Ownership

This lesson discusses ownership and its rules.

We'll cover the following

What Is Ownership?

Ownership in simple terms means to have possession of something.

Let’s look at a real-life analogy to explain this concept. If something belongs to you, you say “It’s mine”.

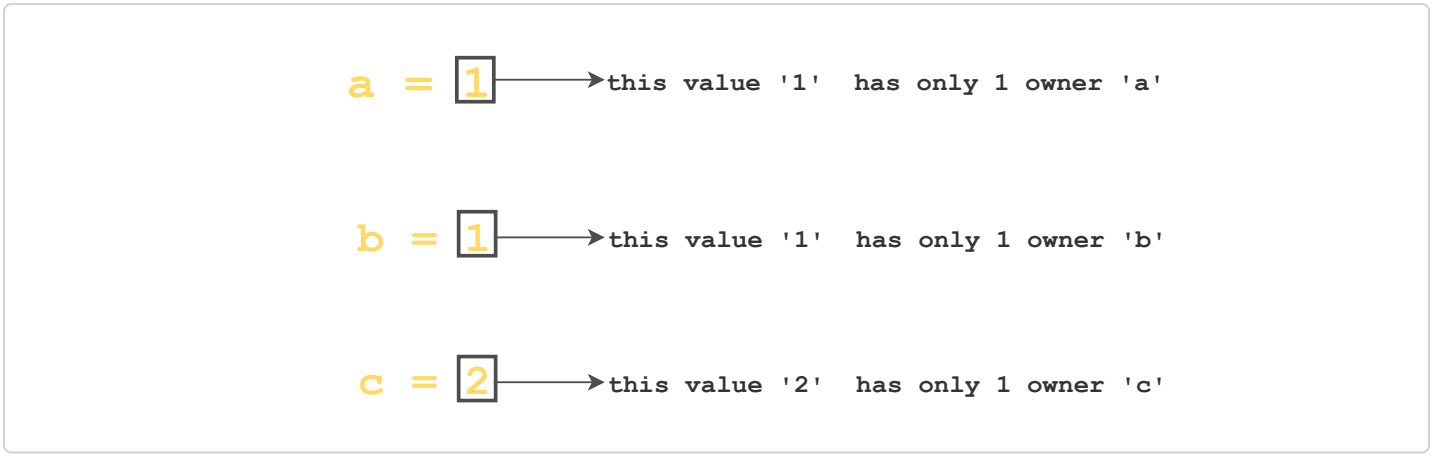
Similarly, in Rust, variable bindings can have ownership of what they are bound to.

Three Rules of Ownership

The following are three rules of ownership:

Rule # 1

Each value has a variable binding called its owner.



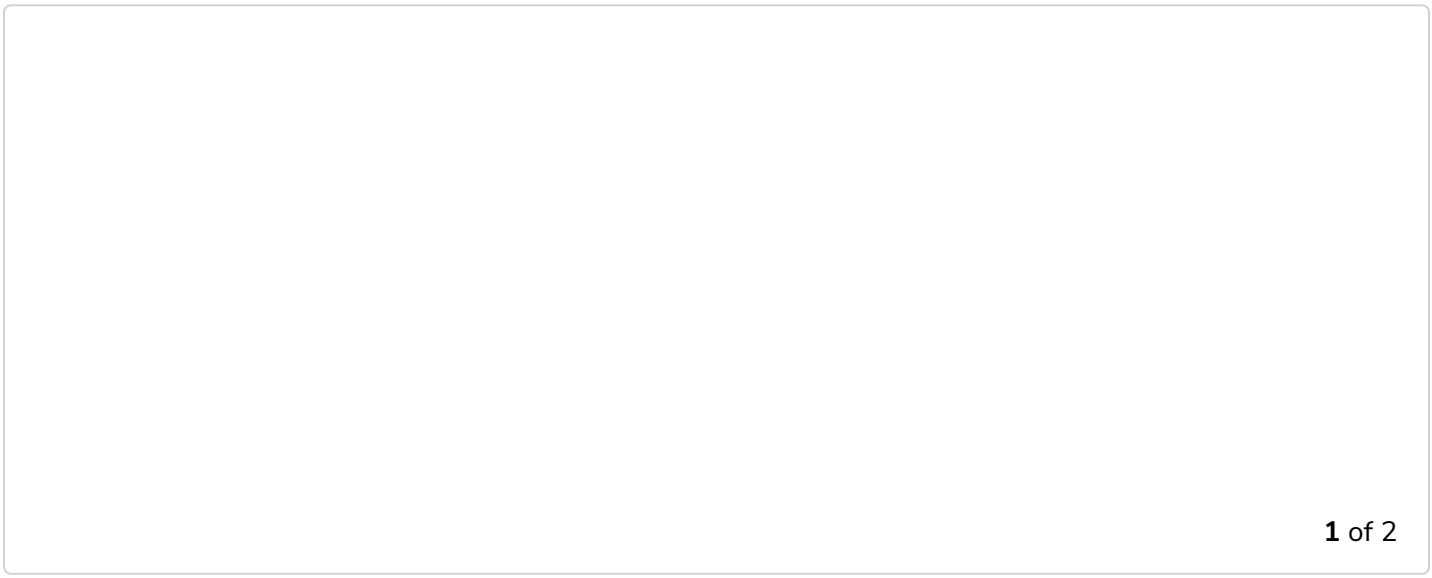
Rule # 2

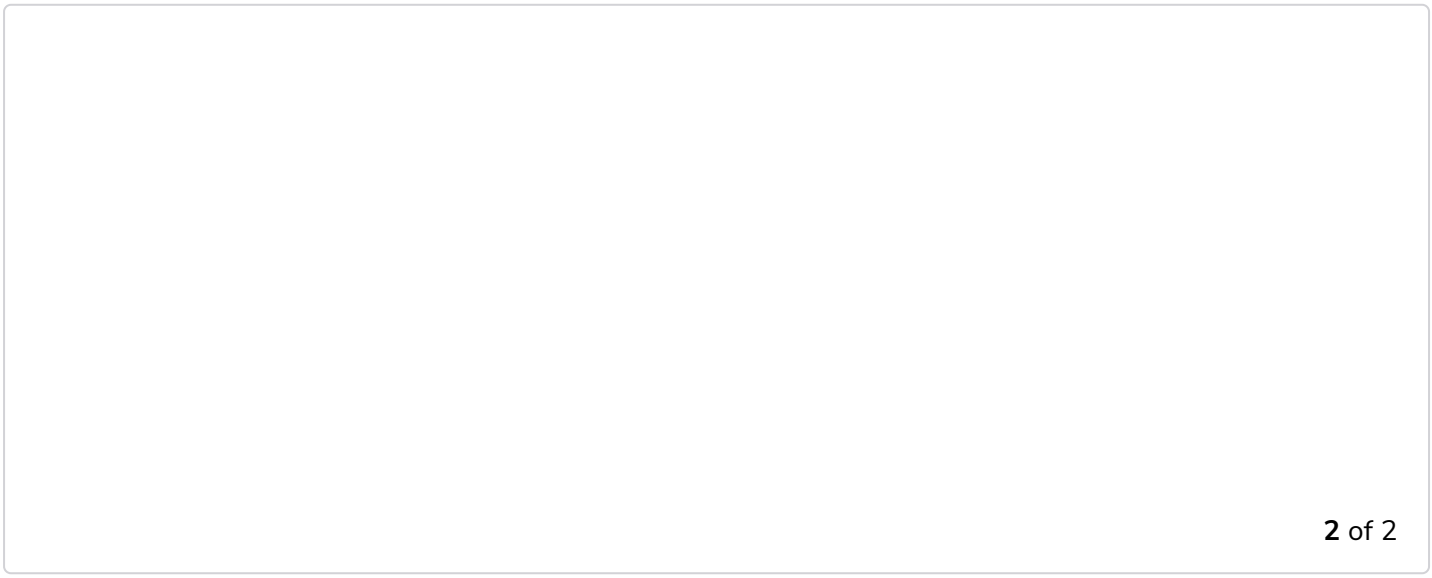
There can only be one owner at a time.

Rule # 3

When an owner goes out of scope, it does not remain accessible.

> When the variable goes out of scope, Rust calls function `drop` automatically at the closing curly bracket (to deallocate the memory).






```
fn main() {
  let a = 1; // variable a is the owner of the value 1
  let b = 1; // variable b is the owner of the value 1
  let c = 3; // variable c is the owner of the value 3

  println!("a : {}", a);
  println!("b : {}", b);
  println!("c : {}", c);
}// value a, b, c are out of scope outside this block
```

When using the assignment, two situations happen:

- The value gets *copied*
- The value gets *moved*

Let’s discuss each type in detail in the next lesson.