TD : L'algorithme de dichotomie

1. Soit f la fonction définie sur $\mathbb{R}$ par $f(x) = x^3 - 3x^2 + 5$.
   a) Montrer que f possède une unique racine, et que celle-ci se situe dans [-2,-1].
   b) Écrire un algorithme de dichotomie permettant de calculer une valeur approchée au centième de cette racine.
   c) Ajouter une ligne à l'algorithme pour qu'il affiche le nombre d'étapes nécessaires.

2. Soit f la fonction définie par $f(x) = \mathrm{x} - 4\ln(\mathrm{x})$, pour x>0.
   a) Montrer que l'équation $f(x) = 0$ possède deux solutions, la plus petite sur [1,e], la seconde sur [e,e^3].
   b) Écrire un algorithme de dichotomie permettant de calculer une valeur approchée au millième la plus petite racine.

3. On appelle fonction logistique la fonction Λ définie sur $\mathbf{R}$ par : $\forall x \in \mathbf{R}, \Lambda(x) = \frac{1}{1+\mathrm{e}^{-x}}$.
   a) Justifier l'existence d'un unique réel x_0 tel que : $\Lambda(x_0) = x_0$, et que $0 < x_0 < 1$.
   a) Étudier la fonction g(x)=lambda(x)-x sur]0,1[et appliquer le théorème de la bijection
   b) Compléter le programme suivant pour qu'il affiche une valeur approchée de x_0 à 0.001 près :

```
a=0
b=1
while (b-a)< 0.001 :
    c=(a+b)/2
    if 1/(1+exp(-c))>c :
            b=c                    # faire un dessin !
    else:
            a=c
print(c)
```

   c) En réalité, en affichant c on donne une valeur approchée plus précise qu'à 0.001. Donner une précision meilleure que 0.001 qui est fournie par l'algorithme.

4. On pose la fonction polynôme $P(x) = x^3$définie sur R. Écrire un programme qui demande à l'utilisateur un réel strictement positif eps et qui renvoie une valeur approchée de la racine cubique de 2 à eps près. On rappelle que la racine cubique de 2 est l'unique antécédent de 2 par la fonction P.

Corrigé

Exercice 1

a) L'image de -2 est négative, celle de -1 est positive, comme f est continue, alors d'après le théorème des valeurs intermédiaires f s'annule sur [-2,-1].

b) et c)

```
a=-2                        # on sait que la solution se situe sur [-2,-1].
b=-1
n=0                         # compteur d'étapes
while (b-a)>0.01 :
        c=(a+b)/2
        n+=1                #compte les étapes
        if c**3-3*c**2+5>0
                b=c
        else :
                a=c
print(c)            # Valeur approchée de la solution
print(n)            # affiche le nombre d'étapes
```

Exercice 2

```
from math import *          # pour utiliser exp et log
a=1
b=exp(1)                    # ou np.(e) dans la bibliothèque numpy
while (b-a)>0.001 :
        c=(a+b)/2
        if c-4*log(c)>0:
                b=c
        else :
                a=c
print(c)
```

Exercice 4

```
        eps=float(input('rentrer la précision souhaitée'))
        a=1                 # on sait que racine cubique de 2 est sur [1,2] car 2^3>3
        b=2
        while (b-a)>eps:
                c=(a+b)/2
                if c**3>2
                        b=c
                else :
                        a=c
        print(c)
```