Chapitre 8

Programmation - Structures de contrôle

8.1 Les Boucles

Il est souvent nécessaire en programmation de devoir répéter un bloc d'instructions un certain nombre de fois, que ce soit pour parcourir une liste, pour faire une action de manière cyclique, etc. Ce nombre d'itérations est parfois connu en avance et doit parfois être déterminé au fur et à mesure des répétitions. En fonction de la situation, il existe plusieurs structures de contrôle permettant ces répétitions. Ces structures sont appelées des *boucles*.

8.1.1 La boucle for

Définition 1 :

En python, la boucle `for` permet de répéter un bloc d'instructions pour chaque élément d'un container (tableau, chaîne de caractères, etc.). Elle est introduite par le mot-clé `for` suivi d'un nom de variable, puis du mot-clé `in` et du container à parcourir, et enfin du symbole " :"

Exemple 1. *L'exemple suivant permet d'afficher 10x le mot "Coucou". À chaque itération de la boucle, le programme va exécuter la ligne contenant l'instruction* `print`.

Listing 8.1 – Boucle for - parcours tableau

```
1 for k in range(10):
2     print("Coucou")
```

Remarque 1 :

Dans les cas où l'on veut exécuter un bloc d'instructions un nombre précis de fois, la fonction `range(begin,end)` permet de générer un tableau de `begin` à `end`-1 qui peut ensuite être utilisé avec une boucle `for`. L'exemple 2 peut ainsi être écrit :

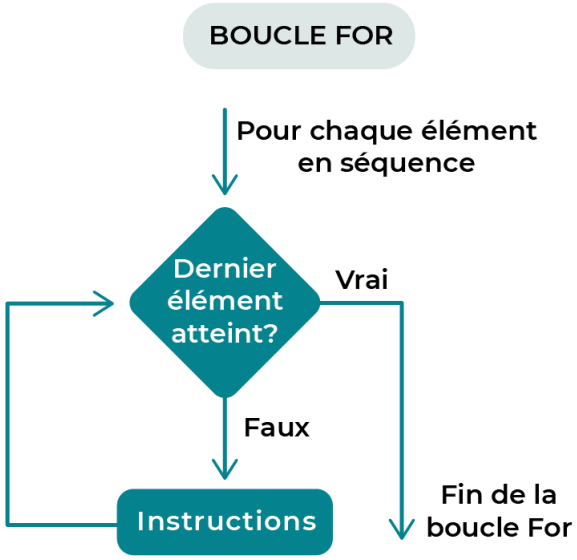

FIGURE 8.1 – Schéma fonctionnel d'une boucle for

Listing 8.2 – Boucle for - compteur trivial

```
1 maxCompteur = 10
2 for compteur in range(maxCompteur+1):
3     print(compteur)
```

8.1.2 La boucle while

Si la boucle `for` est la plus pratique à utiliser dans des cas précis, il existe une boucle plus modulable. C'est la boucle `while`.

> **Définition 2 :**
>
> La boucle `while` permet de répéter un bloc d'instructions tant qu'une condition n'est pas remplie. En python, elle est introduite par le mot-clé `while` suivi d'un test terminé par le symbole " :"

Exemple 2. *L'exemple suivant permet d'afficher un compteur allant de 1 à 10. À chaque début de boucle, le programme va évaluer la condition* `compteur <= maxCompteur` *puis exécuter le bloc d'instruction si elle vaut* `True` *et sortir de la boucle si elle vaut* `False`.

Listing 8.3 – Boucle while - compteur trivial

```
1 maxCompteur = 10
2 compteur = 1
3 while compteur <= maxCompteur:
4     print(compteur)
5     compteur += 1
```

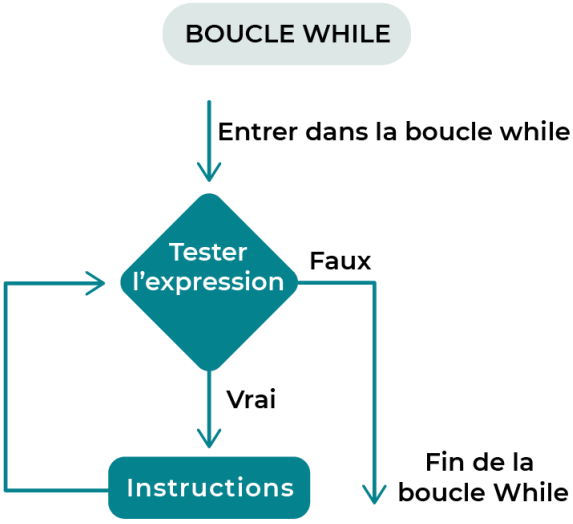

FIGURE 8.2 – Schéma fonctionnel d'une boucle while

Remarque 2 :

L'instruction `compteur += 1` est équivalente à `compteur = compteur + 1` et sert donc à *incrémenter* la variable `compteur` (à l'augmenter de 1).

8.1.3 Exercices sur les ordinateurs

1 Écrire un programme listant :

a) les nombres de 0 à 50.

b) les nombres pairs de 0 à 50.

a) les nombres de 50 à 100.

a) les nombres de 0 à 20, dans l'ordre décroissant.

2 Écrire un programme qui demande à l'utilisateur-trice de saisir un nombre entier et qui affiche la table de multiplication de ce nombre de à 10.

3 Calculer la somme des entiers de 1 à 100.

4 Écrire un programme permettant de réaliser la figure ci-dessous :

```
#########
#########
#########
#########
#########
#########
#########
#########
#########
```

5 Écrire un programme permettant de réaliser la figure ci-dessous :

```
#
##
###
####
#####
######
#######
########
#########
##########
```

6 Demander à l'utilisateur-trice de saisir un mot jusqu'à ce qu'iel saisisse un mot de plus de 5 caractères.

7 Demander à l'utilisateur-trice de saisir un nombre jusqu'à ce qu'iel saisisse un nombre pair.

8 Afficher le menu d'un programme jusqu'à ce que l'utilisateur-trice saisisse "q" pour quitter.

```
   -- Menu --
1. Option 1
2. Option 2
q. Quitter
Choisissez une option :
```

9 Écrire un programme qui demande à l'utilisateur-trice de saisir une phrase et qui affiche cette phrase sans les espaces.

10 Écrire un programme qui demande à l'utilisateur-trice de saisir un mot et qui affiche ce mot à l'envers.

11 Imaginer :

a) une boucle infinie, n'atteignant jamais sa condition de fin.

b) une boucle qui n'exécute jamais son bloc d'instructions.

12 Écrire un programme énumérant les 100 premiers nombres de la suite de Fibonacci.

13 Écrire un programme permettant d'additionner les 100 premiers entiers naturels (1, 2, ..., 100).

14 En vous inspirant de l'exercice 20, écrire un programme trouvant le minimum entre N nombres. Testez avec N=10.

8.2 Les conditions

Avec les concepts vus jusqu'ici, un programme exécuté 100 fois de suite donnera 100 fois les mêmes résultats, ce qui présente assez peu d'intérêt. Pour que l'exécution de notre programme dépende de différents paramètres, on utilise des structures de contrôle nommées *conditions*.

La condition est un élément de base d'algorithmique que l'on utilise au quotidien.

Exemple 3. *S'il fait beau, j'irai en montagne.*

Ces structures permettent d'exécuter ou de ne pas exécuter certaines instructions en fonction de certaines conditions et de changer ainsi le déroulement du programme.

Étapes	Instructions
1	Se lever
2	Regarder la météo
3.1	S'il fait beau, aller en montagne

8.2.1 L'instruction if

Un premier moyen d'exprimer des conditions en programmation est l'instruction if. En python, elle s'utilise de la manière suivante :

Listing 8.4 – Instruction if

```
1 if meteo == "beau":
2     print("Aujourd'hui, je vais en montagne.")
```

Définition 3 :

L'instruction `if` doit être suivie d'une expression retournant un booléen (`True` ou `False`) puis d'un symbole `:`. Cette expression peut être une variable contenant un booléen ou une opération retournant un booléen comme les opérateurs $<$, $>$, $==$ ou $!=$.

Dans cet exemple, si la variable `meteo` a été initialisée à `"beau"`, la phrase Aujourd'hui, je vais en montagne. va s'afficher. Si elle a été initialisée à une autre valeur, par exemple `meteo = "moche"`, rien ne s'affichera.

15 Quelle est la différence entre ces deux programmes ? Qu'afficheront-ils respectivement :

a) dans le cas `meteo = "beau"` ?

b) dans le cas `meteo = "moche"` ?

Listing 8.5 – Programme A

```
1 if meteo == "beau":
2     print("Aujourd'hui, ")
3     print("je vais en montagne.")
```

Listing 8.6 – Programme B

```
1 if meteo == "beau":
2     print("Aujourd'hui, ")
3 print("je vais en montagne.")
```

Réponse :

8.2.2 L'instruction else

L'instruction `if` peut être suivie d'une instruction `else` qui est exécutée lorsque le résultat du test est `False`. Par exemple, on pourrait préciser la situation précédente :

Exemple 4. *S'il fait beau, j'irai en montagne. Sinon, j'irai au cinéma.*

Notre algorithme deviendrait alors :

Étapes	Instructions
1	Se lever
2	Regarder la météo
3.1	S'il fait beau, aller en montagne
3.2	S'il ne fait pas beau, aller au cinéma

En python, cela s'écrirait de la manière suivante :

Listing 8.7 – Instruction if - else

```
1 if meteo == "beau":
2     print("Aujourd'hui, je vais en montagne.")
3 else:
4     print("Aujourd'hui, je vais au cinéma.")
```

8.2.3 L'instruction elif

Dans le cas d'une situation non binaire, on peut introduire des conditions supplémentaires qui sont examinées si les conditions précédentes ne sont pas remplies. On peut par exemple nuancer notre situation ainsi :

Exemple 5. *S'il fait beau, j'irai en montagne. S'il ne fait pas beau et qu'il pleut, j'irai au cinéma. Sinon, j'irai courir au bord du Rhône.*

Remarque 3 :

Ici, le terme *Sinon* signifie "S'il ne fait pas beau et qu'il ne pleut pas". C'est pareil en python, l'instruction conditionnée par `else` est exécutée si toutes les conditions précédentes sont fausses.

Notre algorithme devient alors :

Étapes	Instructions
1	Se lever
2	Regarder la météo
3.1	S'il fait beau, aller en montagne
3.2	S'il ne fait pas beau et qu'il pleut, aller au cinéma
3.3	S'il ne fait pas beau et qu'il pleut pas, aller courir au bord du Rhône

En python, on utilise l'instruction `elif` qui est la contraction de *else if* :

Listing 8.8 – Instruction if - else

```
if meteo == "beau":
    print("Aujourd'hui, je vais en montagne.")
elif meteo == "pluie":
    print("Aujourd'hui, je vais au cinéma.")
else:
    print("Aujourd'hui, je vais courir au bord du Rhône.")
```

8.2.4 Exercices sur papier

16 Que vaut la variable solution à la fin du programme suivant ? Réponse :

Listing 8.9 – Exercice

```
a = True
b = ((2+2) > 4)
c = (2==3)

if c:
    solution=42
elif b:
    solution=2021
elif a:
    solution=1202
else:
    solution=73
```

17 Que vaut la variable solution à la fin du programme suivant ? Réponse :

Listing 8.10 – Exercice

```
a = 42
b = 2021
c = 1202
d = 73

if c < (d+a):
    solution = "A"
elif b+c == a:
    solution = "B"
elif a<=42:
    solution = "C"
else:
    solution = "D"
```

8.2.5 Exercices sur les ordinateurs

18 Écrivez un programme vérifiant si un nombre A est un multiple d'un autre nombre B. Utilisez pour cela l'opérateur modulo "%", qui, pour x%y, retourne le reste de la division euclidienne de x par y.
Testez avec A=2941 et B=17.

19 Écrivez un programme vérifiant si un nombre A est un multiple de 2 nombres B et C.
Testez avec A=26469, B=17 et C=9.

20 Écrivez un programme trouvant le minimum entre 3 nombres.

21 Écrivez un programme déterminant si une année est bissextile.

8.3 Bibliographie

— *L'Informatique autrement*, De la pensée computationelle aux applications numériques, Dimitri Racordon, Damien Morard, Emmanouela Stachtiari,Aurélien Coet, Alexandre-Quentin Berger, Didier Buchs
— https ://openclassrooms.com/fr/courses/7168871-apprenez-les-bases-du-langage-python
— https ://docs.python.org/fr/3/tutorial/index.html