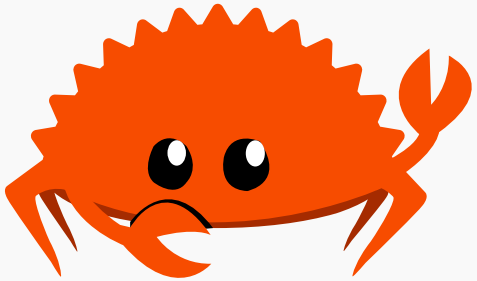
Rust

A boring and expressive language

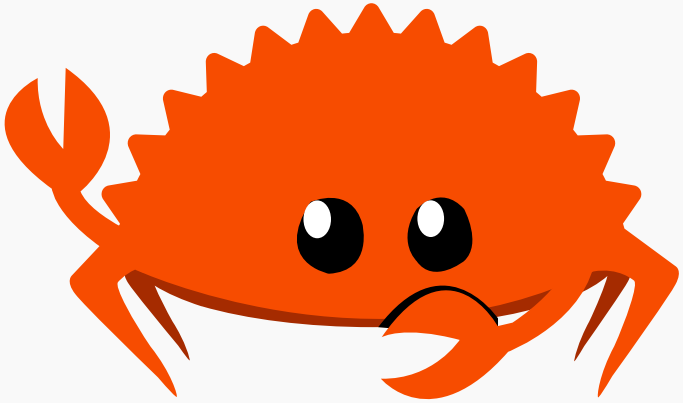
Victor Diez Ruiz



```
1  fn main() {
2    println!("Hello 🦀");
3  }
```

Why Rust rocks

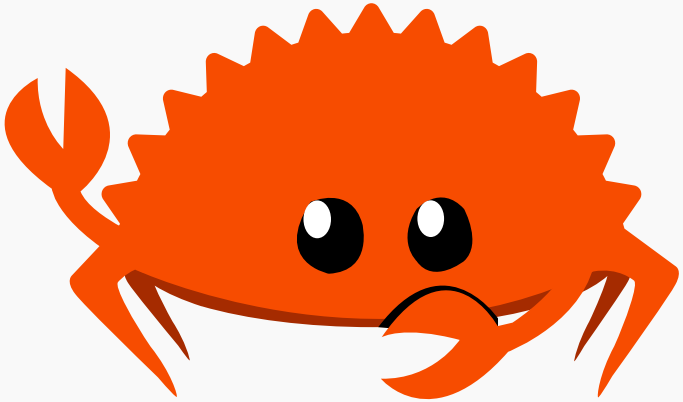
1. Lifetimes & Ownership
2. Inmutability by default
3. Algebraic Data Types
4. Pattern Matching
5. Traits
6. Macros



Lifetimes & Ownership

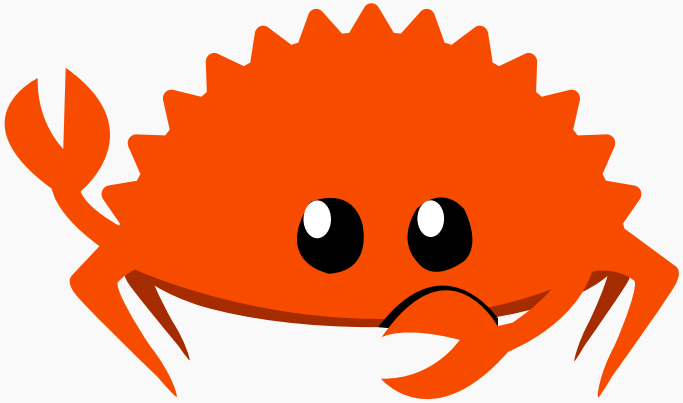
Scopes

```
1  fn main() {
2      let a = 2;
3      let b = 3;
4      println!("{}", a + b);
5  }
```

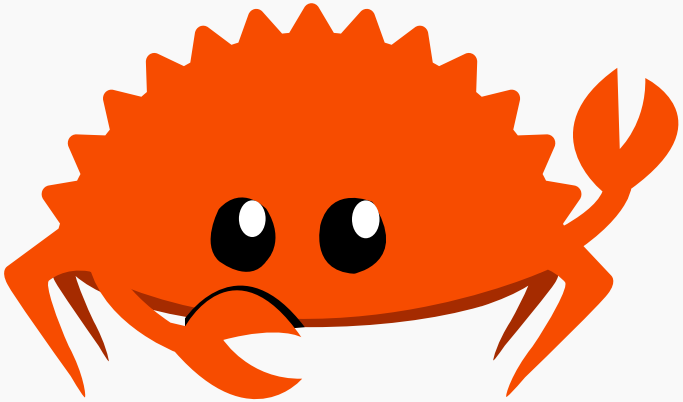


Scopes

```
1  fn main() { <scope>
2      let a = 2;
3      let b = 3;
4      println!("{}", a + b);
5  } </scope>
```

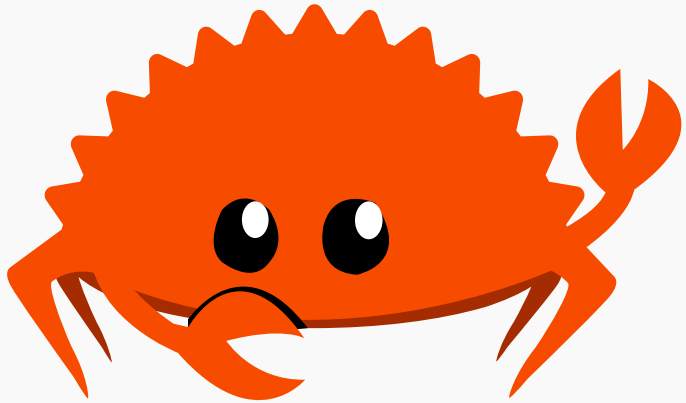


Lifetimes



```
1  fn main() {
2    let a = 2;
3    {
4      let b = 3;
5    }
6    println!("{}", a + b);
7  }
```

Lifetimes

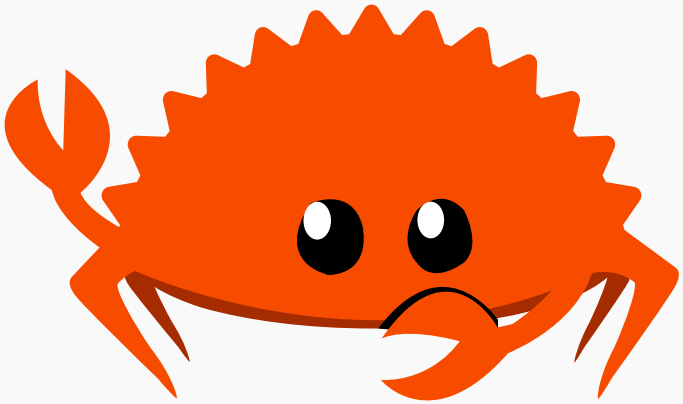


```
1  fn main() { <'a>
2      let a: 'a = 2;
3      { <'b>
4          let b: 'b = 3;
5      } </'b>
6      println!("{}", a + b);
7  } </'a>
```

Ownership

```
1  fn main() {
2    let a: 'a = 2;
3    {
4      let b: 'b = 3;
5    }
6    println!("{}", a + b);
7  }
```

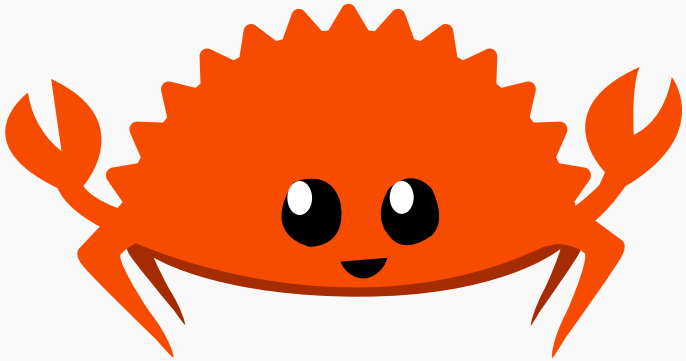
Memory representation
in 2/3 slides



Inmutability by default

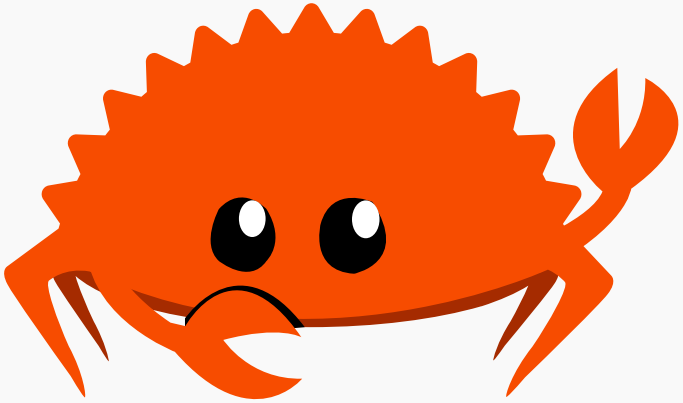
Inmutability by default

```
1  fn main() {
2      let a = 2;
3      let mut b = 3;
4
5      a = 3; // ✗ error
6      b = 2; // ✓ ok
7  }
```



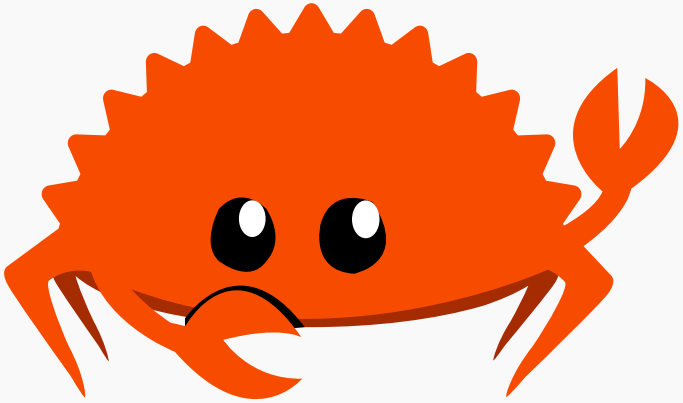
Algebraic Data Types

Algebraic Data Types



```
bool : { true, false }
```

Algebraic Data Types



```
bool : { true, false }
  u8 : { 1, ..., 255 }
```

Pattern Matching

Pattern Matching

destructurar por destructurar

Traits

Traits

interfaces pero mucho mejor

Macros

Macros

python en rust?!?!

Something very important