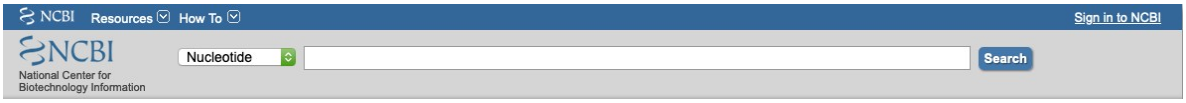
Python y un biólogo principiante – Parte 1

I. ¿Dónde podemos conseguir datos?

Una de las maneras más comunes para obtener secuencias genéticas es en la página de NCIB donde se puede encontrar gran parte de los genomas de organismos registrados hasta la fecha. En nuestro caso que buscamos una secuencia de Nucleótidos vamos a nuestra pestañita en la esquina superior izquierda para seleccionar que dentro de TOODA la base de datos, nos muestre solo la dedicada a nucleótidos: Nucleotide.



Nota: Aún buscando el nombre completo de una secuencia en específico te puede tirar miles de resultados. Por ejemplo, si queremos buscar "dUTPase E Coli" es más rápido escribir con especificaciones extra para que el buscador nos filtre resultados. Escribiendo "dUTPase E AND Coli NOT genome NOT chromosome NOT complete" reducimos enormemente la lista de secuencias con coincidencias en el nombre. Usar "AND" y "NOT" son esenciales para poder encontrar las secuencias problema sin gastar mucho tiempo en ello.

II. ¿Cómo acomodo estos datos?

Primero que nada, ten claro qué es lo que quieres hacer/obtener de estos datos que se te van a presentar. Una vez claro eso, empezaremos por escoger una plataforma en la que trabajar, en este caso va a ser Python y Jupyter Notebook. Normalmente para trabajar los datos obtenidos en la primera parte se utiliza el formato FASTA

¿y qué es el formato FASTA?

Es un tipo de archivo de lo más simple que hay en bioinformática, generalmente constando de un ">", un código de identificación único y una descripción, seguido de una secuencia de letras que no es más que la secuencia de nucleótidos (o péptidos en dado aso). Por ejemplo, la secuencia de nucleótidos de la Superóxido Dismutasa de *E. coli*:

```
>X03951.1 E. coli Mn-superoxide dismutase gene
TCGGGCATTTTCCTGCAAAACCATACCCTTACGAAAAGTACGGCATTGATAATCATTTTCAATATCATTT
AATTAACTATAATGAACCAACTGCTTACGCGGCATTAACAATCGGCCGCCCGACAATACTGGAGATGAAT
ATGAGCTATACCCTGCCATCCCTGCCGTATGCTTACGATGCCCTGGAACCGCACTTCGATAAGCAGACCA
TGGAAATCCACCACACCAAACACCATCAGACCTACGTAAACAACGCCAACGCGGCGCTGGAAAGCCTGCC
AGAATTTGCCAACCTGCCGGTTGAAGAGCTGATTACCAAACTGGACCAGCTGCCAGCAGACAAGAAAACC
GTACTGCGCAACAACGCTGGCGGTCACGCTAACCACAGCCTGTTCTGGAAAGGTCTGAAAAAAGGCACCA
CCCTGCAGGGTGACCTGAAAGCGGCTATCGAACGTGACTTCGGCTCCGTTGATAACTTCAAAGCAGAATT
```

TGAAAAAGCGGCAGCTTCCCGCTTTGGTTCCGGCTGGGCATGGCTGGTGCTGAAAGGCGATAAACTGGCG
GTGGTTTCTACTGCTAACCAGGATTCTCCGCTGATGGGTGAAGCTATTTCTGGCGCTTCCGGCTTCCCGA
TTATGGGCCTGGATGTGTGGGAACATGCTTACTACCTGAAATTCCAGAACCGCCGTCCGGACTACATTAA
AGAGTTCTGGAACGTGGTGAACTGGGACGAAGCAGCGGCACGTTTTGCGGCGAAAAAATAATCATTTGCC
GCCTGCTGCAATGAGGCGTATAGGCCGCATATCAGCTTAAAAAATGAACCATCGCCAACGGCGGTGGTTT
TTTTGTGATCAATTTCAAAATAAAAACAATGATCCGAATAAAAATAAAACAGCGTTTCAATTGATGTGGT
TTTGACACTTTTATGATTAAATGAATGTCTATCTTCGTTTCCATCAACACTGATGCTCCATTGAGGAATT
ACGCATCAGCCCTTAAAAATATGCCGACAGGTGATGGAAATGCAGATAAAACGCTCGATTGAGAAAATCC
CGG

Ok ya tenemos el formato este, ya tengo Python y Jupyter instalado ¿qué sigue, master?

¡Atención aquí que se pone intenso!

Esta secuencia que tenemos arribita, en formato FASTA, la guardaremos en este ejemplo en un formato de texto simple, oseaseme ".txt". En la parte derecha superior de tu secuencia, en la página del NCBI, puedes ver un botón que dice "Send To:" donde te da opciones de **exportar** la secuencia en cuestión. Escoges "File" y seleccionas el formato FASTA, de modo que descargues un documento formato .txt con el nombre que puede o no ser el nombre de tu secuencia, pon atención a qué nombre le pone.

Ubica tu archivo en tus documentos y copia la dirección del directorio dentro de tu variable de elección.

1. Carga los datos en Jupyter asignándole una variable, del nombre que quieras. Pongámosle "x",

```
x = open(r '/Users/mac/Downloads/sequence.fasta', 'r')
a = x.read()
x.close()
```

Nota: Después de la dirección de directorio no olvides especificar el nombre de el archivo con el nombre completo con el tipo de archivo. En mi caso fue "sequence.fasta.txt" en un SO Windows. En Mac parece no ser necesario incluir el tipo de archivo, pues basta con especificar el nombre del archivo sin la extensión.

2. Comprueba que tu variable a la que le asignaste la tarea de leer tu archivo, de hecho saque lo que buscas. Deberías obtener algo asi:

```
In [4]: a

Out[4]: '>X03951.1 E. coli Mn-superoxide dismutase gene\nTCGGGCATTTTCCTGCAAAACCATACCCTTACGAAAAGTACGGCATTGATAATCATTTTCAATATCAT
TT\nAATTAACTATAATGAACCAACTGCTTACGCGGCATTAACAATCGGCCGCCCGACAATACTGGAGATGAAT\nATGAGCTATACCCTGCCATCCCTGCCGTATGCTTACGATGC
CCTGGAACCGCACTTCGATAAGCAGACCA\nTGGAAATCCACCACACCAAACACCATCAGACCTACGTAAACAACGCCAACGCGGCGCTGGAAAGCCTGCC\nAGAATTTGCCAACC
TGCCGGTTGAAGAGCTGATTACCAAACTGGACCAGCTGCCAGCAGACAAGAAAACC\nGTACTGCGCAACAACGCTGGCGGTCACGCTAACCACAGCCTGTTCTGGAAAGGTCTGAA
AAAAGGCACCA\nCCCTGCAGGGTGACCTGAAAGCGGCTATCGAACGTGACTTCGGCTCCGTTGATAACTTCAAAGCAGAATT\nTGAAAAAGCGGCAGCTTCCCGCTTTGGTTCCG
GCTGGGCATGGCTGGTGCTGAAAGGCGATAAACTGGCG\nGTGGTTTCTACTGCTAACCAGGATTCTCCGCTGATGGGTGAAGCTATTTCTGGCGCTTCCGGCTTCCCGA\nTTATG
GGCCTGGATGTGTGGGAACATGCTTACTACCTGAAATTCCAGAACCGCCGTCCGGACTACATTAA\nAGAGTTCTGGAACGTGGTGAACTGGGACGAAGCAGCGGCACGTTTTGCGG
CGAAAAAATAATCATTTGCC\nGCCTGCTGCAATGAGGCGTATAGGCCGCATATCAGCTTAAAAAATGAACCATCGCCAACGGCGGTGGTTT\nTTTTGTGATCAATTTCAAAATAA
AAACAATGATCCGAATAAAAATAAAACAGCGTTTCAATTGATGTGGT\nTTTGACACTTTTATGATTAAATGAATGTCTATCTTCGTTTCCATCAACACTGATGCTCCATTGAGGAA
TT\nACGCATCAGCCCTTAAAAATATGCCGACAGGTGATGGAAATGCAGATAAAACGCTCGATTGAGAAAATCC\nCGG\n\n'
```

Nota que te sale el texto completo, con el nombre, titulo y demás identificadores que te mencioné antes. Incluso un \n que no es más que la representación de un “enter” que comúnmente no vemos en procesadores de texto. En archivos tipo FASTA cada 60 caracteres empieza una nueva línea, osea que cada 60 letras hay un \n.

Ya ¿ahora qué? ¿Puedo dejar el título y los espaciados (\n)?
No tengo idea. Por lo mientras practiquemos quitarlos y dejar nomas la secuencia a trabajar.

- *Quitando el título:* Cuéntale cuántos caracteres tiene el título y quítaselos. De eso va este ejercicio, y es tan sencillo como suena. Solo especifiquemos a Jupyter desde qué numero de carácter queremos trabajar y asígnale una nueva variable, para que cuando queramos trabajar solo la secuencia trabajo, no tengamos que escribirle DE NUEVO desde dónde queremos trabajar.

```
In [3]: b = a[47:]

In [4]: b

Out[4]: 'TCGGGCATTTTCCTGCAAAACCATACCCTTACGAAAAGTACGGCATTGATAATCATTTTCAATATCATTT\nAATTAACTATAATGAACCAA
CTGCTTACGCGGCATTAACAATCGGCCGCCCGACAATACTGGAGATGAAT\nATGAGCTATACCCTGCCATCCCTGCCGTATGCTTACGATGC
CCTGGAACCGCACTTCGATAAGCAGACCA\nTGGAAATCCACCACACCAAACACCATCAGACCTACGTAAACAACGCCAACGCGGCGCTGGAA
AGCCTGCC\nAGAATTTGCCAACCTGCCGGTTGAAGAGCTGATTACCAAACTGGACCAGCTGCCAGCAGACAAGAAAACC\nGTACTGCGCAA
CAACGCTGGCGGTCACGCTAACCACAGCCTGTTCTGGAAAGGTCTGAAAAAAGGCACCA\nCCCTGCAGGGTGACCTGAAAGCGGCTATCGAA
CGTGACTTCGGCTCCGTTGATAACTTCAAAGCAGAATT\nTGAAAAAGCGGCAGCTTCCCGCTTTGGTTCCGGCTGGGCATGGCTGGTGCTGA
AAGGCGATAAACTGGCG\nGTGGTTTCTACTGCTAACCAGGATTCTCCGCTGATGGGTGAAGCTATTTCTGGCGCTTCCGGCTTCCCGA\nTT
ATGGGCCTGGATGTGTGGGAACATGCTTACTACCTGAAATTCCAGAACCGCCGTCCGGACTACATTAA\nAGAGTTCTGGAACGTGGTGAACT
GGGACGAAGCAGCGGCACGTTTTGCGGCGAAAAAATAATCATTTGCC\nGCCTGCTGCAATGAGGCGTATAGGCCGCATATCAGCTTAAAAAA
TGAACCATCGCCAACGGCGGTGGTTT\nTTTTGTGATCAATTTCAAAATAAAAACAATGATCCGAATAAAAATAAAACAGCGTTTCAATTGAT
GTGGT\nTTTGACACTTTTATGATTAAATGAATGTCTATCTTCGTTTCCATCAACACTGATGCTCCATTGAGGAATT\nACGCATCAGCCCTT
AAAAATATGCCGACAGGTGATGGAAATGCAGATAAAACGCTCGATTGAGAAAATCC\nCGG\n\n'
```

El esqueleto de esto es algo así:

|Nombre de variable| = |Variable donde estas leyendo| [|carácter donde empieza la secuencia| : |espacio intencionalmente en blanco|]

Nota: Nota (valga la redundancia) que usamos corchetes, y no paréntesis, para especificar un intervalo.

- Quitando el \n: Vamos a decirle a Jupyter “quítale esta cosa y reemplázala con esta otra cosa”. ¿Qué necesitas? Decirle **dónde** va a trabajar, **qué** hacer, **especificar** lo que quieres reemplazar y **con qué** cosa lo va a reemplazar. De nuevo, como estamos

“arreglando” lo que vamos a trabajar, nos hará más fácil asignarle una nueva variable a lo que estamos modificando, ponle el nombre que quieras.

```
In [5]: c = b.replace("\n", "")

In [6]: c

Out[6]: 'TCGGGCATTTTCCTGCAAAACCATACCCTTACGAAAAGTACGGCATTGATAATCATTTTCAATATCATTTAATTAACTATAATGAACCAACT
GCTTACGCGGCATTAACAATCGGCCGCCCGACAATACTGGAGATGAATATGAGCTATACCCTGCCATCCCTGCCGTATGCTTACGATGCCCTG
GAACCGCACTTCGATAAGCAGACCATGGAAATCCACCACACCAAACACCATCAGACCTACGTAAACAACGCCAACGCGGCGCTGGAAAGCCTG
CCAGAATTTGCCAACCTGCCGGTTGAAGAGCTGATTACCAAACTGGACCAGCTGCCAGCAGACAAGAAAACCGTACTGCGCAACAACGCTGGC
GGTCACGCTAACCACAGCCTGTTCTGGAAAGGTCTGAAAAAAGGCACCACCCTGCAGGGTGACCTGAAAGCGGCTATCGAACGTGACTTCGGC
TCCGTTGATAACTTCAAAGCAGAATTTGAAAAAGCGGCAGCTTCCCGCTTTGGTTCCGGCTGGGCATGGCTGGTGCTGAAAGGCGATAAACTG
GCGGTGGTTTCTACTGCTAACCAGGATTCTCCGCTGATGGGTGAAGCTATTTCTGGCGCTTCCGGCTTCCCGATTATGGGCCTGGATGTGTGG
GAACATGCTTACTACCTGAAATTCCAGAACCGCCGTCCGGACTACATTAAAGAGTTCTGGAACGTGGTGAACTGGGACGAAGCAGCGGCACGT
TTTGCGGCGAAAAAATAATCATTTGCCGCCTGCTGCAATGAGGCGTATAGGCCGCATATCAGCTTAAAAAATGAACCATCGCCAACGGCGGTG
GTTTTTTTGTGATCAATTTCAAAATAAAAACAATGATCCGAATAAAAATAAAACAGCGTTTCAATTGATGTGGTTTTGACACTTTTATGATTA
AATGAATGTCTATCTTCGTTTCCATCAACACTGATGCTCCATTGAGGAATTACGCATCAGCCCTTAAAAATATGCCGACAGGTGATGGAAATG
CAGATAAAACGCTCGATTGAGAAAATCCCGG'
```

Y ya que tenemos la secuencia por fin como la quiero, empieza el ejercicio: Buscar un segmento en especial.

III. Encontrando un intervalo específico (secuencia).

El ejercicio que haremos en este primer paso es, dentro de nuestra secuencia problema, buscar la región codificante utilizando Jupyter Notebook. Asumiré que sabes cómo funciona Jupyter e iré directo al grano, el proceso general es el siguiente:

1) En la pagina del NCBI, busca la parte de CDS, que te resalta (dentro de tu secuencia completa) la parte codificante.

```
ORIGIN
        1 tcgggcattt tcctgcaaaa ccataccctt acgaaaagta cggcattgat aatcattttc
       61 aatatcattt aattaactat aatgaaccaa ctgcttacgc ggcattaaca atcggccgcc
      121 cgacaatact ggagatgaat atgagctata ccctgccatc cctgccgtat gcttacgatg
      181 ccctggaacc gcacttcgat aagcagacca tggaaatcca ccacaccaaa caccatcaga
      241 cctacgtaaa caacgccaac gcggcgctgg aaagcctgcc agaatttgcc aacctgccgg
      301 ttgaagagct gattaccaaa ctggaccagc tgccagcaga caagaaaacc gtactgcgca
      361 acaacgctgg cggtcacgct aaccacagcc tgttctggaa aggtctgaaa aaaggcacca
      421 ccctgcaggg tgacctgaaa gcggctatcg aacgtgactt cggctccgtt gataacttca
      481 aagcagaatt tgaaaaagcg gcagcttccc gctttggttc cggctgggca tggctggtgc
      541 tgaaaggcga taaactggcg gtggtttcta ctgctaacca ggattctccg ctgatgggtg
      601 aagctatttc tggcgcttcc ggcttcccga ttatgggcct ggatgtgtgg gaacatgctt
      661 actacctgaa attccagaac cgccgtccgg actacattaa agagttctgg aacgtggtga
      721 actgggacga agcagcggca cgttttgcgg cgaaaaaata atcatttgcc gcctgctgca
      781 atgaggcgta taggccgcat atcagcttaa aaaatgaacc atcgccaacg gcggtggttt
      841 ttttgtgatc aatttcaaaa taaaaacaat gatccgaata aaaaataaaa cagcgtttca
      901 attgatgtgg tttgacactt ttatgattaa atgaatgtct atcttcgttt ccatcaacac
      961 tgatgctcca ttgaggaatt acgcatcagc ccttaaaaat atgccgacag gtgatggaaa
     1021 tgcagataaa acgctcgatt gagaaaatcc cgg
```

2) Copia y añade los 10 primeros, y últimos, caracteres de la secuencia codificante. En este caso:
   Inicio = atgagctata

Fin = cgaaaaaaaata a (aquí son 11, una excepción pero no una especialmente rara)

Notarás que la secuencia está en minúsculas, lo que debe ser corregido a mayúsculas para hacerlo coincidir con nuestra secuencia problema completa. Para ello utilizamos la función “x.upper()”. Asígnale el nombre que quieras a los nucleótidos de inicio y de fin, y luego convierte esta variable a mayúsculas.

```
In [7]: inicio = "atgagctata"
        in_mayus = inicio.upper()

        fin = "cgaaaaaataa"
        fin_mayus = fin.upper()

In [8]: in_mayus
Out[8]: 'ATGAGCTATA'

In [9]: fin_mayus
Out[9]: 'CGAAAAAATAA'
```

3) Dile a Jupyter que te muestre, dentro de tu secuencia completa, el intervalo que inicia con “xxxxxx” y “yyyyyyy”. Entiéndase como los primeros 10 caracteres “x” y los últimos 10 “y”.

```
In [19]: in_index = c.find(in_mayus)
         fin_index = c.find (fin_mayus)

         finalle = fin_index + 11

         print("Esta es la secuencia codificante del gen Mn-SOD en E.coli: \n", c[in_index:finalle])

Esta es la secuencia codificante del gen Mn-SOD en E.coli:
 ATGAGCTATACCCTGCCATCCCTGCCGTATGCTTACGATGCCCTGGAACCGCACTTCGATAAGCAGACCATGGAAATCCACCACACCAAACA
CCATCAGACCTACGTAAACAACGCCAACGCGGCGCTGGAAAGCCTGCCAGAATTTGCCAACCTGCCGGTTGAAGAGCTGATTACCAAACTGGA
CCAGCTGCCAGCAGACAAGAAAACCGTACTGCGCAACAACGCTGGCGGTCACGCTAACCACAGCCTGTTCTGGAAAGGTCTGAAAAAAGGCAC
CACCCTGCAGGGTGACCTGAAAGCGGCTATCGAACGTGACTTCGGCTCCGTTGATAACTTCAAAGCAGAATTTGAAAAAGCGGCAGCTTCCCG
CTTTGGTTCCGGCTGGGCATGGCTGGTGCTGAAAGGCGATAAACTGGCGGTGGTTTCTACTGCTAACCAGGATTCTCCGCTGATGGGTGAAGC
TATTTCTGGCGCTTCCGGCTTCCCGATTATGGGCCTGGATGTGTGGGAACATGCTTACTACCTGAAATTCCAGAACCGCCGTCCGGACTACAT
TAAAGAGTTCTGGAACGTGGTGAACTGGGACGAAGCAGCGGCACGTTTTGCGGCGAAAAAATAA
```

Para que esto funcione debemos darle la orden de que, dentro de la secuencia completa **encuentre** la secuencia “x” y “y”. Para esto usamos el comando “x.find(nombre de la variable)” donde pondremos dentro de los paréntesis la variable donde tienes los 10 caracteres de inicio/fin convertidos a mayúscula. Una vez hecho esto podemos ir y decirle

print (c[in_index:fin_index]

El resultado debe ser el intervalo de los 10 caracteres del inicio a los últimos 11. PERO aquí ocurre un error. Nos da los 10 del inicio, pero se come letras del final, y aún no se bien porqué, pero pasa con todos los ejercicios. Afortunadamente es sencillo corregir esto.

Crea una nueva variable, donde a tu variable que tenga los 10 finales le sumas el numero de caracteres que le faltan. En mi caso le

puse finalle = fin_index + 11. Con eso te muestra toda la secuencia codificante.

IV. Creando un archivo FASTA

Tienes tu secuencia en texto, miles de pares de bases y quieres guardarlo en formato FASTA para la prosperidad ¿cómo le hacemos? Este más sencillo que leer archivos, ya que consta solo de dos pasos:

1. Crear el contenido FASTA

1.1. Crea una variable a la que le asignarás tu secuencia en formato FASTA, cosa que siempre inicia con un “>”, siempre. Seguido de el código de acceso, género, especie y demás información. Luego la secuencia con sus respectivos \n cada 60 nucleótidos. Es importante que toda tu secuencia este en el mismo renglón/línea, de otro modo lo toma como un error de

```
In [5]: # Paso 1 - Crea una variable con el contenido de lo que va a ser tu archivo FASTA
        ADN = ">CódigoEspecífico Género Especie DemásInfo \n TTAAATTGGCAAAGAAATTTGACATCTTCAATGGGGAATGTCCAAATTTTGTATTTCCCTTAAATTCCAT\nA
```

```
In [7]: ADN

Out[7]: '> CódigoEspecífico Género Especie DemásInfo \n TTAAATTGGCAAAGAAATTTGACATCTTCAATGGGGAATGTCCAAATTTTGTATTTCCCTTAAATTCCAT\nAAT
        CAAGACTATTCAACCAAGGGTTGAAAAGAAAAAGCTTGATGGCTTTATGGGTAGAATTCGATCCGTC\nTATCCAGTTGCGTCACTAAATGAATGCAACCAAATGTGCCTTTCAACTCTCATG
        AAGTGTGATCATTGTG\nGTGAAACTTCATGGCAGACGGGCGATTTTGTTAAAGCCACTTGCGAATTTTGTGGCACTGAGAATTTGAC\nTAAAGAAGGTGCCACTACTTGTGGTTACTTACC
        CCAAAATGCTGTTGTTAAAATTTATTGTCCAGCATGT\nCACAATTCAGAAGTAGGACCTGAGCATAGTCTTGCCGAATACCATAATGAATCTGGCTTGAAAACCATTC'
```

Python.

2. Guardar el contenido en un archivo

2.1. Crearemos otra variable a la cual le asignaremos el nombre descriptivo de lo que hará: guardar. En este ejemplo nos dicen:

guardarFASTA = open (r’MyDNA.FASTA’, ‘w+’)
guardarFASTA.write(ADN)
guardarFASTA.close() ← *Nunca olvides cerrar lo que sea que abriste*

```
In [9]: # Paso 2 - Crea otra variable con el nombre de lo que quieres hacer: guardar. "w+" es el argumento que te dice que es con el
        # propósito de escribir. Así no solo estas creando un nuevo archivo, le estas diciendo que lo abra y escriba!

        guardarFASTA = open(r'MiADN.FASTA','w+')

        guardarFASTA.write(ADN)

        guardarFASTA.close()
```

Con esto Jupyter nos crea un archivo de texto en formato FASTA de la secuencia que nosotros escribimos.

V. Operaciones básicas

A. Longitud.

a. Supongamos que quieres saber la longitud de una secuencia de ADN de un archivo FASTA ¿cómo le hacemos? Para esto ocupamos len(), donde lo que va entre los paréntesis es la variable donde está tu secuencia. Solo recuerda que los archivos FASTA traen el espaciado (\n) incluido, por lo que deberás quitarlo antes de pedir la longitud.

```
In [1]: # Longitud
        PapasCalientes = "ncdjfkhasdjkfhsjdncksjdncsjkdncjkasdncaksjdcnajksdcnj\nsdfjsafjlsdfkjaklsdñjfasñldfkjalkjsjdfkal\njklsdlfja
        <                                                                                                    >

In [2]: PapasCalientes.replace("\n","")

Out[2]: 'ncdjfkhasdjkfhsjdncksjdncsjkdncjkasdncaksjdcnajksdcnjsdfjsafjlsdfkjaklsdñjfasñldfkjalkjsjdfkaljklsdlfjaskldfjdsklafjaslkñd
        fjasdlkjfaklasjdfklasdkfjasñldkfjasdñlfkjjklfdsjfkalsdfjdklasñafskdjfk'

In [3]: print(len(PapasCalientes))

        196
```

B. Cambio de mayúsculas a minúsculas, y viceversa.

a. Tienes una secuencia que está en mayúsculas y la necesitas en minúsculas, o al revés. La operación se resume en el uso de *.upper()* o *.lower()*, dependiendo de si necesitas mayúsculas o minúsculas, respectivamente. Crea una nueva variable a la que le asignarás la variable que contiene tu secuencia más la operación que necesites.

```
In [4]: # Cambio de mayúsculas a minísculas, y viceversa.
        PCU = PapasCalientes.upper()
        PCU

Out[4]: 'NCDJFKHASDJKFHSJDNCKSJDNCSJKDNCJKASDNCAKSJDCNAJKSDCNJ\nSDFJSAFJLSDFKJAKLSDÑJFASÑLDFKJALKJSJDFKAL\nJKLSDLFJASKLDFJDSKLAFJAS
        LKÑDFJASDLKJFA\nKLASJDFKLASDKFJASÑLDKFJASDÑLFKJ\nJKLFDSJFKALSDFJDKLASÑAFSKDJFK'

In [5]: PCL = PapasCalientes.lower()
        PCL

Out[5]: 'ncdjfkhasdjkfhsjdncksjdncsjkdncjkasdncaksjdcnajksdcnj\nsdfjsafjlsdfkjaklsdñjfasñldfkjalkjsjdfkal\njklsdlfjaskldfjdsklafjas
        lkñdfjasdlkjfa\nklasjdfklasdkfjasñldkfjasdñlfkj\njklfdsjfkalsdfjdklasñafskdjfk'
```

C. Concatenación

a. Se refiere a unir, o enlazar, otro segmento a tu secuencia original. Recordemos que por lo general un ARNm, por ejemplo, trae un capuchón que no es codificante y actúa únicamente como protección. Si quisiéramos tenerla presente en nuestra secuencia, pero no venía incluída en la secuencia inicial, podemos simplemente agregarla con concotenación. Para esto la operación es + que unirá las dos variables que elijas.

i. En adición a esto, puedes agregar un hilo (o string, texto que pasa literalmente en Python ya que va entre comillas) para hacer más fácil ubicar dónde se hizo la

concatenación. Aquí yo lo hice con lo que se me ocurrió en el momento.

```
In [6]: # Concatenación - Uniendo cosas con otras cosas.
        # [Variable A] + [Identificador Ocpional En String] + [Variable B]

        Ensalada = "vcbnvmxcbvmnxcbvxmcnvbxmncbvxncbvxcbmvbvbcnmxvbxmncvbxmcnvbxnmcv"
        PapasCalientes = PapasCalientes+"_____Aqui esta el acompañamiento_____"+Ensalada
        PapasCalientes

Out[6]: 'ncdjfkhasdjkfhsjdncksjdncsjkdncjkasdncaksjdcnajksdcnj\nsdfjsafjlsdfkjaklsdñjfasñldfkjalkjsjdfkal\njklsdlfjaskldfjdsklafjas
        lkñdfjasdlkjfa\nklasjdfklasdkfjasñldkfjasdñlfkj\njklfdsjfkalsdfjdklasñafskdjfk_____Aqui esta el acompañamiento_____vcbnvmxc
        bvmnxcbvxmcnvbxmncbvxncbvxcbmvbvbcnmxvbxmncvbxmcnvbxnmcv'
```

D. Reemplazar elementos
   a. En ocasiones habrá que reemplazar bases nitrogenadas para, yo que se, algo. Habrá casos donde debas reemplazar un carácter por otro. Para ello empleamos .replace() dando como primer argumento lo que queremos cambiar, seguido a lo que lo queremos cambiar.

```
In [7]: # Reemplazar elementos
        # .replace()
        print(PapasCalientes.replace("j","x"))

        ncdxfkhasdxkfhsxdncksxdncsxkdncxkasdncaksxdcnaxksdcnx
        sdfxsafxlsdfkxaklsdñxfasñldfkxalkxsxdfkal
        xklsdlfxaskldfxdsklafxaslkñdfxasdlkxfa
        klasxdfklasdkfxasñldkfxasdñlfkx
        xklfdsxfkalsdfxdklasñafskdxfk_____Aqui esta el acompañamiento_____vcbnvmxcbvmnxcbvxmcnvbxmncbvxncbvxcbmvbvbcnmxvbxmncvbxmcn
        vbxnmcv
```

```
In [8]: print(Ensalada.replace("m","ñ"))

        vcbnvñxcbvñnxcbvxñcnvbxñncbvxncbvxcbñvbvbcnñxvbxñncvbxñcnvbxnñcv
```

E. Aislando intervalos.
   a. Quizá querremos visualizar una parte en especial de nuestra secuencia para una inspección de cerca, pero es difícil mantenerla a la vista entre tanta letras. Podemos aislar una parte en específico de nuestra secuencia utilizando corchetes y especificando el lugar que queremos ver. El número de la posición donde se encuentra lo que queremos ver más de cerca.

```
In [9]: # Aislando intervalos
        # print(VariableTrabajo[x:y])

        print(PapasCalientes[6:10])

        hasd
```

F. Contando caracteres específicos o “subtrings”
   a. Quizá quieres contar una base o proteína en especial, pero no quieres contar uno por uno en tu secuencia, y len() no ayuda. Aquí usamos la función .count(), poniendo dentro de los paréntesis el carácter (como string) a contar.

```
In [11]: #Contar caracteres especificos

         PapasCalientes.count("l")

Out[11]: 19
```

G. Spliting Pendiente
   a. ||Pendiente||

Comparando secuencias con BLAST – Parte 2

Funciones y Técnicas - Parte 3

Software de Biología Computacional son programas que ayudan a transformar datos crudos a información que podemos usar para hacer descubrimientos y guiar experimentos. Pasamos de grandes cadenas de A,G,T,C a información que podemos interpretar.

Cuando hablamos de datos crudos los pasamos por una “pipeline” que implica el pulir todo este conjunto de datos para obtener respuestas claras de lo que queremos saber.

Si no queremos meternos en lo que Ciencias Computacionales abarca, desde programación e ingenierías, debemos tener en cuenta dos cosas:

- Evaluar los resultados correctamente desde el punto biológico depende mucho de que el interpretador tenga entendido qué es lo que hace el algoritmo con la secuencia que va a trabajar.
- La estadística es importante, ya que todo experimento debe ser reproducible. Por lo que saber las partes básicas de la estadística es imprescindible para generar conclusiones. De hecho, se han dado incidentes que han derivado en problemas legales debido a que las pruebas estadísticas no fueron transparentes y repetibles, derivados del poco conocimiento en el campo estadístico. Peor aún hay pocos expertos que puedan intervenir en estos campos de investigaciones.

How to Share Data With a Statistician

Para ser transparentes hay que tener un “Data Set” en orden

1. Datos en crudo
2. Datos acomodados para uso interactivo fácil y listo para trabajar
3. Un “libro código” que servirá como índice para usarlo a la par de los datos acomodados. Cada valor debe estar descrito aquí
4. Una “receta” explicita y exacta de lo que sea que hiciste para llegar a tu resultado. Desde el paso 1 hasta el 2 y 3. Una bitácora detallada.

En caso de que necesitemos “asistencia” estadística, podemos recurrir a expertos o páginas de preguntas y respuestas como Cross Validated

donde puedes preguntar por qué modelo se ajusta mejor a lo que necesitas, o si el modelo que usaste tiene sentido o no.

Para compartir resultados estadísticos es importante el plotting o “graficado” que sea amigable a la vista, fácil de explicar y que comunique resultados científicos (datos reales).