# TP7 — Classes abstraites

**Objectif :** Construire des hiérarchies de classes basées sur une classe abstraite ou sur une interface.

On va gérer des fonctions de $\mathbb{R}$ dans $\mathbb{R}$, ainsi que certaines opérations sur ces fonctions.

**Remarque** : *On pourrait aussi procéder autrement en pratique ; cet exemple permet de prendre en main les classes abstraites.*

**Consigne générale** : Dès que possible [1], on donnera des exemples et on les codera, afin de vérifier la justesse de l'implémentation.

1. Écrire une classe abstraite **`Fonction`**. Elle contiendra deux méthodes abstraites :
   (a) `évaluation`, qui évaluera [2] la fonction en un point $x$ de $\mathbb{R}$ ;
   (b) `dérivée`, qui renverra la dérivée de la fonction (et non pas juste son évaluation en un point).
2. Ajouter deux méthodes qui évaluent en un point $x$ de $\mathbb{R}$, en utilisant les méthodes abstraites précédentes, la dérivée d'une somme et la dérivée de la composée de deux fonctions.
3. Écrire une classe fille **`Exponentielle`** pour représenter les fonctions exponentielles, de la forme
$$f(x) = be^{ax}.$$
   Définir les fonctions `évaluation` et `dérivée` pour ces fonctions.
4. Écrire une classe fille **`Trigonométrique`** pour représenter les fonctions trigonométriques, de la forme
$$f(x) = a\cos(x) + b\sin(x).$$
   Définir les fonctions `évaluation` et `dérivée` pour ces fonctions.
5. Écrire une classe fille **`Polynôme`** pour représenter les fonctions polynomiales, de la forme
$$f(x) = \sum_{i=0}^{n} a_i x^i.$$
   Définir les fonctions `évaluation` et `dérivée` pour ces fonctions.

**Pour aller plus loin** :

6. Rajouter des surcharges d'opérateurs dans les classes **`Exponentielle`**, **`Trigonométrique`** et **`Polynôme`**. Par exemple, on pourra surcharger `+` et `*` entre deux instances de la classe **`Polynôme`**.
7. Modifier la classe **`Trigonométrique`** pour représenter les fonctions de la forme
$$f(x) = a\cos(\alpha x) + b\sin(\beta x),$$
   et ajouter (ou modifier) des surcharges d'opérateurs le cas échéant.
8. Réfléchir à l'implémentation des surcharges de `+` et `*` dans la classe **`Fonction`**.

1. Donc dès la question 3 !
2. On pourra aussi surcharger l'opérateur `__call__`, mais ce n'est pas obligatoire.