Desafío - Ciclos

Desafío 1

En el siguiente código, reemplazar la instrucción `while` por `times`.

La impresión debe ser la misma:

```
i = 0
while i < 50
  puts "Iteración #{i}"
  i = i + 1
end
```

> Tip: Cuidado con condición

Desafío 2

En el siguiente código, reemplaza la instrucción `until` por `while`.

La impresión debe ser la misma:

```
puts 'Ingrese un número para comenzar la cuenta: '
cuenta_regresiva = ARGV[0].to_i

puts "Contando desde #{cuenta_regresiva}..."
until cuenta_regresiva < 0
  puts cuenta_regresiva
  cuenta_regresiva -= 1
end
```

Uso:

ruby primeros_pares 10

Desafío 3.a

Crea un programa llamado `solo_pares.rb` que muestre los primeros n números pares, donde n es ingresado por el usuario.

Uso:

`ruby solo_pares 5`

```
0
2
4
6
8
```

Desafío 3.b

Crear una variante del programa anterior llamado `solo_pares2.rb` pero que en este el cero no sea considerado (el cero no es par)

Uso:

`ruby solo_pares2 5`

```
2
4
6
8
10
```

Desafío 4

Crea un programa llamado `solo_impares.rb` que dado n muestre en pantalla los primeros n números impares.

> Tip: el número siguiente a un par siempre es un impar :)

Uso:

solo_impares 5

```
1
3
5
7
9
```

Desafío 5

Crea un programa llamado suma_pares.rb que sume los primeros n números pares, donde n es ingresado por el usuario por linea de comandos.

> Tip: El cero no es par, no afecta en la suma pero tenemos que tener cuidado con los bordes del ciclo.

Uso:

suma_pares.rb 10

Desafío 6

Crear un programa llamado `lorem_generator.rb` en ruby que sea capaz de mostrarn en pantalla varios parrafos de Lorem ipsum, donde el número de párrafos se especifica al cargar el script. (El texto puede ser extraído del primer párrafo de https://www.lipsum.com/feed/html)

Uso:

ruby lorem_generator.rb

> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi ac lacinia nibh, nec faucibus enim. Nullam quis lorem posuere, hendrerit tellus eget, tincidunt ipsum. Nam nulla tortor, elementum in elit nec, fermentum dignissim sapien. Sed a mattis nisi, sit amet dignissim elit. Sed finibus eros sit amet ipsum scelerisque interdum. Curabitur justo nibh, viverra a elit vel, elementum hendrerit erat. Duis feugiat mattis ante vel hendrerit. Etiam nec nibh nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi ac lacinia nibh, nec faucibus enim. Nullam quis lorem posuere, hendrerit tellus eget, tincidunt ipsum. Nam nulla tortor, elementum in elit nec, fermentum dignissim sapien. Sed a mattis nisi, sit amet dignissim elit. Sed finibus eros sit amet ipsum scelerisque interdum. Curabitur justo nibh, viverra a elit vel, elementum hendrerit erat. Duis feugiat mattis ante vel hendrerit. Etiam nec nibh nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

Desafío 7

Sabiendo que `"a.next" => b` y `"b.next" => c`. Crear un método llamado `gen` que reciba el número de letras a generar y devuelva un string con todas las letras generadas concatendas.

Ejemplo:

gen 4

> "abcd"

gen 10

> "abcdefghij"

> Tip: Los ejercicios que piden métodos se evalúan llamando al método directamente y comparando el resultado, para tener la evaluación correcta del ejercicio considera el nombre del método y el resultado. No es necesario que el programa tenga una salida o muestre en pantalla por si solo.

Desafío 8

Se busca crear un programa `fuerza_bruta.rb` que revise cuantos intentos requiere hackear un password por fuerza bruta.

Uso:

ruby fuerza_bruta pass

> 282404 intentos

ruby fuerza_bruta passwo

> 190906392 intentos

Luego el sistema intentará con todas las combinaciones de letras:

- Primero probará con a, luego b, luego c ... luego con z, luego ab, ac, .. az, aba ... azz ... zzz, aaaa ...

Se supone que el password solo contiene letras.**

> Tip: partir con intento = 'a'