5 Les boucles

5.1 Introduction et définition

On veut afficher la table de multiplication de 7, de 7×1 à 7×4 :

```
Algorithme : affichage de la table de 7
Variables : -
DEBUT
Afficher : "7 fois 1 =", 7*1
Afficher : "7 fois 2 =", 7*2
Afficher : "7 fois 3 =", 7*3
Afficher : "7 fois 4 =", 7*4
FIN
```

Imaginons maintenant qu'on veuille aller jusqu'à 7×100. Pour éviter de devoir écrire 100 instructions, on a recours à une boucle

Définition

Une boucle (ou structure itérative), est une structure permettant de répéter plusieurs fois l'exécution d'un bloc d'instructions. Il en existe deux en Python : la boucle "Tant que" et la boucle "Pour".

5.2 Boucle « Tant que »

La boucle « tant que » permet de répéter un bloc d'instructions tant qu'une condition est vérifiée. Lorsque la condition n'est plus vérifiée, on sort de la boucle. Une telle structure se formalise de la façon suivante :

En pseudo-code :

```
Tant Que condition :
    instructions
Fin Tant Que
```

En Python :

```
while condition :
    instructions
```

Exemple (1).

En pseudo-code :

```
Algo : affichage de 1 à 10
Variable numérique : i
DEBUT
i ← 1
Tant Que i ≤ 10
    Afficher : i
    i ← i + 1
Fin Tant Que
Afficher : "Fin de la boucle"
FIN
```

En Python :

```
# Algo : affichage de 1 à 10


i = 1
while i <= 10 :
    print(i)
    i = i + 1

print("Fin de la boucle")
```

Exemple (2).

En pseudo-code :

```
Algorithme : lancer de dé
Variables num : de, compteur
DEBUT
de ← 0
compteur ← 0
Tant Que de ≠ 6 :
    de ← randint(1,6)
    Afficher : "score", de
    compteur ← compteur + 1
Fin Tant Que
Afficher : "nb de lancers : ",
    compteur
FIN
```

En Python :

```
# Algorithme : lancer de dé
from random import *

de = 0
compteur = 0
while de != 6 :
    de = randint(1,6)
    print("score : ", de)
    compteur = compteur + 1

print("nb de lancers : ",
    compteur)
```

Remarques.

- **Attention**, si la boucle « Tant que » est mal utilisée, elle peut donner lieu à une exécution infinie. Pour que l'exécution de la boucle « Tant que » se termine, il faut qu'à un moment la condition devienne fausse. En particulier, les instructions dans la boucle doivent modifier la variable utilisée dans le test.
- Une fois encore, il ne faut pas oublier le " :", ni d'indenter le bloc d'instructions.
- La boucle « Tant que » est une boucle non déterministe : on ne sait pas forcément à l'avance combien d'itérations seront effectuées. Il arrive qu'elle ne soit pas exécutée du tout lorsque la condition est fausse dès le départ.

5.3 Boucle « Pour »

La boucle « Pour » utilise une variable, que l'on appelle compteur, qui sert à repérer le nombre d'itérations déjà effectuées. Le compteur part d'une valeur de départ et augmente à chaque itération d'un nombre fixe, appelé pas, jusqu'à atteindre une valeur d'arrivée. Pour chaque valeur du compteur, le bloc d'instructions est exécuté une fois.

La boucle « Pour » se formalise de la façon suivante :

En pseudo-code :

```
Pour compteur Allant De depart À arrivee (exclue) Par Pas
    De pas:
    instructions
Fin Pour
```

En Python :

```
for compteur in range(depart, arrivee, pas) :
     instructions
```

Exemple.

- **En pseudo-code :**

```
Algorithme : affichage des nombres pairs de 0 à 100
Variable numérique : compteur
DÉBUT
Pour compteur Allant De 0 À 101 (exclu) Par Pas DE 2 :
    Afficher : compteur
Fin Pour
FIN
```

- **En Python :**

```
# Algorithme : affichage des nombres pairs de 0 à 100
for compteur in range(0,101,2) :
    print(compteur)
```

Remarques.

- La boucle "Pour" n'est utilisée que lorsque le nombre d'itérations est connu dès le départ.
- Attention, en python, la valeur de départ est toujours incluse, mais **la valeur d'arrivée est toujours exclue**.
  Ainsi, si l'on écrit `for compteur in range(a,b)`, le compteur ira de `a` à `b-1`.
  En pseudo-code, il faut préciser (systématiquement) si `b` est exclu ou inclus.
- Dans la boucle "Pour" la valeur du compteur est modifiée automatiquement.

On peut ne pas préciser de valeur pour le pas. La valeur par défaut du pas est de 1.
En pseudo-code, cela donne :

```
Pour compteur Allant De depart À arrivee (exclue) :
```

et en Python :

```
for compteur in range(depart, arrivee) :
```

En Python, on peut ne pas donner de valeur de départ, la valeur de départ par défaut étant 0. Par exemple si on écrit :

```
for compteur in range(7) :
```

le compteur ira de 0 à 6 par pas de 1.

Remarque.
On peut imbriquer plusieurs structures (itératives ou autres), toujours en faisant très attention à indenter correctement.

5.4 Exercices

Exercice 28 (Pour se tester).

1) À quoi sert une boucle ?
2) Quels sont les deux types de boucle en Python ?
3) Quelle est la différence entre les deux types de boucle en Python ?
4) Comment faire chacun des deux types de boucle ?

Exercice 29.
Traduire en Python l'algorithme donné ci-dessous en pseudo-code, puis le tester :

```
Algorithme : calcul mental
Variables numériques : nb1, nb2, resultat_saisi
DEBUT
nb1 = randint(1, 10)
nb2 = randint(1, 10)
Afficher : nb1, " * ", nb2, " = ? "
Saisir : resultat_saisi
Tant Que resultat_saisi ≠ nb1 * nb2 :
    Afficher : "Le résultat est incorrect, essayez encore : "
    Saisir : resultat_saisi
Fin Tant Que
Afficher : "Bravo !"
FIN
```

Solution de Fahim

```
#Exercice 29; Algorithme: calcul mental
from random import *

def calculMental():
    nb1 = randint(1,10)
    nb2 = randint(1,10)
    print(nb1," * ", nb2," = ? ")
    resultat_saisi = int(input())
    while resultat_saisi != nb1 * nb2 :
        print("Le résultat est incorrect, essayez encore : ")
        resultat_saisi = int(input())
    print("Bravo!")


calculMental()
reponse = str(input("Voulez-vous rejouez ? (O ou N)"))
while reponse == "O" or reponse == "o" :
    calculMental()
    reponse = str(input("Voulez-vous rejouez ? (O ou N)"))
print("Fin du jeu")
```

Exercice 30.
Les deux algorithmes ci-dessous sont censés afficher la table de multiplication de 3 (de 3×1 à 3×10). Les reproduire puis les corriger en testant autant que nécessaire.

1)

```
# Algo : table de 3 avec for
for i in range(10):
print("3*", i, " = ", 3*i)
endfor
```

2)

```
# Algo : table de 3 avec while
while i ≤ 10:
print("3*", i, " = ", 3*i)
i = i + 1
```

Exercice 31.

a) Écrire en pseudo-code puis en Python une **procédure** prenant en paramètre une valeur `val` et affichant les carrés des nombres entre 1 et `val` (inclus). Cette procédure devra inclure une boucle Tant Que.

b) Même exercice avec une boucle Pour.

Solution d'Ethan

```python
def afficher_carres(val):
    i = 1
    while i <= val:
        carre = i ** 2
        print("Le carré de", i," est", carre,)
        i = i + 1

valeur_limite = int(input("Entrez une valeur : "))
afficher_carres(valeur_limite)
```