Dans un premier temps, on répondra aux questions à l'écrit puis pour vérifier que nos algorithmes sont bien écrits, on les programmera sur machine.

Exercice 1 :

(1) Rappeler comment fonctionne l'algorithme de dichotomie.

(2) Écrire la fonction **approximation_racine_de_2** qui prend en argument un entier et qui renvoie les n premières décimales de $\sqrt{2}$.

(3) Inclure un compteur afin de déterminer le nombre d'itérations nécéssaire pour calculer les n premières décimales.

(4) Tester ce code. Combien faut-t'il d'itérations pour calculer les 10 décimales de $\sqrt{2}$?

(5) Tester ce code avec 15 puis 16 décimales. Comment expliquer ce résultat.

Exercice 2 :

On se donne la suite définie par $\begin{cases} u_{n+1} = 3u_n + 2 \\ u_0 = r \end{cases}$

(1) Écrire la fonction **ntermes** qui prend comme arguments r et n et renvoie la liste des n premiers termes de la suite.

(2) Écrire la fonction **nsomme** qui prend comme arguments la liste L des n premiers termes d'une suite et renvoie la somme de ces termes.

(3) En déduire la fonction **moyenne** qui prend en argument la liste L et calcule la moyenne des termes.

(4) Tester votre code pour calculer la moyenne des 50 premiers termes de la suite.

Exercice 3 :

(1) Importer la bibliothèque **random**. Avec la fonction **random** afficher un flottant aléatoire entre 0 et 1 exclus.

(2) Définir une liste L de 100 nombres aléatoires.

(3) Combien de nombres dans L sont plus petits que $0,5$. Est-ce satisfaisant ?

(4) Écrire une commande qui à partir de la liste L définit une nouvelle liste d'entiers compris entre 1 et 100 (on n'utilisera pas **randint**)

Exercice 4 :

On se donne une liste L. Proposer une fonction sousSuiteCroissante qui renvoie la liste L2 contenant la plus grande sous-suite croissante contenue dans L.

Par exemple si L = [3,6,4,8,9,2,12] la fonction renverra la liste $L2$ qui vaut [3,6,8,9,12] .

Exercice 5 :

Soit *tab[0...n-1]* un tableau de n entiers, $n \geqslant 1$.

(1) On considère dans cette question que *tab* est un tableau trié jusqu'à l'indice $j-1$ ($j < n$).

a. Écrire la fonction **insere(tab, elem, j)** qui insère l'élément *elem* dans *tab[0..j-1]* en laissant le tableau trié jusqu'à l'indice j.

Par exemple :

```
L = [1, 3, 5]
print(insere(L, 2, 2))
--> [1, 2, 3]
L = [1, 4, 7]
print(inselere(L, 3, 1))
--> [1, 3, 7]
```

b. Écrire un jeu de tests.

c. Coder puis tester cette fonction.

(2) Le tableau n'est plus trié.

On se donne la fonction **a_quoi_ca_sert** :

```
def a_quoi_ca_sert(tab):
    for j in range(1, len(tab)):
        insere(tab, tab[j], j)
```

a. Dire à quoi sert cette fonction.

b. Écrire un jeu de tests de cette fonction.

c. Vérifier qu'il fonctionne.