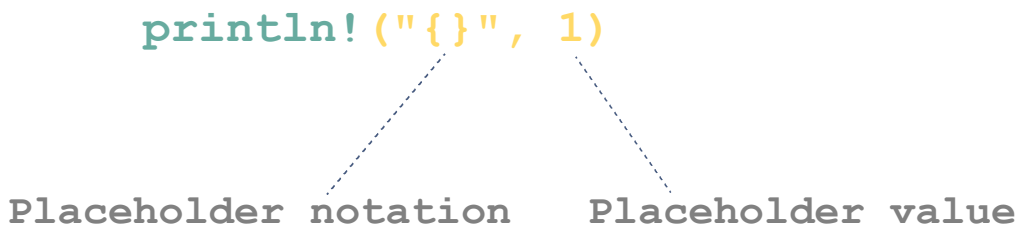
The Basic Formatting

This lesson will teach you how to format a statement in Rust in different ways.

We'll cover the following

In the previous lesson, we saw how we could print a string to the console. But what if we want to print strings in a particular format? Rust can help by using placeholders.



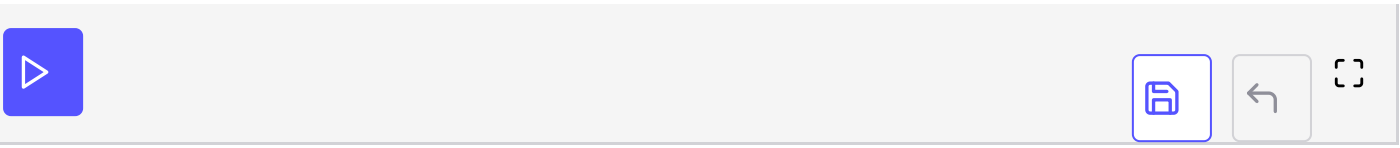
> **Note:** In Rust, we cannot directly print numbers or variables within the `println!()` macro, unlike other languages. We need a placeholder `{}`.

Single Placeholder

A single placeholder is used when it is required to print a single value.

```
fn main() {
    println!("Number: {}", 1);
```

```
}
```

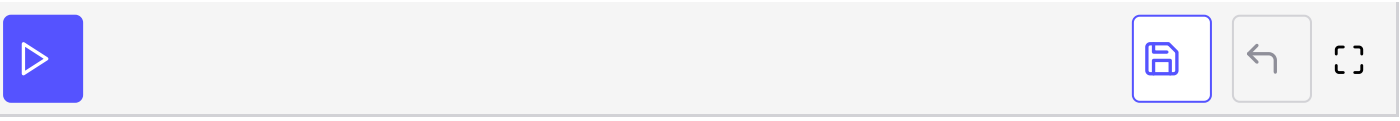


Multiple Placeholders

We can use multiple placeholders within the `println!()` macro.

> The number of placeholders should be equal to the number of values to be printed.

```
fn main() {
    println!("{} is a {} company", "Educative", "Software");
}
```



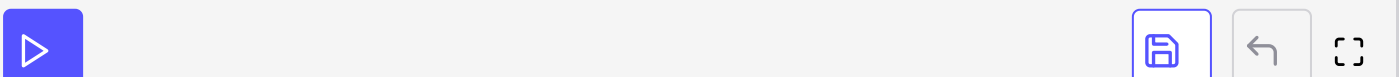
Positional Arguments

Positional arguments specify the positions of the values in a sentence.

Each value is assigned a number based on the order of occurrence. The first value is assigned 0, the next is assigned 1, and so on and so forth. The placeholder takes an integer positive number (greater than or equal to 0) which indicates the value to be inserted in the placeholder is to be picked from the list of values in a given order.

> **Why positional arguments?**
>
> - The list of values can be put in any order.
> - A value can be typed out once and used in the formatted output several times.

```
fn main() {
    println!("Enhance your coding skills from {0} courses.  {0} courses are very {1}", "Educative"
}
```



Named Arguments

A placeholder can take a named argument and assign it a value explicitly. It does this by specifying a name within the placeholder and assign that name the value to be inserted in the string.

```
fn main() {
    println!("{company} provides {kind} courses", company = "Educative", kind = "interactive");
}
```

Placeholder Traits

If we want to convert the value to binary, hexadecimal, or octal write:

```
{:b},{:x},{:o}
```

In the placeholder for binary, hexadecimal, or octal respectively and in the value specify the number.

> You can use all three of them or one of them in a single expression.

```
fn main() {
    println!("Number : 10 \nBinary:{:b} Hexadecimal:{:x} Octal:{:o}", 10, 10, 10);
}
```

Basic Math

We can perform basic math and the placeholder gets replaced with the result.

```
fn main() {
    println!("{} + {} = {}",10, 10, 10 + 10);
}
```

Placeholder for a Debug Trait

It is possible to display multiple values using a single placeholder with the help of the debug trait (a colon followed by a question mark `{:?}`).

This prevents having to write placeholders for each value.

> You can use a debug trait and write as many values as desired within the parentheses.

```
fn main() {
    println!("{:?}", ("This is a Rust Course", 101));
}
```

Quiz

Test your understanding of basic formatting in a Rust program!

Quick Quiz on Basic Formatting!

1

What is the output of the following code?

```
fn main() {
    println!("Enhance your coding skills from {1} courses.  {0} c
ourses are very {1}", "Educative", "interactive");
}
```

2

What is the output of the following code?

```
fn main() {
    println!("{}{}", 2, 1);
}
```

Retake Quiz

Now that you have learned the basics let’s learn about different printing styles in the next lesson.