1 Tracé graphique

1.1 Déclaration de liste par compréhension

En Mathématiques, donné un sous-ensemble $A \subset E$ et une application f de E dans F, on peut définir le sous-ensemble suivant de F :

$$f(A) = \{f(x) \mid x \in A\}$$

l'ensemble des images par f des éléments de A.

En Python, la définition d'une liste peut s'obtenir sur le même principe : si `X` est une liste, séquence, ou itérateur (`range(n,m)`) et si `f` est une fonction pouvant s'appliquer aux éléments de `X`, alors on peut définir la liste **par compréhension** :

```
L = [ f(e) for e in X ]
```

Exemples :

```
In [1]: L1 = [ x for x in range(10) ]

In [2]: L1
Out[2]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [3]: L2 = [ x**2 for x in L1 ]

In [4]: L2
Out[4]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

1.2 Tracé avec `matplotlib.pyplot`

Le module `matplotlib.pyplot` permet d'effectuer des tracés de points, courbes et surfaces dans une fenêtre graphique.

Pour l'utiliser il faudra d'abord choisir comme "shell" dans les préférences celui de "anaconda".
(Au démarrage du shell si la mention "anaconda" apparaît, c'est ok, sinon configurer le shell.)

puis saisir dans le programme ou la console, l'instruction d'importation :

```
from matplotlib.pyplot import *
```

• La fonction `plot(X,Y)` prend en paramètres les deux listes `X` des abscisses et `Y` des ordonnées des points de la courbe à tracer. Il réalise le tracé dans une fenêtre graphique.

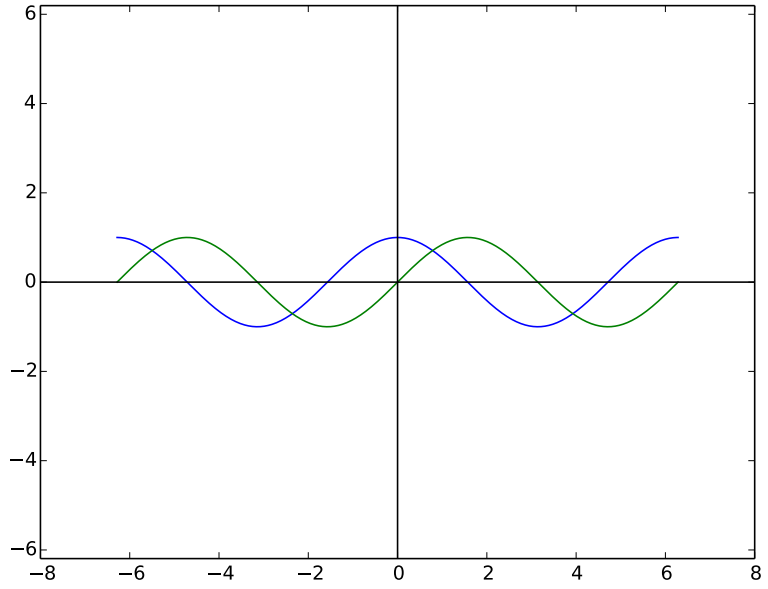
Ainsi par exemple, le code suivant trace les fonction cos et sin au dessus du segment $[-2\pi; 2\pi]$:

```
from matplotlib.pyplot import *

from math import pi, cos, sin

n = 1000
a = -2*pi
b = 2*pi
# 1000 points regulièrements espaces dans [a ; b] :
X = [ a + k*(b-a)/(n-1) for k in range(n) ]
Yc = [ cos(x) for x in X ]       # Valeurs de cos en X
Ys = [sin(x) for x in X ]        # Valeurs de sin en Y
plot(X,Yc)       # Trace de cos
plot(X,Ys)       # Trace de sin
axhline(color='black')        # Trace axe des x (en noir)
axvline(color='black')        # Trace axe des y (en noir)
axis("equal")                 # Repère orthonorme
show()                        # Affichage
```

ce qui produit le graphique :



On a utilisé aussi les fonctions suivantes :

Fonctions de `pyplot` utilisées	
`plot(X,Y)`	trace la courbe des points d'abscisses dans `X` et ordonnées dans `Y`
`show()`	affiche le tracé
`axhline()`	trace l'axe des abscisses
`axvline()`	trace l'axe des ordonnées
`axis("equal")`	impose un repère orthonormé

2 Application

On rappelle le théorème des valeurs intermédiaires qui énonce l'existence d'une racine our une application continue sur un intervalle $[a,b]$ qui y change de signe :

Théorème. Soit f une application réelle continue sur un intervalle $[a,b]$ et telle que $f(a)\times f(b)\leqslant 0$. Alors $\exists c\in[a,b]$, tel que $f(c)=0$.

On pourra alors chercher une racine une valeur approchée d'une racine de f sur $[a,b]$ par dichotomie :

```
def dichotomie(f,a,b,e):
        # f est la fonction
        # (definie par : f = lambda x: f(x))
        # a < b : bornes de l'intervalle de recherche
        # e : l'erreur maximale autorisee
        while b-a > e:
            m = (a+b)/2
            if f(a)*f(m) <= 0:
                a, b = a, m
            else:
                a, b = m, b
        return (a+b)/2
        # ou a : valeur par defaut à e près
        # ou b : valeur par excès à e près
        # ou [a,b] : encadrement à e près
```

Exercice. Soit $f(x)=x^3-x^2-2x+1$.

1. Effectuer le tracé de la courbe de f sur l'intervalle $[-1,5;2]$ à l'aide de `pyplot`.
2. Obtenir à l'aide de la fonction `dichotomie` des encadrements à 10^{-3} près de toutes les racines de f ; justifier.