# Algo & Prog, avec Python
# (L1-Sciences)
# TD n° 7, Automne 2016

## Importation d'un module

**Exercice 7.1** Proposez trois exemples d'utilisation de `import` permettant de calculer $\sqrt{5}$ au toplevel.

```
> sqrt(5)
NameError: name 'sqrt' is not defined
> import math
> sqrt(5)
NameError: name 'sqrt' is not defined
```

## Polynômes

**Exercice 7.2** a) Quelle serait en Python la représentation creuse sous forme de liste du polynôme p = $2 - 3x^5 + x^4$ ?
b) Et sa représentation dense ?

**Exercice 7.3** *Préparation de l'exercice 7.4*
a) En utilisant le tri prédéfini de Python, comment programmeriez-vous la fonction `insertion(x,LT)` prenant une liste de nombres `LT` triée en croissant, et retournant une copie croissante de `LT` dans laquelle on aura inséré `x` à sa place (il est interdit de faire muter la valeur de `LT`) ?

```
insertion(5,[2,4,7,8,9]) → [2,4,5,7,8,9]
```

b) Critiquez cette première solution sur le plan du coût du calcul.
c) Reprogrammez la fonction `insertion` sans utiliser une fonction de tri.

**Exercice 7.4** Programmez la fonction `mono_plus_poly(c,e,p)` retournant le nouveau polynôme obtenu en insérant le monôme $cx^e$ à sa place dans le polynôme `p`. Exemples :

```
mono_plus_poly(5,3, [[2,5],[-1,4],[-2,1]]) → [[2,5],[-1,4],[5,3],[-2,1]]
mono_plus_poly(3,4, [[2,5],[-1,4],[-2,1]]) → [[2,5],[2,4],[-2,1]]
mono_plus_poly(1,4, [[2,5],[-1,4],[-2,1]]) → [[2,5],[-2,1]]
```

*N.B. Il est très important d'assimiler les programmes du cours et de savoir les rédiger jusqu'à ce qu'ils deviennent « naturels ». Un programmeur connaît ses schémas algorithmiques de base, comme le mathématicien connaît ses théorèmes et leurs démonstrations, ou le physicien les lois physiques et les exemples classiques les illustrant. Tous trois vont sans cesse les retrouver dans la pratique et vont s'en inspirer, les adapter, les déformer, etc.* ***Une grande partie de l'intelligence humaine provient de la capacité (éduquée) à reconnaître des schémas déjà vus.*** *A bon entendeur…*

# Algo & Prog, avec Python
# (L1-Sciences)
# TP n° 7, Automne 2016

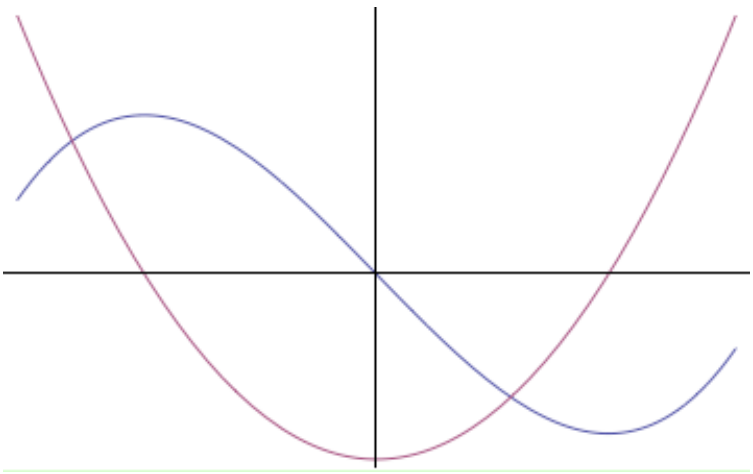
***Vous complétez directement le fichier polycreux.py que vous commencez par télécharger.***

## Polynomes creux

**Exercice 7.1** a) Programmez la fonction `sub(p,q)` de soustraction de deux polynômes.
b) Programmez la fonction `mul(p,q)` de multiplication de deux polynômes.
c) Programmez la fonction `valeur(p,x)` retournant la valeur du polynôme `p` au point x. Attention, `p` n'est pas une fonction mais un polynôme ! *C'est la fonction `valeur` qui permettra de transformer `p` en fonction grâce à `fpoly`...*
d) Rajoutez la fonction `deriv(p)` retournant le **polynôme dérivé** du polynôme `p`. Par exemple :

```
>>> p = [[1, 4], [7, 2], [-1, 0]]          # p = x^4+7x^2-1
>>> deriv(p)
[[4, 3], [14, 1]]                          # 4x^3+14x
```

**Exercice 7.2** Utilisez une tortue graphique pour programmer une fonction `dessiner(p)` sans résultat, chargée de dessiner la courbe du polynôme `p` [en bleu] ainsi que celle de sa dérivée [en rouge], sur l'intervalle `[-2,2]`. Utilisez `fpoly` pour passer d'un polynôme à sa fonction associée. Voici le résultat de `dessiner([[1,3],[-5,1]])` , donc pour `p = x^3-5x`.



## Exercice optionnel (typique examen)

**Exercice 7.3** Les polynômes de Chebyshev sont définis par $T_n(x) = cos(n\ acos(x))$.
**a)** Heureusement pour nous, ils vérifient aussi la relation de récurrence :

$$T_n = 2x\ T_{n-1} - T_{n-2}$$

Programmez une fonction `(cheby n)` retournant le polynôme $T_n$. Vérifiez si vous obtenez bien par exemple $T_2 = 2x^2 - 1$ soit en Python `[[2,2],[-1,0]]`.
b) Déduire de la valeur de $T_3$ une formule trigonométrique pour $cos(3x)$ en fonction de $cos(x)$.