ackee | blockchain security

School of Solana

LECTURE 2

Rust Introduction

Ondrej Rehacek | Josef Gattermayer, Ph.D.

About this lecture

About this lecture

- Introduction to Rust
- No Solana concepts today
- Hands on examples

Rust

Rust

- Modern systems programming language
  - Safety
  - Speed
  - Concurrency
- Statically and strongly typed
- Graydon Hoare personal project (2006) / @graydon_pub
  - Later sponsored and acquired by Mozilla
- Support WASM

Rust

- Very first release in 2015
- Its own foundation in 2021
- Most loved languages since 2016 ❤️
- Companies using Rust
  - Solana
  - CloudFlare
  - 1Password

Hello world - again

ackee | blockchain security

```
fn main() {
    println!("Hello, world!");
}
```

Rust - Data Types

```rust
fn main() {
    let string_type = "Solana";        // string type
    let float_type = 4.5;              // float type
    let boolean_type = true;           // boolean type
    let char_type = '♥';               // unicode character
type
    println!("Winter School of {}",string_type);
    println!("Last lesson rating on 5 is:{}",float_type);
    println!("We like Solana :{}",boolean_type);
    println!("Solana is in our :{}",char_type);
}
```

```rust
fn main() {
    let result = 10;     // i32 by default
    let age:u32 = 20;
    let sum:i32 = 5-15;
    let mark:isize = 10;
    let count:usize = 30;
    println!("result value is {}",result);
    println!("sum is {} and age is {}",sum,age);
    println!("mark is {} and count is {}",mark,count);
}
```

Rust - String

String literal

```rust
fn main() {
    let course:&str="Winter School of Solana";
    let lecture:&str = "Rust";
    println!("I do attend {} lecture on {}",course,lecture);
}
```

String

```rust
fn main() {
  let empty_string = String::new();
  println!("length is {}",empty_string.len());

  let content_string = String::from("AckeeBlockchain");
  println!("length is {}",content_string.len());
}
```

Rust - Variables

WSoS

```
let variable_name = value;                //no type
let variable_name:dataType = value;       //type specified
```

Variables

- Letters, digits, and the underscore characters
- Must begin with either a letter or an underscore
- Upper and lowercase letters are distinct
- By default, variables are immutable

Rust - Functions

function

```
fn function_name(param1,param2..paramN) {
    // function body
}
```

function

```rust
//Defining a function
fn fn_hello(){
    println!("hello from function fn_hello ");
}
fn main(){
    //calling a function
    fn_hello();
}
```

function

```
// Syntax1
fn function_name() -> return_type {
    //statements
    return value;
}

//Syntax2
fn function_name() -> return_type {
    value //no semicolon means this value is returned
}
```

Rust - Flow control + Loops

Condition

```rust
fn main() {
    let number = 3;

    if number < 5 {
        println!("condition was true");
    } else {
        println!("condition was false");
    }
}
```

If let Condition

```rust
fn main() {
    let condition = true;
    let number = if condition { 5 } else { 6 };

    println!("The value of number is: {}", number);
}
```

Rust - Memory Management

ownership

```
fn main(){
    let v = vec![1,2,3];
    let v2 = v;
    println!("{:?}",v);
}
```

Ownership rules

- Each value in Rust has a variable that's called its owner
- <u>Every value in Rust ONLY have a single owner</u>
- When the owner goes out of scope, the value will be dropped
- Use of references
- <u>A value can have ANY number of references to it</u>

Rust - Slices

Slice

```rust
fn main() {
    // Array - fixed length ; lives on the stack
    let my_array: [i32; 3] = [1, 2, 3];

    // Slice - [T]
    // - a temporary view into an array or a vector
    let my_array_slice: &[i32] = &my_array[..=1];

    println!("my_array_slice: {:?}", my_array_slice); // -> my_array_slice: [1,
}
```

Rust - Structure

Struct

```
struct Name_of_structure {
    field1:data_type,
    field2:data_type,
    field3:data_type
}
```

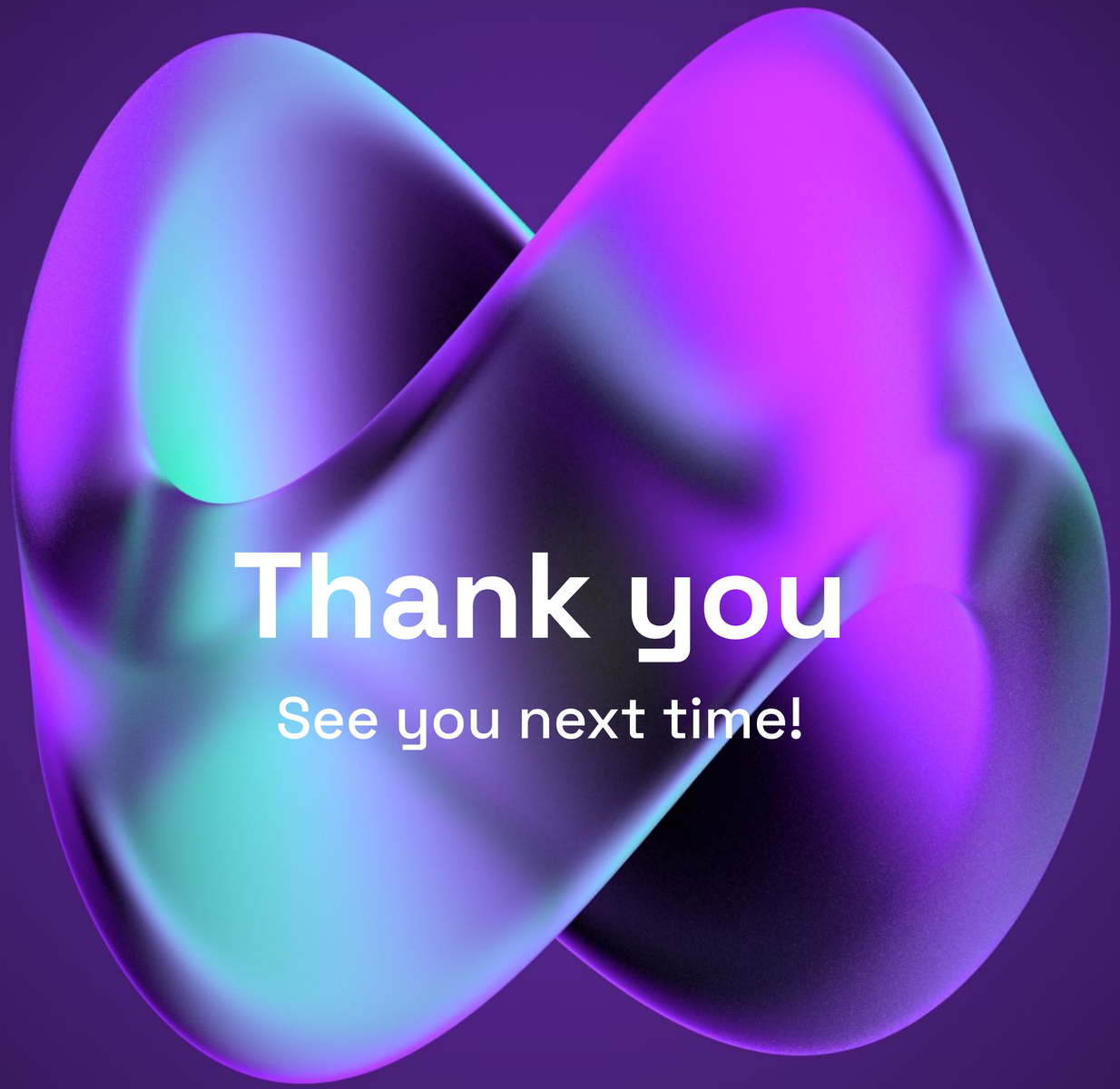
Rust - Enums

Enums

```
enum Direction {
    Up,
    Down,
    Left,
    Right
}
```

Task 2

ackee | blockchain security



Thank you

See you next time!