



Kotlin

Kotlin : Variable, Opérateur, Condition

- 📎 Déclaration de variable
- 📎 Différents types de variable
- 📎 Opérateur
- 📎 Condition
- 📎 Boucle

Variable, Opérateur, Condition

Déclaration de variable

```
var mVariable = 10  
val mConstant = 10 // ♥ Préférence pour l'Immuable
```

i var comme variable

Variable, Opérateur, Condition

Différents types de variable

String
Boolean
Char
Array

 Les classiques

Variable, Opérateur, Condition

Exemples de déclaration

```
val mHello: String = "Hello you"  
val s = "abc"  
val str = "$s length is ${s.length}" // evaluates to "abc length is 3"
```

i interpolation de *String*

Variable, Opérateur, Condition

Type Unit

```
val mObject: Unit = Unit // Objet vide
```

i ≈ void

Variable, Opérateur, Condition

Différents types

List

MutableList

ArrayList

IntArray

 Les collections

Variable, Opérateur, Condition

Différents types : Tableau

```
var mArray = Array(6, { i: Int -> MagicCircle(i, i) })
```


Variable, Opérateur, Condition

Différents types : Tableau

```
var mArray = arrayOf("Jake", "Jill", "Ashley", "Bill")
```

Variable, Opérateur, Condition

Différents types : Liste

```
var mutableArray: MutableList<MagicCircle> = ArrayList()
```

Variable, Opérateur, Condition

Différents types

Double (64)

Float (32)

Long (64)

Int (32)

Short (16)

Byte (8)

 Les nombres

Variable, Opérateur, Condition

Exemple de déclaration

i déclaration type optionnelle

```
val mX = 20 // Conversion implicite en Int  
val mZ = 10F // Conversion implicite en Float  
val mY: Float = mX.toFloat() // Conversion explicite en Float
```

i conversion de nombres

Variable, Opérateur, Condition

Différents types : correspondances avec Java

Double

double

Float

float

Int

int

Boolean

boolean

Double?

java.lang.Double

Float?

java.lang.Float

Int?

java.lang.Integer

String?

String

Boolean?

java.lang.Boolean

IntArray

int[]

Array<Int>

Integer[]



Variable, Opérateur, Condition

Opérateur

 Unaire

 Binaire

 Tableau

 Égalité

Variable, Opérateur, Condition

Opérateur

Unary prefix operators

Expression	Translated to
<code>+a</code>	<code>a.unaryPlus()</code>
<code>-a</code>	<code>a.unaryMinus()</code>
<code>!a</code>	<code>a.not()</code>

 Unaire

Variable, Opérateur, Condition

Opérateur

Increments and decrements

Expression	Translated to
<code>a++</code>	<code>a.inc()</code> + see below
<code>a--</code>	<code>a.dec()</code> + see below

 Unaire

Variable, Opérateur, Condition

Opérateur

Arithmetic operators

Expression	Translated to
<code>a + b</code>	<code>a.plus(b)</code>
<code>a - b</code>	<code>a.minus(b)</code>
<code>a * b</code>	<code>a.times(b)</code>
<code>a / b</code>	<code>a.div(b)</code>
<code>a % b</code>	<code>a.rem(b)</code> , <code>a.mod(b)</code> (deprecated)
<code>a..b</code>	<code>a.rangeTo(b)</code>

 Binaire

Variable, Opérateur, Condition

Opérateur

```
if (a >= 0 && a <= 10)
```

≈

```
if(a in 0..10 )
```

 Binaire

 inclusif !

Variable, Opérateur, Condition

Opérateur

'In' operator

Expression	Translated to
<code>a in b</code>	<code>b.contains(a)</code>
<code>a !in b</code>	<code>!b.contains(a)</code>

 Binaire

Variable, Opérateur, Condition

Opérateur

Augmented assignments

Expression	Translated to
<code>a += b</code>	<code>a.plusAssign(b)</code>
<code>a -= b</code>	<code>a.minusAssign(b)</code>
<code>a *= b</code>	<code>a.timesAssign(b)</code>
<code>a /= b</code>	<code>a.divAssign(b)</code>
<code>a %= b</code>	<code>a.modAssign(b)</code>

 Binaire

Variable, Opérateur, Condition

Opérateur

Indexed access operator

Expression	Translated to
<code>a[i]</code>	<code>a.get(i)</code>
<code>a[i, j]</code>	<code>a.get(i, j)</code>
<code>a[i_1, ..., i_n]</code>	<code>a.get(i_1, ..., i_n)</code>
<code>a[i] = b</code>	<code>a.set(i, b)</code>
<code>a[i, j] = b</code>	<code>a.set(i, j, b)</code>
<code>a[i_1, ..., i_n] = b</code>	<code>a.set(i_1, ..., i_n, b)</code>

 Tableau

Variable, Opérateur, Condition

Opérateur

Equality and inequality operators

Expression	Translated to
<code>a == b</code>	<code>a?.equals(b) ?: (b === null)</code>
<code>a != b</code>	<code>!(a?.equals(b) ?: (b === null))</code>

 Égalité

Variable, Opérateur, Condition

Opérateur

Comparison operators

Expression	Translated to
<code>a > b</code>	<code>a.compareTo(b) > 0</code>
<code>a < b</code>	<code>a.compareTo(b) < 0</code>
<code>a >= b</code>	<code>a.compareTo(b) >= 0</code>
<code>a <= b</code>	<code>a.compareTo(b) <= 0</code>

 Comparaison

Variable, Opérateur, Condition

Opérateur

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    var tv = findViewById(R.id.mainTextView) as TextView  
}
```

 Conversion

Variable, Opérateur, Condition

Condition

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    when (item.itemId) {  
        R.id.action_settings -> {  
            showAlert()  
            return true  
        }  
        R.id.action_info -> {  
            showInfo()  
            return true  
        }  
        else -> return super.onOptionsItemSelected(item)  
    }  
}
```

Variable, Opérateur, Condition

Condition

```
when (view) {  
    is TextView -> toast(view.text)  
    is RecyclerView -> toast("Item count : ${view.adapter.itemCount}")  
    is SearchView -> toast("Current query: ${view.query}")  
    else -> toast("View type not supported")  
}
```

 exclusif !

Variable, Opérateur, Condition

Condition : *range*

```
val res = when {  
  x in 1..10 -> "cheap"  
  s.contains("hello") -> "it's a welcome!"  
  v is ViewGroup -> "child count: ${v.getChildCount()}"  
  else -> ""  
}
```

i variable dans un rang

Variable, Opérateur, Condition

Condition : le classique

```
if (dx < 0)
    cx = 0F
else
    cx = maxX.toFloat()
```

Variable, Opérateur, Condition

Condition : Elvis

`s2?.length ?: 0`

Variable, Opérateur, Condition

Boucle

```
val names = arrayOf("Jake", "Jill", "Ashley", "Bill")  
for (name in names) {  
    toast(name)  
}
```

Kotlin : Variable, Opérateur, Condition

</> Fonctionnelle :

📎 Déclaration de variable

📎 Type de base

📎 Opérateur **range**

📎 Condition **if** ou **when**

📎 Collection : **list** ou **array**

📎 Boucle

THE EXPERT
AT ANYTHING
WAS ONCE
A BEGINNER.

_ Helen Hayes

Références

Kotlin for Android

- [TRY Kotlin](#)
- [Kotlin Workshop on Github: Slides and Questions](#)
- <https://antonioleiva.com/free-kotlin-android-course/>
- [ChillCoding.com : Introduction à Kotlin](#)
- [ChillCoding.com : Configurer Kotlin dans un projet Android Studio](#)

Library

- [ChillCoding.com : Utiliser des bibliothèques graphiques Kotlin dans un projet Android](#)

Fonction d'extension

- [Odelia Technologies : Les fonctions d'extension de Kotlin](#)

Kotlin in videos

- [Jake Wharton and Kotlin \(DEC 2015\)](#)
- [Tue Dao & Christina Lee on The Road to Kotlin town \(KotlinConf 2017\)](#)
- [Introduction to Kotlin Google I/O '17](#)

Références

Why Kotlin?

- [Swift is like Kotlin](#)
- [Langage Java](#)
- [API Java : Google a enfreint les brevets d'Oracle, selon la Cour Suprême](#)
- [Antoniroleiva: 12 reasons to strat Kotlin for Android](#)

Kotlin in brief

- [Kotlin: pourquoi ce nouveau langage est une bonne nouvelle](#)

Android and Kotlin

- [Kotlin - Official Site](#)
- [developer.android: Get Started with Kotlin on Android](#)

Type Kotlin

- <https://code.tutsplus.com/tutorials/kotlin-from-scratch-variables-basic-types-arrays-type-inference-and-comments--cms-29328>
- <https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-array/index.html>
- <http://kotlinlang.org/docs/reference/basic-types.html#arrays>

-

try.kotlinlang.org

The screenshot shows the try.kotlinlang.org website in a web browser. The browser's address bar displays the URL. The website has a dark header with the Kotlin logo and navigation links for LEARN, COMMUNITY, and TRY ONLINE. Below the header, there are social media icons (Facebook, Google+, GitHub, JetBrains) and utility buttons (Shortcuts, Convert from Java, Fullscreen). The main content area is titled 'Kotlin Koans' and shows a breadcrumb trail: Introduction > Hello, world! > Task.kt. A sidebar on the left lists various Kotlin topics, with 'Hello, world!' and 'Task.kt' highlighted. The main panel displays the 'Simple Functions' task, which instructs the user to implement a function named 'start' that returns the string 'OK'. The code editor shows the function signature: `fun start() = "OK"`. Below the code, there are buttons for 'Check', 'Revert', and 'Show answer'.

try.kotlinlang.org

Kotlin

LEARN COMMUNITY TRY ONLINE

f G+ GitHub JB

Shortcuts Convert from Java Fullscreen

Kotlin Koans > Introduction > Hello, world! > Task.kt

Examples

Kotlin Koans 2/42

Introduction

Hello, world!

Task.kt

Test.kt

Java to Kotlin conversions

Named arguments

Default arguments

Lambdas

Strings

Data classes

Nullable types

Smart casts

Extension functions

Object expressions

SAM conversions

Extensions on collections

Save Save as Arguments JUnit Run

Program arguments

Simple Functions

Take a look at [function syntax](#) and make the function `start` return the string `"OK"`.

In the tasks the function `TODO()` is used that throws an exception. Your job during the koans will be to replace this function invocation with a meaningful code according to the problem.

Check Revert Show answer

```
1 fun start() = "OK"
```