왕초보 코틀린 #5 분기를 위한 조건문과 조건을 만드는 비교연산자

```
fun main() {
    var a = 7
    if(a>10){
        println("a는 10보다 크다")
    } else {
        println("a는 10보다 작거나 같다")
    }
}
```

✔ 중괄호를 치는 이유

함수처럼 실행할 구문이 '여러 개'가 될 수 도 있기 때문인데 하나뿐이라면 중괄호 생략이 가능함.

✔ 비교연산자

< <= > >= != ==

- **is** 자료형이 맞는지 체크

```
a is Int
```

좌측에는 변수를 쓰고 우측에는 확인할 자료형을 쓰면 '호환'되는지 확인하고 형변환까지 시켜주는 똑똑한 연산자

- **!is** 자료형이 틀린지 체크

✔ 다중조건문 when

if가 참과 거짓만을 비교할 수 있는 반면 when은 여러개의 값과 비교할 수 있다는 장점이 있음

```
fun main() {
    doWhen(1)
    doWhen("Dimo')
    doWhen(12L)
    doWhen(3.14159)
    doWhen("Kotlin")
}
fun doWhen(a:Any){
// 패러미터 a는 Any(어떤 자료형이든 상관없이 호환되는 코틀린의 최상위 자료형)라는 자료형 사용
    when(a){
        1 -> println("정수 1입니다.")
        "Dimo' -> println("디모의 코틀린 강좌")
        is Long -> println("Long타입입니다.")
        !is String -> println("String타입이 아닙니다.")
```

```
        else -> println ("어떤 조건도 만족하지 않는다.")
    }
}
```
*다만 when을 사용할 때는 등호나, 부등호의 사용은 불가능하다.
*여러 개의 조건이 맞을 경우에도 먼저 부합하는 조건이 실행됨

위의 경우는 when을 조건에 맞는 동작을 하는 조건문으로서 이용한 경우이고 when의 조건이 맞을 때 동작 대신 값을 반환하는 표현식으로서의 역할을 하게 하려면 동작대신 값을 써주면 됨

```
fun doWhen (a:Any) {
    var result = when(a){
        1 -> "정수 1입니다."
        "Dimo' -> "디모의 코틀린 강좌"
        is Long -> "Long타입입니다."
        !is String -> "String타입이 아닙니다."
        else -> "어떤 조건도 만족하지 않는다."
    }
    println(result)
}
```
이렇게 할 경우 when의 결과를 변수에 할당하거나 직접 값으로서 사용할 수 있다.