

Mise à Niveau

Sébastien Gagné, Université d'Orléans

Cours 04

Compléments sur les itérables et les boucles itératives

Plan du cours 4

- La boucle "tant que"
- Parcours total d'un itérable
- Parcours partiel d'un itérable

La boucle "tant que" - *while*

Python propose, pour les instructions itératives, la structure *for*

```
1 for i in range(7, 20, 2)
2     print(i, end = '; ')
```

Le résultat produit est

```
7; 9; 11; 13; 15; 17; 19;
```

La variable *i* est initialisée à 7, puis progresse de 2 en 2, et cette progression continue tant que $i < 20$.

La boucle "tant que" - *while*

Le mot clé *while* va permettre d'écrire la boucle *for* précédente d'une autre façon. Le résultat produit est exactement le même.

```
1 for i in range(7, 20, 2)
2     print(i, end = '; ')
```

```
1 i = 7
2 while i < 20 :
3     print(i, end = '; ')
4     i = i + 2
```

Le codeur gère explicitement l'initialisation ($i = 7$), l'évolution ($i = i + 2$) et la valeur max ($i < 20$) de i , ce qui était implicite dans

for i in range(7, 20, 2)

La boucle "tant que" - *while*

Plus généralement, la boucle *while* permet de répéter un bloc d'instructions tant que la condition donnée est évaluée à *True*.

Attention, ceci implique que l'expression booléenne qui sert de condition finisse par être évaluée à *False* (sinon, boucle infinie...)

```
1 i = 0
2 while i < 5 :
3     print(i)
```

une boucle infinie

La boucle "tant que" - *while*

Le code est corrigé : la condition finira pas être évaluée à *False*, grâce à l'instruction

$$i = i + 1$$

```
1 i = 0
2 while i < 5 :
3     print(i)
4     i = i + 1
```

L'essentiel est de s'assurer que l'expression $i < 5$ ne soit pas toujours vraie.

La boucle "tant que" - *while*

Exemple : demander à l'utilisateur de saisir une lettre minuscule. La demande sera renouvelée **tant que la saisie n'est pas correcte**.

```
entrez une minuscule : A
entrez une minuscule : X
entrez une minuscule : 5
entrez une minuscule : $
entrez une minuscule : aze
entrez une minuscule : h
Merci ! Vous avez compris ! Vous avez saisi h
```

La boucle "tant que" - *while*

Voici un exemple de résolution

- initialisation du booléen de la boucle *while*
- question posée à l'utilisateur, réponse collectée
- actualisation du booléen selon la réponse donnée
- une fois le booléen passé à False, sortie de boucle et suite (fin) du programme.

La boucle "tant que" - *while*

```
1 # initialisation du booléen pour entrer dans la boucle
2 # on donne un nom explicite à cette variable booléenne
3
4 saisie_incorrecte = True
5
6 # la boucle de saisie
7
8 while saisie_incorrecte :
9
10     # 1. on demande une minuscule
11
12     reponse = input("entrez une minuscule : ")
13
14     # 2. on actualise le booléen
15     #   la saisie ne devient correcte que si :
16     #   - la longueur de la réponse est 1
17     #   - la réponse est entre 'a' et 'z'
18
19     saisie_incorrecte = not(len(reponse) == 1 and 'a' <= reponse <= 'z')
20
21 # instruction en sortie de boucle
22
23 print("Merci ! Vous avez compris ! Vous avez saisi", reponse)
```

Différents parcours d'un itérable

Selon le problème posé, on peut être amené à utiliser de différentes façons un itérable. Par exemple :

On demande à l'utilisateur une chaîne de caractères.

Problème 1

Le programme doit dire combien de 'a' sont présents dans la chaîne.

Problème 2

Le programme doit dire si la chaîne contient un 'a'.

Différents parcours d'un itérable

- Le problème 1 oblige à parcourir TOUTE la chaîne de caractères, et à tester chaque caractère.

On parle de parcours TOTAL.

- Pour résoudre le problème 2, Il suffit de parcourir la chaîne **jusqu'au premier** caractère 'a'.

On parle de parcours PARTIEL.

Un parcours total, quand il n'est pas nécessaire, peut occasionner des pertes non négligeables d'efficacité.

Parcours total d'un itérable

Une solution au Problème 1

```
1 chaîne = input("entrez une chaîne de caractères : ")
2 n = 0
3
4 for caractere in chaîne :
5     if caractere == 'a' :
6         n = n + 1
7
8 print(chaîne, "contient", n, "caractères 'a'.")
```

La chaîne est totalement parcourue, et la variable n compte le nombre de 'a'.

Parcours partiel d'un itérable

Une solution au Problème 2

```
1 chaîne = input("entrez une chaîne de caractères : ")
2
3 # un booléen qui dit si 'a' a été trouvé (initialisé à False)
4 trouve = False
5
6 # un int qui indique l'indice du caractère qu'on va examiner
7 i = 0
8
9 # on recherche un 'a' tant que :
10 # - on n'en a pas encore trouvé
11 #   (c'est-à-dire trouve == False ou encore not trouve)
12 # - on n'est pas encore arrivé à la fin de la chaîne
13 #   (c'est-à-dire i < len(chaîne))
14
15 while (not trouve and i < len(chaîne)) :
16     # on actualise trouve
17     trouve = (chaîne[i] == 'a')
18
19     # on actualise i pour la recherche suivante
20     i = i + 1
21
22 # instruction en sortie de boucle
23 if trouve :
24     print(chaîne, "contient le caractère 'a'.")
25 else :
26     print(chaîne, "ne contient pas le caractère 'a'.")
```