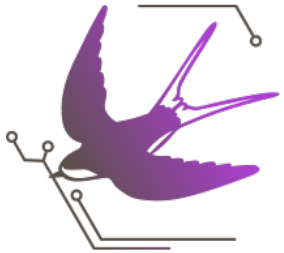
Introducción: vectores

Curso 0



DIPLOMA EXPERTO

Bioinformática
y Genómica Computacional

Rodrigo Santamaría
2020

Definiciones

Muchas cajas

- Decíamos que podíamos guardar cualquier cosa en una caja, incluidas palabras:

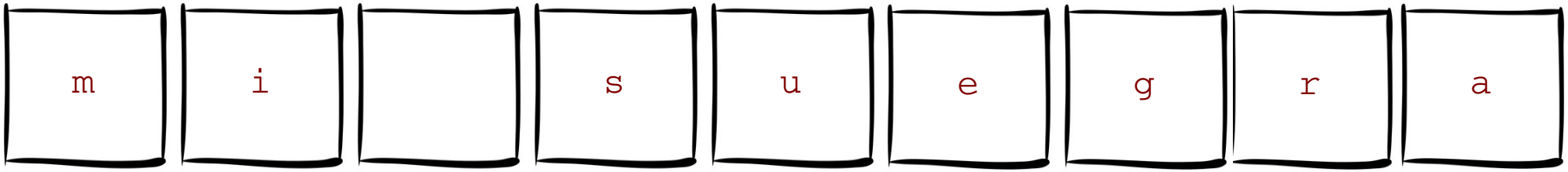
¡Maldita sea, al final me despisté y he metido a mi suegra en la caja!

Ninguna suegra ha resultado herida en la confección de estas transparencias.
Mi suegra es maravillosa y todo esto es sólo una excusa para el chascarrillo fácil

Muchas cajas

- Una palabra en una caja puede verse también como varias letras en varias cajas:

`miCaja`



Nunca intentéis trocear a vuestra suegra en cajas

Muchas cajas: definiciones

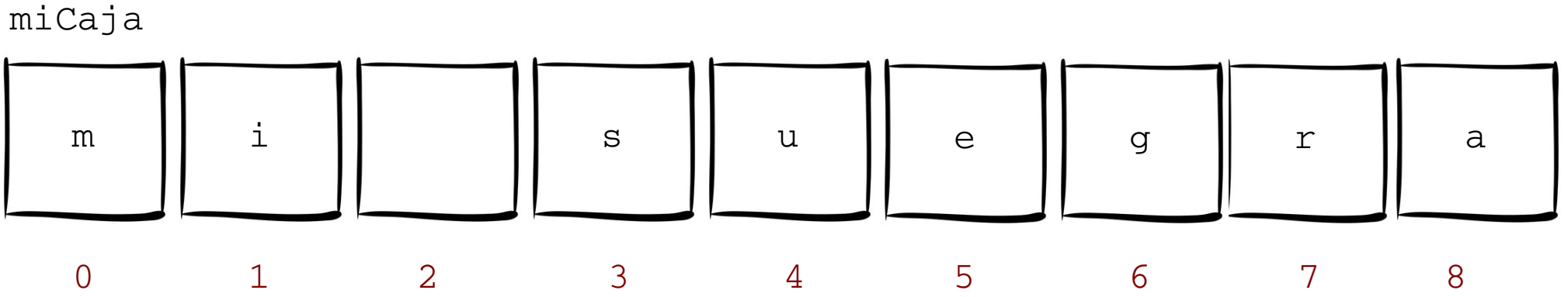
- Esta forma de ver una variable, como una sucesión ordenada de elementos, recibe varios nombres:
  - **Cadena** (o string): cuando cada elemento es un carácter
  - **Vector**: cuando cada elemento es un número
  - **Lista** (o list): cuando cada elemento es cualquier otra cosa
  - **Array***: intercambiable por cualquiera de las anteriores

*Array a veces se traduce literalmente como ‘arreglo’ pero sería más correcto hablar de surtido, formación o matriz. Por ejemplo, los pasillos de un supermercado muestran un ‘array’ de productos, dispuestos en orden en estanterías

Posiciones

Muchas cajas: posiciones

- Cada elemento de un array tiene su posición única:

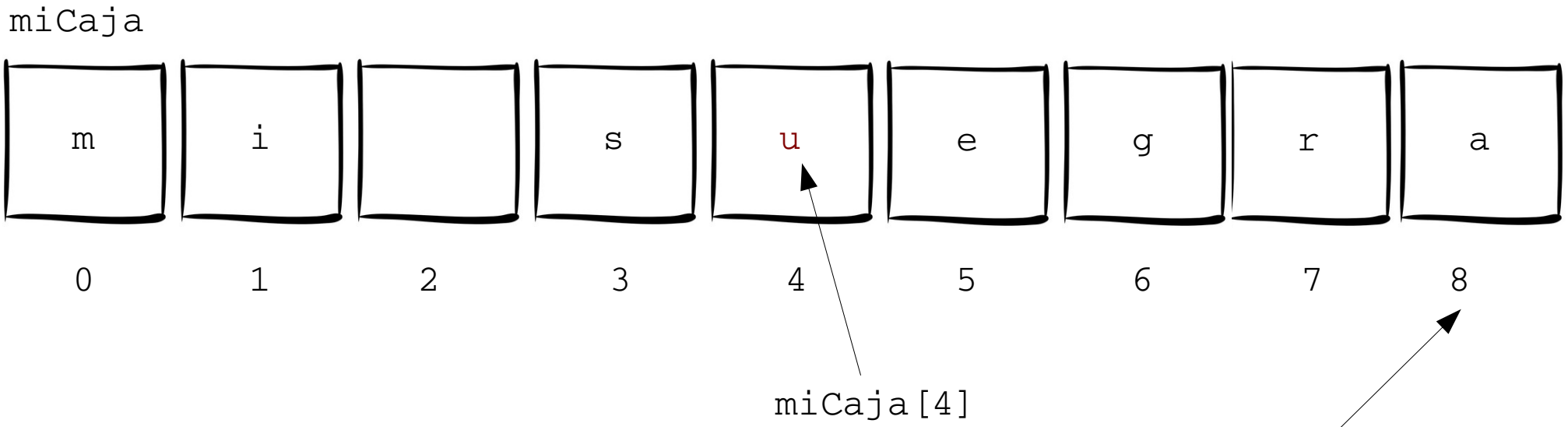


- Mucho cuidado, empezamos a contar desde la **posición 0***

*¿Por qué desde la posición 0 y no la 1? ¡Mira que sois raros!
Originalmente, parece que ahorraba algo de tiempo en los lenguajes de programación primigenios.
Luego muchos lenguajes lo adoptaron de manera un tanto acrítica.
Guido Van Rossum, creador de Python, sostiene que es una *solución más elegante* para el indizado.
Si os interesa saber más sobre el tema: http://exple.tive.org/blarg/2013/10/22/citation-needed/
También os digo, si queréis conceptos contraintuivos, mirad los nombres que se ponen a los genes! :)

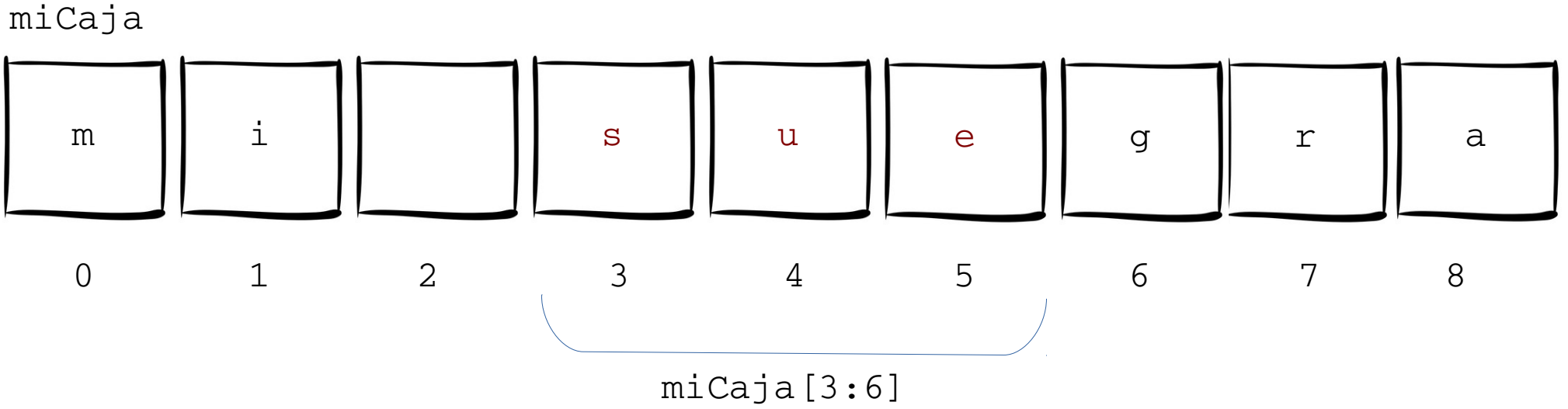
Muchas cajas: posiciones

- Para indicar una letra en concreto, usamos los corchetes `[]`



Muchas cajas: posiciones

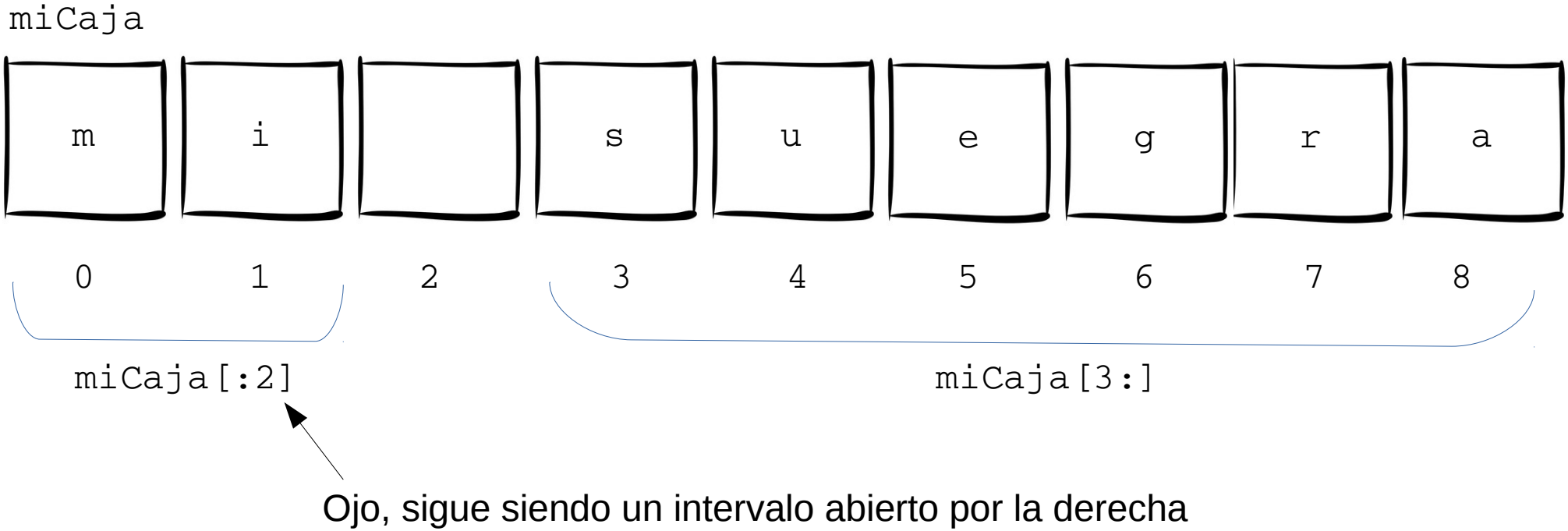
- Para indicar un intervalo, usamos los dos puntos `[:]`
  - Se trata de un intervalo que incluye al índice de la izquierda, pero no al de la derecha*



*En matemáticas, se definiría como un intervalo cerrado por la izquierda y abierto por la derecha

Muchas cajas: posiciones

- En los intervalos, podemos dejar sin indicar un valor
  - Si interpreta como ‘hasta el final’ o ‘hasta el comienzo’



Ejercicio 1

- En el *este enlace** está la secuencia de aminoácidos de la cadena alfa de EnaC**
- Almacénala en una variable que se llame `enac`

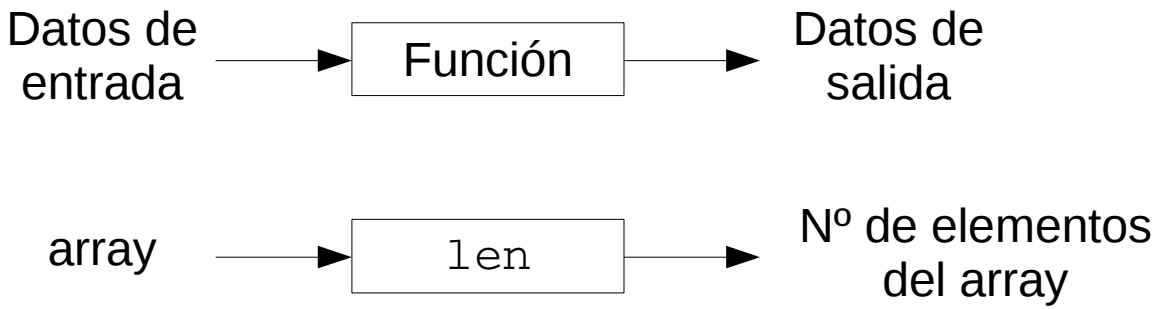
- Consulta en el ejercicio 2 de la sesión 1 del curso 0 la posición de inicio y fin de la secuencia `RRARSVAS`
  - Toma ese intervalo de la variable `enac`.
  - ¿Es la secuencia esperada?*

* Recuerda que las posiciones comienzan en Python por el cero.

Longitud

Muchas cajas: longitud

- Para saber la longitud de un array, usamos la función `len`*
- ¿Qué es una función?
  - Es un concepto un poco más avanzado de programación que no veremos en el curso 0
  - Básicamente, es un programa que a partir de unos datos retorna otros:



* `len` es una abreviatura de length (longitud en inglés). Muchas funciones, tanto en programación como en el terminal, tendrán nombres intuitivos que si sabemos un poco de inglés podremos recordar fácilmente

Muchas cajas: longitud

- La sintaxis de una función es la siguiente:
  - `nombreFuncion(datosDeEntrada)`
  - Por ejemplo
    - `len(array)`
- ¿Y los datos de salida?
  - Se retornan como valores, que se mostrarán por pantalla, o bien podemos almacenar en una variable

Muchas cajas: longitud

```
In [6]: miCaja="mi suegra"

In [7]: len(miCaja)
Out[7]: 9

In [8]: longitud=len(miCaja)

In [9]: longitud
Out[9]: 9
```

Atrapados en el bucle!

Recordad que no podemos poner espacios ni signos raros como nombres de variable
El espacio en este caso hace que Python interprete dos variables, `mi` y `caja`, separadas

```
In [10]: mi caja="mi suegra"
```

```
  File "<ipython-input-10-da08215b5e59>", line 1
    mi caja="mi suegra"
          ^
SyntaxError: invalid syntax
```

Si no acertamos con el nombre de la función, nos lo indicará claramente (`not defined`)

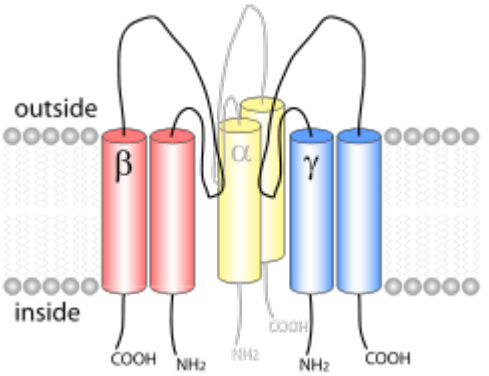
```
In [11]: length(miCaja)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-fba373c8ebb8> in <module>
----> 1 length(miCaja)

NameError: name 'length' is not defined
```

Ejercicio 2

- ¿Qué longitud tiene la proteína ENaC-α?
- ENaC-α es una de las tres proteínas del complejo ENaC que forma el complejo que regula la homeostasis.
- *SCNN1A*, *SCNN1G* y *SCNN1D* son los tres genes que codifican ENaC-α, ENaC-β y ENaC-γ
- Copia las tres secuencias en variables*
- ¿En cuanto al número total de nucleótidos del complejo, qué proporción es ENaC-α?



Fuente: Wikipedia

* Las secuencias se encuentran en los enlaces subrayados con los nombres de los tres genes.
¿De dónde salen estas secuencias? De GenBank, por ejemplo, esta es la página de *SCNN1D* En la asignatura de Bases de Datos veremos cómo buscar información y descargar secuencias en GenBank y otros repositorios.

La información de secuencias suele estar en formato FASTA. He tenido que hacer algunas operaciones para que lo podáis ver como una sola línea de texto. Esto se puede hacer en Python o, por ejemplo, yo esta vez lo he hecho con Linux, con la siguiente línea:

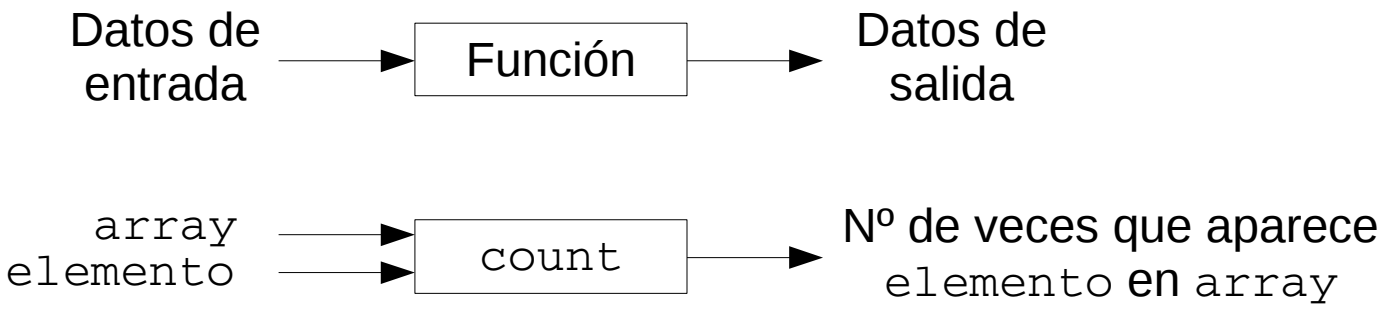
```
tail -n+2 SCNN1G.fasta | paste -d'X' -s | sed 's/X//g' > SCNN1G.txt
```

Aprenderemos también esta escuela de magia en la asignatura de Sistemas UNIX.

Cuentas

Muchas cajas: cuentas

- Otra función muy útil es `count*`
- Recordad que una función, a partir de unos datos de entrada, nos retorna un resultado, en este caso:



* contar, en inglés.

Muchas cajas: cuentas

- En el caso de `len` aplicado a un vector `v`, se escribía así:
  - `len(v)`
- Sin embargo, en el caso de `count` para contar las veces que aparece el elemento `e` en el vector `v` lo haremos así:
  - `v.count(e)*`

* ¿Por qué nos lías, Rodrigo? ¿No tenía esto siempre una sintaxis muy estricta y blablabla?
Tenéis parte de razón, pero a veces las cosas no son tan sencillas :S
- ¿Es porque hay dos argumentos de entrada (`v` y `e`)?
No, generalmente, si hay múltiples argumentos se introducen separados por comas, p. ej:
`miFuncion(argumento1, argumento2, argumento3)`
- ¿Entonces?
En este caso se trata de una función asociada a un tipo de variable (vectores), que se ha construido en base a un tipo de programación que se denomina *orientada a objetos*. No quiero liaros con eso, pero digamos que en estos casos las funciones 'pertenecen' a la variable y se acceden así: `nombreVariable.nombreFuncion(argumentos)`

Muchas cajas: cuentas

```
In [15]: peptido="RRARSVAS"

In [16]: peptido.count("R")
Out[16]: 3
```

```
In [17]: enac="MEGNKLEEQDSSPP
```

```
In [20]: peptido count("R")
  File "<ipython-input-20-e8bde55ea769>", line 1
    peptido count("R")
                ^
SyntaxError: invalid syntax
```

```
In [13]: len(enac)
Out[13]: 669
```

```
In [18]: enac.count("S")
Out[18]: 68
```

```
In [21]: p.count("R")
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-21-8f3996ccf1dd> in <module>
----> 1 p.count("R")

NameError: name 'p' is not defined
```

```
In [19]: enac.count("SS")
Out[19]: 10
```

```
In [22]: peptido.count(R)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-22-939b206924ab> in <module>
----> 1 peptido.count(R)

NameError: name 'R' is not defined
```

Ejercicio 3

- El contenido en GC de algunas bacterias es muy alto. Sin embargo, el parásito de la malaria (*Plasmodium falciparum*) tiene un contenido en GC muy bajo:
  - https://en.wikipedia.org/wiki/GC-content#Among-genome_variation
- *Aquí* tenéis las 600.000 primeras bases del P falciparum
  - Comprobad si efectivamente su contenido en GC es el que dice Wikipedia

Cadenas

Cadenas

- El tratamiento de cadenas es muy sencillo en Python
- A nivel general, podemos acceder a las posiciones y la longitud como en cualquier otro vector o lista.
- Pero además, existen otras operaciones propias de cadenas muy útiles.
- Lo primero es saber si una determinada variable es una cadena (`srt`), para eso usamos la función `type`*
  - Entrada → variable `v`
  - Salida → tipo de `v`

*Python tiene un tipado muy flexible, lo cual ayuda a aprenderlo. No obstante, a menudo es importante conocer el tipo actual de nuestros datos

Tipos

```
In [27]: a=4
         type(a)

Out[27]: int

In [28]: a=3.5
         type(a)

Out[28]: float

In [29]: a="casa"
         type(a)

Out[29]: str
```

Estos son los tipos más comunes a los que nos vamos a enfrentar, al menos de momento*:

- `int` – número entero
- `float` – número real
- `str` – cadena

Una variable es de uno u otro tipo dependiendo del valor que contenga, y puede cambiar de tipo según sus asignaciones

*`int` viene de integer (entero en inglés), `str` de string (cadena) y `float` de floating, flotante en inglés. Este último es el más raro, ya que hace referencia a la aritmética de representación computacional de los números reales, llamada de punto flotante o floating point.
Es algo muy técnico y poco interesante tal vez para vosotros, pero si alguien está interesado, os dejo el *enlace* de Wikipedia.

Operaciones con cadenas

- Una vez confirmamos que estamos ante una cadena, hay muchas operaciones predefinidas que nos van a hacer la vida mucho más fácil:

Función	Entrada	Salida	Descripción
`count`	texto	entero	Cuenta el número de veces que aparece el texto en la cadena
`find`	texto	entero	Retorna la primera la posición de la cadena en la que comience el texto
`replace`	texto1, texto2	cadena	Reemplaza cada ocurrencia de texto1 con texto2
`split`	delimitador	vector	Divide la cadena en subcadenas tomando como separación el delimitador

*Python tiene un tipado muy flexible, lo cual ayuda a aprenderlo. No obstante, a menudo es importante conocer el tipo actual de nuestros datos

Ejercicio 4

- ¿En qué posición comienza la traducción del gen *SCNN1A*?*
- Muestra la sección de la secuencia del gen desde el inicio del gen hasta el codón de inicio (inclusive)

*Recuerda que la traducción arranca en el codón de inicio, ATG
Recuerda también que en el ejercicio 2 hay un enlace para descargar la secuencia de SCNN1A
Saber en qué punto termina, o traducir los codones a aminoácidos son tareas un poco más complejas que dejaremos para la asignatura de Progamación en Python

Ejercicio 5

- Edita la cadena ENaC-alfa para que el coronavirus no pueda aprovecharse del péptido de 'corte'
  - Para ello, p. ej. reemplaza `RRARSVAS` por `- - - - - - - -`

- En ocasiones, las secuencias que descargamos de repositorios tienen símbolos indeseables.
  - Por ejemplo, se suele 'trocear' la secuencia en distintas líneas*
  - Esto suele dejar 'rastro' en forma de caracteres especiales que reflejan los cambios de línea, cuando utilizamos un editor de texto 'plano'**
  - ¿Cómo eliminarías los saltos de línea (`\n`) de esta secuencia?

`"ACG\nCCC\nGGG\TTT"`

*Esto se hace para que los editores de texto no se 'cuelguen' por leer líneas muy largas, ya que están optimizados para documentos con muchas líneas, pero no para documentos con una línea muy larga.
** Un editor de texto plano, muestra todos los caracteres. Word (editor no plano) cambia los caracteres especiales (p.ej. tabuladores o cambios de línea) por su efecto sobre el texto.

Ejercicio 6

- A menudo, los datos que leemos se encuentran en archivos por columnas, donde las columnas están separadas por comas o por tabuladores*

- Mediante Python, separa las siguientes líneas de texto en sus distintos campos:

```
"Nombre,Edad,Sexo,ExpresionGen1,ExpresionGen2"
"Ramón\t44\tVarón\t33.2\t0.8"
```

- Intenta acceder a algunos de ellos mediante `[]`

*En informática, el tabulador tiene el símbolo \t, donde \ es la barra invertida o backslash que encontraras arriba a la izquierda en el teclado, y que se usa para introducir muchos caracteres especiales.