# **TP 7**

**Alumnos: Cuoco Carlos, Markon Mariano, Verdecanna Mariano**

### 

### **RETO TI: ¿Podrías buscar un ejemplo de macromoléculas que almacenen**

### **información sobre la ‘identidad’ de una célula?**

El ejemplo más común es el ADN (ácido desoxirribonucleico). ADN es el nombre químico de la molécula que contiene la información genética en todos los seres vivos. La molécula de ADN consiste en dos cadenas que se enrollan entre ellas para formar una estructura de doble hélice. Cada cadena tiene una parte central formada por azúcares (desoxirribosa) y grupos fosfato. Enganchado a cada azúcar hay una de de las siguientes 4 bases: adenina (A), citosina (C), guanina (G), y timina (T). Las dos cadenas se mantienen unidas por enlaces entre las bases; la adenina se enlaza con la timina, y la citosina con la guanina. La secuencia de estas bases a lo largo de la cadena es lo que codifica las instrucciones para formar proteínas y moléculas de ARN.

### **RETO TI:**

##### ***Ahora bien, es importante conocer nuestra herramienta de trabajo, en este caso nuestra PC o teléfono inteligente.***

##### ***¿Podrías identificar qué parte de la computadora estaremos trabajando?***

Software

### **RETO I: ¿Podés descubrir y anotar el orden en que se ha ejecutado cada operación?**

#### **((4+5)\*2)/5**

In [32]:

**def** f1():

print('4+5')

**return** 4**+**5

​

**def** f2(suma):

print('{}\*2'.format(str(suma)))

**return** suma**\***2

​

**def** f3(multiplicacion):

print('{}/5'.format(str(multiplicacion)))

**return** multiplicacion**/**5

​

resultado **=** f3(f2(f1()))

print(resultado)

print(str(((4**+**5)**\***2)**/**5))

4+5

9\*2

18/5

3.6

3.6

Primero se ejecuta la suma que se encuentra entre paréntesis, luego la multiplicación y por último la división

### **RETO II: Crea una variable llamada ​ doble, que sea el doble de la suma entre a y b.**

In [33]:

a **=** 5

b **=** 500

doble **=** 2**\***(a**+**b)

print(doble)

1010

### **RETO III:**

Digamos que el ADN no es más que un mensaje en clave, que debe ser descifrado o interpretado para la síntesis de proteínas. El mensaje está escrito por una secuencia determinada de 4 nucleótidos distintos representados por las letras A, T, G y C. Dentro de la célula, el mensaje es transportado por otra molécula, el ARN, muy similar al ADN pero con U en vez de T. En este mensaje, cada triplete o grupo de tres letras del ARN se denomina codón, y cada aminoácido de las proteínas está codificado por uno o varios codones. Así por ejemplo el codón ‘AUG’ codifica para el aminoácido Metionina , el codón ‘AAA’ para Lisina, el codón ‘CUA’ para Leucina, etc. ¿Podrías escribir una cadena de ARN que codifica para el péptido (una cadena corta de aminoácidos) ‘Met-Lis-Lis-Lis-Leu-Leu-Met’ combinando las variables met = ‘AUG’ , lis = ‘AAA’ y leu = ‘CUA’ utilizando operadores matemáticos?

In [38]:

met **=** 'AUG'

lis **=** 'AAA'

leu **=** 'CUA'

cadena **=** 'Met-Lis-Lis-Lis-Leu-Leu-Met'

​

**def** codificar\_cadena(peptido):

cadenaresultado**=**''

lista **=** cadena.split('-')

**for** aa **in** lista:

**if** aa **==** 'Met':

cadenaresultado **+=** met

**elif** aa **==** 'Lis':

cadenaresultado **+=** lis

**elif** aa **==** 'Leu':

cadenaresultado **+=** leu

**return** cadenaresultado

​

print(codificar\_cadena(cadena))

​

AUGAAAAAAAAACUACUAAUG

**### RETO IV:**

¿Cadenas?¿letras? Si hablamos de cadenas y letras en Biología, lo

primero que se nos viene a la cabeza son las macromoléculas. Como bien

sabemos, el ADN es un mensaje en clave que guía la síntesis de proteínas. Este

mensaje está escrito por una secuencia determinada de 4 nucleótidos distintos

representados por las letras A, T, G y C. El contenido de C y G (es decir el

porcentaje de CG) presente en el ADN de un organismo es una característica

distintiva: por ejemplo las • Actinobacterias • tienen un contenido

característicamente más alto de CG que otros organismos. Ahora, contar la

cantidad de C y G en una cadena de ADN larguísima a mano puede ser un

verdadero tedio ¿Podrías crear un programa que calcule el porcentaje de C y G

de una cadena dada de ADN?

In [35]:

*#Algunas secuencias paraa pruebas*

*#s1 = 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACC'*

*#s2 = 'AAAAAAAAAG'*

*#s3 = 'AAAAAAAACG'*

*#s4 = 'CG'*

secuencia\_ejemplo **=** 'TGATAAGAGTACCCAGAATAAAATGAATAACTTTTTAAAGACAAAATCCTCTGTTATAATATTGCTAAAATTATTCAGAGTAATATTGTGGATTAAAGCCACAATAAGATTTATAATCTTAAATGATGGGACTACCATCCTTACTCTCTCCATTTCAAGGCTGACGATAAGGAGACCTGCTTTGCCGAGGAGGTACTACAGTTCTCTTCACAAACAATTGTCTTACAAAATGAATAAAACAGCACTTTGTTTTTATCTCCTGCTTTTAATATGTCCAGTATTCATTTTTGCATGTTTGGTTAGGCTAGGGCTTAGGGATTTATATATCAAAGGAGGCTTTGTACATGTGGGACAGGGATCTTATTTTAGATTTATATATCAAAGGAGGCTTTGTACATGTGGGACAGGGATCTTATTTTACAAACAATTGTCTTACAAAATGAATAAAACAGCACTTTGTTTTTATCTCCTGCTCTATTGTGCCATACTGTTGAATGTTTATAATGCATGTTCTGTTTCCAAATTTCATGAAATCAAAACATTAATTTATTTAAACATTTACTTGAAATGTTCACAAACAATTGTCTTACAAAATGAATAAAACAGCACTTTGTTTTTATCTCCTGCTTTTAATATGTCCAGTATTCATTTTTGCATGTTTGGTTAGGCTAGGGCTTAGGGATTTATATATCAAAGGAGGCTTTGTACATGTGGGACAGGGATCTTATTTTAGATTTATATATCAAAGGAGGCTTTGTACATGTGGGACAGGGATCTTATTTTACAAACAATTGTCTTACAAAATGAATAAAACAGCACTTTGTTTTTATCTCCTGCTCTATTGTGCCATACTGTTGAATGTTTATAATGCATGTTCTGTTTCCAAATTTCATGAAATCAAAACATTAATTTATTTAAACATTTACTTGAAATGTGGTGGTTTGTGATTTAGTTGATTTTATAGGCTAGTGGGAGAATTTACATTCAAATGTCTAAATCACTTAAAATTTCCCTTTATGGCCTGACAGTAACTTTTTTTTATTCATTTGGGGACAACTATGTCCGTGAGCTTCCATCCAGAGATTATAGTAGTAAATTGTAATTAAAGGATATGATGCACGTGAAATCACTTTGCAATCAT'

​

**def** porcentaje\_nucleotido(cadena, nucleotido):

ocurrencias **=** cadena.count(nucleotido)

porcentaje **=** ocurrencias **\*** 100 **/** len(cadena)

print('En la secuencia hay {} ocurrencias y representan un {} %'.format(str(ocurrencias), porcentaje))

**return** porcentaje

​

​

**def** porcentaje\_cg(cadena):

pc **=** porcentaje\_nucleotido(cadena, 'C')

pg **=** porcentaje\_nucleotido(cadena, 'G')

print('El porcentaje total de CG es {} %'.format(pc**+**pg))

**return** pc **+** pg

​

print(porcentaje\_cg(secuencia\_ejemplo))

En la secuencia hay 169 ocurrencias y representan un 14.811568799298861 %

En la secuencia hay 194 ocurrencias y representan un 17.002629272567923 %

El porcentaje total de CG es 31.814198071866784 %

31.814198071866784

### **RETO V:**

La Asombrosa Maravillosa es nuestra valiente superheroína. Sus poderes son producto de mutaciones en un gen muy común, cuya secuencia en la mayoría de las personas es 'ATGGAACTTGCAATCGAAGTTGGC'. A diferencia de nosotros, el gen mutado de la Asombrosa Maravillosa incluye la secuencia 'GTTTGTGGTTG' en su interior. La Asombrosa Maravillosa adquirió sus poderes al beber Jugo Vencido. El primer sorbo de esta poción prohibida causa el cambio de todas las citosinas (C) por timinas (T). El siguiente sorbo cambia todas las adeninas (A) por guaninas (G). El tercer sorbo cambia las citosinas (C) por adeninas (A). El cuarto sorbo... puede ser mortal. ¿Podés escribir un programa que nos diga cuántos sorbos de Jugo Vencido debe beber un portador del gen normal, para ganar los poderes de la Asombrosa Maravillosa?

<div><br class="Apple-interchange-newline">comun = 'ATGGAACTTGCAATCGAAGTTGGC' maravilla = 'GTTTGTGGTTG' def ya\_es\_maravilla(secuencia): print(secuencia) return maravilla in secuencia def conteo\_super(): primer\_sorbo = comun.replace('C','T') if not ya\_es\_maravilla(primer\_sorbo): segundo\_sorbo = primer\_sorbo.replace('A', 'G') if not ya\_es\_maravilla(segundo\_sorbo): tercer\_sorbo = segundo\_sorbo.replace('C', 'A') if not ya\_es\_maravilla(tercer\_sorbo): print('CHEPACHO') else: print('Es super al tercer sorbo') else: print('Es super al segundo sorbo') else: print('Es super al primer sorbo') conteo\_super()</div>

comun **=** 'ATGGAACTTGCAATCGAAGTTGGC'

maravilla **=** 'GTTTGTGGTTG'

​

**def** ya\_es\_maravilla(secuencia):

print(secuencia)

**return** maravilla **in** secuencia

​

**def** conteo\_super():

primer\_sorbo **=** comun.replace('C','T')

**if** **not** ya\_es\_maravilla(primer\_sorbo):

segundo\_sorbo **=** primer\_sorbo.replace('A', 'G')

**if** **not** ya\_es\_maravilla(segundo\_sorbo):

tercer\_sorbo **=** segundo\_sorbo.replace('C', 'A')

**if** **not** ya\_es\_maravilla(tercer\_sorbo):

print('CHEPACHO')

**else**:

print('Es super al tercer sorbo')

**else**:

print('Es super al segundo sorbo')

**else**:

print('Es super al primer sorbo')

​

conteo\_super()

ATGGAACTTGCAATCGAAGTTGGC

ATGGAATTTGTAATTGAAGTTGGT

GTGGGGTTTGTGGTTGGGGTTGGT

Es super al segundo sorbo

### **RETO VI:**

#### **¿Se te ocurre qué operadores podrías usar para las listas?**

In [66]:

dinos **=** ['Patagotitan', 'Titanosaurus', 'Argentinosaurus']

print(dinos)

dinos.append('Puertasaurus')

dinos.remove('Titanosaurus')

print(dinos)

​

unir\_listas **=** dinos **+** ['Agregasaurus']

print('#### UNIMOS DOS LISTAS ####')

print(unir\_listas)

​

print('#### MULTIPLICAMOS UNA LISTA PO UN NUMERO Y DEVUELVE UNA LISTA QUE CONTIENE 2 VECES LA LISTA ORIGINAL ####')

mul\_listas **=** unir\_listas **\*** 2

print(mul\_listas)

​

print('#### RECORREMOS LOS ELEMENTOS DE UNA LISTA IMPRIMIENDO Y CONTANDO A MEDIDA QUE LA RECORREMOS ####')

contador **=** 1

**for** dino **in** dinos:

print(str(contador) **+** dino)

contador **+=** 1

​

print('#### VEMOS LOS METODOS DISPONIBLES DE UNA CLASE. EN ESTE CASO DE UNA LISTA ####')

print(dir(dinos))

['Patagotitan', 'Titanosaurus', 'Argentinosaurus']

['Patagotitan', 'Argentinosaurus', 'Puertasaurus']

#### UNIMOS DOS LISTAS ####

['Patagotitan', 'Argentinosaurus', 'Puertasaurus', 'Agregasaurus']

#### MULTIPLICAMOS UNA LISTA PO UN NUMERO Y DEVUELVE UNA LISTA QUE CONTIENE 2 VECES LA LISTA ORIGINAL ####

['Patagotitan', 'Argentinosaurus', 'Puertasaurus', 'Agregasaurus', 'Patagotitan', 'Argentinosaurus', 'Puertasaurus', 'Agregasaurus']

#### RECORREMOS LOS ELEMENTOS DE UNA LISTA IMPRIMIENDO Y CONTANDO A MEDIDA QUE LA RECORREMOS ####

1Patagotitan

2Argentinosaurus

3Puertasaurus

#### VEMOS LOS METODOS DISPONIBLES DE UNA CLASE. EN ESTE CASO DE UNA LISTA ####

['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_delitem\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_iadd\_\_', '\_\_imul\_\_', '\_\_init\_\_', '\_\_init\_subclass\_\_', '\_\_iter\_\_', '\_\_le\_\_', '\_\_len\_\_', '\_\_lt\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_reversed\_\_', '\_\_rmul\_\_', '\_\_setattr\_\_', '\_\_setitem\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']

### **RETO VII:**

Ya que encontramos el espécimen de rana con pelo en marte, nos gustaría contrastar sus características con las ranas terrestres. Sabiendo que el gen de la proteína diminuta es ‘ATGGAAGTTGGAATCCAAGTTGGA’ y el gen de una proteína similar de rana terrestre es ‘ATGGAAGTTAATGGAAGTTGGAGGAGA’ ¿podés crear un programa que compare la longitud de ambos genes y según cuál sea más grande nos imprima un mensaje informándonos el resultado?

diminuta **=** 'ATGGAAGTTGGAATCCAAGTTGGA'

terrestre **=** 'ATGGAAGTTAATGGAAGTTGGAGGAGA'

​

**def** proteina\_mas\_grande():

print('La mas grande es la terrestre') **if** len(diminuta) **<** len(terrestre) **else** print('La mas grande es la marciana')

​

proteina\_mas\_grande()

La mas grande es la terrestre

### **RETO VIII:**

Si nos ponemos un poco más estrictos, y siguiendo con el tema de los clones de bacterias, el programa que creamos antes tiene algunas fallas ‘numéricas’: en cada vuelta de división celular binaria se generarán dos clones, no uno. ¿Podrías escribir un programa que imprima ‘¡Somos 2 clones nuevos!’ en cada una de 20 vueltas?

In [71]:

**for** i **in** range(0,20):

print('Somos 2 clones nuevos')

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

Somos 2 clones nuevos

### **RETO IX:**

Si ahora queremos hacer nuestro programa un poco más estricto, por cada vuelta deberíamos sumar el total de células que tenemos e imprimir ese número en el mensaje. Entonces, por ejemplo, como en la primer vuelta tenemos dos células, imprimiríamos como mensaje ‘¡Somos 2 clones!’ , pero en la segunda vuelta serán en total 4 células y el mensaje a imprimir debería ser ‘¡Somos 4 clones!’. ¿Podrías escribir esta modificación del programa?

In [75]:

**for** i **in** range(0,20):

print('Somos {} clones'.format(str((i**+**1)**\***2)))

Somos 2 clones

Somos 4 clones

Somos 6 clones

Somos 8 clones

Somos 10 clones

Somos 12 clones

Somos 14 clones

Somos 16 clones

Somos 18 clones

Somos 20 clones

Somos 22 clones

Somos 24 clones

Somos 26 clones

Somos 28 clones

Somos 30 clones

Somos 32 clones

Somos 34 clones

Somos 36 clones

Somos 38 clones

Somos 40 clones