# Informatique - Notes de cours 4

## Structures itératives - Boucles *while*

M. Marmorat

12 novembre 2024

Nous avons vu lors du cours précédent que pour répéter des instructions un certain nombre de fois on peut utiliser une boucle `for` . La syntaxe utilisant `range` fait clairement apparaître qu'on utilise une boucle `for` lorsqu'on sait à l'avance le nombre de fois qu'on veut répéter les instructions (par exemple pour les répéter $n$ fois, on peut utiliser `range ( )` ou `range ( , )`). Lorsqu'on ne sait pas à l'avance combien de fois on veut répéter des instructions, on peut utiliser une boucle `while` ("tant que"), on va alors répéter des instructions tant qu'une certaine condition est vérifiée.

# 1 Syntaxe des boucles `while`

La condition est exprimée à l'aide d'un booléen. La syntaxe est la suivante :

```
while condition:
    #bloc d'instructions
#instructions hors de la boucle while
```

Tant que le booléen exprimé par `condition` est vrai, Python exécute le blocs d'instructions à l'intérieur de la boucle `while` . Dès que `condition` devient faux, Python sort de la boucle et exécute les instructions situées hors de la boucle `while` .

**Exemple 1.** Que fait le script suivant ?

```
k = 0
while k < 5:
    print(k)
    k += 1
```

**Remarque 1.**
- Python n'entre pas dans la boucle si la condition n'est pas vraie initialement.
- Contrairement à la boucle `for` , la boucle `while` ne modifie pas naturellement la condition qu'elle teste. C'est donc à vous de modifier l'état de la condition, à l'intérieur de la boucle.

**Exemple 2.** Que fait le script suivant ?

```
k = 10
while k < 5:
    k = 2*k
print(k)
```

**Remarque 2.** Il faut s'assurer que la condition testée prendra la valeur `False` au bout d'un moment, sinon la boucle s'exécutera indéfiniment !

**Exemple 3.** Que fait le script suivant ?

```
compteur = 2
while compteur > 1:
    compteur = 2*compteur
    print(compteur)
```

**Exercice 1.** Écrire une fonction Python qui prend en entrée un entier $n$ et qui retourne la plus grande puissance de $2$ qui lui est inférieure. Par exemple pour l'entier $n = 9$ la fonction doit renvoyer $8$, et pour l'entier $n = 21$ la fonction doit renvoyer $16$.

## 2 Algorithmes de seuil

Les boucles `while` sont particulièrement utiles pour déterminer des seuils, c'est-à-dire pour déterminer le plus petit entier $n$ tel qu'une quantité $u_n$ est supérieure à un seuil fixé $M$.

**Exemple 4.** Supposons qu'un laboratoire de biologie dispose d'une population de souris. On note $u_n$ le nombre de souris de la population à la $n$-ème année, et l'on suppose que la suite $(u_n)$ est donnée par $u_0 = 50$ (50 souris la première année) et

$$\forall n \in \mathbb{N}, \quad u_{n+1} = 2u_n + 3$$

(chaque année la population double et 3 individus sont ajoutés - remarquez que la suite $(u_n)$ est une suite arithmético-géométrique). On souhaite déterminer la première année à partir de laquelle le nombre de souris dépassera 200. On peut utiliser le code Python suivant :

```
u = 50 # on stocke la valeur u_0 dans u
n = 0 # annee 0
while u < 200:
    # u represente u_n
    u = 2*u+3 #on met a jour la population u (recurrence)
    # u represente u_n+1
    n += 1 # on incremente l'annee
print(n)
```

**Exercice 2.** On rappelle que $n! = 1 \times 2 \times \cdots \times (n-1) \times n$.

1. Écrire un script Python calcule le plus petit nombre entier $n$ tel que $n! \geq 100$. De quel entier $n$ s'agit-il ?
2. Écrire une fonction Python `seuil_factorielle` qui prend en argument un nombre réel positif $M$ et qui renvoie le plus petit nombre entier $n$ tel que $n! \geq M$.

# 3 Boucles `for` vs boucles `while`

Une boucle `for` peut toujours être écrite à l'aide d'une boucle `while` , mais l'inverse n'est pas vrai. Par exemple le script suivant

```
for k in range(a, b):
    #vos instructions
```

peut-être remplacé par le script

```
k = a
while k < b:
    #vos instructions
    k += 1
```

Quand c'est possible, on préférera écrire une boucle `for` qu'une boucle `while` .

**Exercice 3.** Pour chacun des cas suivants, réécrire le programme avec l'autre boucle.

1.
```
somme = 0
for k in range(11):
    somme += k
```

2.
```
produit = 2
i = 3
while i < 20:
    produit *= i
    i += 2
```

# 4 Entrainement

**Exercice 4.** Écrire un programme qui calcule le plus petit $n$ tel que $(n+1) \times (n+3)$ soit supérieur à 12345.

**Exercice 5.** On note $(F_n)$ la suite de Fibonacci, définie par $F_0 = 0$ et $F_1 = 1$. Déterminer le plus petit entier $n$ tel que $F_n > 20$.

**Exercice 6.** Écrire un programme qui calcule le plus petit entier $n$ tel que $1^2 + 2^2 + 3^2 + \cdots + n^2 > 123456$.

**Exercice 7.** Écrire une fonction `puissance_de_deux` qui prenne en argument un entier $n$ et qui indique la plus grande puissance de $2$ par laquelle $n$ est divisible.

Par exemple `puissance_de_deux`(16)=4, `puissance_de_deux`(5)=0 et `puissance_de_deux`(12)=2.