Numeric Types: Integers and Floats

This lesson will teach you about numeric data types in Rust, i.e., integers and floats.

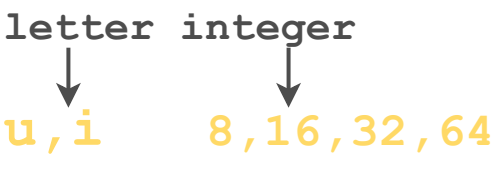
We'll cover the following

Integers

Variables of Integer data type hold whole number values. There are two subtypes of integer data type in Rust, based on the number of bits occupied by a variable in memory.

Fixed Size Types

The fixed integer types have a specific number of bits in their notation. This notation is a combination of a letter and a number. The former denotes the category of the integer, whether it is, unsigned or signed, and the latter denotes the size of an integer, i.e., 8, 16, 32, 64.

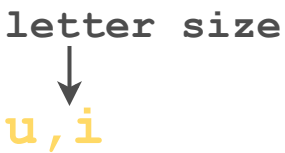


Below is the list of fixed length integer types:

- `i8`: The **8-bit signed** integer type.
- `i16`: The **16-bit signed** integer type.
- `i32`: The **32-bit signed** integer type.
- `i64`: The **64-bit signed** integer type.
- `u8`: The **8-bit unsigned** integer type.
- `u16`: The **16-bit unsigned** integer type.
- `u32`: The **32-bit unsigned** integer type.
- `u64`: The **64-bit unsigned** integer type.

Variable Size Types

The integer type in which the particular size depends on the underlying machine architecture.



- `isize`: The **pointer-sized signed** integer type.
- `usize`: The **pointer-sized unsigned** integer type.

> 💡 **Why are there so many types of integers and how do you pick a data type?**
>
> The choice depends on what values a variable is expected to hold. So, a programmer should pick a data type that is not so small that the data is lost. Nor should they pick a data type that is so big that it wastes memory.

Example

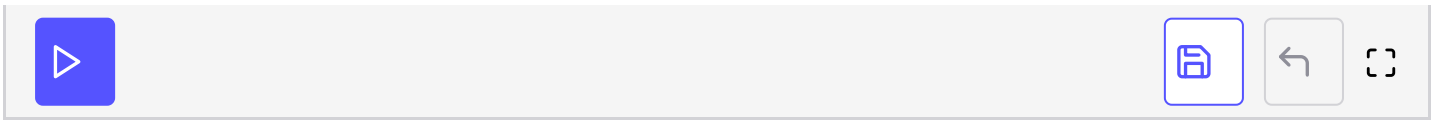
The code below defines an integer type both explicitly and implicitly:

Explicit Definition

The following code explicitly defines the integer variables using the integer type fixed or variable):

```
fn main() {
    //explicitly define an integer
    let a:i32 = 24;
    let b:u64 = 23;
    let c:usize = 26;
    let d:isize = 29;
    //print the values
    println!("a: {}", a);
    println!("b: {}", b);
    println!("c: {}", c);
    println!("d: {}", d);

}
```

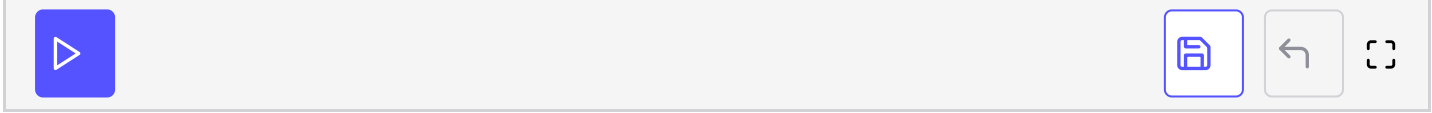


Implicit Definition

The following code implicitly defines the integer type of the variable by assigning an integer value to the variable.

```
fn main() {
    //explicitly define an integer
    let a = 21;
    let b = 1;
    let c = 54;
    let d = 343434;
    //print the variable
    println!("a: {}", a);
    println!("b: {}", b);
    println!("c: {}", c);
    println!("d: {}", d);

}
```

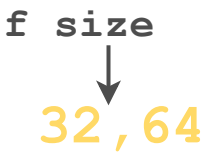


Floating Point

Floating-point numbers refer to numbers with a fractional part.

The representation of floating-point numbers in a computer's memory is such that the precision with which a number is stored in memory depends on the number of bits used for storing the variable.

In this respect, there are two subtypes: single-precision `f32` and double-precision `f64` floating-point, with the latter having more bits to store the number.



- `f32`: The **32-bit floating point** type.
- `f64`: The **64-bit floating point** type.

Example

The code below defines a floating-point number both explicitly and implicitly:

Explicit Definition

The following code explicitly defines the float variable using the float type (`f32` or `f64`):

```
fn main() {
    //explicitly define a float type
    let f1:f32 = 32.9;
    let f2:f64 = 6789.89;
    println!("f1: {}", f1);
    println!("f2: {}", f2);
}
```

Implicit Definition

The following code implicitly defines the float type of the variable by assigning a floating-point value to the variable:

```
fn main() {
    //implicitly define a float type
    let pi = 3.14;
    let e = 2.17828;
    println!("pi: {}", pi);
    println!("e: {}", e);
}
```

Quiz

Test your understanding of Numeric Types in Rust!

Quick Quiz on Numeric Types!

1

What is the data type of the variable below?

```
let  a = 123;
```

2

Which of the following is an incorrect notation to declare a float type variable?

Retake Quiz

Now that you have learned about integers and float data types, let’s learn about the boolean data type in the next lesson.