BOUCLES (2)

2. Boucles While

(a) Principe

Une alternative à l'instruction for est la boucle **while**.
Le principe est simple : une série d'instructions est exécutée tant qu'une condition est vraie.

Exemple d'introduction

On se donne la suite (u_n) définie par $\begin{cases} u_0 = 1000 \\ u_n = -4000 \times 0{,}98^n + 5000 \end{cases}$ pour tout $n \in \mathbb{N}^*$

Soit S un réel.
On admettra que la suite (u_n) est strictement croissante.
Le script suivant permet de déterminer le plus petit entier N tel que : $u_n > S$.
(L'algorithme mis en jeu est appelé « algorithme de seuil »)

```
1  s=float(input("saisir le seuil : "))
2
3  u=1000
4  n=0
5
6  while u<=s:
7      n=n+1
8      u=-4000*0.98**n+5000
9
10 print("le seuil S est dépassé pour : ",n)
```

Commentaires
Une boucle **while** nécessite généralement trois éléments pour fonctionner correctement :

1. Initialisation de la variable d'itération avant la boucle (ligne 3).
2. Test de la variable d'itération associée à l'instruction while (ligne 6).
   Pour cela on utilise les instructions de comparaison vues dans le TP 4
3. Mise à jour de la variable d'itération dans le corps de la boucle (ligne 8).

On remarque qu'il est encore une fois nécessaire d'indenter le bloc d'instructions correspondant au corps de la boucle (ici, les instructions lignes 5 et 6).

(b) Exemples et mise en garde

On donne les programmes suivants :

programme 1

```
i = 0
while i < 10:
    print("Le python c'est cool !")
```

A. Saisissez et exécutez ce programme.

B. Que se passe-t-il ? Justifier.

ii. programme 2

On reprend le script donné en introduction.

A. Exécuter le programme et choisir S=5000 .

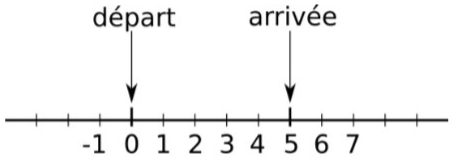
B. Que se passe-t-il ? Justifier.

Remarque à retenir

Vous pouvez toujours stopper l'exécution d'un script Python à l'aide de la combinaison de touches Ctrl-C (c'est-à-dire en pressant simultanément les touches Ctrl et C).

(c) Exemples applications

1. Sauts de puce



On imagine une puce qui se déplace aléatoirement sur une ligne, en avant ou en arrière, par pas de 1 ou -1. Par exemple, si elle est à l'emplacement 0, elle peut sauter à l'emplacement 1 ou -1; si elle est à l'emplacement 2, elle peut sauter à l'emplacement 3 ou 1, etc.

a. Avec une boucle while, simuler le mouvement de cette puce de l'emplacement initial 0 à l'emplacement final 5 (voir le schéma de la figure).

b. Combien de sauts sont nécessaires pour réaliser ce parcours ?

c. Relancez plusieurs fois le programme. Trouvez-vous le même nombre de sauts à chaque exécution ?

Vous utiliserez l'instruction **random.choice([-1,1])** qui renvoie au hasard les valeurs -1 ou 1 avec la même probabilité. Avant d'utiliser cette instruction vous mettrez au tout début de votre script la ligne **import random.**

2. La suite de Fibonacci

La suite de Fibonacci est une suite mathématique qui porte le nom de Leonardo Fibonacci, un mathématicien italien du XIIIe siècle. Initialement, cette suite a été conçue pour décrire la croissance d'une population de lapins, mais elle peut également être utilisée pour décrire certains motifs géométriques retrouvés dans la nature (coquillages, fleurs de tournesol...).

La suite de Fibonacci (x_n) est définie par: $\begin{cases} x_0 = 0 \\ x_1 = 1 \\ x_{n+2} = x_{n+1} + x_n \quad pour\ n \geqslant 0 \end{cases}$

À titre d'exemple, les 10 premiers termes de la suite de Fibonacci sont donc :

0 1 1 2 3 5 8 13 21 34

a. Créez un script qui afficher une liste **`fibo`** constituée des 15 premiers termes de la suite de Fibonacci puis l'affiche et le rapport du dernier du dernier élément de cette liste par l'avant-dernier.

b. Ce rapport tend-il vers une constante ? Si oui, laquelle ?

3. Résolution approchée d'une équation $x^3+x=3$ (E) : méthode de dichotomie

Soit f la fonction définie sur $\mathbb{R}$ par $f(x)=x^3+x-3$

Résoudre (E) revient donc à résoudre l'équation $f(x)=0$.

On admet que l'équation $f(x)=0$ admet une unique solution sur $[1;2]$.

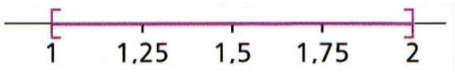
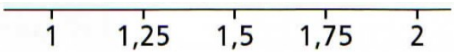
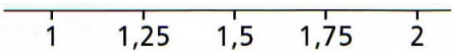
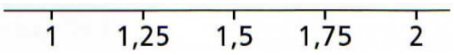
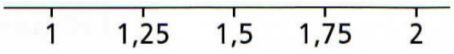
Nous la nommerons dans toute la suite x_0 .

Le but de ce qui suit est de proposer un algorithme dit de « dichotomie » permettant d'obtenir un encadrement de x_0 d'amplitude h donnée : c'est-à-dire d'obtenir deux réels a et b tels que : $x_0 \in [a;b]$ et $b-a \leqslant h$

3. On sait d'après ce qui précède que : $1 \leqslant x_0 \leqslant 2$.

   Après avoir calculé l'image par f du centre de l'intervalle $[1;2]$, déterminer un nouvel encadrement de x_0 . Quelle est son amplitude ?

4. Compléter le tableau suivant

	a	b	$[a;b]$ en gras	$\frac{a+b}{2}$	Signe de $f(a)\,f\left(\frac{a+b}{2}\right)$
Initialisation	1	2			
Étape 1					
Étape 2					
Étape 3					
Étape 4					

3. Quel encadrement de x_0 a-t-on à l'étape 4 ? Quel est son amplitude ?

4. Par ce processus, à chaque étape, que se passe-t-il pour l'amplitude de l'encadrement ?

5. Quelle est l'amplitude au bout de n étapes ?

6. Cette méthode permet-elle d'obtenir un encadrement d'amplitude aussi petite que l'on veut ?

7. Écrire l'algorithme précédent en langage naturel, puis en Python et le tester pour $h=0{,}01$.