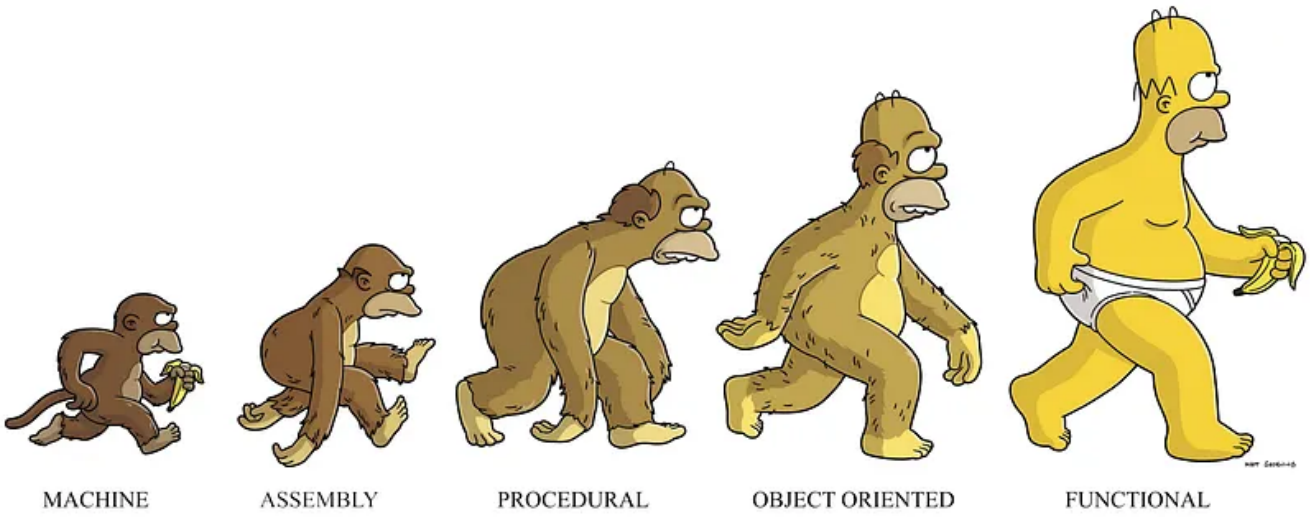
Programmation fonctionnelle



Basée sur l'utilisation des fonctions, à la condition de respecter certains principes.

La prog fonctionnelle n'admet pas le changement d'états et la mutation des données(contrairement à la prog impérative).

Les langages fonctionnels sont ceux vouent un culte à ces principes ou sont basés sur eux :

> Lisp (1958), Scheme (1975), Common Lisp (1984), Haskell (1987), OCaml (1996), Scala (2003), PureScript (2013)...

Les langages de programmation impératifs acceptant le passage de fonctions en paramètres peuvent être utilisés dans le cadre d'une approche fonctionnelle :

> ECMAScript, Java, C#, PHP, Perl, Python, Ruby, Kotlin...

Effets de bord, changement d'états

La programmation fonctionnelle s'affranchit de façon radicale des effets secondaires (ou effets de bord) en interdisant les opérations d'affectation (immutabilité).

Le paradigme fonctionnel n'utilise pas de machine à états pour décrire un programme, mais un emboîtement de fonctions qui agissent comme des “boîtes noires” que l'on peut imbriquer les unes dans les autres.

Chaque boîte possédant plusieurs paramètres en entrée mais une seule sortie, elle ne peut sortir qu'une seule valeur possible pour chaque n-uplet de valeurs présentées en entrée. De cette façon, les fonctions n'introduisent pas d'effets de bord.

Exemples d'effets de bord (side effects) :

- Modifier une variable globale ou la propriété d'un objet
- Ecrire dans la console ou à l'écran
- Ecrire ou lire dans un fichier
- Communiquer avec un réseau
- Communiquer avec un processus externe
- Appeler une fonction qui a des effets de bord