

## SÉQUENCES

1. Instructions et algorithmes
2. Utilisation des boucles
3. Utilisation des instructions conditionnelles
4. Utilisation des variables
5. Utilisation d'un bloc d'instructions

## PROJETS

1. Le crabe aux pinces magiques
2. Le mage et la grenouille
3. Le chiffre de César



# Instructions et algorithme

- Une **séquence d'instructions** est une suite d'actions à exécuter dans un ordre donné.

**Exemple :** Marc joue à un célèbre jeu de football sur console. Il communique des **instructions** aux joueurs à l'aide d'une manette. Voici la **séquence** d'instructions qui permet de célébrer un but en faisant des saltos arrière :

- *Maintenir la touche* .
- *Appuyer deux fois brièvement sur la touche* .

- Un **algorithme** est une suite finie d'instructions permettant de résoudre un problème.

**Exemple :** Une recette de cuisine est un algorithme.

En effet, on dispose d'ingrédients au départ, on applique les instructions données par la recette et on obtient le plat désiré à la fin.

## J'applique

### 1 À la cantine Niveau 1

Prendre du pain

Choisir un fruit

Choisir une entrée

Choisir un plat principal

Prendre un plateau

Choisir un laitage

Passer la carte de cantine



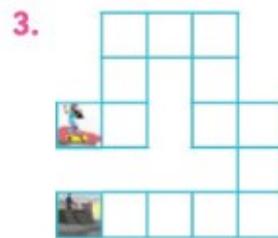
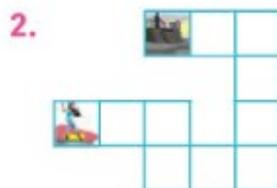
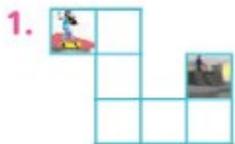
Écris un algorithme de ton passage à la cantine en remettant dans l'ordre ces instructions.

### 2 Julie la skateuse Niveau 2

Ici, pour que Julie la skateuse retrouve le skate park, elle doit suivre la séquence d'instructions suivante :    



Dans chacun des cas ci-dessous, écrire l'algorithme permettant à Julie de retrouver le skate park à l'aide des quatre instructions :    



### 3 Le chemin du collège Niveau 3



Écris un algorithme qui te permet d'aller de ton lit au collège le matin.

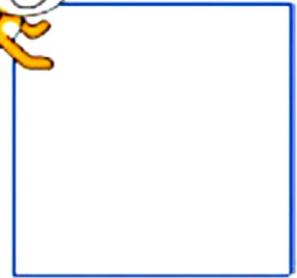
## Instructions et algorithme

On peut programmer les algorithmes avec un logiciel dédié comme le logiciel **Scratch**. Ce logiciel, avec lequel nous allons apprendre à travailler toute l'année, est téléchargeable et utilisable gratuitement : <http://scratch.mit.edu/>

### Exemple

Ce programme permet de faire tracer au lutin un carré de côté 100 à chaque fois que l'utilisateur appuie sur 

```
quand le drapeau est cliqué  
effacer tout  
stylo en position d'écriture  
avancer de 100  
tourner de 90 degrés  
avancer de 100  
tourner de 90 degrés  
avancer de 100  
tourner de 90 degrés  
avancer de 100  
tourner de 90 degrés
```



Entre ce programme dans Scratch et exécute-le afin de vérifier qu'il produit bien le résultat attendu.

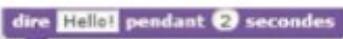
## 4 Construire des rectangles Niveau 1

1. Programmer un algorithme permettant de tracer un rectangle de longueur 100 et de largeur 80, puis enregistrer ce programme.
2. Modifier le programme en faisant dire au chat : « Je vais tracer un rectangle... » avant qu'il ne commence.



Tu peux ici mettre une temporisation :  
  
pour voir le lutin tracer chaque segment.

### Aide

Utiliser l'instruction 

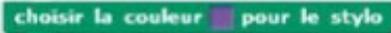
3. Améliorer le programme en masquant le chat une fois que le rectangle est tracé.

### Aide

Utiliser les instructions  et 

4. Améliorer le programme en traçant chaque côté du rectangle d'une couleur différente.

### Aide

Utiliser l'instruction  ou 

## 5 Les signes opératoires Niveau 2

1. Programmer un algorithme permettant de dessiner le signe d'addition ci-contre, puis enregistrer ce programme.

### Aide

Pour que le lutin démarre toujours au même endroit, on peut utiliser l'instruction  et 



2. Programmer un algorithme permettant de dessiner un signe de multiplication, puis enregistrer ce programme.

3. Améliorer les deux programmes précédents pour que les quatre branches du signe soient de couleurs différentes.



# Utilisation des boucles

Dans un algorithme, une **boucle** consiste à faire répéter un certain nombre de fois (connu à l'avance ou non) une même séquence d'instructions. Il existe principalement deux types de boucles : la boucle « Répéter  $x$  fois » et la boucle « Répéter jusqu'à ».

**Exemple :** Pour aider Julie à rejoindre le skate park, on peut donner comme instructions .



On peut aussi utiliser des boucles :

« Répéter 4 fois  »

« Répéter  jusqu'à "Le skate park est atteint" »



On utilise la boucle « Répéter  $x$  fois » quand on sait déjà combien de fois on doit faire répéter les instructions.



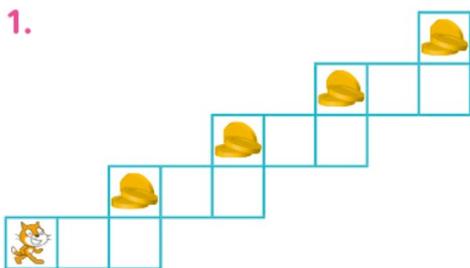
On utilise la boucle « Répéter jusqu'à » quand on ne sait pas combien de fois on doit répéter les instructions mais quand on sait à quel moment on doit s'arrêter.

## J'applique

### Exercice 1 Le labyrinthe

Difficulté ★

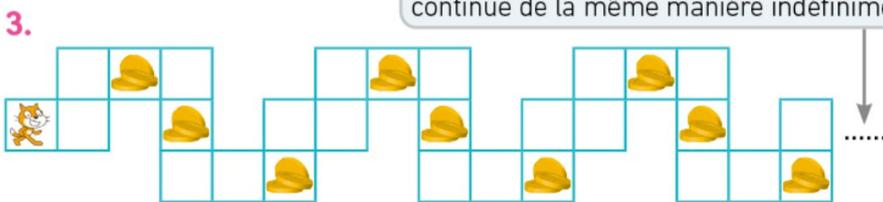
Dans chaque cas, aider Scratchy à récupérer le maximum de pièces d'or en utilisant les instructions , ,  et  pour se déplacer à l'intérieur d'une boucle.



Ces pointillés « ..... » signifient que l'on continue de la même manière un certain nombre de fois.



Ces pointillés « ..... » signifient que l'on continue de la même manière indéfiniment.

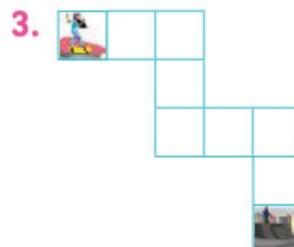
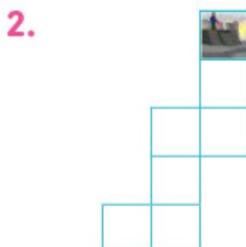
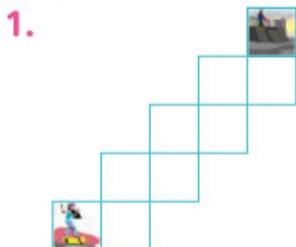


Question 1.

Question 2.

Question 3.

Dans chacun des cas ci-dessous, écrire un algorithme permettant à Julie d'atteindre le skate park à l'aide des quatre instructions : , ,  et  et un autre algorithme utilisant une boucle.



# Utilisation des boucles

## Exercice 2 Les caisses automatiques Difficulté ★★

Dans les supermarchés actuels, on trouve de plus en plus de caisses automatiques qui remplacent progressivement les hôte(sse)s de caisse. Pour rendre la monnaie aux clients, ces automates en libre-service suivent et produisent des algorithmes.

1. À l'aide de boucles et des deux instructions : Donner une pièce de 1<sup>DT</sup> et Donner une pièce de 20 cents, à n'utiliser qu'une seule fois chacune, écrire un algorithme qui permette de rendre 3,80<sup>DT</sup> au client.

2. Le distributeur peut donner des pièces de 2<sup>DT</sup>, 1<sup>DT</sup>, 50 cents, 20 cents, 10 cents, 5 cents, 2 cents ou encore 1 cent. Écrire un algorithme contenant des boucles qui permette de rendre 9,48 € en donnant le moins de pièces possible au client.

Question 1.

Question 2.



## Exercice 3 Compétition de natation Niveau 3

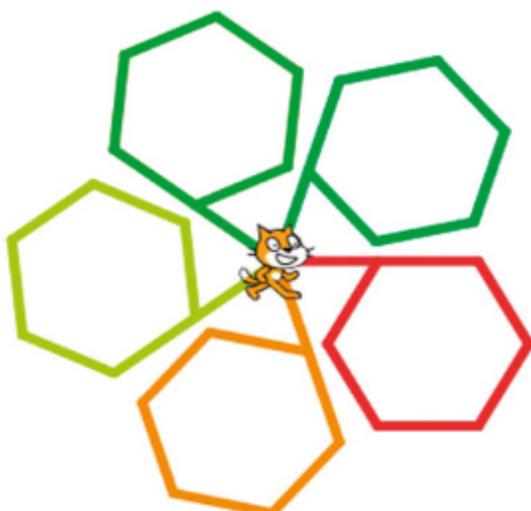
Dans chaque cas, en utilisant une boucle « Répéter x fois » et l'instruction « Traverser le bassin », écrire un algorithme décrivant le parcours d'une nageuse effectuant en grand bassin (50 m de longueur) :

- a. un 100 m nage libre ;
- b. un 800 m nage libre.



## Exercice 4 Un motif coloré

Écrire dans l'emplacement ci-contre un programme qui permette de réaliser le dessin ci-dessous où les 5 motifs ont des couleurs différentes.



# Utilisation des boucles

Les différents types de **boucles** de Scratch sont dans le menu **Contrôle**.

- On peut indiquer le nombre de répétitions souhaitées : ce nombre peut être une variable.



- La répétition est ici effectuée jusqu'à ce qu'un test soit validé. Ces tests sont dans le menu **Opérateurs**. On a, par exemple : ,  ou encore .

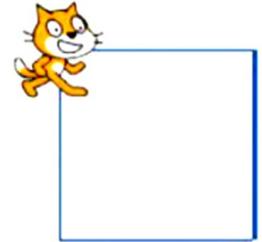


- Cette boucle est utilisée par exemple quand on attend une réponse au clavier. Cela nécessitera généralement l'usage d'une instruction conditionnelle : elle sera placée dans ce bloc.



## Exemple

Ce programme va permettre de faire tracer un carré de côté 100 à chaque fois que l'utilisateur appuie sur .



Entre ce programme dans Scratch et exécute-le afin de vérifier qu'il produit bien le résultat attendu.

## 4 Construire des polygones Niveau 1

1. Simplifier le programme ci-contre à l'aide d'une boucle pour construire un pentagone (polygone à 5 côtés), puis enregistrer ce programme.
2. Faire évoluer ce programme afin d'obtenir un hexagone.
3. Faire évoluer ce programme afin d'obtenir un triangle équilatéral.



## 5 Les décompteurs Niveau 2

1. Programmer un compteur qui va de 1 jusqu'à 20 et s'arrête dès que la valeur 20 est atteinte.
2. Améliorer le programme en faisant compter le chat au fur et à mesure.
3. Modifier le programme en faisant un décompte de 20 jusqu'à 0.
4. Programmer un décompte de 7 en 7 à partir de 343 qui s'arrête quand 0 est atteint.

**Aide**

Utiliser l'instruction  pour voir le décompte défiler.

## 6 Le marquage au sol Niveau 3

Programmer un algorithme qui reproduit le dessin ci-contre. Chaque segment mesure 20 et chaque espace mesure 20 également.

Pense à choisir la taille du stylo !



## 7 Un arc-en-ciel Niveau 4

En faisant évoluer la valeur de la couleur de 0 à 200, réaliser ce nuancier :



**Aide**

On peut réduire la taille du chat et choisir une taille de stylo élevée.

# Utilisation des instructions conditionnelles

Une **instruction conditionnelle** est de la forme :

**Si « Condition »**

**Alors « Instruction(s) »**

Dans ce cas, si la « **Condition** » est réalisée, alors les « **Instruction(s)** » seront effectuées.

## Exemple

Si **le soleil brille cet après-midi**, alors **i'irai à la piscine**.

**ou Si « Condition »**

**Alors « Instruction(s) 1 »**

**Sinon « Instruction(s) 2 »**

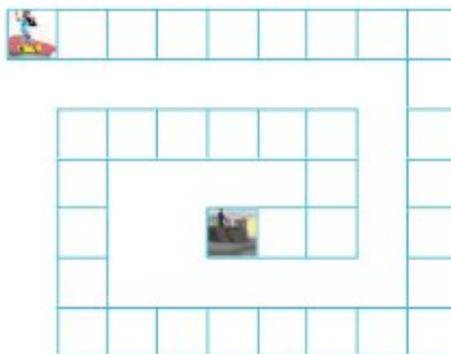
Dans ce cas, si la « **Condition** » est réalisée, alors les « **Instruction(s) 1** » seront effectuées, sinon ce sont les « **Instruction(s) 2** » qui seront effectuées.

## Exemple

Si **le soleil brille cet après-midi**, alors **i'irai à la piscine**, sinon **i'irai au cinéma**.

## 1 Les instructions de Julie Niveau 1

Il s'agit ici d'écrire des algorithmes permettant à Julie d'atteindre le skate park utilisant les instructions **avancer d'une case**, **tourner à droite**, des boucles et des instructions conditionnelles avec la condition **Pas de case devant**.



1. Écrire un algorithme en utilisant une instruction conditionnelle de type « Si ... Alors ... »
2. Écrire un algorithme en utilisant une instruction conditionnelle de type « Si ... Alors ... Sinon ... »

## 2 Les tirs au but Niveau 2

1. Durant la Coupe du monde de football, à la fin du temps réglementaire d'un match, quand les équipes sont à égalité, le match continue durant une période que l'on appelle prolongations.

Écrire un algorithme utilisant une instruction conditionnelle « Si ... Alors ... » pour décrire cette situation.



2. À la fin des prolongations, si les équipes n'ont pas réussi à se départager, elles entament une séance de tirs aux but. Chaque équipe tire cinq fois. Si à la fin de cette séance, les équipes sont toujours à égalité, vient la « mort subite ». Chaque équipe tire au but une fois jusqu'à ce que l'une marque et l'autre non. Écrire un algorithme utilisant une instruction conditionnelle « Si ... Alors ... Sinon ... » pour décrire la phase de « mort subite ».

## Utilisation des instructions conditionnelles

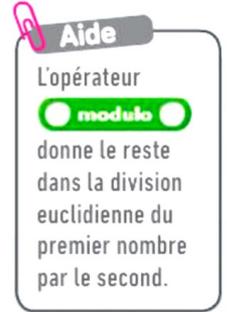
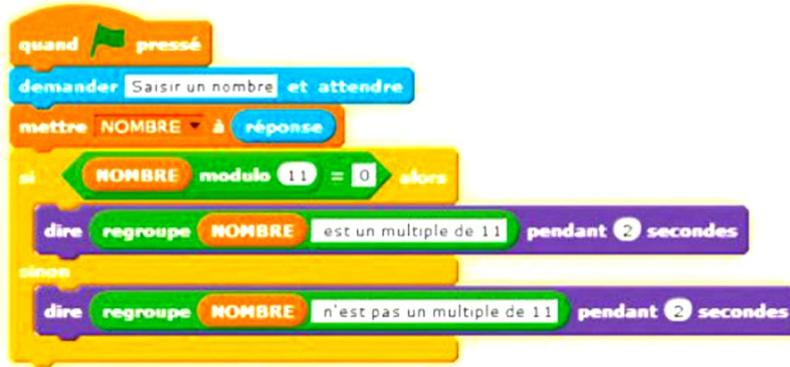
Les différents types d'instructions conditionnelles de Scratch sont dans le menu **Contrôle**.



Les conditions sont à construire à partir des éléments du menu **Opérateurs** comme ,  ou .

### Exemple

Ce programme va permettre de dire si un nombre est ou n'est pas un multiple de 11.



Entre ce programme dans Scratch et exécute-le afin de vérifier qu'il produit bien le résultat attendu.

### 3 Un carré pas trop grand Niveau 1

1. Programmer un algorithme demandant un nombre entre 0 et 200. Si le nombre entré est inférieur à 200, le chat doit dire : « Merci ! » Enregistrer le programme.
2. Modifier ce programme pour qu'il demande la longueur du côté d'un carré et qu'il le trace uniquement si le côté est plus petit que 200.
3. Améliorer le programme pour que dans le cas où il ne trace pas le carré, le chat dise : « Désolé ce nombre est trop grand ! »

### 4 Un nombre au hasard Niveau 2

1. Programmer un algorithme qui met dans une variable **Nombre** une valeur aléatoire de 1 à 3, puis qui cache cette variable à l'écran.
2. Compléter le programme pour qu'il demande ensuite à l'utilisateur de choisir un nombre entre 1 et 3, puis faire dire au chat si les deux nombres sont identiques ou non.

### 5 La balade du chat Niveau 3

Programmer un algorithme faisant déplacer le chat vers la droite ou vers la gauche à l'aide des flèches du clavier.



## Utilisation des variables

Une **variable** est une boîte dans laquelle on stocke une information pour l'utiliser plus tard. On désigne une variable par un nom.

**Exemple :** Le score d'un joueur est une variable qui pourra évoluer tout au long du jeu, une nouvelle valeur efface la précédente.

score joueur 1

0



1

### Le jeu des pièces d'or Niveau 1

Ce jeu se joue à deux joueurs ou plus.

**Matériel :** un dé à six faces numérotées de 1 à 6.

**Règle du jeu :** Au début d'une partie, chaque joueur reçoit 50 pièces d'or.

Tout au long du jeu, ce nombre de pièces va évoluer et les joueurs vont devoir gérer le nombre de pièces qu'ils possèdent. Pour cela, chaque joueur prend une feuille blanche pour noter son nouveau nombre de pièces en barrant l'ancien à chaque fois.

Le joueur qui débute la partie doit aussi comptabiliser, à un endroit de sa feuille, le nombre de tours effectués. À tour de rôle, en tournant dans le sens des aiguilles d'une montre, chaque joueur lance le dé et exécute l'action correspondant au résultat du dé :

	Échange tes pièces avec un des joueurs qui en a le plus.
	Ton nombre de pièces augmente de 10.
	Échange tes pièces avec le joueur de ton choix.
	Ton nombre de pièces diminue de 10.
	Donne 10 pièces au joueur qui joue juste après toi
	Ton nombre de pièces est multiplié par 2.

La partie se joue en 10 tours, le joueur qui a le plus de pièces à la fin de ces 10 tours gagne.

Faire une ou plusieurs parties avec des camarades de la classe en se plaçant par groupe autour d'une table. Une fois la partie terminée, répondre aux questions ci-dessous.

1. Dans ce jeu, il y a plusieurs **variables** :

- une variable qui contient le nombre de tours effectués. On peut l'appeler « TOURS » ;
- des variables qui contiennent les nombres de pièces de chaque joueur, on leur donnera comme nom le « prénom » de chaque joueur.

Combien de variables ont été utilisées dans cette partie ?

2. Si l'un des joueurs se nomme Victor, à quel résultat du dé correspond cette action programmée dans le logiciel Scratch :

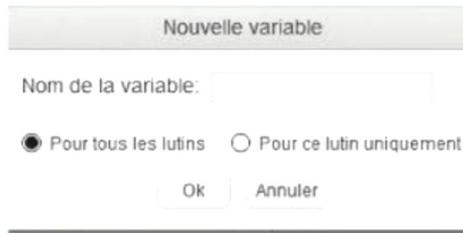
mettre Victor à Victor + 10 ?

# Utilisation des variables

• Avec le logiciel Scratch, on peut créer des variables à partir du menu **Données**.

On commence par choisir le nom de ces variables.

• Dès que la variable est créée, plusieurs actions se découvrent.



Il est pratique de choisir un nom ayant une signification



## Exemple

Ce programme va demander un nombre et donner son double à chaque fois que l'utilisateur appuiera sur 



Entre ce programme dans Scratch et exécute-le afin de vérifier qu'il produit bien le résultat attendu.

## 2 On compte Niveau 1

1. Créer une variable « COMPTEUR » et écrire un programme qui fasse augmenter de 1 cette variable à chaque clic sur le drapeau vert. Enregistrer ce programme.
2. Améliorer le programme en faisant dire au chat : « Le compteur est arrivé à ... »



**Aide**  
Utiliser l'opérateur `regroupe hello world`

## 3 Le périmètre d'un rectangle Niveau 2

1. Programmer un algorithme demandant la longueur du côté d'un carré avant de le tracer. Enregistrer ce programme.
2. Modifier ce programme en demandant une longueur et une largeur avant de tracer un rectangle aux dimensions saisies.
3. Faire évoluer ce programme en créant une variable « PERIMETRE » et en faisant dire au chat à la fin du tracé : « Le périmètre de ce rectangle est égal à ... »

**Aide**  
Utiliser l'instruction `demander What's your name? et attendre`.  
On attribue alors la réponse à la variable créée `mettre longueur à réponse`

**Aide**  
Utiliser l'opérateur `◯ + ◯` et/ou l'opérateur `◯ * ◯`

## 4 Qui es-tu ? Niveau 3

Programme un algorithme demandant ton prénom, puis ton âge et enregistre ce programme. Fais alors dire au chat

[www.najahn.fr](http://www.najahn.fr)

# Utilisation d'un bloc d'instructions

Un **bloc d'instructions** peut parfois être remplacé par un seul intitulé.

## Exemple

L'envoi d'un courriel par messagerie est constitué de plusieurs actions simples, intégrées dans un même bloc nommé « Envoi d'un courriel ».

Envoi d'un courriel
• Ouvrir sa messagerie
• Écrire à l'aide des touches du clavier de l'ordinateur
• Choisir la ligne destinataire marquée « à » ou « pour »
• Ecrire l'adresse du destinataire
• Cliquer sur « envoyer »

**Exemple ① :** Leïla souhaite tracer un carré de 5 cm de côté.

Elle répète 4 fois le processus suivant :

- elle utilise sa règle pour tracer un segment de 5 cm ;
- elle utilise son équerre pour faire un angle droit à l'extrémité de ce segment.

Pour un carré de côté 50 pixels, on peut créer le bloc ci-contre.

Un bloc d'instructions peut être paramétré, c'est-à-dire qu'il contient au moins une instruction qui contient une information à préciser pour utiliser ce bloc.

## SCRATCH



**Exemple ② :** Leïla souhaite tracer un carré de longueur donnée.

Elle répète 4 fois le processus suivant :

- avec sa règle elle trace un segment de la longueur choisie ;
- elle utilise son équerre pour faire un angle droit à l'extrémité de ce segment.

Pour tracer un carré dont le côté a une longueur précisée dans le programme, on peut créer le bloc ci-contre.



### Coup de pouce

Dans ce cas, si on définit la longueur par 50, l'instruction **carré 50** donnera un carré de 50 pixels de côté.

## 1 Travaux ménagers Niveau 1



En utilisant les instructions suivantes, écris deux groupements : l'un correspondant à « mettre la table pour une personne » et l'autre correspondant à « préparer une lessive ».

Mettre un verre

Mettre le linge dans le tambour

Préparer un pichet d'eau

Mettre les couverts

Mettre la lessive

Poser une assiette sur la table

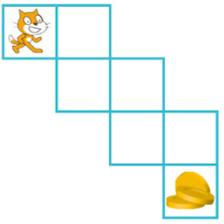
Mettre un adoucisseur si besoin

Choisir le programme de lavage

# Utilisation d'un bloc d'instructions

## Exercice 1 Le labyrinthe Difficulté ★

Dans chaque labyrinthe, aider Scratchy à récupérer toutes les pièces d'or en définissant le bloc Prog1 que l'on répètera deux ou trois fois. On utilisera les instructions de déplacement : ◀, ▶, ▲ et ▼.

1. 

Répéter 3 fois

  | Prog1

Définir Prog1 :

.....

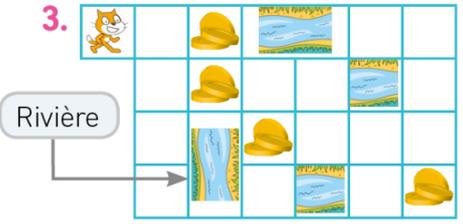
2. 

Répéter 2 fois

  | Prog1

Définir Prog1 :

.....

3. 

Rivière

→

Répéter 3 fois

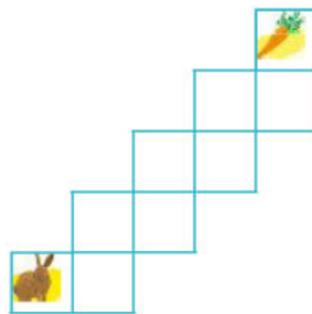
  | Prog1

Définir Prog1 :

.....

## 2 Le repas du lapin Niveau 2

Dans chacun des cas ci-dessous, écrire le groupement d'instructions (GI) permettant au lapin de retrouver sa carotte à l'aide des quatre instructions ◀ ▶ ▲ ▼

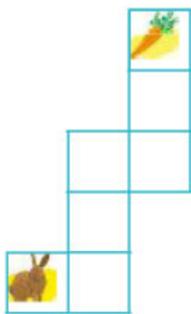
1. 

Répéter 4 fois

{ Groupement

{ d'instructions GI1

Fin

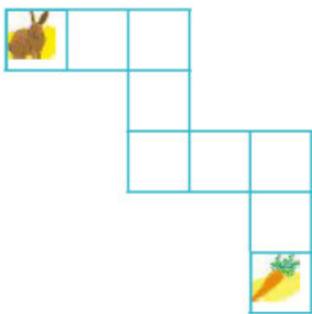
2. 

Répéter 2 fois

{ Groupement

{ d'instructions GI2

Fin

3. 

Répéter 2 fois

{ Groupement

{ d'instructions GI3

Fin



Utiliser un bloc d'instructions ici permet d'éviter d'avoir à mettre une boucle dans une boucle.

## Utilisation d'un bloc d'instructions

Un bloc permet de mémoriser un petit algorithme intervenant dans un algorithme plus élaboré, éventuellement plusieurs fois, ce qui facilite également la compréhension de l'algorithme principal.

On peut créer des blocs à partir du menu **Ajouter blocs**.

Lorsqu'il est créé, le bloc est utilisable comme les autres éléments.

### Exemple

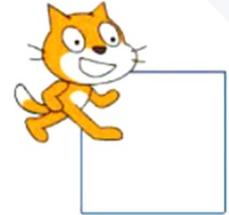
Le bloc **carré** permet de tracer, à chaque fois qu'il est appelé, un carré de côté 100 à l'endroit où se trouve le lutin.



Créer un bloc



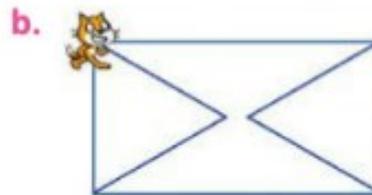
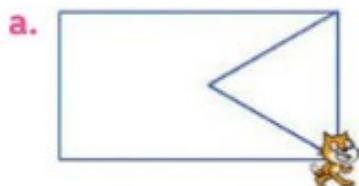
Ajouter une extension



Entre ce programme dans Scratch et exécute-le afin de vérifier qu'il produit bien le résultat attendu.

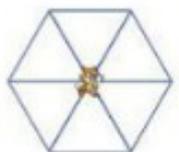
### 3 Géométries d'une enveloppe Niveau 1

1. Écrire un algorithme définissant un bloc rectangle de 80 sur 150 et un bloc triangle équilatéral de côté 80.
2. Utiliser alors ces blocs de façon à obtenir les figures ci-dessous :



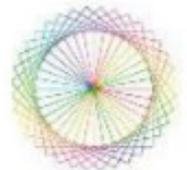
### 4 Quel cirque ! Maximus ! Niveau 2

Écrire un algorithme faisant appel à un bloc « triangle équilatéral » pour construire un hexagone.



### 5 Silence ! On tourne ! Niveau 3

1. Créer un bloc « CARRÉ » qui trace un carré de côté 100 à partir de l'endroit où se trouve le lutin.
2. Programmer un algorithme qui efface tout, va au centre de l'écran (0 ; 0), pose le stylo et répète 36 fois la même séquence :
3. Améliorer le programme pour que la couleur du stylo change à chaque carré.



La couleur dans Scratch correspond à des nombres entiers compris entre 0 et 200. Pour changer la couleur, il suffit de changer la valeur.

## Le crabe aux pinces magiques

Dans un univers maritime, l'ancre d'un crabe est attaquée par des ballons ! Heureusement, ses pinces magiques lui permettent de détruire ces ballons avant qu'ils n'atteignent leur but !

### Étape 1 ■ Positionner et déplacer le crabe

- Prendre comme lutin un crabe et le positionner en bas de l'écran.
- Programmer le déplacement horizontal (gauche et droite) du crabe à l'aide des flèches du clavier.



Pense à réduire la taille du crabe si besoin.



### Étape 2 ■ Positionner et déplacer un ballon

- Ajouter un ballon qui descend automatiquement en démarrant toujours de la même abscisse -170. Cette abscisse sera modifiée par la suite.



Pense aussi à réduire la taille du ballon si besoin.



### Étape 3 ■ Programmer les actions

- Modifier le programme pour que lorsque le crabe touche le ballon, celui-ci réapparaisse en haut à son point de départ.
- Prévoir aussi un retour au point de départ si le ballon arrive en bas de la scène, sans être touché par le crabe.





### Étape 4 ■ Ajouter un décor et un compteur

- Ajouter un compteur permettant de savoir combien de fois le crabe a touché le ballon.
- Ajouter aussi un arrière-plan adapté.



Un double-clic sur **compteur**  permet d'obtenir cet affichage . On obtient le même résultat en faisant un clic droit et en choisissant dans le menu déroulant.



### Étape 5 ■ Faire apparaître le ballon n'importe où

- Faire en sorte que le ballon puisse surgir de n'importe quelle abscisse entre -200 et +200.



C'est l'occasion d'utiliser l'élément **nombre aléatoire entre**  **et** .



### Étape 6 ■ Créer une fin pour le jeu

- Terminer le jeu avec les indications suivantes :
- lorsque le compteur atteint 5, augmenter la vitesse de déplacement du crabe ainsi que la vitesse de descente des ballons ;
- lorsque le compteur atteint 10, augmenter encore la vitesse de descente des ballons ;
- lorsque le compteur atteint 20, faire dire au crabe : « Gagné ! » et arrêter le jeu.



### ÉVOLUTIONS POSSIBLES

- Faire varier la couleur des ballons, le bleu valant plus de points que les autres.
- Remplacer les ballons par des images de coquillages.
- Faire varier les fonds en fonction de l'avancée dans le jeu.

## Le mage et la grenouille

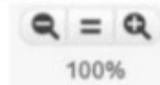
Un mage doit viser une grenouille avec sa baguette magique. Lorsque la grenouille a reçu assez de magie, elle se transforme en prince ou en princesse...

### Étape 1 ■ Préparer la grenouille

- Choisir comme lutin la grenouille de Scratch et aller dans l'onglet costumes.
- Dupliquer la grenouille.
- Avec l'outil Gomme, supprimer la langue sur le costume 1.



Pour plus de précision dans l'usage de la gomme, on peut utiliser l'outil loupe en bas à droite de l'écran.



### Étape 2 ■ Le mage et sa magie

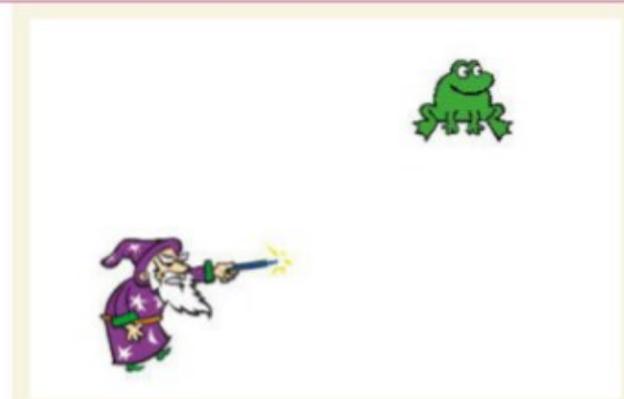
- Écrire un algorithme de déplacement de la grenouille dans les quatre directions. On pourra utiliser

touche flèche haut pressée?

- Ajouter le lutin Wizard (le nommer *mage*) et créer un lutin *magie* de ce style :



- Créer une variable `x mage` et une variable `y mage`.
- Attribuer à `x mage` une valeur aléatoire entre -210 et 60 et à `y mage` une valeur aléatoire entre -120 et 120.
- Créer une variable `angle` qui pourra prendre une valeur aléatoire entre -30 et +30. Faire tourner le *mage* de cette valeur.



### Étape 3 ■ Introduction du hasard dans le jeu

- Créer une variable `dé`, lui attribuer une valeur aléatoire entre 1 et 10.

nombre aléatoire entre 1 et 10

Si la valeur de `dé` est inférieure à 8, le *mage* envoie de la magie par sa baguette : à cet effet, un message est envoyé au lutin *magie*.

Prévoir ensuite une temporisation d'une seconde environ.

Sinon, on refait bouger le mage et on relance le dé.

À la réception de ce message, *magie* se positionne par rapport à la baguette du *mage* en prenant la même orientation. *Magie* avance alors jusqu'à toucher le bord : il est désormais caché.





## Étape 4 ■ Traitement de l'événement victorieux

– Dès que la grenouille est touchée par *magie*, elle tremble en tirant et rentrant la langue.

L'objectif est d'être touché le plus souvent possible par *magie*.

Un **compteur** doit donc mémoriser le nombre de fois où la grenouille est touchée.



## Étape 5 ■ Traitement de la fin de partie

On peut désormais cacher le compteur.

À partir de la 11<sup>e</sup> fois où la grenouille a été touchée :

- si le dé donne 1, la grenouille se transforme en prince (autre lutin), et le jeu s'arrête ;
- si le dé donne 2, la grenouille se transforme en princesse (autre lutin) et le jeu s'arrête ;
- dans les autres cas, le jeu continue.



## Étape 6 ■ Comptabilisation des points

- Choisir un arrière-plan.
- Terminer le jeu en créant une variable « POINTS ».
- Au démarrage, on a 1 000 points.

mettre points à 1000

- À chaque fois que la grenouille tremble, on ajoute 200 points.
- À chaque fois que la grenouille n'est pas touchée, on perd 100 points.
- Lorsque le compteur dépasse les 10, on a 20 points à chaque tir jusqu'à la transformation.



## ÉVOLUTIONS POSSIBLES

- Faire varier la taille de la grenouille en fonction de la magie reçue.
- Commander le mage et sa magie à l'aide de touches du clavier.
- Imaginer un niveau 2 dans lequel deux mages combattent.

## Le chiffre de César

Crypter... décrypter... voilà ce à quoi s'attaque le chiffre de César ! Cette forme simple de cryptage était déjà utilisée dans l'Antiquité. À ton tour de chiffrer un message comme les Romains !

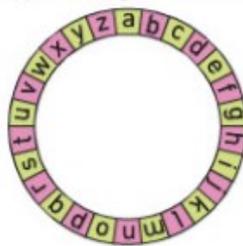
### Étape 1 ■ Création et stockage de l'alphabet

- Le chiffre de César est un procédé qui consiste à décaler les lettres de l'alphabet vers la droite ou vers la gauche d'un nombre de crans déterminés.

On choisit ici de décaler vers la droite.

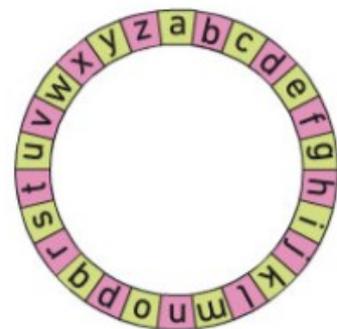


Si on décale de 2, on voit que le « a » devient « c » ou encore que le « y » devient « a ».



- Créer une variable `alphabet` contenant les 26 lettres de l'alphabet dans l'ordre.

Il s'agira de reconstituer le principe du cercle de lettres ci-dessous à partir des 26 lettres de l'alphabet dans l'ordre mises dans la variable.



```
mettre alphabet à abcdefghijklmnopqrstuvwxyz
```

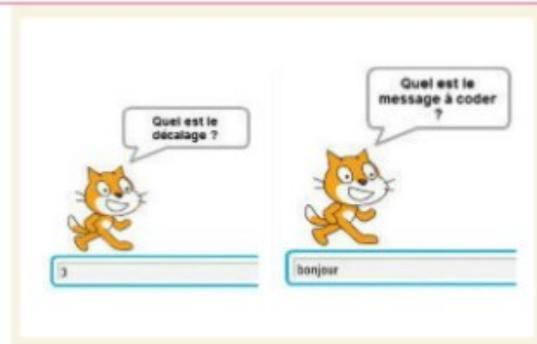
### Étape 2 ■ Saisie du décalage et du message à coder

- Demander le nombre correspondant au `décalage` souhaité.

- Demander le `message` à coder.

**Attention !** Ce message sera tout d'abord un mot simple, c'est-à-dire sans trait d'union, sans apostrophe, etc.

On dispose désormais du message (mot) à coder et du décalage souhaité : il faudra, à l'étape suivante, créer le message codé.



### Étape 3 ■ Création des variables nécessaires au codage

- Créer une variable `message codé` qui sera initialisée par du vide.
- Créer ensuite une variable `lettre` qui prendra successivement comme valeur chacune des lettres du mot à coder.
- Créer une variable `compteur` qui permettra de compter les lettres codées et que l'on initialisera à 1.

```
mettre message codé à
```

```
mettre compteur à 1
```



### Étape 4 ■ Création du bloc permettant le codage

- On affecte à la variable **lettre** la lettre à coder.
- On appelle un bloc **CODER** qui va :
  - déterminer la position de cette lettre dans l'alphabet et la stocker dans une variable **position**, en utilisant éventuellement l'instruction **lettre 1 de world** ;
  - ajouter le **décalage** à la variable **position** pour trouver la position de la lettre codée ;
  - affecter à la variable **lettre** sa nouvelle valeur.

Par exemple avec la lettre **b** et un décalage de 3, on aura comme valeur successives des variables :

- **lettre** **b**
- **CODER**
- **position** **2**
- **position** **5**
- **lettre** **e**

**Attention !** Dans le cas où la somme **position** + **décalage** dépasse 26, il faut trouver un moyen de redémarrer au début de l'alphabet (27 correspond à 1, 28 à 2, etc.)

Pour cela, utiliser l'instruction **modulo** qui donne le reste d'une division euclidienne.

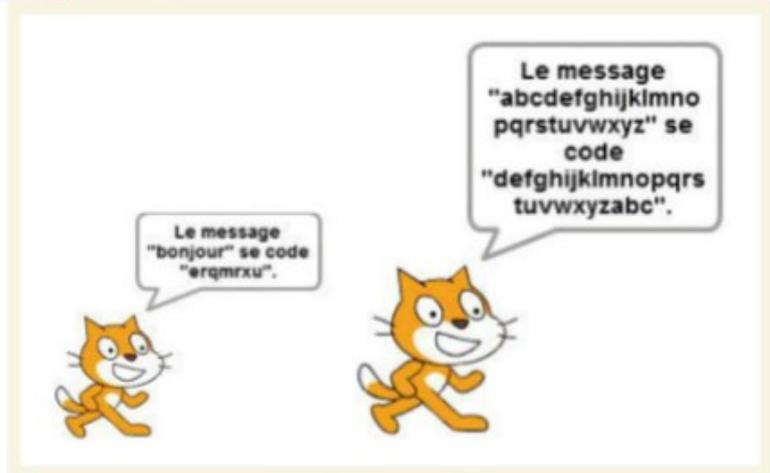
### Étape 5 ■ Création et affichage du message codé

- Créer une boucle qui va, pour chacune des lettres du message initial, trouver celle qui doit la remplacer à l'aide du bloc **CODER** et l'ajouter à la suite du message codé.

On pourra utiliser pour la boucle :



et pour construire le message codé :



Vérifier que le codage fonctionne bien, par exemple en tapant l'alphabet entier comme message avec un décalage de 3 pour observer si chaque lettre se code bien.

### Étape 6 ■ Amélioration du programme pour coder des phrases entières

- Modifier le bloc **CODER** pour prendre en compte les espaces, les apostrophes ou les traits d'union dans une phrase.

Si le programme ne trouve pas la lettre à coder dans l'alphabet lors de la recherche de position, il doit conserver la lettre d'origine.

Un espace sera donc codé par un espace, une apostrophe par une apostrophe, etc.

On pourra pour cela utiliser **ou** dans la boucle.



### ÉVOLUTIONS POSSIBLES

- Intégrer un codage spécifique pour les lettres accentuées.
- Construire le programme de décryptage.
- Améliorer le cryptage à partir d'une fonction affine (modulo 26).