# Langages et programmation: programmation fonctionelle:

La programmation fonctionnelle est un paradigme de programmation comme la programmation impérative ou la programmation orientée objet.

Comme dit l'année dernière dans la partie du cours consacrée aux effets de bord (`https://pixees.fr/informatiquelycee/n_site/nsi_prem_pythonbase.html#effet_bord`), le paradigme fonctionnel cherche à éviter au maximum les effets de bord, dit autrement, en programmation fonctionnelle on va éviter de modifier les valeurs référencées par des variables. Pour se faire, on va chercher au maximum à utiliser les fonctions (d'où le nom de programmation fonctionnelle), mais ces fonctions ne devront pas modifier les variables: en programmation fonctionnelle, on s'efforce de coder des fonctions qui ne modifient pas l'état courant des variables. Les fonctions utilisées en programmation fonctionnelle sont parfois appelées "fonction pure": le résultat retourné par une fonction pure doit uniquement dépendre des paramètres passés à la fonction et pas des valeurs externes à la fonction (elle ne doit pas non plus engendrer d'effet de bord):

**Exercice 1:** Analysez le programme suivant:

```
i = 5
def fct():
  if i > 5:
    return True
  else :
    return False
fct()
```

La fonction ci-dessus n'est pas une fonction pure, car la valeur renvoyée par la fonction fct (True ou False) dépend d'une valeur extérieure à la fonction.

**Exercice 2:** Analysez le programme suivant:

```
def fct(i):
  if i > 5:
    return True
  else :
    return False
fct(5)
```

La fonction ci-dessus est une fonction pure, car la valeur renvoyée par la fonction fct (True ou False) dépend uniquement du paramètre passé à la fonction. Même si certains langages de programmation ont été conçus pour "imposer" au programmeur le paradigme fonctionnel (Lisp, Scheme, Haskell...), il est tout à fait possible d'utiliser le paradigme fonctionnel avec des langages de programmation plus "généralistes" (Python par exemple).

Nous allons maintenant travailler sur un exemple de programme Python utilisant le paradigme fonctionnel:

**Exercice 3:** Analysez le programme suivant:

```
l = [4,7,3]
def ajout(i):
  l.append(i)
```

Selon vous, le programme ci-dessus respecte-t-il le paradigme fonctionnel? La réponse à la question posée ci-dessus est non, car nous avons un effet de bord (la fonction ajout modifie le tableau l)

**Exercice 4:** Analysez le programme suivant:

```
def ajout(i,l):
        tab = l + [i]
  return tab
```

Selon vous, le programme ci-dessus respecte-t-il le paradigme fonctionnel? La fonction ajout ne modifie aucune variable, elle crée un nouveau tableau (tab) à partir du tableau l et du paramètre i (le signe + permet de créer un nouveau tableau, ce nouveau tableau est constitué des éléments contenus dans le tableau l auxquels on ajoute la valeur i), la fonction renvoie le tableau ainsi créé.

D'une façon plus générale, la méthode append de Python ne respecte pas le paradigme fonctionnel puisque append modifie une donnée existante. Le paradigme fonctionnel va amener le programmeur non pas à modifier une valeur existante, mais plutôt à créer une nouvelle grandeur à partir de la grandeur existante : une grandeur existante n'est jamais modifiée, donc aucun risque d'effet de bord.

**Utiliser le bon paradigme de programmation:**

Il est important de bien comprendre qu'un programmeur doit maitriser plusieurs paradigmes de programmation (impératif, objet ou encore fonctionnelle). En effet, il sera plus facile d'utiliser le paradigme objet dans certains cas alors que dans d'autres situations, l'utilisation du paradigme fonctionnelle sera préférable. Être capable de Choisir le "bon" paradigme en fonction des situations fait partie du bagage de tout bon programmeur.