Chapitre 6

Boucles bornées ou non

L'objectif de ce chapitre est de décrire des algorithmes en langage naturel ou dans un langage de programmation ; d'en réaliser quelques uns à l'aide d'un programme simple écrit dans un langage de programmation textuel et d'interpréter, compléter ou modifier des algorithmes plus complexes.

Le saviez-vous ?

Comme leur nom l'indique certaines boucles peuvent être non bornées c'est-à-dire qu'elles risquent de durer un temps infini et vous faire tourner en boucle ...

1. Boucles bornées

Définition

.....

.....

.....

.....

Exemple : en janvier 2019, M. Rapetou possède un compte bancaire crédité de 850 € ; tous les mois il y dépose 25 €. On souhaite connaître pour 2019 l'évolution de son compte mois par mois.

Algorithme

Compte ← 850
pour *i variant de 1 à 12* **faire**
  Compte ← *Compte* + 25
  Afficher *Compte*
fin

en Python

```
Compte = 850
for i in range(1,13):
        Compte = Compte + 25
        print("Au mois ",i," le compte s'élève à
: ",Compte," euros.")
```

a) Compteur

Afin de compter les itérations, le nombre de fois que l'on exécute le bloc d'instructions, ces boucles sont munies d'une variable compteur que l'on peut utiliser dans les instructions. Ce compteur est initialisé en début de boucle ensuite il s'incrémente automatiquement à chaque itération jusqu'à la valeur n.

Exemple : pour obtenir une table d'addition, de multiplication, le carré d'une série de nombres entiers, rien de plus simple

Table d'addition de 6 entre 15 et 20

```
for i in range(15,21):
        s = 6 + i
        print("6 + ",i," = ",s)
```

Carré des entiers compris entre 33 et 37

```
for i in range(33,38):
        carre = i**2
        print("Le carré de ",i," est ",carre)
```

Remarques :

- en Python, comme pour les instructions conditionnelles, dans une boucle, le début du bloc d'instructions concernées par la boucle, est signifié par les deux points : et l'indentation des lignes qui suivent ;
- la fin de l'indentation marque la fin du bloc d'instructions concernées par la boucle ;
- il faut être attentif au paramétrage du compteur, notamment avec l'utilisation de la fonction **range()** .

b) Fonction range()

La fonction **range()** génère par défaut une séquence de nombres entiers de valeurs croissantes et différents d'une unité. Si vous appelez la fonction **range()** avec un seul argument, la liste contiendra un nombre de valeurs égal à l'argument fourni mais commençant à 0 et finissant à $n-1$.

Exemple : ici la fonction **range()** *permet de constituer des listes, on peut constater l'influence des paramètres :*

list(range(6)) ... *[0, 1, 2, 3, 4, 5]*
list(range(5,12)) ... *[5, 6, 7, 8, 9, 10, 11]*
list(range(7,26,4)) ... *[7, 11, 15, 19, 23]*
list(range(37,9,-5)) ... *[37, 32, 27, 22, 17, 12]*

2. Boucles non bornées

Définition

.....

.....

.....

.....

Exemple : au jeu de plateau des petits chevaux, pour sortir un cheval de l'écurie on doit faire un 6. Tant que cette valeur n'est pas obtenue on doit relancer le dé. L'algorithme proposé ci-après permet de simuler cette situation et de connaître le nombre de lancers nécessaires pour le précieux sésame.

Algorithme

$n \leftarrow 1$
$de \leftarrow$ nb entier aléatoire de 1 à 6
tant que $de \neq 6$ **faire**
  $de \leftarrow$ nb entier aléatoire de 1 à 6
  $n \leftarrow n+1$
fin
Afficher Nb de lancers = n

en Python

```
from random import*

n = 1
de = randint(1,6)
while de != 6 :
        de = randint(1,6)
        n = n+1
print("Nb de lancers = ",n)
```


Il faut être très vigilant à la condition retenue pour arrêter l'exécution du programme car si elle reste toujours vraie alors le programme ne s'arrêtera jamais !

Exemple : on simule le tirage d'un dé à six faces numérotées de 1 à 6 ; soit la condition : Tant que la face est plus petite que 7 alors relancer le dé ; ici le programmera ne s'arrêtera pas.

Algorithme

$n \leftarrow 1$
$de \leftarrow$ nb entier aléatoire de 1 à 6
tant que $de < 7$ **faire**
  $de \leftarrow$ nb entier aléatoire de 1 à 6
  $n \leftarrow n + 1$
  Afficher Nb de lancers $= n$
fin

en Python

```
from random import*

n = 1
de = randint(1,6)
while de < 7 :
        de = randint(1,6)
        n = n+1
        print("Nb de lancers = ",n)
```

Remarque : on rentre le **print** dans la boucle du **Tant que** sinon rien ne s'afficherait, comme cela on peut constater que le programme tourne en permanence.

3. Caisse à outils

a) Savoir-faire

Comprendre et écrire une boucle bornée $\Longrightarrow$ on identifie les instructions qui vont se répéter un nombre connu de fois que l'on insère dans une boucle Pour (For) avec comme paramètre le nombre d'itérations. Pour connaître la valeur des variables en fin d'algorithme on établit un tableau ayant pour tête de colonnes le compteur et les variables. À chaque passage dans la boucle on ajoute une ligne au tableau et l'on change l'état du compteur et des variables.

Application : pour la préparation d'une compétition, Léo suit un planning d'entraînement sur 12 semaines. La première semaine il parcourt 21 km puis chaque semaine la distance parcourue augmente de 7 km.

1. *Quelle est la distance initiale ? Quelle est l'évolution d'une semaine sur l'autre ?*
2. *Quelle est la distance parcourue la 3e semaine ? la distance totale parcourue en fin de 3e semaine ?*
3. *Traduire cette situation par un algorithme. Puis programmer cet algorithme en Python.*
4. *Quelle est la distance totale parcourue pendant cette préparation ?*

Comprendre et écrire une boucle non bornée $\Longrightarrow$ on identifie les instructions qui vont se répéter un certain nombre de fois que l'on insère dans une boucle Tant que (While) avec comme paramètre la condition à vérifier. Pour connaître la valeur des variables en fin d'algorithme on établit un tableau ayant pour tête de colonnes la condition et les variables. À chaque passage dans la boucle on teste la condition qui est vérifié ou non.

Application : une entreprise de forage facture le premier mètre creusé 100 €. Le prix de chaque nouveau mètre creusé augmente de 35 €.

Algorithme 1

```
m ← 100
s ← m
n ← 1
tant que s ⩽ 3 500 faire
    m ← m + 35
    s ← s + m
    n ← n + 1
fin
Afficher n, s
```

Algorithme 2

```
Saisir h
m ← 100
s ← m
n ← 1
tant que n ⩽ h faire
    m ← m + 35
    s ← s + m
    n ← n + 1
fin
Afficher n, s
```

1. *Quelle est le prix du 3e mètre creusé ? le prix total d'un puits de 3 m de profondeur ?*
2. *Un client dispose de 3 500 € pour un creuser un puits, il souhaite connaître la profondeur maximale qui pourra atteindre. Lequel des algorithmes ci-dessus correspond à cette situation ?*
3. *Un propriétaire terrien sait que la nappe phréatique se situe à 10 m de profondeur, il souhaite connaître le prix du forage du puits. Lequel des algorithmes ci-dessus correspond à cette situation ?*
4. *Programmer les algorithmes en Python et les tester.*

4. Évaluations

Devoir en temps libre n° 6 : Boucles bornées ou non

Il est rappelé que la qualité de la rédaction, la clarté et la précision des raisonnements entreront pour une part importante dans l'apprécitaion des copies. Le barème est donné à titre indicatif. Le sujet sera rendu avec la copie.

Exercice n°1 : ***Pour ou Tant que ?***

1. On considère l'algorithme 1 ci-dessous. Expliquer ce qu'il réalise.

Algorithme 1

```
u ← 5
pour i allant de 1 à 5 faire
    u ← 2 × u + 3
fin
Afficher résultat = u
```

Algorithme 2

```
u ← 5
i ← 1
tant que ... faire
    ...
fin
Afficher résultat = u
```

2. Recopier et compléter l'algorithme 2 ci-dessus pour qu'il donne le même résultat que le précédent.
3. Traduire ces deux algorithmes en langage Python.
4. Mise en application : la suite de Fibonacci est définie de la manière suivante :
   - le premier et deuxième terme de la suite sont égaux à 1 ;
   - tous les termes suivants se calculent en additionnant les deux précédents.

   Le 3^e est $1 + 1 = 2$, le 5^e est $2 + 3 = 5$, le 6^e est $3 + 5 = 8$, ainsi de suite ...

   Écrire deux algorithmes, l'un avec une boucle bornée, l'autre avec une boucle non bornée, qui permettent de calculer le 100^e terme.
5. Traduire ces deux algorithmes en langage Python puis les tester. Quelle est la valeur obtenue ?
6. Modifier ces algorithmes puis ces programmes pour calculer le terme d'un rang quelconque souhaité.