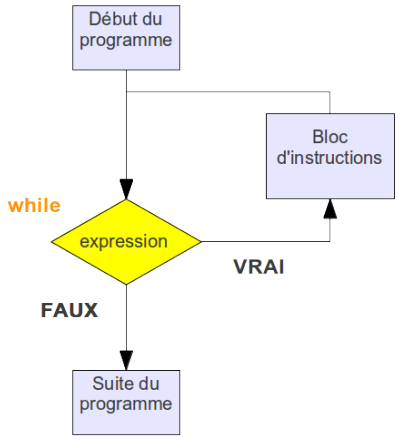
Chapitre 3 - Les boucles

Une boucle permet d'exécuter une portion de code plusieurs fois de suite.

L'instruction `while`



Syntaxe

```
while expression:     # ne pas oublier le signe de ponctuation ':'
    bloc d'instructions     # attention à l'indentation
# suite du programme
```

Si l'expression est vraie (`True`) le bloc d'instructions est exécuté, puis l'expression est à nouveau évaluée. Le cycle continue jusqu'à ce que l'expression soit fausse (`False`) : on passe alors à la suite du programme.

Exemple : un script qui compte de 1 à 4

```
# script Boucle1.py

# initialisation de la variable de comptage
compteur = 1
while compteur < 5:
    # ce bloc est exécuté tant que la condition (compteur < 5) est vraie
    print(compteur, compteur < 5)
    compteur += 1    # incrémentation du compteur, compteur = compteur + 1
print(compteur < 5)
print("Fin de la boucle")
```

```
>>>
1 True
```

```
2 True
3 True
4 True
False
Fin de la boucle
```

Table de multiplication par 8

```
# script Boucle2.py

compteur = 1    # initialisation de la variable de comptage
while compteur <= 10:
    # ce bloc est exécuté tant que la condition (compteur <= 10) est vraie
    print(compteur, '* 8 =', compteur*8)
    compteur += 1    # incrémentation du compteur, compteur = compteur + 1
print("Et voilà !")
```

```
>>>
1 * 8 = 8
2 * 8 = 16
3 * 8 = 24
4 * 8 = 32
5 * 8 = 40
6 * 8 = 48
7 * 8 = 56
8 * 8 = 64
9 * 8 = 72
10 * 8 = 80
Et voilà !
```

Affichage de l'heure courante

```
# script Boucle3.py

import time     # importation du module time
quitter = 'n'   # initialisation
while quitter != 'o':
    # ce bloc est exécuté tant que la condition est vraie
    # strftime() est une fonction du module time
    print('Heure courante ', time.strftime('%H:%M:%S'))
    quitter = input("Voulez-vous quitter le programme (o/n) ? ")
print("A bientôt")
```

```
>>>
Heure courante  09:48:54
Voulez-vous quitter le programme (o/n) ? n
Heure courante  09:48:58
```

```
Voulez-vous quitter le programme (o/n) ? o
A bientôt
```

L'instruction `for`

Syntaxe

```
for élément in séquence :
    bloc d'instructions
# suite du programme
```

Les éléments de la séquence sont issus d'une chaîne de caractères ou bien d'une liste.

Exemple avec une séquence de caractères

```
# script Boucle4.py

chaine = 'Bonsoir'
for lettre in chaine:       # lettre est la variable d'itération
    print(lettre)
print("Fin de la boucle")
```

La variable `lettre` est initialisée avec le premier élément de la séquence (`'B'`). Le bloc d'instructions est alors exécuté. Puis la variable `lettre` est mise à jour avec le second élément de la séquence (`'o'`) et le bloc d'instructions à nouveau exécuté... Le bloc d'instructions est exécuté une dernière fois lorsqu'on arrive au dernier élément de la séquence (`'r'`) :

```
>>>
B
o
n
s
o
i
r
Fin de la boucle
```

Exemple avec les éléments d'une liste

```
# script Boucle5.py

maliste = ['Pierre', 67.5, 18]
for element in maliste:
    print(element)
print("Fin de la boucle")
```

Là, on affiche dans l'ordre les éléments de la liste :

```
>>>
Pierre
67.5
18
Fin de la boucle
```

Fonction `range()` L'association avec la fonction `range()` est très utile pour créer des séquences automatiques de nombres entiers :

```
# script Boucle6.py
print(range(1,5))

for i in range(1,5):
    print(i)
print("Fin de la boucle")
```

```
>>>
[1, 2, 3, 4]
1
2
3
4
Fin de la boucle
```

Table de multiplication La création d'une table de multiplication paraît plus simple avec une boucle `for` qu'avec une boucle `while` :

```
# script Boucle7.py

for compteur in range(1,11):
    print(compteur, '* 9 =', compteur*9)
print("Et voilà !")
```

```
>>>
1 * 9 = 9
2 * 9 = 18
3 * 9 = 27
4 * 9 = 36
5 * 9 = 45
6 * 9 = 54
7 * 9 = 63
8 * 9 = 72
9 * 9 = 81
10 * 9 = 90
Et voilà !
```

L'instruction `break`

L'instruction `break` provoque une sortie immédiate d'une boucle `while` ou d'une boucle `for`. Dans l'exemple suivant, l'expression `True` est toujours ... vraie : on a une boucle sans fin. L'instruction `break` est donc le seul moyen de sortir de la boucle.

Affichage de l'heure courante

```
# script Boucle8.py

import time     # importation du module time
while True:
    # strftime() est une fonction du module time
    print('Heure courante ', time.strftime('%H:%M:%S'))
    quitter = input('Voulez-vous quitter le programme (o/n) ? ')
    if quitter == 'o':
        break
print("A bientôt")
```

```
>>>
Heure courante  14:25:12
Voulez-vous quitter le programme (o/n) ? n
Heure courante  14:25:20
Voulez-vous quitter le programme (o/n) ? o
A bientôt
```

Astuce

Si vous connaissez le nombre de boucles à effectuer, utiliser une boucle `for`. Autrement, utiliser une boucle `while` (notamment pour faire des boucles sans fin).

Exercices

Exercice 3.1 * Ecrire un script qui affiche toutes les tables de multiplication (de 1 à 10).

Exercice 3.2 * Ecrire un script qui calcule la moyenne d'une série de notes. On pourra utiliser une variable qui contient la somme intermédiaire des notes.

```
>>>
Nombre de notes ? 3
-->  15
-->  11.5
-->  14
```

```
Moyenne : 13.5
>>>
```

Exercice 3.3 *

1. Avec une boucle `for`, écrire un script qui compte le nombre de lettres z dans une chaîne de caractères.
   Par exemple :

   ```
   >>>
         Entrer la chaîne : Zinedine Zidane
         Résultat : 2
   ```

2. Faire la même chose, directement avec la méthode `count()` de la classe `str`.
   Pour obtenir de l'aide sur cette méthode :

   ```
   >>> help(str.count)
   ```

Exercice 3.4 * Avec une boucle `while`, écrire un script qui affiche l'heure courante avec une actualisation chaque seconde.
On utilisera la fonction `sleep()` du module `time`.
Pour obtenir de l'aide sur cette fonction :

```
>>> import time
>>> help(time.sleep)
```

Par exemple :

```
>>> # Taper CTRL + C pour arrêter le programme.
>>>
Heure courante  12:40:59
Heure courante  12:41:00
Heure courante  12:41:01
Heure courante  12:41:02
KeyboardInterrupt
>>>
```

Remarque : il est vilain de forcer l'arrêt d'un programme avec CTRL + C
Nous verrons comment interrompre proprement un programme dans le chapitre Gestion des exceptions.

Exercice 3.5 **

1. Ecrire le script du jeu de devinette suivant :

```
>>>
  Le jeu consiste à deviner un nombre entre 1 et 100 :

  --->   50
  trop petit !
  --->   75
  trop petit !
  --->   87
  trop grand !
  --->   81
  trop petit !
  --->   84
  trop petit !
  --->   85
  Gagné en 6 coups !
```

2. Quelle est la stratégie la plus efficace ?
3. Montrer que l'on peut deviner un nombre en 7 coups maximum.

Bibliographie : La dichotomie Remarque : pour créer un nombre entier aléatoire entre 1 et 100 :

```
import random
nombre = random.randint(1,100)
```

Exercice 3.6 ** Code de César En cryptographie, le code de César est une technique de chiffrement élémentaire qui consiste à décaler une lettre de 3 rangs vers la droite :
A $\rightarrow$ D
B $\rightarrow$ E
...
Z $\rightarrow$ C

1. 1) Ecrire le script de ce codage. Par exemple :

   ```
   >>>
       Message à coder ? abcdefghijklmnopqrstuvwxyz
       defghijklmnopqrstuvwxyzabc

   >>>
       Message à coder ? Monty Python's Flying Circus
       prqwb sbwkrq'v ioblqj flufxv
   ```

   On pourra utiliser la chaîne `'abcdefghijklmnopqrstuvwxyz'` et la méthode `find()` de la classe `str`.
   Pour obtenir de l'aide sur cette méthode :

   ```
   >>> help(str.find)
   ```

2. Ecrire le script du décodage. Par exemple :

```
>>>
      Message à décoder ? prqwb sbwkrq'v ioblqj flufxv
      monty python's flying circus
```

Exercice 3.7 Geralt kiffe ses cheveux.

Pour ne pas être dérangé dans ses aventures il décide de les couper chaque fois qu'ils font plus de 40 cm. Ce matin Geralt a coupé ses cheveux et ils mesurent 17 cm. Chaque jour la longueur de ses cheveux augmente d'un pour cent (leur longueur est multipliée par 1.01).
Ecrire un script qui affiche dans combien de jours Geralt devra couper ses cheveux.

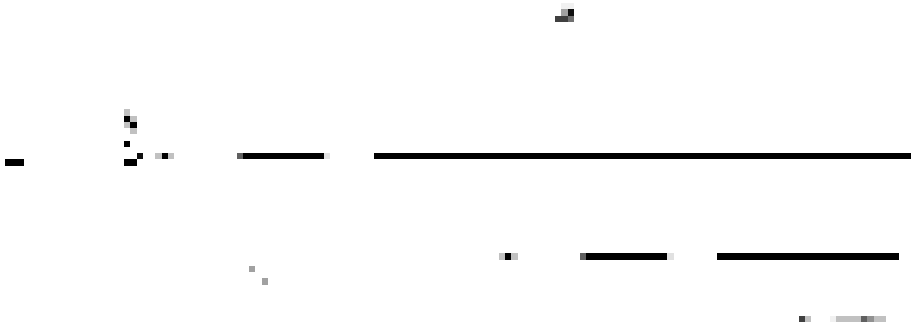
Les exercices suivants sont nécessitent quelques aptitudes en mathématiques...

Exercice 3.8 ** Ecrire un script qui donne l'évolution de la suite convergente : $u_{n+1} = u_n/2 + 1/u_n$ Par exemple :

```
>>>
Valeur initiale ? 20
Jusqu'à quel rang ? 10
0 20.0
1 10.05
2 5.12450248756
3 2.75739213842
4 1.74135758045
5 1.44494338196
6 1.41454033013
7 1.41421360012
8 1.41421356237
```

```
9 1.41421356237
10 1.41421356237
>>>
```

Exercice 3.9 ** Fraction continue infinie
Estimer la valeur numérique de la fraction continue suivante :



Comparer avec la valeur exacte : $(1 + \sqrt{5})/2$

Exercice 3.10

1. 1) * Ecrire un script qui détermine si un nombre entier est premier ou pas.

   Rappel : un nombre

   premier est un entier naturel qui n'a que deux diviseurs : 1 et lui même. Ex : 3 est premier mais 6 ne l'est pas car 2*3=6. Par exemple :

   ```
   >>>
       Nombre ? 17
       17 est un nombre premier
   ```

2. 2) ** Ecrire un script qui décompose un nombre entier en un produit de facteurs premiers.

   ```
   >>>
       Nombre à décomposer ? 2142
       2142 = 2 * 3 * 3 * 7 * 17
   ```

QCM

QCM sur les boucles