열거형 (ENUM)

2020.3

열거형이란?

- 하나의 타입이 가질 수 있는 모든 값을 다 열거하는 것.

```
enum IpAddrKind { // IP 주소는 동시에 4버젼과 6버젼이 될 수 없고, 4이거나 6입니다. 또한 5나 2가 될 수도 없습니다.
    V4,
    V6,
}

let four = IpAddrKind::V4; // 열거형의 하나의 variant를 변수에 할당한 경우.
let six = IpAddrKind::V6;
fn route(ip_type: IpAddrKind) { } // 함수의 인자로 열거형 타입을 정의한 경우.

enum IpAddr {
    V4(String), // 열거형은 데이터를 포함할 수 있음.
    V6(String),
}
let home = IpAddr::V4(String::from("127.0.0.1"));
let loopback = IpAddr::V6(String::from("::1"));
```

Option

- Null 값이 갖는 치명적 결함.
- Option 열거형을 사용 ==> 코드가 모든 경우를 다 처리하는지 컴파일타임에 체크가 가능.

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

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; // error, type이 다름.
```

Match~1

```
enum Coin {
    Penny,
    Nickel,
    Dime,
    Quarter,
}

fn value_in_cents(coin: Coin) -> u32 {
    match coin {
        Coin::Penny => 1,
        Coin::Nickel => 5,
        Coin::Dime => 10,
        Coin::Quarter => 25,
    }
}

fn plus_one(x: Option<i32>) -> Option<i32> {
    match x {
        None => None,
        Some(i) => Some(i + 1),
    }
}

let five = Some(5);
let six = plus_one(five);
let none = plus_one(None);
```

Match~2

```
fn plus_one(x: Option<i32>) -> Option<i32> {
    match x {
        Some(i) => Some(i + 1),
    }
}
```

```
let some_u8_value = 0u8;
match some_u8_value {
    1 => println!("one"),
    3 => println!("three"),
    5 => println!("five"),
    7 => println!("seven"),
    _ => (),
}
```

If let

```
let some_u8_value = Some(0u8);
match some_u8_value {
    Some(3) => println!("three"),
    _ => (),
}

if let Some(3) = some_u8_value {
    println!(“three");
}

let mut count = 0;
match coin {
    Coin::Quarter(state) => println!("State quarter from {:?}!", state),
    _ => count += 1,
}

let mut count = 0;
if let Coin::Quarter(state) = coin {
    println!("State quarter from {:?}!", state);
} else {
    count += 1;
}
```