Scope and Shadowing

This lesson teaches you about the scope of a variable.

We'll cover the following

Scope of a variable

The scope of a variable refers to the visibility of a variable, or , which parts of a program can access that variable.

It all depends on where this variable is being declared. If it is declared inside any curly braces `{}`, i.e., a block of code, its scope is restricted within the braces, otherwise the scope is global.

Types of Variables

There are two types of variables in terms of scope.

Local Variable

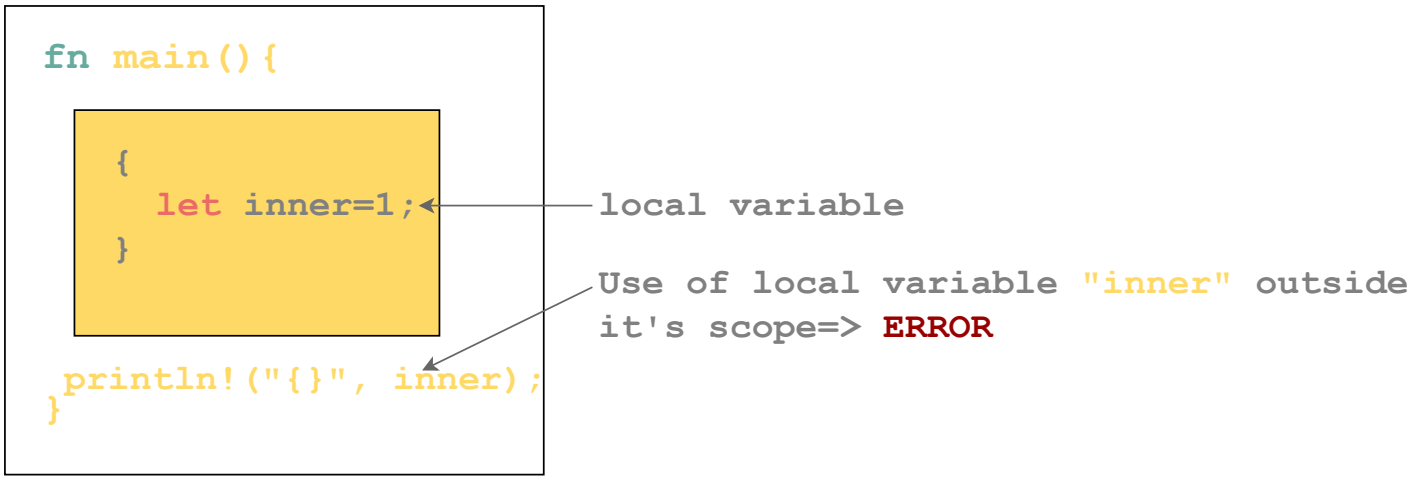
A variable that is within a block of code, `{ }`, that cannot be accessed outside that block is a local variable. After the closing curly brace, `}` , the variable is freed and memory for the variable is deallocated.

Global Variable

The variables that are declared outside any block of code and can be accessed within any subsequent blocks are known as global variables.

The variables declared using the `const` keyword can be declared in local as well as

global scope. You'll study constant variables in the next chapter.



> **Note:** The following code gives an error, ✕, since the variable created inside the inner block of code has been accessed outside its scope.

```
fn main() {
  let outer_variable = 112;
  { // start of code block
        let inner_variable = 213;
        println!("block variable inner: {}", inner_variable);
        println!("block variable outer: {}", outer_variable);
  } // end of code block
    println!("inner variable: {}", inner_variable); // use of inner_variable outside scope
}
```

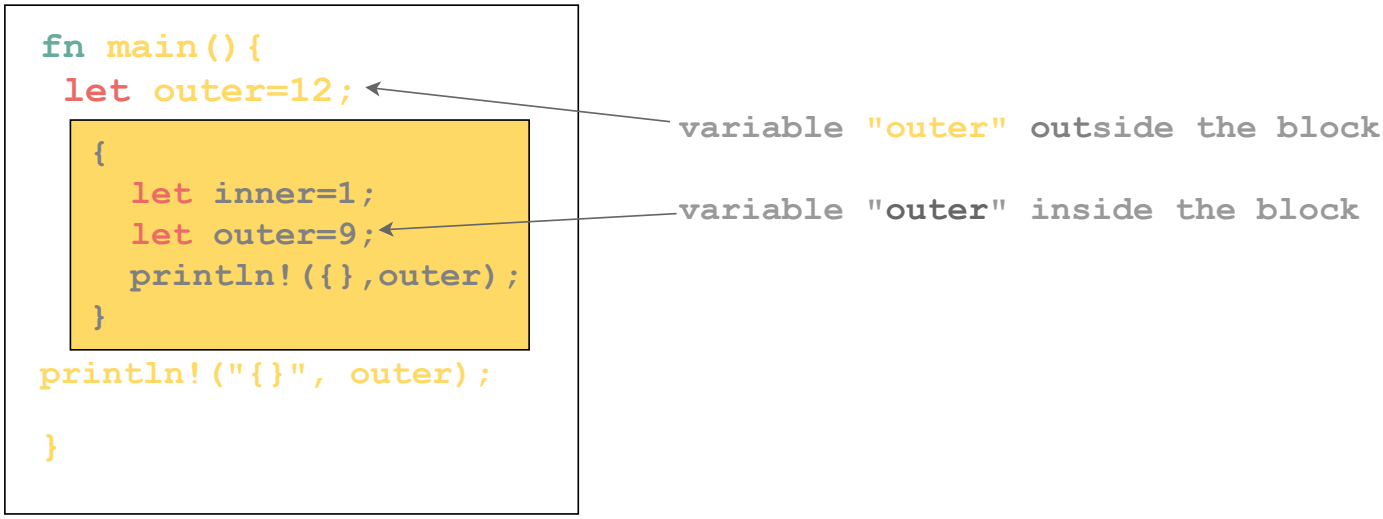
To fix this error, the variable declaration can be moved outside the inner block of code. That way, the scope of the variable spans the entire `main()` function. This is shown below:

```
fn main() {
  let outer_variable = 112;
  let inner_variable = 213;
  { // start of code block
        println!("block variable inner: {}", inner_variable);
        println!("block variable outer: {}", outer_variable);
  } // end of code block
    println!("inner variable: {}", inner_variable);
  }
```

Shadowing

Shadowing

Variable shadowing is a technique in which a variable declared within a certain scope has the same name as a variable declared in an outer scope. This is also known as masking. This outer variable is said to be shadowed by the inner variable, while the inner variable is said to mask the outer variable.



The following code explains the concept.

```
fn main() {
  let outer_variable = 112;
  { // start of code block
        let inner_variable = 213;
        println!("block variable: {}", inner_variable);
        let outer_variable = 117;
        println!("block variable outer: {}", outer_variable);
  } // end of code block
    println!("outer variable: {}", outer_variable);
  }
```

Quiz

Test your understanding of variables!

Quick Quiz on Scope and Shadowing !

1 A variable cannot be accessed outside it’s scope.

2

What is the output of the following code?

```
fn main(){
   let a=1;
  {
         let b=1;
  }
  println!("The value of b is {}",b);
}
```

3

A variable that takes the same name in the inner block as that of variable in the outer block. This concept is called

Retake Quiz

Now that you have learned about the variables and scope of variables, let’s check your knowledge in the upcoming challenge.