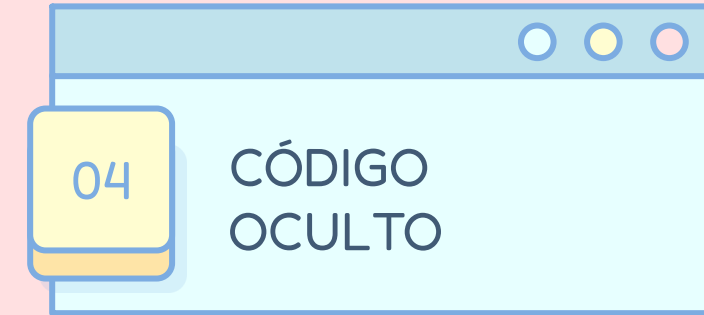
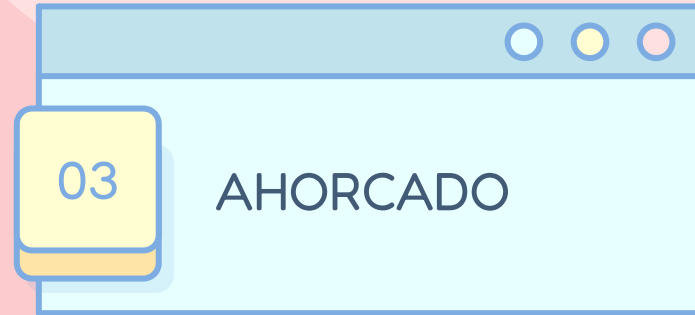
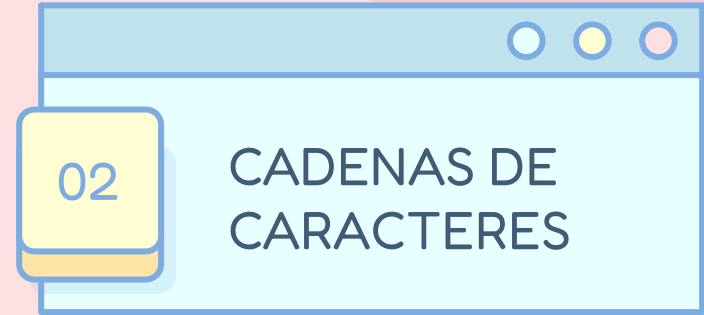
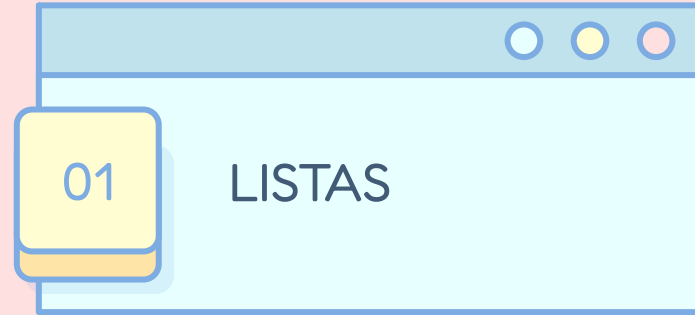




# TALLER DE PROGRAMACIÓN II

CLASE 2

# Clase de hoy



01

# LISTAS



# ¿PARA QUÉ SIRVEN LAS LISTAS?

Las variables hasta ahora solamente permiten guardar una variable.

Esto significa que estamos limitados a la hora de estructurar los datos.

Necesitamos una variable que nos deje guardar más información en una variable.



# ¿QUÉ HAY DENTRO DE LAS LISTAS?

- Se puede acceder a cualquier elemento de una lista utilizando un índice especificado entre corchetes

```
colores=['rojo', 'amarillo', 'azul']  
print(colores[1])  
amarillo  
print(colores[-1])  
azul
```

rojo	amarillo	azul
0	1	2

# ¿CUÁNTO MIDE UNA LISTA?

- La función **len()** toma una lista como parámetro y devuelve el número de elementos de la lista

```
colores=['rojo', 'amarillo', 'azul']  
print(len(colores))  
3
```

```
colores=['rojo', 'amarillo', 'azul','azul']  
print(len(colores))  
4
```

# EXPLORANDO EL MÁS ALLÁ

- Python arroja un error si se intenta indexar más allá del final de una cadena

```
colores=['rojo', 'amarillo', 'azul']  
print(len(colores))  
3  
print(colores[6])
```

- ¡Cuidado cuando se trabaja con los índices!

# EXPLORANDO EL MÁS ALLÁ

- Tenemos dos operadores que permiten verificar si un elemento está en una lista. Estos son operadores lógicos que devuelven Verdadero o Falso.



```
nums = [1, 9, 21, 10, 16]
```

```
9 in nums
```

```
True
```

```
15 in nums
```

```
False
```

```
20 not in nums
```

```
True
```



# RECORRIENDO LISTAS

- Se puede recorrer por índice

```
colores=['rojo', 'amarillo', 'azul']  
for i in range(len(colores)) :  
    print(colores[i])
```

- Se puede recorrer por elemento

```
for i in ['rojo', 'amarillo', 'azul'] :  
    print(i)
```

# USANDO RANGE

- La función `range` devuelve una lista de números que van desde cero hasta uno menos que el parámetro dado.
- Se utiliza frecuentemente para construir un bucle de índice utilizando `for` y un iterador entero.

```
print(range(4))  
[0, 1, 2, 3]  
nombres = ['Juan', 'Luis', 'Ana']  
print(len(nombres))  
3  
print(list(range(len(nombres))))  
[0, 1, 2]
```

# MANIPULANDO LISTAS

- Se puede crear una nueva lista sumando dos listas existentes

```
a = [1, 2, 3]
b = [4, 5, 6]
c = a + b
print(c)
[1, 2, 3, 4, 5, 6]
print(a)
[1, 2, 3]
```

# MANIPULANDO LISTAS

- Se puede cortar una lista utilizando ":"

```
t = [9, 41, 12, 3, 74, 15]
print(t[1:3])
[41, 12]
print(t[:4])
[9, 41, 12, 3]
print(t[3:])
[3, 74, 15]
print(t[:])
[9, 41, 12, 3, 74, 15]
```

# HACER UNA LISTA DESDE CERO

- Podemos crear una lista vacía y luego añadir elementos utilizando el método **append()**
- La lista se mantiene en orden y los nuevos elementos se añaden al final de la lista

```
cosas = list()
cosas.append('libro')
cosas.append(99)
print(cosas)
['libro', 99]
```


- Un método es una función de un tipo de dato



# MÁS MÉTODOS



Métodos	Significado
<code>&lt;list&gt;.append(x)</code>	Añade el elemento x al final de la lista.
<code>&lt;list&gt;.sort()</code>	Ordena la lista.
<code>&lt;list&gt;.reverse()</code>	Invierte la lista.
<code>&lt;list&gt;.index(x)</code>	Devuelve el índice de la primera aparición de x.





# MÁS MÉTODOS

Métodos	Significado
<code>&lt;list&gt;.insert(i, x)</code>	Insertar x en la lista en el índice i.
<code>&lt;list&gt;.count(x)</code>	Devuelve el número de ocurrencias de x en la lista.
<code>&lt;list&gt;.remove(x)</code>	Elimina la primera que aparece de x en la lista.
<code>&lt;list&gt;.pop(i)</code>	Borra el iésimo elemento de la lista y devuelve su valor.

# FUNCIONES BUILT-IN Y LISTAS

- Python tiene varias funciones Built-in que permite facilitar ciertas operaciones de rutina.

<https://docs.python.org/es/3/library/functions.html>

```
nums = [3, 41, 12, 9, 74, 15]
```

```
print(len(nums))
```

**6**

```
print(max(nums))
```

**74**

```
print(min(nums))
```

**3**

```
print(sum(nums))
```

**154**

```
print(sum(nums)/len(nums))
```

**25.6**





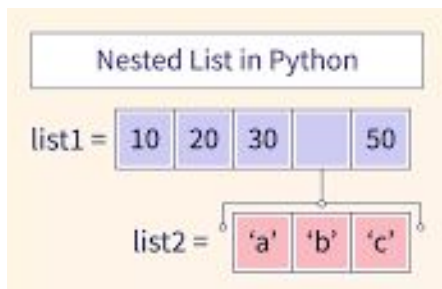
# COPIAR LISTAS

Hay tres maneras para hacer una copia de una lista:

- Crear una nueva lista vacía y usar un bucle **for** para agregar una copia de cada elemento de la lista original a la nueva lista.
- Crear una nueva lista vacía y concatenar la lista antigua a la nueva lista vacía.
- Usar el método **.copy()**.

# LISTAS ANIDADAS

- Las listas anidadas son listas que contiene otras listas como elementos
- Es común pensarlas como tablas, es decir como si tuvieran filas y columnas.
- Son útiles para trabajar con múltiples conjuntos de datos.
- Para procesar datos en una lista bidimensional se necesitan usar dos índices.

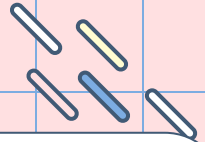




02

# CADENAS DE CARACTERES

# CADENA DE CARACTERES



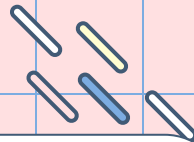
- Una cadena es una secuencia de caracteres
- Una cadena utiliza comillas simples o dobles 'Hola' o "Hola"
- Cuando una cadena contiene números, sigue siendo una cadena

```
str1 = "Hola"
```

```
str2 = 'Hola'
```

```
str3 = '123'
```

# ¿QUÉ HAY DENTRO DE LAS CADENAS?



- Podemos llegar a cualquier carácter de una cadena utilizando un índice especificado entre corchetes

```
str='ajo'  
print(str[1])  
j  
print(str[-2])  
j
```

a	j	o
0	1	2

# ¿ES UNA CADENA UNA LISTA?

- **No**, pero muchas de las funciones se aplican a ambas.
- La función **len()** toma una cadena de caracteres como parámetro y devuelve el número de elementos de la cadena

```
colores='amarillo'
print(len(colores))
8
print(colores[7])
o
```

- **¡Cuidado cuando se trabaja con los índices!**

# SELECCIONADO SUBCADENAS

- Se puede seleccionar partes de una cadena de caracteres para hacer una subcadena.
- `cadena[inicio:final]`
- El inicio y el final deben ser ambos int (números enteros)
- La subcadena comienza en la posición inicio y llega hasta la posición final, pero **no la incluye!**

```
colores='amarillo'  
print(colores[2:5])  
ari
```

# OPERANDO CON CADENAS

- La concatenación junta dos cadenas (+)

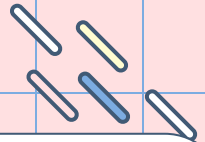
```
print("spam" + "eggs")  
spameggs
```

- La repetición construye una cadena mediante múltiples concatenaciones de una cadena consigo misma (\*)

```
print("spam" * 3)  
spamspamspam
```



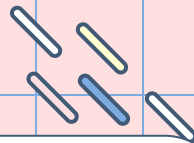
# ¿ESTÁ DENTRO DE LA CADENA?



- Python tiene un operador que permiten comprobar si un elemento está en una cadena
- Es un operador lógicos que devuelven True o False

```
colores = 'amarillo'
print('ama' in colores)
True
print('Rojo' in colores)
False
```

# CARACTERES ESPECIALES



- La barra invertida se utiliza para introducir un carácter especial.

	Significado
\\	Barra invertida
\'	Comilla simple
\"	Comilla doble
\n	Línea nueva
\t	Tabulación

# CADENAS LARGAS

- Las comillas triples permiten que las cadenas abarquen varias líneas, incluyendo literalmente NEWLINEs, TABs, y cualquier otro carácter especial.

```
"""Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!"""
```

# ITERANDO CON CADENAS

```
fruta = 'banana'
for letra in fruta:
    print(letra)
```

```
index = 0
while index < len(fruta):
    letra = fruta[index]
    print(letra)
    index = index + 1
```



# CADENAS vs LISTAS

- Las cadenas son "**inmutables**" - no se puede cambiar el contenido de una cadena
- Las listas son "**mutables**": se puede cambiar un elemento de una lista mediante el operador de índice

```
fruta = 'Banana'  
fruta[0] = 'b'
```

**Traceback**

**TypeError: 'str' object does not  
support item assignment**

```
num = [2, 14, 26]  
num[2] = 28  
print(num)  
[2, 14, 28]
```



# MÉTODOS DE CADENAS

- Un método es una función que está definida para un tipo de dato concreto.
- ¡Las cadenas son **inmutables**! No pueden ser modificadas y en su lugar se crea una nueva.

La sintaxis es `<tipo de dato>.<método>(<parámetro>)`

En este [link](#) hay una lista completa de métodos.



# MÉTODOS

Métodos	Significado
<code>&lt;str&gt;.capitalize()</code>	Copia de str con sólo el primer carácter en mayúscula
<code>&lt;str&gt;.title()</code>	Copia de str. El primer carácter de cada palabra en mayúscula
<code>&lt;str&gt;.center(ancho)</code>	Centrar str en un campo de anchura determinada
<code>&lt;str&gt;.count(sub)</code>	Cuenta el número de ocurrencias de sub en str



# MÁS MÉTODOS

Métodos	Significado
<code>&lt;str&gt;.find(sub)</code>	Encuentra la primera posición en la que aparece sub en str. Si el sub no es encontrado, devuelve -1
<code>&lt;str&gt;.join(lista)</code>	Concatenar una lista de cadenas en una cadena grande usando str como separador.
<code>&lt;str&gt;.lower()</code>	Copia de str en todas las letras minúsculas
<code>&lt;str&gt;.upper()</code>	Copia de str. Todos los caracteres convertidos a mayúsculas





# MÁS MÉTODOS

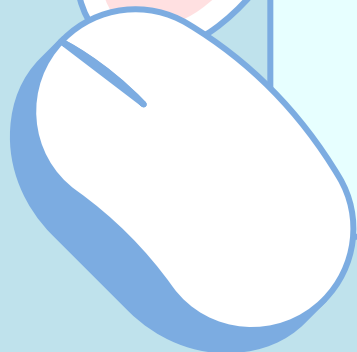
Métodos	Significado
<code>&lt;str&gt;.replace(oldsub, newsub)</code>	Sustituir las apariciones de oldsub en str por newsub
<code>&lt;str&gt;.strip()</code>	Copia de str sin espacios en blanco
<code>&lt;str&gt;.split()</code>	Dividir str en una lista de subcadenas

# MÁS MÉTODOS

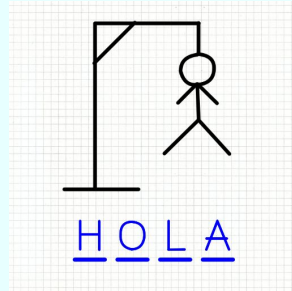
Métodos	Significado
<code>&lt;str&gt;.startswith(sub)</code>	Muestra los str que comienzan con sub
<code>&lt;str&gt;.endswith(sub)</code>	Muestra los str que terminan con sub

03

AHORCADO




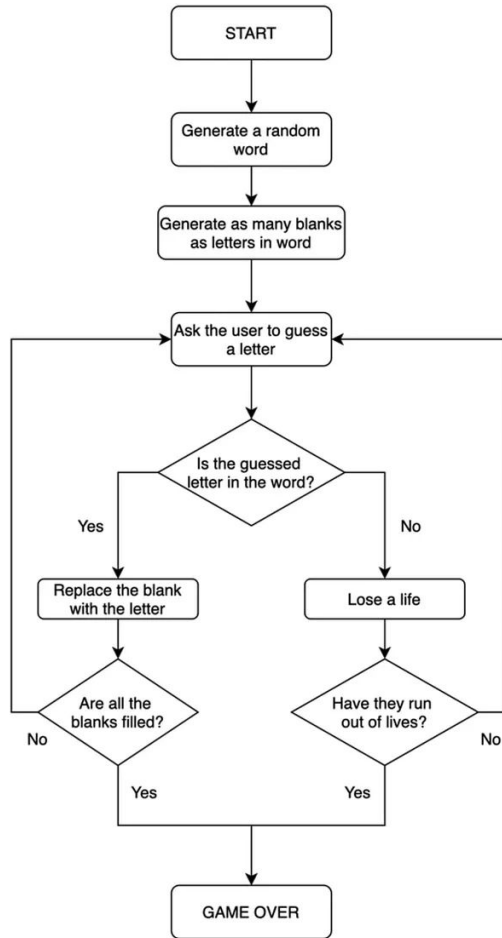
*Crear una implementación del juego el ahorcado que sea lo más completo posible.*





### *Reglas del ahorcado:*

- *Definir el número de intentos. Cada vez que el jugador adivina una letra incorrecta, pierdo un intento.*
  - *Se intenta adivinar una letra. Si la letra es correcta, se indica. Si la letra es incorrecta, se resta un intento.*
  - *Se gana si se adivinan todas las letras. Se pierde si se acaban los intentos.*
- 



Cmd

+

A

# Empecemos a ejercitar!

Ctrl

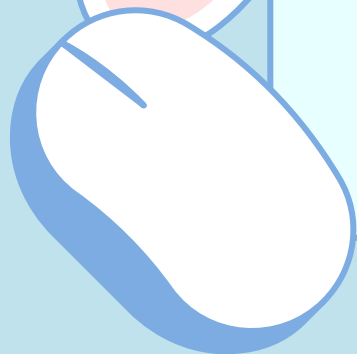
Z



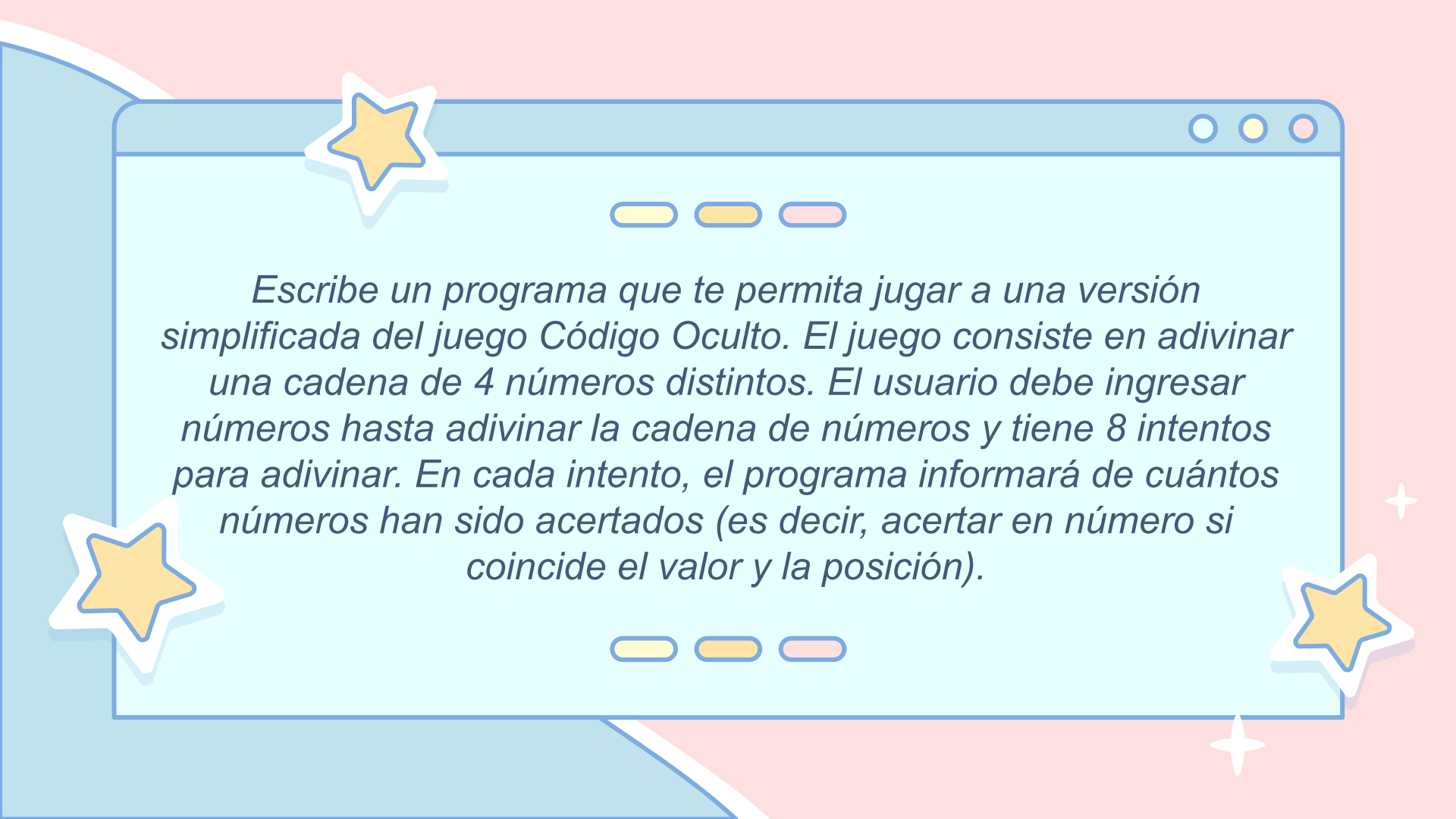
C

04

# CÓDIGO OCULTO







*Escribe un programa que te permita jugar a una versión simplificada del juego Código Oculto. El juego consiste en adivinar una cadena de 4 números distintos. El usuario debe ingresar números hasta adivinar la cadena de números y tiene 8 intentos para adivinar. En cada intento, el programa informará de cuántos números han sido acertados (es decir, acertar en número si coincide el valor y la posición).*





Cmd

+

A

# Vamos a resolver!

Ctrl

Z

C

