Programmation fonctionnelle

1. Fonctions

1.1. Définition

Une fonction est une expression qui prend en entrée des arguments et renvoie un résultat. Elle est semblable à une méthode mais est un objet et peut donc être utilisée de façon différente.

Pour définir une fonction, on utilise la syntaxe `(t: T) => ???`. La partie gauche correspond aux arguments du type désiré et la partie droite aux opérations effectuées sur ce paramètre.

La fonction est alors un objet de type `T => U` où `U` est le type de l'objet retourné dans la partie de droite. Une fonction peut prendre un, aucun ou plusieurs arguments.

Exemple de fonction à un argument

Prenons comme exemple une fonction qui multiplie un `Int` par `1.2`.

```
val fonctionUnArg = (i: Int) => i * 1.2
// fonctionUnArg: Int => Double = <function>785a95b
```

Sans spécifier son type de retour, il est automatiquement inféré à `Double` par le compilateur.

Exemple de fonction à deux arguments

Prenons comme exemple une fonction qui multiplie un `Int` à un `Double`. Le résultat par défaut sera un `Double`.

```
val fonctionDeuxArg = (i: Int, j: Double) => i * j
// fonctionDeuxArg: (Int, Double) => Double = <function>25521b7
```

On peut forcer le type de retour à `BigDecimal` et compter sur la conversion entre le type `Double` et `BigDecimal`. Le type de retour spécifié prévaut sur le type inféré par le compilateur.

```
val fonctionDeuxArgDecimal: (Int, Double) => BigDecimal = (i:
Int, j: Double) => i * j
// fonctionDeuxArgDecimal: (Int, Double) => BigDecimal =
<function>25521b7
```