**Objectifs du TP :**

- programmer des algorithmes simples à l'aide du langage de programmation Python,
- appréhender les limites de la machine,
- illustrer le cours d'analyse de L1 via quatre grandes méthodes d'approximation de solution de $f(x) = 0$

**Exercice 1**

Voici une suite de commandes permettant de calculer la surface d'un disque de rayon $r = 1$ :

```
In [1]: from math import pi
        r = 1.; # on fixe la valeur de r
        S = pi * r**2; # on calcule la surface
        print("la surface du disque de rayon", r, "est", S);
```

```
la surface du disque de rayon 1.0 est 3.141592653589793
```

1. En modifiant les lignes ci-dessus, calculer les surfaces des disques de rayon 2 et $1/2$.
2. Compléter les lignes ci-dessous pour calculer le volume d'une sphère de rayon $r = 1$.
3. Lancer la commande **x = sqrt(2)** ci-dessous. Que se passe-t-il ? Lancer la commande d'inclusion **from math import sqrt** et recommencer. Quelle est la différence entre **from math import sqrt** et **from math import *** ?

```
In [2]: #
        # Question 2.
        #
        V=0; # à compléter...
        print("le volume de la sphère de rayon", r, "est", V);
```

```
le volume de la sphère de rayon 1.0 est 0
```

```
In [3]: #
        # Question 3.
        #
        x=sqrt(2); # à compléter...
```

**A partir de maintenant, on travaille sous IDLE Python**

**Exercice 2**

1. Soit la suite $x_{n+1} = a\,x_n - b$ avec les valeurs initiales $x_0 = 1.0$, $b = 4096.1$ et $a = b + 1$. Calculer à la main ce que vaut $x_n$ pour tout $n$. Programmer le calcul des 10 premières valeurs de $x_n$ avec une boucle while. Recommencer avec la valeur $b = 4095.1$. Que se passe-t-il ? Recommencer avec $b$ de la forme $4^k - 1$ pour $k = 3$, $k = 4$, etc...

2. Programmer le calcul des 30 premiers termes des suites $(u_n)$ et $(v_n)$ données par

$$\begin{cases} u_0 = 1/2, \\ u_{n+1} = 3\,u_n - 1 \end{cases} \quad et \quad \begin{cases} v_0 = 1/3, \\ v_{n+1} = 4\,v_n - 1 \end{cases}$$

**Exercice 3 - Dichotomie**

On considère le programme suivant qui calcule une approximation de la valeur de $\sqrt{2}$ par la méthode de la dichotomie appliquée à la résolution de l'équation $x^2 - 2 = 0$ sur l'intervalle $[1,2]$ avec la précision $p = 10^{-3}$.

Modifier ce programme pour qu'il calcule cette approximation avec la précision $p = 10^{-6}$ et qu'il affiche le nombre d'itérations effectuées.

```
In [10]: from math import *

         p = 1e-6;
         a=1.0;
         b=2.0;
         m=(a+b)/2;

         while (abs(m**2 - 2) > p) :
             if (m**2 - 2)*(a**2 - 2) < 0 :
                 b=m;
                 m=(a+b)/2;
             elif (a**2 - 2) != 0 :
                 a=m;
                 m=(a+b)/2;
             else :
                 m=a

         print("on trouve le résultat : ", m)
```

```
on trouve le résultat :  1.4142136573791504
```

**Exercice 4 - Sécante**

Programmer la méthode de la sécante pour calculer une approximation de la valeur de $\sqrt{2}$ par la résolution de l'équation $x^2 - 2 = 0$ sur l'intervalle $[1,2]$ avec la précision $p = 10^{-3}$, et qui affiche le nombre d'opérations effectuées.

Recommencer avec $p = 10^{-6}$.

**Exercice 5 - Méthode de Newton**

Programmer la méthode de Newton pour calculer une approximation de la valeur de $\sqrt{2}$ par la résolution de l'équation $x^2 - 2 = 0$ sur l'intervalle en partant du point $x_0 = 2$ avec la précision $p = 10^{-3}$, et qui affiche le nombre d'opérations effectuées.

Pour cela, on rappelle que la méthode de Newton pour résoudre l'équation $f(x) = 0$ consiste à considérer la suite des points

$$\begin{cases} x_0 \text{ fixé} \\ x_{n+1} = x_n - f(x_n)/f'(x_n) \end{cases}$$

Recommencer avec $p = 10^{-6}$, et comparer le nombre d'itérations avec ceux des deux méthodes précédentes.