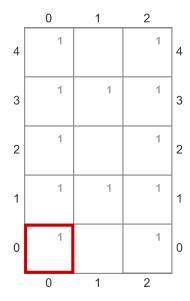
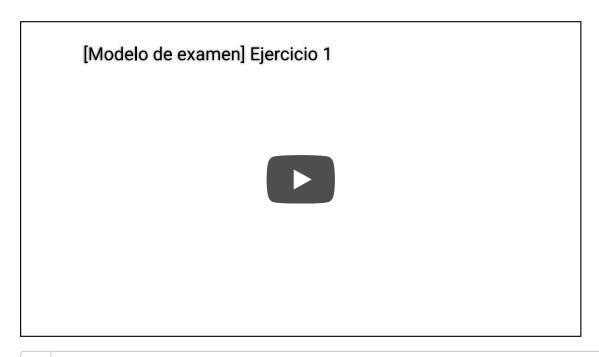




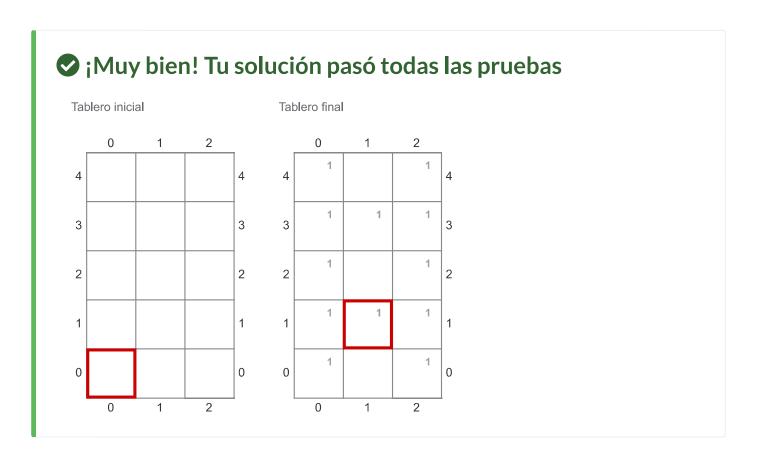
Con el tablero y las bolitas de Gobstones podemos representar muchos objetos. En este caso queremos hacer una escalera 🔂 como esta:



Creá un programa que dibuje una escalera de color Negro. El cabezal empieza en el origen (o sea, en el borde Sur-Oeste) pero no te preocupes por dónde finaliza.



```
1 program {
 2
     repeat(4){
                                                                              三
 3
       Poner(Negro)
 4
       Mover(Norte)
 5
     Poner(Negro)
 6
     Mover(Sur)
 7
     Mover(Este)
 8
 9
     Poner(Negro)
     Mover(Este)
10
11
     IrAlBorde(Norte)
     repeat(4){
12
       Poner(Negro)
13
14
       Mover(Sur)
     }
15
     Poner(Negro)
16
     Mover(Oeste)
17
     Mover(Norte)
18
     Poner(Negro)
19
20 }
```



Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.







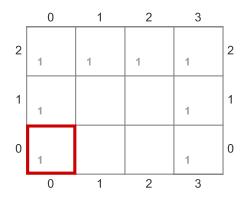




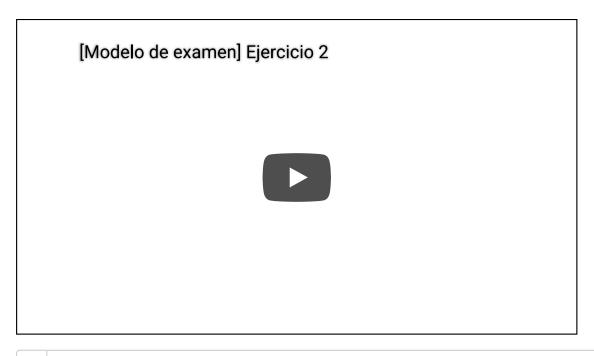


•

Queremos dibujar una mesa que quede bien con cualquier decoración  $\Theta$ . Para ello vamos a definir un procedimiento DibujarMesa que tome un color y dibuje una mesa. Por ejemplo, si lo invocaramos con Rojo haciendo DibujarMesa(Rojo), la misma se vería así:



Definí el procedimiento DibujarMesa para que dibuje una mesa como la que te mostramos del color que reciba como argumento. El procedimiento debe dibujar desde el extremo Sur Oeste pero no te preocupes por dónde termina el cabezal.



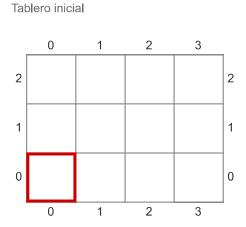
- 1 procedure DibujarMesa(color){
- 2 Pata1(color)

```
3
     Mover(Este)
     Poner(color)
 4
 5
     Mover(Este)
     Poner(color)
 6
     Mover(Este)
 7
     Pata2(color)
 8
 9
   }
10
11 procedure Pata1(color){
     repeat(2){
12
13
       Poner(color)
       Mover(Norte)
14
15
     Poner(color)
16
17 }
18 procedure Pata2(color){
19
     repeat(2){
       Poner(color)
20
21
       Mover(Sur)
22
23
     Poner(color)
24 }
```

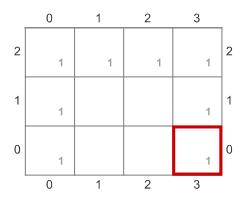
#### ¡Muy bien! Tu solución pasó todas las pruebas

#### Resultados de las pruebas:



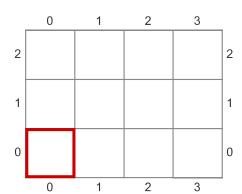










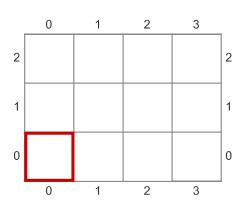


#### Tablero final

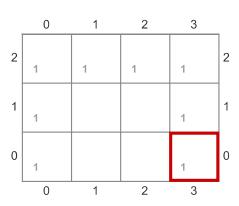
	0	1	2	3	
2	1	1	1	1	2
1	1			1	1
0	1			1	0
	0	1	2	3	-



#### Tablero inicial

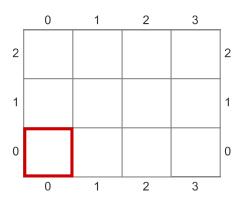


Tablero final

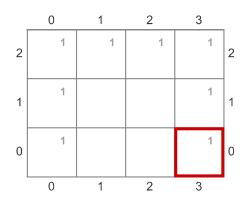




Tablero inicial



Tablero final



Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2021 💪 Mumuki

Información importante Términos y Condiciones Reglas del Espacio de Consultas













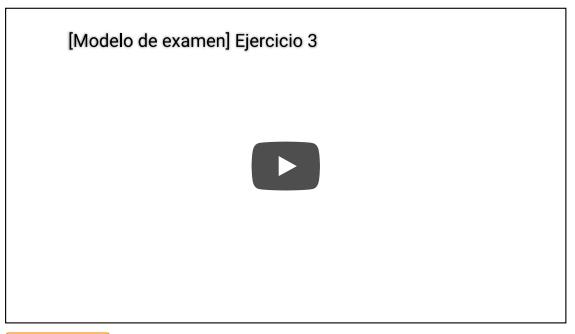
oora vamos a dofinir una fur

Ahora vamos a definir una función concatenacionEsIgual que recibe tres strings y nos dice si la concatenación de los dos primeros es igual al tercero (?):

```
    concatenacionEsIgual("mari", "posa", "langosta")
    false

    concatenacionEsIgual("guarda", "ropas", "guardaropas")
    true
```

Definí la función concatenacion Es Igual.



```
✓ Solución >_ Consola
```

```
function concatenacionEsIgual(string1,string2,concatenacion){
  return string1 + string2 === concatenacion
}
```

22/12/21	3:19 Modelo de examen - Ejercicio 3 - Argentina Programa
	▶ Enviar

**⊘** ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.









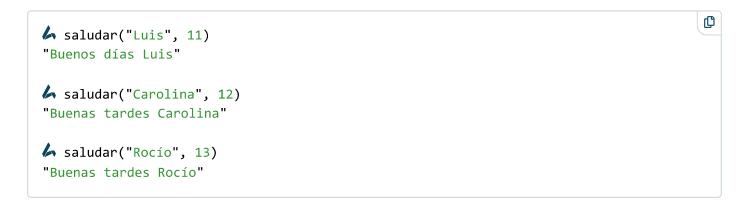




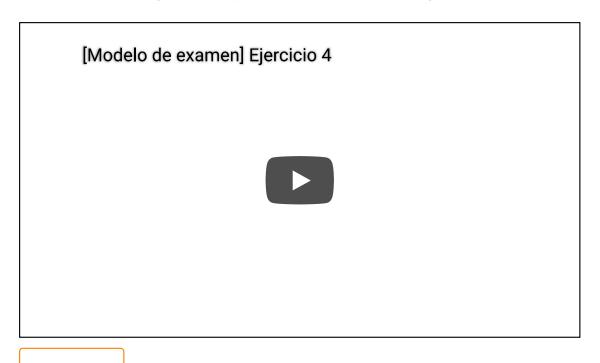
•

Ya sabemos que es muy importante saludar al llegar a un lugar, el problema está en que el saludo depende de la hora del día. ¡Por suerte este problema también se puede resolver programando!

Podríamos tener una función que dado un nombre y un horario retorne el saludo correcto. Si es antes de las 12 debería ser "Buenos días", si es después "Buenas tardes". Por ejemplo:



Definí la función saludar.





```
function saludar(nombre,hora){
   if (hora < 12) {
     return "Buenos días " + nombre;}
   else {return "Buenas tardes " + nombre;}
}</pre>
```

¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.











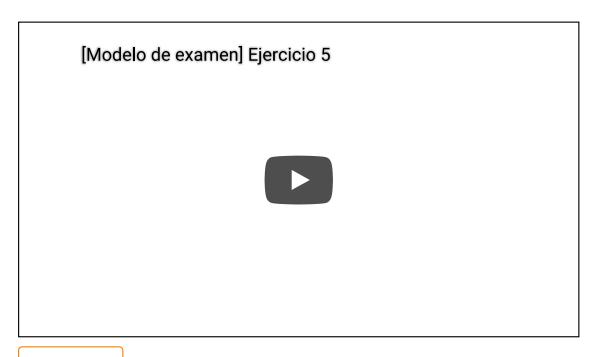


Vamos a sumarle (ⓐ) funciones a una extravagante calculadora. Queremos obtener la suma de los elementos mayores a 7 de una lista. Por ejemplo:

```
sumaDeLosMayoresASiete([2, 5, 9, 8, 11])
28 //Porque es la suma de 9, 8 y 11.
```

Definí la función sumaDeLosMayoresASiete.

En este video te dejamos una posible resolución a este ejercicio:



Solución

>\_ Consola

```
function sumaDeLosMayoresASiete(numeros){
  let sumatoria = 0;
  for (let numero of numeros){
    if (numero > 7){
       sumatoria += numero
    }
}
return sumatoria;
```





Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.













Una conocida aplicación para escuchar música online quiere hacer un resumen con la información importante de sus canciones ☐. Las canciones se almacenan como registros de la siguiente forma:

```
let elGenioDeLaNada = {
  nombre: "El genio de la nada",
  banda: "Eruca Sativa",
  duracion: 4
};

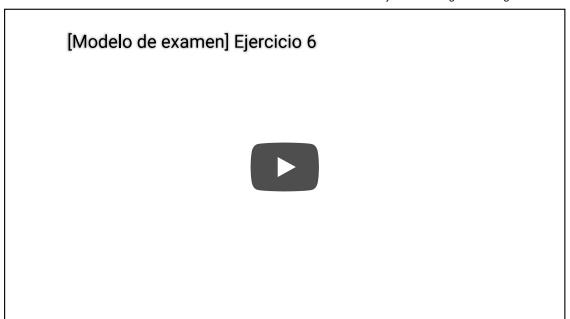
let lotusFlower = {
  nombre: "Lotus flower",
  banda: "Radiohead",
  duracion: 5
};
```

Queremos definir una función que retorne un resumen de manera simple. Por ejemplo:

```
resumenCancion(elGenioDeLaNada)
"El genio de la nada de la banda Eruca Sativa tiene una duración de 240 segundos"

resumenCancion(lotusFlower)
"Lotus flower de la banda Radiohead tiene una duración de 300 segundos"
```

Definí la función resumenCancion que nos permita obtener el resumen solicitado.



#### ♀¡Dame una pista!

```
Solución >_ Consola
```

Enviar

¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.













¡Dejemos atrás a JavaScript para pasar a Ruby!

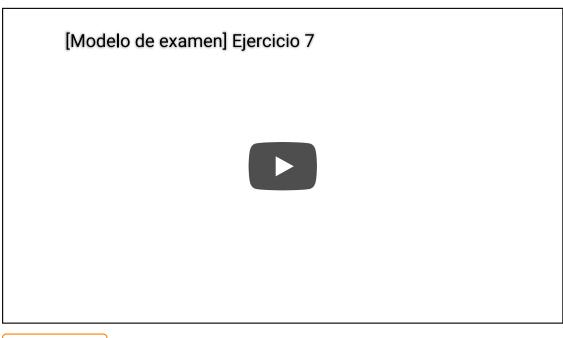
Vamos a tomar unos mates 🗳, mejor aún ¡programémoslo! 😭

Para eso vamos a crear el objeto AguaParaMate para poder:

- calentarla una temperatura (si la calentamos 10 grados, su temperatura incrementa en 10);
- ver si tiene la temperatura correcta, es decir, si está a exactamente 80 grados.

Definí en Ruby, el objeto AguaParaMate que tenga un atributo @temperatura que inicialmente está en O con su getter. AguaParaMate entiende los mensajes calentar\_agua! (que recibe la cantidad de grados a sumar por parámetro) y temperatura\_exacta?.

En este video te dejamos una posible resolución a este ejercicio:





**>\_** Consola

1 module AguaParaMate

2 @temperatura = 0

```
3
 4
     def self.temperatura
 5
       @temperatura
 6
     end
 7
    def self.calentar_agua!(grados)
 8
       @temperatura += grados
 9
10
     end
11
     def self.temperatura_exacta?
12
13
       @temperatura == 80
14
     end
15 end
```

#### ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015-2021 Mumuki
Información importante
Términos y Condiciones

Reglas del Espacio de Consultas











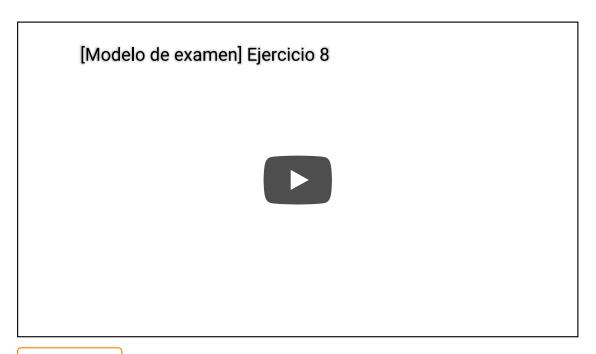


Que hermoso leer y transportarse a otros mundos. Por eso tenemos una Biblioteca con algunos libros. 🗏

Teniendo en cuenta que los libros saben responder al mensaje nombre ...

Definí en Ruby el método nombres\_de\_libros que responda el nombre de los libros de la Biblioteca.

En este video te dejamos una posible resolución a este ejercicio:



```
Solución
```

>\_ Consola

```
module Biblioteca

@libros = [Fundacion, Contacto, LaInsoportableLevedadDelSer,
Socorro, ComoAguaParaChocolate]

def self.nombres_de_libros
    @libros.map { |libro| libro.nombre}
end
end
```



¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.













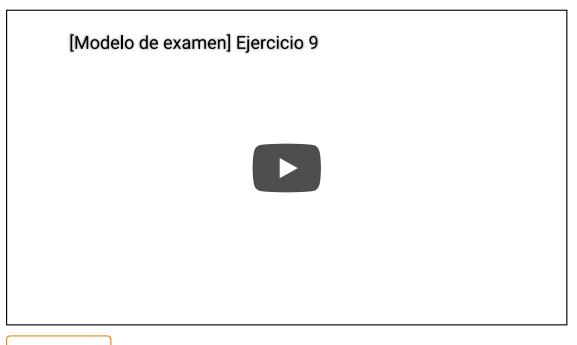
Una empresa de mudanzas, cuenta con varios transportes, para realizarlas elige un Camion.

Cada vez que un Camion carga un mueble, el mueble sufre una modificación:

- cada Colchon pierde 4 resortes;
- cada Sillon sube su nivel\_de\_polvo en 20;
- a los Electrodomestico s no les pasa nada.

Definí los metodos cargar\_muebles! en la clase Camion. También definí el método ser\_cargado! en los muebles junto con su respectivo getter.

En este video te dejamos una posible resolución a este ejercicio:



```
Solución
```

>\_ Consola

```
1 class Camion
2  def initialize(muebles)
3   @muebles = muebles
4  end
5  def cargar_muebles!
```

```
6
       @muebles.each{|mueble|mueble.ser cargado!}
 7
 8 end
 9
10 class Colchon
11
     def initialize(resortes)
12
       @cantidad de resortes = resortes
13
     def cantidad de resortes
14
       @cantidad_de_resortes
15
16
     end
17
     def ser cargado!
18
       @cantidad de resortes -=4
19 end
20 end
21 class Sillon
     def initialize(polvo)
22
23
       @nivel_de_polvo = polvo
24
     end
25
     def nivel_de_polvo
       @nivel_de_polvo
26
27
     end
28
     def ser_cargado!
       @nivel de polvo +=20
29
30
     end
31 end
32
33 class Electrodomestico
34
     def ser_cargado!
35
     end
36 end
```

#### ¡Muy bien! Tu solución pasó todas las pruebas

Esta guía fue desarrollada por Gustavo Trucco, Rocío Gonzalez, Mayra Mosqueira bajo los términos de la Licencia Creative Commons Compartir-Igual, 4.0.

© 2015, 2021 in poMameuki

Términos y Condiciones

Reglas del Espacio de Consultas







