Enums and Structures

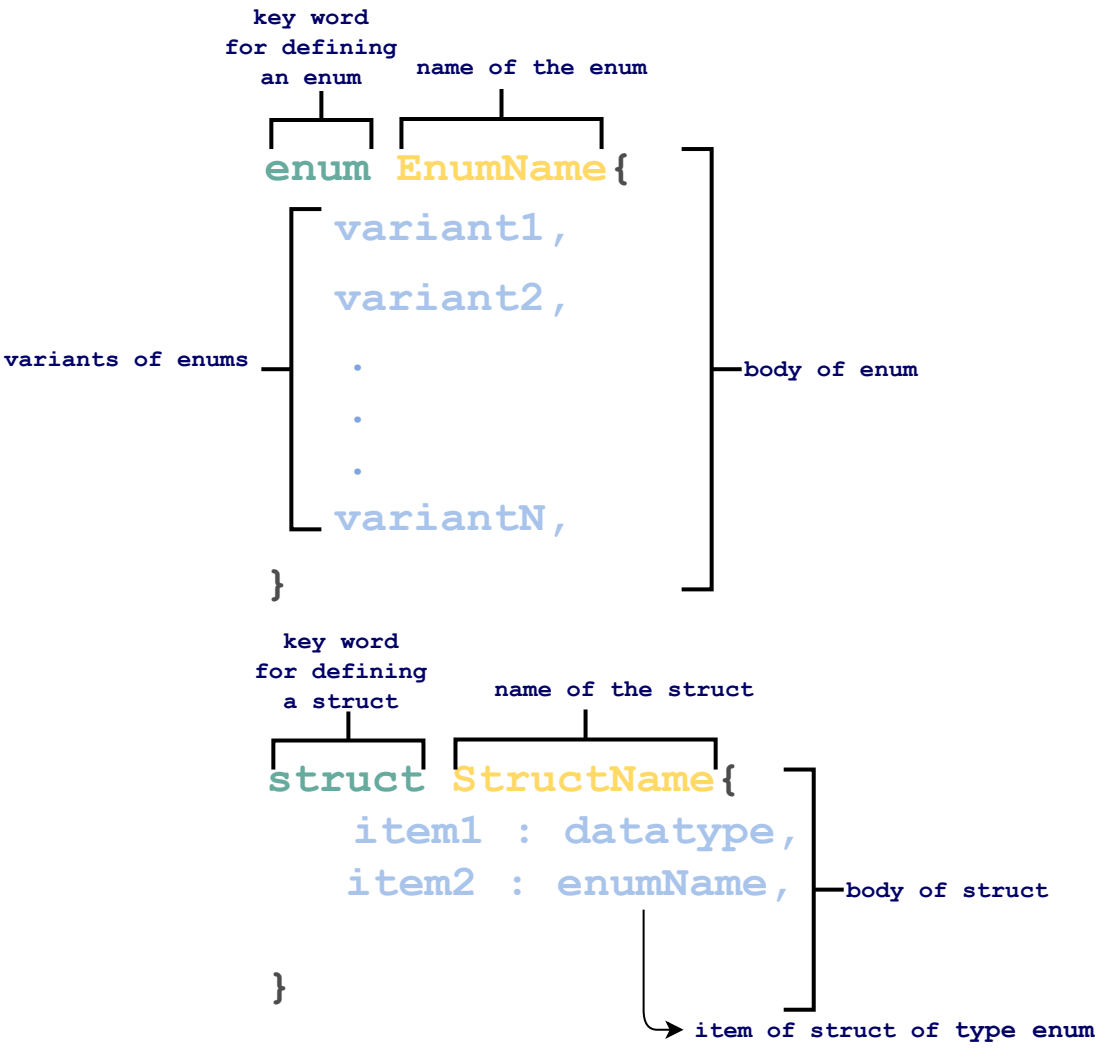
This lesson will teach how struct items can be a type of enum.

We'll cover the following

Structures can have an item that is of type `enum`.

Syntax

The following illustration explains the syntax:

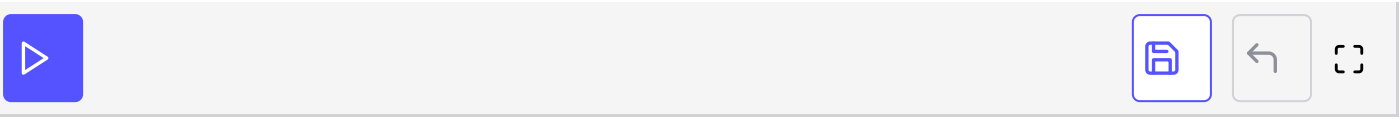


enum and struct in one program code

Example

The following example creates an `enum` `KnightMove` and a `struct` `Player.`

```
// make this `enum` printable with `fmt::Debug`.
#[derive(Debug)]
//define an enum
enum KnightMove{
   Horizontal, Vertical
}
#[derive(Debug)]
// make this `struct` print values of type `enum`  with `fmt::Debug`.
struct Player {
   color:String,
   knight:KnightMove
}
fn main() {
      // instance 1
      let p1 = Player{
      color:String::from("black"),
      knight:KnightMove::Horizontal
   };
      // instance 2
      let p2 = Player{
      color:String::from("white"),
      knight:KnightMove::Vertical
   };
   println!("{:?}", p1);
   println!("{:?}", p2);
}
```



Explanation

- `main` **Function**

  The body of the `main` function is defined from **line 13 to line 26**.

  - On **line 15** through **line 18**, instance `p1` is initialized.
  - On **line 20** through **line 23**, instance `p2` is initialized.

- On **line 2**, `#[derive(Debug)]` is declared which helps to print the values of the enum.

- `enum`

  - On **line 4**, `enum` `KnightMoves` is defined.

    - On **line 5**, **variants** of enum `Horizontal` and `Vertical` is defined.

- On **line** 7, `#[derive(Debug)]` is declared which helps to print the values of the `struct` items of type `enum`.

- `struct`

    - On **line 9**, `struct` `Player` is defined.
    - On **line 10** and **line 11**, **items** of `struct` , `color` of type `String` and `knight` of type `KnightMove` are defined.

Now that you have learned about the enums and structures, let's learn about a predefined enum in the Rust library in the next lesson.