Android Development

Kotlin

- Expressive and concise
- Safer code
- Interoperable
- Structured Concurrency

Basic Types

Char

Boolean

String

Collection

Any

Unit

Nothing

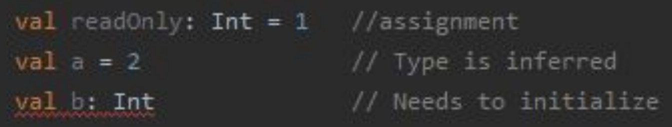
Number

Type	**Size(bits)**
Double	64
Float	32
Long	64
Int	32
Short	16
Byte	8

Kotlin Basics

Variables

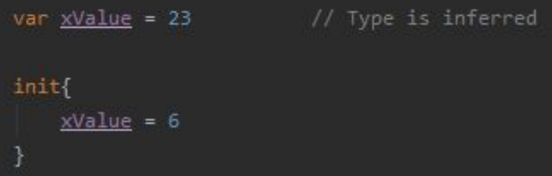
Read-only local variables are defined using the keyword val



```
val readOnly: Int = 1   //assignment
val a = 2               // Type is inferred
val b: Int              // Needs to initialize
```

Variables that can be reassigned use the var keyword.

```
var count: Int = 10
```



```
var xValue = 23          // Type is inferred

init{
    xValue = 6
}
```

Null safety

Kotlin variables can't hold null values by default. This means that the following snippet is invalid

```
// Fails to compile
val languageName: String = null
```

For a variable to hold a null value, it must be of a *nullable* type. You can specify a variable as being nullable by suffixing its type with `?`

```
val languageName: String? = null
```

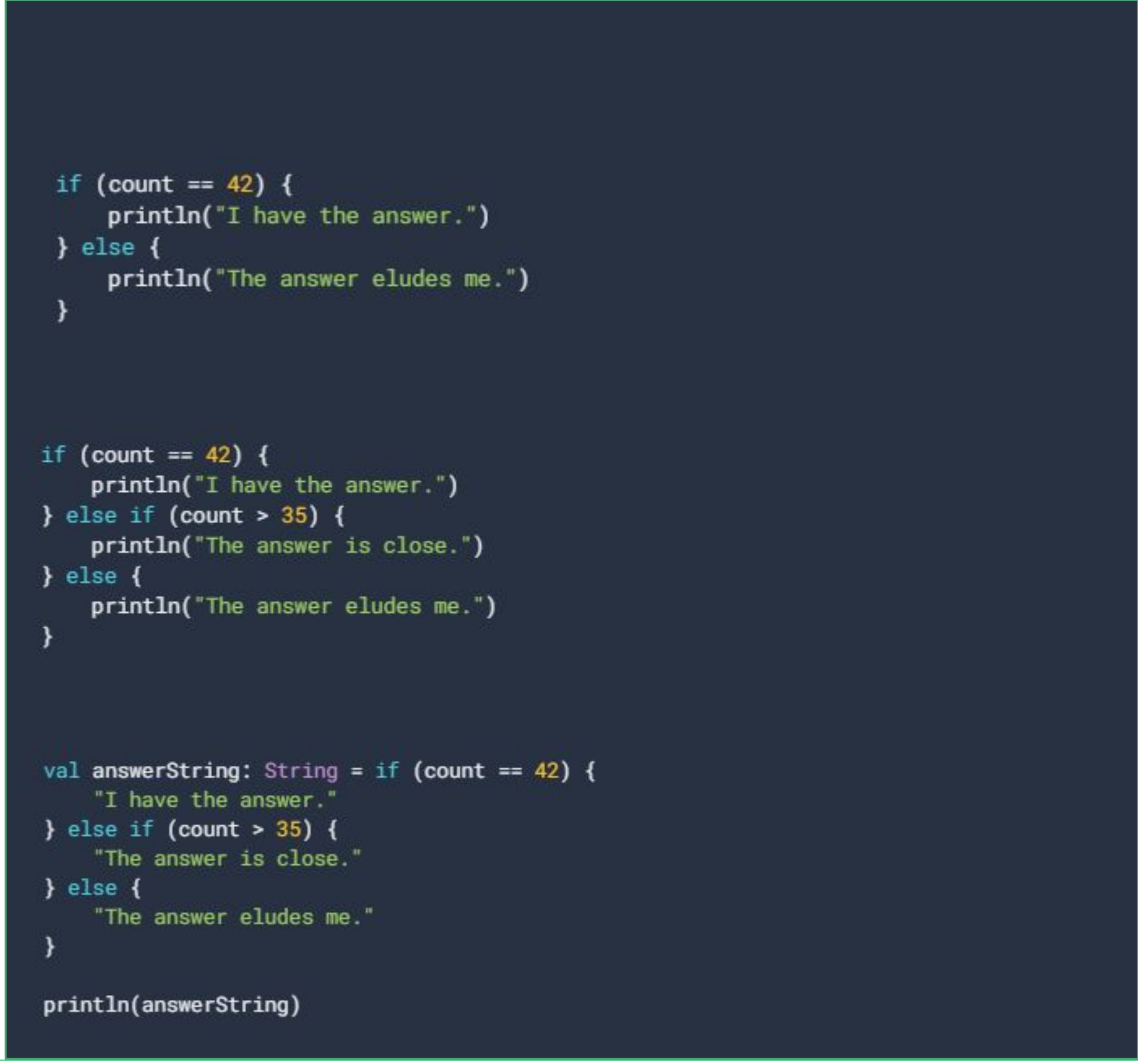
```
val languageName: String? = null
if (languageName != null) {
    // No need to write languageName?.toUpperCase()
    println(languageName.toUpperCase())
}
```

Conditionals

There is no ternary operator

```
condition ? then : else
```

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



```
if (count == 42) {
    println("I have the answer.")
} else {
    println("The answer eludes me.")
}
```

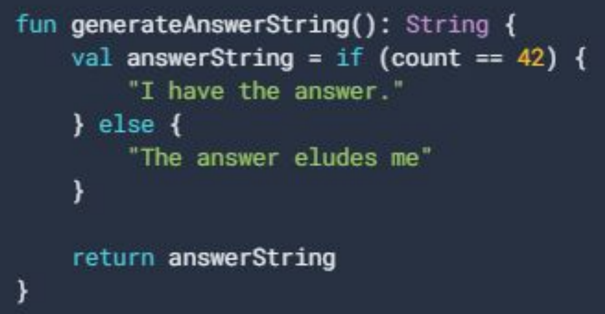
```
if (count == 42) {
    println("I have the answer.")
} else if (count > 35) {
    println("The answer is close.")
} else {
    println("The answer eludes me.")
}
```

```
val answerString: String = if (count == 42) {
    "I have the answer."
} else if (count > 35) {
    "The answer is close."
} else {
    "The answer eludes me."
}

println(answerString)
```

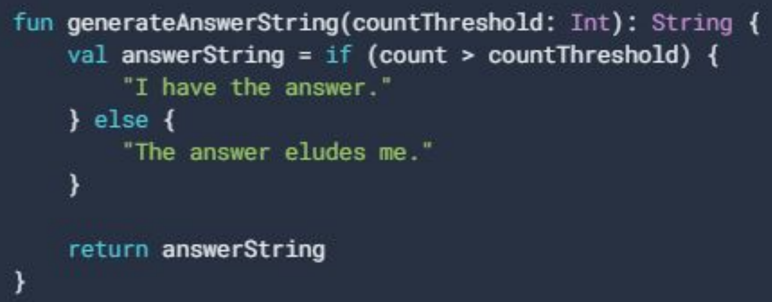
Kotlin Basics

Functions



```kotlin
fun generateAnswerString(): String {
    val answerString = if (count == 42) {
        "I have the answer."
    } else {
        "The answer eludes me"
    }

    return answerString
}
```

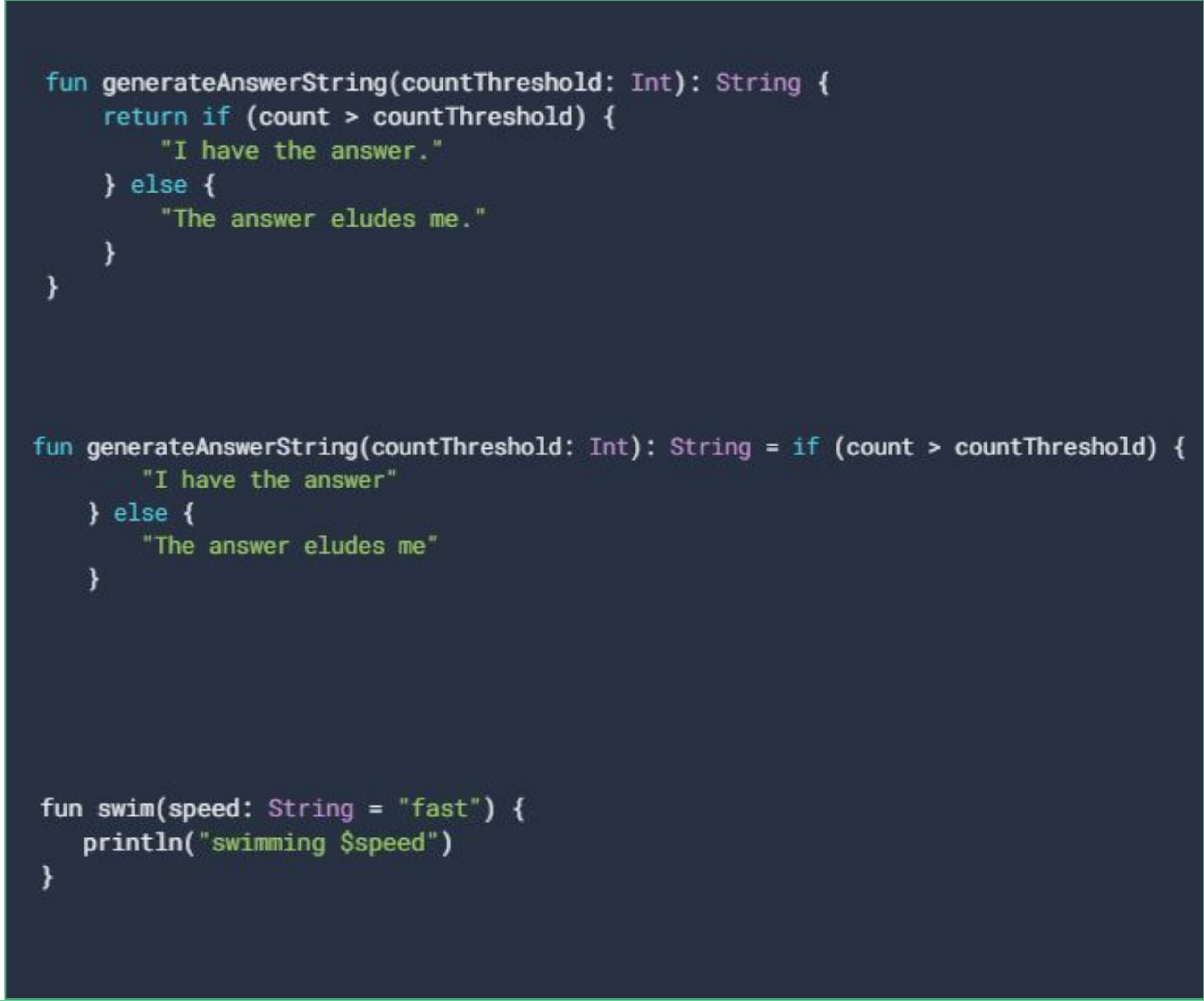


```kotlin
fun generateAnswerString(countThreshold: Int): String {
    val answerString = if (count > countThreshold) {
        "I have the answer."
    } else {
        "The answer eludes me."
    }

    return answerString
}
```

Kotlin Basics

Simplifying function declarations



```
fun generateAnswerString(countThreshold: Int): String {
    return if (count > countThreshold) {
        "I have the answer."
    } else {
        "The answer eludes me."
    }
}
```

```
fun generateAnswerString(countThreshold: Int): String = if (count > countThreshold) {
        "I have the answer"
    } else {
        "The answer eludes me"
    }
```

```
fun swim(speed: String = "fast") {
   println("swimming $speed")
}
```

Functions

Default Parameters

Named Arguments