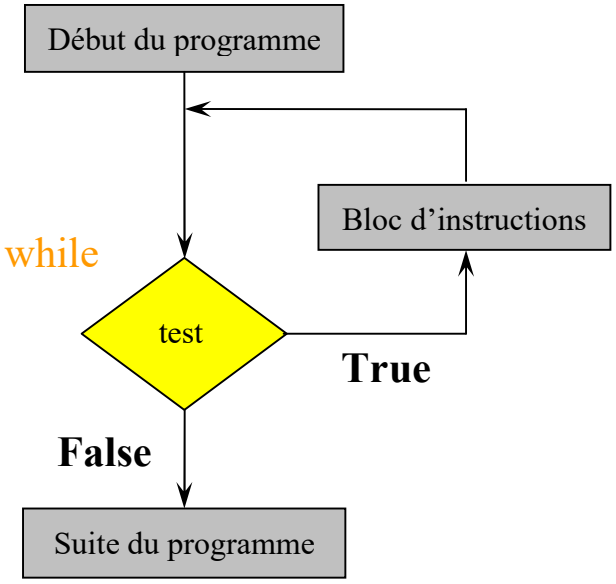
CHAPITRE 3 : LES BOUCLES

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

L'instruction while



Syntaxe :

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

Si le test est vrai (True) le bloc d'instructions est exécuté, puis le test est à nouveau effectué. Le cycle continue jusqu'à ce que le test soit faux (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("Fin de la boucle")
>>>
1 True
2 True
3 True
4 True
Fin de la boucle
```

Table de multiplication par 8 :

```
# script Boucle2.py

compteur = 0        # initialisation de la variable de comptage
while compteur <= 9:
   # ce bloc est exécuté tant que la condition (compteur<=9) est vraie
```

```
    print(compteur, '* 8 =', compteur * 8)
    compteur += 1    # incrémentation du compteur, compteur = compteur + 1
>>>
0 * 8 = 0
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
```

Affichage de l'heure courante :

```
# script Boucle3.py

From datetime import datetime     # importation du module time
quitter = 'n'   # initialisation
while quitter != 'o':

    # ce bloc est exécuté tant que la condition est vraie
    print('Date et heure courante :', datetime.now())
    quitter = input('Voulez-vous quitter le programme (o/n) ? ')
>>>
Date et heure courante :  2020-11-02 14:48:53.580699
Voulez-vous quitter le programme (o/n) ? n
Date et heure courante :   2020-11-02 14:48:57.728268
Voulez-vous quitter le programme (o/n) ? o
```

L'instruction for

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

Syntaxe :

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

```
# script Boucle4.py

li = ['Pierre', 67.5, 18]
for el in li: #el désigne un pointeur
    print(el)
print('Fin de la boucle')
```

On affiche dans l'ordre les éléments de la liste :

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

```
# script Boucle5.py

ch = 'Bonsoir'
for el in ch:                    # el est un pointeur sur la chaine ch
   print(el, end = '')
print('\nFin de la boucle')
```

La variable el pointe sur le premier élément de la séquence ('B').
Le bloc d'instructions est alors exécuté.
Puis la variable el pointe sur le deuxième 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') :

```
Bonsoir
Fin de la boucle
```

Un autre exemple avec une liste :

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): #i désigne un indice
   print(i)
print('Fin de la boucle')
>>>
range(1,5)
1
2
3
4
Fin de la boucle
```

Pour afficher une liste de nombres :

```
print(list(range(1, 5)))
>>>
 [1, 2, 3, 4]
```

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(10):
   print(compteur, "* 8 =", compteur*8)
print('Et voilà !')
>>>
0 * 8 = 0
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
```

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 :

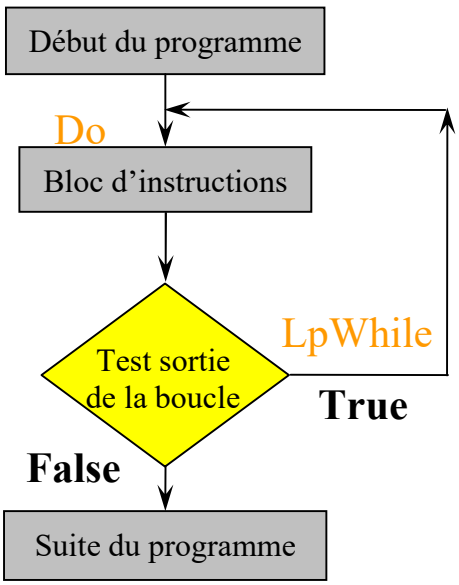
```
# script Boucle8.py

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
```

L'instruction Do LoopWhile (Faire … Tant Que).

Ce test, bien qu'utile, n'existe pas en Python.
Il est donc nécessaire de "ruser" en utilisant :

- une boucle infinie ;
- un booléen ;
- un test If pour sortir de la boucle infinie.



Syntaxe :

```
boucle = True
while boucle:                          # création de la boucle infinie
   bloc d'instructions
   if test:                            #test permettant de sortir de la boucle infinie
      boucle = False
# suite du programme
```

Le test de sortie de la boucle **Faire … Tant Que** n'est exécuté qu'après après avoir exécuté le bloc d'instruction.
Si le test **if** est faux, alors le booléen boucle contient toujours le booléen True et on reste donc encore dans la boucle infinie.
Si le test **if** est vrai, le **booléen** boucle passe à **False** et on quitte ainsi la boucle infinie à la prochaine boucle.

Astuces :

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

- **L'instruction break**
L'instruction break provoque une sortie immédiate d'une boucle while ou d'une boucle for.

LES BOUCLES TESTS

I. Concevoir un programme avec les contraintes suivantes :
- En entrée : un ou plusieurs caractères.
- En sortie : la parité du nombre, après avoir testé que l'entrée correspond à un nombre entier positif ou nul. On doit ressaisir un nouveau nombre si l'entrée n'est pas un nombre entier.

II. Écrire un script qui affiche une table de multiplication. Le programme demandera le numéro de la table à afficher.

III. Écrire un script qui affiche toutes les tables de multiplication (de 1 à 10).

IV. Écrire 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
>>>
```

V. 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)
```

VI. 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
KeyboardInterrupt
>>>
```

VII. 1°) Écrire 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.
Remarque : pour créer un nombre entier aléatoire entre 1 et 100 :

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

VIII. Écrire un script qui donne la valeur des termes et la somme des termes d'une suite arithmétique puis géométrique, à partir de la saisie du nombre de termes à afficher, de la valeur du premier et de la raison.

```
Suite géométrique
Entrer le premier terme : 2
Entrer la raison : 0.5
Nombre de termes à afficher : 5
u_0 = 2.0       Somme = 2.0
u_1 = 1.0       Somme = 3.0
u_2 = 0.5       Somme = 3.5
u_3 = 0.25      Somme = 3.75
u_4 = 0.125     Somme = 3.875
```

```
>>>
```

IX. Écrire 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
Rang =  1          Valeur =  20.0
Rang =  2          Valeur =  10.05
…
Rang =  10         Valeur =  1.414213562373095
>>>
```

X. 1°) Écrire un script qui détermine si un nombre entier positif est premier ou pas.
Par exemple :

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

Faire le test avec les nombres suivants et remplir le tableau avec le temps d'exécution.

Nombre	23 456 789	99 999 989	1 000 000 007
Temps d'exécution			

2°) Écrire un script qui décompose un nombre entier positif en un produit de facteurs premiers.

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

XI. Fraction continue infinie.
Estimer la valeur numérique de la fraction continue suivante :

$$1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\ldots}}}}$$

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