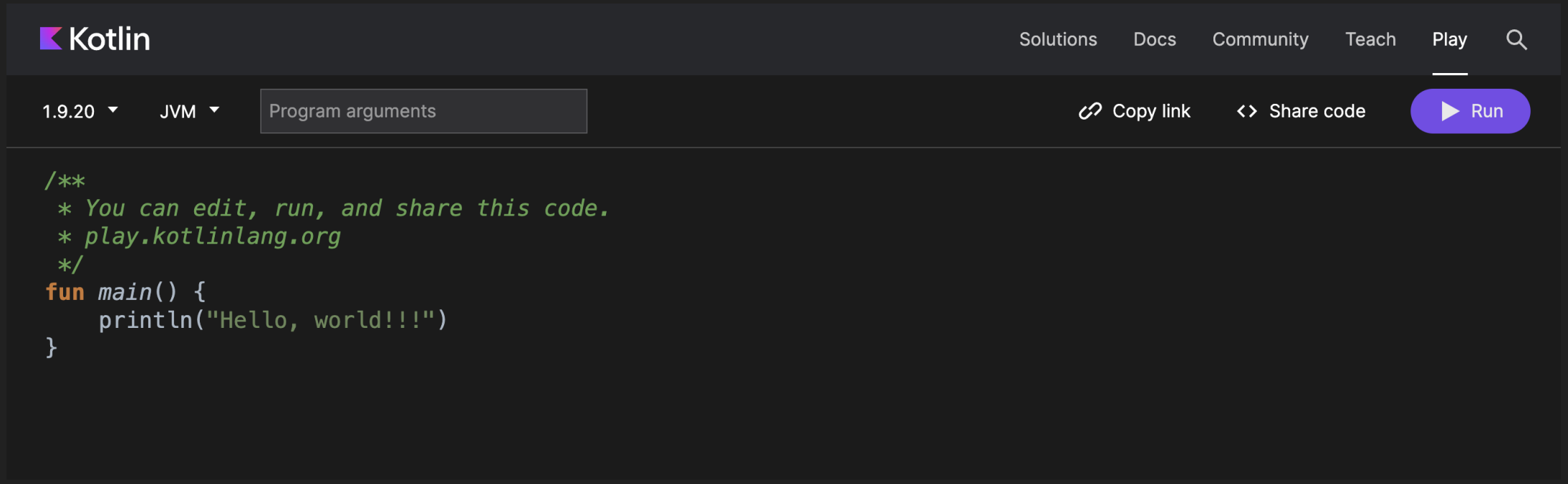
1 - Les bases de Kotlin

Introduction

- Développé par JetBrains, créateur d'IntelliJ
- Langage open-source recommandé par Google pour Android
- Se base sur Java
- Plus concis et robuste

Outil – Kotlin Playground

- Utilisation de Kotlin Playground https://play.kotlinlang.org/



Premier programme Kotlin

```
fun main() {
    println("Hello, world!")
}
```

Que fait ce programme à votre avis ?

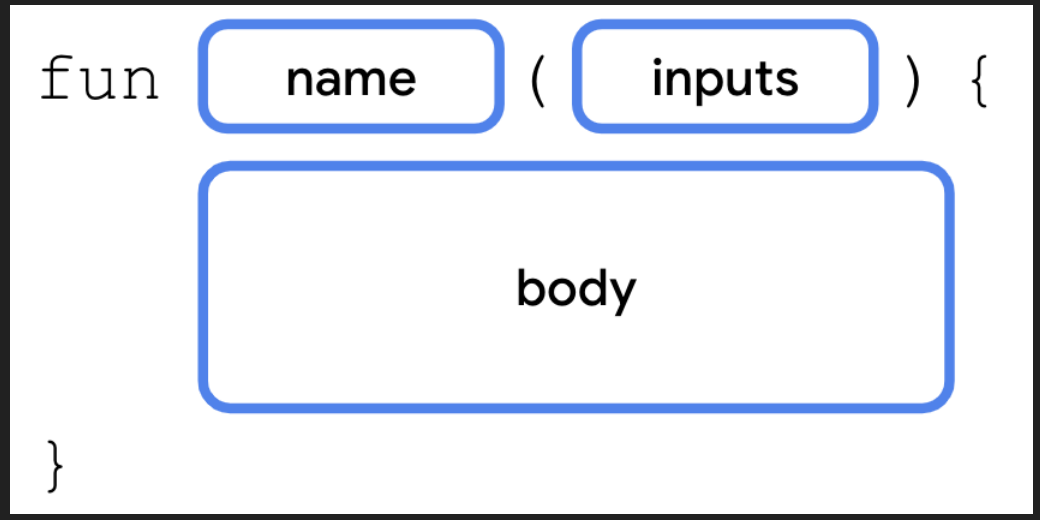
Premier programme Kotlin

- Exécutons ce programme

```
Hello, world!
```

Définition d'une fonction

- **fun** mot clé pour les fonctions
- **main** est le nom de la fonction principale : point d'entrée du programme
- Les **paramètres** entre parenthèses
- Le **corps** de la fonction entre accolades



Petit exercice

- Comment afficher deux fois « Hello, world! » ?

Plusieurs instructions

- Pas besoin de ; si une instruction par ligne

```
fun main() {
    println("Hello, world!")
    println("Hello, world!")
}
```

- Ce code est aussi valable :

```
fun main() {
    println("Hello, world!");println("Hello, world!")
}
```

Sortie des deux fonctions :

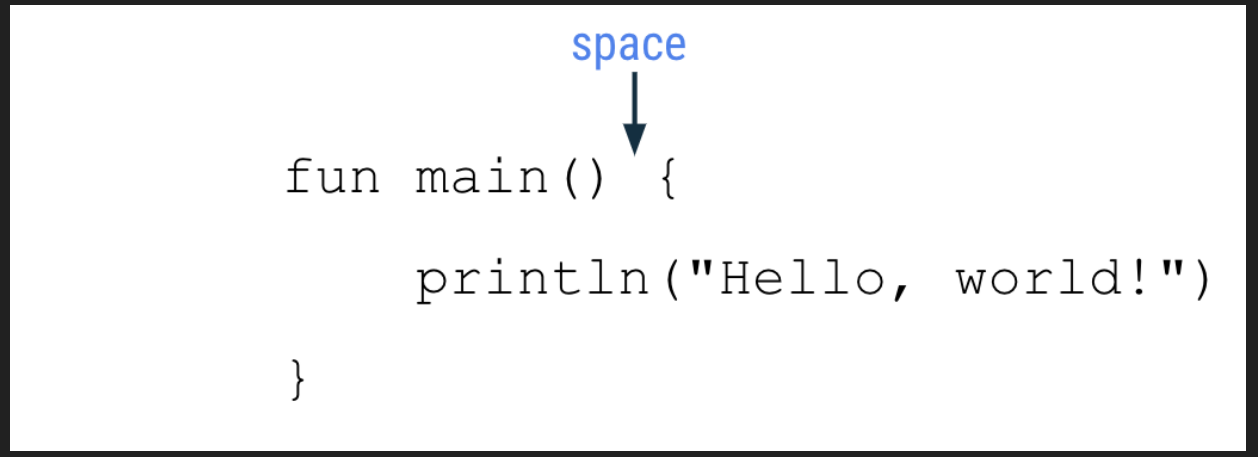
```
Hello, world!
Hello, world!
```

Guide de style

- Pour quoi ?
  - Harmoniser le code entre développeur
  - Plus facile à lire
  - Collaboration plus simple

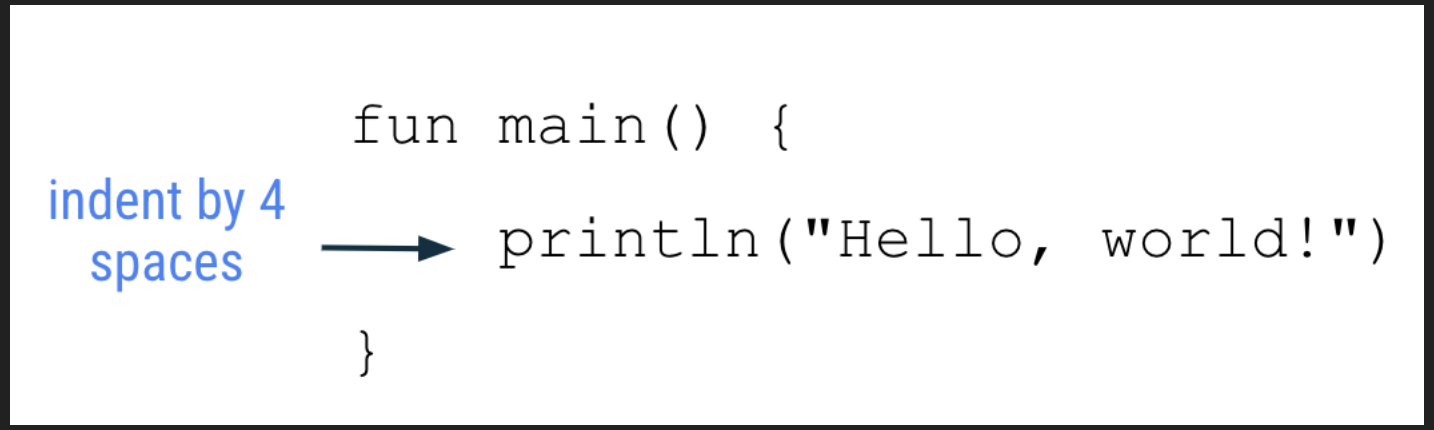
Guide de style

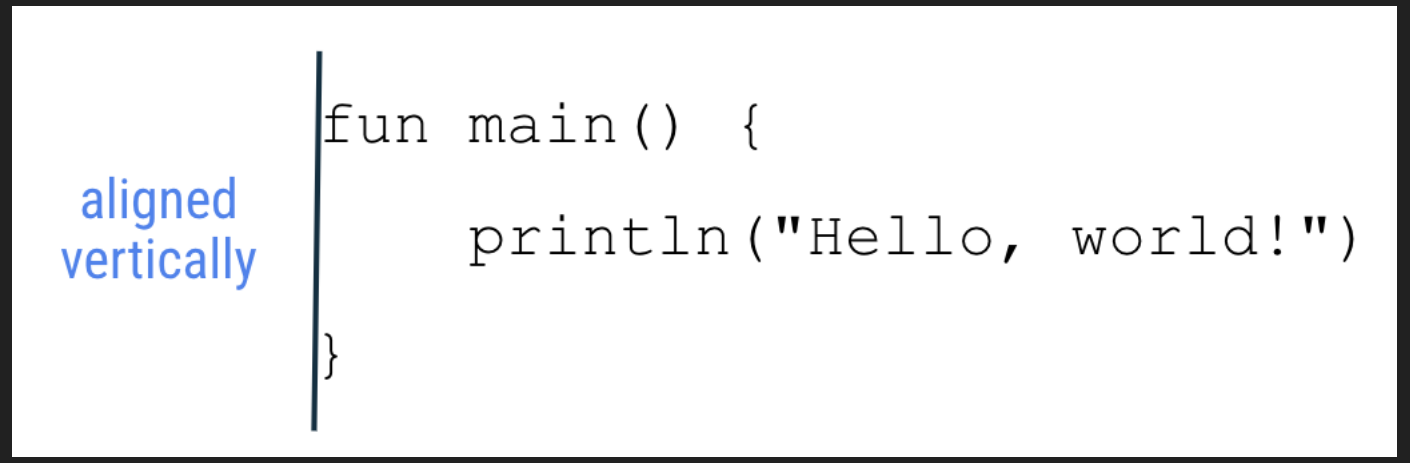
- Les noms de fonctions doivent être en camel case et être des verbes ou des expressions verbales. (getTemperature, fetchTimetables())
- Chaque instruction doit figurer sur une ligne distincte.
- L'accolade ouvrante doit apparaître à la fin de la ligne où la fonction commence.
- Il doit y avoir une espace avant l'accolade ouvrante.



Guide de style

- Le corps de la fonction doit être en retrait de 4 espaces. N'utilisez pas de tabulation pour mettre votre code en retrait. Saisissez quatre espaces.
- L'accolade fermante se trouve sur sa propre ligne après la dernière ligne de code dans le corps de la fonction. L'accolade doit être alignée sur le mot clé fun au début de la fonction.





Trouver les problèmes

```
fun main() {println("Bonjour à tous")}
```

```
fun main() (
    println("J'aime Kotlin <3")
)
```

Trouver les problèmes

Problème de style

```
fun main() {
    println("Bonjour à tous")
}
```

Problème de syntaxe

```
fun main() {
    println("J'aime Kotlin <3")
}
```

```
! Function 'main' must have a body
! Expecting a top level declaration
```

Les variables

- Quelles variables pouvons-nous reconnaitre sur cette capture d'écran ?

Les variables

- Boolean
- String
- Int
- Float/Double

Variables primitives

Type de données Kotlin	Types de données qu'il peut contenir	Exemples de valeurs littérales

`String`	Texte	`"Add contact"`
		`"Search"`
		`"Sign in"`

`Int`	Nombre entier	`32`
		`1293490`
		`-59281`

`Double`	Nombre décimal	`2.0`
		`501.0292`
		`-31723.99999`

`Float`	Nombre décimal (moins précis qu'un `Double`), suivi de `f` ou `F`.	`5.0f`
		`-1630.209f`
		`1.2940278F`

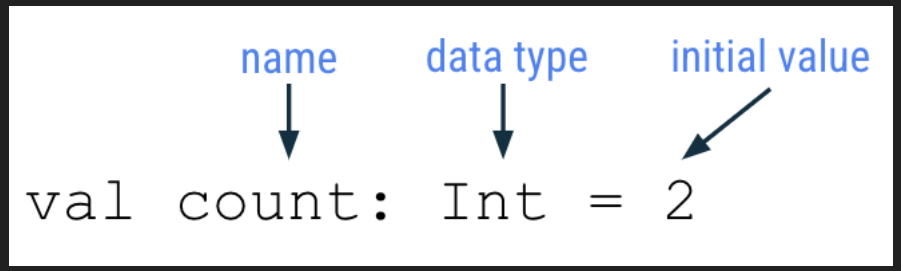
`Boolean`	`true` ou `false`. Utilisez ce type de données quand il n'y a que deux valeurs possibles. Notez que `true` et `false` sont des mots clés en Kotlin.	`true` `false`

Variables

expression	value
`count`	2

- Une variable est une expression qui s'évalue en une valeur

Déclarer une variable



Déclarer une variable

- Val mot clé pour déclarer une variable
- Le nom dit être au format camelCase et commencer par une minuscule. Pas d'espace.
- Type de donnée toujours avec une première lettre majuscule
- = opérateur d'attribution

Inférence de type


```
val count = 2
```

- Quand le compilateur peut déduire le type d'une donnée sans avoir à le préciser

- S'il n'y a pas de valeur initial, le type est obligatoire !

```
val count
```

-> Pas possible !

Réassigner une variable

```
fun main() {
    val heuresDeCours: Int = 4
    heuresDeCours = heuresDeCours + 4
    println(heuresDeCours)
}
```

```
Val cannot be reassigned
```

Réassigner une variable

- 2 mots clés
  - **val** indique que la valeur de la variable sera toujours la même, elle est en lecture seule, on ne peut pas la modifier. À utiliser dès que possible.
  - **var** indique une variable qui peut changer de valeur

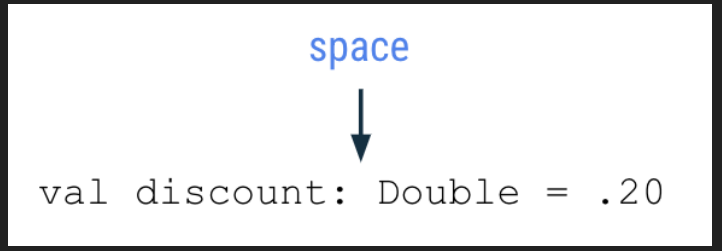
```
fun main() {
    var heuresDeCours: Int = 4
    heuresDeCours = heuresDeCours + 4
    println(heuresDeCours)
}
```

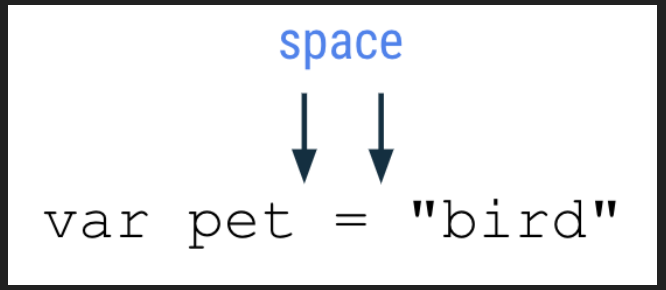
```
8
```

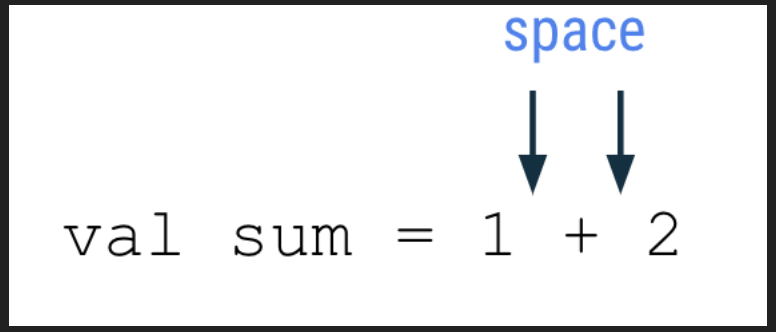
Guide de style

- Dans une déclaration de variable, vous devez insérer une espace après les deux-points (:) lorsque vous spécifiez le type de données.

- Il doit y avoir une espace avant et après un opérateur comme l'attribution (=), l'addition (+), la soustraction (-), la multiplication (*), la division (/) et plus encore.







Guide de style

- Lorsque vous écrivez des programmes plus complexes, la limite est fixée à 100 caractères par ligne. De cette façon, vous pouvez lire facilement l'ensemble du code d'un programme sur l'écran de votre ordinateur, sans avoir à faire défiler la page horizontalement.

Commentaires

- Simple //
- Multilignes
  /*
  */
- Commentaire de fonction (Javadoc)
  /**
  *
- */

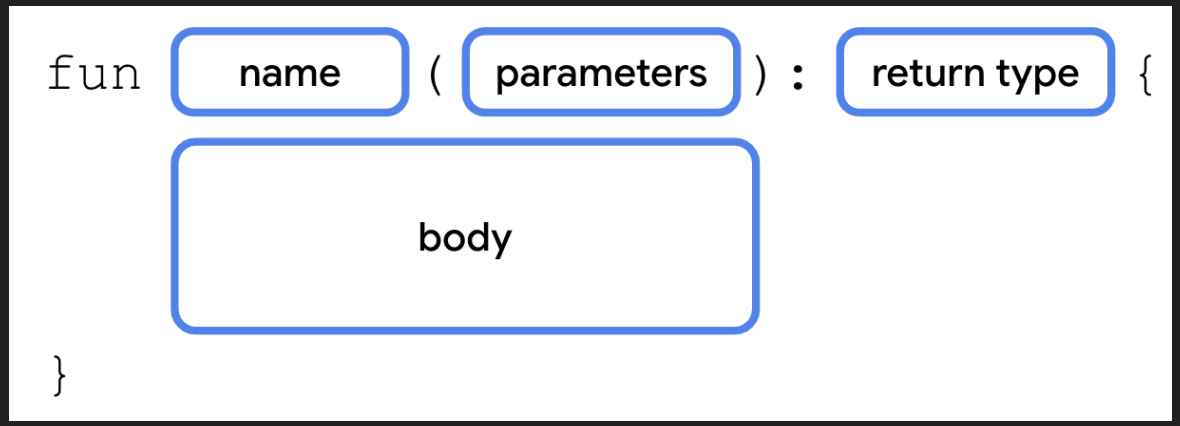
```
// Je suis un commentaire
```

```
/*
    Je suis un très long
    commentaire sur plusieurs
    lignes c'est pourquoi
    j'utilise un autre format.
*/
```

```
/**
 * C'est la fonction principale.
 */
fun main() {
    println("Je suis la fonction principale")
}
```

Déclaration d'une fonction

- Si pas de type de retour précisé, **Unit** est utilisé.
- **Unit** est le type qui signifie qu'il n'y a pas de valeur de retour.
- Si le type de retour est Unit, le **return** n'est pas nécessaire.



Déclaration d'une fonction - Paramètres

- Utiliser la syntaxe **nom: TypeDeDonnée** par exemple **nom: String.**
- Dans le corps, le paramètre est défini comme **val : il est immuable**

```
fun main() {
    afficherNombreDeChiens(3)
}


fun afficherNombreDeChiens(nombre: Int) {
    nombre = 1
    println(nombre)
}
```

```
Val cannot be reassigned
```

Déclaration d'une fonction - Paramètres

- Plusieurs paramètres sont séparés par des ,
- On peut spécifier l'argument en entrée en précisant le nom de celui-ci : c'est un **argument nommé.** Cela permet de changer l'ordre des arguments.
- On peut définir des arguments par défaut avec l'opérateur **=** suivi de la valeur par défaut. Cela permet d'avoir toujours une valeur en entrée même quand il est omis.

Déclaration d'une fonction - Paramètres

```
fun main() {
    afficherNombreDeChiens(nombre = 3)
}


fun afficherNombreDeChiens(nombre: Int, proprietaire: String = "Raoul") {
    println("$proprietaire a $nombre chien(s).")
}
```

```
Raoul a 3 chien(s).
```