

OUR  
SOPT

SHOUT OUR PASSION TOGETHER

# 1차 보충 세미나

— Kotlin 기초 —

SHOUT OUR PASSION TOGETHER  
SOPT

## 1. 변수 선언

val - 읽기 전용 변수  
var - 변경 가능 변수

## 2. 함수 선언

fun

## 3. 문자열 템플릿

`${}`

## 4. 반복문 for

`for(l in 0..10)`  
`for(l in 0 until 10)`

## 5. 스마트 캐스트 is

## 6. If와 when의 사용

## 7. 코틀린 null 처리

# 1. 변수 선언

val - 읽기 전용 변수

var - 변경 가능 변수

**val** num1 : Int = 2

val 로 정의한  
변수는 **변경 불가!**

**var** num2 : Int = 3

var 로 정의한  
변수는 **변경 가능!**

**val** name = “호준”

Variable을 생각하면서  
변경 가능한 놈이구나~ 하고 외우자!

## 1. 변수 선언

```
val num1 : Int = 2
```

```
var num2 : Int = 3
```

```
val name = "호준"
```

### 타입 지정하기

val 변수 : Int = 숫자

val 변수 : String = 문자

val 변수 : Float = 숫자

**: 타입** 을 이용해서  
타입을 지정 해 줄 수 있다.

Kotlin 타입 지정은 모두 대문자로!

Q. 앤 뭐죠?

A. Kotlin 은 자동으로  
타입을 추론할 수 있다!  
: String 생략 가능.

```
var name = "호준"
```

타입 추론  
-> String이구나

```
var number = 250
```

타입 추론  
-> Int 타입이구나

# lateinit

## 프로퍼티 초기화를 이루는 방법

```
lateinit var data1: String //성공
```

```
class User2(lateinit var data: String) { //에러
```

```
    lateinit val data2: String //에러
```

```
    lateinit var data3: String? //에러
```

```
    lateinit var data4: Int //에러
```

```
}
```

- lateinit는 var로 선언한 프로퍼티에만 사용할 수 있다.
- lateinit는 클래스 몸체, Top-Level, 함수 내부에 선언한 프로퍼티에 사용할 수 있다. 주 생성자에서는 사용할 수 없다.
- null 허용 프로퍼티에는 사용할 수 없다.
- 기초 타입 프로퍼티에는 사용할 수 없다.

## 2. 함수 선언

### fun



## 2. 함수 선언

### Kotlin의 함수 선언

deSigner  
develop  
Planner  
sopT

```
fun max(a: Int, b: Int): Int {  
    val num1 : Int = 2  
    var num2 : Int = 3  
  
    Log.d("num", "num1 = ${num1}")  
    Log.d("plus", "num1 + num2 = ${num1+num2}")  
    return if (a>b) a else b  
}
```

- fun : 함수 선언
- max : 함수 이름
- (a: Int, b: Int) : 파라미터 목록
- :Int 함수 반환 타입

함수 본문

# 3. 문자열 템플릿

## `{}`

### 3. 문자열 템플릿

```
fun exerciseLog() {  
    val num1 : Int = 2  
    var num2 : Int = 3
```

```
    Log.d("num", "num1 = ${num1}")  
    Log.d("plus", "num1 + num2 = ${num1+num2}")  
}
```

#### 문자열 템플릿

`${}`을 이용하여  
문자열 출력 시 변수 / 식을 함께 출력 가능하다.

#### 로그

Log.d("태그이름", "로그 내용")

#### 실행결과

```
D/num: num1 = 2  
D/plus: num1 + num2 = 5
```

## 중간 점검

a, b에 해당하는 Int값을 받아  
두 수의 곱을 로그로 출력하고,  
두 수의 합을 반환하는 함수를 만들어봅시다.

## 중간 점검

a, b에 해당하는 Int값을 받아  
두 수의 곱을 로그로 출력하고,  
두 수의 합을 반환하는 함수를 만들어봅시다.

```
fun sum(a:Int, b:Int):Int{  
    Log.d("multiply", "a 곱하기 b의 값은 ${a*b}입니다.")  
    return a+b  
}
```

sum

## 4. 반복문 for

for(l in 0..10)

for(l in 0 until 10)

## 4. 반복분 for

### 출력 값

```
for(i in 0..10){//i가 0부터 10까지 반복(10 포함)
    Log.d("for","..을 이용하면 i:${i}")
}

for(i in 0 until 10){//i가 0부터 9까지 반복
    Log.d("for","until을 이용하면 i:${i}")
}
```

```
D/for: ..을 이용하면 i:0
    ..을 이용하면 i:1
    ..을 이용하면 i:2
    ..을 이용하면 i:3
    ..을 이용하면 i:4
    ..을 이용하면 i:5
    ..을 이용하면 i:6
    ..을 이용하면 i:7
    ..을 이용하면 i:8
    ..을 이용하면 i:9
    ..을 이용하면 i:10
```

```
until을 이용하면 i:0
until을 이용하면 i:1
until을 이용하면 i:2
until을 이용하면 i:3
until을 이용하면 i:4
until을 이용하면 i:5
until을 이용하면 i:6
until을 이용하면 i:7
for: until을 이용하면 i:8
until을 이용하면 i:9
```

## 5. 스마트 캐스트 is



## 5. 스마트 캐스트 is

```
private fun printSum(voca : Any){  
    if(voca is Int){  
        val a = 15  
        val b = voca as Int  
        Log.d("test", "a + b는 ${a+b} 입니다")  
    }  
}
```

타입 검사를 했는데 또  
변환을 해준다.. 흠..?

코틀린 스마트 캐스트 is를  
이용해 타입 검사를 한 경우  
변환된 타입이 적용된다!



```
private fun printSum(voca : Any){  
    if(voca is Int){  
        val a = 15  
        val b = voca  
        Log.d("test", "a + b는 ${a+b} 입니다")  
    }  
}
```

다시 Int로 변환할 필요 없이  
이 if문 내부에서는 voca가 Int타입이다!

## 6. if와 when의 사용

## 6. if와 when의 사용

```
private fun printSum(voca : String){  
    if(voca.equals("A")){  
        Log.d("test","A 입니다")  
    }else if(voca.equals("B")){  
        Log.d("test","B 입니다")  
    }else if(voca.equals("C")){  
        Log.d("test","C 입니다")  
    }else{  
        Log.d("test","A도 B도 C도 아니네")  
    }  
}
```

else if의 향연..  
더 가독성 좋은 게 없을까?

## 6. if와 when의 사용

```
private fun printSum(voca : String){  
    when(voca){  
        ("A") -> {Log.d("test","A 입니다")}  
        ("B") -> {Log.d("test","B 입니다")}  
        ("C") -> {Log.d("test","C 입니다")}  
        else -> {Log.d("test","A도 B도 C도 아니네")}  
    }  
}
```

Java의 switch문을 대체

## 6. if와 when의 사용

```
private fun printSum(voca : String){  
    when(voca){  
        ("A") -> {Log.d("test","A 입니다")}  
        ("B") -> {Log.d("test","B 입니다")}  
        ("C") -> {Log.d("test","C 입니다")}  
        else -> {Log.d("test","A도 B도 C도 아니네")}  
    }  
}
```

이렇게도 표현 가능하다!

```
when(voca){  
    "A","B","C" -> {Log.d("test","${voca} 입니다")}  
    else -> {Log.d("test","A도 B도 C도 아니네")}  
}
```

# 7. 코틀린 null 처리

## <NullPointerException>

정의

Null때문에 발생하는 Runtime Error

## <NullPointerException>

정의

Null때문에 발생하는 Runtime Error

문제점?



## <NullPointerException>

정의

Null때문에 발생하는 Runtime Error

문제점?

- Null 자체의 의미가 모호해서 자주 발생

## <NullPointerException>

정의

Null때문에 발생하는 Runtime Error

문제점?

- Null 자체의 의미가 모호해서 자주 발생
- 에러 발생 이후 디버깅이 매우 어렵다.

### 1) Null 가능 ?

```
et_nickname.addTextChangedListener(object : TextWatcher{  
    override fun afterTextChanged(s: Editable?) {}  
    override fun beforeTextChanged(s: CharSequence? start: Int, count: Int, after: Int) {}  
    override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {  
        setBtnEnable()  
        if(s.isNullOrBlank()){ Null이 들어갈 수 있다!  
            setBtnDisable()  
        }  
    }  
})
```

## 2) Null safe operator **?.**

`s?.toUpperCase()`

S가 null이면 null을 리턴

Null이 아니라면 toUpperCase() 함수 실행

### 3) Elvis 연산자 **?:**

```
val name = hi ?: "안녕"
```

좌항이 null인 경우 default 값으로 우항을 설정

4) Safe cast

as?

```
val objA = someObj as? Something ?: DefaultObj
```

someObj 를 Something으로 캐스팅 시도

불가능하다면 → DefaultObj를 objA로 초기화

가능하다면 → Something을 objA로 초기화

### 5) 강제 not null 처리

!!

```
var num1 : String? = null
```

```
private fun mixNum(){
```

```
    num1 = "one"
```

```
    val completeNumber : String = num1!!
```

```
}
```

Num1이 nullable타입이므로

val completeNumber : String = num1 로 한다면 빨간줄이 그이게 된다

프로그래머가 num1이 null이 아님을 알고 있으므로  
!!를 통해 강제 not null처리!

6) let 함수

let{ }

```
email?.let{sendEmailTo()}
```

Email이 null이 아닐 때만 let{} 내부 로직 수행.





## 7. 코틀린 null 처리

deSigner  
develOper  
Planner  
sopT

7) property의 초기화 지연

# lateinit var

```
lateinit var name : String
```

변수 name을 나중에 초기화 하겠다!

OUR  
SOPT

SHOUT OUR PASSION TOGETHER

감사합니다^^

SHOUT OUR PASSION TOGETHER  
SOPT