

독학 Kotlin 2

Kotlin 기초 문법 완벽 가이드

변수 초기화 미루기

1. lateinit

`lateinit` 키워드를 사용하여 변수 선언하면 변수의 초기화를 미룰 수 있습니다.

사용 조건


| 조건 | 설명 |
|------|--|
| 조건 1 | <code>var</code> 키워드로 선언한 변수만 가능 |
| 조건 2 | <code>Int</code> , <code>Long</code> , <code>Double</code> , <code>Float</code> , <code>Boolean</code> , <code>Byte</code> 에는 사용 불가능 |

코드 예제

```
lateinit var data: Int           // ❌ 오류 - 기본 타입 사용 불가
lateinit val data1: String       // ❌ 오류 - val 사용 불가
lateinit var data2: String       // ✅ 성공
```

2. lazy

변수 선언문 뒤에 `by lazy {}` 형식으로 선언합니다. 변수가 최초로 이용되는 순간, 중괄호로 묶은 부분이 자동으로 실행되어 그 결과값이 변수 초기값으로 할당됩니다.

 **중요:** `val` 로 선언해야 합니다.

코드 예제

```
val expensiveResource by lazy {
    println("초기화 중...")
    "값이 계산되었습니다"
}

fun main() {
    println("프로그램 시작")
    println(expensiveResource) // 이 시점에 초기화 실행
```

```
println(expensiveResource) // 이미 초기화됨  
}
```

Type (타입)

코틀린의 모든 변수는 객체입니다. 자바에서 제공하는 원시 타입 개념이 존재하지 않습니다.

타입 종류

- 기초 타입 객체
- 문자와 문자열
- 모든 타입: `Any`
- 반환문 없는 함수: `Unit`
- null이나 예외 반환하는 함수: `Nothing`

3-1. 기초 타입 객체

`Int`, `Short`, `Long`, `Double`, `Float`, `Byte`, `Boolean`

3-2. 문자와 문자열

`Char`, `String`

3-3. 모든 타입: `Any`

 Java의 `Object`와 동일한 역할

```
val data: Any = "문자열"  
val number: Any = 42  
val boolean: Any = true
```

3-4. 반환문 없는 함수: `Unit`

`Unit` 은 생략 가능합니다.

```
fun sumAB(a: Int, b: Int): Unit { // Unit 생략 가능
    println(a + b)
}

fun main() {
    sumAB(1, 2)
}
```

3-5. ❌ null이나 예외 반환하는 함수: Nothing

Nothing 으로 선언한 변수는 null 만 대입 가능합니다.

```
fun fail(message: String): Nothing {
    throw IllegalArgumentException(message)
}

fun main() {
    fail("This function always throws an exception and never returns.")
}
```

Collection Type

여러 개의 데이터를 표현하는 방법입니다.

Collection 종류

- **Array**: 배열
- **List**: 순서 있는 데이터 집합
- **Set**: 순서 없는 고유 데이터 집합
- **Map**: 키-값 쌍의 데이터 집합

1. Array

Array는 배열을 표현하는 클래스입니다. 배열 접근 시 [] 를 이용하거나 set() , get() 을 이용합니다.

일반 Array 사용

```
fun main() {
    val data: Array<Int> = Array(3) { 0 }
    data[0] = 10
    data[1] = 20
}
```

```
data.set(2, 30)
}
```

Array 생성자 설명

```
Array(3) { 0 }
```

| 매개변수 | 설명 |
|-------|-------------------------------|
| 3 | 배열의 길이가 3인 배열을 생성 |
| { 0 } | 람다식 - 배열의 각 인덱스에 대해 초기값 0을 할당 |

기초 타입 전용 Array

기초 타입의 경우 전용 배열 클래스를 사용합니다:

```
BooleanArray, ByteArray, CharArray, IntArray,  
ShortArray, LongArray, FloatArray, DoubleArray
```

기초 타입 Array 예제

```
val data: IntArray = IntArray(3) { 0 }  
val data1: IntArray = arrayOf(1, 2, 3)
```

2. List, Set, Map

List, Set, Map은 Collection Interface를 타입으로 표현한 클래스입니다.

Collection 타입 비교

| 타입 | 순서 | 중복 허용 | 특징 |
|------|----|----------|--------------------|
| List | ✓ | ✓ | 순서 있는 데이터 집합 |
| Set | ✗ | ✗ | 고유한 데이터만 저장 |
| Map | ✗ | Key 중복 ✗ | key & value 쌍으로 구성 |

가변성 분류

Collection Type Class는 **가변(mutable)**과 **불변(immutable)** 클래스로 나뉩니다.

 **불변 클래스:** 초기에 데이터를 삽입하면 더 이상 변경 불가능

```
// 불변 컬렉션
val immutableList = listOf(1, 2, 3)
val immutableSet = setOf(1, 2, 3)
val immutableMap = mapOf("key1" to "value1")

// 가변 컬렉션
val mutableList = mutableListOf(1, 2, 3)
val mutableSet = mutableSetOf(1, 2, 3)
val mutableMap = mutableMapOf("key1" to "value1")
```

? Null 허용 / 불허용

코틀린의 모든 변수는 객체이므로 `null` 값 삽입이 가능합니다.

Null 안전성

변수 선언 시 `null` 대입 가능 여부를 명확하게 구분해서 선언해야 합니다.

 **Null 허용:** 타입 뒤에 `?`를 추가

코드 예제

```
var data1: Int = 10
data1 = null    // ❌ 오류 - null 허용하지 않음

var data2: Int? = 20
data2 = null    // ✅ 성공 - null 허용
```

Null 안전 연산자

```
val length: Int? = data2?.toString()?.length    // 안전한 호출
val result: Int = data2 ?: 0                    // 엘비스 연산자
val definitelyNotNull: Int = data2!!           // 강제 언래핑 (주의!)
```

함수 TMI

매개변수 특징

함수의 매개변수는 `var`, `val` 키워드를 사용하지 않습니다. 자동으로 `val` 이 적용됩니다.

```
fun processData(data: String) {    // 자동으로 val data: String
    // data = "새 값"    // ❌ 오류 - val이므로 변경 불가
```

```
println(data)
}
```

배열 출력

배열의 모든 요소를 출력하는 다양한 방법입니다.

출력 방법들

```
val arr: IntArray = arrayOf(1, 2, 3)

// 1 인덱스를 이용한 방법
for (i in arr.indices) {
    println(arr[i])
}

// 2 인덱스와 값을 함께 사용하는 방법
for ((index, value) in arr.withIndex()) {
    println("인덱스 $index: $value")
}

// 3 직접 값에 접근하는 방법
for (value in arr) {
    println(value)
}

// 4 함수형 스타일
arr.forEach { value ->
    println(value)
}

// 5 전체 배열을 문자열로 출력
println(arr.contentToString()) // [1, 2, 3]
```

핵심 요약 정리

체크리스트

| 개념 | 핵심 포인트 |
|------------|-------------------|
| lateinit | var + 참조 타입만 가능 |
| lazy | val + 최초 접근 시 초기화 |
| Null 안전성 | ? 로 null 허용 명시 |
| Collection | 가변/불변 구분해서 사용 |

| 개념 | 핵심 포인트 |
|-------|--------------------|
| Array | 기초 타입은 전용 Array 사용 |

🔍 자주 하는 실수

```
// ❌ 잘못된 사용
lateinit var number: Int
lateinit val text: String
val data: String = null

// ✅ 올바른 사용
lateinit var text: String
val lazyValue by lazy { "값" }
val data: String? = null

// 기본 타입 사용 불가
// val 사용 불가
// null 허용하지 않는 타입에 null 대입

// 참조 타입 + var
// lazy는 val과 함께
// null 허용 타입
```

🎉 Kotlin 기초 마스터하기! 이 가이드를 통해 Kotlin의 핵심 개념들을 완벽하게 이해해보세요!