**Week 1 : 안드로이드 Essential – Kotlin**

0. Kotlin이란?

- Jetbrains 사에서 개발되었다.

- 2016년 정식 버전이 발표되어, 2017년에는 안드로이드 공식지원언어로 채택되었다.

1. Kotlin의 장점은?

- 문법이 간단하다.

- 표현력이 좋다.

- NullPoiinterExeption (Java 언어에서 흔히 발생하는 오류)를 피할 수 있다.

= 즉, ‘Null Safe’ 언어이다.

- 기존 라이브러리와 상호운용성이 좋다. (JAVA 와 100% 호환 가능)

2. 변수

1) 변수(Variable) 란?

값을 넣을 수 있는 상자

2) 변수의 종류

내 마음대로 원하는 것을 넣을 수 있는 상자 -> Variable

한번 넣으면 바꿀 수 없는 상자 -> Value

3) 변수의 선언 (1)

var/val 변수 = 값

주의 : val 변수를 바꾸려고 하면, “Val cannot be reassigned”라는 오류가 발생

또한, 여러 단어로 된 변수명을 네이밍 할 때에는 카멜 표기법 혹은 스네이크(언더바) 표기법을 사용한다.

3. 자료형

변수의 Type을 나타낸다

1) 정수형

음수, 양수 모두 표현 가능하다.

Long > Int > Short > Byte

2) 실수형

실수를 표현한다.

Double > Float

3) 문자

한글자만을 표현한다.

작은 따옴표로 표현한다.

Char

4) 문자열

여러 글자를 표현한다.

큰 따옴표로 표현한다.

String

5) 논리형

참과 거짓의 표현한다.

True, False (참/거짓) -> Boolean

주의 : 만약, 자료형에 맞지 않는 변수를 넣는다면 ‘The floating-point literal does not conform to the expected type Int’ 라는 오류가 뜬다.

6) 변수의 선언 방법 (2)

var/val 변수 : 자료형 = 값

7) 자료형 추론

변수를 선언할 때, 자료형을 적어주는 것이 좋다.

정수의 경우 Int로 추론한다.

실수의 경우 Double로 추론한다.

8) Variable or Value??

변하지는 않는 값이라면 -> Value (외에는 Varialbe)

진짜 모르겠다! -> Value (이후에 바꿀 일이 있을 때 변경하는 것을 추천)

9) null

Int, Double 등은 Null을 가질 수 없는 변수이다.

var x : Int? = null (가능하다.)

"?." 연산자는 안전 호출 연산자라고도 한다.

객체의 속성이나 함수를 안전하게 호출할 수 있게 해 준다.

즉, 객체를 참조하는 변수의 값이 null이 아닐 때만 호출하게 해주는 것이다.

4. 함수

1) 함수란?

어떤 input을 넣어주면 어떤 output 나오는 것

2) 함수를 선언하는 방법

fun 함수명 (변수명: Type, 변수명: Type ....) : 반환형 Type {

함수 내용

return 반환값

}

3) 디폴트 값을 갖는 함수 만들기

인자에 아무런 값이 들어오지 않으면 기본값(default)으로 함수가 실행된다.

fun plusFive(first: Int, second: Int = 5): Int {

val result: Int = first + second

return result

}

4) 반환값이 없는 함수 만들기 만들기(1)

fun printPlus(first: Int, second: Int): Unit {

val result: Int = first + second

println(result)

}

<Unit은 Java의 Void와 비슷한 역할을 한다. 생략 가능>

5) 간단하게 함수를 선언하는 방법

fun plusShort(first: Int, second: Int) = first + second

6) 가변인자를 갖는 함수 선언하는 방법

vararg는 가변인자를 뜻한다.

fun plusMany(vararg numbers: Int) {

for (number in numbers) {

println(number)

}

}

7) main 함수에서 출력하기

fun main(array: Array<String>) {

// 함수를 호출 하는 방법

val result = plus(5, 10)

println(result)

// 인수를 명시적으로 전달하는 방법

val result2 = plus(first = 20, second = 30)

println(result2)

val result3 = plus(second = 100, first = 10)

println(result3)

// 디폴트 값을 갖는 함수 호출하기

println()

val result4 = plusFive(10, 20)

println(result4) #30

val result5 = plusFive(10)

println(result5) #15

println()

printPlus(10, 20) #30

}

5. 연산자

1) 산술 연산자

+, -, \*, /(몫만 취함), %(나머지만 취함)

2) 대입 연산자

좌변 = 우변 (우변 값이 좌면에 들어간다)

a = 5 (o), 5 = a (x)

3) 복합 대입 연산자

+=, -=, \*=, /=, %=

a += 10 -> a = a + 10

4) 증감 연산자

++, --

a++, a--

5) 비교 연산자

>, >=, <, <=, ==, !=

True == True -> True

True == False -> False

True != Ture -> False

True != False -> True

6) 논리 연산자

&&, ||, !

True && True -> True

True || False -> True

!True -> False

!False -> True

6. 분기문

1) 제어 흐름 (분기문)

if, else 를 사용한다.

fun main(args: Array<String>) {

val a: Int = 5

val b: Int = 10

2) if/else 사용 하는 방법 (1)

if (a > b) {

println("a 가 b 보다 크다")

} else {

println("a 가 b 보다 작다")

}

3) if/else 사용 하는 방법 (2)

if (a > b) {

println("a 가 b 보다 크다")

}

4) if/else/else if 사용 하는 방법 (2)

if (a > b) {

println("a 가 b 보다 크다")

} else if (a < b) {

println("a 가 b 보다 작다")

} else if (a == b) {

println("a 와 b는 같다")

} else {

}

5) 값을 리턴하는 if 사용방법

val max = if (a > b) {

a //5

} else {

b //10

}

6) 값을 리턴하는 if 사용방법(2)

val max1 = if (a > b) a else b

7) when 사용 방법

when (value) {

1 -> println("value is 1")

2 -> println("value is 2")

3 -> println("value is 3")

else -> println("I do not know value")

}

8) 값을 리턴하는 when 사용 방법

val value2 = when (value) {

1 -> 10

2 -> 20

3 -> 30

else -> 100

}

7. 배열

1)배열이 필요한 이유

그룹(모음집)이 필요할 때

fun main(array: Array<String>) {

2) 배열을 생성하는 방법(1)

var number: Int = 10

var group1 = arrayOf<Int>(1, 2, 3, 4, 5)

println(group1 is Array)

3) 배열을 생성하는 방법(2)

var number1 = 10

var group2 = arrayOf(1, 2, 3.5, "Hello")

4) Index 란

순서(번째)

[1, 2, 3, 4, 5]

"0"부터 시작

index 0 -> 1, index 1 ->2

5) 배열의 값을 꺼내는 방법(1)

val test1 = group1.get(0)

val test2 = group1.get(4)

println(test1)

println(test2)

6) 배열의 값을 꺼내는 방법(2)

val test3 = group1[0]

println(test3)

7) 배열의 값을 바꾸는 방법(1)

group1.set(0, 100)

println(group1[0])

8) 배열의 값을 바꾸는 방법(2)

group1[0] = 200

println(group1[0])

}

8. 반복문

fun main(array: Array<String>) {

val a = mutableListOf<Int>(1, 2, 3, 4, 5, 6, 7, 8, 9)

1)반복하는 방법

for (item in a) {

if (item == 5) {

println("item is Five")

} else {

println("item is not Five")

}

}

println()

2) 반복하는 방법

for ((index, item) in a.withIndex()) {

println("index : " + index + " value : " + item)

// 문자열 + Int(정수) = 문자열

// 문자열 + 아무거나 = 문자열

}

println()

3) 반복하는 방법

a.forEach {

println(it)

}

println()

4) 반복하는 방법

a.forEach { item ->

println(item)

}

println()

5) 반복하는 방법

a.forEachIndexed { index, item ->

println("index : " + index + " value : " + item)

}

6) 반복하는 방법

println(a.size)

println()

for (i in 0 until a.size) {

// until은 마지막을 포함 하지 않는다

// 0 부터 8까지 이다

println(a.get(i))

}

println()

7) 반복하는 방법

for (i in 0 until a.size step (2)) {

println(a.get(i))

}

println()

8) 반복하는 방법

for (i in a.size - 1 downTo (0)) {

// 8 부터 0 까지 반복

println(a.get(i))

}

println()

9) 반복하는 방법

for (i in a.size - 1 downTo (0) step (2)) {

println(a.get(i))

}

println()

10) 반복하는 방법

for (i in 0..a.size) {

// .. -> 마지막을 포함한다

println(i)

}

println()

11) 반복하는 방법

var b: Int = 0 // -> 1 -> 2 -> 3 -> 4

var c: Int = 4

while (b < c) {

b++ // while문을 정지 시키시키기 위한 코드

println("b")

}

var d: Int = 0

var e: Int = 4

println()

12) 반복하는 방법

do {

println("hello")

d++

} while (d < e)

}