

# 조건식

코틀린 기초강좌

강사: 배정만

# 학습요약

- 학습목표: 기초 학습 - 표현식, if, 순환문
- 핵심키워드: if Expression, when, for loop, while loop, return, ranges
- 학습내용: 조건 표현식과 순환문, 범위에 대해 배웁니다.

# Statement vs Expression

- 명령문은 값을 가지지 않는다.
- 표현식은 항상 값을 갖는다.
- if를 명령문이 아닌 표현식으로 사용할 경우 표현식에 else 분기가 있어야함 ( 항상 값을 갖기 위해 )
- 코틀린에서 리턴값을 지정하지 않으면 kotlin.Unit 값을 갖는다.

# 작업 흐름 ( Control Flow )

- If
- When
- For
- While

# if Statement

```
if( a > b ) {  
    max = a;  
} else {  
    max = b;  
}  
return max;
```

```
max = ( a > b ) ? a : b
```

# Kotlin if Expression

```
val max = if (a > b) {  
    a  
} else {  
    b  
}
```

```
val max = if (a > b) a else b
```

# When Statement

- when은 C와 유사한 언어의 스위치 연산자를 대체 합니다. 가장 간단한 형태로 다음과 같이 보입니다.

```
when (x) {  
    1 -> print("x == 1")  
    2 -> print("x == 2")  
    else -> {  
        ... run something ...  
        print("x is neither 1 nor 2")  
    }  
}
```

# When

```
when (x) {  
    0, 1 -> print("x == 0 or x == 1")  
    else -> print("otherwise")  
}
```

```
when (x) {  
    parseInt(s) -> print("s encodes x")  
    else -> print("s does not encode x")  
}
```

# Ranges

```
if (i in 1..10) { // 1 <= i && i <= 10
    println(i)
}
```

```
for (i in 4 downTo 1) print(i) // 역순
```

```
for (i in 1..4 step 2) print(i) // 2씩 증가
```

```
for (i in 1 until 10) println(i) // 10전 까지
```

# For Loops

```
for (item in collection) print(item)
```

```
for (item: Int in ints) {  
    ...  
}
```

```
for ((index, value) in array.withIndex()) {  
    println("the element at $index is $value")  
}
```

# While Loops

```
while (x > 0) {  
    x--  
}  
  
do {  
    val y = retrieveData()  
} while (y != null) // y is visible here!
```

# Returns and Jumps

- ➊ return
- ➋ break
- ➌ continue

# with Labels

```
loop@ for (i in 1..100) {  
    for (j in 1..100) {  
        if (...) break@loop  
    }  
}
```

```
fun foo() {  
    listOf(1, 2, 3, 4, 5).forEach lit@{  
        if (it == 3) return@lit  
        print(it)  
    }  
    print(" done with explicit label")  
    return@a 1  
}
```