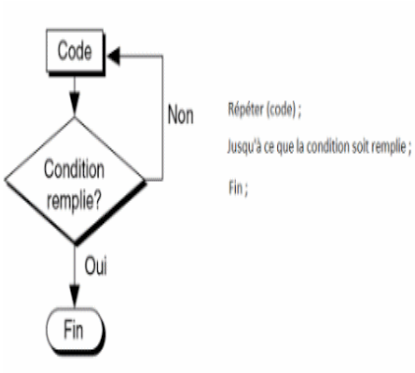
Boucles inconditionnelles – FOR - WHILE

LES BOUCLES WHILE

1. Introduction

Comme nous avons pu le voir, une boucle est un système d'instruction permettant de répéter un certain nombre de fois toute une série d'opérations. ***Le mot "while" signifie "tant que" en anglais.***



Exemple:

```
>>> a = 0
>>> while (a < 7):        # (n'oubliez pas le double point !)
...     a = a + 1         # (n'oubliez pas l'indentation !)
...     print(a)
```

L'instruction `while` de la 2ème ligne indique à Python qu'il faut ***répéter continuellement le bloc d'instruction qui suit***, tant que le contenu de la variable a reste inférieur à 7.

Remarques:
✔ la variable évaluée dans la condition doit exister au préalable (affectation d'une valeur)
✔ si la condition est fausse au départ, le corps de la boucle n'est jamais exécuté
✔ si la condition reste toujours vraie, alors le corps de la boucle est répété indéfiniment !!

```
>>> n = 3
>>> while n < 5:
...     print("hello !")
```

boucle sans fin !!

2. L'algorithme d'une boucle WHILE

Exercice 1:

Déterminer le plus petit entier n tel que $n! > 10^{10}$

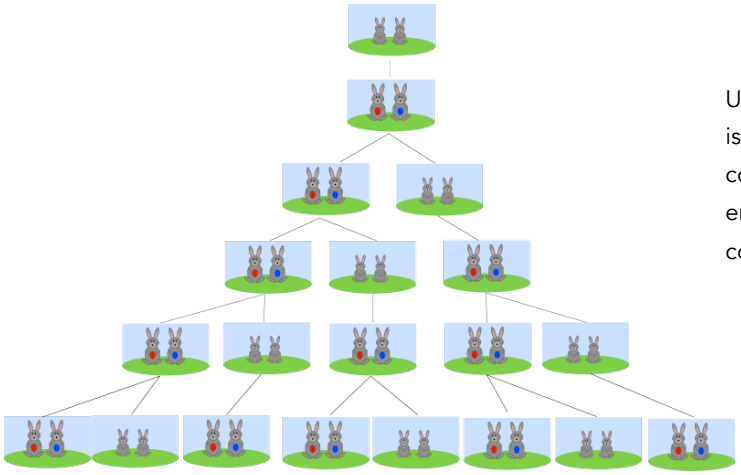
- Algorithme :
  $n \leftarrow 1$; $f \leftarrow 1$ # f vaudra $n!$ à (presque) tout moment
  tant que $f < 10^{10}$ **faire**
    $n \leftarrow n+1$
    $f \leftarrow f \times n$

```
# Programme n!>10 puissance 10
# Nicolas Pernot

n,f=1,1

while f<10**10:
  n=n+1
  f=f*n
print("pour n=",n,"n! =",f)
print("le plus petit entier n tel que n!>10**10 est: ",n)
```

Boucles inconditionnelles – FOR - WHILE

Exercice 2:

Ecrire un petit programme permettant d'afficher les dix premiers termes d'une suite appelée "***suite de Fibonacci***". il s'agit d'une suite de nombre dont chaque terme est égal à la somme des deux termes qui le précèdent. On souhaite afficher le résultat en ligne.



Un homme met un couple de lapins dans un lieu isolé de tous les côtés par un mur. Combien de couples obtient-on en un an si chaque couple engendre tous les mois un nouveau couple à compter du troisième mois de son existence ?

```
# Programme suite de Fibonacci
# Nicolas Pernot


print("Suite de Fibonacci:")
print("1", end=" ")

a,b=1,1        #a et b servent au calcul des termes successifs
c=1            #c est un simple compteur

print(b,end=" ")          #affichage du 1er terme U1

while c<10:
  a,b=b,a+b
  c=c+1
  print(b,end=" ")
```

```
# Programme suite de Fibonacci
# Nicolas Pernot


print("Suite de Fibonacci:")
print("1", end=" ")

a,b=1,1
c=1

print(b,end=" ")

while c<10:
  b=a+b
  a=b-a
  c=c+1
  print(b,end=" ")
```

Exercice 3:

Ecrire un programme qui affiche les 20 premiers termes de la table de multiplication par 7, en signalant au passage (à l'aide d'une astérisque) ceux qui sont multiples de 3, par exemple: 7, 14, 21*, 28, 35, 42*, 49…

```
# Programme table de multiplication par 7
# Nicolas Pernot
# Octobre 2013


i=1       #i est un compteur de 1 à 20

while i<21:
  t=i*7
  print(t,end=" ")
  if t%3==0:
    print("*",end=" ")
  i=i+1    # incrémentation du compteur
```

Boucles inconditionnelles – FOR - WHILE

3. Comment démontrer qu'une boucle se termine ?

SAVOIR-FAIRE Démontrer qu'une boucle se termine effectivement

On identifie un variant, autrement dit une expression (c'est souvent le simple contenu d'une variable)

- qui est un entier positif tout au long de la boucle,
- et qui diminue strictement après chaque itération.

On peut alors en conclure que la boucle termine.

Exercice 1:

- Montrer que le programme suivant va terminer :

```
i, cpt = 5000, 0
while i>0:
    i = i//2
    cpt = cpt+1
```

Que valent *i* et *cpt* après exécution ?

- Que se passe-t-il à l'exécution de ce programme ?

```
n = 10
while n>1:
    print(n)
    if n%2 ==0:
        n = n/2
    else:
        n = 3*n+1
```

Que se passe-t-il si $n = 1000$ au départ ? Et pour n quelconque ?

Exercice 2:

On considère le programme suivai

```
c = 0
while p > 0:
    if c == 0:
        p = p - 2
        c = 1
    else:
        p = p + 1
        c = 0
```

Partant de l'état initial p = 5 on obtient successivement les états suivants :

	c	p
1	0	5
2	1	3
3	0	4
4	1	2
5	0	3
6	1	1
7	0	0

On note p_i et c_i les valeurs des variables p et c après l'itération i.