Rust Fundamentals

Basics of Rust
Part II

AZZAM S.A

RustCourse
Sep. 22th, 2023

Azzam S.A

OSS devotee, speaker, and teacher.

Open sourceror. Namely Rust, Python, and Emacs.

azzamsa azzamsyawqi azzamsa.com

Course. Not talk!

Follow along!

- Rust Playground
- Exercises

▶ Show hints

Simple Function

```rust
fn main() {
    // Hi, 🦀
    let name = "ferris";
    println!("My name is {}!", name);
}
```

More Examples

```rust
/// Adds two numbers and returns the result.
///
/// # Examples
///
/// ```
/// let result = add();
/// assert_eq!(result, 10);
/// ```
fn add(x: i32, y: i32) -> i32 {
    x + y
}

fn greet(name: &str) -> String {
    return format!("Hi, {}. 👋🏻", name); // 🛠️
}

fn main() {
    let result = add(5, 4);
    println!("5 + 4 = {}", result);

    println!("{}", greet("ferris"));
}
```

Exercises 1

- ☐ `intro2`
- ☐ `variables1`
- ☐ `variables2`
- ☐ `variables3`
- ☐ `primitive_types1`
- ☐ `primitive_types2`
- ☐ `if1`
- ☐ `if2`

More about Functions

Early Return

```rust
fn divide(a: i32, b: i32) -> Option<i32> {
    // Check if the divisor is zero and return early with None
    if b == 0 {
        println!("Error: Division by zero is not allowed.");
        return None;
    }

    Some(a / b)
}

fn main() {
    let result = divide(8, 0);
    println!("Result: {:?}", result);
}
```

Methods

```rust
struct Rectangle {
    width: i32,
    height: i32,
}

impl Rectangle {
    fn area(&self) -> i32 {
        self.width * self.height
    }

    fn inc_width(&mut self, delta: i32) {
        self.width += delta;
    }
}

fn main() {
    let mut rect = Rectangle { width: 10, height: 5 };
    println!("old area: {}", rect.area());

    rect.inc_width(5);
    println!("new area: {}", rect.area());
}
```

Function Overloading

```rust
fn pick_one<T>(a: T, b: T) -> T {
    if std::process::id() % 2 == 0 { a } else { b }
}

fn main() {
    println!("random number: {}", pick_one(500, 1000));
    println!("random figure: {}", pick_one("aragorn", "legolas"));
}
```

Variables

- Static and Constant Variables
- Type Inference

```rust
fn takes_i32(x: i32) {
    println!("i32: {x}");
}

fn main() {
    let x = 10;
    takes_i32(x);
}
```

Scopes and Shadowing

```rust
fn main() {
    let a = 10;
    println!("before: {a}");

    {
        let a = "hello";
        println!("inner scope: {a}");

        let a = true;
        println!("shadowed in inner scope: {a}");
    }

    println!("after: {a}");
}
```

Exercises 2

- [ ] `functions1`
- [ ] `functions2`
- [ ] `functions3`
- [ ] `variables5`
- [ ] `variables6`

Credits 🌟

- Mo’s (mo8it) Comprehensive Rust 🦀
- rustlings 🦀