

Assignment and Compound Assignment Operators

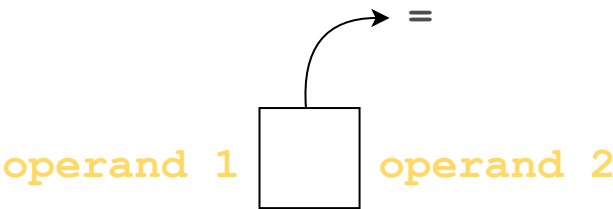
This lesson teaches assignment and compound assignment operators in Rust.

We'll cover the following

- Assignment Operator
 - Type
- Compound Assignment Operator
 - Types
- Quiz

Assignment Operator

The assignment operator is used to save a value in the variable.



Type

Rust has only one assignment operator, `=`. The following table defines the function of the operator.

Operator	Operation	Explanation	Example
<code>operand 1 = operand 2</code>	assign a value	Assign a value of operand 2 to operand 1	<code>a = 1</code> <code>b = a</code>





Assignment operator

The following example demonstrates the use of some of the assignment operator in a program:

```
fn main() {
```

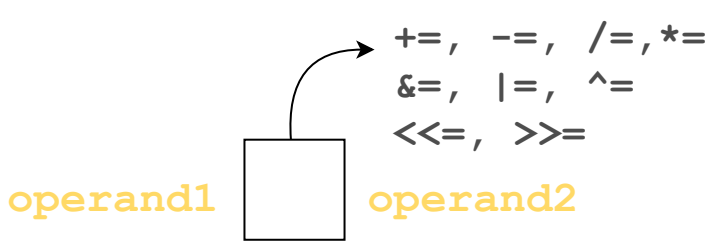
```
let a = 2;
let b = a;
println!("b = a");

println!("Value of a:{}", a);
println!("Value of b:{}", b);
}
```

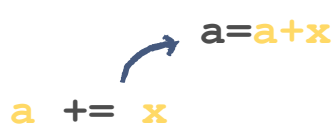


Compound Assignment Operator

The compound assignment operator is used to perform an operation and then assign that value to the operand.



Example



Types

The following table summarizes the types of compound assignment operators:

Operator	Operation	Explanation
operand 1 += operand 2	Add a value and assign	Add left hand-side to the right hand-side and then save the updated value in left hand-side operand
operand 1 -= operand 2	Subtract a value and assign	Subtract right hand-side from the left hand-side and then save the updated value in left hand-side operand
operand 1 /= operand 2	Divide a value and assign	Divide left hand-side with the right hand-side and then save the updated value in left hand-side operand
operand 1 *= operand 2	Multiply a value and assign	Multiply left hand-side with the right hand-side operand and then save the updated value in left hand-side operand
operand 1 %= operand 2	Modulus and assign	Take modulus of left hand-side with the right hand-side operand and then save the updated value in the left hand-side operand
operand 1 &= operand 2	Bitwise AND and assign	Bitwise AND of the left hand-side operand with right hand-side and then save the updated value in the left hand-side operand
operand 1 = operand 2	Bitwise OR and assign	Bitwise OR of the left hand-side operand with right hand-side and then save the updated value in the left hand-side operand
operand 1 ^= operand 2	Bitwise NOT and assign	Bitwise NOT of the left hand-side operand with right hand-side operand and then save the updated value in the left hand-side operand
<<= operand 1	Shift left and assign	Left shift the operand x times and then save the updated value in the operand
>>= operand 1	Shift right and assign	Right shift the operand x times and then save the updated value in the operand

Compound Assignment Operators

The following example demonstrates the use of some of these operators in a program:

```
fn main() {
    //define a mutable variable
```

```
//define a mutable variable
let mut a = 2;
println!("a:{}", a);
a += 1;
println!("a+=1:{}", a);
println!("a:{}", a);
a -= 1;
println!("a-=1:{}", a);
println!("a:{}", a);
a /= 1;
println!("a/=1:{}", a);
println!("a:{}", a);
a *= 3;
println!("a/=3:{}", a);
}
```



Quiz

Test your understanding of the assignment and compound assignment operators in Rust.

Quick Quiz on Assignment Operators!

Q

What is the output of the following code?

```
fn main() {
    let mut a = 2;
    let mut b = 3;

    a += a;
    b -= b;
    a *= 1;
    b *= 3;
    a -= 1;
    println!("a: {}", a);
    println!("b: {}", b);
}
```

[Retake Quiz](#)

The next lesson will cover type casting operator in Rust.