

kintsugi-stack-rust

kintsugi-stack-rust

1. Getting Started

1.1. Installation

```
sudo apt update
sudo apt install -y curl
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

- Rust Lang: Rust Install <https://doc.rust-lang.org/book/ch01-01-installation.html>
 - Windows
 - Install Linux, Just kidding !!
 - <https://visualstudio.microsoft.com/downloads/>
 - Install VS
 - Install VSC
 - Install Build Tools for Visual Studio
 - then Restart Computer
 - <https://rust-lang.org/tools/install/>
 - Install Rust
 - `rustup toolchain install stable-x86_64-pc-windows-gnu`
 - `rustup default stable-x86_64-pc-windows-gnu`
 - Linux:
`$ curl --proto '=https' --tlsv1.2 https://sh.rustup.rs -sSf | sh`
- Rust Server Dev: Rust Analyzer Install <https://marketplace.visualstudio.com/items?itemName=rust-lang.rust-analyzer>

1.2. Hello, World!

```
// 1_2_hello_world.rs
fn main(){
    println!("Hello, World! ")
}
// rustc main.rs && ./main
```

- rust code file extension `.rs`

```
// 1_2_hello_world.rs
fn main(){
    println!("Hello, World! ")
```

```
}
```

```
// rustc main.rs && ./main
```

- Compile command

```
rustc main.rs
```

- Rust Binary Run command

```
./main
```

1.3. Cargo

```
cargo --version # cargo version check
&& cargo new project_name # create proj.
&& cd project_name
&& cargo build # build executable
&& cargo run # run project
&& cargo check # check for err without any executable
&& cargo help # help

# .
# └── Cargo.lock
# └── Cargo.toml
# └── src
#     └── main.rs

# .
# └── Cargo.lock
# └── Cargo.toml
# └── src
#     └── main.rs
# └── target
#     ├── CACHEDIR.TAG
#     └── debug
#         ├── build
#         └── deps
#             ├── libone_three_hello_cargo-f9884884092cd48a.rmeta
#             ├── one_three_hello_cargo-5885dd703046e3fc
#             ├── one_three_hello_cargo-5885dd703046e3fc.d
#             └── one_three_hello_cargo-f9884884092cd48a.d
#             ├── examples
#             └── incremental
#                 ├── one_three_hello_cargo-10ah7hvrv4gzi
#                 |   ├── s-heho9i1rut-1ysdico-bwbrfy6ptxbomf5iwqz5vt3f0
#                 |   |   ├── dep-graph.bin
#                 |   |   └── query-cache.bin
```

```

#
#   └── work-products.bin
#       └── s-heho9i1rut-1ysdico.lock
#           one_three_hello_cargo-3dgwin0zbxstr
#               └── s-heho98ij63-039vdoh-143bk3qfw5zxnyx5otl9s0tja
#                   ├── 00ylhni9avwle6wyqpyzm6par.o
#                   ├── 19k6gm7hj98zo0jv2b5mu1std.o
#                   ├── 1jqdzhkz0e02p777bbobcmna2j.o
#                   ├── 5fgvfmdk1vtvncsc4ze5a0wi9.o
#                   ├── 8z1o97dthkm4wl9qy6anckmmy.o
#                   ├── 9fwica1fdmiqw5oux5l4cedjc.o
#                   ├── dep-graph.bin
#                   ├── query-cache.bin
#                   └── work-products.bin
#                       └── s-heho98ij63-039vdoh.lock
#               one_three_hello_cargo
#               one_three_hello_cargo.d
#

```

- Cargo:
 - Rust's Build System
 - - Package manager
 - Builtin When we Install Rust (Painpoint of other prog. lang.)
- Compile command

```
rustc main.rs
```

- Rust Binary Run command

```
./main
```

- Cargo version check

```
cargo --version
```

- Create New Cargo Project

```
cargo new one_three_hello_cargo
```

- File Organisation
 - **Cargo.toml**
 - package config file
 - **.gitignore**
 - default code ver. ignore file

- ignore flags for git ver.
- \src
 - contains actual code
 - main.rs
 - Starter code

```
.  
└── Cargo.lock  
└── Cargo.toml  
└── src  
    └── main.rs
```

- Build command
 - Build
 - ▪ Create **Cargo.lock**
 - contain dependencies
 - ▪ Create \target
 - contain \debug
 - contain our actual executable
 - other supporting stuff

```
cargo build
```

- run command

```
cargo run
```

- help command
 - to view all commands

```
cargo help
```

- check command
 - check your prog. for err.
 - without producing any executable
 - faster than running the prog.

```
cargo check
```

```
bali-king@war-machine:~/BaliGit/kintsugi-stack-rust/one_three_hello_cargo$ cargo run
   Compiling one_three_hello_cargo v0.1.0 (/home/bali-king/BaliGit/kintsugi-stack-rust/one_three_hello_cargo)
     Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.10s
       Running `target/debug/one_three_hello_cargo`
Hello, world!
bali-king@war-machine:~/BaliGit/kintsugi-stack-rust/one_three_hello_cargo$ cargo build
   Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.00s
bali-king@war-machine:~/BaliGit/kintsugi-stack-rust/one_three_hello_cargo$ cargo check
   Checking one_three_hello_cargo v0.1.0 (/home/bali-king/BaliGit/kintsugi-stack-rust/one_three_hello_cargo)
     Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.02s
bali-king@war-machine:~/BaliGit/kintsugi-stack-rust/one_three_hello_cargo$
```

```
.
├── Cargo.lock
├── Cargo.toml
└── src
    └── main.rs
└── target
    ├── CACHEDIR.TAG
    └── debug
        ├── build
        └── deps
            ├── libone_three_hello_cargo-f9884884092cd48a.rmeta
            ├── one_three_hello_cargo-5885dd703046e3fc
            ├── one_three_hello_cargo-5885dd703046e3fc.d
            └── one_three_hello_cargo-f9884884092cd48a.d
        ├── examples
        └── incremental
            ├── one_three_hello_cargo-10ah7hvrv4gzi
            │   ├── s-heho9i1rut-1ysdico-bwbrfy6ptxbomf5iwqz5vt3f0
            │   │   ├── dep-graph.bin
            │   │   ├── query-cache.bin
            │   │   └── work-products.bin
            │   └── s-heho9i1rut-1ysdico.lock
            └── one_three_hello_cargo-3dgwin0zbxstr
                ├── s-heho98ij63-039vdoh-143bk3qfw5zxnyx5otl9s0tja
                │   ├── 00ylhni9avwle6wyqpyzm6par.o
                │   ├── 19k6gm7hj98zo0jv2b5mu1std.o
                │   ├── 1jqdhkz0e02p777bbobcmna2j.o
                │   ├── 5fgvfmdk1tvncsc4ze5a0wi9.o
                │   ├── 8z1o97dthkm4wl9qy6anckmmy.o
                │   ├── 9fwica1fdmiqw5oux5l4cedjc.o
                │   ├── dep-graph.bin
                │   ├── query-cache.bin
                │   └── work-products.bin
                └── s-heho98ij63-039vdoh.lock
```

```
└── one_three_hello_cargo
    └── one_three_hello_cargo.d
```

2. Programming a Guessing Game

```
use std::cmp::Ordering, io; // io lib in scope

// Random Library
// to add deps "rand" package => add `deps = "version"` in `Cargo.toml` =>
cargo build
// [dependencies]
// rand = "0.5.5"
use rand::{Rand, Rng};

fn main() {
    // intro lines print
    println!("Guess the Number !!!"); // like python/c
    println!("Input Your Guess:");

    // variable to store stuff
    // String, A type is Rust Standard library, utf-8, growable string
    // new() is associative func. static method, create empty string
    // Variables in Rust are DEFAULT IMMUTABLE, to make them mutable, use
mut keyword
    let mut guess = String::new(); // like java

    // io lib in scope
    // use std::io; // io lib in scope
    // .read_line method to read line
    // Result cases to 1. Ok() & 2. Err()
    io::stdin() // like java
        .read_line(&mut guess)
        .expect("Failed to Read Line"); // iff err comes, .expect() crash
program, and display message

    // .trim() remove whitespaces
    // .parse() helps to parse
    let guess: u32 = guess.trim().parse()
    .expect("Failed to Read Line");// error handling strict by language

    println!("You Guessed: {}", guess); // like c
    // Guess the Number !!!
    // Input Your Guess:
    // 12
    // You Guessed: 12

    // Now Random Check is Left

    // // Random Library
    // // to add deps "rand" package => add `deps = "version"` in
```

```

`Cargo.toml` => cargo build
    // // [dependencies]
    // // rand = "0.5.5"
    // use rand::{Rand, Rng};
    let secret_nos = rand::thread_rng().gen_range(1, 101); // lower limit is
inclusive, upper limit is exclusive
    println!("Actual Number: {}", secret_nos);

    // cmp::Ordering library
    match guess.cmp(&secret_nos){
        Ordering::Equal => print!("YOU WIN !!!"),
        Ordering::Less => print!("TOO SMALL !!!"),
        Ordering::Greater => print!("TOO BIG !!!")
    }

}

// Guess the Number !!!
// Input Your Guess:
// 2
// You Guessed: 2
// Actual Number: 2
// YOU WIN !!!

```

- process
- initialize project

```

cargo new two_guessing_game
&& cd two_guessing_game

```

- `src/main.rs` code the basic logic

```

use std::io; // io lib in scope

fn main() {
    // intro lines print
    println!("Guess the Number !!!"); // like python/c
    println!("Input Your Guess:");

    // variable to store stuff
    // String, A type is Rust Standard library, utf-8, growable string
    // new() is associative func. static method, create empty string
    // Variables in Rust are DEFAULT IMMUTABLE, to make them mutable, use
mut keyword
    let mut guess = String::new(); // like java

    // io lib in scope

```

```

// .read_line method to read line
// Result cases to 1. Ok() & 2. Err()
io::stdin() // like java
    .read_line(&mut guess)
    .expect("Failed to Read Line"); // iff err comes, .expect() crash
program, and display message

    println!("You Guessed {} !!!", guess); // like c

    // Guess the Number !!!
    // Input Your Guess:
    // 12
    // You Guessed 12
    // !!!
    // Now Random Check is Left

}

```

- use **cargo run** or Run Button in Vsc at the main line(comes with extension)
- Now Random Check is Left
- to add deps "rand" package
 - add **deps = "version"** in **Cargo.toml**

```

[package]
name = "two_guessing_game"
version = "0.1.0"
edition = "2024"

[dependencies]
rand = "0.5.5"

```

- then

```
cargo build
```

```

bali-king@war-machine:~/BaliGit/kintsugi-stack-rust/two_guessing_game/src$ cargo build
Compiling rand_core v0.4.2
Compiling libc v0.2.178
Compiling rand_core v0.3.1
Compiling rand v0.5.6 # Gotcha
Compiling two_guessing_game v0.1.0 (/home/bali-king/BaliGit/kintsugi-
stack-rust/two_guessing_game)

```

- then random number and check logic :

```
// // Random Library
// // to add deps "rand" package => add `deps = "version"` in
`Cargo.toml` => cargo build
// [dependencies]
// rand = "0.5.5"
// use rand::{Rand, Rng};
let secret_nos = rand::thread_rng().gen_range(1, 101); // lower limit is
inclusive, upper limit is exclusive
println!("Actual Number: {}", secret_nos);

// cmp::Ordering library
match guess.cmp(&secret_nos){
    Ordering::Equal => print!("YOU WIN !!!"),
    Ordering::Less => print!("TOO SMALL !!!"),
    Ordering::Greater => print!("TOO BIG !!!")
}
```

- thus, the output is :

```
Guess the Number !!!
Input Your Guess:
2
You Guessed: 2
Actual Number: 2
YOU WIN !!!
```

- now guess logic is done
- to make game more interesting we can have game on loop to guess till user guess the number correctly
- put the guess input and match logic code in this `loop{ ... }`

```
// to make game more interesting we can have game on loop to guess till
user guess the number correctly
loop {

    println!("Input Your Guess:");

    // variable to store stuff
    // String, A type is Rust Standard library, utf-8, growable string
    // new() is associative func. static method, create empty string
    // Variables in Rust are DEFAULT IMMUTABLE, to make them mutable, use
mut keyword
    let mut guess = String::new(); // like java

    // io lib in scope
    // use std::io; // io lib in scope
```

```

// .read_line method to read line
// Result cases to 1. Ok() & 2. Err()
io::stdin() // like java
    .read_line(&mut guess)
    .expect("Failed to Read Line"); // iff err comes, .expect() crash
program, and display message

// Shadowing, we declare one variable (let mut guess = String::new();)
and then redeclare to convert the datatype but to preserve the value
// .trim() remove whitespaces
// .parse() helps to parse
// let guess: u32 = guess.trim().parse().expect("Failed to Read
Line");// error handling strict by language // old way
let guess: u32 = match guess.trim().parse(){
    Ok(num)=> num,
    Err(_)=> continue // `__` means catch all
    // to whatever any wrong input comes, continue the loop
}; // new way

println!("You Guessed: {}",guess); // like c
// Guess the Number !!!
// Input Your Guess:
// 12
// You Guessed: 12

// cmp::Ordering library
match guess.cmp(&secret_nos){
    Ordering::Equal => {print!("YOU WIN !!!");break}, // to terminate
after win is to break the loop // New way
    // Ordering::Equal => print!("YOU WIN !!!"), // Old way
    Ordering::Less => print!("TOO SMALL !!!"),
    Ordering::Greater => print!("TOO BIG !!!")
}

// basic working
// Guess the Number !!!
// Input Your Guess:
// 2
// You Guessed: 2
// Actual Number: 2
// YOU WIN !!!

}

```

```

Guess the Number !!!
Input Your Guess:
20
You Guessed: 20
TOO SMALL !!!Input Your Guess:
30
You Guessed: 30

```

```

TOO SMALL !!!Input Your Guess:
40
You Guessed: 40
TOO SMALL !!!Input Your Guess:
60
You Guessed: 60
TOO SMALL !!!Input Your Guess:
80
You Guessed: 80
TOO SMALL !!!Input Your Guess:
90
You Guessed: 90
TOO BIG !!!Input Your Guess:
81
You Guessed: 81
TOO SMALL !!!Input Your Guess:
89
You Guessed: 89
TOO BIG !!!Input Your Guess:
85
You Guessed: 85
YOU WIN !!!Input Your Guess:
85
You Guessed: 85
YOU WIN !!!Input Your Guess:
YOU WIN !!!Input Your Guess:
exit

```

```

thread 'main' (49618) panicked at src/main.rs:46:6:
Failed to Read Line: ParseIntError { kind: InvalidDigit }
note: run with `RUST_BACKTRACE=1` environment variable to display a
backtrace

```

- now, to terminate after win is to break the loop:
 - convert `Ordering::Equal => print!("YOU WIN !!!")` to `Ordering::Equal => {print!("YOU WIN !!!");break;}`

```

match guess.cmp(&secret_nos){
    Ordering::Equal => {print!("YOU WIN !!!");break;}, // to terminate
after win is to break the loop // New way
    // Ordering::Equal => print!("YOU WIN !!!"), // Old way
    Ordering::Less => print!("TOO SMALL !!!"),
    Ordering::Greater => print!("TOO BIG !!!")
}

```

- working

```

Guess the Number !!!
Input Your Guess:

```

```

50
You Guessed: 50
TOO BIG !!!Input Your Guess:
25
You Guessed: 25
TOO SMALL !!!Input Your Guess:
40
You Guessed: 40
TOO SMALL !!!Input Your Guess:
45
You Guessed: 45
YOU WIN !!!

```

- futher improvement: at wrong input ,the program **panick**'s
 - put match case handling at parsing of input string
 - convert `let guess: u32 = guess.trim().parse().expect("Failed to Read Line");// error handling strict by language to this below`

```

// .trim() remove whitespaces
// .parse() helps to parse
// let guess: u32 = guess.trim().parse().expect("Failed to Read
Line");// error handling strict by language // old way
let guess: u32 = match guess.trim().parse(){
    Ok(num)=> num,
    Err(_)=> continue // `_` means catch all
    // to whatever any wrong input comes, continue the loop
};// new way

```

- Key Concept **Shadowing**: we declare one variable `let mut guess = String::new();` and then re-declare to convert the datatype but to preserve the value

```

let mut guess = String::new();

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

```

- working, handling wrong inputs gracefully

```

Guess the Number !!!
Input Your Guess:
MOMO and Chutney
Input Your Guess:
50
You Guessed: 50

```

```
TOO SMALL !!!Input Your Guess:  
75  
You Guessed: 75  
TOO BIG !!!Input Your Guess:  
65  
You Guessed: 65  
TOO BIG !!!Input Your Guess:  
55  
You Guessed: 55  
TOO SMALL !!!Input Your Guess:  
53  
You Guessed: 53  
TOO SMALL !!!Input Your Guess:  
51  
You Guessed: 51  
TOO SMALL !!!Input Your Guess:  
52  
You Guessed: 52  
TOO SMALL !!!Input Your Guess:  
54  
You Guessed: 54  
TOO SMALL !!!Input Your Guess:  
58  
You Guessed: 58  
YOU WIN !!!
```