한 번에 끝내는
블록체인 개발 A to Z

Chapter 1

Rust Introduction

Chapter 1

Rust Introduction

Programming a Guessing Game

A Guessing Game

❶ Create New Project with Cargo

```
$ cargo new guessing_game
$ cd guessing_game
```

❷ Write Cargo.toml

```
[package]
name = "guessing_game"
version = "0.1.0"
edition = "2021"

[dependencies]
rand = "0.9.0"
```

A Guessing Game

❸ Write src/main.rs

```
use std::io;
use rand::Rng;

fn main() {
    println!("Guess the number!");

    let secret_number = rand::thread_rng().gen_range(1..101);

    println!("The secret number is: {}", secret_number);
    println!("Please input your guess.");

    let mut guess = String::new();

    io::stdin()
        .read_line(&mut guess)
        .expect("Failed to read line");

    println!("You guessed: {}", guess);
}
```

A Guessing Game

❹ Compare secrete number

```
use rand::Rng;
use std::cmp::Ordering;
use std::io;

fn main() {
    // --snip-

    println!("You guessed: {}", guess);

    match guess.cmp(&secret_number.to_string()) {
        Ordering::Less => println!("Too small!"),
        Ordering::Greater => println!("Too big!"),
        Ordering::Equal => println!("You win!"),
    }
}
```

A Guessing Game

❺ Multiple Guessing with loop

```
// --snip-

    println!("The secret number is: {}", secret_number);

    loop {
        println!("Please input your guess.");

        // --snip-

        match guess.cmp(&secret_number) {
            Ordering::Less => println!("Too small!"),
            Ordering::Greater => println!("Too big!"),
            Ordering::Equal => println!("You win!"),
        }
    }
}
```

A Guessing Game

❻ Quitting After a Correct Guess

```
        // --snip-

        match guess.cmp(&secret_number) {
            Ordering::Less => println!("Too small!"),
            Ordering::Greater => println!("Too big!"),
            Ordering::Equal => {
                println!("You win!");
                break;
            }
        }
    }
}
```

A Guessing Game

❼ Handling invalid input

```
// --snip-

io::stdin()
    .read_line(&mut guess)
    .expect("Failed to read line");

let guess: u32 = match guess.trim().parse() {
    Ok(num) => num,
    Err(_) => continue,
};

println!("You guessed: {}", guess);

// --snip--
```