Recherche par dichotomie

I- Akinator

Vous jouez à essayer de deviner le nombre auquel pense votre adversaire : en début de partie, celui-ci pense à un nombre entre 0 et 100, le note sur une feuille qu'il retourne (pour éviter la triche). Vous devez trouver ce nombre en un minimum de propositions. L'adversaire ne peut dire que "c'est plus", "c'est moins" et "bravo".

1) Essayez de décrire votre stratégie.

2) En justifiant votre réponse, indiquez quel serait le nombre minimum de questions à poser afin de trouver le résultat de façon sûre et certaine.

3) Écrire une fonction akinator(n) qui va chercher à deviner le nombre x auquel vous pensez ($0 \leq x \leq n$) en utilisant la même stratégie que vous.

II- Recherche d'une approximation d'une solution de l'équation f(x)=0

Application : La fonction **$f(x)=x^3-1{,}45x^2-11{,}55x+5{,}39$** admet une solution **s** telle que **f(s)=0** avec **$0 \leq s \leq 1$**
Donner une valeur arrondie à 10^{-3} de **s** à l'aide d'une recherche par dichotomie.
On donne `def f(x):return x**3-1.45*x**2-11.55*x+5.39`
Proposer une adaptation de l'algorithme **akinator** pour arriver à ce résultat.

III- Recherche par dichotomie dans une liste

Une **liste** de nombres est générée par la commande

```
liste=[randint(0,10)]
for i in range(1,100000) :
  liste.append(liste[i-1]+randint(1,10))
```

On souhaite rechercher si un nombre est présent dans cette liste le plus rapidement possible.
1) D'après ce code, la liste est-elle déjà ordonnée ? Si oui, en ordre croissant ou décroissant ?
2) Est-il possible que la liste contienne deux fois le même nombre ?
3) Écrire une fonction **recherche_dicho(nombre,liste)** qui recherche par dichotomie le **nombre** dans **liste** et retourne si elle le trouve son indice et si elle ne le trouve pas les 2 indices encadrant la position qu'il aurait dû avoir
4) Utiliser le benchmark de la séance *algorithmes01* pour vérifier si cette méthode est réellement plus rapide que l'algorithme **indice(liste,valeur)** écrit précédemment.
Attention : votre fonction **bench** génère par défaut une liste aléatoire, ce qui n'est pas le cas de la liste ci-dessus, modifiez la fonction en conséquence !