1. Répéter quand on connaît une condition d'arrêt : Tant que …

Cette structure de boucle non bornée, s'utilise quand on veut répéter des instructions :
– jusqu'à ce qu'une condition soit réalisée ;
– ou, ce qui revient au même, tant qu'une condition n'est pas réalisée.

Exemples			
	Répéter Jusqu'à ce que $S > 50$	revient à	Répéter Tant que $S \leq 50$
	Répéter Jusqu'à ce que $S \geq 50$	revient à	Répéter Tant que $S < 50$
	Répéter Jusqu'à ce que $S \leq 50$	revient à	Répéter Tant que $S > 50$
	Répéter Jusqu'à ce que $S < 50$	revient à	Répéter Tant que $S \geq 50$

.

Boucle non bornée : Tant que....
Tant que *condition* Faire
    suite d'instructions
Fin Tant que

Exemple
Tant que le dé ne donne pas 6 Faire
    Lancer le dé.
    Augmenter le nombre de lancers de 1
Fin Tant que

Cette boucle repose sur une condition, qui devra être réalisée à un moment donné pour que l'exécution s'arrête.
On ne connaît pas a priori le nombre de répétitions.
Quand on a besoin de savoir combien de répétitions ont été faites, on crée une variable qui compte le nombre de passages dans la boucle en étant augmentée (incrémentée) de 1 à chaque passage. On dit que c'est un **compteur**.

Exemple

Algorithme	Commentaires
VARIABLES : S et k des nombres	
INITIALISATION : S prend la valeur 4 k prend la valeur 0	La variable S prend la valeur 4 et la variable k la valeur 0. On dit qu'on **initialise** S et k (on leur affecte leurs valeurs initiales.)
TRAITEMENT : Tantque $S \leq 30$ Faire S prend la valeur $S \times 2$ k prend la valeur $k + 1$ FinTantque	La valeur contenue dans S est inférieure à 50, on entre dans la boucle. Les deux instructions dans la boucle sont exécutées : S prend la valeur $4 \times 2 = 8$ k prend la valeur $0+1 = 1$ La valeur de S est inférieure 30 donc on repasse dans la boucle, les instructions sont à nouveau effectuées, etc... Quand la valeur contenue dans S devient strictement supérieure à 30, on sort de la boucle.
SORTIE : Afficher S et k	L'instruction suivante est exécutée : les valeurs de S et k sont affichées.

Tableau d'état des variables lors de l'exécution du programme

S	4	8	16	32
k	0	1	2	3
$S \leq 50$	Vrai	Vrai	Vrai	Faux

→ Le test $S \leq 30$ renvoie la valeur Faux, donc on sort de la boucle. L'algorithme affiche donc 32 et 3. Cela signifie qu'il a fallu multiplier la valeur initiale de S par 2 trois fois de suite pour que S dépasse 32.

2. Tester une boucle dans un algorithme

L'exemple détaillé ci-dessus a été choisi pour avoir un petit nombre de répétitions, ce qui permet de dresser le tableau d'état des variables de façon complète.

En pratique, on ignore le nombre de répétitions nécessaires pour sortir de la boucle !
Il faut déjà s'assurer que la condition d'arrêt sera bien remplie et que la boucle se termine.

Pour tester le programme, il ne faut pas hésiter à modifier la condition d'arrêt pour diminuer le nombre de répétitions et examiner si le programme renvoie bien l'information attendue dans ce cas. Si le test est concluant, on remet les valeurs qui correspondent à la situation étudiée.

3. Programmation en Python

Exemple

Scratch	Algorithme	Python
	VARIABLES : *S*, *k* sont des nombres INITIALISATION : *S* prend la valeur 4 *k* prend la valeur 0 TRAITEMENT : Tantque $S \leq 30$ Faire *k* prend la valeur k+1 S prend la valeur $S \times 2$ FinTant Que SORTIE : Afficher *k*	1 S = 4 2 k = 0 3 while S <= 30: 4 k = k + 1 5 S = S * 2 6 print(k)

Commentaires sur le script en Python
Les lignes 3 à 5 correspondent à la boucle « Tant que … Fin Tant que ».

- Le mot clef **while** introduit la boucle et les deux points à la fin de la ligne ouvrent le bloc indenté (décalé vers la droite) comportant les instructions à répéter.

- Le retour à l'indentation précédente (ligne 6) marque la sortie de la boucle. La valeur de *k* n'est donc affichée qu'une seule fois, à la fin du programme.

Remarque
La variable *k* est augmentée (incrémentée) de 1 à chaque passage dans la boucle. Elle compte donc le nombre de passages dans la boucle. On l'appelle un **compteur**.

Faire afficher *k* en ligne 6 revient à faire afficher le nombre de passages dans la boucle effectués donc le nombre de répétitions effectuées.

Dans l'exemple donné, on fait ainsi afficher le nombre de fois où l'on a dû multiplier *S* par 2 pour obtenir une valeur strictement supérieure à 30.