Algorithmique 4
Boucles non bornées

Boucle non bornée : Tant que

On utilise une boucle non bornée (Tant que) pour répéter un bloc d'instructions dont on ne connaît pas à l'avance le nombre de répétitions.
La boucle se répète tant qu'une condition est vérifiée, et s'arrête dès que la condition n'est pas vérifiée.

Exercice 1

On cherche à déterminer le plus petit entier naturel n tel que n^3 soit supérieur ou égal à 500.

1. Faire tourner à la main l'algorithme suivant :

   $n \leftarrow 0$
   Tant que $n^3 < 500$
   $\quad n \leftarrow n + 1$
   Fin tant que
   Afficher n

n											
n^3											
Test $n^3 < 500$											

2. Quelle est la valeur de n affichée en sortie ? En déduire la réponse au problème.

   . . . . . .
   . . . . . .

Boucle non bornée en Python

Langage naturel	Langage Python
Tant que *condition* *instructions* Fin Tant que	while *condition* : *instructions*

Remarque

En Python, c'est l'indentation (décalage horizontal) d'une ou plusieurs lignes d'instructions qui indique la fin de la boucle.

Rappel des principaux tests en Python :

Instruction Python	`a==b`	`a!=b`	`a<b`	`a<=b`	`a>b`	`a>=b`
Teste la condition	$a = b$	$a \neq b$	$a < b$	$a \leqslant b$	$a > b$	$a \geqslant b$

Fonction Python associée l'exercice 1 (elle est sans argument)

```
def Ex1():
  n=0
  while n**3<500 :
    n=n+1
  return(n)
```

Exercice 2 (fonction mystère)

```
def mystere(n):
  while n>=7 :
    n=n-7
  return(n)
```

1. Faire tourner à la main cet algorithme pour $n = 48$. Que renvoie `mystere(31)` ?
2. Que renvoie la fonction `mystere` pour les entiers suivants : 10 ;21 ;12 ; 7 ?
3. Expliquer ce que revoie la fonction de façon générale.

Exercice 3

On place 1000 euros à un taux d'intérêts annuel de 2%.

1. Quel est le coefficient multiplicateur d'une hausse de 2% ?
2. Calculer le montant obtenu au bout d'un an.
3. Compléter la fonction Python pour que `duree(1000)` renvoie le nombre d'années de placement à partir duquel le montant obtenu devient supérieur ou égal à 1200.

```
def duree(montant):
  n=.........
  while montant.........:
    montant=.........
    n=.........
  return(n)
```

4. Adapter la fonction précédente pour obtenir le nombre d'années à partir duquel on double le montant initial.