TD Informatique : Révisions de sup
listes Python, boucles, fonctions,…

Exercice 1 : créer les listes suivantes :
L1 : liste croissante des entiers naturels pairs strictement inférieurs à 20
L2 : coupe (slicing) du 3ème au 6ème élément inclus de la liste L1 précédente
L3 : liste L1 sans les premier et dernier éléments
L4 : liste décroissante des entiers naturels impairs strictement inférieurs à 10
L5 : liste décroissante des racines carrées des entiers naturels impairs inférieurs à 10.

Exercice 2 : écrire une fonction echange(L,i,j) qui échange les éléments i et j d'une liste L.
Tester la fonction en inversant les 2ème et 5ème éléments de la liste L1 de l'exercice précédent.

Exercice 3 : opération sur des listes

On considère les deux listes :

```
L1 = [i for i in range(5)]
L2 = [i**2 for i in range(12) if (i**2)%3==1]
```

1) Concaténation ≠ addition terme à terme

a) effectuer la concaténation des deux listes : L1+L2

b) écrire une fonction sum_liste qui renvoie une liste dont les termes correspondent à la somme terme à terme de deux listes de même taille (sans modifier les deux listes). La fonction retourne None si les listes ne sont pas de même taille.

2) multiplication des listes ≠ multiplication terme à terme

a) que représente 3*L1 ?

b) écrire une fonction mult_liste permettant d'obtenir une liste dont les termes correspondent aux termes d'une liste L multipliés par un facteur k.

Exercice 4 : écrire une fonction qui prend en argument un entier $n > 0$ et retourne le seul entier k positif ou nul tel que $2^k \leq n < 2^{k+1}$.

Exercice 5 : recherche d'élément dans une liste (ou une chaîne de caractère)
1) Écrire une fonction estdans(L,x) qui renvoie True si l'élément x se trouve dans la liste L et False sinon.
2) Modifier la fonction précédente pour qu'il renvoie la position de l'élément x lorsqu'il apparaît la première fois dans la liste L, ou renvoie –1 si x n'est pas présent dans L.
3) Écrire une fonction occurrence(L,x) qui renvoie le nombre de fois qu'apparaît l'élément x dans la liste L.

Exercice 6 : recherche du minimum dans une liste de nombres
Écrire une fonction mini(L) qui renvoie la valeur du minimum et la position de ce minimum (la première fois qu'il apparaît) dans la liste L.

Exercice 7 : insertion d'un élément dans une liste
Écrire une fonction insere(L,i,x) qui renvoie une liste dans laquelle l'élément x a été inséré à la position i de la liste L.

Exercice 8 : Approximations de racines d'une équation
On cherche à approcher la solution c de l'équation $f(x) = 0$ (la fonction f possédant les « bonnes » propriétés).

1) Dichotomie
On suppose que f : [a,b]$\rightarrow R$ est continue et que $f(a).f(b)<0$.
a) Écrire une fonction nommée *dicho* qui prend pour arguments f, a, b et epsilon, et qui renvoie une valeur approchée à epsilon près d'une racine de f appartenant au segment [a,b].
b) De même avec une fonction *dichotomie* qui renvoie également le nombre d'itérations.

2) Méthode de Newton
Soit f : [a,b]$\rightarrow R$. On suppose que :
- la fonction f est de classe C^1 et que $f(a).f(b)<0$.
- sa dérivée $f' \neq 0$ sur [a,b]

On définit alors la suite (u_n) par u_0 = a ou b et $u_{n+1} = u_n - f(u_n) / f'(u_n)$ pour tout n dans N.
Proposer une fonction nommée *newton* qui prend pour arguments f, df (dérivée de f), x0 et epsilon, et qui renvoie une valeur approchée à epsilon près d'une racine de f appartenant au segment [a,b], ainsi que le nombre d'itérations.

3) Faire un test avec une fonction de votre choix (polynôme de degré 2 par exemple). Comparer le nombre d'itérations des deux méthodes. Conclure.