación del semestre en el cual revisaremos los requerimientos paso a paso de cómo fue abordado para lograr cada uno de estos, El desafío consistió en leer un archivo fasta que contenía una secuencia de nucleótidos y luego esta secuencia convertirla en la secuencia de la proteína correspondiente luego imprimir las estadísticas de cada nucleótidos y codones después hacer una pequeña investigación de los aminoácidos polares positivos, apolares, polares negativos y polares sin carga e indicar cuantos en la secuencia habían y para finalizar el desafío nos pidió un diagrama de relación de aminoácido.

Requerimiento 1.1

Este requerimiento solicitaba que se pudiera leer un archivo ingresado por el usuario el que contiene la información de la genética luego aplicar la eliminación de intrones que serian los tramos desde una “I” hasta una “F”, también el reemplazo de “T” por “U” después la secuencia debía ser separada por codones o tripletes los cuales son importantes para saber desde donde se empieza a contar la secuencia que sería desde “AUG”.

Lo primero que hice fue crear un diccionario donde las **Keys** van a ser los tripletes(codones) y los **values** que serán la letra de cada **codón**, luego procedo a pedir el nombre del archivo fasta y abrirlo con la función **Open (nombre archivo)** después creo una lista llamada genética limpia en la cual se guardara la genética normal de la secuencia donde se guardara la genética con todas las eliminaciones de intrones y reemplazos de letras

```
codigog={"GCU":"A","GCC":"A","GCA":"A","GCG":"A","CGU
         "AAU":"N","AAC":"N","GAU":"D","GAC":"D","UGU
         "GAA":"E","GAG":"E","GGU":"G","GGC":"G","GGA
         "AUU":"I","AUC":"I","AUA":"I","UUA":"L","UUG
         "AAA":"K","AAG":"K","AUG":"M","UUU":"F","UUC
         "UCU":"S","UCC":"S","UCA":"S","UCG":"S","AGU
         "UGG":"W","UAU":"Y","UAC":"Y","GUU":"V","GUC
genetica=open(input("Ingrese nombre de secuencia:"))
geneticalimpia=[]
```

El siguiente paso es leer el archivo con la función **Readlines** la cual nos permite leer todas las líneas de un archivo, pero en este caso le indicamos que solo lea la primera línea con **Readlines(1),** ya que en esta se suele encontrar el nombre de la secuencia por lo que no nos sirve leerlo todo de una vez o sino también haría cambios en esta parte, luego procedemos a leerlo todo con la función **Read ().**

```
nombre=genetica.readlines(1)
linea=genetica.read()
```

Requerimiento 1

El segundo paso es eliminar los tramos de “I” ”F” incluyéndolos para esto se utiliza una condicional **while** en la que empezaremos una variable **i** en 0 y va a ser un mientras i sea menor al largo de la secuencia de archivo esta se iría sumando un uno y así recorrer toda la secuencia luego con un **if** de la posición **[i]** era igual a una “I” entraríamos en otro while para que esta i se valla sumando desde la posición “I” hasta que sea distinto de “F” y al momento de encontrar este tramo estos serían salteados y la secuencia añadida a la lista genética limpia.

```
i=0
while i<len(linea):
    if linea[i]=="I":
        while linea[i]!="F":
            i+=1
    else:
        geneticalimpia+=linea[i]
    i+=1
```

luego pasamos a reemplazar las “T” por “U” esto se hizo con el ciclo **for** que recorrería el rango del largo de la lista genética limpia a la que ya se le realizo las eliminaciones de los tramos “I” “F” entonces usaremos un **IF** que si se encuentra una “T” esta sería reemplazada por una “U” después a través de la función **.Join** que nos permite modificar una lista y convertirla a texto e imprimimos la secuencia sin “I”,”F” y reemplazando las “T” por “U”.

```
for j in range(len(geneticalimpia)):
    if geneticalimpia[j]=="T":
        geneticalimpia[j]="U"
geneticalimpia="".join(geneticalimpia)
print("Sin IF:",geneticalimpia)
```

```
Sin IF: AUGAAGCUCGGCAAGAACGACCGGUCCAUGGACAUCGAGGCCCGGCUCUAACCCUUC
```

Para empezar a contar desde el codón start que es “AUG” y luego parar de contar tripletes cuando encuentre alguna de estos tres tripletes “UAG”,”UAA” y “UGA” primero se crearon dos listas una llamada start que lo único que incluía era el triplete “AUG” y otra lista llamada stop1 que incluirá los tres tripletes de stop de la secuencia “UAG”,”UAA” y “UGA” después se creo una variable llamada largogeneticalimpia que me indicaba el largo de la secuencia y también una lista vacía llamada codones donde se guardaran todos los tripletes de la secuencia primero se inicio una **i** en 0 luego con un ciclo **while** del largo de la genética limpia y si **start[0]** era igual a la geneticalimpia esta empezará a contar desde la posición en que encuentre el triplete de 3 en 3 y si no esta sumara uno a **i** hasta encontrar “AUG” y lo que había antes no lo contara y para finalizar se hará lo mismo con el **stop1** y es que cuando encuentre alguno de los tres este dejara de contar y no contara el triplete de stop luego con la función **.append** que es para agregar elemento a una lista añadiremos los tripletes contados a la lista llamada codones por ultimo imprimo la lista con todos los tripletes.

```
start=["AUG"]
stop1=["UAG","UAA","UGA"]
largogeneticalimpia=len(geneticalimpia)
codones=[]
i=0
while i<largogeneticalimpia-2:
    if start[0]==geneticalimpia[i:i+3]:
        while (i+3)<=len(geneticalimpia):
            if stop1[0]==geneticalimpia[i:i+3] or stop1[1]==geneticalimpia[i:i+3] or stop1[2]==geneticalimpia[i:i+3]:
                i+=largogeneticalimpia
            else:

                codones.append(geneticalimpia[i:i+3])
                i+=3
    i+=1
print("Tripletes:",codones)
```

```
Tripletes: ['AUG', 'AAG', 'CUC', 'GGC', 'AAG', 'AAC', 'GAC', 'CGG', 'UCC', 'AUG', 'GAC', 'AUC', 'GAG', 'GCC', 'CGG', 'CUC']
```

El ultimo paso de este requerimiento que solicitaba pasar los tripletes a sus respectivas letras y crear un archivo nuevo con la secuencia final de la proteína para esto se creo nuevamente otra lista llamada cambio que estará vacía luego creamos otra variable llamada largo que será el largo de la lista codones después con un ciclo **while** empezamos nuevamente una **i** en 0 y que recorra la variable llamada largo luego esta si esta en el diccionario antes definido llamamos una variable encontrar que básicamente con la función **.get** buscara la **Key** encontrada en el largo de los codones y este será agregado a la lista cambio con la función **append** para finalizar creamos el archivo nuevo con la secuencia de proteínas con la función “w” **write** esta nos permite crear un archivo con lo que deseemos o agregarlo a un archivo que no lo tenga.

```
cambio=[]
largo=len(codones)
i=0
while i<largo:
    if codones[i] in codigog:
        encontrar=codigog.get(codones[i])
        i+=1
        cambio.append(encontrar)
print("".join(cambio))
with open("secuencia.fasta","w") as f:
    f.write(nombre[0])
    f.write("".join(cambio))
```

Requerimiento 2

Este requerimiento nos solicita las estadísticas de nucleótidos, codones y numero de aminoácidos por secuencia válida primero creamos un contador con todos los nucleótidos “A”,”C”,” U”,” G” todas empezando en 0 luego recorremos el largo de la genéticalimpia y cada vez que se encentre una “A”,”C”,” U”,” G” se les sumará 1 a cada uno después para sacar el porcentaje el total de cada uno será multiplicado por 100 y dividido por el largo de la genéticalimpia.

```
A=0
C=0
U=0
G=0
largo=len(geneticalimpia)
for i in geneticalimpia:
            if i=="A":
                A+=1

            if i=="C":
                C+=1

            if i=="U":
                U+=1

            if i=="G":
                G+=1

porcentajeA=(A*100)/largo
porcentajeC=(C*100)/largo
porcentajeU=(U*100)/largo
porcentajeG=(G*100)/largo
```

Para contar los codones por secuencia valida se definió una función llamada **contarcod(codonestotal, p)** luego con un contador en 0 recorrerá la lista codones y cada vez que encuentre codones iguales estos se irán sumando después para sacar el porcentaje esta será multiplicada por 100 y dividida por el largo de codones.

```
def contarcodones(codonestotal,p):
    contador=0
    for i in codonestotal:
        if i ==p:
            contador+=1
    return contador
```

```
for i in codones:
    contadorcod=contarcodones(codones,i)
    print("El codon",i,"se encuentra",contadorcod,"en la secuencia y su porcentaje es:",(contadorcod*100)/len(codones),"%")
```

Para finalizar con las estadísticas de aminoácidos se creo las variables de aminoácidos comenzadas en 0 luego se crearon listas con las respectivas letras de aminoácidos, con un ciclo **for i in cambio** recorremos la lista cambio que contiene la secuencia de proteínas si la i encuentra alguna de las letras de las listas a los contadores de aminoácidos se les sumara uno.

```
polarespositivos=0
polaresnegativos=0
polaressincarga=0
apolares=0
sincarga=["S","T","C","Y","N","Q"]
negativos=["D","E"]
positivos=["H","R","K"]
apolar=["G","A","V","L","I","F","W","M","P"]

for i in cambio:
    if i in positivos:
        polarespositivos+=1

    elif i in negativos:
        polaresnegativos+=1
    elif i in apolar:
        apolares+=1
    elif sincarga:
        polaressincarga+=1
```

Requerimiento 3

El ultimo requerimiento solicitado fue generar un diagrama de relación de aminoácido para esto importamos la librería de **Graphviz** y **sys** luego definimos unas funciones que nos ayudaran a crear el diagrama donde configuraremos los colores de las líneas donde serán negras las que unan

los tripletes y azul las letras de la secuencia de proteínas, en este caso usaremos el largo de la lista de codones y el diccionario creado al principio para agregar los nodos.

```
def agregarNodo(nombreA,nombreB):
    gr.add_edge((nombreA,nombreB))

def agregarNodoAzul(nombreA,nombreB):
    gr.add_edge(nombreA,nombreB,color='blue')
```

```
gr = pgv.AGraph(rotate='360',bgcolor='white',directed=True)
gr.graph_attr['label']='Codon y Aminoacido tres letras'
# se agregan nodos.
i = 1
while (i < len(codones)):
    temp = codones[i-1]
    agregarNodoAzul(temp,codigog[temp])
    #agregarNodoAzul(dic[temp],dic2[dic[temp]])
    agregarNodo(temp,codones[i])
    i = i + 1
# Generación de grafo en formato PNG.
gr.layout(prog='dot')
gr.draw('codonyAA.png')
```

Para concluir se conoció como crear y usar un diccionario, como leer un archivo línea por línea o todas las líneas del archivo, cambiar o crear un archivo con datos determinados, luego como a través de diferentes ciclos el poder recorrer listas donde podemos eliminar tramos de un **string**, reemplazar letras por otras y empezar a leer desde una parte determinada y detenerlo al momento de encontrar alguna letra o palabra, por otro lado a sacar estadísticas basadas en elementos de una lista y poder generar un gráfico de una lista con distintos elementos y como se enlazan entre los mismos, para finalizar este Desafío nos ayudó a conocer los diccionarios y archivos los que más tardes nos ayudaran a ingresar a algún valor a través de las **Keys** de un diccionario.