Leçon 19
Les fonctions

Introduction

Imaginez une machine à café :

- Entrée : Vous mettez une capsule de café et de l'eau dans la machine.
- Processus : La machine prépare le café en chauffant l'eau et en la faisant passer à travers la capsule.
- Sortie : Vous obtenez une tasse de café prête à boire.

En programmation, une fonction fonctionne de la même manière. Vous lui donnez des entrées (les paramètres), elle exécute un processus (le bloc de code), et elle vous renvoie une sortie (le résultat).

Déclaration d'une Fonction

Voici comment on déclare une fonction :

```
fun nomDeLaFonction(param1: Type, param2: Type): TypeDeRetour {
    // Bloc de code à exécuter
    return valeurDeRetour
}
```

Pour créer une fonction en Kotlin, suivez ces étapes, comme assembler une recette :

1. **Commencez par le mot-clé fun** : Cela indique que vous allez définir une fonction. Pensez à fun comme à un panneau qui dit "Ici commence la recette !".
2. **Nommez votre fonction** : C'est comme donner un nom à votre recette. Par exemple, additionner.
3. **Définissez les paramètres entre parenthèses (param1: Type, param2: Type)** : Ce sont les ingrédients dont vous avez besoin pour votre recette. Vous pouvez avoir plusieurs ingrédients séparés par des virgules.
4. **Ajoutez le type de retour après un deux-points :** : Cela indique ce que votre recette produira à la fin. Si votre fonction renvoie un résultat, indiquez son type ici. Par exemple, Int pour un nombre entier. Si votre fonction ne renvoie rien, vous pouvez omettre cette partie ou utiliser Unit.

5. **Écrivez le code de la fonction entre accolades {}** : C'est le processus que vous suivez pour préparer le café. Le code à l'intérieur des accolades est ce que la fonction fera avec les ingrédients.
6. **Utilisez le mot-clé return si vous devez renvoyer un résultat** : C'est comme servir le café dans une tasse. Si votre fonction doit renvoyer un résultat, utilisez return pour donner ce résultat à l'extérieur de la fonction.

Exemple de Déclaration

Imaginons une fonction simple qui additionne deux nombres :

```
fun additionner(a: Int, b: Int): Int {
    return a + b
}
```

- fun : Indique que nous créons une fonction.
- additionner : Le nom de la fonction.
- a: Int, b: Int : Les paramètres que la fonction utilise.
- : Int : Le type de retour, ici Int (un nombre entier).
- return a + b : Le code de la fonction, qui additionne a et b et renvoie le résultat.

Appeler une Fonction

Pour utiliser la fonction que vous avez créée, vous l'appelez avec les arguments nécessaires :

```
val resultat = additionner(5, 3)  // Appelle la fonction additionner avec 5 et 3
setContent {
    Text("Le résultat est : $resultat")
}
```

Vous pouvez aussi utiliser des paramètres nommés, ce qui est pratique pour plus de clarté :

```
val resultat = additionner(a = 1, b = 2)
setContent {
    Text("Le résultat est : $resultat")
}
```

Fonctions Sans Paramètres et/ou avec Retour Unit

Certaines fonctions ne nécessitent pas de paramètres et/ou ne renvoient rien. Elles sont utiles pour des actions simples, comme afficher un message ou effectuer un traitement sans produire de résultat particulier. Le type de retour par défaut pour ces fonctions est Unit, ce qui signifie qu'elles ne renvoient rien.
Voici une fonction qui n'a pas de paramètres et ne renvoie rien :

```
fun saluer() {
    setContent {
        Text("Bonjour tout le monde !")
    }
}
```

- saluer : Nom de la fonction.
- () : Pas de paramètres.
- {} : Bloc de code exécuté.

Vous l'appelez simplement comme ceci :

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    saluer()
}
```

Si vous ne spécifiez pas de type de retour, Kotlin suppose par défaut que c'est Unit.

Fonctions avec Valeurs par Défaut

Vous pouvez définir des valeurs par défaut pour les paramètres d'une fonction. Cela signifie que si vous ne fournissez pas un argument lors de l'appel de la fonction, la valeur par défaut sera utilisée.
Pour définir une valeur par défaut, ajoutez = valeur après le type du paramètre dans la déclaration de la fonction.

```
fun saluer(nom: String = "ami") {
    setContent {
        Text("Bonjour, $nom !")
    }
}
```

- nom: String = "ami" : Le paramètre nom a une valeur par défaut de "ami".

Vous pouvez appeler cette fonction avec ou sans valeur :

```
saluer()  // Affiche "Bonjour, ami !"
saluer("Marie")  // Affiche "Bonjour, Marie !"
```

Exercices

Exercice 1

Créez une fonction pour préparer une boisson en machine à café. La fonction doit prendre le type de boisson et le nombre de tasses comme paramètres. Elle doit renvoyer un message indiquant combien de tasses de chaque boisson sont prêtes.

Exercice 2

Créez une fonction qui prend deux nombres et une opération (une String pouvant valoir : addition, soustraction, multiplication, division) comme paramètres. La fonction doit renvoyer le résultat de l'opération.

Exercice 3

Créez une fonction qui prend un nom et un âge comme paramètres, et qui renvoie un message de bienvenue personnalisé. La fonction doit également inclure 25 comme valeur par défaut pour l'âge si aucun âge n'est fourni.

Exercice 4

Créez une fonction qui répète les paroles d'une chanson un nombre de fois spécifié. La fonction doit prendre le titre de la chanson et le nombre de répétitions comme paramètres.

Exercice 5

Créez une fonction qui génère la table de multiplication pour un nombre donné. La fonction doit afficher les résultats sous forme de chaîne de texte.

Exercice 6

Créez une fonction qui prend une chaîne de texte et compte le nombre de voyelles dans cette chaîne.

Exercice 7

Créez une fonction qui convertit des mètres en kilomètres, des grammes en kilogrammes, et des litres en millilitres. La fonction doit prendre un nombre et l'unité à convertir.

Exercice 8

Créez une fonction qui génère un nom de super-héros. Elle prend en paramètre un prénom. Elle pioche une valeur aléatoire d'une liste de String contenant : "Super", "Mega", "Giga", "Ultra" et retourne le résultat collé au prénom.
Note : Sur une liste, utiliser la fonction random() pour récupérer une valeur aléatoire de celle ci.

Exercice 9

Créez une fonction qui prend la température en Celsius et retourne un message indiquant :

- "C'est froid" si inférieur à 10
- "C'est doux" si c'est entre 10 et 25
- "C'est chaux" sinon

Exercice 10

Créez une fonction qui calcule le prix total d'articles en fonction du prix unitaire et de la quantité achetée. La fonction doit aussi appliquer une réduction de 10% si le total dépasse 100 unités.

Exercice 11

Créez une fonction qui prend une map de noms de produit avec leurs prix et retourne un résumé formaté des produits et de leur prix total.