코틀린 기초 문법

main 함수

fun main() { ... }

fun = function

```
package com.example.study11


fun main() {
    |
}
```

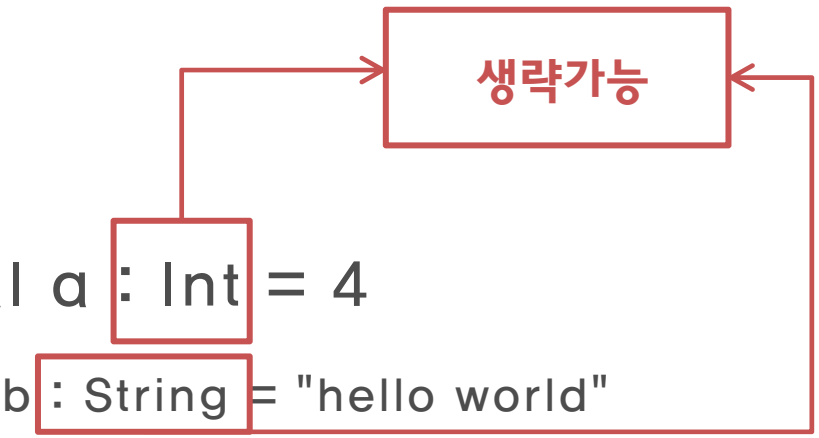
코틀린 기초 문법
기본 변수

val vs var

val : 변하지 않는 값, 한번 선언하면 변경 불가
var : 일반 변수 선언시 쓰는 듯

선언 방법

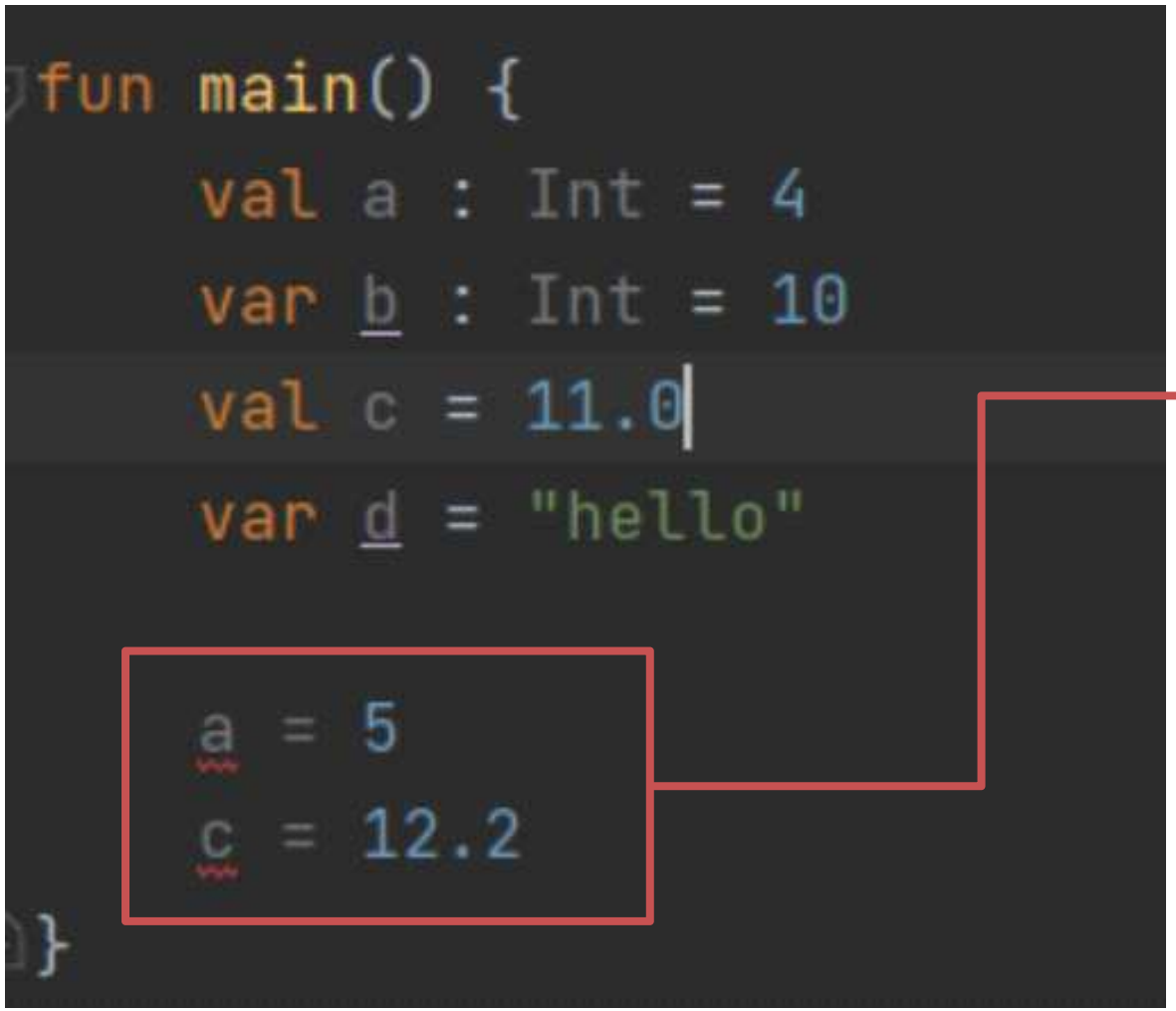
val 변수이름 : 타입 = 값 ex) val a : Int = 4
var 변수이름 : 타입 = 값 ex) var b : String = "hello world"



단, " val a " 처럼 이름만 선언시
" : 타입 " 을 반드시 명시해야함

코틀린 기초 문법

기본 변수



a, b는 val이기 때문에
값 변경 불가데숭

코틀린 기초 문법

String Templete

```
val name = "Yumin"
println("my name is $name Hello")
```

'$' + 선언된 변수이름 을 사용함으로써 C언어에서 %d,%f,%c 같은 효과를 낼수 있음

코틀린 기초 문법

String Templete

주의할 점

```
println("my name is $nameHello")
```

'$'뒤로 공백없이 쓰게되면 공백을 만날때까지 변수이름으로 인식됨
-> nameHello로 인식

-> println("my name is ${name}Hello")
' {} '를 사용하면 괄호 안에만 변수이름으로 인식한다

-> println("my name is ${name + name2}Hello")
안에 + 를 사용함으로써 다른 변수와도 연결 가능 !

코틀린 기초 문법

String Templete

예시

```
fun main() {
    val name = "yumin"
    val name2 = "yumi"

    println("My name is $name Hello")
    println("My name is ${name}Hello")
    println("My name is ${name + name2}Hello")
}
```

```
My name is yumin Hello
My name is yuminHello
My name is yuminyumiHello

Process finished with exit code 0
```

코틀린 기초 문법

사용자 지정 함수 만들기

fun 함수이름 (매개변수이름 : 타입, ...) : 반환타입 { . . . }

반환타입
Unit -〉 void, 즉 아무 반환값이 없는것 (생략가능)
String
Int
Float
Double 등등..

코틀린 기초 문법

사용자 지정 함수 만들기

예시

```
fun main() {

}

fun studyFun(a : Int, b: Int) : Int {
    return a + b
}

fun studyFun2(a : Double, b: Double) : Double {
    return a - b
}
fun voidFun() : Unit { }
fun voidFun2() {}
```

코틀린 기초 문법

조건문 - if

1)
```
fun maxby(a : Int ,b : Int) : int {

  if (a > b) return a
  else return b

}
```

2)
```
fun maxby2(a : Int, b : Int) = if(a > b) a else b
```

-> 반환 타입은 자동으로 해줌

코틀린 기초 문법

조건문 - if

```
fun main() {

    println(maxBy( a: 10, b: 20))
    println(maxBy2( a: 20, b: 10))
}
fun maxBy(a : Int, b : Int) : Int {
    if(a > b)return a
    else return b
}

fun maxBy2(a : Int, b : Int) = if(a > b) a else b
```

```
"C:\Program Files\Android\Android Studio\jre\bin\java.exe" ...
20
20

Process finished with exit code 0
```

코틀린 기초 문법

조건문 - when

when 조건문은 코틀린에 있는 switch - case 같은 문법임

1) 일반사용법

```
var score = 5
when(score) {

  1-> println("점수는 일")
  2-> println("점수는 이")
  3-> println("점수는 삼")
  else-> println("범위벗어남")

}
```

switch-case의 default, 생략가능

코틀린 기초 문법

조건문 - when

2) 변수에 사용

```
var = b = when(score) {
  1–〉 1
  2–〉 2
  else – 〉 3
}
```

score의 값에 따라서 변수 b의 값이 달라짐 !

변수에 사용할때는 생략 불가

코틀린 기초 문법

조건문 - when

3) when 범위 지정

```
var score = 80
when(score) {

  in 90..100-> println("잘했어요")
  in 40..89-> println("보통이네요")
  else-> println("왜케 못함")

}
```

코틀린 기초 문법

배열 vs 리스트

배열(array) : 크기가 정해져 있는 값들의 집합
크기만 아니면 값들은 변경 가능

리스트(List) : 크기가 정해져 있지 않고 쭉 이어진 값들의 집합

리스트는 종류가 두가지 있음
1. inMutable List : 수정불가능 리스트
2. Mutable List : 수정가능 리스트

코틀린 기초 문법

배열 vs 리스트

배열 사용법

```
val array = arrayOf(1,2,3) //선언 및 초기화

array[0] = 3 // 인덱스 0 값 수정
```

코틀린 기초 문법

배열 vs 리스트

리스트 사용법

```
val list = listOf(1,2,3)  // 리스트 선언 및 초기화
val arrayList = arrayListof<Integer>() // 수정가능한 list 선언

println(list.get(0))  //리스트 인덱스0 값 가져오기
arrayList.add(10) // 수정가능 리스트 값 추가
arrayList.add(20)
```