Reutilizando códigos

Reutilizando códigos 1
- ¿Qué aprenderás? 2
- Introducción 2
  - El mismo propósito 3
  - No son exactamente lo mismo 3
- Métodos 4
  - Llamar vs Definir 4
- Definiendo métodos 4
- El espacio principal (main) 4

¡Comencemos!

¿Qué aprenderás?

- Conocer la importancia de los métodos.
- Diferenciar entre la definición de un método y su llamado.

Introducción

Los métodos nos permiten abstraernos del cómo resolvemos los problemas, esto lo logramos agrupando instrucciones bajo un nombre. Supongamos que tenemos el siguiente algoritmo:

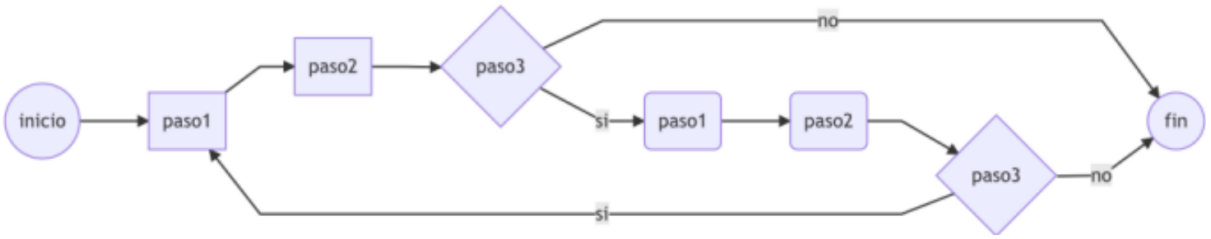

Imagen 1. Ejemplo Algoritmo.
Fuente: Desafío Latam.

¿Qué pasaría si renombramos el paso1 y el paso2 y el paso3 a pasoX?, ¿sería lo mismo?

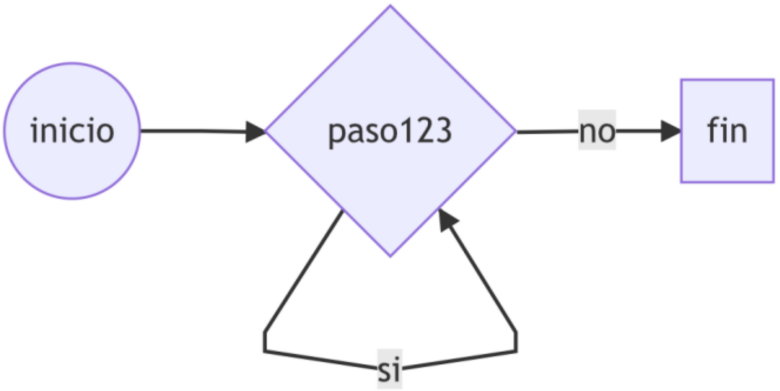

Imagen 2. Pasos renombrados.
Fuente: Desafío Latam.

Esta forma de agrupar instrucciones recibe diferentes nombres según el lenguaje de programación:

- Procedimiento.
- Subrutina.
- Función.
- Método.

El mismo propósito

Pero todas estas tienen el mismo propósito: Evitar que el programador repita (o copie y pegue) código, esto se llama enfoque **DRY** (Do not Repeat Yourself).

No son exactamente lo mismo

Si bien existen diferencias entre procedimiento, subrutina, función y método; por ahora las ignoraremos, ya que en Ruby sólo trabajaremos con métodos.

¡Vamos con todo!

Métodos

Los métodos nos permiten agrupar instrucciones y reutilizarlas.

Podemos crear nuestros propios métodos, a esto le llamaremos definir.

Al utilizar métodos ya creados por nosotros, o por otras personas, le denominaremos llamar.

Llamar vs Definir

- Cuando creemos un método diremos que lo estamos definiendo.
- Cuando ocupemos un método diremos que lo estamos llamando, usando o invocando.

Dentro de Ruby hay varios métodos definidos que hemos utilizado, por ejemplo:

- gets.
- puts.

Definiendo métodos

Existen distintos lugares donde se puede definir métodos y eso tiene consecuencias sobre cómo los llamamos.

- **El espacio principal de trabajo, ejemplo:** `gets` y `puts` y `rand`.
- **Dentro de objetos, ejemplo:** `2.even?`.
- **Dentro de clases, ejemplo:** `File.open`.
- **Dentro de módulos, ejemplo:** `Math.sqrt`.

El espacio principal (main)

En esta unidad trabajaremos definiendo los métodos en el espacio principal. Sabemos que un método está definido dentro del espacio principal cuando lo ocupamos directamente, sin anteponer un objeto, clase o módulo. Los métodos suelen organizarse en torno a clases y módulos, y podemos cargarlas desde un archivo o desde **gemas**. En los próximos capítulos aprenderemos a crear nuestros propios métodos.