Enum

- Enum is a custom data type which contains some definite values.
- enumerations, also referred to as enum.
- Use PascalCase

Enum syntax

```
#[derive(Debug)]
enum GenderCategory {
    Male,
    Female,
}
fn main() {
    let one = GenderCategory::Male;
    let two = GenderCategory::Female;
    println!("{:?}", one);
    println!("{:?}", two);
}
```

Structure and Enum

```
#[derive(Debug)]
enum GenderCategory {
    Male,
    Female,
}

struct People {
    name: String,
    gender: GenderCategory,
}

fn main() {
    let one = GenderCategory::Male;
    let two = GenderCategory::Female;
    println!("{:?}", one);
    println!("{:?}", two);

    let show = People {
        name: String::from("Abhishek"),
        gender: GenderCategory::Male,
    };
    println!("{}", show.name);
    println!("{:?}", show.gender);
}
```

ENUM IN RUST

Enum with Data-type

```
#[derive(Debug)]
enum H {
    Age(i32),
    Friend(String),
}
fn main() {
    let s = H::Age(23);
    let d = H::Friend(String::from("Abhishek"));
    println!("{:?}", s);
    println!("{:?}", d);
}
```

Match with Enum

```
enum Vehicle {
    Bus,
    Car,
    Bike,
}
fn take_enum(x: Vehicle) {
    match x {
        Vehicle::Bus => println!("Big size"),
        Vehicle::Car => println!("Normal size"),
        Vehicle::Bike => println!("Small size"),
    }
}
fn main() {
    take_enum(Vehicle::Bus);
    take_enum(Vehicle::Car);
    take_enum(Vehicle::Bike);
}
```