

EDCAN 수업 자료

Kotlin 2

이 수업에서 배우는 것들

함수, 람다, 객체지향,

class, 상속, 접근 제한자

추상 클래스, 인터페이스, 데이터 클래스

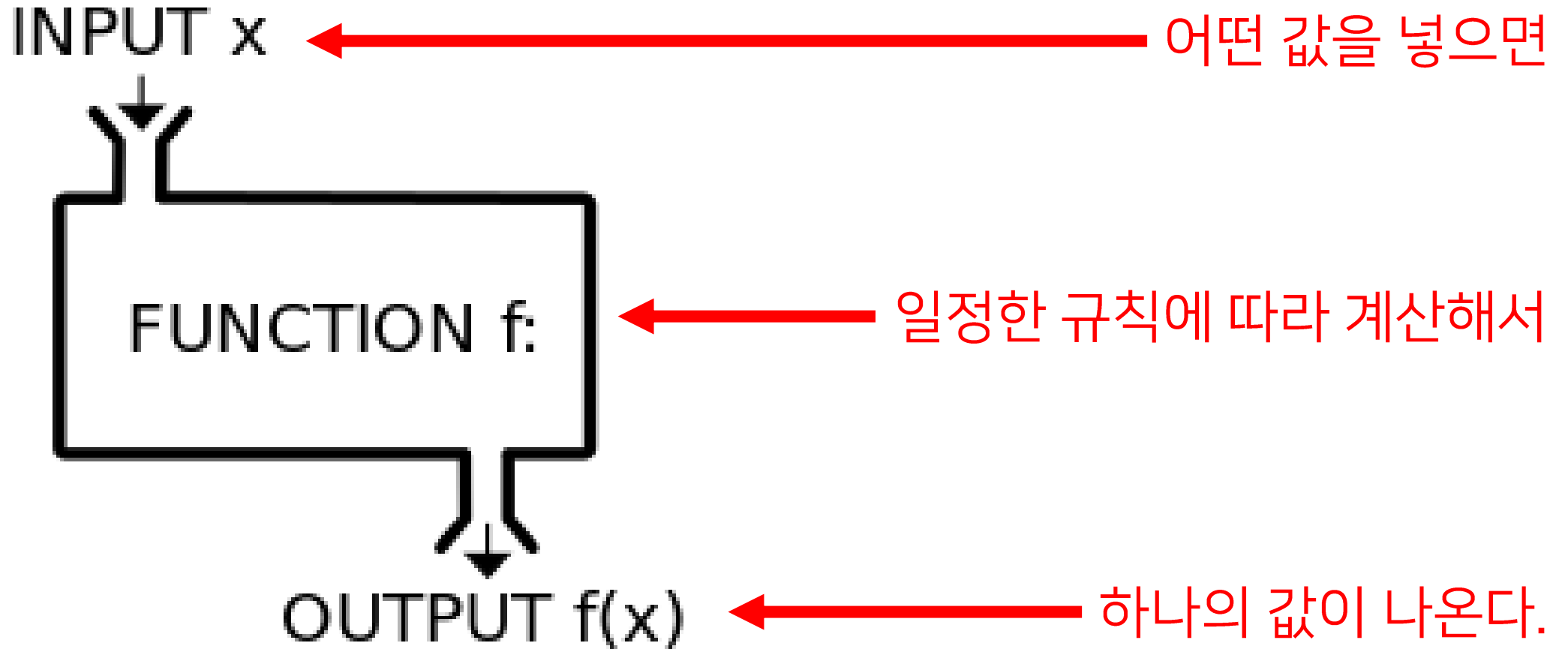
함수

function

함수란?

어떤 값을 넣으면
일정한 규칙에 따라 계산해서
하나의 값이 나온다.

함수란?



함수란?

$$f(x) = x^2 + 2x + 7$$

$$f(1) = 10$$

$$f(0) = 7$$

코틀린에서 함수란?

어떤 값을 넣으면
일정한 규칙에 따라 계산해서
하나의 값이 나온다.

하나의 기능을 만들어 준다.

- 문자열을 출력하는 기능
- 입력을 받는 기능
- 자료형을 변경하는 기능
- etc...

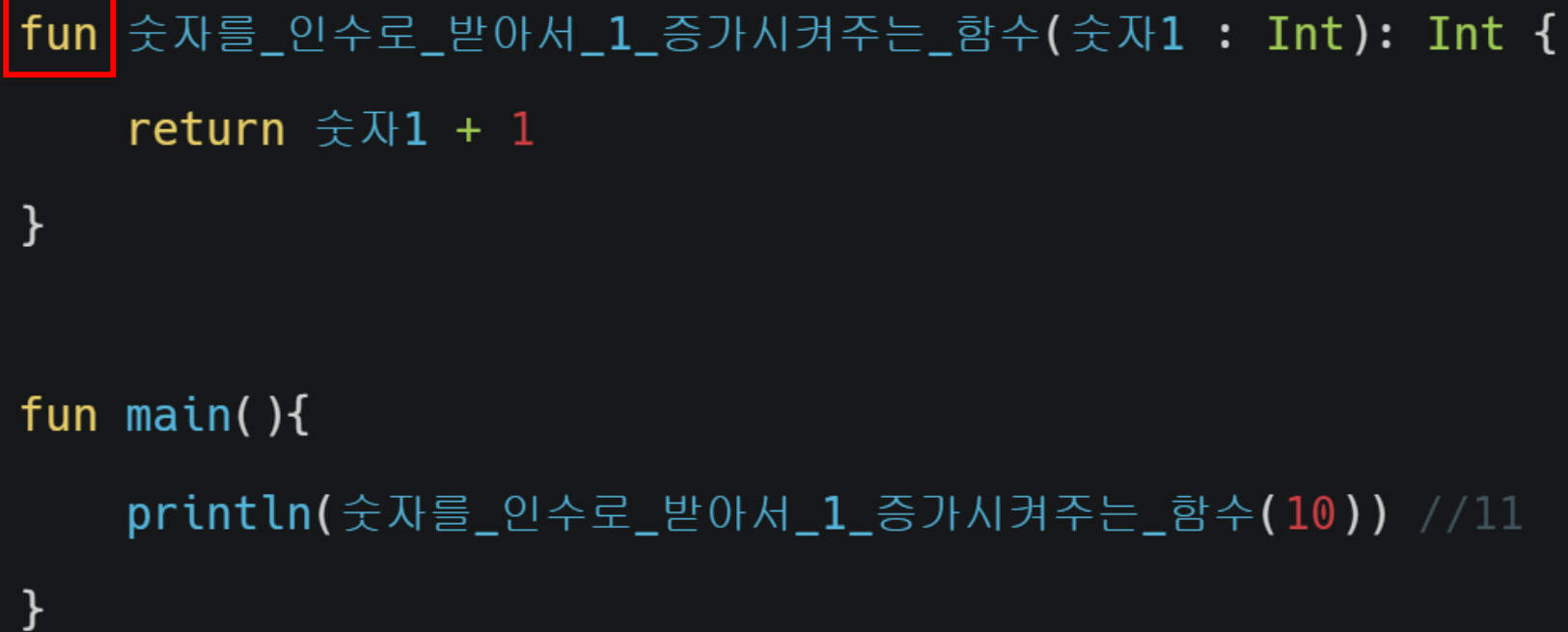
코틀린에서 함수란?



```
fun 숫자를_인수로_받아서_1_증가시켜주는_함수(숫자1 : Int): Int {  
    return 숫자1 + 1  
}
```

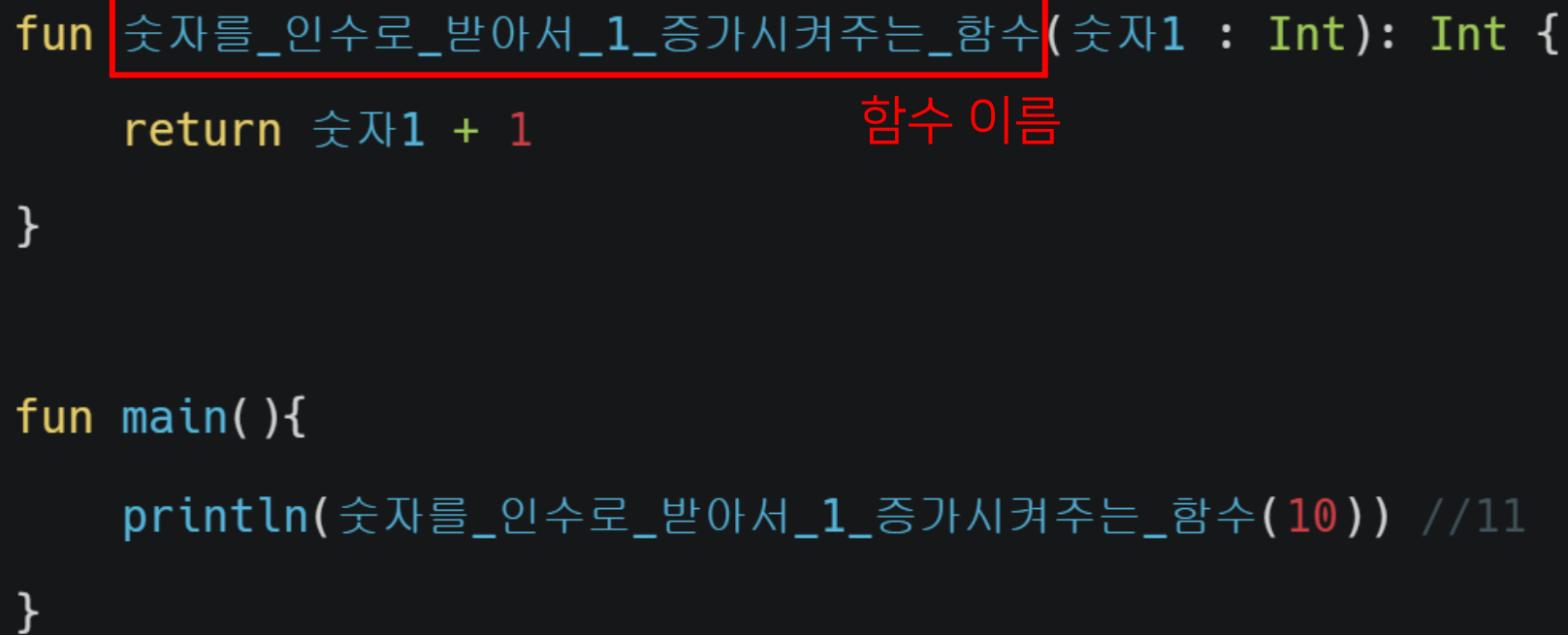
```
fun main(){  
    println(숫자를_인수로_받아서_1_증가시켜주는_함수(10)) //11  
}
```


코틀린에서 함수란?



```
fun 숫자를_인수로_받아서_1_증가시켜주는_함수(숫자1 : Int): Int {  
    return 숫자1 + 1  
}  
  
fun main(){  
    println(숫자를_인수로_받아서_1_증가시켜주는_함수(10)) //11  
}
```


코틀린에서 함수란?



```
fun 숫자를_인수로_받아서_1_증가시켜주는_함수( 숫자1 : Int): Int {  
    return 숫자1 + 1  
}  
  
fun main(){  
    println( 숫자를_인수로_받아서_1_증가시켜주는_함수(10)) //11  
}
```

함수 이름

코틀린에서 함수란?



```
fun 숫자를_인수로_받아서_1_증가시켜주는_함수(숫자1 : Int): Int {  
    return 숫자1 + 1  
}
```

함수에 들어오는 값
(= 파라미터 = 인자 = 인수 = 장인수)

```
fun main(){  
    println(숫자를_인수로_받아서_1_증가시켜주는_함수(10)) //11  
}
```


코틀린에서 함수란?

```
fun 숫자를_인수로_받아서_1_증가시켜주는_함수(숫자1 : Int): Int {  
    return 숫자1 + 1  
}  
  
fun main(){  
    println(숫자를_인수로_받아서_1_증가시켜주는_함수(10)) //11  
}
```

함수가 반환하는 값의 자료형

함수가 반환하는 값

코틀린에서 함수란?



```
fun 숫자를_인수로_받아서_1_증가시켜주는_함수( 숫자1 : Int): Int {  
    return 숫자1 + 1  
}
```

```
fun main(){  
    println(숫자를_인수로_받아서_1_증가시켜주는_함수(10)) //11  
}
```

함수 사용하기 (=함수를 호출한다.)

코틀린에서 함수란?

iceCreamMachine(🍓) // 🍦

iceCreamMachine(🍫) // 🍦

iceCreamMachine(🥛) // 🍦

코틀린에서 함수란?

$$f(x) = x^2 + 2x + 7$$

```
fun f(x : Int): Int {  
    return x * x + 2 * x + 7  
}  
  
fun main(){  
    println(f(1)) //10  
}
```


함수 내부의 변수



```
fun 대충_함수_이름() {  
    val x = 100  
    println(x)  
}
```

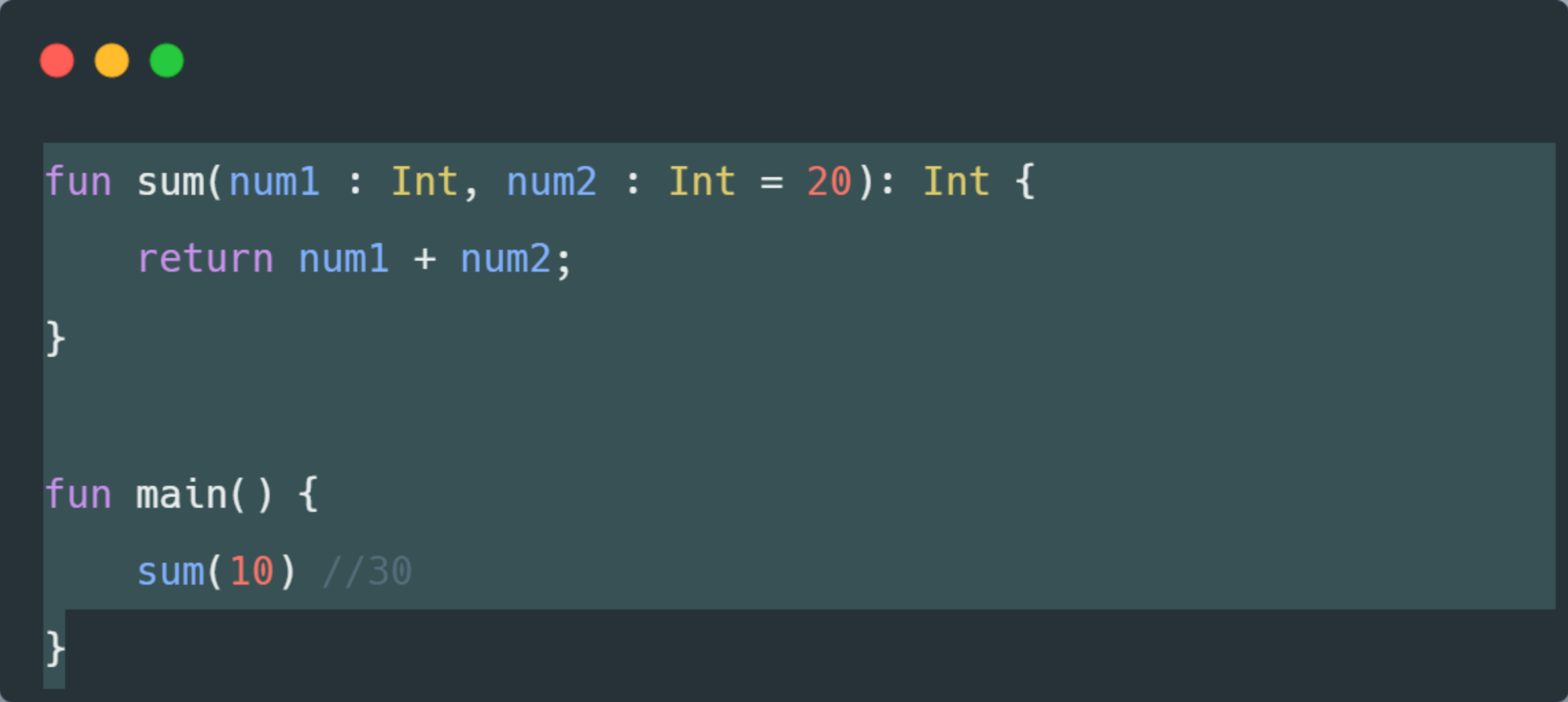
```
fun main(){  
    val x = 20  
  
    println(x)  
    대충_함수_이름()  
    println(x)  
}
```


함수 내부의 변수



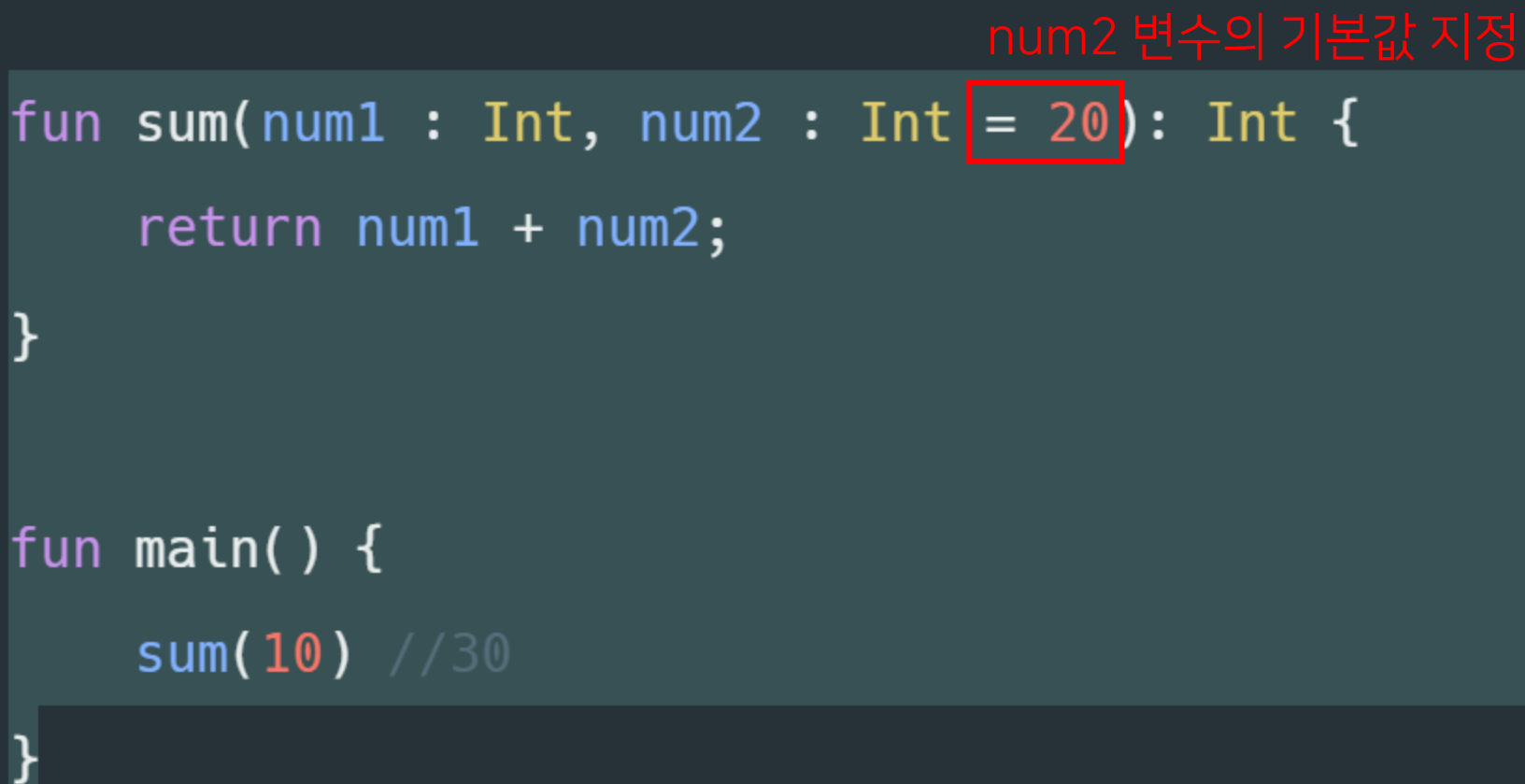
```
fun 대충_함수_이름() {  
    val x = 100  
    println(x) // 100  
}  
  
fun main(){  
    val x = 20  
  
    println(x) // 20  
    대충_함수_이름()  
    println(x) // 20  
}
```

함수 인수의 기본값



```
fun sum(num1 : Int, num2 : Int = 20): Int {  
    return num1 + num2;  
}  
  
fun main() {  
    sum(10) //30  
}
```

함수 인수의 기본값



num2 변수의 기본값 지정

```
fun sum(num1 : Int, num2 : Int = 20): Int {  
    return num1 + num2;  
}  
  
fun main() {  
    sum(10) //30  
}
```

코틀린에서 함수 예제

- 두 수를 인수로 받아서 더 큰 값을 반환하는 함수
- 이름을 인수로 받아서 "안녕하세요. {이름}"을 출력하는 함수
- 두 수를 인수로 받아서 더한 값을 반환하는 함수


람다

lamda

람다

- 람다 : 함수를 더 쉽게 표현해서 변수에 저장한 것


람다



```
fun sum(x: Int, y: Int) {  
    println("$x + $y")  
    return x + y  
}
```

함수로 표현

람다로 표현



```
var sum = { x: Int, y: Int ->  
    println("$x + $y")  
    x + y  
}
```

```
sum(10, 20) // 20
```

람다



```
var sum = { 인수들  
    x: Int, y: Int ->  
    println("$x + $y")  
    x + y 반환 값  
}
```

```
sum(10, 20) // 20 람다의 사용
```


람다를 사용하는 이유

- 편리하게 함수를 만들고, 조작할 수 있다.
- 함수의 인자로 함수를 넘겨줄 수 있다. (= 콜백 함수)

시간이 걸리는 작업이 끝난 뒤 어떤걸 수행해라...

람다를 사용하는 이유



```
fun highOrder(인자로_받은_람다: (Int, Int) -> Int, num1: Int, num2: Int): Int {  
    return 인자로_받은_람다(num1, num2)  
}
```

```
fun main() {  
  
    var result : Int = highOrder({ x, y -> x + y}, 10, 20)  
    println(result)  
}
```

람다를 사용하는 이유



람다식을 인자로 받을 때는 자료형을
람다식이 인자로 받는 값, 반환하는 값의 자료형을 적어준다.

```
fun highOrder(인자로_받은_람다: (Int, Int) -> Int, num1: Int, num2: Int): Int {  
    return 인자로_받은_람다(num1, num2) "Int형과 Int형을 받아서 Int형을 반환한다" 라는 뜻  
}
```

```
fun main() {  
  
    var result : Int = highOrder({ x, y -> x + y}, 10, 20)  
    println(result)  
}
```