
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 :

On se donne la suite u définie par :

$$\begin{cases} u_{n+1} &= 2u_n + 1 \\ u_0 &= 1 \end{cases}$$

(1) Proposer un algorithme qui permet d'afficher les 100 premiers termes.

(2) Proposer un algorithme qui permet d'afficher le 100ème terme.

(3) Proposer un algorithme qui permet d'afficher le plus petit indice n tel que $u_n > 1000000$.

Exercice 2 :

(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 3 :

(1) Écrire une fonction **chercher** qui prend en argument une liste et un nombre et renvoie l'indice de ce nombre dans la liste si il existe et -1 sinon. Par exemple :

```
L = [1,2,5]
print (chercher(L,2))
  --> 1
print (chercher(L,3))
  --> -1
```

(2) Écrire dans le fichier TP4.py cet algorithme.

(3) Lancer le fichier **test_chercher.py**, analyser et comprendre ce fichier.

Exercice 4 :

(1) Écrire la fonction **moyenne(tab)** qui prend un tableau de nombre en paramètre et renvoie la moyenne des nombres.

(2) Écrire un jeu de tests pertinent dans le fichier **test_moyenne.py**.

(3) Coder la fonction **moyenne(tab)** et tester là.

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.