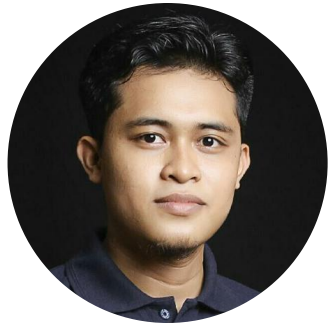
Rust Fundamentals

Basics of Rust
Part I

AZZAM S.A

RustCourse
Sep. 8th, 2023

Azzam S.A



OSS devotee, speaker, and teacher.

Open sourceror. Namely Rust, Python, and Emacs.

azzamsa azzamsyawqi azzamsa.com

Why Rust?

- ⚙️ Reliable: "if it compiles, it works"
- 🤸 Versatile: "you can do anything with Rust"

Basic Syntax

Comment

```rust
fn main() {
    // Rust programs start with fn main()
    let some_number = 100; // We can write as much as we want here and the compiler won't look at it
}
```

```rust
fn main() {
    let some_number/*: i16*/ = 100;
    /* Block comment
    It's 100, which is my favourite number.
    It's called some_number but actually I think that... */
}
```

Types

Scalar Types

	Types	Literals
Signed integers	`i8`, `i16`, `i32`, `i64`, `i128`, `isize`	`-10`, `0`, `1_000`, `123_i64`
Unsigned integers	`u8`, `u16`, `u32`, `u64`, `u128`, `usize`	`0`, `123`, `10_u16`
Floating point numbers	`f32`, `f64`	`3.14`, `-10.0e20`, `2_f32`
Strings	`&str`	`"foo"`, `"two\nlines"`
Unicode scalar values	`char`	`'a'`, `'α'`, `'∞'`
Booleans	`bool`	`true`, `false`

Compound Types

	Types	Literals
Arrays	`[T; N]`	`[20, 30, 40]`, `[0; 3]`
Tuples	`()`, `(T,)`, `(T1, T2)`, …	`()`, `('x',)`, `('x', 1.2)`, …

```rust
fn main() {
    let mut array: [i8; 10] = [42; 10];
    array[5] = 0;
    println!("array contais: {:?}", array);

    let tuple: (i8, bool) = (7, true);
    println!("2nd index: {}", tuple.1);
}
```

```rust
fn main() {
    let number: u8 = 10;
    let number = 10u8;
    let number = 10_u8;
    let number = 0_______u8;

    let name = "Ponyo";
    println!("Hello, {}!", name);
}
```

🦀 Try me

```
/// Adds two numbers and returns the result.
///
/// # Examples
///
/// ```
/// let result = add();
/// assert_eq!(result, 10);
/// ```
fn add() -> i32 {
    8 + 2
}

fn main() {
    println!("result: {}", add());
}
```

The stack, the heap, and pointers

Aspect	Stack 📚	Heap 🗄️
Memory Location	Fast.	Relatively slower
Allocation	Known size	Unknown size (String, Vectors)

The pointer in Rust is called a *reference* (`&`).

Strings

&str

- Fast
- *Does not have ownership.*
- Immutable.

String

- *Has ownership.*
- `String` is a heap-allocated, growable string type.
- Mutable.
- Dynamic memory allocation on the heap.
- Supports operations like appending, resizing, and more.

```rust
fn main() {
    let name = "Ashitaka";

    let name: String = "アシタカ".to_string();
    let name: String = String::from("アシタカ");
    let name: String = format!("{}", "アシタカ");

    let name = &name;

    println!("My name is {}", name);
}
```

```rust
fn return_str() -> &str {
    let country = String::from("Austria");
    let country_ref = &country;
    country_ref // ⚠️
}

fn main() {
    let country = return_str();
}
```

Control Flow

```rust
fn main() {
    // Using if-else to check a condition
    let number = 10;
    if number % 2 == 0 {
        println!("{} is even.", number);
    } else {
        println!("{} is odd.", number);
    }

    // Using a for loop to iterate over a range of numbers
    println!("Counting from 1 to 5:");
    for i in 1..=5 {
        println!("{}", i);
    }

    // Using a while loop to count down from 3 to 1
    let mut countdown = 3;
    println!("Countdown:");
    while countdown > 0 {
        println!("{}", countdown);
        countdown -= 1;
    }
}
```

```rust
fn main() {
    let mut counter = 0; // set a counter to 0
    loop {
        counter +=1; // increase the counter by 1
        println!("The counter is now: {}", counter);
        if counter == 5 { // stop when counter == 5
            break;
        }
    }
}
```

Variables

Const and Static

```rust
const NUMBER_OF_MONTHS: u32 = 12;
static SEASONS: [&str; 4] = ["Spring", "Summer", "Fall", "Winter"];
```

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}");
}
```