Ce qu'il faut retenir sur les fonctions.

Comment écrire une fonction

Une fonction est composée des éléments suivants :

- Un **type de retour** (int, bool, string[], etc) qui indique le type de la variable qui sera retourné par la fonction à l'aide du mot clef return. Les fonctions ne peuvent retourner qu'un seul élément ! Pour retourner plusieurs informations, il y a d'autres astuces.
- Si la fonction ne retourne rien, je dois utiliser le mot clef `void` pour vide.
- Un **nom** pour la fonction. En générale on utilise un nom qui décrit ce que la fonction fait et on le fait commencer par une majuscule.
- **Entre parenthèses les paramètres** que je souhaite passer à ma fonction pour pouvoir les utiliser à l'intérieur de celle-ci.
- SI je n'ai pas de paramètres je mets quand même des accolades mais je les laisse vide.
- Puis **entre accolades** j'écris le corps de ma fonction.

Voici quelques exemples:

Cette fonction calcule ajoute trois variables entre elles et me retourne le résultat..

```
int Add(int a, int b, int c) {
return a + b + c;
}
```

int est le type de retour.

Add est le nom.

int a, int b et int c sont trois paramètres.

Concernant les paramètres optionnels

Il est possible d'ajouter une valeur par défaut aux paramètres ce qui fait que les paramètres deviennent optionnels !

Par exemple si je fais :

```
int Add(int a, int b, int c = 0) {
return a + b + c;
}
```

La variable c aura, par défaut la valeur 0 (j'aurai pu mettre 36, ça n'a pas d'importance). Donc, ce paramètre n'est plus obligatoire.

Ce qui signifie que je peux appeler la méthode **Add** des deux manières suivantes :

`Add(1,2,3);` ce qui me retournera la valeur 6.

Ou bien

`Add(1,2);` ce qui me retournera la valeur 3

Attention cependant, les paramètres optionnelles doivent être placés **à la droite** des autres arguments obligatoires.

par exemple je n'aurai **pas** pu faire :

```
int Add(int a, int b = 0, int c) {
return a + b + c;
}
```

"C'est quoi la définition d'une fonction ?"

Soit la fonction suivante :

public void DisBonjour(string name){ Console.WriteLine($"Bonjour {name} !"}}

La définition de cette fonction sera "`void DisBonjour(string name)`". La définition comprend donc

1. Le type de retour
2. Le nom
3. Les paramètres

Et elle ne comprend pas

1. Le corps de la fonction (son contenu, les accolades).
2. L'accessibilité de la fonction (`public`, `private`...).

"Une fonction peut-elle s’appeler elle même ?"

Oui ! Et c'est ce qu'on appelle une fonction **"récursive"** à la différence des fonctions que nous avons vu jusqu'à présent qui sont dites **itératives** ! Il faut bien faire attention à ne pas tomber dans le cas d'une fonction qui s’appellerait indéfiniment sous peine de causer une erreur bien connue la **StackOverflow** !