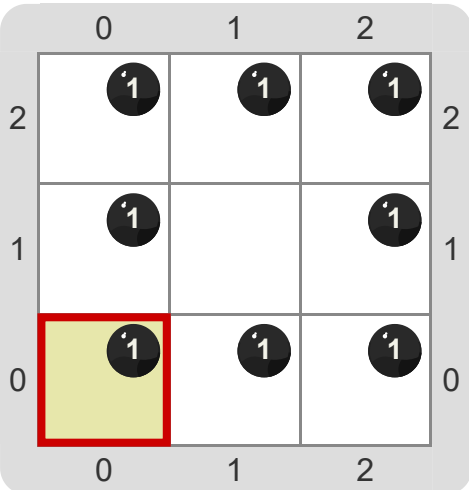


🕒 Te quedan 01:25:35

# Ejercicio 1: Ejercicio 1



Ale nos regaló una hermosa pintura para que colguemos en nuestra pared. Para eso vamos a tener que ponerle un marco el cual pintaremos de Negro:



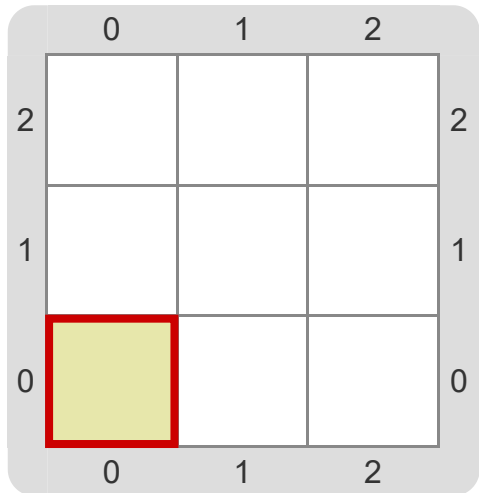
Creá un programa que pinte de Negro los bordes del marco. El cabezal empieza en el origen (o sea, en el borde Sur-Oeste) pero no te preocupes por dónde finaliza.

```
1 program {
2   Poner(Negro)
3   Mover(Este)
4   Poner(Negro)
5   Mover(Este)
6   Poner(Negro)
7   Mover(Norte)
8   Poner(Negro)
9   Mover(Norte)
10  Poner(Negro)
11  Mover(Oeste)
12  Poner(Negro)
13  Mover(Oeste)
14  Poner(Negro)
15  Mover(Sur)
16  Poner(Negro)
17
18 }
```

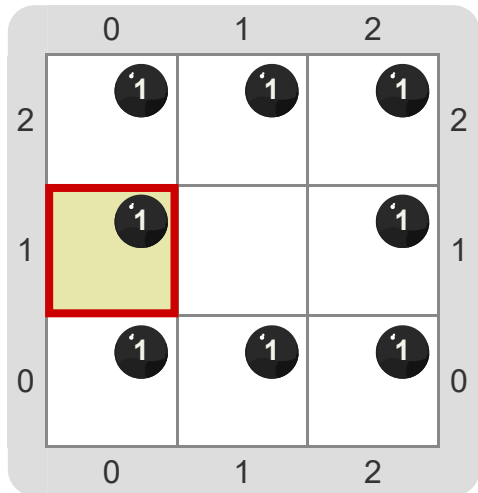
▶ Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

Tablero inicial



Tablero final



🕒 Te quedan 01:25:29

## Ejercicio 2: Ejercicio 2



El color negro del marco del ejercicio anterior no combinó mucho con la pintura .  
¡Hagamos algo para poder probar como quedaría con cualquier color!

Definé el procedimiento `PintarMarco` para que pinte el marco con el `color` que tome por parámetro. No te preocupes por donde termina el cabezal.

```
1 procedure PintarMarco(color){
2   Poner(color)
3   Mover(Este)
4   Poner(color)
5   Mover(Este)
6   Poner(color)
7   Mover(Norte)
8   Poner(color)
9   Mover(Norte)
10  Poner(color)
11  Mover(Oeste)
12  Poner(color)
13  Mover(Oeste)
14  Poner(color)
15  Mover(Sur)
16  Poner(color)
17 }
```

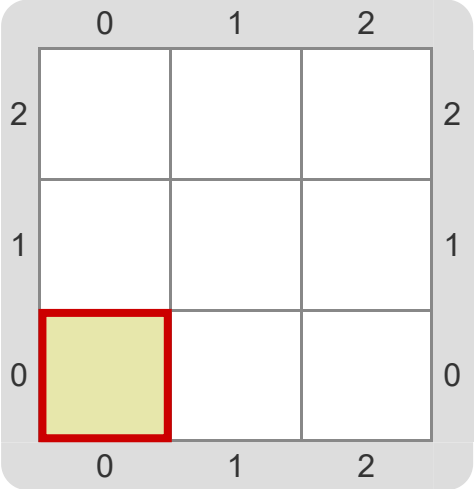
▶ Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

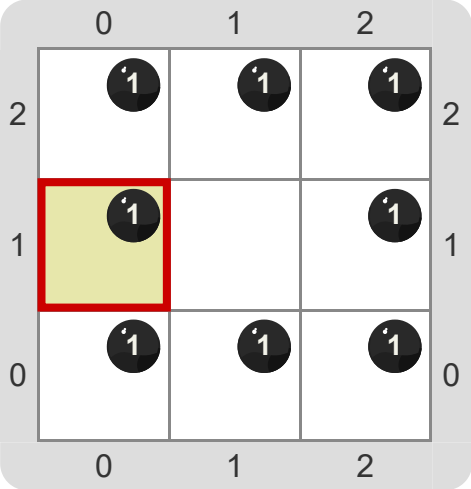
Resultados de las pruebas:



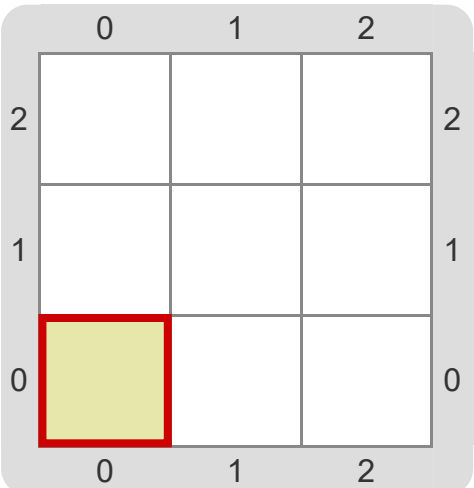
Tablero inicial



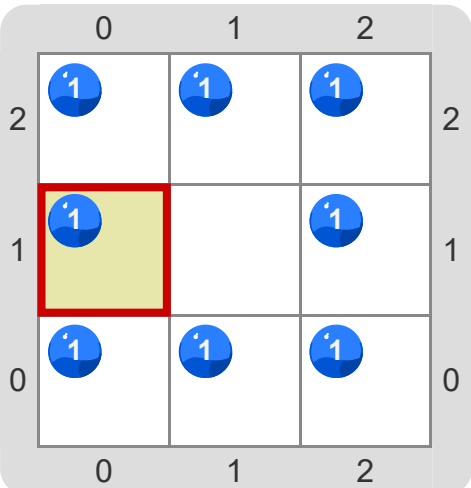
Tablero final



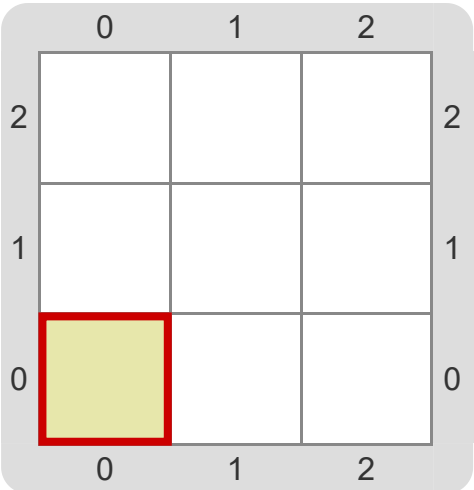
Tablero inicial



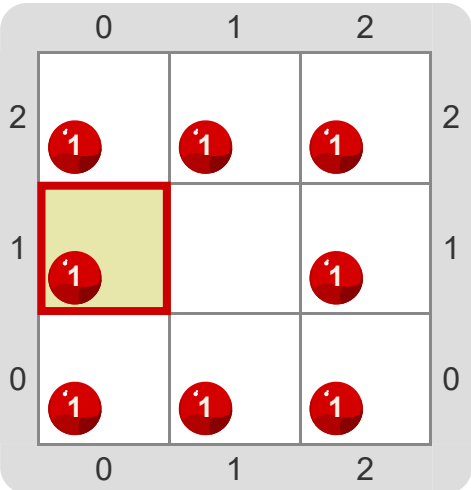
Tablero final



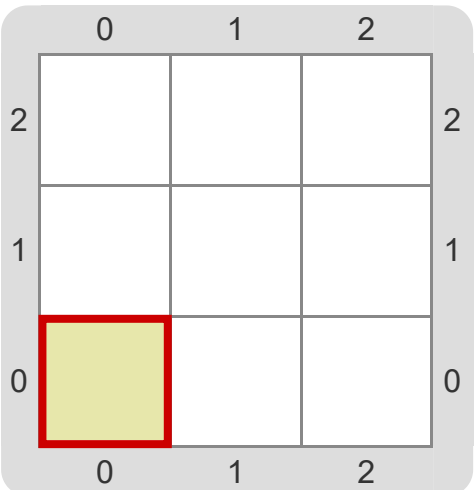
Tablero inicial



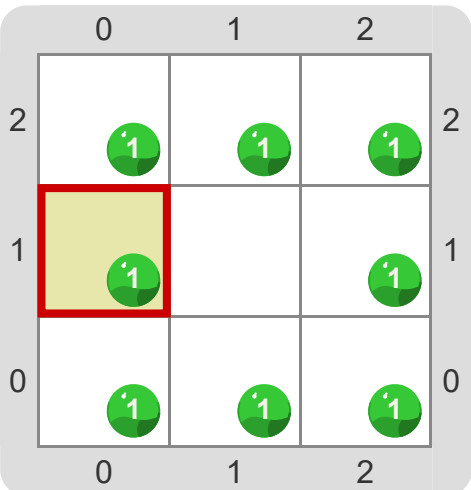
Tablero final



Tablero inicial



Tablero final



 Te quedan 01:25:24

# Ejercicio 3: Ejercicio 3



Vamos a un maravilloso mundo... ¡el de la matemática!

Los números se pueden operar y comparar. Nada nos impide hacer un poco de ambas al mismo tiempo.

Para eso vamos a crear una función que reciba 3 números y nos diga si la suma de los 2 primeros es menor al tercero. Por ejemplo:

```
> laSumaEsMenor(2, 4, 8)
true //Porque 6 es menor que 8

> laSumaEsMenor(3, 5, 7)
false //Porque 8 es mayor a 7
```


Definí la función `laSumaEsMenor`.

 Solución

 Consola

```
1 function laSumaEsMenor(numero1,numero2,numero3){
2
3   return (numero1 + numero2) < numero3;
4
5 }
```

 Enviar

 ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Franco Bulgarelli, Rocío Gonzalez bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).



🕒 Te quedan 01:25:19

# Ejercicio 4: Ejercicio 4



Si hay algo que a Ale le molesta es o pasar frío o abrigarse de más . Pero lo que sí sabe, más allá de la temperatura, es de qué color vestirse ese día. Para eso, pensó en una función que recibe una temperatura y un color y responde qué ropa de ese color ponerse. Si la temperatura es 22 grados o más, se pone una remerita de ese color. Si no, se pone una campera de ese color:

```
> vestirseSegun(22, "negra")
"Remera negra"

> vestirseSegun(21, "verde")
"Campera verde"

> vestirseSegun(23, "violeta")
"Remera violeta"
```

Definí la función `vestirseSegun` .

 Solución  Consola

```
1 function vestirseSegun(temperatura,color){
2   if (temperatura >= 22){
3     return "Remera " + color;
4   } else {
5     return "Campera " + color;
6   }
7 }
8 }
```

 Enviar

✔ ¡Muy bien! Tu solución pasó todas las pruebas



 Te quedan 00:49:30

# Ejercicio 5: Ejercicio 5



Para quienes no suelen leer, la concentración puede variar cuando aparecen palabras largas . Para filtrarlas vamos a crear una función que dada una lista de palabras nos devuelva una lista nueva con las que tengan más de 6 caracteres.

```
> obtenerPalabrasMayores(["jarra", "polilla", "caracol", "gato", "provincia"])
["polilla", "caracol", "provincia"]
```


Definí la función `obtenerPalabrasMayores` .

 Solución

 Consola

```
1 function obtenerPalabrasMayores(listapalabras){
2   let palabramayores = []
3   for(let palabra of listapalabras){
4     if (longitud(palabra) > 6){
5       agregar(palabramayores, palabra)
6     }
7   }
8   return palabramayores;
9 }
```

 Enviar

 ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Franco Bulgarelli, Rocío Gonzalez bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).



🕒 Te quedan 01:25:03

# Ejercicio 6: Ejercicio 6

JS

En una biblioteca guardan registro de todos los libros leídos por las personas que la concurren. Estos registros tienen la siguiente forma:

```
let juan = {
  nombre: "Juan Arrever",
  librosLeidos: ["El conde de Montecristo", "La palabra", "Mi planta de naranj
a lima"],
  anioSuscripcion: 1992
};

let elena = {
  nombre: "Elena Chalver",
  librosLeidos: ["Rabia", "Vida de Bob Marley"],
  anioSuscripcion: 1987
};
```

Ahora debemos definir una función que permita obtener un resumen de la información registrada de manera simple. Por ejemplo:

```
> resumenInformacion(juan)
"Juan Arrever se suscribió hace 28 años y leyó 3 ejemplares"

> resumenInformacion(elena)
"Elena Chalver se suscribió hace 33 años y leyó 2 ejemplares"
```

Definí la función `resumenInformacion` que nos permita obtener la información requerida. Asumí que estamos en 2021.

Solución >\_ Consola

```
1 function resumenInformacion(registro){
2   return registro.nombre + " se suscribió hace " + (2021 -
  registro.anioSuscripcion) + " años y leyó " +
  longitud(registro.librosLeidos) + " ejemplares";
3 }
```

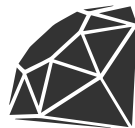
▶ Enviar

✔ ¡Muy bien! Tu solución pasó todas las pruebas



🕒 Te quedan 01:24:58

# Ejercicio 7: Ejercicio 7



¡Dejemos atrás a JavaScript para pasar a Ruby!

Vamos a modelar `Almuerzo` s para poder:

- agregarle cucharadas de sal;
- ver si tiene demasiada sal, es decir, si tiene más de 7 cucharadas de sal.

Definí en Ruby, la clase `Almuerzo` que tenga un atributo `@cucharadas_sal` con su getter. Las instancias de esta clase entienden los mensajes `agregar_cucharadas!` (que recibe la cantidad a agregar por parámetro) y `exceso_de_sal?`. No te olvides de definir un `initialize` que reciba las cucharadas de sal iniciales como parámetro.

 Solución  Consola

```
1 class Almuerzo
2   def initialize(cucharadas)
3     @cucharadas_sal = cucharadas
4   end
5
6   def cucharadas_sal
7     @cucharadas_sal
8   end
9
10  def agregar_cucharadas!(cantidad)
11    @cucharadas_sal += cantidad
12  end
13
14  def exceso_de_sal?
15    @cucharadas_sal > 7
16  end
17
18 end
```

 Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas





 Te quedan 01:24:54

# Ejercicio 8: Ejercicio 8



Los compilados son discos que tienen la característica de recopilar canciones que comparten alguna característica, por ejemplo artista, época o género. Algunas de ellas con mayor duración que otras.

Teniendo en cuenta que las canciones saben responder al mensaje `demasiado_corta?` ...

Definí en Ruby el método `cantidad_de_canciones_cortas` que responda a cuántas canciones cortas tiene el `Compilado`.

 Solución

 Consola

```
1 module Compilado
2   @canciones = [AmorAusente, Eco, Agujas, ElBalcon,
3                 GuitarrasDeCarton]
4
5   def self.cantidad_de_canciones_cortas
6     @canciones.count {|cancion| cancion.demasiado_corta?}
7   end
8 end
```

 Enviar

 ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Franco Bulgarelli, Rocío Gonzalez bajo los términos de la [Licencia Creative Commons Compartir-Igual, 4.0](#).





🕒 Te quedan 01:24:49

# Ejercicio 9: Ejercicio 9



A la hora de relajarse muchas `Persona`s juegan con su mascota. Los animales hacen distintas cosas cuando juegan:

- A los `Perro`s les da hambre;
- los `Conejo`s incrementan en 6 su nivel de felicidad;
- las `Tortuga`s no hacen nada.

Definí el método `jugar_con_mascota!` en la clase `Persona` y el método `jugar!` en los distintos tipos de animales. Definí los getters necesarios en cada una.

🔍 Solución

➤ Consola

```
1 class Persona
2   def initialize(mascota)
3     @mascota = mascota
4   end
5
6   def mascota
7     @mascota
8   end
9
10  def jugar_con_mascota!
11    @mascota.jugar!
12  end
13
14 end
15
16 class Perro
17   def initialize()
18     @tiene_hambre = false
19   end
20
21   def tiene_hambre
22     @tiene_hambre
23   end
24
25   def jugar!
26     @tiene_hambre = true
27   end
28
29 end
30
31 class Conejo
32   def initialize(nivel_de_felicidad)
33     @nivel_de_felicidad = nivel_de_felicidad
34   end
35
36   def nivel_de_felicidad
37     @nivel_de_felicidad
38   end
39
40   def jugar!
41     @nivel_de_felicidad += 6
42   end
43
44 end
45
46 class Tortuga
47
48   def jugar!
49   end
50 end
```

▶ Enviar

✅ ¡Muy bien! Tu solución pasó todas las pruebas

