



[스파르타코딩클럽] 안드로이드 앱개발 기초반 - 3주차



매 주차 강의자료 시작에 PDF파일을 올려두었어요!

▼ PDF 파일

[수업 목표]

1. 코틀린(Kotlin) 기본 문법 익히기
 - 1) 안드로이드에서 사용하는 프로그래밍언어인 코틀린 기본 문법 배우기
2. 코틀린 활용 - 뷰를 활용한 동작 구현하기

[목차]

- 01. 3주차 오늘 배울 것
- 02. 코틀린(Kotlin) 맛보기
- 03. Kotlin 기초 문법 배우기(1)
- 04. Kotlin 기초 문법 배우기(2)
- 05. Kotlin 연습하기
- 06. Quiz_함수와 조건문 활용하기
- 07. 뷰 정보를 활용해 동작 구현하기
- 08. Quiz_email 값이 빈값인 경우 'Email을 입력해주세요' 띄우기
- 09. 3주차 끝 & 숙제 설명
- HW. 3주차 숙제 답안



모든 토글을 열고 닫는 단축키

Windows : **Ctrl** + **alt** + **t**

Mac : **⌘** + **⌥** + **t**

01. 3주차 오늘 배울 것

- ▼ 1) 3주차: Kotlin, 앱 컴포넌트



오늘은 드디어 '코틀린'을 배웁니다.

우선 문법을 연습하고, 화면에서 더 복잡한 형태의 동작을 관리하는 방법을 배워볼게요!

02. 코틀린(Kotlin) 맛보기

▼ 2) 왜 코틀린인가요?

☞ 코틀린 이름은 JetBrains의 R&D센터 근처의 코틀린 섬에서 이름을 가져왔어요.

- 코틀린은 무료 오픈소스 프로젝트예요. 안드로이드 스튜디오의 기반이 되는 IntelliJ를 만든 JetBrains에서 2011년에 공개한 언어랍니다. JetBrains사를 주축으로 구글도 참여해 개발하고 있어요. 코틀린은 현대적인 프로그래밍 언어로, 전문 안드로이드 개발자 60% 이상이 코틀린을 사용하고 생산성, 개발만족도 및 코드 안전성을 높여준답니다.

☞ 블로그글에서 봤는데 인기 개발언어 순위가 JavaScript, Python, Java, C++ 순서라고 하던데 처음 Kotlin을 배워도 괜찮을까요?

네! 코틀린을 강력 추천드려요.

왜냐구요? 우선, 구글에서 공식으로 지원하는 형태로 안드로이드 앱을 개발하려면 Kotlin, Java, C++언어만 가능해요. 안드로이드 개발을 하신다면 다른 후보군은 고려대상이 아니에요.

그리고 Kotlin은 Java와 C++에 비해 문법이 간결해 배우기도 좋아요. 또한 나중에 서버개발을 하고싶으신 경우 Spring이나 Ktor를 이용해 서버개발도 가능해요.

- 코틀린은 간결해서 적은 노력으로 많은 작업을 할 수 있어요. 간결하다보니 생산성이 높아지는 장점이 있어요. 언어가 실수를 방지하게 만들어져서 안전한 코드에 도움을 줘요. Java 대비 '버그'가 발생할 가능성이 낮답니다. 무엇보다 코틀린을 이용해서 안드로이드 개발에 편리하도록 구글에서도 코틀린 사용자를 염두에두고 안드로이드 개발 도구를 디자인하고 신규 기능을 추가하고 있어요.

☞ 주변 개발자나 회사에서 Java로 안드로이드 개발을 한데 코틀린만 배워둬도 괜찮을까요?

네! 괜찮습니다.

이미 많은 기업의 앱이 코틀린으로 만들어졌어요. 구글의 여러앱들 뿐만아니라 에어비앤비, 트위터, 넷플릭스, 슬랙, 위챗, 에버노트 등등 알려진 글로벌 기업들도 이미 코틀린으로 안드로이드 앱을 만들고 있어요.

좋은 비유는 아니지만, 사투리를 먼저 배운다고 문제될 건 없잖아요! 심지어 사투리가 국가에서 인정받고 사람들이 사투리를 쓰기시작하면 표준어말고 사투리를 먼저 배워도 괜찮지 않을까요?

- 코틀린은 Java와 완벽히 호환이 돼요. 그래서 안드로이드 프로젝트에서 두개의 언어를 혼용해서 사용가능해요. 무엇보다 코틀린은 안드로이드 공식언어이며 구글도 코틀린 사용을 우선적으로 권장하고 있답니다.

☞ 그럼 코틀린 찬양은 이쯤하고 실제로 코틀린 언어를 구경해볼까요?

▼ 3) 코틀린 출력하기 연습



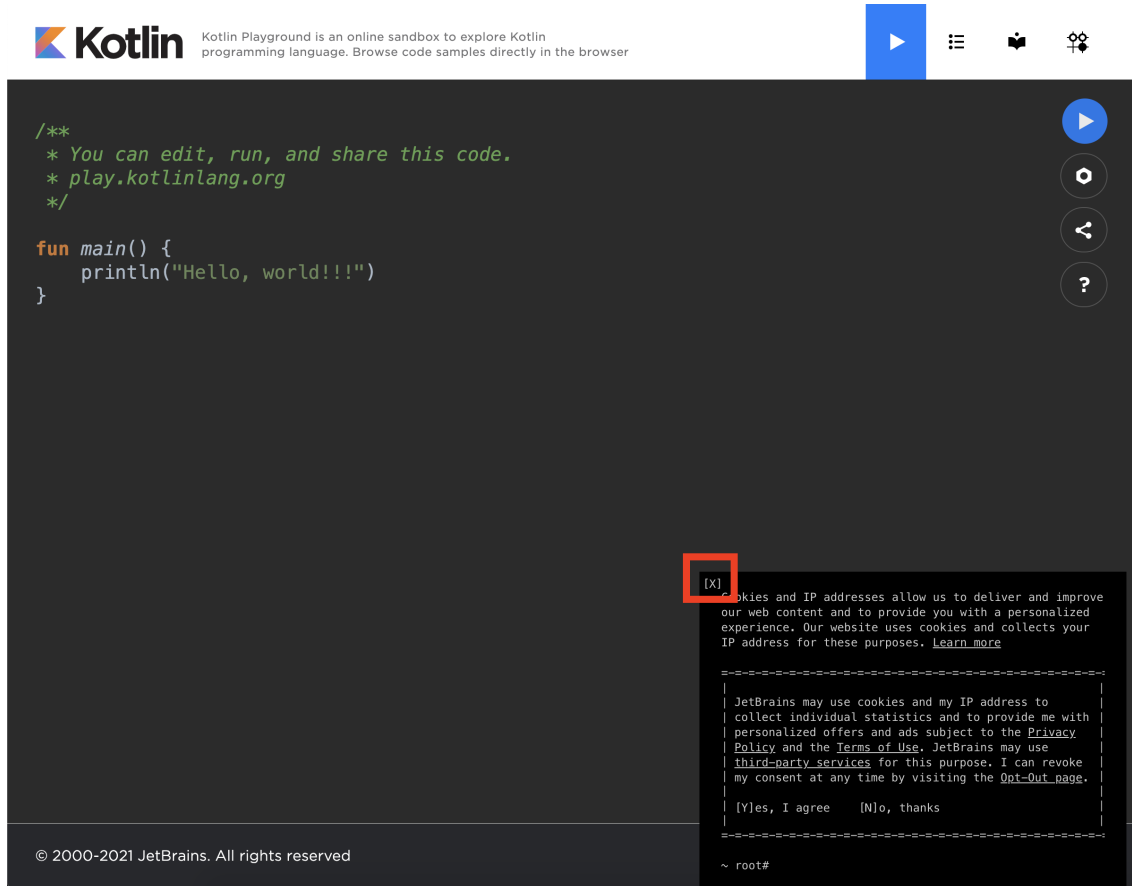
문법을 자세히 배우기 전에 코틀린에서 값을 출력하는 연습을 해볼거예요. 일단 부딪혀봅시다!

잠깐! 코틀린 문법을 배울때는 안드로이드 스튜디오를 쓰지않을거예요.

코틀린을 만든 JetBrains에서 간단한 코틀린 코드를 웹에서 사용해볼 수 있는 '코틀린 플레이그라운드'를 제공하고 있어요. 코틀린 놀이터에서 놀면서 배워봅시다.

(1) 그럼, 코틀린 플레이그라운드(Kotlin Playground)로 이동해주세요.

(2) 다음과 같은 화면이 나왔을거예요. 우선 오른쪽 아래에 있는 팝업창을 **[x]** 버튼을 눌러 꺼둡시다



(3) 코드를 한번 살펴볼까요? 초록색으로 사이트에 대한 정보를 담고있네요. 그 밑에 형태는 뭔가 익숙하지 않나요? 맞아요! 2주차 마지막에 버튼 클릭할 때 추가한 코드와 비슷하게 생겼네요.



초록색으로 된 `/** */` 형태는 **주석(Comment)**이라고 불러요. 주석은 코드에 있어도 동작에 영향을 주지 않아요. 기록을 위한 기능이라고 생각하시면 돼요. 여러줄인 경우 `/*` 로 시작해서 `*/` 로 끝나면 여러줄 주석을 남길 수 있어요.



2주차에서 다룬 `fun clickIstj(view: View) { }` 기억하시나요?

`clickIstj`를 만들고 버튼의 `onClick`에 연결하니까 버튼 클릭하면 `clickIstj`에 있던 동작이 실행됐었죠?

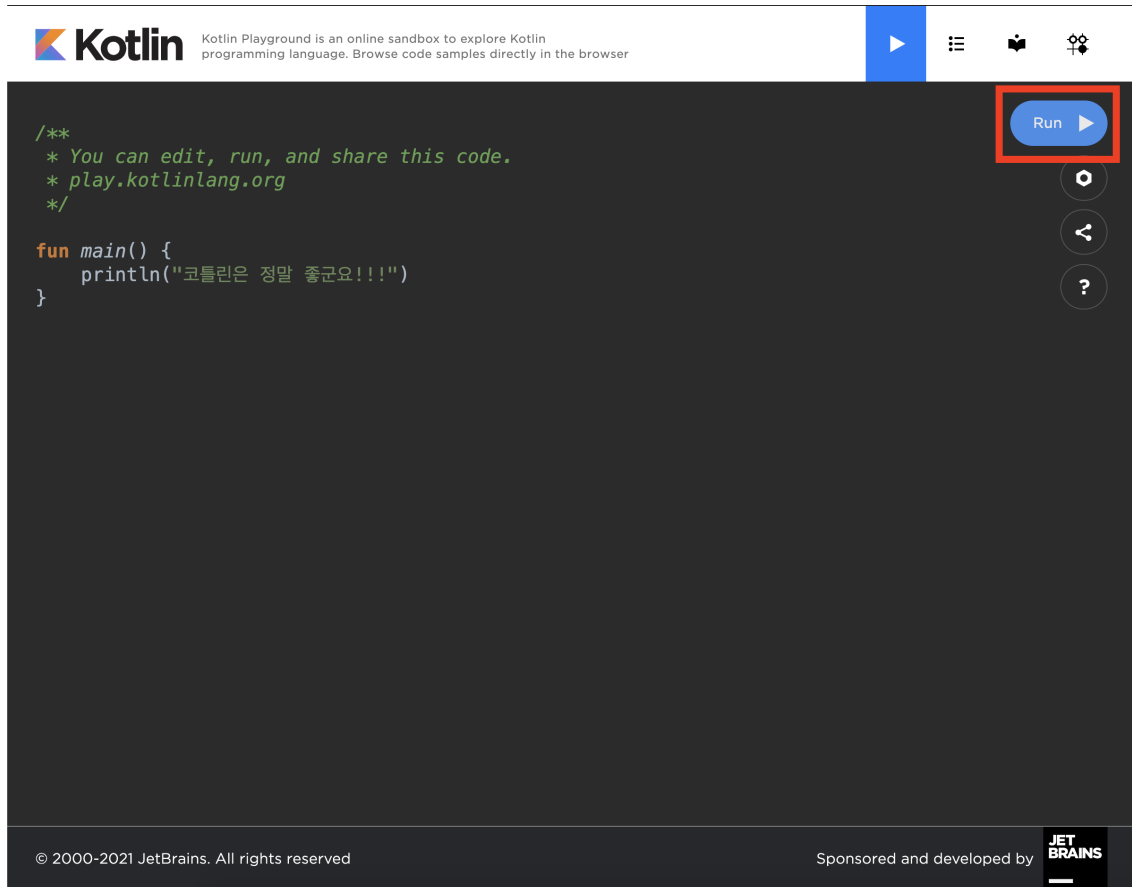
코틀린 플레이그라운드에서는 실행버튼을 누르면 `fun main() { }` 안에 있는 코드가 실행된다고 이해하시면 돼요. 일종의 약속이랍니다.

(4) "Hello, world!!!"를 "코틀린은 정말 좋군요!!!"라고 변경하고 'Run'이라고 적힌 실행버튼 눌러보세요.

▼ [코드스니펫] - 코틀린 써보기

```
/**
 * You can edit, run, and share this code.
 * play.kotlinlang.org
 */

fun main() {
    println("코틀린은 정말 좋군요!!!")
}
```



☞ 아래쪽에 로딩이 진행되다가 "코틀린은 정말 좋군요!!!"라고 표시된게 보이시나요?
'println(메세지)'는 값을 확인하거나, 에러를 찾을 때 자주 씁니다. 출력해서 버그찾는(Debugging By Printing) 스킬!

지난번 사용한 토스트메세지와 차이가 뭐냐구요? println은 토스트 메세지와 다르게 안드로이드 앱 화면에 표시되지 않아요. 콘솔창이라고 불리는 개발자들을 위한 화면에 표시돼요.

03. Kotlin 기초 문법 배우기(1)

▼ 4) 코틀린 기초 문법을 배워볼까요?



기억하기!

1. main 함수 안에 코드 넣기

코틀린 플레이그라운드(Kotlin Playground)에서는 main 함수 안에 있는 코드를 실행해요. 이것은 일종의 약속으로 생각해주시면 좋아요.

2. 출력하기

값을 확인하거나, 에러를 찾을 때 쓰입니다. 출력해서 버그찾기(Debugging By Printing) 스킬! Kotlin에서는 출력할 때, `println("출력할 값")` 사용합니다. 다만, 안드로이드 앱의 동작을 확인할 때는 안드로이드에서 지원하는 `Log`를 사용해요. 이걸 뒤에서 다뤄볼게요.

- 프로그래밍 언어는 보통 아래의 요소들로 구성되어있어요. 언어가 명사, 동사, 목적어 등으로 구성되어 있는 것과 비슷하답니다.
- 코틀린 플레이그라운드(Kotlin Playground)를 열어서 main 함수 안에 작성합니다!

▼ 변수(Variable) & 기본연산

- 변수 대입(`a = 2`)의 의미: "오른쪽에 있는 것을 왼쪽에 넣는 것!"
(2를 a라는 변수에 넣는다)
- `var` 로 변수를 선언합니다.

```
var temp = 20
temp = 5

// 변수는 값과 타입을 저장하는 박스로 생각해주세요.
```

- 변수는 처음 값의 타입을 기억해요

```
var temp1 = 20 //숫자 (Int)
var temp2 = "Bob" //문자 (String)

// temp1은 앞으로 숫자만 넣을 수 있고 temp2는 앞으로 문자만 넣을 수 있어요.
```

```
// 만약 숫자 저장공간에 글자를 넣는다면?
var temp = 20
temp = "Bob" // error 발생!! Type mismatch: inferred type is String but Int was expected
```

- 변수명은 어떻게나?

```
//변수명은 숫자로 시작되거나 일부 특수문자 또는 띄워쓰기는 불가능합니다!

//코틀린에서는 주로 camelCase를 많이 사용해요.
var firstName = "bob"

//쉽게 알아볼 수 있게 쓰는 게 중요합니다.
//a, num1, temp와 같은 변수는 무슨 용도인지 파악이 힘들겠죠?
```

```
var a = 1
var b = 2

println(a+b) // 3
println(a/b) // 0.5

var first = "Bob"
var last = "Lee"
```

```
println(first + last) // 'BobLee'

println(first + " " + last) // 'Bob Lee'

println(first+a) // Bob1 -> 문자+숫자를 하면, 숫자를 문자로 바꾼 뒤 수행합니다.
```

▼ 컬렉션 - 리스트(List), 맵(Map)



변수와 기본 연산을 다루게 되었으니 복잡한 데이터를 다루기 위해 컬렉션을 한번 살펴볼까요?

대부분의 프로그래밍 언어에는 자료구조가 있어요. 자료구조는 자료(Data)를 담은 형태(type)를 의미해요. 많은 자료형이 있지만, 가장 많이 쓰이는 두 가지 자료형 리스트와 맵을 배워보겠습니다.

컬렉션(Collections)이란, 자료 구조들을 한 곳에 모아 관리와 사용을 편하게 하기 위해 만들어 둔 것이랍니다. 대표적으로 List, Map, Set이 있어요.

- 리스트: 순서를 가지고 있는 형태의 자료 구조예요.

```
var numList = listOf(1,2,3) // 리스트를 선언. 변수 이름은 역시 아무렇게나 가능!

// 또는,

var stringList = listOf("a","b","c") // 리스트에는 같은 타입만 들어가야해요.

println(numList[0]) // 1 출력
println(numList[2]) // 3 출력
println(stringList[1]) // b 출력

// 리스트의 길이 구하기
stringList.size // 3 출력
```

- 맵: 키(key)-밸류(value) 값의 묶음

```
//사전이라고 생각하시면 편해요. key에 value를 저장하는 형태랍니다.
var myMap = mapOf("FirstName" to "Rtan", "LastName" to "Lee") // 맵을 선언. 변수 이름은 역시 아무렇게나 가능!

// 또는,

var asciiMap = mapOf(65 to "A", "0x30" to 0)

println(myMap["FirstName"]) // Rtan 출력
println(asciiMap[65]) // A 출력
println(asciiMap["0x30"]) // 0 출력

// 맵에 있는 key의 수
myMap.size
// 맵에 있는 key값 가져오기
myMap.keys // [] 출력
// 맵에 있는 value값 가져오기
myMap.values // [] 출력
```

- 수정가능한 리스트와 맵



listOf()와 mapOf()는 생성하면 읽기전용 모드로 생성되어 수정이 불가능해요.
수정 가능하게 만들려면 mutableListOf()와 mutableMapOf()를 사용해야해요.

```
//수정가능한 리스트 생성
var alphabetList = mutableListOf("a", "b", "c")

//새로운 값을 마지막에 추가
alphabetList.add("d") //d 추가
println(alphabetList) //[a, b, c, d]

//n번째에 값을 추가
alphabetList.add(2, "d")
println(alphabetList) //[a, b, d, c, d]

//값을 변경
alphabetList[0] = "A"
println(alphabetList) //[A, b, d, c, d]
// 또는,
alphabetList.set(1, "B")
println(alphabetList) //[A, B, d, c, d]

//해당 하는 값이 있으면 가장 먼저 나오는 값을 제거
alphabetList.remove("d")
println(alphabetList) //[A, B, c, d]

//n번째 값을 제거
alphabetList.removeAt(3)
println(alphabetList) //[A, B, c]
```

```
//수정가능한 맵 생성
var myMap = mutableMapOf("FirstName" to "Rtan", "LastName" to "Lee")

//새로운 값을 추가
myMap.put("language", "korean")
println(myMap) //{FirstName=Rtan, LastName=Lee, language=korean}

//값을 변경
myMap["FirstName"] = "Rtanny"
println(myMap) //{FirstName=Rtanny, LastName=Lee, language=korean}
//또는, 맵에서 값을 추가할 때 기존에 있는 키라면 값을 변경해요.
myMap.put("language", "kotlin")
println(myMap) //{FirstName=Rtanny, LastName=Lee, language=kotlin}

//값을 제거
myMap.remove("LastName")
println(myMap) //{FirstName=Rtanny, language=kotlin}
```

- 리스트와 맵의 조합

```
var students = mutableListOf(
    mapOf("name" to "bob", "age" to 20),
    mapOf("name" to "carry", "age" to 28)
)

println(students[0]["name"]) //bob 출력
println(students[1]["name"]) //carry 출력

students.add(mapOf("name" to "john", "age" to 17))

println(students)
//[{"name=bob, age=20}, {"name=carry, age=28}, {"name=john, age=17}]
```

- 왜 자료형이 필요할까요?



순서를 표시할 수 있고, 정보를 묶을 수 있습니다.

<스파르타과일가게>가 정말 잘 되어서 전국에서 손님이 찾아오고 있습니다. 대기표를 작성하기 위해서 온 순서대로 이름, 휴대폰 번호를 적도록 하였습니다. 변수만을 사용한 모습은 다음과 같습니다.

```
var customer1Name = "김스파"
var customer1Phone = "010-1234-1234"
var customer2Name = "박르탄"
var customer2Phone = "010-4321-4321"
...(휴, 알아보기 힘듭니다.)
```

👉 맵을 활용한다면 다음과 같이 고객 별로 정보를 묶을 수 있습니다.

```
var customer1 = mapOf("name" to "김스파", "phone" to "010-1234-1234")
var customer2 = mapOf("name" to "박르탄", "phone" to "010-4321-4321")
```

👉 그리고 순서를 나타내기 위해 리스트를 사용하면, 이렇게나 깔끔해집니다.

```
var customers = mutableListOf(
    mapOf("name" to "김스파", "phone" to "010-1234-1234"),
    mapOf("name" to "박르탄", "phone" to "010-4321-4321")
)
```

✅ 보기에도 깔끔해지고, 다루기도 쉬워지고, 고객이 새로 한 명 더 오더라도 .add 함수를 이용해 간단하게 대응할 수 있습니다.

▼ 기본 함수들

- 사칙연산 외에도, 기본적으로 제공하는 여러 함수들이 존재합니다.



왠지 이걸 있을 것 같은데?(예 - 특정 문자를 바꾸고 싶다 등) 싶으면 직접 만들지 말고 **구글에 먼저 찾아보세요!**

//예를 들면, '나눗셈의나머지'를 구하고 싶은 경우

```
var a = 20
var b = 7

println(a % b)
```

//또, 모든 알파벳을 대문자로 바꾸고 싶은 경우

```
var myname = "spartacodingclub"

println(myname.toUpperCase()) // SPARTACODINGCLUB
```

//또, 특정 문자로 문자열을 나누고 싶은 경우

```
var myemail = "sparta@gmail.com"

var result = myemail.split("@") // [sparta, gmail.com]

println(result[0]) // sparta
```



```
println(result[1]) // gmail.com

let result2 = result[1].split(".") // [gmail, com]

println(result2[0]) // gmail -> 우리가 알고 싶었던 것!
println(result2[1]) // com

myemail.split("@")[1].split(".")[0] // gmail -> 간단하게 쓸 수도 있다!
```

04. Kotlin 기초 문법 배우기(2)

▼ 5) 본격적으로 문법을 배워볼까요!

▼ 함수

• 기본 생김새

```
// 만들기
fun 함수이름(필요한 변수들) {
    내릴 명령어들을 순차적으로 작성
}
// 사용하기
함수이름(필요한 변수들)

// 동작을 실행하고 결과를 돌려주는(return) 경우
fun 함수이름(필요한 변수들): returnType {
    내릴 명령어들을 순차적으로 작성
    return 돌려주려는 결과 값
}
// 사용하기
var 함수결과저장변수 = 함수이름(필요한 변수들)
```

```
// 자주 쓰이는 타입

//숫자
var one: Int = 1
var pi: Double = 3.14

//글자열
var name: String = "르탄이"

//참거짓
var isEasy: Boolean = true
var isHard: Boolean = false
```

• 예시

```
// 두 숫자를 입력받으면 더한 결과를 출력하는 함수
fun printSum(num1: Int, num2: Int) {
    println(num1 + num2)
}

printSum(3, 5) //8
printSum(4, -1) //3
```

```
// 두 숫자를 입력받으면 더한 결과를 돌려주는(return) 함수
fun sum(num1: Int, num2: Int): Int {
    return num1 + num2
}

var sumResult = sum(3, 5) // 8
println(sumResult)
```


```
sumResult = sum(4, -1)
println(sumResult) // 3
```

```
// 출생년도와 계산하고 싶은 연도를 입력받아 나이를 계산해 돌려주는(return) 함수
fun calculateAge(calYear: Int, birthYear: Int): Int {
    return calYear - birthYear
}

var calAge = calculateAge(2020, 1980)
println(calAge) // 40

// 잠깐! 값을 입력하는 순서를 바꾸어도 원하는 값이 나오나요?
println(calculateAge(1980, 2020)) // -40
// 아니네요! 입력하는 순서도 맞춰줘야하는군요!
```

▼ 조건문

 **코드 조건문 01** 90보다 작으면 작다고, 크면 크다고 알려주는 함수

```
fun compareNinety(num: Int){
    if (num > 90) {
        println("90보다 커요!")
    } else {
        println("90보다 작거나 같아요!")
    }
}

compareNinety(30)
```

 **코드 조건문 02** 응용 - 다음 함수의 기능을 생각해볼까요?

```
// 함수를 정의하기
fun isAdult(age: Int): Boolean {
    if (age > 19){
        return true
    } else {
        return false
    }
}

// 함수를 사용하기
var result = isAdult(20)
println(result) // true
```

```
// 함수를 정의하기
fun isRtanny(name: String): Boolean {
    if (name == "Rtanny"){
        return true
    } else {
        return false
    }
}

// 함수를 사용하기
var result = isRtanny("Rtanny")
println(result) // true
```

 **코드 조건문 03** AND 조건과 OR 조건!

```
// AND 조건 : 모든 조건들이 true 여야 true
// 나이가 20과 같거나 크고 30보다 작은 경우 true
fun isTwenty(age: Int): Boolean {
    if (age >= 20 && age < 30){
        return true
    } else {
        return false
    }
}

// OR 조건 : 조건 중 하나라도 true이면, true!
// 나이가 20보다 작거나, 30보다 크면 true
fun isNotTwenty(age: Int): Boolean {
    if (age < 20 || age >= 30){
        return true
    } else {
        return false
    }
}

println(isTwenty(25))
println(isNotTwenty(25))
```

코드 조건문 04] if, else if, else if else

```
// 조건을 여러 개 수행하고 싶을 때
fun checkGeneration(age: Int) {
    if (age > 120) {
        println("와 19세기에 태어나셨군요!")
    } else if (age >= 100) {
        println("100세 이상! 숫자가 세자리네요!")
    } else if (age >= 80) {
        println("80세 이상! 인생은 여든부터!")
    } else {
        println("젊으시군요! 개발을 취미로 가져보세요!")
    }
}

var myAge = 55
checkGeneration(myAge)
```

▼ 반복문

- 예를 들어, 0부터 99까지 출력해야 하는 상황이라면!

```
println(0)
println(1)
println(2)
println(3)
println(4)
println(5)
...
println(99)

// 이렇게 백 줄을 쓰기엔 무리가 있겠죠? 그래서, 반복문이라는 것이 존재합니다!
```

- 반복문을 이용하면 아래와 같이 단 세 줄로, 출력할 수 있습니다.

```
for (i in 0..99) {
    println(i)
}

//0..99 는 숫자의 범위를 나타내는 방법이에요.
//1부터 10까지를 표시하고싶다면 1..10 이랍니다.
```

```
for (변수 in 리스트) {
    매 반복마다 실행될 명령어들
}
```

- 그러나 위처럼 숫자를 출력하는 경우보다는, 반복문은 주로 리스트와 함께 씁니다.
아래 예시를 볼까요? 일단 아래를 복사 붙여넣기 하고, 함께 코딩해볼게요

▼ [코드스니펫] - 반복문 예제1

```
var languages = listOf("Korean", "English", "Kotlin", "Python", "Java", "JavaScript")
```

```
var languages = listOf("Korean", "English", "Kotlin", "Python", "Java", "JavaScript")

for (language in languages) {
    println(language)
}
```

- 리스트도 그냥 리스트가 아닙니다! 맵이 들어간 리스트와 찰떡이죠
다시 아래를 복사 붙여넣기 해볼까요?

▼ [코드스니펫] - 반복문 예제2

```
var users = listOf(
    mapOf("FirstName" to "르탄", "LastName" to "박"),
    mapOf("FirstName" to "안드", "LastName" to "이"),
    mapOf("FirstName" to "로이", "LastName" to "김"),
    mapOf("FirstName" to "개발", "LastName" to "최"),
    mapOf("FirstName" to "스파", "LastName" to "김"),
)
```

```
var users = listOf(
    mapOf("FirstName" to "르탄", "LastName" to "박"),
    mapOf("FirstName" to "안드", "LastName" to "이"),
    mapOf("FirstName" to "로이", "LastName" to "김"),
    mapOf("FirstName" to "개발", "LastName" to "최"),
    mapOf("FirstName" to "스파", "LastName" to "김"),
)
```

```
for (user in users) {
    println(user)
}
```

// 이렇게 하면 리스트 내의 맵을 하나씩 출력할 수 있고,

```
var users = listOf(
    mapOf("이름" to "르탄", "성" to "박"),
    mapOf("이름" to "안드", "성" to "이"),
    mapOf("이름" to "로이", "성" to "김"),
    mapOf("이름" to "개발", "성" to "최"),
    mapOf("이름" to "스파", "성" to "김"),
)
```

```

    )

    for (user in users) {
        if(user["성"] == "김") {
            println(user)
        }
    }

    // 이렇게 하면 성이 '김'인 사용자의 이름만 출력할 수도 있습니다.

```

▼ 앓, 문법이 어렵다고요?

- 그럴 줄 알고 연습 예제들을 준비해뒀어요. (워밍업!)
- 기억하시나요? 문법을 외우는 것은 중요하지 않아요. 예를 들어 if문을 어떻게 썼더라~ 하는 것은 괜찮습니다. 코드는 복사해서 쓰세요. 구조를 이해하는 게 중요합니다!

05. Kotlin 연습하기

▼ 6) 전형적인 패턴 함께 연습하기



튜터님의 설명을 통해 이해해보겠습니다.
이번 시간에는 따라치기보다 원리를 이해하는데 집중해보세요!

▼ (1) 1부터 100까지 짝수의 합 출력하기

```

var result = 0
for (num in 1..100) {
    if(num % 2 == 0) {
        result = result + num
    }
}
println(result)

```

100이든 200이든 유용하게 쓸 수 있게, 함수로 만들어볼까요?

```

fun sumEven(n: Int): Int {
    var result = 0
    for (num in 1..n) {
        if(num % 2 == 0) {
            result = result + num
        }
    }
    return result
}

// 이러면 아래와 같은 것이 가능!
println(sumEven(100)) //1부터 100까지 짝수만 출력!
println(sumEven(200)) //1부터 200까지 짝수만 출력!

```

▼ (2) 80점 이상의 점수 갯수 출력하기

▼ [코드스니펫] - Kotlin 연습하기 점수 리스트

```

var scores = listOf(81, 79, 80, 30, 50, 100, 99, 56, 0, 12)

```

```
var scores = listOf(81, 79, 80, 30, 50, 100, 99, 56, 0, 12)

var count = 0
for (score in scores) {
    if (score >= 80) {
        count += 1 // += 는 count = count + 1 의 약어입니다
    }
}
println(count)
```

마찬가지로 유용하게 쓸 수 있게, 함수로 만들어볼까요?

```
var scores = listOf(81, 79, 80, 30, 50, 100, 99, 56, 0, 12)

fun countScore(bottom: Int): Int {
    var count = 0
    for (score in scores) {
        if (score >= bottom) {
            count += 1 // += 는 count = count + 1 의 약어입니다
        }
    }
    return count
}

// 이라면 아래와 같은 것이 가능!
println(countScore(80)) //80 이상의 점수 갯수 출력
println(countScore(90)) //90 이상의 점수 갯수 출력
```

06. Quiz_함수와 조건문 활용하기

▼ 7) 🗨️ n부터 m까지의 합 출력하기

▼ Q. 퀴즈설명: n과 m을 입력받아 n~m까지의 합을 출력하는 함수 sum을 만들어보세요!

🧠 힌트:

- 1부터 n까지는 1..n이었죠? 그럼 n부터 m까지는 어떻게 표현할 수 있을까요?
- 함수의 입력값을 여러개 받는 방법이 기억나지 않으신다면 기초문법 함수를 참고하세요!

▼ A. 함께하기

▼ [코드스니펫] - n부터 m까지 홀수의 합 출력하기

```
fun sum(n: Int, m: Int): Int {
    var result = 0
    for (num in n..m) {
        result = result + num
    }
    return result
}

sum(50, 100) //50부터 100까지의 합 출력!

//사실 이 외에도 다양한 방법이 있습니다! 코드에 절대적인 정답은 없어요!
```

▼ 8) 🗨️ 성적표에서 성적 A의 갯수 출력하기

▼ Q. 퀴즈설명: 성적표에서 성적 A의 갯수 출력해보세요!

▼ [코드스니펫] - 함수와 조건문 활용하기 성적표

```
var reportCards = listOf("A", "B", "A", "C", "D", "F", "B", "A", "A", "C")
```

🧠 힌트:

1. 반복문은 리스트를 활용하기 좋아요
2. 조건문에서 같은 것인지 확인할 때 == 을 사용해요

▼ A. 함께하기

▼ [코드스니펫] - 성적표에서 성적 A의 갯수 출력하기

```
var reportCards = listOf("A", "B", "A", "C", "D", "F", "B", "A", "A", "C")

var count = 0
for (score in reportCards) {
    if (score == "A") {
        count += 1 // += 는 count = count + 1 의 약어입니다
    }
}
println(count)
```

▼ [코드스니펫] - 성적표에서 성적 A의 갯수 출력하기 (함수화)

```
var reportCards = listOf("A", "B", "A", "C", "D", "F", "B", "A", "A", "C")

fun countScore(grade: String): Int {
    var count = 0
    for (score in reportCards) {
        if (score == grade) {
            count += 1
        }
    }
    return count
}

// 이러면 아래와 같은 것이 가능!
println(countScore("A")) //A의 갯수 출력
println(countScore("F")) //F의 갯수 출력
```

07. 뷰 정보를 활용해 동작 구현하기

▼ 9) 로그인하기 버튼 클릭하면 Toast 메시지 띄우기



Login 프로젝트를 열어주세요. File > Open Recent 를 이용하면 편합니다.
만약 못 찾았다면 새로 만들어주세요.

- 2주차 내용 복습할 겸 로그인하기 버튼을 클릭하면 Toast 메시지를 띄워볼게요.

(1) activity_main.xml을 로그인 화면 activity_main.xml 코드로 변경해주세요.

▼ [코드스니펫] - 로그인 화면 Toast 메시지 띄우기 activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <TextView
                android:id="@+id/textView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="96dp"
                android:background="#6200EE"
                android:gravity="center"
                android:padding="48dp"
                android:text="로그인 페이지"
                android:textColor="@color/white"
                android:textSize="36sp"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:orientation="vertical"
                android:padding="32dp"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@id/textView">

                <EditText
                    android:id="@+id/editTextTextEmailAddress"
                    android:layout_width="200dp"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="24dp"
                    android:hint="email"
                    android:inputType="textEmailAddress" />

                <EditText
                    android:id="@+id/editTextTextPassword"
                    android:layout_width="200dp"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="24dp"
                    android:hint="password"
                    android:inputType="textPassword" />

                <Button
                    android:id="@+id/button"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="24dp"
                    android:padding="16dp"
                    android:text="로그인하기" />

            </LinearLayout>
        </androidx.constraintlayout.widget.ConstraintLayout>
    </ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>

```

(2) MainActivity.kt 에 코드를 위치에 맞춰서 추가해주세요.

▼ [코드스니펫] - 로그인 화면 로그인하기 버튼 동작

```
fun clickLogin(view: View) {  
    Toast.makeText(view.context, "로그인!", Toast.LENGTH_LONG).show()  
}
```

```
...  
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main2)  
}  
  
fun clickLogin(view: View) {  
    Toast.makeText(view.context, "로그인!", Toast.LENGTH_LONG).show()  
}  
}
```



{ } 괄호 범위를 주의해주세요. 코드의 범위를 표현하는 것이라서 시작과 끝이 잘 맞는지 제대로 된 위치에 넣었는지 체크해주세요.

만약 범위가 잘못되었다면 실행 버튼을 눌렀을 때 에러가 발생하거나 동작이 안될 수 있어요!

(3) 이번엔 레이아웃 에디터가 아닌 코드로 추가해볼거예요. activity_main.xml 파일에서 60번째줄에 있는 Button 에 `android:onClick="clickLogin"` 을 추가해주세요.

```
<Button  
    android:id="@+id/button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="24dp"  
    android:onClick="clickLogin"  
    android:padding="16dp"  
    android:text="로그인하기" />
```

(4) 앱을 실행해서 로그인하기 버튼을 클릭하면 '로그인!'이라는 메시지가 나오나요?

▼ [코드스니펫] - 로그인 화면 로그인하기 버튼 동작 activity_main.xml (완성)

```
<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".MainActivity">  
  
    <ScrollView  
        android:layout_width="match_parent"  
        android:layout_height="match_parent">  
  
        <androidx.constraintlayout.widget.ConstraintLayout  
            android:layout_width="match_parent"  
            android:layout_height="wrap_content">  
  
            <TextView  
                android:id="@+id/textView"  
                android:layout_width="match_parent"  
                android:layout_height="wrap_content"  
                android:layout_marginTop="96dp"  
                android:background="#6200EE"  
                android:gravity="center"
```

```

        android:padding="48dp"
        android:text="로그인 페이지"
        android:textColor="@color/white"
        android:textSize="36sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="32dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/textView">

    <EditText
        android:id="@+id/editTextTextEmailAddress"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:hint="email"
        android:inputType="textEmailAddress" />

    <EditText
        android:id="@+id/editTextTextPassword"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:hint="password"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:onClick="clickLogin"
        android:padding="16dp"
        android:text="로그인하기" />

</LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
</ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>

```

▼ 10) 사용자가 입력한 Email 가져오기



레이아웃에 있는 글자를 활용하는 방법을 다뤄볼거예요. 로그인하기 버튼을 클릭하면 사용자가 입력한 EditText에 있는 email 정보를 가져와봅시다.

(1) 레이아웃에 있는 EditText 중 email을 관리할 변수를 생성해볼까요?

▼ [코드스니펫] - 사용자가 입력한 Email 가져오기 editText 선언

```
private lateinit var emailEditText: EditText
```

```
class MainActivity : AppCompatActivity() {
    private lateinit var emailEditText: EditText
}
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    ...
}
```

👉 lateinit은 나중에 값을 저장해두겠다는 의미예요. 변수를 처음 만들때가 아니라 필요한 시점에 저장하고 사용하겠다는 것으로 kotlin에만 있는 문법이랍니다.

👉 혹시 EditText 글자가 빨간색으로 보이시나요? 그렇다면, EditText위에 커서를 두고 Alt + Enter를 눌러 import 를 해주세요!
하얀색으로 나온다면 문제가 없어요.

(2) 생성한 emailEditText 변수에 실제 EditText를 찾아서 넣을거예요. email을 관리할 EditText의 아이디 값을 activity_main.xml에서 확인해봅시다.

(3) id를 editTextTextEmailAddress 로 정해두었네요. 그럼 코틀린 코드에서 findViewById 함수를 이용해 뷰를 찾아 emailEditText에 저장해볼게요. 코드를 onCreate 함수에 넣어주세요.

▼ [코드스니펫] - 사용자가 입력한 Email 가져오기 editText 찾기

```
emailEditText = findViewById(R.id.editTextTextEmailAddress)
```

```
...
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main2)
    emailEditText = findViewById(R.id.editTextTextEmailAddress)
}
...
```

👉 findViewById는 이름그대로 id 값을 이용해 view를 찾는 함수예요. id 값은 레이아웃에서 사용한 android:id 에 사용한 값이랍니다. 안드로이드 앱을 빌드(Run)하면 프로젝트가 자동으로 id 값을 관리해 줘요. 그래서 R.id.~~을 이용하면 되는 것이랍니다.

(5) emailEditText를 찾아 저장했으니 이제 emailEditText에 있는 글자를 email이라는 변수에 저장해볼게요. clickLogin 함수 코드를 변경하여 Toast를 출력해봅시다.

▼ [코드스니펫] - 사용자가 입력한 Email 가져오기 login 함수

```
fun clickLogin(view: View) {
    var email = emailEditText.text
    Toast.makeText(view.context, email, Toast.LENGTH_LONG).show()
}
```

(6) 프로젝트를 실행해서 email 항목을 바꿔보고 버튼을 눌러보세요. 입력한 email에 맞춰서 Toast값이 바뀌어 나타나네요!

▼ [코드스니펫] - 사용자가 입력한 Email 가져오기 (완성)

```
class MainActivity : AppCompatActivity() {
    private lateinit var emailEditText: EditText

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        emailEditText = findViewById(R.id.editTextTextEmailAddress)
    }

    fun clickLogin(view: View) {
        var email = emailEditText.text
        Toast.makeText(view.context, email, Toast.LENGTH_LONG).show()
    }
}
```

▼ 11) Email 값을 이용해 '로그인 페이지' 타이틀 변경하기

(1) emailEditText 선언한 곳 밑에 titleTextView를 선언할게요.

▼ [코드스니펫] - '로그인 페이지' 타이틀 변경하기 TextView 선언

```
private lateinit var titleTextView: TextView
```

```
class MainActivity : AppCompatActivity() {
    private lateinit var emailEditText: EditText
    private lateinit var titleTextView: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        ...
    }
}
```

(2) '로그인 페이지'가 적혀있는 TextView의 아이디 값을 activity_main.xml에서 확인해봅시다. `textView` 로 되어있네요. textView는 직관적이지 않으니 id를 'textViewTitle'로 변경해볼까요?

```
<TextView
    android:id="@+id/textViewTitle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="96dp"
    android:background="#6200EE"
    android:gravity="center"
    android:padding="48dp"
    android:text="로그인 페이지"
    android:textColor="@color/white"
    android:textSize="36sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```



앗, id를 변경했는데 미리보기 레이아웃이 망가졌어요. 왜냐하면 밑에있는 LinearLayout이 textView라는 id를 참조하고 있어서 그래요. 기존의 textView id가 textViewTitle로 바뀌었으니 LinearLayout에서 참조하고 있는 id도 변경해야겠죠?

```
<TextView
    android:id="@+id/textViewTitle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="96dp"
    android:background="#6200EE"
```

```

        android:gravity="center"
        android:padding="48dp"
        android:text="로그인 페이지"
        android:textColor="@color/white"
        android:textSize="36sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="32dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/textViewTitle">

```

(3) onCreate 함수에서 `findViewById` 함수를 이용해 뷰를 찾아 저장할게요.

▼ [코드스니펫] '로그인 페이지' 타이틀 변경하기 TextView 찾기

```
titleTextView = findViewById(R.id.textViewTitle)
```

```

...
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main2)
    emailEditText = findViewById(R.id.editTextTextEmailAddress)
    titleTextView = findViewById(R.id.textViewTitle)
}
...

```

(4) clickLogin에서 titleTextView의 글자를 바꾸는 코드를 추가해볼게요.

▼ [코드스니펫] '로그인 페이지' 타이틀 변경하기 TextView 찾기

```
titleTextView.text = email
```

```

fun clickLogin(view: View) {
    var email = emailEditText.text
    Toast.makeText(view.context, email, Toast.LENGTH_LONG).show()
    titleTextView.text = email
}

```

(5) 프로젝트를 실행해서 어떻게 변경되는지 확인해볼까요?

▼ [코드스니펫] - '로그인 페이지' 타이틀 변경하기 가져오기 (완성)

```

class MainActivity : AppCompatActivity() {
    private lateinit var emailEditText: EditText
    private lateinit var titleTextView: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        emailEditText = findViewById(R.id.editTextTextEmailAddress)
    }
}

```

```

        titleTextView = findViewById(R.id.textViewTitle)
    }

    fun clickLogin(view: View) {
        var email = emailEditText.text
        Toast.makeText(view.context, email, Toast.LENGTH_LONG).show()
        titleTextView.text = email
    }
}

```

▼ 12) email 형식이 아닌 경우 'Email을 확인해주세요' 띄우기

- clickLogin 함수에 조건문을 추가해서 email 형식이 아닌 경우 Toast 메시지를 띄워볼게요.

(1) email 형식 확인을 시작하기전에 '로그인 페이지' 타이틀 변경하는 코드는 불필요해보이니 제거해볼까요? 코드를 제거할 때 코드가 어떤 동작인지 파악하시는 것도 도움이 많이 돼요.

```

class MainActivity : AppCompatActivity() {
    private lateinit var emailEditText: EditText
    private lateinit var titleTextView: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        emailEditText = findViewById(R.id.editTextTextEmailAddress)
        titleTextView = findViewById(R.id.textViewTitle)
    }

    fun clickLogin(view: View) {
        var email = emailEditText.text
        Toast.makeText(view.context, email, Toast.LENGTH_LONG).show()
        titleTextView.text = email //제거
    }
}

```

(2) email 형식인지 아닌지는 단순히 '@'가 있는지 없는지만 판단할게요. contains 함수를 이용해서 문자열에 해당하는 문자가 포함되어 있는지 확인할 수 있어요. contains 함수를 이용해 조건문을 추가해봅시다.

▼ [코드스니펫] - email 형식 확인 조건문

```

fun clickLogin(view: View) {
    var email = emailEditText.text
    if (email.contains("@")) {
        Toast.makeText(view.context, email, Toast.LENGTH_LONG).show()
    } else {
        Toast.makeText(view.context, "Email을 확인해주세요", Toast.LENGTH_LONG).show()
    }
}

```

```

class MainActivity : AppCompatActivity() {
    private lateinit var emailEditText: EditText
    private lateinit var titleTextView: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        emailEditText = findViewById(R.id.editTextTextEmailAddress)
        titleTextView = findViewById(R.id.textViewTitle)
    }

    fun clickLogin(view: View) {
        var email = emailEditText.text
        if (email.contains("@")) {
            Toast.makeText(view.context, email, Toast.LENGTH_LONG).show()
        } else {

```

```

        Toast.makeText(view.context, "Email을 확인해주세요", Toast.LENGTH_LONG).show()
    }
}
}

```

(3) 프로젝트를 실행해서 입력받은 text에 '@'가 있으면 email을 출력하고 아니면 'Email을 확인해주세요'가 되는지 확인해봅시다.

▼ [코드스니펫] - 사용자가 입력한 Email 가져오기 (완성)

```

class MainActivity : AppCompatActivity() {
    private lateinit var emailEditText: EditText
    private lateinit var titleTextView: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        emailEditText = findViewById(R.id.editTextTextEmailAddress)
        titleTextView = findViewById(R.id.textViewTitle)
    }

    fun clickLogin(view: View) {
        var email = emailEditText.text
        if (email.contains("@")) {
            Toast.makeText(view.context, email, Toast.LENGTH_LONG).show()
        } else {
            Toast.makeText(view.context, "Email을 확인해주세요", Toast.LENGTH_LONG).show()
        }
    }
}

```

08. Quiz_email 값이 빈값인 경우 'Email을 입력해주세요' 띄우기

▼ 13) 🚩 email 값이 빈값인 경우 'Email을 입력해주세요' 띄우기

▼ Q. 퀴즈설명

💡 힌트:

1. 조건문을 활용하면 되겠죠!
2. length를 이용하면 길이를 알 수 있습니다.
3. 아니면, 구글에 '코틀린 문자열 빈값 체크'로 검색해보세요.
4. if () { } else if { } else { } 형태로 조건문에서 조건을 여러개 사용할 수 있어요.
4. 코드는 순서대로 진행됩니다. 'Email을 확인해주세요'보다 먼저 조건문이 들어가야 해요.

▼ A. 함께하기(완성본)

▼ [코드스니펫] - Email을 입력해주세요 띄우기

```

class MainActivity : AppCompatActivity() {
    private lateinit var emailEditText: EditText
    private lateinit var titleTextView: TextView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main2)
        emailEditText = findViewById(R.id.editTextTextEmailAddress)
        titleTextView = findViewById(R.id.textViewTitle)
    }
}

```

```

    }

    fun clickLogin(view: View) {
        var email = emailEditText.text
        if (email.length == 0) {
            Toast.makeText(view.context, "Email을 입력해주세요", Toast.LENGTH_LONG).show()
        } else if (email.contains("@")) {
            Toast.makeText(view.context, email, Toast.LENGTH_LONG).show()
        } else {
            Toast.makeText(view.context, "Email을 확인해주세요", Toast.LENGTH_LONG).show()
        }
    }
}

```

09. 3주차 끝 & 숙제 설명

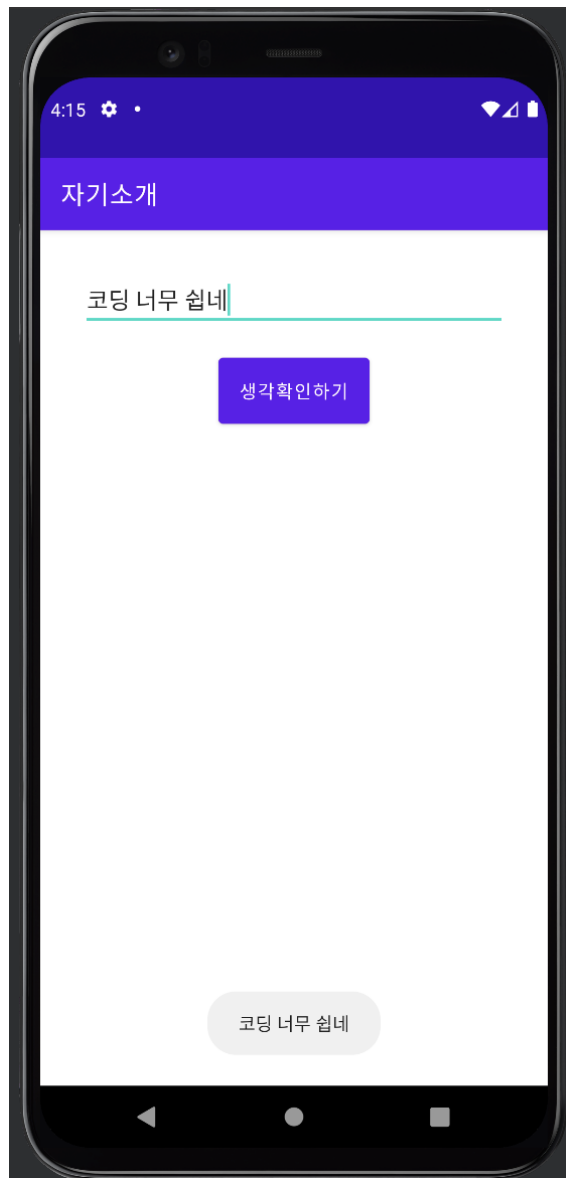


자기소개 앱에 버튼 동작을 구현해주세요.

기능: 생각확인하기 버튼을 눌렀을 때 EditText에 입력되어있는 내용을 그대로 Toast에 띄워주세요.

* 2주차 과제를 못하신 분들은 빈 프로젝트에 EditText와 Button을 하나씩 추가해서 진행해주세요.

▼ 예시 화면



입력값 Toast로 출력

HW. 3주차 숙제 답안

▼ [코드스니펫] MainActivity 일부

```
class MainActivity : AppCompatActivity() {
    private lateinit var thinkEditText: EditText

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        thinkEditText = findViewById(R.id.editTextThink)
    }

    fun clickThinkButton(view: View) {
        var thinkText = thinkEditText.text
        Toast.makeText(view.context, thinkText, Toast.LENGTH_LONG).show()
    }
}
```

Copyright © TeamSparta All rights reserved.