



Interrogation 2

Algorithmes Classiques

Correction

Exercice 1 :

1. Écrire une fonction `minimum(tab:list) -> float` qui renvoie la valeur minimale d'un tableau de nombre `tab` (on renverra $-\infty$ si le tableau est vide).

```
1 def minimum(tab:list) -> float :
2     if len(tab) == 0 :
3         return(-float("inf"))
4     else :
5         m=tab[0]
6         for k in range(1,len(tab)) :
7             if tab[k]<m :
8                 m=tab[k]
9         return(m)
```

2. Écrire une fonction `miniOcc(tab:list) -> tuple` qui renvoie l'une des valeurs du tableau `tab` qui apparaît le moins, avec son nombre d'occurrence (exemple : `miniOcc([1,1,4,1,7,2,7,3,4])` renverra (2,1) ou (3,1)).

```
1 def miniOcc(tab:list) -> tuple :
2     C = float("inf") # Compteur le plus petit
3     val = float("inf") # Valeur apparaissant le moins de fois
4     for k in range(len(tab)) :
5         c = 0 # nouveau compteur pour le nombre d'apparition de tab[k]
6         for j in range(len(tab)) :
7             if tab[j] == tab[k] :
8                 c += 1
9         if c < C : # Si tab[k] a un compteur plus petit
10            C = c # enregistre le compteur
11            val = tab[k] # enregistre la valeur
12     return((val,C))
```

3. Écrire une fonction `zeroDicho(f:"function", a:float, b:float, epsilon:float) -> float` qui renvoie le zéro d'une fonction f continue s'annulant sur l'intervalle $[a,b]$ par la méthode dichotomique.

```
1 def zeroDicho(f:"function", a:float, b:float, epsilon:float) -> float :
2     c,d = a,b
3     while d-c>epsilon :
4         m = (c+d)/2
5         if f(c)*f(m) < 0 :
6             d=m
7         elif f(m)==0 :
8             return(m)
9         else :
10            c=m
11     return((c+d)/2)
```