COURS

Programmer en blocs — Séquences, entrées, sorties et boucles

5e

Programme de mathématiques du cycle 4 — arrêté du 18 février 2026, BO n° 10 du 5 mars 2026

Un programme est une liste d'ordres très simples que la machine suit sans rien deviner : elle les exécute **un par un, dans l'ordre, de haut en bas**. Pour savoir ce qu'il va faire, **on l'exécute soi-même, bloc après bloc, sur son brouillon.**

1 Des instructions, dans un ordre

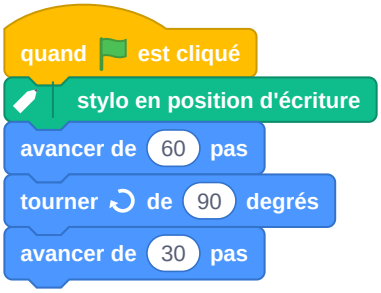
DÉFINITION

Une **instruction** est un ordre simple que la machine sait exécuter : avancer, tourner, afficher un message. Dans un langage de programmation par blocs, comme Scratch, chaque instruction est un **bloc**. Un **programme** est une suite de blocs emboîtés les uns sous les autres, sous un bloc de départ comme « quand le drapeau vert est cliqué ».

RÈGLE

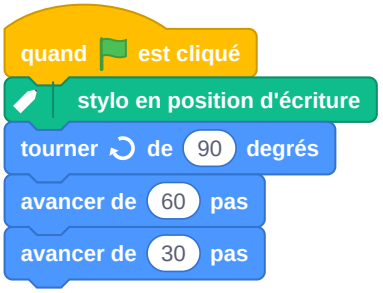
Séquencer, c'est choisir l'ordre des instructions. La machine les exécute de haut en bas, une seule à la fois : **les mêmes blocs, rangés dans un autre ordre, donnent un autre résultat.**

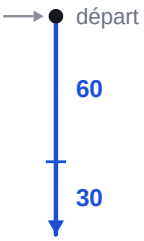
Programme A au départ, le lutin regarde vers la droite





Programme B mêmes blocs, autre ordre





A et B contiennent les mêmes blocs de déplacement : A trace un angle droit, B un seul trait de 90 pas.

« avancer » fait avancer le lutin **droit devant lui** ; « tourner ↻ » le fait pivoter sur place dans le sens des aiguilles d'une montre, « tourner ↺ » dans l'autre sens : **tourner ne déplace pas le lutin.**

EXÉCUTER UN PROGRAMME À LA MAIN

1. **Marquer le départ** sur le brouillon : un point pour la position, une flèche pour la direction.
2. **Exécuter un seul bloc**, noter ce qui a changé (position, direction, affichage), puis le cocher.
3. **Passer au bloc suivant**, jusqu'au dernier, sans jamais en regrouper deux.

EXEMPLE

Prévoir le tracé du programme A. Au départ, le lutin regarde vers la droite.

1. « avancer de 60 pas » : un trait de 60 pas vers la droite.
2. « tourner ↻ de 90 degrés » : un quart de tour, sans tracer ; il regarde **vers le bas**. « avancer de 30 pas » : un trait de 30 pas vers le bas.

deux traits perpendiculaires : 60 pas vers la droite, puis 30 pas vers le bas

PIÈGE

Le lutin regarde vers la droite ; après « tourner ↻ de 90 degrés », le faire monter. — ↻ est le sens des aiguilles d'une montre : une aiguille sur 3 h, après un quart de tour, est sur 6 h.

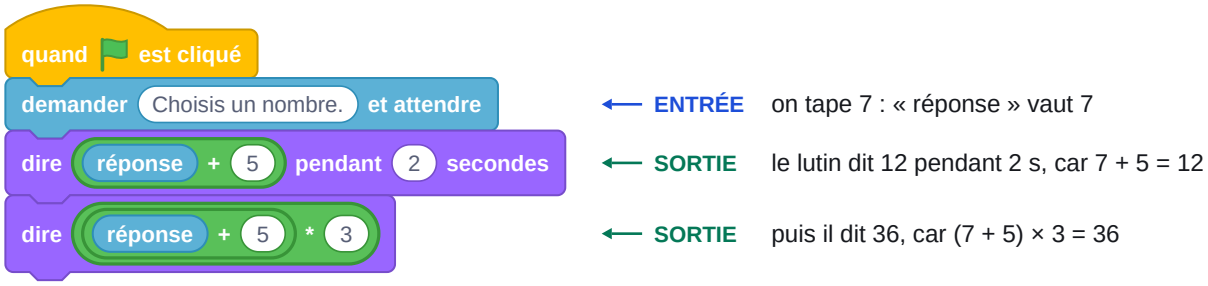
Ce que ça coûte : Le dessin est retourné, et chaque trait suivant est faux.

Le réflexe : mimer la rotation avec le doigt sur un cadran de montre avant de tracer.

2 Entrées et sorties

DÉFINITION

Une **entrée** est une donnée que le programme reçoit pendant son exécution : « demander … et attendre » pose une question, l'utilisateur tape une valeur, rangée dans le bloc arrondi « réponse », qui vaut **la dernière valeur saisie** et se glisse dans la case d'un autre bloc. Une **sortie** est ce que le programme produit : un message ou un nombre affiché par « dire », un dessin, un déplacement.



Le nombre tapé entre par « demander » ; les résultats sortent par les deux « dire ».

IDENTIFIER LES ENTRÉES ET LES SORTIES

1. **Chercher les blocs « demander »** : chacun correspond à une entrée, la valeur que l'utilisateur tape.
2. **Chercher ce que l'utilisateur voit** : un « dire », un tracé, un déplacement. Ce sont les sorties.
3. **Résumer en une phrase** : « le programme reçoit …, et il affiche … ».

EXEMPLE

Quelles sont l'entrée et les sorties du programme de la figure ?

1. Un seul bloc « demander » : l'entrée est **le nombre que l'utilisateur tape**, rangé dans « réponse ».
2. Deux blocs « dire » : les sorties sont **les deux résultats** que dit le lutin.

le programme reçoit un nombre et affiche deux résultats calculés à partir de lui

PIÈGE

Taper le mot réponse au clavier dans la case blanche du bloc « dire ». — la case blanche affiche exactement ce qu'on y tape : le lutin dit le mot « réponse », pas le nombre saisi.

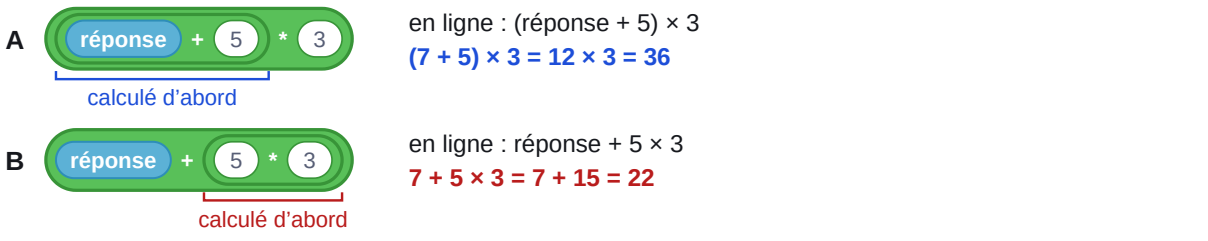
Ce que ça coûte : Le programme ne marche pour aucune valeur.

Le réflexe : glisser le bloc arrondi « réponse » dans la case, ne jamais le recopier.

3 Des formules en blocs

RÈGLE

Un bloc opérateur effectue **une seule opération** : « + », « - », « * » pour multiplier, « / » pour diviser. Pour une formule à plusieurs opérations, on emboîte les blocs : **le bloc le plus intérieur est calculé en premier.** L'imbrication joue le rôle des parenthèses. Une expression prend sa valeur **au moment de l'exécution** : pour la **prévoir**, on remplace « réponse » par le nombre tapé et on calcule de l'intérieur vers l'extérieur.



Mêmes nombres, mêmes opérations : seule l'imbrication change, et le résultat avec.

TRADUIRE UNE FORMULE EN BLOCS

1. **Trouver l'opération faite en dernier**, avec les priorités habituelles (parenthèses, puis × et ÷, puis + et −) : c'est le bloc extérieur.
2. **Remplir chacune de ses cases** : un nombre, le bloc « réponse », ou un nouveau bloc opérateur pour un morceau de formule, que l'on remplit à son tour.
3. **Relire de l'intérieur vers l'extérieur** : on doit retrouver la formule de départ.

EXEMPLE

Représenter $2 \times \text{réponse} + 1$ dans un langage de programmation par blocs.

1. La multiplication est prioritaire, l'addition est faite **en dernier** : le bloc extérieur est « + ». Sa case de gauche reçoit un bloc « * » contenant 2 et « réponse » ; sa case de droite, le nombre 1.

un bloc « + » contenant le bloc « 2 * réponse » et le nombre 1

EXEMPLE

On lance le programme de calcul de la partie 2 et on tape 7. Prévoir ce que dit le lutin, avant de l'exécuter.

1. « réponse » vaut 7. Le premier « dire » contient réponse + 5 :
$7 + 5 = 12$
2. Le second contient (réponse + 5) * 3 : bloc intérieur d'abord, puis la multiplication par 3.
$(7 + 5) \times 3 = 12 \times 3 = 36$

le lutin dit 12 pendant 2 secondes, puis 36

PIÈGE

Dans le second « dire », remplacer « réponse » par 12 et trouver $(12 + 5) \times 3 = 51$. — le premier « dire » affiche un résultat, il ne modifie pas « réponse » : elle vaut toujours le nombre tapé, 7.

Ce que ça coûte : Toute la fin du programme est fausse.

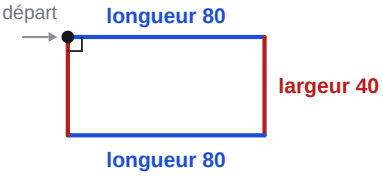
Le réflexe : « réponse » ne change que si l'on tape une nouvelle valeur.

4 Analyser un programme, modifier ses paramètres

DÉFINITION

Les **paramètres** d'un programme sont les valeurs écrites dans les cases blanches de ses blocs : une longueur, un angle, un nombre de répétitions, un texte. Les modifier change le résultat **sans changer l'ordre des blocs**.





Pour tracer un carré de côté 50 :

· remplacer les deux 80 et les deux 40 par 50 ;

· laisser les trois 90 : les angles restent droits.

Les deux 80 commandent la longueur, les deux 40 la largeur, les 90 les angles droits.

ANALYSER UN PROGRAMME, PUIS LE MODIFIER

1. **L'exécuter à la main**, et dire en une phrase ce qu'il fait.
2. **Relier chaque grandeur** du résultat aux paramètres qui la commandent.
3. **Modifier tous ces paramètres, et eux seuls**, puis exécuter de nouveau à la main pour vérifier.

EXEMPLE

Que trace ce programme ? Comment le modifier pour tracer un carré de côté 50 ?

1. Il avance de 80, tourne d'un angle droit, avance de 40, et ainsi de suite : c'est un **rectangle** de 80 sur 40, de périmètre :
$80 + 40 + 80 + 40 = 240$
2. Pour un carré, on remplace **les quatre paramètres de longueur** par 50 et on garde les 90 ; périmètre :
$4 \times 50 = 200$

un rectangle de 80 sur 40 ; pour le carré, les deux 80 et les deux 40 deviennent 50

PIÈGE

Ne remplacer que les deux premiers paramètres de longueur, 80 et 40, par 50. — chaque côté a son propre bloc « avancer » : les deux derniers gardent 80 et 40, la figure ne se referme pas.

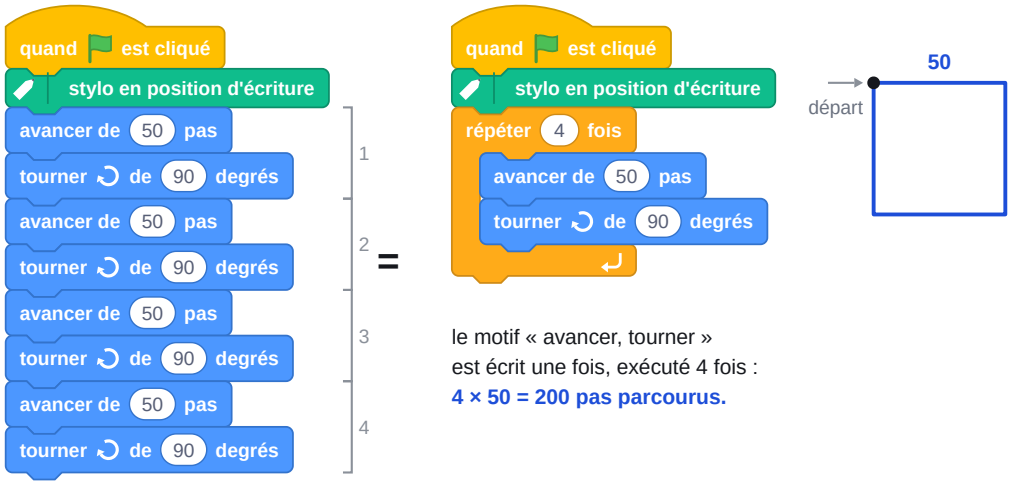
Ce que ça coûte : La figure demandée n'est pas obtenue.

Le réflexe : compter tous les blocs qui commandent une même grandeur, puis les modifier tous.

5 Répéter sans réécrire : la boucle

DÉFINITION

Les blocs placés **à l'intérieur** de « répéter n fois » sont exécutés dans l'ordre, **n fois en tout** ; le programme continue ensuite avec le bloc placé **sous** la boucle. Le nombre de répétitions est fixé d'avance, sans condition : c'est une **boucle inconditionnelle**.



La boucle fait exactement le même tracé que les huit blocs écrits à la suite.

REMPLACER UNE RÉPÉTITION PAR UNE BOUCLE

1. **Repérer le motif** qui revient à l'identique, et **compter** combien de fois il revient.
2. **Placer le motif une seule fois** dans « répéter n fois », dans le même ordre.
3. **Laisser hors de la boucle** ce qui ne se fait qu'une fois, comme « stylo en position d'écriture ».

EXEMPLE

Programme avec boucle de la figure : quelle longueur le lutin parcourt-il ? Et si l'on remplace 4 par 2 ?

1. « avancer de 50 pas » est écrit une fois mais exécuté 4 fois :
   $4 \times 50 = 200$
2. Avec 2 répétitions, seuls deux côtés sont tracés, en angle droit :
   $2 \times 50 = 100$

200 pas pour le carré ; 100 pas avec 2 répétitions

PIÈGE

Placer « tourner ↻ de 90 degrés » sous la boucle « répéter 4 fois » au lieu de l'y mettre. — seul l'intérieur est répété : le lutin avance de 50 pas quatre fois, soit 200 pas en ligne droite, puis tourne une seule fois.

Ce que ça coûte : Un trait droit au lieu d'un carré.

Le réflexe : vérifier, bloc par bloc, s'il est **dans** la boucle ou **sous** elle.

À VÉRIFIER

- Ai-je exécuté le programme **bloc par bloc**, de haut en bas, en cochant chaque bloc ?
- Ai-je repéré l'**entrée** (ce qu'on tape) et la **sortie** (ce qui s'affiche ou se dessine) ?
- Pour un bloc emboîté, ai-je calculé **le bloc le plus intérieur d'abord** ?
- Pour modifier un programme, ai-je changé **tous** les paramètres qui commandent la grandeur visée ?

À RETENIR

- Les blocs s'exécutent **un par un, de haut en bas** : l'ordre compte.
- L'**entrée** est ce que l'utilisateur tape (rangé dans « réponse ») ; la **sortie** est ce qui s'affiche ou se dessine.
- **Le bloc le plus intérieur est calculé en premier** : l'imbrication remplace les parenthèses.
- Pour modifier un programme, on change **tous** les paramètres de la grandeur voulue, et eux seuls.
- « répéter n fois » exécute les blocs qu'elle contient **n fois en tout**, puis passe au bloc placé dessous.