Tâches répétitives sur Python

L'instruction while

Par Michael SPORTICHE

Introduction

Sur Python, on peut écrire un programme permettant de répéter plusieurs fois des tâches sans savoir combien de fois on doit les répéter avant que ce programme soit réalisé.

On utilise alors l'instruction while qui signifie en français tant que.

On ne connait pas le nombre de répétitions à l'avance et on parle donc de boucle non bornée.

L’instruction while

SYNTAXE :

```
while condition :
        Instruction(s)
```

On écrit le while ainsi que la condition qui s’ensuit en rajoutant les deux points, ce qui permet d’indenter vers la droite les instructions.

Pour sortir de la boucle, il faudra revenir a la ligne sans indenter (c’est-à-dire sans décalage vers la droite)

Exemples

```
x=0
while(x<30):
    x=x+3
    print(x)
```

Dans ce programme on affecte à la variable x, la valeur de zéro. En utilisant l'instruction while on demande au programme de répéter les instructions x=x+3 et print(x) jusqu'à que la condition ne soit plus vérifiée.

Tant que la condition est vérifiée, la boucle continue.

Exemples

```
x=0
while(x<30):
    x=x+3
    print(x)
```

En premier x est égal a 0 donc il est plus petit que 30 et la condition est respectée. Python opère alors x =0 +3 ce qui donne 3, avant d'afficher à l'utilisateur 3. Le 3 est toujours plus petit que 30 donc Python recalcule x=3+3 avant d'afficher à l'utilisateur 6 , ce 6 respectant encore la condition, la boucle continue et ainsi de suite. Lorsque Python affiche 30, le 30 n'étant pas strictement inferieur a 30, la condition n'est plus respectée et la boucle prend fin.

A la fin du programme , les valeurs qui s'affichent sont : 3 6 9 12 15 18 21 24 27 30.

Exemples

```
x=0
while(x<30):
    x=x+3
print(x)
```

Ce programme qui ressemble n'est pas tout a fait le même. Le print(x) n'est pas indenté. Il n'appartient donc pas à la boucle contrairement à l'exemple précèdent.
Le processus de calcul sera le même mais seul la dernière valeur de x s'affichera ici et non toutes les valeurs, c'est-à-dire seulement 30.

Afficher un compteur

```
x=0
n=0
while(x<30):
    n=n+1
    x=x+3
print(n)
```

Dans ce programme qui est sensiblement le même, on incrémente de 1la valeur n à chaque tour et on affiche le dernier n. Cela nous permet de compter combien de fois la boucle est réalisée, avant que la condition ne soit plus respectée.

Attention !

```
x=0
while(x<30):
   print(x)
```

Dans un cas pareil, il n'y a pas d'incrémentation de x, et le programme Python ne peut plus vérifiée la condition.

La boucle devient infinie et ne s'arrêtera donc jamais. Ici 0 sera affichée à l'infinie.

Ecrire une table de multiplication

Avec l'instruction while on peut par exemple facilement écrire des tables de multiplications.

Ecrivons par exemple la table de 4 :

```
c=0
while(c<10):
    c=c+1
    print(c,"x 4 =",c*4)
```

Dans ce programme on affecte 0 à la variable c. Tant que la valeur de c est inférieur à 10, on incrémente de 1 la valeur de c, avant d'afficher cette même valeur * 4 et d'en donner le résultat.

Ecrire une table de multiplication

```
c=0
while(c<10):
    c=c+1
    print(c,"x 4 =",c*4)
```

Ce que fait Python :

c=0 , condition vérifiée, donc c= 0+1. La valeur de c est de 1,donc le programme affiche 1x4=4. (Le x4= sera toujours affichée, alors que le c et le c*4 seront la valeur d'après l'incrémentation.)

c=1, condition vérifiée, donc c=1+1. La valeur de c est de 2 donc le programme affiche 2*4=8

c=2, condition vérifiée, donc c=2+1. La valeur de c est de 3 donc le programme affiche 3*4 = 12 … Et ainsi de suite, jusqu'à que c=10 et donc que la condition ne soit plus vérifiée.

Ecrire une table de multiplication

On peut réaliser ce programme directement dans la console de calcul:

```
>>> c=0
>>> while(c<10):
      c=c+1
      print(c,"x 4 =",c*4)

1 x 4 = 4
2 x 4 = 8
3 x 4 = 12
4 x 4 = 16
5 x 4 = 20
6 x 4 = 24
7 x 4 = 28
8 x 4 = 32
9 x 4 = 36
10 x 4 = 40
```

Ecrire une table de multiplication

```
c=0
while(c<10):
    c=c+1
    print(c,"x 4 =",c*4)
```

Si on modifie la condition du while, on pourra afficher plus ou moins de valeurs à notre table de 4.

```
while(c<20):  #On affichera alors les vingt premiers termes de la table au lieu des dix.
```

```
while(c<5): #Dans ce cas, on affichera seulement les cinq premiers termes de la table.
```

Tables de multiplications

- On peut écrire une fonction qui permet de réaliser plusieurs tables de multiplications en fonction de celle que l'on souhaite.
- On utilise alors l'instruction if, suivie de plusieurs elif représentant chacun une table différente.

```
def table(a):
    if (a==1):
        c=0
        while (c<10):
            c=c+1
            print(c,"x 1 =",c*1)
    elif(a==2):
        c=0
        while(c<10):
            c=c+1
            print(c,"x 2 =",c*2)

    elif(a==3):
        c=0
        while(c<10):
            c=c+1
            print(c,"x 3 =",c*3)
    elif(a==4):
        c=0
        while(c<10):
            c=c+1
            print(c,"x 4 =",c*4)
```

Tables de multiplications

- Cette fonction possède un seul argument a. En fonction du a que l'on choisie, le programme réalisera la table que l'on souhaite.
- Dans ce cas les tables sont de 1 a 4. On aurait pu continuer selon ce schéma sur beaucoup d'autres.
- Si l'on veut afficher bien plus ou moins que les 10 premiers calculs de la table, il suffit simplement de modifier la condition dans le while.

```
def table(a):
    if (a==1):
        c=0
        while (c<10):
            c=c+1
            print(c,"x 1 =",c*1)
    elif(a==2):
        c=0
        while(c<10):
            c=c+1
            print(c,"x 2 =",c*2)

    elif(a==3):
        c=0
        while(c<10):
            c=c+1
            print(c,"x 3 =",c*3)
    elif(a==4):
        c=0
        while(c<10):
            c=c+1
            print(c,"x 4 =",c*4)
```

Tables de multiplications

- Pour exécuter la fonction, on écrit dans la console le nom de la fonction en remplaçant l'argument par la valeur que l'on souhaite.

Si on veut afficher la table de 3 on écrira alors : >>>table(3) et le programme affichera alors la table de 3.

```
>>> table(2)
1 x 2 = 2
2 x 2 = 4
3 x 2 = 6
4 x 2 = 8
5 x 2 = 10
6 x 2 = 12
7 x 2 = 14
8 x 2 = 16
9 x 2 = 18
10 x 2 = 20
>>> table(4)
1 x 4 = 4
2 x 4 = 8
3 x 4 = 12
4 x 4 = 16
5 x 4 = 20
6 x 4 = 24
7 x 4 = 28
8 x 4 = 32
9 x 4 = 36
10 x 4 = 40
>>>
```

La clause else

On peut ajouter la clause else, à la suite de l'instruction while. Elle s'exécutera quand la boucle while sera terminée, c'est-à-dire que la condition ne soit plus vérifiée.

```
c=0
while(c<10):
    c=c+1
    print(c,"x 4 =",c*4)
else:
print("To Be Continued")
```

Ce programme affichera To Be Continued après avoir finie d'afficher la table de quatre.