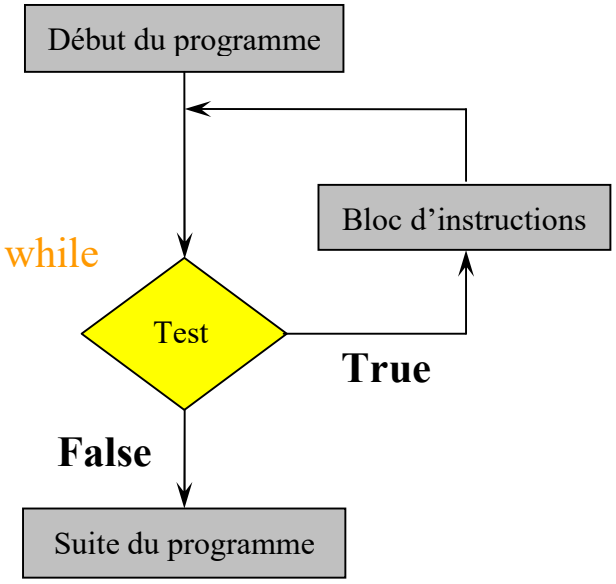
Stéphane KELLER – Lycée agricole Louis Pasteur
https://github.com/KELLERStephane/KELLER-Stephane-Tests2maths

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
```

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.

```
# script Boucle4.py

ch = 'Bonsoir'
for el in ch:                    # el est un pointeur sur la chaine ch
    print(el)
print('Fin 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 :

```
# script Boucle5.py

li = ['Pierre',67.5,18]
for element in li:
    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
```

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')
>>>
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 :

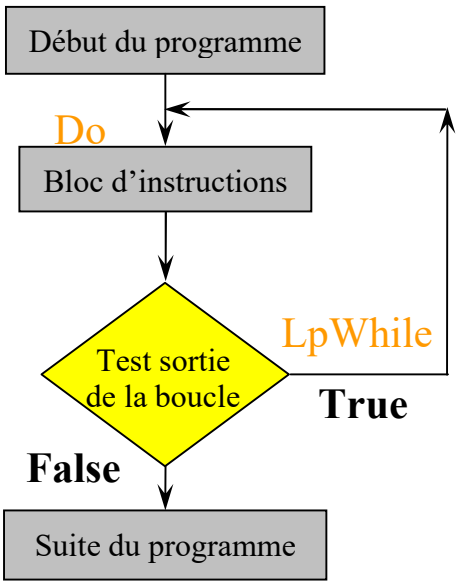
```
# script Boucle7.py

for compteur in range(10):
    print(compteur, "* 9 =", compteur*9)
>>>
0 * 9 = 0
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
```

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.