

1INDF chapitre 7 : boucle « tant que »

Nom:	Groupe:	Date:
------	---------	-------

Une boucle est une structure de contrôle destinée à exécuter une séquence d'instructions (« bloc ») plusieurs fois de suite.

- Nous avons déjà vu la boucle « répéter X fois » (« for » en anglais), qui répète l'exécution d'un bloc d'instructions un certain nombre de fois. Ce nombre de fois doit être connu à l'avance.
- La boucle « tant que » (« while » en anglais) répète l'exécution d'un bloc d'instructions tant que la condition d'un test est vraie ; il n'est pas nécessaire de savoir d'avance le nombre de fois que ces instructions se répéteront.

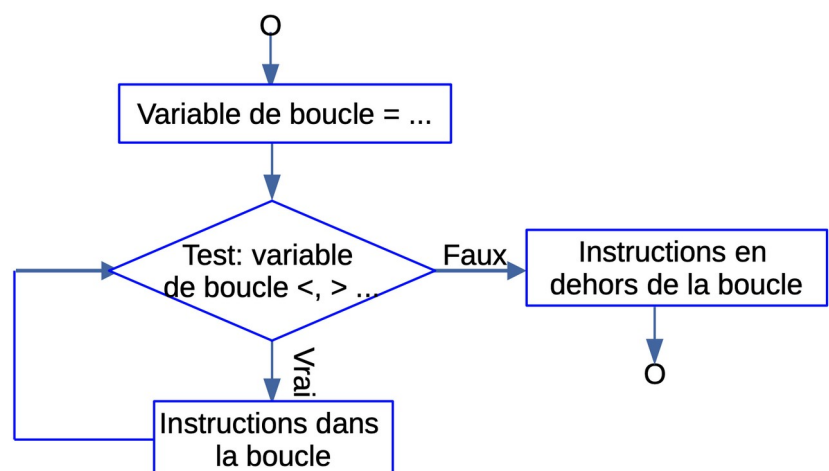
Dans l'exemple Blockly ci-dessous, la commande « repeat until » est utilisée pour répéter l'instruction « avancer d'un pas » jusqu'à ce que la destination soit atteinte. Le test consiste à vérifier si le bonhomme est arrivé à destination ; le bloc n'est composé que d'une seule instruction « avancer d'un pas ».



Fonctionnement d'une boucle « tant que »

Dans une boucle « tant que »,

1. avant d'écrire le programme, il faut identifier quelles instructions doivent se répéter, et à quelle condition elles se répètent (ou arrêtent de se répéter) ;
2. dans le programme, il faut tout d'abord déclarer la variable de boucle en lui donnant une valeur de départ ;
3. ensuite, il faut tester si la valeur de la variable de boucle conduit à la répétition des instructions de la boucle ou à leur interruption ;
4. si oui, les instructions de la boucle s'exécutent ; sinon, la boucle est terminée ;
5. lorsque la boucle est terminée, les instructions qui se trouvent en dehors de la boucle sont exécutées.



- Une **itération** correspond à un passage dans la boucle, i.e. à chaque fois que l'ensemble des instructions d'une boucle sont exécutés.
- La **portée** de la boucle correspond à l'ensemble des instructions qui font partie de la boucle ; les instructions de ce **bloc** sont **indentées** de 4 espaces.

Boucles bornées

Lorsque le nombre de répétitions est connu d'avance, il est possible d'utiliser une boucle « tant que » ou une boucle « répéter X fois ». La boucle est dite bornée car le nombre d'itérations est explicitement défini dans le programme, soit par le test de la boucle « tant que », soit par la commande « range » de la boucle « for ».

Par exemple, les programmes 1 et 2 ci-dessous affichent les nombres de 0 à 5, puis « fin » :

Programme 1	Programme 2	Affichage
1. <code>compteur = 0</code>	1. <code>for i in range(6):</code>	0
2. <code>while compteur < 6:</code>	2. <code>print(i)</code>	1
3. <code>print(compteur)</code>	3. <code>print("Fin !")</code>	2
4. <code>compteur = compteur + 1</code>		3
		4
		5
5. <code>print("Fin !")</code>		Fin !

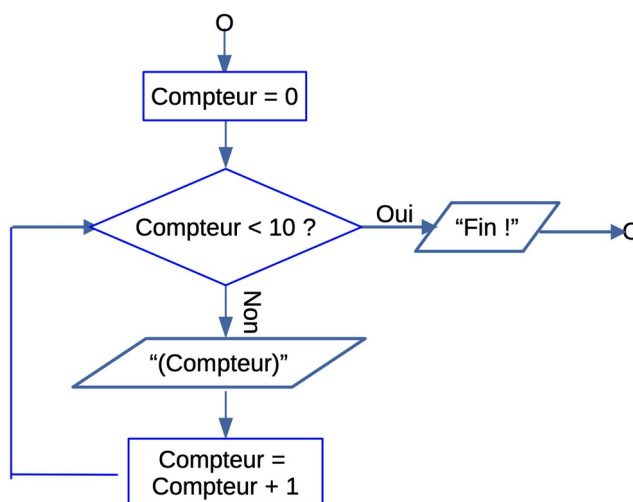
Dans le programme 1,

- la variable de la boucle est « compteur »,
- le test de la boucle « tant que » est « le compteur est-il inférieur à 6 ? »,
- le bloc d'instructions de la boucle est composé des lignes 3 et 4,
- l'instruction de la ligne 5 ne fait pas partie de la boucle.

Dans le programme 2, il n'y a pas de variable de boucle ni de test. La variable « i » est un itérateur qui prend les valeurs 0 à 4 grâce à la commande « range ». Le bloc d'instruction de la boucle ne comporte que la ligne 2, et la ligne 3 ne fait pas partie de la boucle.

Les deux programmes sont représentés par le même logigramme (montré à droite), et le bloc d'instructions de la boucle se répète 6 fois, il y a donc 6 itérations de la boucle.

Notons que dans le cas d'une boucle « tant que », il est nécessaire d'incrémenter le compteur explicitement, alors que dans une boucle « répéter X fois », la commande « range » le fait automatiquement.



Boucles non-bornées

Dans d'autres cas, on ne connaît pas d'avance combien de fois les instructions doivent être répétées ; il n'est donc pas possible d'utiliser une boucle « répéter X fois ». Par exemple :

- reposer la même question jusqu'à ce que la réponse correcte soit obtenue ;
- rechercher à partir de quelle valeur une fonction atteint un certain seuil.

La boucle est alors dite « non-bornée » car le début et la fin sont décidés lors de l'exécution du programme, grâce au test de la boucle « tant que ».

« while_motdepasse.py »

Ce programme demande à l'utilisateur d'entrer une chaîne de caractères jusqu'à ce qu'elle corresponde à celle de la variable « mot_de_passe ». Dans ce cas, le nombre de fois que seront exécutées les instructions de la boucle « tant que » n'est pas connu, car on ne sait pas si l'utilisateur écrira son mot de passe correctement du premier coup ou s'il fera plusieurs essais, il n'est donc pas possible d'utiliser une boucle « for » pour écrire ce programme.

```
1. mot_de_passe = ""
2. while mot_de_passe != "Bachibouzouk":
3.     print("Entrez votre mot de passe:")
4.     mot_de_passe = input()
5. print("Mot de passe correct, login accepté")
```

« while_comptebancaire.py »

Ce programme exécute des retraits tant que le solde du compte bancaire le permet. Dans ce cas aussi, il n'est pas nécessaire de connaître le nombre de retraits qui auront lieu avant d'exécuter les retraits.

```
1. solde = 600
2. retrait = 250

3. while solde >= 250:
4.     print("Dans la boucle while, solde avant le retrait:",solde)
5.     solde = solde - retrait
6.     print("Dans la boucle while, solde après le retrait:",solde)
7. print("A la fin, le solde est de",solde)
```

Boucles infinies

Une erreur classique consiste à oublier d'incrémenter la variable de boucle ; la boucle se répète alors à l'infini, car la condition reste toujours vraie. Dans le cas des microprocesseurs comme le micro:bit, il est parfois souhaitable que les instructions se répètent à l'infini. La condition de boucle est alors simplement « True », qui par définition reste toujours vraie.

Programme avec erreur	Programme de microprocesseur
<pre>i = 0 while i < 4: print("Bonjour !")</pre>	<pre>from microbit import * while True : display.show(Image.HEART)</pre>

Sans boucle, le logigramme ressemble à ceci :

