



Kotlin

Kotlin : Classe, Propriété, Fonction

📎 Déclaration de classe, constructeurs

📎 Propriété mutable ou immuable

📎 Visibilité des membres

📎 Fonction

📎 Qualificatifs

📎 Héritage et interface

Classe, Propriété, Fonction

Déclaration :   MagicCircle

```
package macha.agua

class MagicCircle {
    var cx: Int = 0
    var cy: Int = 0
}
```

</> Création d'une classe représentant un cercle

Classe, Propriété, Fonction

Déclaration :   MagicCircle

```
package macha.agua

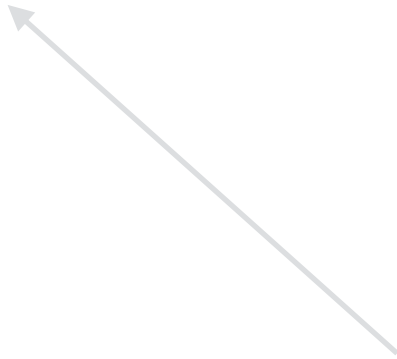
class MagicCircle(cx: Int = 10, cy: Int = 10) {
    var cx: Int = cx
    var cy: Int = cy
}
```

</> Création d'un constructeur

Classe, Propriété, Fonction

Déclaration : constructeur primaire

```
class MagicCircle(var cx: Int = 10, var cy: Int = 10)
```



i déclaration des propriétés dans le constructeur

Classe, Propriété, Fonction

Déclaration : constructeur secondaire

```
class MagicCircle {  
    var cx = 0  
    var cy = 0  
  
    constructor(cx:Int, cy:Int) {  
        this.cx = cx  
        this.cy = cy  
    }  
}
```

i explicite
dans la classe

Classe, Propriété, Fonction

Data class

```
data class MagicCircle(var cx: Int = 10, var cy: Int = 10)
```

 gérer des données

💡 equals

💡 hashCode

💡 toString



Classe, Propriété, Fonction

Déclaration de propriété **mutable** ou **immutable**

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

i var comme variable

Classe, Propriété, Fonction

Propriété

```
public class MagicCircle {  
    private final int cx;  
    private final int cy;  
  
    public MagicCircle(int cx, int cy) {  
        this.cx = cx;  
        this.cy = cy;  
    }  
  
    public int getCx() {  
        return cx;  
    }  
  
    public int getCy() {  
        return cy;  
    }  
}
```

`magic.getCx()`



```
class MagicCircle(  
    var cx: Int,  
    var cy: Int  
)
```

`magic.cx`



Classe, Propriété, Fonction

Exemple de variable d'instance

```
val magic = MagicCircle(10, 10)
```

```
magic.cx = 26
```

```
val magic0 = MagicCircle()
```

```
val magicx = MagicCircle(cx = 10)
```

```
val magicy = MagicCircle(cy = 10)
```

i *Ø new*

i *getters/setters implicite*

Classe, Propriété, Fonction

Visibilité des membres

public	visible au dehors de la classe
private	seulement visible à l'intérieur de la classe
protected	visible également par les sous classes
internal	visible dans le module

i module : ensemble de classe
ou fichier compiler ensemble

Classe, Propriété, Fonction

Fonction membre à expression simple

```
data class MagicCircle(var cx: Int, var cy: Int) {  
    fun move() = cx += 1  
}
```

i expression simple

Classe, Propriété, Fonction

Fonction : paramètre

```
data class MagicCircle(val maxX: Int, val maxY: Int) {  
  
    var cx = 0  
    var cy = 0  
  
    fun move(delta: Int) {  
        cx += delta  
        cy += delta  
    }  
}
```

Classe, Propriété, Fonction

Fonction : type de retour

```
data class MagicCircle(val maxX: Int, val maxY: Int) {  
  
    ...  
  
    fun move(delta: Int): Boolean {  
        cx += delta  
        cy += delta  
        return true  
    }  
}
```

Classe, Propriété, Fonction

Fonction : argument par défaut

```
data class MagicCircle(val maxX: Int, val maxY: Int) {
```

```
    ...
```

```
    fun move(delta: Int = 5) {
```

```
        cx += delta
```

```
        cy += delta
```

```
    }
```

```
}
```

 argument par défaut

Classe, Propriété, Fonction

Fonction

```
data class MagicCircle(val maxX: Int, val maxY: Int) {  
  
    ...  
  
    fun move(dx: Int = 5, dy: Int = 5) {  
        cx += dx  
        cy += dy  
    }  
}
```

i plusieurs arguments

Classe, Propriété, Fonction

Qualificatifs

data

enum

inner

sealed

nested

❗ open, abstract,
final par défaut

❗ private, internal,
public par défaut

Classe, Propriété, Fonction

Héritage

```
class MainActivity : AppCompatActivity()
```

i extends -> :

Classe, Propriété, Fonction

Interface

```
class CustomView : View(), SensorEventListener
```

❶ implements -> :

Kotlin : Classe, Propriété, Fonction

</> POO :

📎 Déclaration de classe,

📎 Constructeurs, attributs

📎 Méthode et fonction

📎 Héritage et interface

📎 Classe Abstraite

📎 Data, Enum, Nested, Sealed class

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'.

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"
```