Introducción a patrones con ciclos

¡Comencemos!

¿Qué aprenderás?

- Reconocer patrones de repetición en un ciclo.
- Determinar si un número es par o impar.
- Utilizar la paridad de un número para dibujar patrones

Introducción

Una de las dificultades más frecuentes -en principiantes- al momento de resolver problemas de ciclos es la de identificar/entender el patrón.

No entender un patrón de manera rápida e instantánea no nos hace menos inteligentes. A pesar de que hay personas que pueden resolver estos problemas de manera intuitiva, la mayoría de nosotros tuvo que aprender a resolverlos.

En este capítulo vamos a resolver problemas desde muy sencillos hasta complejos, identificando el patrón.

¡Vamos con todo!

Ejercicio - Dibujando puntos

Crear el programa `solo_puntos.rb` que dibuje `n` puntos. Donde `n` es un valor ingresado por el usuario al momento de ejecutar el script.

Uso:

```
ruby solo_puntos.rb 5
resultado:
*****

ruby solo_puntos.rb 1
resultado:
*
```

Solución no recomendada

El primer intento para resolverlo podría ser utilizando instrucciones if:

```
n = ARGV[0].to_i
if n == 1
   puts '*'
elsif n == 2
   puts '**'
elsif n == 3
   puts '***'
elsif n == 4
   puts '****'
elsif n == 5
   puts '*****'
end
```

Sin embargo, la solución es bastante limitada. ¿Qué sucedería si el usuario ingresa el valor 6? ¿ o 7? ¿ o 100?

¿Vamos a programar todas las opciones hasta 100?. Este tipo de problemas se resuelve mucho mejor con ciclos.

Solución 2

Para resolver el problema con ciclos debemos, simplemente, identificar el patrón. Si el usuario ingresa 1, se dibuja un asterisco; si el usuario ingresa 2, se dibujan 2 asteriscos. Si el usuario ingresa `n` entonces hay que dibujar `n` asteriscos .

```
n = ARGV[0].to_i
n.times do
   print '*' # Tenemos que utilizar `print` en lugar de `puts`
   # porque `puts` insertaría automáticamente un salto de línea
end
```

Solución 3

En Ruby es fácil lograr una solución sin ciclos, multiplicando `"*"` con `n`:

```
n = ARGV[0].to_i
print '*' * n
```

Pero la solución con ciclos es la que tenemos que poder lograr.

Ejercicio - Puntos y números

Crear el programa `puntos_y_numeros.rb` que dibuje `N` números intercalados por puntos. Donde `N` es un valor ingresado por el usuario al momento de ejecutar el script.

Uso:

```
ruby solo_puntos.rb 5
resultado:
0.2.4

ruby solo_puntos.rb 9
resultado:
0.2.4.6.8
```

¿Cuál es el patrón?

La primera pregunta es cuál es el patrón, en este caso vemos que los números impares son los que son reemplazados por puntos y los números pares muestran el mismo número. Para este experimento consideraremos el cero como número par.

- Si número es par => número.
- Si número es impar => punto.

¿Par o impar?

Hay dos formas de saber si un elemento es par.

- Utilizar el método `.even`.
- Utilizar la operación resto (o módulo).

Utilizando el método .even?

En ruby existe un método muy sencillo llamado .even?

```
2.even? # => true
3.even? # => false
```

Utilizando la operación resto

Pero existe otra forma un poco más flexible y muy utilizada en general que es calcular el resto de la operación. O sea, si al dividir un número por dos, la división es exacta, es par, y si queda un resto, entonces es impar.

```
5 % 2 == 0 # => false
5 % 2 == 1 # => true
```

Solución

```
n = ARGV[0].to_i
n.times do |i|
 if i % 2 == 0 # Si es par
   print i
```

```
 else
   print '.'
 end
end
```

Ejercicio - Dibujando asteriscos y puntos

Crear el programa `asteriscos_y_puntos.rb` que dibuje asteriscos y puntos intercalados hasta **n**. Donde **n** es un valor ingresado por el usuario al momento de ejecutar el script.

Uso:

```
ruby asteriscos_y_puntos.rb 3
resultado:
*.*
```

```
ruby asteriscos_y_puntos.rb 4
resultado:
*.*.
```

```
ruby asteriscos_y_puntos.rb 5
resultado:
*.*.*
```

Solución

Para resolver este ejercicio debemos iterar utilizando el índice, ya que debemos identificar si nos encontramos en una posición par o impar.

```
#n = ARGV[0].to_i
n = 8
n.times do |i|
   if i.even?
       print '*'
```

```
    else
        print "."
    end
end
```

Ejercicio - Dos por dos

Crear el programa `dos_por_dos.rb` que dibuje el siguiente patrón de asteriscos y puntos intercalando hasta **n**. Donde **n** es un valor ingresado por el usuario al momento de ejecutar el script.

```
ruby dos_por_dos.rb 5
**..*
```

```
ruby dos_por_dos.rb 6
**..**
```

Siempre hay que partir estudiando la solución. En este caso el patrón se repite cada 4 caracteres.

- caracter 1 y 2 => `*`.
- caracter 3 y 4 => `.`.

Para resolver este tipo de patrones podemos ocupar nuevamente la operación resto (o módulo) y como el patrón se repite cada 4 caracteres entonces utilizaremos el resto de 4.

i	i%4
0	0
1	1
2	2
3	3
4	0
5	1
6	2
7	3
8	0
9	1
10	2
11	3
12	0

Tabla 1. Patrón resto `i%4`.
Fuente: Desafío Latam.

- Si `i%4` es 0 o 1 mostraremos un `*`.
- En otro caso mostraremos un `.`.

```
n = ARGV[0].to_i # 24
n.times do |i|
   if i%4 == 0 || i%4 == 1
       print '*'
   else
       print "."
   end
end
```

```
**..**..**..**..**..**..
```

Ejercicio - Patrón

Escribir el programa patron3.rb que permita dibujar el siguiente patrón:

```
..**||..**||..**||
```

Ejemplo de uso

```
ruby patron3.rb 4
..**||..
```