FICHE N°6 : LES BOUCLES NON BORNÉES

A) Qu'est-ce qu'une boucle non bornée ?

Définition : Dans un algorithme, une **boucle** est une suite d'instructions que l'on répète (en boucle) un certain nombre de fois.
On dit que la boucle est **non bornée** lorsque l'on ne sait pas à l'avance le nombre d'itérations nécessaire pour effectuer la boucle. Un **test d'arrêt** est donc effectué à chaque passage de la boucle.
Au lycée, on utilise aussi les termes "boucle While" ou "boucle Tant que".

Important : Boucle non bornée = danger !
Le test d'arrêt est obligatoire et il faut s'assurer, avant de lancer le programme, que celui-ci sera bien atteint au bout d'un certain nombre d'itérations. Si la boucle utilise un compteur, il faut par exemple vérifier si la valeur du compteur change durant le programme pour permettre à la condition d'arrêt d'être vérifiée.
Si la condition d'arrêt n'est jamais vérifiée, alors on entre dans ce qu'on appelle une **boucle infinie** qui fera planter le programme.

Exemple n°1 : On souhaite connaître une valeur de x à 10^{-2} près pour laquelle $x = \sqrt{13}$. Autrement dit, on cherche x tel que $x^2 = 13$. Puisque $3^2 = 9$, on peut commencer le programme avec $x = 3$. Puisque l'on cherche une valeur approchée à 10^{-2} près, on va augmenter la valeur de x de 0,01 à chaque passage de la boucle et tester la condition d'arrêt.

Algorithme	Python

$x \leftarrow 3$
Tant que $x^2 < 13$ faire :
  $x \leftarrow x + 0,01$
Fin Tant que
Afficher x
	`1 x = 3`
	`2 while x**2 < 13:`
	`3  x = x + 0.01`
	`4 print(x)`

Ligne 1 : Initialisation de la variable x à 3.
Ligne 2 : À chaque passage de la boucle, la valeur de x est différente donc on vérifie à chaque itération si $x^2 < 13$. Si c'est le cas, la boucle s'effectue, sinon, la boucle s'arrête et le programme continue avec le prochain bloc.
Ligne 3 : Si la condition du test n'est pas vérifiée, on effectue les instructions de la boucle. Ici, il suffit d'augmenter la valeur de x de 0,01.
Ligne 4 : Cette dernière ligne n'est exécutée que lorsque la boucle est terminée, c'est-à-dire lorsque $x^2 \geqslant 13$. La valeur affichée répond au problème posé.

Important :

Dans l'exemple n°1, on ne peut pas écrire "Tant que $x^2 \neq 13$" et attendre que la boucle se termine exactement lorsque $x^2 = 13$. En effet, comme on cherche une valeur approchée de x à 10^{-2} près, la condition $x^2 = 13$ ne sera jamais vérifiée et la boucle se poursuivra de façon infinie, faisant planter le programme.

B) Symboles de comparaison :

Ce sont les mêmes que ceux utilisés avec les instructions conditionnelles :
== : ce symbole teste si deux valeurs sont égales.
!= : ce symbole teste si deux valeurs sont différentes.
> : ce symbole teste si la première valeur est strictement supérieure à la deuxième.
>= : ce symbole teste si la première valeur est supérieure ou égale à la deuxième.
< : ce symbole teste si la première valeur est strictement inférieure à la deuxième.
<= : ce symbole teste si la première valeur est inférieure ou égale à la deuxième.

C) À quel moment utiliser une boucle non bornée ?

On utilise une boucle non bornée lorsque l'on a une suite d'instructions à répéter un certain nombre de fois mais que l'on ne sait pas combien d'itérations vont être nécessaires. En revanche, avant de lancer le programme, il faut **obligatoirement** s'assurer que le test d'arrêt sera vérifié au bout d'un certain nombre d'itérations.

Exemple n°2 : On sait que $(-3)^3 = -27$ et $(-2)^3 = -8$. Comme la fonction cube est strictement croissante et continue sur l'intervalle $[-3\,;-2]$, on peut conjecturer qu'il existe une valeur de x telle que $x^3 = -20$. On pourra donc faire un programme permettant de trouver une valeur approchée de x.

Exemple n°3 : L'algorithme suivant ne s'arrêtera jamais.

```
n ← 4
Tant que 2 × n < 10 faire :
    n ← n - 1
Fin Tant que
```

En effet, au début, la valeur de 2 × n est 8 et elle est donc plus petite que 10. Donc l'instruction de la boucle sera exécutée. Et comme n devient de plus en plus petit, alors la condition sera toujours respectée et l'arrêt de la boucle n'aura jamais lieu.

D) Exercices

✓ Exercice 1 :

1) Écrire un algorithme permettant de trouver une valeur approchée de x à 10^{-2} près tel que $x^3 = -20$.
2) Programmer cet algorithme avec Python.
3) À l'aide de la calculatrice, calculer $\sqrt[3]{-20}$ et comparer le résultat obtenu par la calculatrice avec celui obtenu par le programme.

✓ Exercice 2 :

1) Programmer une fonction RacineCarree(n) avec Python qui détermine une valeur approchée à 10^{-2} près de $\sqrt{n}$ où n est un nombre positif.
2) Modifier la fonction précédente en ajoutant un argument p pour préciser la précision souhaitée. Par exemple, RacineCarree(42, 0.0001) renvoie $\sqrt{42}$ à 10^{-4} près.

✓ Exercice 3 :

Jonathan dispose de 10 000 € en banque avec un taux d'intérêt composé à 2 %. C'est-à-dire que, chaque année, Jonathan gagne 2 % du montant de l'année précédente. On suppose que le montant de 10 000 € correspond à l'année 0.

1) À la main ou à la calculatrice, démontrer que le montant en banque lors de la deuxième année (année 2) est égal à 10 404 €.
2) Écrire un programme avec Python qui calcule le montant en banque après la dixième année.
3) Jonathan souhaite avoir au moins 15 000 € en banque.
Écrire un programme qui utilise une boucle non bornée pour déterminer le nombre d'années que Jonathan va devoir attendre.

✓ Exercice 4 :

On rappelle qu'un entier naturel est dit premier lorsqu'il possède exactement deux diviseurs distincts : 1 et lui-même. Par conséquent, 1 n'est pas premier.
On cherche à écrire un programme qui détermine si un entier naturel est un nombre premier. On définit la fonction ci-contre où n est un entier naturel.

```
1  def EstPremier(n):
2      i = 2
3      while n % i != 0:
4          i = i + 1
5      return i
```

1) Justifier que la boucle While utilisée n'est pas une boucle infinie.
2) Dans la ligne 2, pourquoi ne pas avoir commencé avec "i = 0" ou "i = 1" ?
3) Comment utiliser cette fonction pour déterminer si n est premier ou non ?