Guía de actividades práctico Python

Docentes: Msc. Florencia Grattarola y Dr. Andrés Iriarte

"Programando en Python"

- CEAR UN PROGRAMA (SCRIPT)

La idea es primero ver el ejercicio entre todos y después ejecutar las líneas de código sugeridas.

1) Crear un archivo *.py ejecutable:

En primer lugar para que el código que escribamos sea ejecutable tenemos que indicar en la primer línea del archivo dónde se encuentra Python, es decir, dónde se encuentra el intérprete que usará nuestro *script*. Mediante el par de caracteres "#!" (llamados *shebang* en Unix) se indica la ruta completa al intérprete de las órdenes contenidas en el programa ejecutable: "#!/usr/bin/env python". De esta forma, el script llama al intérprete del lenguaje para ejecutar el código dentro del script y el *shebang* es la "guía" para encontrar Python.

En segundo lugar, para volver al *script* ejecutable por terminal debemos cambiar los permisos del archivo mediante el comando "chmod +x" y así poder correrlo directamente en el shell con el promt ./

1. Abrimos un editor de texto. Por ejemplo: kate (Unix) o Notepad++ (Windows)
2. Copiamos el código #!/usr/bin/env python en la primer línea del archivo.
3. Guardamos nuestro archivo como "programa.py" en el directorio "/estudiantes/*nombre_apellido*/PRACTICO9/".
4. Vamos al *shell* (la terminal de línea de comando) y hacemos que el archivo pase a ser ejecutable mediante el comando: chmod +x programa.py
5. Para probar si nuestro programa funciona escribimos en la tercer línea del archivo: print 'Programa para calcular el contenido GC de una secuencia', guardamos el archivo y ejecutamos nuestros programa en el *shell*: ./programa .py

```
#!/usr/bin/env python

print 'Programa para calcular el contenido GC
de una secuencia'
```

2) Recorrer una lista (for/while):

Las estructuras de repetición son útiles para recorrer listas. El ***for*** nos permite recorrer cada uno de los elementos de la lista, mientras ***while*** se suele utilizar recorrer los elementos hasta alcanzar un cierto valor pre-definido, que funciona como tope del recorrido. A modo de ejemplo vamos a recorrer una lista de secuencias de ADN.

1. En nuestro programa previamente creado primero definimos la variable que contiene a la lista de secuencias a ser analizadas. Para eso copiamos la siguiente línea: lista_secuencias = ['ACGATAGCTAGC', 'TAGCTAGCTAGCTAGCTA', 'ATCGATCGATAGC', 'ATCGATCGATCGATG', 'TATCGATCGATTAG']

2. Luego recorremos la lista e imprimimos cada una de las secuencias en la lista: for secuencia in lista_secuencias: print secuencia.

```
#!/usr/bin/env python

lista_secuencias = ['ACGATAGCTAGC', 'TAGCTAGCTAGCTAGCTA',
'ATCGATCGATAGC', 'ATCGATCGATCGATG', 'TATCGATCGATTAG']

for secuencia in lista_secuencias:
  print secuencia
```

3) Uso de condicionales (if/elif/else).

Las sentencias condicionales son sumamente útiles cuando necesitamos decidir si ejecutar un fragmento de código o no, o cuando dadas ciertas condiciones tendríamos varias alternativas de código a ejecutar. A modo de ejemplo, decidiremos entre varias secuencias, si éstas son adecuadas para el diseño de *primers* (cebadores de ADN) o no. Para esto, por ahora tendremos en cuenta que las secuencias tengan más de 15 nucleótidos.

1. Vamos a analizar cada secuencia de la lista de secuencias evaluando si tienen un largo mayor o igual a 15 nucleótidos. Para esto recorremos la lista y evaluamos el largo de las secuencias usando la función ***len*** de Python, que nos devuelve el número de elementos de una lista o *string*. Si la secuencia fuera de largo mayor a 15, entonces la imprimimos en pantalla. Para esto escribiremos las siguientes líneas de código: for secuencia in lista_secuencias: if len(secuencia) >= 15: print secuencia

```
#!/usr/bin/env python

lista_secuencias = ['ACGATAGCTAGC', 'TAGCTAGCTAGCTAGCTA',
'ATCGATCGATAGC', 'ATCGATCGATCGATG', 'TATCGATCGATTAG']

for secuencia in lista_secuencias:
  if len(secuencia) >= 15:
    print secuencia
```

*¿Cómo haría para calcular el largo de una secuencia de ADN sin utilizar la función ***len***?
*Y si las secuencias además de ser mayores a 15pb tienen que ser menores a 40 pb, ¿cómo integro esta condición al *script*?

4) Calcular contenido GC de una secuencia.

Otra característica a tener en cuenta para poder considerar una secuencia como *primer* es que tenga bajo contenido GC (consideraremos menor a 50%). Para calcular el contenido GC de una secuencia, debemos sumar las Cs y Gs (G+C), dividir entre el total de nucleótidos de la secuencia (A+C+G+T) y multiplicar por 100. Haremos uso nuevamente de la función ***len*** y sumaremos a la función ***count***, que cuenta la cantidad de elementos en un *string*. Definiremos primero el largo de cada secuencia como largo = len(secuencia). Luego contaremos la cantidad de Cs cant_C = secuencia.count('C') y de Gs cant_G = secuencia.count('G'). Finalmente, calculamos el porcentaje GC porcentaje_GC = (float(cant_C + cant_G) / largo) * 100. El tipo de dato *float* es necesario para considerar valores decimales. Imprimiremos únicamente aquellas secuencias

que cumplan con la característica de ser de largo mayor a 15 nucleótidos y tener un contenido GC menor a 50% if porcentaje_GC < 50: print secuencia, porcentaje_GC, '%'

```
#!/usr/bin/env python

lista_secuencias = ['ACGATAGCTAGC', 'TAGCTAGCTAGCTAGCTA',
'ATCGATCGATAGC', 'ATCGATCGATCGATG', 'TATCGATCGATTAG']

for secuencia in lista_secuencias:
  if len(secuencia) >= 15:
    largo = len(secuencia)
    cant_C = secuencia.count('C')
    cant_G = secuencia.count('G')
    porcentaje_GC = (float(cant_C + cant_G) / largo) * 100
    if porcentaje_GC < 50:
      print secuencia, porcentaje_GC, '%'
```

*¿Cómo haría para calcular la cantidad de Cs en una secuencia de ADN sin utilizar la función ***count***?

*¿Y para calcular la Tm de la secuencia de nucleótidos?

$$T_m = 64 + 41 \cdot \frac{G + C - 16.4}{A + G + C + T}$$

- PROGRAMA PARA LEER Y ESCRIBIR ARCHIVOS

Crearemos un programa que nos permita extraer datos de secuencias de un archivo en formato FASTA para copiarlos en un nuevo archivo de texto. A partir de una base de datos de secuencias d-loop de carnívoros obtendremos un nuevo archivo listando el ID de cada secuencia y a qué especie pertenece.

1) Abrir un archivo "file.fasta" y leerlo:

Para trabajar con archivos existen varias opciones de lectura. En todos los casos se debe abrir el archivo para poder operar sobre las líneas.

1. Abrir un archivo de texto nuevo y copiar en la primer línea el código que indica la ruta hacia el intérprete de Python. Guardar el archivo como extraer_datos.py (recordar luego hacer ejecutable el archivo por línea de comando).
2. Para que nuestro programa sea independiente del archivo fasta que queramos leer, indicaremos en el programa que el archivo será pasado como argumento en la línea de comando, incluyendo la siguiente línea de código: file = sys.argv[1] (para poder usar este código debo importar la librería ***sys***, import sys). Este código nos permitirá luego correr nuestro programa del siguiente modo: ./extraer_datos.py file.fasta siendo *file*, cualquier archivo tipo fasta que quiera analizar.
3. Para poder trabajar con el archivo, primero tenemos que abrirlo en modo lectura y recorrer línea por línea: for line in open(archivo, 'r'). A modo de ejemplo usaremos print linea para visualizar el contenido.

```
#!/usr/bin/env python

import sys

archivo = sys.argv[1]

for linea in open(archivo, 'r'):
  print linea
```

2) Recorrer cada secuencia fasta y extraer datos:

Una secuencia en formato FASTA consiste en una primer línea de descripción (nombre de secuencia), seguida de líneas de datos de secuencia. El primer carácter de la línea de descripción es un símbolo mayor que ('>'). Utilizaremos este signo para distinguir una línea de nombre de una de secuencia.

Leeremos las secuencias de nucleótidos que se encuentran dentro del archivo FASTA carnivores.fasta para extraer de cada una de ellas el ID y la especie. Este archivo contiene 11281 secuencias de D-loop de ADN mitocondrial.

Secuencia ejemplo:

```
>gi|2995798|gb|AF053055.1|AF053055 Panthera tigris altaica isolate S12 mitochondrial D-loop
CCAATACCAAAAAACAACCCCATGACTTTCATAATTCATATATTGCATATACCCGTACTGTGCTTGCCCA
```

1. Para poder extraer el ID y la especie, primero debemos quedarnos con las líneas que hacen referencia a los nombres, que son las que empiezan con el carácter '>'. Para filtrarlas haremos uso de la función 'startswith' de Python: if linea.startswith('>')
2. Una vez que tenemos las líneas que nos interesan, debemos sub-dividirlas para obtener los datos de la descripción que nos interesan. En este caso vemos que los separadores son barras verticales ('|') y por tanto nos serán útiles para partir la línea. Separamos la línea: linea.split('|')
3. El ID de la secuencia es el tercer dato de la línea y el nombre de la especie está contenido en el cuarto dato. Para obtener específicamente el valor que queremos debemos dividir la línea: linea.split('|')[0], donde 0 representa el índice de la lista al sub-dividir la línea.
4. Definiremos primero la variable ID, ID = linea.split('|')[3]. En el caso del ejemplo, ID sería AF053055.1
5. Para extraer de la última porción de información únicamente la especie, debemos hacer un nuevo split, ésta vez sub-dividiendo por espacios y no por '|'. En el caso del ejemplo la especie es Panthera tigris. Para quedarnos sólo con esta información definiremos genero = linea.split('|')[4].split(' ')[1] y especie = linea.split('|')[4].split(' ')[2].
6. Finalmente línea a línea se irá imprimiendo la información de interés: print ID, genero, especie.

```
#!/usr/bin/env python

import sys

archivo = sys.argv[1]

for linea in open(archivo, 'r'):
  if linea.startswith('>'):
```

```
    ID = linea.split('|')[3]
    genero = linea.split('|')[4].split(' ')[1]
    especie = linea.split('|')[4].split(' ')[2]
    print ID, genero, especie
```

3) Guardar los datos extraídos en un archivo nuevo:

Para guardar los datos de interés en un nuevo archivo debemos primero crear el archivo de texto, abrirlo luego en modo escritura, escribir la información y finalmente cerrarlo.

1. Para crear el archivo basta con abrirlo creando una nueva variable: nuevo_archivo = open('lista_datos.txt','w') ('w' indica el modo de escritura).
2. Luego cambiamos nuestro print por un código de escritura: nuevo_archivo.write(ID + '\t' + genero + '\t' + especie + '\n'). Utilizamos '\t' para separar dentro de la misma línea de escritura cada variable, y '\n' para que cada vez que termina una línea pase a la siguiente para seguir escribiendo. Notar que para separar los datos que escribo utilizo '+'.
3. Y finalmente cerramos el archivo: nuevo_archivo.close()

```
#!/usr/bin/env python

import sys

archivo = sys.argv[1]

nuevo_archivo = open('lista_datos.txt','w')

for linea in open(archivo, 'r'):
  if linea.startswith('>'):
    ID = linea.split('|')[3]
    genero = linea.split('|')[4].split(' ')[1]
    especie = linea.split('|')[4].split(' ')[2]
    nuevo_archivo.write(ID + '\t' + genero + '\t' +
especie + '\n')

nuevo_archivo.close()
```