# TP ALGO 5 : LA BOUCLE NON BORNEE WHILE

Pour écrire certains programmes, il est parfois nécessaire de répéter une ou plusieurs instructions un nombre inconnu de fois. On utilise une boucle non bornée qui est parcourue jusqu'à ce qu'une certaine condition ne soit plus vérifiée. Tant que cette condition est vérifiée, la boucle continue.

On utilise une « boucle while » appelée « boucle non bornée ».

**Syntaxe d'une boucle non bornée**

```
while condition:
    instructions
```

**Tant que** la *condition* est vraie (elle a pour valeur True), on répète les *instructions* qui figurent dans le bloc **indenté** (décalé vers la droite)

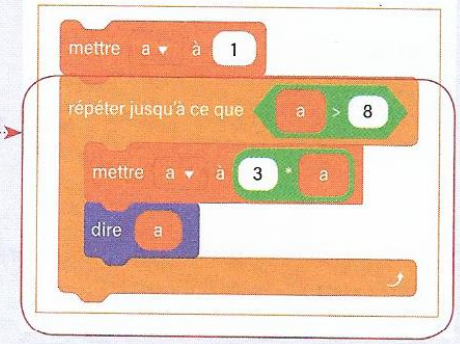
**Exemple 1**

```
a = 1
while a <= 8:
    a = 3*a
    print(a)
```

Exécutons « à la main » ce script :

a prend la valeur 1
On effectue le test a ≤ 8 : True (Vrai)
  Donc a vaut 3 et on affiche 3
On effectue le test a ≤ 8 : True (Vrai)
  Donc a vaut 9 et on affiche 9
On effectue le test a ≤ 8 : False (Faux)
On sort de la boucle.

« Tant que a ≤ 8 » correspond à « Répéter jusqu'à ce que a > 8 ».




**Exemple 2**

```
a = 1
while a <= 8:
    a = 3*a
print(a)
```

**Importance de l'indentation** (décalage vers la droite) :
Dans l'exemple 2, l'instruction `print(a)` n'est pas indentée. Elle n'appartient pas au bloc d'instructions à répéter et n'est exécutée qu'une fois la boucle terminée. Seule la dernière valeur de a, c'est-à-dire 9, est affichée.

**Exemple 3**

```
def seuil (n):
    S = 3
    while S <= n:
        S = 2*S
    return S
```

On exécute `seuil(20)`.
La variable S et le test (S<=120) prennent successivement les valeurs suivantes :

| S <= 20 | | True | True | True | False |
|---|---|---|---|---|---|
| S | 3 | 6 | 12 | 24 | |

L'appel `seuil(20)` renvoie donc 24,
qui est la 1re valeur de S supérieure à 20.

## Exercices d'application

**Exercice 1** : Décrire, comme dans l'exemple 1 ci-dessus, ce qui se passe lors de l'exécution « à la main » du script ci-contre.

```
a = 1
while a < 11:
    a = a + 4
    print(a)
```

**Exercice 2** : Avant exécution du script ci-contre, n a pour valeur 0 et P a pour valeur 1.

```
while P < 100:
    n = n + 1
    P = P * 5
```

1. On exécute le script à la main. Compléter le tableau suivant :

| n | 0 | 1 | | |
|---|---|---|---|---|
| P | 1 | | | |
| P<100 | True | | | |

2. On dit que la variable n est un compteur. Pourquoi ?

**Exercice 3** : Ecrire une fonction premier(n) qui renvoie True si l'entier n est premier, et False sinon.

**Exercice 4** : Dans une culture bactérienne constituée de $N$ bactéries, on suppose qu'à chaque seconde, 2 % des bactéries meurent.

1. Ecrire une fonction $bacteries(N, n)$ qui renvoie le nombre de bactéries encore vivantes après $n$ secondes, $n$ entier naturel.

2. Une culture contient 1 million de bactéries. Combien de bactéries sont encore vivantes après 10 secondes ? Arrondir le résultat à l'unité.

3. En écrivant une fonction appelée $moitié(N)$, déterminer le nombre de secondes qu'il faut au minimum pour que le nombre de bactéries soit strictement inférieur à la moitié de la population initiale. Ce temps dépend-il du nombre de bactéries au départ ?

**Exercice 5** : La conjecture de Syracuse est non encore démontrée à ce jour. Elle s'énonce comme suit : « Soit $n$ un entier naturel. Si $n$ est pair, on le divise par 2, sinon on le multiplie par 3 et on ajoute 1. On recommence ce processus avec le résultat obtenu. La conjecture de Syracuse affirme que l'on obtient toujours 1 au bout d'un nombre fini d'itérations.

1. Ecrire une fonction $Syracuse(n)$ qui répond à la conjecture et la tester pour plusieurs valeurs de $n$.

2. Modifier cette fonction afin de compter le nombre d'itérations nécessaires jusqu'à l'obtention de 1. Ce nombre d'itérations s'appelle le temps de vol de $n$.

3. Soit $N$ un entier donné. Ecrire une fonction $maximum(N)$ qui renvoie l'entier $n$ entre 1 et $N$ pour lequel le temps de vol est le plus grand, et le temps de vol correspondant.