Solution Review: Find If the Day Is a Weekend

This lesson gives a detailed solution review to the challenge in the previous lesson.

We'll cover the following

Solution:

```
#![allow(dead_code)]
#[derive(Debug)]
// declare an enum
enum Days{
    Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
}
//implement Days methods
impl Days{
    // if the day is a weekend
    fn is_weekend(&self)->i32{
      match self{
        &Days::Saturday=>return 1,
        &Days::Sunday=>return 1,
        _=>return 0
      }
    }
}
fn main() {
    let mut check_day = Days::Saturday;
    println!("Is Saturday a weekend ? : {}", check_day.is_weekend());
    check_day = Days::Monday;
    println!("Is Monday a weekend ? : {}", check_day.is_weekend());
}
```

Explanation

- `enum` **construct**
  - On **line 4**, an `enum` `Days` is defined, which has 7 variants, i.e., all seven days of the week.

- `impl` `Days` **construct**

  This is defined from **line 8 to line 17**.

  - A function `is_weekend()` is defined, which takes a parameter `&self`, i.e., reference to the `enum`, and returns a boolean value.
  - Within the function `match` statement takes the `&self` and checks if the value of instance invoking the function is
    - **Saturday**, then it **returns** `1`
    - **Sunday**, then it **returns** `1`
    - if there is any other value, it returns `0`

  The following illustration explains the execution of a program line by line:

Now that you have learned about Enums, let's learn to achieve good software design practices through "traits and generics" in the next chapter.