# What Are Variables?

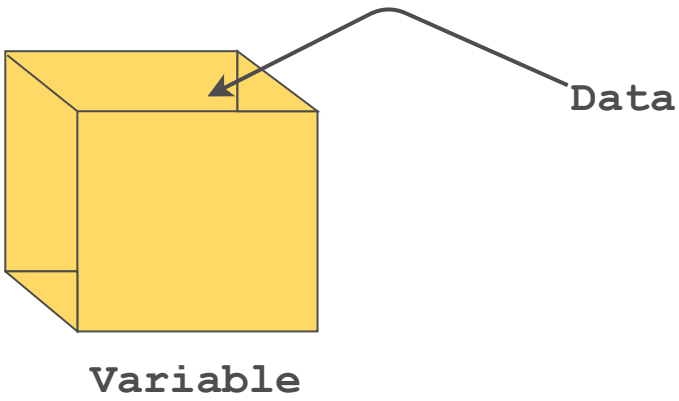
This lesson teaches what variables are and how they can store data.

**We'll cover the following**



## Variables #

A variable is like a **storage box** paired with an **associated name** which contains **data**. The associated name is the identifier and the data that goes inside the variable is the value. They are **immutable by default**, meaning, you cannot reassign value to them.




## Create a Variable #

To create a variable, use the `let` binding followed by the variable name.

> **What is binding?**

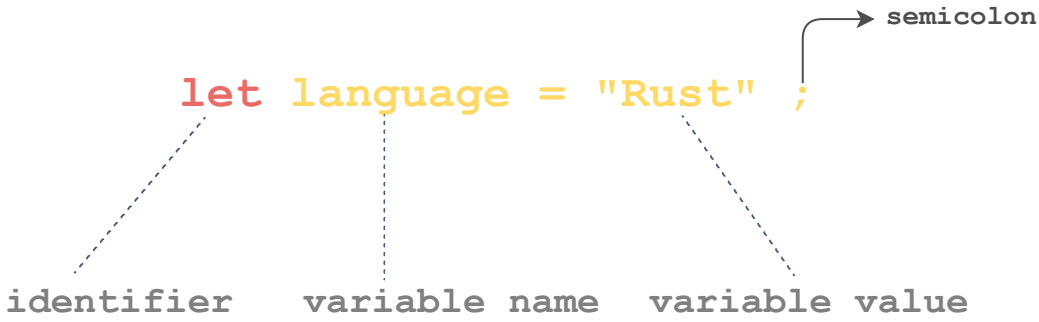
Rust refers to **declarations** as bindings as they bind a name at the time of creation. `let` is a kind of **declaration statement**.


**Naming Convention:** By convention, you would write a variable name in a **snake_case** i.e.,

- All letters should be lower case.
- All words should be separated using an underscore ( _ ).

## Initialize a Variable #

A variable can be initialized by assigning a value to it when it is declared. The value is said to be bound to that variable.

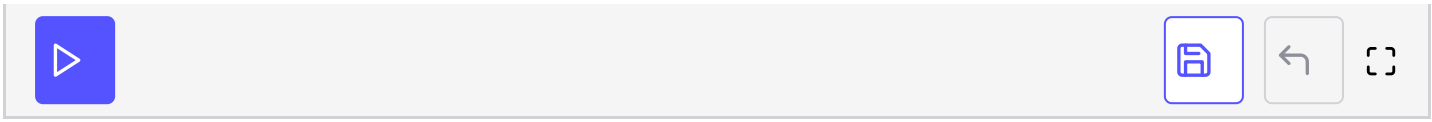



**Note:** It's possible to declare the variable first and assign it a value later. However, it is not recommended to do this as it may lead to the use of uninitialized variables.

The example below declares a variable, `language` , and initializes it with a value, `Rust` , and then displays the value of said variable:

```
fn main() {
```

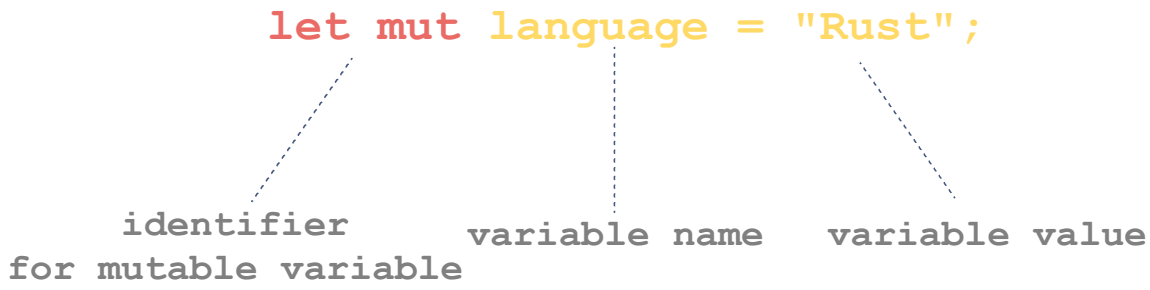
```
    let language = "Rust"; // define a variable
    println!("Language: {}", language); // print the variable
}
```



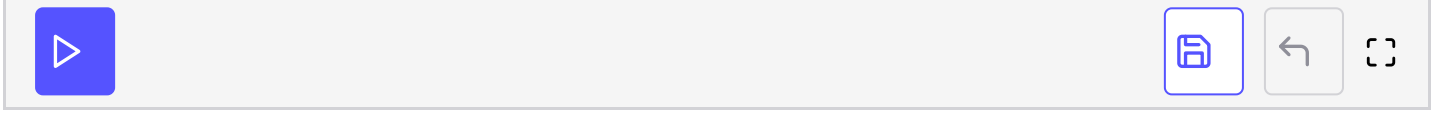
> **Note:** Just like numbers it is not possible to directly print a variable within a `println!()`. You need a placeholder.

## What if You Want to Make a Variable Mutable? #

At the beginning of this lesson, it was mentioned that a variable is immutable until you want to make a change in the variable, then it can be made mutable. To make a variable mutable, write `let` followed by the `mut` keyword and the variable name.




```
fn main() {
    let mut language = "Rust"; // define a mutable variable
    println!("Language: {}", language); // print the variable
    language = "Java"; // update the variable
    println!("Language: {}", language); // print the updated value of variable
}
```



## Assigning Multiple Variables #

It is possible to assign multiple variables in one statement.

name name value value

```
fn main() {
    let (course,category) =("Rust","beginner"); // assign multiple values
    println!("This is a {} course in {}.", category, course); // print the value
}
```

> **Note:** If a variable is kept **un-assigned or unused, you'll get a warning**. To remove such a warning write the expression `#[allow(unused_variables, unused_mut)]` at the start of the program code. However, it's not a good practice to keep unassigned/unused variables.

## Quiz #

Test your understanding of variables in Rust!

Quick Quiz on Variables!

1 Which one is not a property of a default variable?

2 Which of the following code snippets help to make a mutable variable?

Retake Quiz

Now that you have learned about variables, let’s learn about the scope and shadowing of the variable in the next lesson.