

# Boucles

La zone « Commande » contient des blocs qui contrôlent si d'autres blocs placés à l'intérieur de celle-ci sont exécutés. Il y a deux types de blocs de contrôle : les blocs **si-sinon-Blöcke** (décrits sur une page) et les blocs qui contrôlent le nombre de fois où leur intérieur est exécuté. Ces derniers sont appelés boucles, car leur intérieur, également appelé corps ou corps de boucle, est répété (éventuellement) plusieurs fois. Chaque passage d'une boucle est appelé itération.

## Blocs pour créer des boucles

### Répétition permanente

Le bloc **Répétition permanente** exécute le code dans son corps jusqu'à la fin du programme.

### Répéter

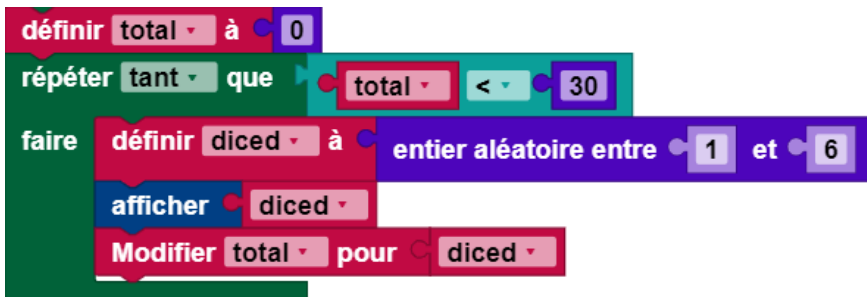
Le bloc **Répétition** exécute le code dans son corps autant de fois que prévu, par exemple dix fois « Bonjour ! ». Le bloc suivant émet par exemple dix fois « Salut ! » :



### Répétition tant que

Imaginez un jeu où un joueur lance un dé et additionne toutes les valeurs lancées tant que la somme est inférieure à 30. Les blocs suivants exécutent cette partie :

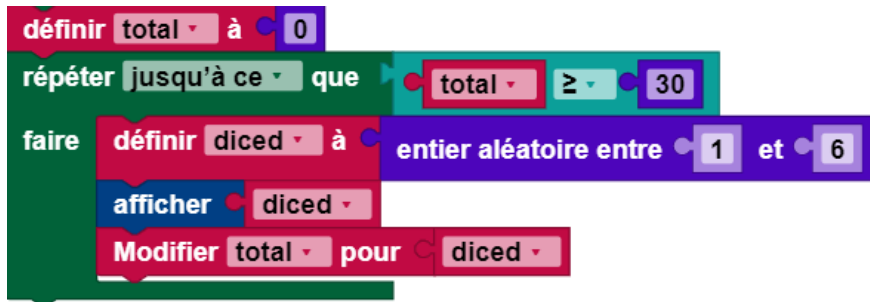
1. Une variable nommée **total** obtient une valeur initiale de 0.
2. La boucle commence par vérifier si le **total** est inférieur à 30. Si c'est le cas, les blocs passent dans le corps.
3. Un nombre aléatoire compris entre 1 et 6 est généré (pour simuler un lancer de dé) et stocké dans une variable nommée **dés**.
4. Le nombre en dés est édité.
5. La variable **totale** est augmentée de la valeur **en dés**.
6. Une fois la fin de la boucle atteinte, le contrôleur revient à l'étape 2.



Après la fin de la boucle, tous les blocs suivants (non représentés) sont passés. Dans l'exemple, le passage en boucle se termine après qu'un certain nombre de nombres aléatoires se situent dans la plage de 1 à 6, et la variable **total** a alors comme valeur la somme de ces nombres qui est d'au moins 30.

## Répétition jusqu'à

Les boucles **Répétition tant que** répètent leur corps, **tant que** une condition est remplie. Les boucles **Répétition jusqu'à** sont similaires, à la différence qu'elles répètent leur corps **jusqu'à** ce qu'une condition définie soit remplie. Les blocs suivants sont équivalents à l'exemple précédent, car la boucle est exécutée jusqu'à ce que le **total** soit supérieur ou égal à 30.



## Compter-de-à

La boucle **Compter-de-à** augmente la valeur d'une variable en commençant par une première valeur, se terminant par une deuxième valeur et par incréments d'une troisième valeur, le corps étant exécuté une fois pour chaque valeur de la variable. Le programme suivant donne par exemple les chiffres 1, 3 et 5.



Comme le montrent les deux boucles suivantes, qui émettent respectivement les nombres 5, 3 et 1, cette première valeur peut être supérieure à la seconde. Le comportement est le même, que le montant incrémental (troisième valeur) soit positif ou négatif.



## Pour chacun

Le bloc **pour chacun** est similaire à celui de la boucle **Compter-de-à**, sauf qu'il utilise les valeurs d'une liste à la place de la variable de boucle dans un ordre numérique. Le programme suivant expose chaque élément de la liste « alpha », « bêta », « gamma » :



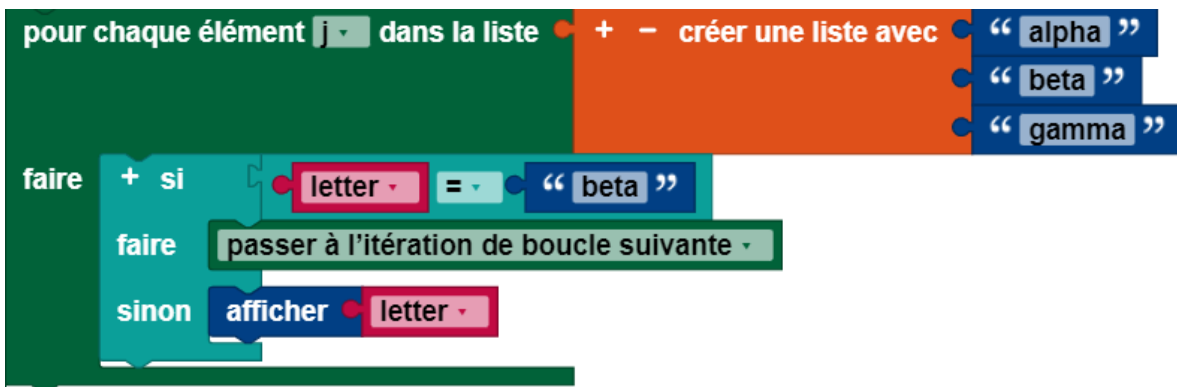
## Blocs de rupture de boucles

La plupart des boucles sont exécutées jusqu'à ce que la condition d'interruption soit satisfaite (pour les blocs **Répéter**) ou jusqu'à ce que toutes les valeurs de la variable de boucle soient acceptées (pour les boucles **Compter avec** et pour les boucles **pour chacun**). Deux blocs rarement nécessaires mais parfois utiles offrent des possibilités supplémentaires de contrôle du comportement de la boucle. Ils peuvent être utilisés pour n'importe quel type de boucle, même si les exemples suivants montrent leur utilisation pour les boucles **pour chacun**.

### Poursuivre avec la prochaine itération

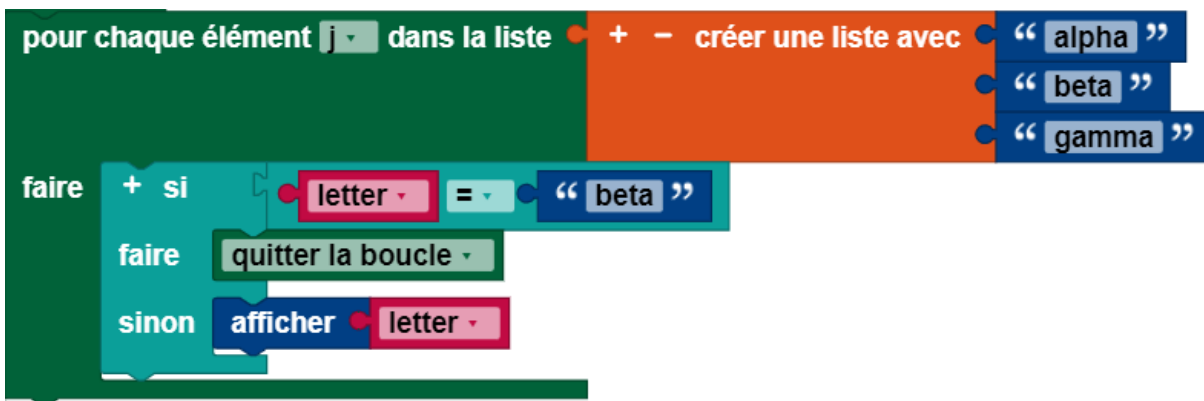
La boucle **Poursuivre avec la prochaine itération** permet de passer les blocs restants dans le corps de boucle et de commencer l'itération suivante de la boucle.

Le programme suivant donne « alpha » à la première itération de la boucle. Lors de la deuxième itération, le bloc **continue avec la prochaine intégration**, ce qui fait sauter la sortie de « beta ». Lors de la dernière itération, « gamma » est imprimé.



### Rupture de boucle

Le bloc **Rupture de boucle** permet une sortie prématurée d'une boucle. Le programme suivant donne « alpha » à la première itération et interrompt la boucle à la seconde itération si la variable de boucle est égale à « beta ». Le troisième point de la liste n'est jamais atteint.



---

Revision #6

Created 17 November 2021 20:44:18 by Admin

Updated 18 February 2022 14:13:39 by Admin