



# Construire le plan de réalisation d'un objectif complexe



En utilisant Scratch (V. 3.29), langage de  
programmation graphique

Sébastien Nowak - Septembre 2023

# Bienvenue !





Vous êtes ici



## Période préparatoire : aborder la formation de formateur numérique avec les bons outils et la bonne posture

N° Fiche AT	Activités types	N° Fiche CP	Compétences professionnelles
1	Concevoir et préparer la formation	1	Elaborer la progression pédagogique d'une formation multimodale à partir d'une demande
		2	Concevoir un scénario pédagogique et d'accompagnement en intégrant la multimodalité
		3	Concevoir des activités d'apprentissage et d'évaluation en intégrant la multimodalité
2	Animer une formation et évaluer les acquis des apprenants	4	Animer une formation et faciliter les apprentissages selon différentes modalités
		5	Evaluer les acquis de formation des apprenants
		6	Remédier aux difficultés individuelles d'apprentissage
3	Accompagner les apprenants en formation	7	Accompagner les apprenants dans leur parcours de formation
		8	Accueillir un apprenant en formation et co-construire son parcours
		9	Tutorer les apprenants à distance
		10	Accompagner le développement professionnel des apprenants
4	Inscrire sa pratique professionnelle dans une démarche de qualité et de responsabilité sociale des entreprises	11	Respecter et faire respecter la réglementation en vigueur en formation et dans sa spécialité
		12	Réaliser une veille pour maintenir son expertise de formateur et de professionnel dans sa spécialité
		13	Analyser ses pratiques professionnelles

Période préparatoire : aborder la formation de formateur numérique avec les bons outils et la bonne posture

Séance 0

Fondements collectifs : référentiels et règles de fonctionnement (1j)

Séance 1

Le métier de formateur numérique (2j)

Séance 2

Pratique de l'écrit et de l'oral (2j)

Séance 3

Démarche de réalisation d'un objectif complexe (2j)

Séance 4

Les outils professionnels du formateur numérique (1j)

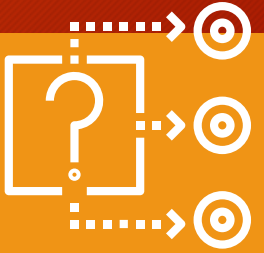
Séance 5

Apprendre à apprendre (2j)

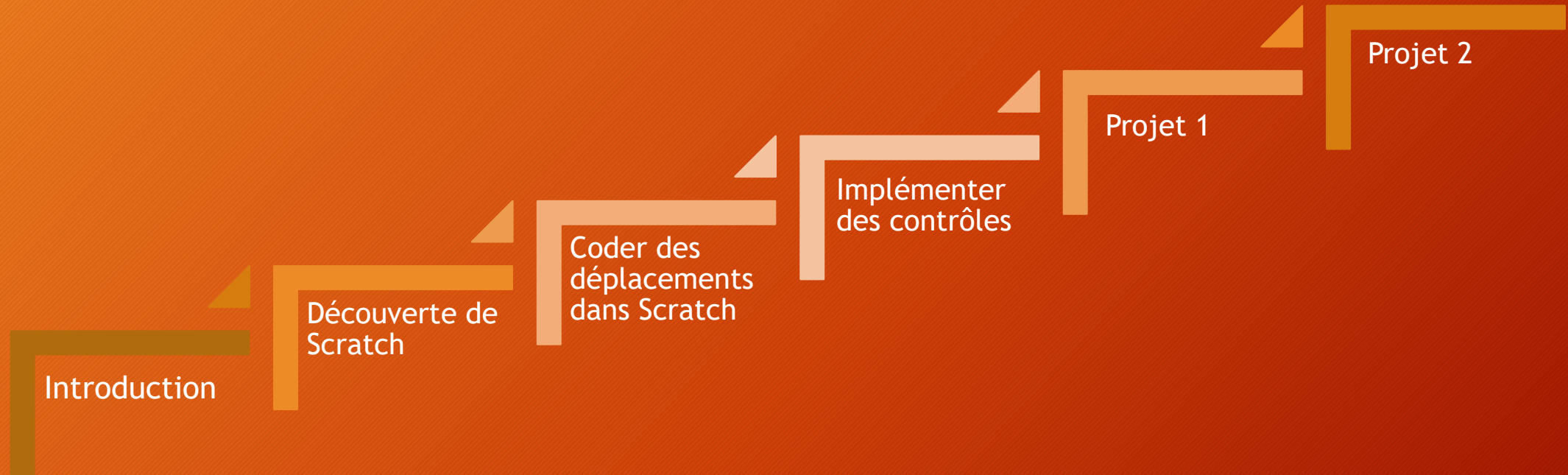
Vous êtes ici



**Simuler la réalisation d'un objectif complexe, en utilisant  
un langage de programmation simple, et en respectant le  
besoin exprimé**







# Plan





# Quelques questions vers vous

- Vous a-t-on déjà fixé des objectifs complexes ?
- Si oui, quels étaient-ils ?
- Comment avez-vous procédé pour les relever ?
- Connaissez-vous le codage ?
- Quelle serait votre définition du codage ?
- Connaissez-vous Scratch ?
- Si je vous disais que créer un code est analogue à la construction pédagogique d'une action de formation, me croiriez-vous ?

# Introduction : qu'est-ce que le codage ?



Ensemble des moyens utilisés pour mettre en forme des informations, afin de pouvoir les manipuler, les stocker, ou les transmettre. Le codage ne s'intéresse pas au contenu (le sens) d'une information, mais à sa forme et à son volume.

Par extension, le codage désigne aussi tout procédé qui vise à transposer des données ou des informations d'une forme vers une autre, sans perte de sens.





# Introduction : qu'est-ce que le codage ?



Bienvenue  
chez Pop  
School

295142151420  
538526161516  
1938151512



# Le codage informatique (1/4)



Un ordinateur ne comprend quasiment rien au langage humain. Il possède son propre langage : le langage machine. C'est le seul que le processeur reconnaisse

Le langage machine est basiquement une suite de bits (des 0 et des 1) que le processeur lit et interprète.



# Le codage informatique (2/4)



« *L'ordinateur est complètement con.* »

Gérard Berry

Il ne fait rien d'intelligent par lui-même (pas encore du moins). Il faut lui donner des instructions pour chaque tâche qu'on souhaite lui confier.

Ces instructions doivent être simples. Le champ des possibles d'un processeur reste limité, quelle que soit sa puissance.



## Le codage informatique (3/4)



On est donc dans une situation où l'ordinateur ne reconnaît pas notre langage, et où le commun des mortels ne comprend absolument rien au langage machine.

Bref, il va falloir un peu de traduction. C'est la finalité du codage informatique.

# Le codage informatique (4/4)



Langage naturel

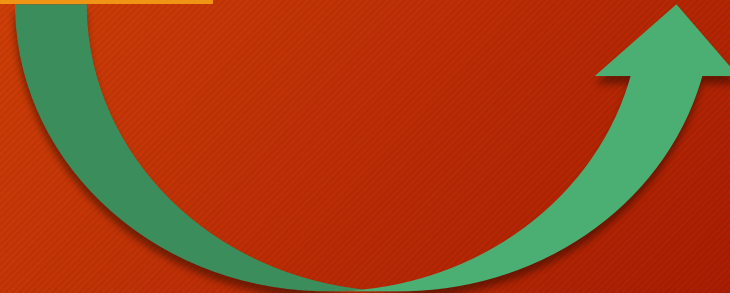
Code/programme  
source

Code exécutable,  
lisible et traitable par  
le processeur

Codage informatique  
Programmation

Compilation

Langage de programmation





# En résumé...



Le codage informatique est une première étape de formalisation des instructions données à un ordinateur.

On utilise pour coder un langage de programmation. Chaque langage possède ses règles et ses conventions. Chaque langage a ses utilisations préférentielles.

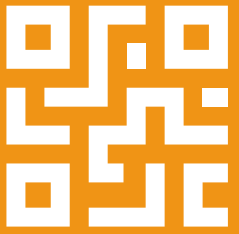
Parmi les langages les plus connus et utilisés : Python, html, Java, JavaScript, C, C++...

A screenshot of a code editor showing Python code. The code is color-coded and includes line numbers on the left. The visible code includes class attributes, a classmethod, and two instance methods.

```
31
32 self.file = None
33 self.fingerprints = set()
34 self.logdupes = True
35 self.debug = debug
36 self.logger = logging.getLogger(__name__)
37 if path:
38     self.file = open(os.path.join(path, 'requests.json'),
39                     'a')
40     self.file.seek(0)
41     self.fingerprints.update(e.request() for e in self.requests)
42
43 @classmethod
44 def from_settings(cls, settings):
45     debug = settings.getbool('SUPERFINGER_DEBUG')
46     return cls(job_dir(settings), debug)
47
48 def request_seen(self, request):
49     fp = self.request_fingerprint(request)
50     if fp in self.fingerprints:
51         return True
52     self.fingerprints.add(fp)
53     if self.file:
54         self.file.write(fp + os.linesep)
55
56 def request_fingerprint(self, request):
57     return request_fingerprint(request)
```



# Un peu de C++



```
1 #include <iostream>
2 #include <string>
3
4 template <typename Traits>
5 class Information{
6     private:
7         Traits pf_name;           // fp == public field name
8         Traits pf_family;        // fp == public field family
9
10    public:
11        void set_information(Traits arg_name, Traits arg_family){
12            pf_name = arg_name;
13            pf_family = arg_family;
14        }
15
16        void get_information(){
17            std::cout << "Your name is " << pf_name << " " << pf_family << std::endl;
18        }
19 };
20
21 auto main(int argc, char* argv[]) -> decltype(0){
22     Information<std::string> o_person;
23
24     o_person.set_information("Milad", "Kahsari Alhadi");
25     o_person.get_information();
26
27     return 0;
28 }
```

# Un peu de html



```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8" />
5      <title>Ma page</title>
6      <link href="/style.css" rel="stylesheet" />
7    </head>
8    <body>
9      Ma ligne à commenter|
10   </body>
11 </html>
12
```

Aller sur Scratch

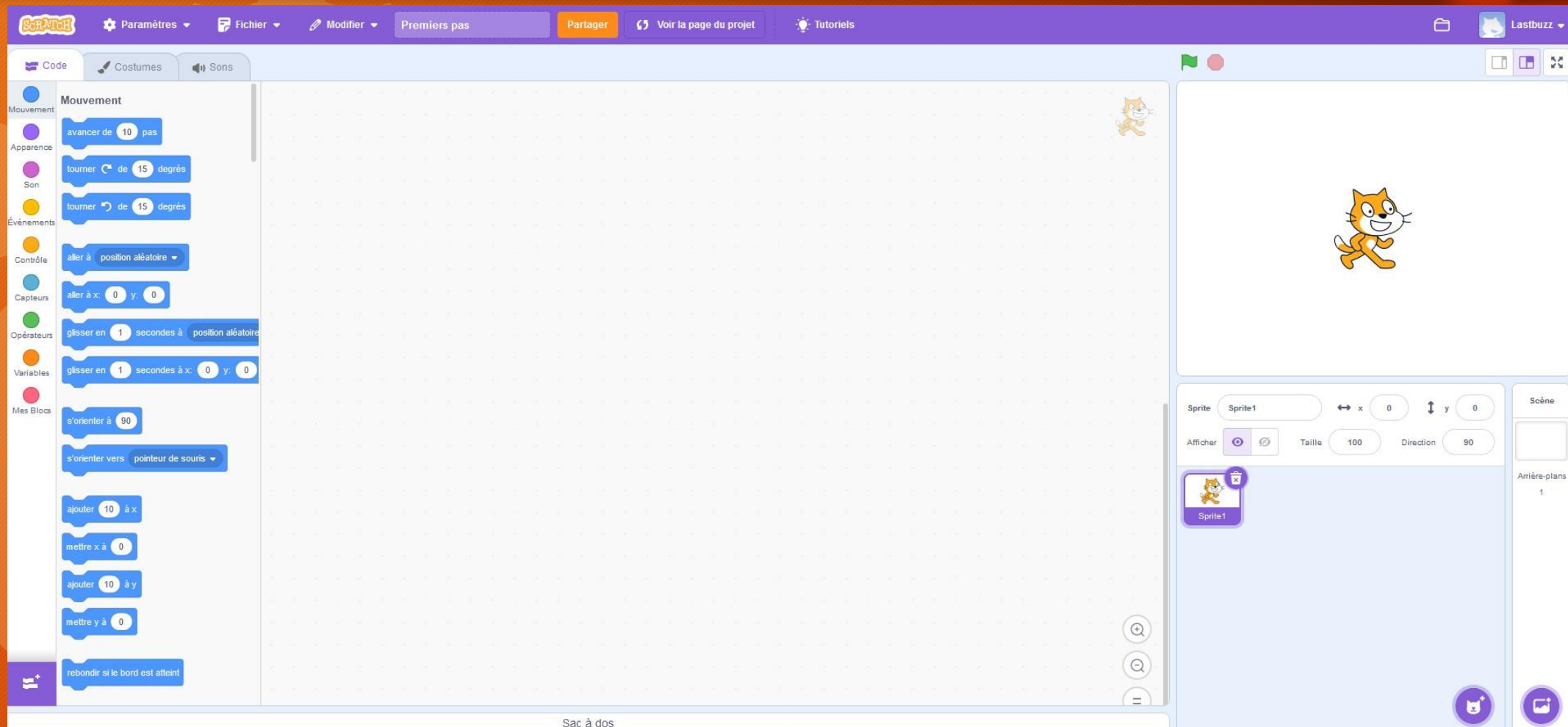


<https://scratch.mit.edu/>

Puis créez-vous un compte



# Votre premier écran de projet Scratch



Code

Costumes

Sons

Mouvement

Apparence

Son

Événements

Contrôle

Capteurs

Opérateurs

Variables

Mes Blocs

avancer de 10 pas

tourner de 15 degrés

tourner de 15 degrés

aller à position aléatoire

aller à x: 0 y: 0

glisser en 1 secondes à position aléatoire

glisser en 1 secondes à x: 0 y: 0

s'orienter à 90

s'orienter vers pointeur de souris

ajouter 10 à x

mettre x à 0

ajouter 10 à y

mettre y à 0

rebondir si le bord est atteint





Sprite

Sprite1

x 0

y 0

Afficher

Taille 100

Direction 90

 Sprite1

Scène

Arrière-plans

1

Sac à dos

# La zone rouge : les blocs d'instructions



Ils sont classés en catégories : mouvements, apparence, événements, contrôle...

Pour chaque catégorie, ils sont présentés dans la colonne de droite.

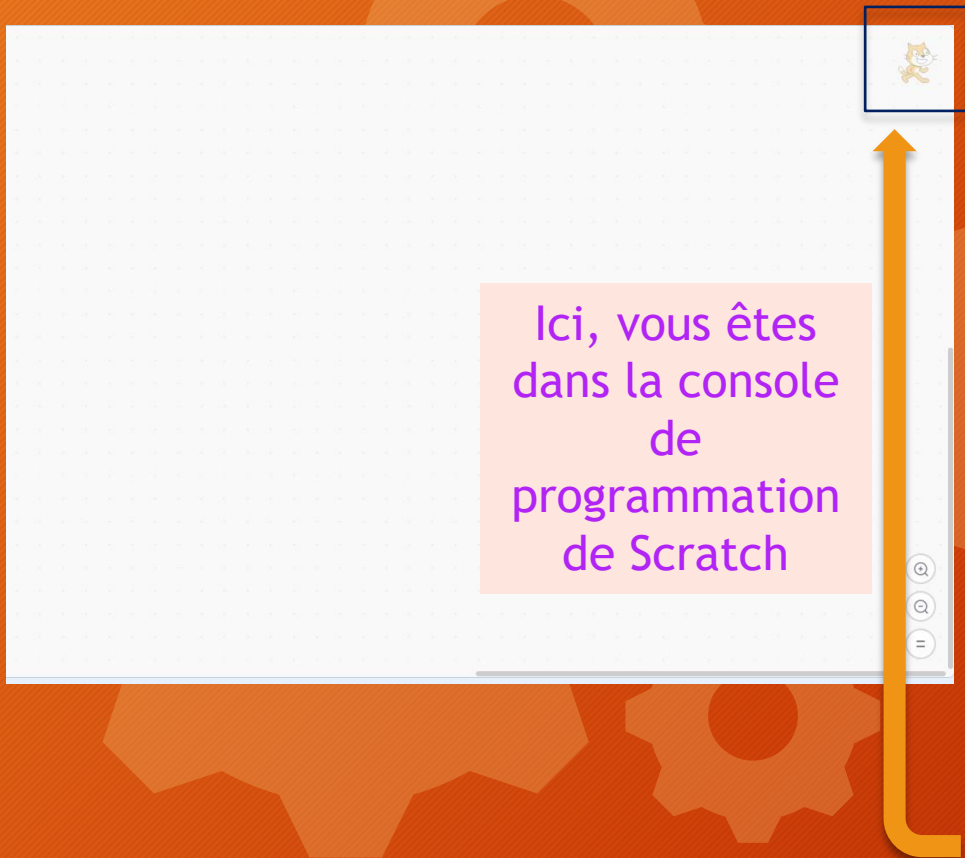
Chaque bloc est écrit « en langue humaine », pas dans un langage de programmation dont il faudrait maîtriser la syntaxe et les conventions. Il correspond à une instruction.

Certains paramètres des blocs d'instruction sont modifiables

Le principe de fonctionnement est : on glisse et on dépose les blocs qu'on souhaite utiliser dans la console de programmation (zone orange), puis on les emboîte, comme un puzzle, pour créer une suite d'instructions = un code



# La zone orange : la console de programmation



C'est la grande zone vide (au début) située au ventre de votre écran.

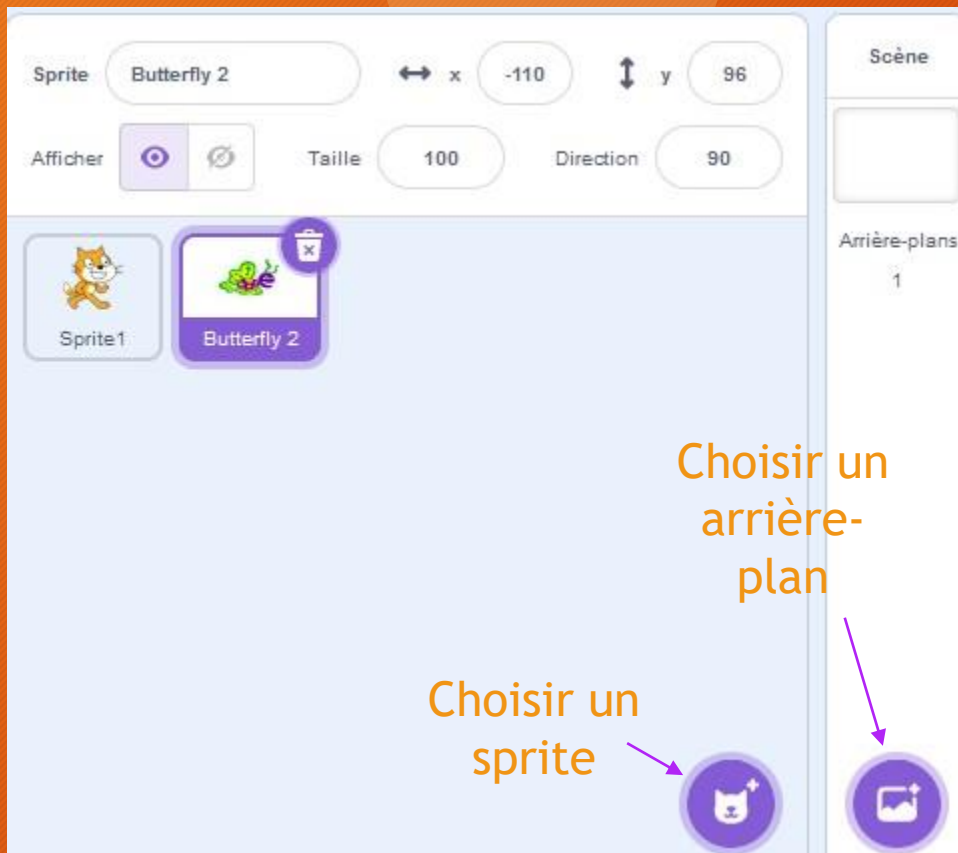
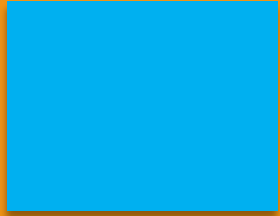
C'est dans cette zone qu'on va faire glisser et emboîter les blocs d'instruction, pour créer du code.

Quand vous avez fait glisser un bloc que vous ne vouliez pas dans cette zone, il suffit de le sélectionner (clic dessus), puis « suppr »

Chaque sprite que vous allez utiliser aura sa propre console de programmation pour donner des instructions. Vous basculerez de la console d'un sprite à celle d'un autre en cliquant sur le sprite que vous désirez dans la **zone bleue**.

Une petite icône en transparence en haut à droite de la console vous indique à quel sprite elle est rattachée.

# La zone bleue : la gestion des sprites et des arrière-plans



Par défaut, il n'y a qu'un sprite de Scratch dans votre projet.

C'est ici que vous allez pouvoir ajouter des sprites et des arrière-plans à votre projet.

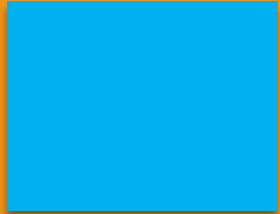
C'est ici que vous allez pouvoir ajuster leur taille, leur orientation initiale, les rendre invisibles (temporairement) le temps que vous travaillez sur un autre sprite.

Quand vous supprimez un sprite (la poubelle), tout le code saisi pour lui disparaît aussi.

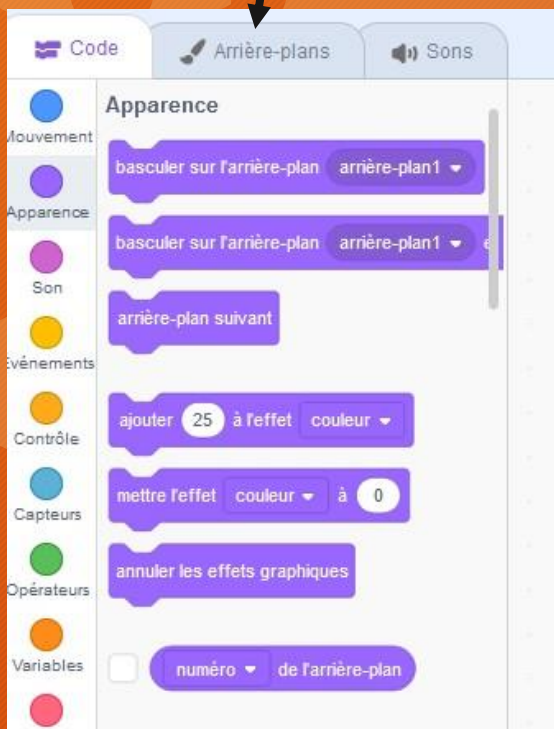
En cliquant sur un sprite, vous accédez à sa console de programmation



# La zone bleue : la gestion des sprites et des arrière-plans



C'est ici



Nota : il y a toujours un arrière-plan blanc par défaut.

Quand vous ajoutez un arrière-plan, il remplace celui présent par défaut, lequel reste néanmoins stocké dans votre répertoire d'arrière-plans.

Si vous souhaitez supprimer définitivement un arrière-plan de votre projet, il faut cliquer sur la colonne « scène » dans la zone bleue, puis dans l'onglet « arrière-plan » (au-dessus de la zone rouge)



# La zone verte : la visualisation de votre projet et de l'exécution de votre code



Dans cette zone apparaissent tous vos sprites (si vous ne les invisibilisez pas) et l'arrière-plan retenu. Par défaut, il n'y a que le sprite de Scratch et un arrière-plan blanc.

Quand vous exécuterez votre code, c'est ici que l'ensemble des instructions saisies dans la console de programmation vont se manifester... Si vous les avez bien saisies 😊

Notez qu'il est possible de manipuler les sprites dans cette zone (glisser/déposer), indépendamment de toute instruction, si vous souhaitez agencer votre fenêtre pour plus de clarté.

# La zone rose : exécuter le code/arrêter l'exécution du code



Ce sont deux boutons qui servent à lancer l'exécution du code (drapeau vert) et à l'interrompre (bouton rouge).



Attention : l'exécution ne va se faire que si l'instruction de lancement du code avec le drapeau vert a été spécifiée dans la console de programmation

Si votre code prévoit des itérations illimitées d'une instruction (boucle), le bouton rouge est le seul moyen d'arrêter le programme.

# Principe de manipulation de Scratch



Je sélectionne des blocs d'instructions dans la galerie de blocs



Je glisse et dépose ces blocs dans ma console de programmation



J'assemble mes blocs d'instructions pour réaliser des suites d'instructions

Dans une suite d'instructions, Scratch va toujours lire les éléments du haut vers le bas



J'exécute mes instructions (drapeau vert)



Je vérifie sur la scène ce qui fonctionne ou pas dans l'exécution de mon code

Et je répète cette suite d'actions pour tous les sprites (et éventuellement les arrière-plans) qui composent mon projet



# À vous !!!



Créez un nouveau projet dans Scratch. Titrez-le

Intégrez un deuxième et un troisième sprite dans votre projet, en plus de Scratch. Réglez la taille de chacun à 50. Invisibilisez Scratch

Intégrez deux arrière-plans supplémentaires (en plus du fond blanc) dans votre répertoire.

Supprimez l'un des deux arrière-plans de votre répertoire.

# Tadam !!!



Scratch interface showing a project titled "Sac à dos" (Backpack). The code editor displays a script for a dragon sprite, including movement and orientation blocks.

**Code Editor:**

- Mouvement (Movement):**
  - avancer de 10 pas (move forward 10 steps)
  - tourner de 15 degrés (turn 15 degrees)
  - tourner de 15 degrés (turn 15 degrees)
  - aller à position aléatoire (go to random position)
  - aller à x: 27 y: 49 (go to x: 27, y: 49)
  - glisser en 1 secondes à position aléatoire (slide in 1 seconds to random position)
  - glisser en 1 secondes à x: 27 y: 49 (slide in 1 seconds to x: 27, y: 49)
  - s'orienter à 90 (set direction to 90)
  - s'orienter vers pointeur de souris (set direction to mouse pointer)
  - ajouter 10 à x (increase x by 10)
  - mettre x à 27 (set x to 27)
  - ajouter 10 à y (increase y by 10)
  - mettre y à 49 (set y to 49)
  - rebondir si le bord est atteint (bounce if edge reached)

**Stage Area:**

- Background: A forest scene with a path and a castle in the distance.
- Sprite: A green dragon.
- Other sprites: A small butterfly.

**Sprite Properties:**

- Sprite: Dragon
- Afficher: Visible
- Taille: 100
- Direction: 90

**Stage Area:**

- Scène: 3



# Le référentiel de la scène : origine, abscisses, ordonnées





# Le référentiel de la scène : taille de la scène



# À vous !!! Familiarisez-vous avec les mouvements



Ouvrez un nouveau projet. Titrez-le. Gardez-le tel quel (Sprite Scratch, arrière-plan blanc).

Essayez de faire avancer Scratch de 50 pas (50 px) vers la droite à l'aide d'un bloc « mouvements ».

Les blocs « mouvements » sont en **bleu clair** dans la liste.

... Normalement, ça ne marche pas. Pourquoi ?

# Initialisation d'une série d'instructions



Si ça ne fonctionne pas, c'est parce que dans votre code, à aucun moment vous n'indiquez quand et à quelle condition débiter l'exécution.

Toute série d'instructions doit débiter par une commande de démarrage, qui donne le « départ » de l'exécution.

Dans Scratch, les commandes de démarrage sont regroupées dans la section « événements » (pastille jaune). Observez les blocs de cette section : on ne peut rien emboîter au-dessus, on ne peut emboîter qu'en dessous.

Toute série d'instructions doit commencer par une commande de ce type.



# À vous !!! Familiarisez-vous avec les mouvements



Faites avancer Scratch de 50 pas avec la commande d'initiation « quand on clique sur le drapeau vert ».

Maintenant faites avancer Scratch de 50 pas, puis demandez-lui d'aller à l'origine (x=0, y=0).



Que se passe-t-il ?

L'exécution du code est quasi-instantanée, la première instruction (avancer de 50 pas) est invisible à l'œil, la seconde ne se voit que parce que la position d'arrivée est différente de la position de départ.

# À vous !!! Familiarisez-vous avec les mouvements



Il va falloir « ralentir » peu l'exécution du code pour pouvoir visualiser les mouvements.

Pour cela, on va utiliser les blocs « glisser en x secondes ».

Scratch va d'abord glisser de 50 pas vers la droite, puis glisser vers l'origine.

Si vous souhaitez ralentir encore le mouvement, agissez sur le paramètre du nombre de secondes dans le bloc.



# À vous !!! Familiarisez-vous avec les mouvements



Si vous n'aimez pas les calculs avec les nombres positifs et négatifs, on peut procéder autrement.

On va marquer un temps de pause entre les deux mouvements, pour bien les dissocier

On va utiliser un bloc de la section « contrôles » (pastille orange), le bloc « attendre x secondes »

On fait un premier déplacement, on attend, on fait le second déplacement. Le mouvement est matérialisé.





# À vous !!! Familiarisez-vous avec les mouvements



Mais on peut encore faire autrement !!!

On peut décomposer le premier déplacement de 50 pas en 5 déplacements de 10 pas, puis aller vers l'origine.

On va utiliser un autre bloc de la section « contrôle », le bloc « répéter x fois ». En langage informatique, ce bloc est une **boucle**.

On se place au point de départ, on répète 5 fois « avancer de 10 pas », puis on file ou on glisse vers l'origine. Le mouvement est matérialisé.



# Qu'avons-nous constaté ?



Un objectif, 3 solutions, et il en existait bien plus...

C'est exactement la même chose lorsqu'on travaille sur une formation. Il n'y a jamais une manière unique d'atteindre un objectif pédagogique.

Il y a toujours des choix à faire. Ils dépendent de vous, de vos apprenants, des moyens à votre disposition.

# Petit test pour voir si vous avez saisi l'idée



Ouvrez un nouveau projet, ou effacez tout le codage de Scratch dans le projet en cours.

Programmez les déplacements de Scratch pour qu'il monte un escalier de **10 marches**.

Hauteur de chaque marche : **20 pas/20px**

Longueur de chaque marche : **30 pas/30px**



# Suggestions de correction



Méthode optimisée temps-  
efficacité



Méthode « bootcamp » :  
vite et à l'essentiel



Méthode  
« doucement  
mais sûrement »

# Compliquons un peu...



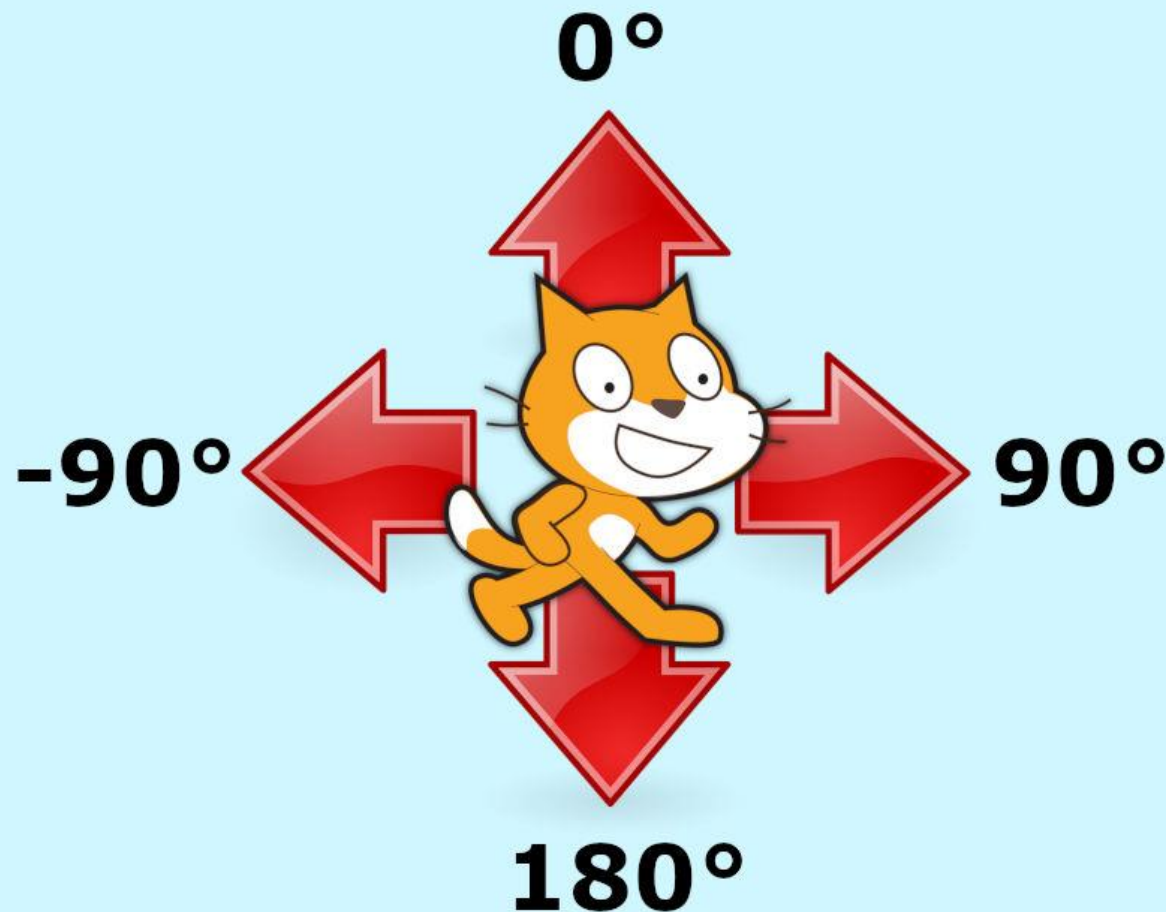
Je veux maintenant que mon Scratch monte son escalier en regardant toujours dans la direction le long de laquelle il avance :

- Quand il avance vers la droite, il regarde vers la droite
- Quand il monte la marche, il regarde vers le haut

Il va falloir le faire pivoter à chaque changement de direction.

On va utiliser pour ce faire les blocs « tourner » et/ou les blocs « s'orienter », dans la section « mouvements ».

# Comment fonctionne l'orientation dans Scratch



Si vous voulez que Scratch regarde dans la direction de la flèche rouge, orientez-le dans le bon angle.



# À vous : faites-lui grimper une marche en l'orientant toujours dans le bon sens



Notez que dans ce cas, quand Scratch regarde vers le haut, « avancer de x pas » revient à monter. Scratch avance toujours dans la direction vers laquelle il regarde.

Donc on ne s'encombre plus de considérations sur les abscisses (x) et les ordonnées (y). On avance et c'est tout.

C'est la notion de « position relative » : pour vous il monte, pour lui il avance.

# Et maintenant un escalier complet, toujours dans le bon sens



# Qu'avons-nous constaté ?



Nous avons atteint un premier objectif : monter un escalier. C'était un objectif intermédiaire.

Avec de nouveaux savoirs (nouveaux blocs), et une nouvelle notion, nous avons atteint un objectif plus complexe : monter un escalier en changeant de position.

Félicitations ! Vous venez de découvrir deux notions fondamentales à respecter lorsqu'on construit une formation ou qu'on prépare l'animation d'une séance ou d'une séquence : le **séquençage** (on découpe notre séquence ou notre séance en sous-unités avec leur propre objectif pédagogique intermédiaire), et la **progression pédagogique** (on complexifie d'une sous-unité pédagogique à une autre, d'un objectif intermédiaire vers un autre, jusqu'à l'objectif pédagogique poursuivi)



# Initiation aux contrôles



Nous avons vu jusqu'à maintenant comment coder un déplacement ou une série de déplacements d'un sprite.

Nous allons désormais vouloir déplacer notre sprite « à la demande », c'est-à-dire en programmant des contrôles :

D'abord un contrôle à la souris ou au pavé tactile ;  
Ensuite un contrôle en associant des touches à certains déplacements.



# À vous !!! Contrôlons Scratch (1/7)



Ouvrez un nouveau projet. Trouvez une façon de contrôler Scratch au moyen de la souris ou du pavé tactile.

Indice : le contrôle influe sur les mouvements, c'est donc avant tout dans ces blocs qu'il pourrait être bon de chercher.

Et si je veux en plus orienter Scratch vers mon pointeur ?



# À vous !!! Contrôlons Scratch (2/7)



Je veux maintenant pouvoir contrôler Scratch au clavier :

Une touche pour aller vers le bas ;

Une touche pour aller vers le haut ;

Une touche pour aller à droite ;

Une touche pour aller à gauche.

Commençons en maintenant notre commande initiale « quand le drapeau vert est pressé ». Comment faire ?

Regardez vos blocs « contrôle ». Et considérez le bloc de condition « Si... Alors »



# À vous !!! Contrôlons Scratch (3/7)



On va créer une liste de 4 conditions : Par exemple :

- Si la touche directionnelle haut est pressée, alors monter de 10 pas
- Si la touche directionnelle bas est pressée, alors descendre de 10 pas
- Si la touche directionnelle gauche est pressée, alors aller de 10 pas vers la gauche
- Si la touche directionnelle droite est pressée, alors aller de 10 pas vers la droite

Et on va demander à ce que cette liste puisse être exécutée indéfiniment tant que le code est en cours d'exécution → On intègre la liste dans une boucle infinie.

# À vous !!! Contrôlons Scratch (4/7)



Pour que cette boucle de conditions puisse s'exécuter, il va falloir que la pression sur les touches soit captée (détectée). On va donc utiliser des blocs spécifiques pour « détecter » la pression sur les touches : les blocs « capteurs » (en bleu-gris)

Ils servent à repérer l'apparition d'un événement ou à mesurer la grandeur d'une variable, pour ensuite enclencher une réaction en fonction de ce qui est constaté.

# À vous !!! Contrôlons Scratch (5/7)



Le principe d'intégration d'un capteur dans une condition dans Scratch :







OU



# À vous !!! Contrôlons Scratch (7/7)



On peut présenter pour terminer une dernière façon de contrôler Scratch.

Jusqu'à maintenant, nous avons toujours initié l'exécution du code par un clic sur le drapeau vert.

On peut utiliser d'autres événements pour initier la séquence d'instructions, par exemple un événement initial de type « quand telle touche est pressée »

Il est tout à fait possible d'avoir plusieurs événements d'initiation dans un même code, chacun conduisant à exécuter les instructions situées en-dessous.

Le point de vigilance : si vous utilisez plusieurs fois un même événement d'initiation, ne donnez pas d'instructions contradictoires.

quand  est cliqué

aller à x: 0 y: 0

s'orienter à 90

quand la touche  est pressée

s'orienter à 0

avancer de 10 pas

quand la touche  est pressée

s'orienter à 180

avancer de 10 pas

quand la touche  est pressée

s'orienter à 90

avancer de 10 pas

quand la touche  est pressée

s'orienter à -90

avancer de 10 pas

OU


quand  est cliqué

s'orienter à 90

aller à x: 0 y: 0

quand la touche  est pressée

ajouter 10 à y

quand la touche  est pressée

ajouter 10 à x

quand la touche  est pressée

ajouter -10 à y

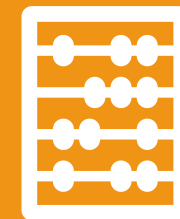
quand la touche  est pressée

ajouter -10 à x





# Projet 1



Sous Scratch V. 3.29

# Mon besoin



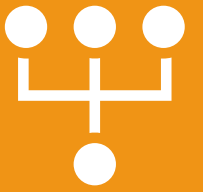
Il me faudrait un jeu dans lequel :

- Je contrôle un personnage
- Qui doit aller toucher des papillons qui apparaissent à droite de l'écran et défilent jusqu'à sa gauche. Un seul papillon à la fois à l'écran, mais ils doivent apparaître et circuler à des hauteurs aléatoires

Je veux un petit décor de fond et un petit nuage qui défile en continu à l'écran pour simuler une progression

Optionnel : je veux un score qui s'incrémente de 1 chaque fois que je touche un papillon avec mon personnage.

# Votre objectif complexe



Construire le plan de réalisation du jeu (code), sous Scratch, en répondant à mon besoin



# De quoi vais-je avoir besoin ?



Un personnage



Un ou des papillons



Un nuage



Un décor

# Comment ces éléments vont-ils se comporter ?



On le contrôle



Il part de la droite, défile vers la gauche, et recommence en changeant de hauteur à chaque fois



Il part de la droite, défile vers la gauche, et recommence



Il est inerte et ne fait rien



# Dans quel ordre vais-je procéder ?

## Suggestion



1



3



2



4