Thème 6 - Langages et programmation

03 Les Boucles WHILE en Python

1. Boucle WHILE

1.1. Premiers exemples

À la différence essentielle des boucles `for`, dont on peut savoir à l'avance combien de fois elles vont être exécutées, les boucles `while` sont des boucles dont on ne sort que lorsqu'une condition n'est plus satisfaite.

Avec donc le risque de rester infiniment bloqué à l'intérieur !

Algo

```
a = 0
while a < 3:
    print("ok")
    a = a + 1
print("fini")
```

Algo

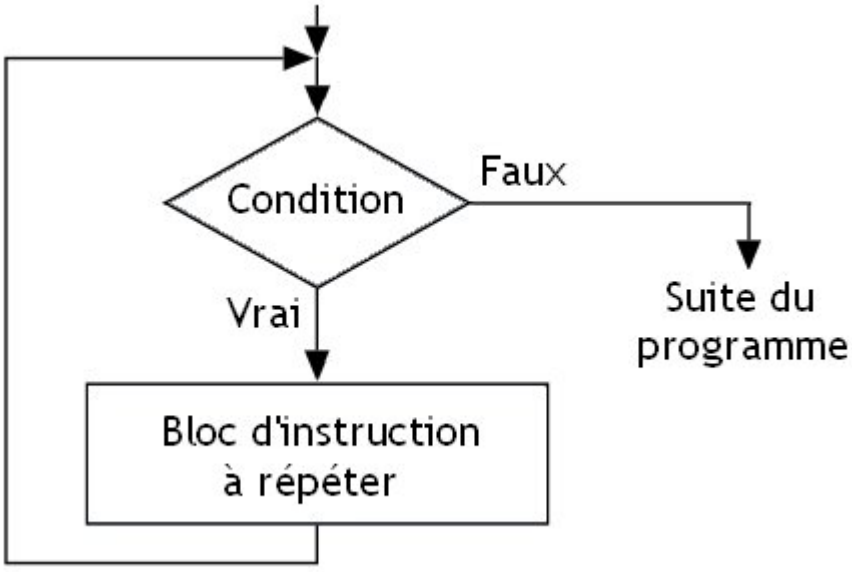
```
ok
ok
ok
fini
```

1.2. Syntaxe générale

❤ 📚 **A retenir :**

Écriture d'une boucle `while`

```python while condition: instruction1 instruction2 ... instructionN ````



La condition

La `condition` est un booléen, c'est-à-dire une expression que Python évaluera à `True` ou à `False`.

Exemple de booléens résultant d'une évaluation :

Script Python

```
>>> 1 < 3
True
>>> 5 > 7
False
>>> a = 10
>>> a > 8
True
```

1.3. Les pièges ...

piège n°1 : ne JAMAIS SORTIR de la boucle

Exemple : Le programme suivant :

Script Python

```
a = 0
while a < 3:
    print("ok")
    a = a + 1
```
```

```
    a = a * 0
print("ce texte ne s'écrira jamais")
```

va écrire une suite infinie de `ok` et ne **jamais s'arrêter**

piège n°2 : ne JAMAIS ENTRER dans la boucle

Exemple : Le programme suivant :

Script Python

```
a = 0
while a > 10:
    print("ce texte non plus ne s'écrira jamais")
    a = a + 1

print("fini")
```

va écrire `fini` et s'arrêter.

Algo

```
a = 0
while a > 10:
    print("ce texte non plus ne s'écrira jamais")
    a = a + 1

print("fini")
```

fini

!!! exo "💻 Exercice 8 : Trouver le plus petit nombre entier n tel que 2^n soit supérieur à 1 milliard.

Algo

```
```python
n = 1
while 2**n...:
    ...
    print("trop petit")
print("trouvé : ",n)
```
```

La boucle infinie a été présentée comme un danger qu'il faut éviter.

Pourtant, dans quelques situations, il est d'usage d'enfermer *volontairement* l'utilisateur dans une boucle infinie.

C'est notamment le cas des codes p5 où la fonction `draw()` est une boucle infinie dont on ne sort que lorsqu'un évènement est intercepté (par exemple, le clic sur la fermeture de la fenêtre d'affichage).
Ou lors de la création d'un jeu....

Observez et exécutez le code suivant :

Script Python

```
while True :
    reponse = input("tapez sur la lettre S du clavier pour me sortir de cet enfer : ")
    if reponse == 'S' or reponse == 's':
        break

print("merci, j'étais bloqué dans une boucle infinie")
```

Algo

```
tapez sur la lettre S du clavier pour me sortir de cet enfer : E
tapez sur la lettre S du clavier pour me sortir de cet enfer : S
merci, j'étais bloqué dans une boucle infinie
```

- le début du code : `while True` est typique des boucles infinies volontaires. On aurait tout aussi bien pu écrire `while 3 > 2` (on rencontre même parfois des `while 1`)
- vous avez découvert l'expression `break` qui comme son nom l'indique permet de casser la boucle (cela marche pour `while` comme pour `for`) et donc d'en sortir. Son emploi est controversé parmi les puristes de la programmation. Nous dirons juste que c'est une instruction bien pratique.