


# Creación y uso de variables en Rust.

Los desarrolladores escriben programas para trabajar con datos. Los datos se recopilan, analizan, almacenan, procesan, comparten y notifican. Usamos *variables* para almacenar nuestros datos en una referencia con nombre, que podremos consultar más adelante en el código.

## Variables.

En Rust, una variable se declara con la palabra clave **let**. Cada variable tiene un **nombre único**. Cuando se declara una variable, se le puede asignar un valor o puede ser asignado con posterioridad en el programa. El código siguiente declara una variable denominada (a\_number), la cual no tiene asignado ningún valor.


Rust

 Copiar

```
let a_number;
```

Podemos modificar esta instrucción para asignarle un valor a la variable:

Rust

 Copiar

```
let a_number = 10;
```

### ⓘ Nota

**Palabras clave** Al igual que con otros lenguajes de programación, determinadas *palabras clave*, como **fn** y **let**, están reservadas para que las use Rust únicamente. Las palabras clave no se pueden usar como nombres de funciones o variables.

Veamos otro ejemplo. El código siguiente declara dos variables.

La primera variable se declara, pero no se le asigna un valor. La segunda variable se declara y asigna un valor. Más adelante en el programa, el valor de la primera variable se asigna a una palabra. El código llama a la macro `println!` para mostrar los valores de la variable.

```
Rust Copiar

// Declare a variable
let a_number;

// Declare a second variable and bind the value
let a_word = "Ten";

// Bind a value to the first variable
a_number = 10;

println!("The number is {}.", a_number);
println!("The word is {}.", a_word);
```

Salida:

```
Resultados Copiar

The number is 10.
The word is Ten.
```

Si hubiesemos llamado a la macro `println!` e intentado mostrar el valor de la variable `a_number` antes de asignarle un valor, el compilador habria devuelto un error.

## Inmutable VS Mutable:

En Rust, las asignaciones de valores a las variables son **inmutables por defecto**. Una variable es inmutable, cuando después de habersele asignado un valor, este no puede ser modificado.


Por ejemplo, si intentamos cambiar el valor de la variable `a_number` del ejemplo anterior, recibiremos un mensaje de error del compilador.

```
Rust Copiar

// Change the value of an immutable variable
a_number = 15;
```

Para poder reasignar un valor a una variable, debemos utilizar la palabra clave **mut**.

Rust


 Copiar

```
// The `mut` keyword lets the variable be changed
let mut a_number = 10;
println!("The number is {}.", a_number);

// Change the value of an immutable variable
a_number = 15;
println!("Now the number is {}.", a_number);
```

Esto imprime la siguiente salida:

Resultados

 Copiar


```
The number is 10.
Now the number is 15.
```

## Propiedad reemplazada de variables(SHADOWING).

Puede declarar una variable nueva que use el nombre de una existente. La declaración nueva crea un enlace. En Rust, esta operación se denomina "**SHADOWING**" porque la nueva variable prevalece sobre la anterior. La antigua variable sigue existiendo, pero ya no se puede hacer referencia a ella en este ámbito.

En el código siguiente se muestra cómo usar el shadowing. Declaramos una variable denominada shadow\_num. No definimos la variable como mutable porque cada operación let crea una variable nueva denominada shadow\_num mientras se reemplaza la propiedad del enlace de la variable anterior.

Rust

 Copiar

```
// Declare first variable binding with name "shadow_num"
let shadow_num = 5;

// Declare second variable binding, shadows existing variable "shadow_num"
let shadow_num = shadow_num + 5;

// Declare third variable binding, shadows second binding of variable "shadow_num"
let shadow_num = shadow_num * 2;

println!("The number is {}.", shadow_num);
```

