Loop Labels

This lesson discusses loop labels in Rust.

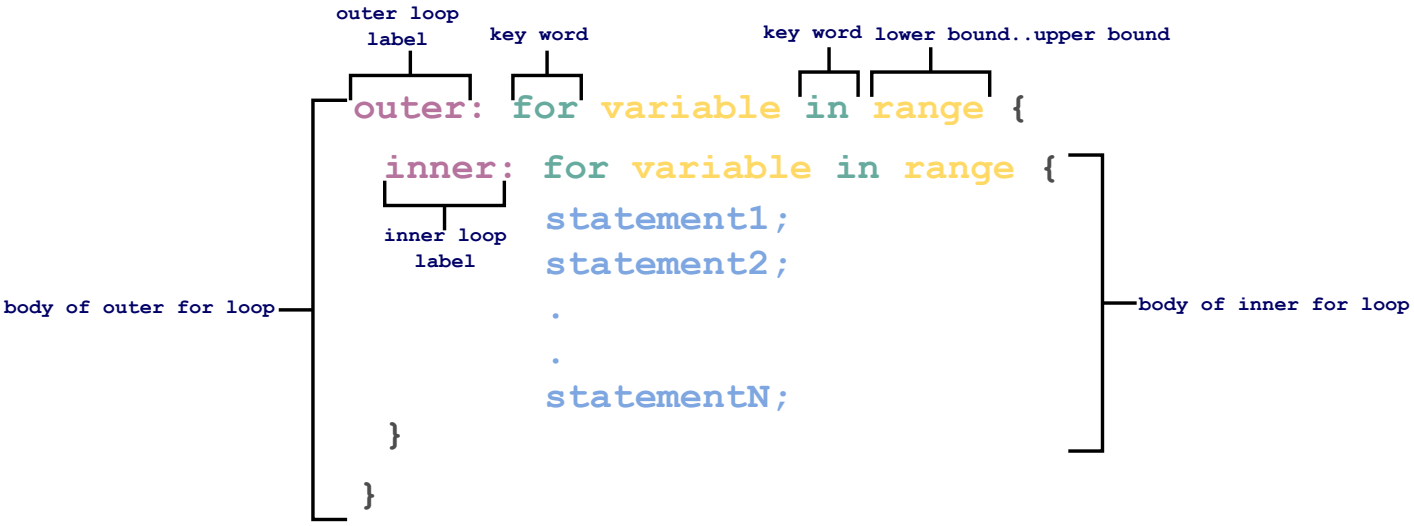
We'll cover the following

What Is a Loop Label?

A loop label assigns an identifier to a loop.

Syntax

Write a label and colon before the loop.

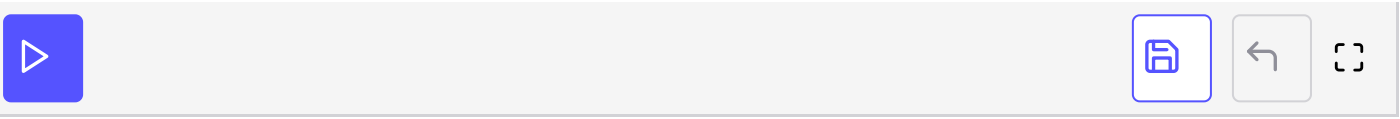


Example

The code below prints the multiplication table of 1 to 5 except 3. See how specifying a loop label helps you to skip the table of 3.

```
fn main() {
 'outer:for i in 1..5 { //outer loop
```

```
    println!("Muliplication Table : {}", i);
   'inner:for j in 1..5 { // inner loop
        if i == 3 { continue 'outer; } // Continues the loop over `i`.
        if j == 2 { continue 'inner; } // Continues the loop over `j`.
        println!("{} * {} = {}", i, j, i * j);
    }
  }
}
```



Explanation

Outer `for` Loop

- A `for` loop is defined on **line 2**.
- The loop has a label `outer` . It takes `i` as an iterator that iterates over values from 1 to 4.

Inner `for` Loop

- A `for` loop is defined on **line 3**
- The loop has a label `inner`. It takes `j` as an iterator that iterates over values from 1 to 5.
- For each `i` the inner loop iterate j times and prints the product `i * j`.
- When the outer loop increments `i` to 3 and the inner loop starts from `j = 1`, the condition `i == 3` is found to be true and the `continue 'outer` statement causes execution to be transferred to the next iteration of the outer loop on **line 2**. The variable `i` is incremented to 4 and the execution continues.
- When the value of `j` increments to 2, then the 2nd iteration of the inner loop gets skipped and `continue 'inner` causes the execution to be transferred to the next iteration of the inner loop on **line 4**. The variable `j` is incremented to 3 and the execution continues.

i	j	Output
1	1 2 3 4	1*1=1 1*3=3 1*4=4
2	1 2 3 4	2*1=2 2*3=6 2*4=8
3		
4	1 2 3 4	4*1=4 4*3=12 4*4=16

```
continue at j=2 and i=3
```

Now that you have learned about loops, check your knowledge in the next lesson through a challenge.