

## Desafío - Ciclos

- Para realizar este desafío debes haber estudiado previamente todo el material disponible correspondiente a la unidad.
- Una vez terminado el desafío, comprime la carpeta que contiene el desarrollo de los requerimientos solicitados y sube el `.zip` en el LMS.
- Puntaje total: **10 puntos**
- Desarrollo desafío:
  - El desafío se debe desarrollar de manera Individual.

## Instrucciones

A continuación se detallan variados desafíos a desarrollar. Para su correcta evaluación, los programas deben ser almacenados en un archivo comprimido `.zip` de la siguiente manera:

```
desafios.zip
├── cuenta_regresiva.rb
├── fuerza_bruta.rb
├── gen.rb
├── iterador.rb
├── lorem_generator.rb
├── solo_impares.rb
├── solo_pares2.rb
├── solo_pares.rb
└── suma_pares.rb
```

## Requerimientos

### Iterador

En el siguiente código, reemplazar la instrucción `while` por `times` dentro del programa llamado `iterador.rb`.

La impresión debe ser la misma:

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



**Tip:** Cuidado con la condición.

## Cuenta regresiva

En el siguiente código, reemplaza la instrucción `until` por `while` dentro del programa llamado `cuenta_regresiva.rb`.

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 cuenta_regresiva.rb 10
```

```
Contando desde 10... 10 9 8 7 6 5 4 3 2 1
```

## Solo pares 1

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.rb 5
```

```
0 2 4 6 8
```

## Solo pares 2

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.rb 5
```

```
2 4 6 8 10
```

## Solo impares

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:

```
ruby solo_impares.rb 5
```

```
1 3 5 7 9
```

## Suma pares

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



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

Uso:

```
ruby suma_pares.rb 20
```

```
420
```

## Lorem generator

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

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.
```

## Gen

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

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.

## Fuerza bruta

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

Uso:

```
ruby fuerza_bruta.rb pass
```

```
282404 intentos
```

```
ruby fuerza_bruta.rb 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'.