한 번에 끝내는 블록체인 개발 A to Z

Chapter 1

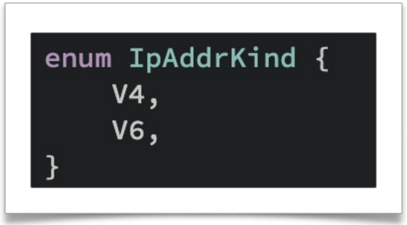
Rust Introduction

Chapter 1 Rust Introduction

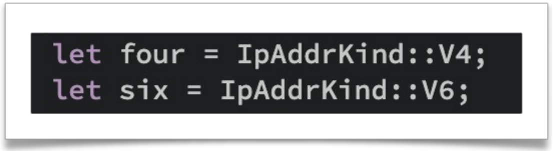
Enums

Enums

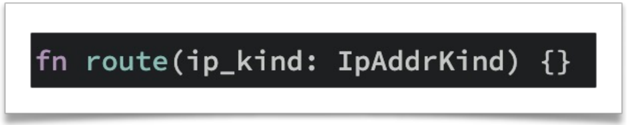
Enums are a way of defining custom data types in a different way than you do with structs. Let's look at a situation we might want to express in code and see why enums are useful and more appropriate than structs in this case.



```rust
enum IpAddrKind {
    V4,
    V6,
}
```

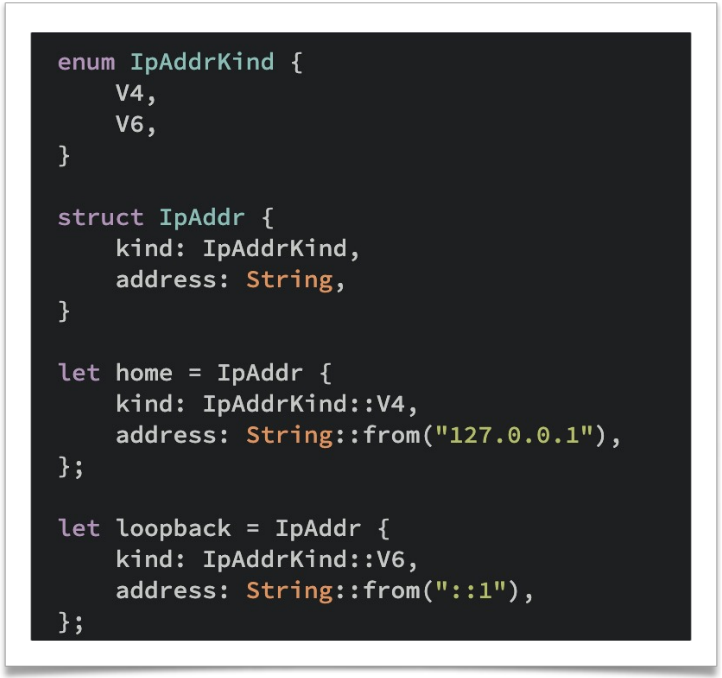


```rust
let four = IpAddrKind::V4;
let six = IpAddrKind::V6;
```

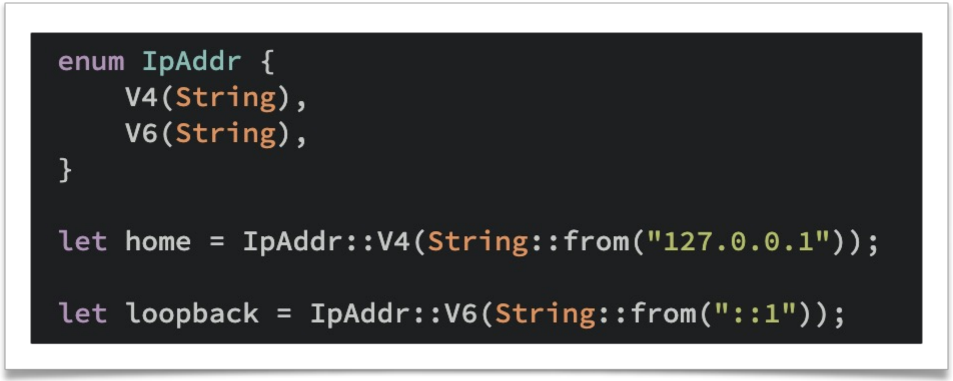


```rust
fn route(ip_kind: IpAddrKind) {}
```

Enums with Data



```
enum IpAddrKind {
    V4,
    V6,
}

struct IpAddr {
    kind: IpAddrKind,
    address: String,
}

let home = IpAddr {
    kind: IpAddrKind::V4,
    address: String::from("127.0.0.1"),
};

let loopback = IpAddr {
    kind: IpAddrKind::V6,
    address: String::from("::1"),
};
```



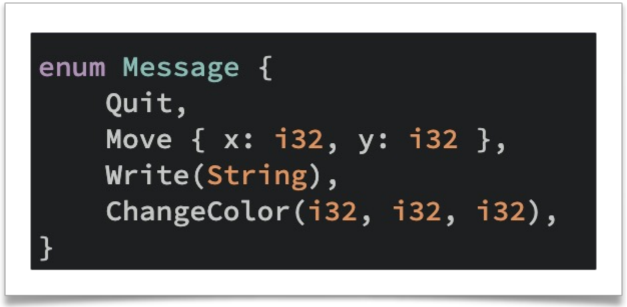
```
enum IpAddr {
    V4(String),
    V6(String),
}

let home = IpAddr::V4(String::from("127.0.0.1"));

let loopback = IpAddr::V6(String::from("::1"));
```

rather than an enum inside a struct, we can put data directly into each enum variant. This new definition of the IpAddr enum says that both V4 and V6 variants will have associated String values

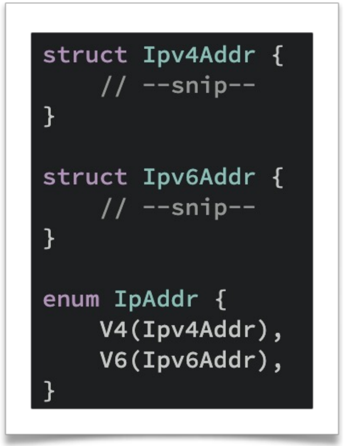
Enums with Data

There's another advantage to using an enum rather than a struct: each variant can have different types and amounts of associated data.

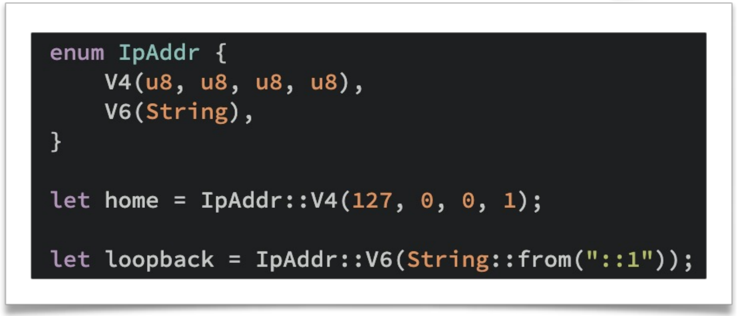
There is one more similarity between enums and structs: just as we're able to define methods on structs using impl, we're also able to define methods on enums.



```rust
enum Message {
    Quit,
    Move { x: i32, y: i32 },
    Write(String),
    ChangeColor(i32, i32, i32),
}
```



```rust
struct Ipv4Addr {
    // --snip--
}

struct Ipv6Addr {
    // --snip--
}

enum IpAddr {
    V4(Ipv4Addr),
    V6(Ipv6Addr),
}
```

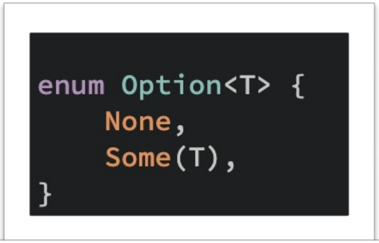


```rust
enum IpAddr {
    V4(u8, u8, u8, u8),
    V6(String),
}

let home = IpAddr::V4(127, 0, 0, 1);

let loopback = IpAddr::V6(String::from("::1"));
```

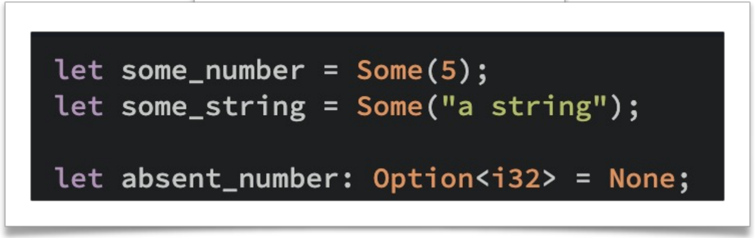
Option Enum

Rust does not have nulls, but it does have an enum that can encode the concept of a value being present or absent.

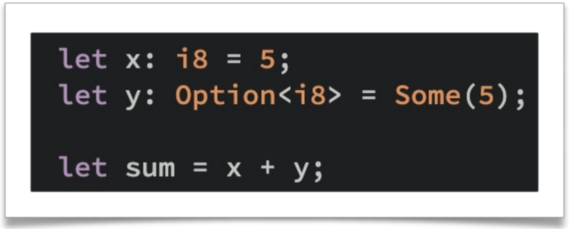
The <T> syntax is a feature of Rust we haven't talked about yet. It's a generic type parameter.



```
enum Option<T> {
    None,
    Some(T),
}
```



```
let some_number = Some(5);
let some_string = Some("a string");

let absent_number: Option<i32> = None;
```



```
let x: i8 = 5;
let y: Option<i8> = Some(5);

let sum = x + y;
```