Please insert this new section after the “Repeating Code with loop” section ends and before the “Conditional Loops with while” section starts, on page 53.

Returning Values from Loops

One of the uses of a loop is to retry an operation you know might fail, such as checking whether a thread has completed its job. However, you might need to pass the result of that operation to the rest of your code. To do this, you can add the value you want returned after the break expression you use to stop the loop; that value will be returned out of the loop so you can use it, as shown here:

fn main() {

let mut counter = 0;

let result = loop {

counter += 1;

if counter == 10 {

break counter \* 2;

}

};

println!("The result is {}", result);

}

Before the loop, we declare a variable named counter and initialize it to 0. Then we declare a variable named result to hold the value returned from the loop. On every iteration of the loop, we add 1 to the counter variable, and then check whether the counter is equal to 10. When it is, we use the break keyword with the value counter \* 2. After the loop, we use a semicolon to end the statement that assigns the value to result. Finally, we print the value in result, which in this case is 20.