Constant Variables

This lesson discusses constant variables and how they differ from let variables.

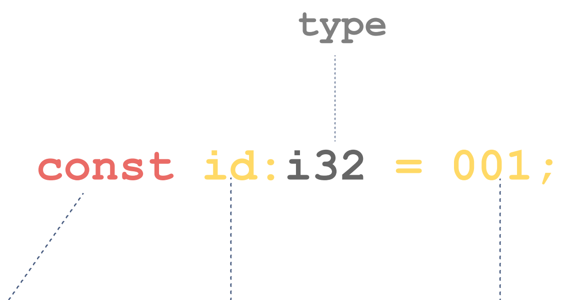
We'll cover the following

What Are Constant Variables?

Constant variables are ones that are declared constant throughout the program scope, meaning, their value cannot be modified. They can be defined in global and local scope.

Syntax

They are declared using the `const` keyword followed by the name of the variable, colon (`:`), and then the data type of the variable.



> **Naming Convention:** By convention, you write a constant variable name in a SCREAMING_SNAKE_CASE, i.e.,
>
> - All letters should be UPPER case.
> - All words should be separated using an underscore (_).

Example

The following example defines two `const` variables:

- `ID_1` in **global scope**
- `ID_2` in **local scope**

```
const ID_1: i32 = 4; // define a global constant variable
fn main() {
    const ID_2: u32 = 3; // define a local constant variable
    println!("ID:{}", ID_1); // print the global constant variable
    println!("ID:{}", ID_2); // print the local constant variable
}
```

Difference Between `const` and `let` Variables

There are many differences between `const` and `let` variables.

Declaration

- Constant variables are declared using the `const` keyword unlike `let` variables.

Scope

- `const` variables are declared in global and local scope unlike `let` variables that are declared only in the local scope.

Mutability

- `const` variable cannot be mutable unlike `let` which can be made mutable using `mut` keyword.

Data Type

- Unlike `let` variables, it is mandatory to define the data type of `const` variables.

Set Value at Run-time

- The value of `const` variable can only be set before running the program whereas the `let` variable can store the result at runtime.

Shadowing

- Unlike `let` variables, `const` variables cannot be shadowed.

Now that you have an insight into data types and const variables, let’s check your knowledge in the upcoming challenge in the next lesson.