



Introduction

Ce guide vise à développer les compétences de programmation des enseignants et leur capacité à les intégrer efficacement dans leur enseignement. Il adopte une approche progressive et pratique, permettant aux participants d'acquérir à la fois les bases techniques et les méthodes pédagogiques nécessaires.

Les objectifs s'articulent autour de trois axes.

- **Maîtriser les fondamentaux de la programmation par blocs** : Cette approche permet de comprendre les concepts clés de la programmation sans être freiné par la complexité syntaxique des langages traditionnels.
- **Pratiquer avec des outils pédagogiques variés** : Les participants découvrent et expérimentent différents outils d'apprentissage de la programmation ainsi que plusieurs robots éducatifs, ce qui leur permet de développer une compréhension approfondie de leurs possibilités et de leurs limites.
- **Transposer dans sa pratique pédagogique** : Les participants apprennent à concevoir et à adapter des activités de programmation à la fois stimulantes et adaptées au niveau de leurs élèves, tout en s'intégrant naturellement au programme existant.

Cette formation repose sur une méthodologie constructiviste où théorie et pratique sont étroitement liées. Les participants développent leurs connaissances par l'expérimentation directe, la réflexion critique et les échanges entre pairs, préparant ainsi le terrain pour une intégration réussie de la programmation dans leur pratique pédagogique.

Financé par l'Union européenne. Les vues et opinions exprimées n'engagent que leur(s) auteur(s) et ne reflètent pas nécessairement celles de l'Union européenne ou de l'Agence Erasmus+ France / Education Formation. Ni l'Union européenne ni l'autorité chargée de l'octroi ne peuvent en être tenues pour responsables.



Objectifs pédagogiques

- Comprendre les concepts fondamentaux de la programmation par blocs : Acquérir les principes de base – événements, boucles, séquençage.
- Comprendre la logique de programmation : Appréhender la logique et la structure de la programmation par blocs, et savoir comment ces concepts peuvent être appliqués sur différentes plateformes.
- Découvrir la robotique éducative et les outils de programmation : Se familiariser avec divers outils de pensée informatique – plateformes de programmation, cartes de programmation, capteurs, robots.
- Concevoir et mettre en œuvre des activités de codage pratiques : Créer et animer des activités et des projets de codage interactifs utilisant la programmation par blocs, adaptés aux élèves du primaire.
- Identifier les problèmes potentiels d'inclusion dans les pratiques de codage : Reconnaître et traiter les obstacles à l'inclusion qui peuvent survenir dans les activités de codage.

Compétences à développer

Compétence	Description	Pertinence pour l'enseignement
Pensée logique	Compréhension des événements, des boucles et du séquençage dans la programmation par blocs.	Favorise une pensée structurée et logique chez les élèves.
Compétences pratiques en programmation	Expérimentation concrète du codage à travers des activités de programmation.	Renforce les capacités de résolution de problèmes et les connaissances techniques.
Sélection d'outils pour les activités éducatives	Identification et choix des outils de robotique et de programmation les plus adaptés aux différentes activités.	Améliore l'intégration pédagogique de la technologie en classe.
Pensée algorithmique	Élaboration de solutions pas à pas pour résoudre des problèmes.	Développe les compétences en résolution de problèmes méthodique et en planification.
Créativité informatique	Utilisation du codage pour créer des projets innovants.	Encourage la créativité et l'innovation en classe.



Théorie, concepts et outils clés

Aperçu

Ce guide adopte une approche constructiviste, directement inspirée des théories de Seymour Papert. L'apprentissage est structuré pour permettre aux participants de construire activement leurs connaissances par l'expérimentation et la réflexion critique. Le parcours d'apprentissage se déroule en trois phases :

- **Phase 1 – Découverte débranchée** : Le guide débute par une activité sans ordinateur, favorisant l'émergence naturelle des concepts clés de la programmation. Cette approche expérientielle permet aux participants de construire leurs propres représentations tout en découvrant intuitivement les fondamentaux. Un apport théorique complémentaire peut être proposé pour approfondir certains concepts.
- **Phase 2 – Expérimentation avec les outils** : Les participants expérimentent différentes ressources, permettant l'émergence progressive de nouvelles compétences. Cette phase offre un cadre structuré où les enseignants découvrent et manipulent divers outils tout en réfléchissant à leur application pratique en classe. Ils sont encouragés à explorer des projets de différents niveaux en utilisant une variété d'outils : environnements de programmation comme MakeCode, robots éducatifs comme Thymio, dispositifs à base de capteurs comme micro:bit.
- **Phase 3 – Transposition pédagogique** : Cette phase est centrée sur l'application pédagogique des connaissances acquises. Les participants conçoivent des activités adaptées à leur contexte d'enseignement, réfléchissent à l'adaptation de leurs projets au niveau de leurs élèves et élaborent des stratégies pour favoriser l'inclusion dans leurs pratiques. Afin de documenter et de partager leur travail, ils créent une documentation détaillée décrivant leur conception, leur programme et leurs réalisations. Cette approche valorise l'apprentissage et constitue une ressource partageable et réutilisable par d'autres enseignants.



La programmation en classe : l'héritage de Seymour Papert

Dans les années 1960, au MIT, un jeune mathématicien nommé Seymour Papert observait des enfants en train d'apprendre. Il remarqua leur curiosité naturelle et leur capacité à comprendre des concepts complexes lorsqu'ils pouvaient manipuler, construire et explorer par eux-mêmes. Cette observation fit naître une idée : et si les enfants pouvaient apprendre à programmer aussi naturellement qu'ils apprennent une langue ?

C'est ainsi qu'en 1967, alors que les ordinateurs occupaient encore des pièces entières, Papert créa Logo, un langage de programmation destiné aux enfants. Son intuition était simple : plutôt que de considérer l'ordinateur comme une machine pédagogique, il l'imaginait comme un outil que les enfants pourraient programmer eux-mêmes. Cette approche a ouvert la voie aux langages de programmation éducatifs que nous connaissons aujourd'hui, comme Scratch et ScratchJr.

En 1980, Papert publia ses travaux dans un ouvrage intitulé *Mindstorms: Children, Computers, and Powerful Ideas*. Il y décrit comment les enfants apprennent mieux lorsqu'ils construisent leurs propres connaissances, tels de petits chercheurs qui explorent et expérimentent. Cette approche, qu'il nomma constructionnisme, continue d'inspirer les enseignants qui utilisent la programmation comme outil pédagogique.

Dans ce guide, nous adoptons une démarche similaire. Plutôt que de présenter la programmation comme un ensemble de concepts théoriques à mémoriser, nous encourageons les enseignants à créer des environnements d'apprentissage où les élèves peuvent découvrir ces concepts par eux-mêmes, à travers l'expérimentation et la création de projets. Bien que nous présentions les concepts fondamentaux de la programmation (séquences, boucles, conditions), l'objectif n'est pas de les enseigner de manière frontale. Nous suggérons plutôt de laisser les élèves les découvrir naturellement en travaillant sur des projets qui les intéressent — qu'il s'agisse de créer une animation, un jeu ou de programmer un robot.

Explorer et expérimenter la programmation : un premier pas sans ordinateur

Avant de découvrir les concepts fondamentaux de la programmation par blocs, nous vous invitons à participer à une activité débranchée intitulée « Mode Danse ! ». Conçue initialement pour les élèves, elle est particulièrement adaptée à ce module car elle permet une expérimentation directe des concepts de programmation tout en donnant à voir leur future application en classe. Aucune compétence préalable n'est requise et, grâce au mouvement corporel, elle permet aux participants de découvrir intuitivement comment les blocs s'assemblent et interagissent.



L'activité se déroule en groupes : l'un est chargé de concevoir le programme, l'autre de l'exécuter.

Première phase : exécution d'un programme préétabli : Le premier groupe reçoit un ensemble de cartes prédéfinies représentant différents types de blocs de programmation (boucles, conditions, séquences...), inspirées d'outils éducatifs tels que Scratch. Une fois les cartes assemblées de manière logique, le second groupe doit « exécuter » la chorégraphie correspondante. Cette approche kinesthésique permet une compréhension intuitive de l'influence de chaque bloc sur le « programme » final, tout en favorisant la collaboration et les échanges entre pairs.

Deuxième phase : création libre : L'activité se poursuit par une étape de création où les participants reçoivent des blocs vierges qu'ils doivent compléter eux-mêmes. S'appuyant sur la théorie constructiviste, cette phase permet aux apprenants de construire leurs connaissances en vérifiant leur compréhension de chaque type de bloc, tout en illustrant les concepts de modularité et de réutilisation du code. Cette étape renforce la compréhension de la logique de programmation tout en stimulant la créativité pédagogique.

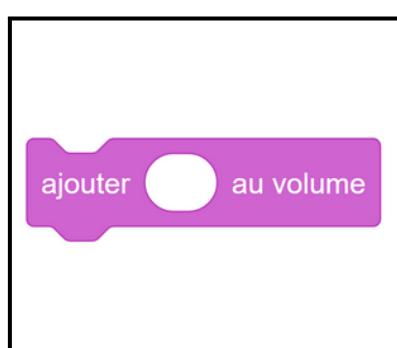
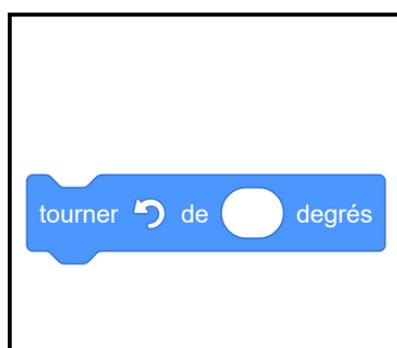
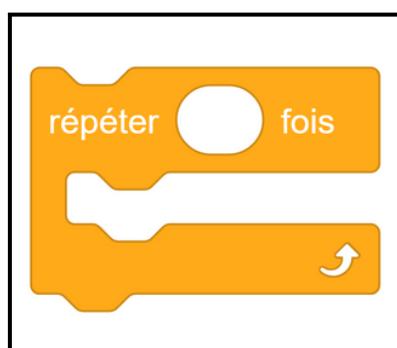
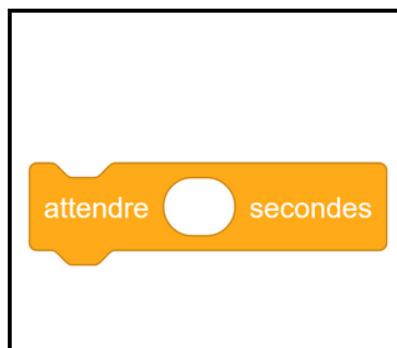
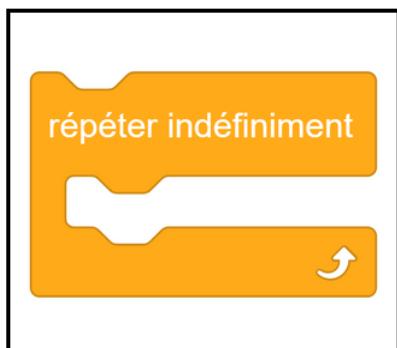
Troisième phase : réflexion collective : L'expérience se conclut par un temps d'échange sur les parallèles entre le mouvement physique et l'exécution du programme, permettant aux enseignants de visualiser comment ces concepts peuvent être adaptés à leur propre classe. Cette approche favorise l'expression émotionnelle et exerce une influence positive sur les processus cognitifs des apprenants : perception, attention, apprentissage, mémoire, raisonnement et résolution de problèmes.

Pourquoi cette activité fonctionne-t-elle ? La force de cette activité réside dans sa capacité à rendre les concepts abstraits concrets : une boucle devient un mouvement répétitif, une condition se transforme en choix de direction, une séquence se manifeste par une série d'étapes. Cette approche interdisciplinaire, alliant technologie et expression corporelle, favorise non seulement une meilleure mémorisation des concepts, mais contribue également au bien-être des apprenants en transformant l'apprentissage en une expérience ludique et bienveillante.

Transposition en classe. Une fois cette activité maîtrisée, les enseignants peuvent facilement la transposer avec leurs élèves. Elle constitue une introduction particulièrement efficace à la programmation, car elle vise à développer plusieurs compétences clés du XXI^e siècle : pensée créative et logique, perception spatiale, résolution de problèmes, structuration et collaboration. En permettant aux élèves d'expérimenter concrètement les concepts de la programmation avant même de toucher un ordinateur, ils développent une compréhension intuitive des principes fondamentaux. L'aspect ludique et collaboratif de l'activité contribue également à réduire l'anxiété que certains élèves peuvent ressentir face à la programmation, créant ainsi un environnement d'apprentissage positif et inclusif.



Exemples de blocs à utiliser dans l'activité débranchée



Cette activité est issue d'une ressource développée dans le cadre du projet Unplugged, un projet européen visant à promouvoir la créativité et la pensée logique chez les jeunes grâce à des activités sans écran. Disponible sous licence Creative Commons, elle peut être librement utilisée, adaptée et partagée par les enseignants pour répondre aux besoins spécifiques de leurs classes.



Traduire l'exploration en concepts clés

Au cœur de la programmation : l'algorithme

Un algorithme est une séquence finie d'instructions précises permettant de résoudre un problème ou d'atteindre un objectif. Par exemple, suivre une recette de cuisine est un algorithme : chaque étape doit être exécutée dans un ordre précis pour obtenir le résultat escompté. En programmation, les algorithmes décrivent les actions qu'un ordinateur ou un robot doit effectuer pour accomplir une tâche donnée. La programmation consiste à traduire un algorithme en un langage compréhensible par la machine. Cette traduction peut prendre différentes formes selon les langages utilisés : langages textuels (comme Python ou Java) ou langages visuels (comme les langages par blocs). Les langages textuels sont précis, mais leur syntaxe stricte peut rendre l'apprentissage complexe pour les débutants. De plus, la diversité des langages – chacun conçu à des fins différentes (calculs scientifiques, applications mobiles, intelligence artificielle) – peut créer de la confusion chez ceux qui abordent la programmation pour la première fois. C'est pourquoi la programmation par blocs s'avère être une approche privilégiée pour les débutants : elle simplifie la syntaxe tout en permettant de se concentrer sur la logique et les concepts fondamentaux, rendant ainsi la programmation accessible et ludique pour tous.

Les blocs : une représentation visuelle du code

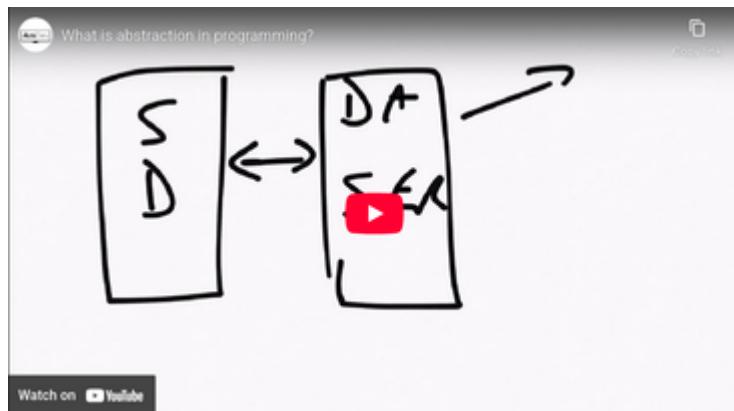
La programmation par blocs est un langage de programmation visuel où les blocs de code représentent différentes commandes ou fonctions, ce qui la rend accessible même aux personnes n'ayant aucune expérience préalable en programmation. Cette méthode simplifie les concepts complexes de la programmation en éléments intuitifs et faciles à manipuler, permettant ainsi aux enseignants de comprendre la logique de la programmation sans être submergés par les détails de la syntaxe.

Historiquement, la programmation par blocs est apparue comme une méthode d'initiation à la programmation, notamment dans un contexte éducatif. Des outils tels que Scratch et MakeCode ont été développés pour rendre la programmation plus accessible et attrayante, en particulier pour les jeunes apprenants. Ces outils permettent aux utilisateurs de créer des programmes par glisser-déposer de blocs, rendant ainsi l'apprentissage interactif et ludique.

Les blocs de programmation sont des éléments visuels qui représentent des instructions de code spécifiques. Cette approche repose sur deux concepts fondamentaux : **l'abstraction et la modularité**.



L'abstraction est le processus fondamental qui réduit la complexité d'un système en ne conservant que les éléments essentiels. En programmation par blocs, cette approche se manifeste par l'utilisation d'éléments visuels qui masquent la complexité du code sous-jacent – à l'image d'un plan de métro qui simplifie un réseau complexe en n'affichant que l'essentiel.

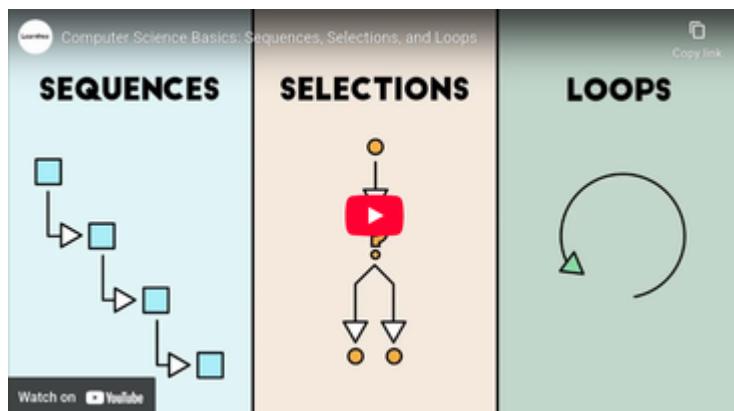


La **modularité**, quant à elle, permet de diviser le code – un système complexe – en composants indépendants et réutilisables, à l'image des briques LEGO® qui peuvent être assemblées de différentes manières. Ces pièces – ici, les blocs – sont autonomes mais peuvent être combinées de multiples façons pour créer des structures complexes.

Ainsi, chaque bloc est conçu pour accomplir une tâche spécifique : déplacer un personnage, émettre un son, effectuer un calcul. Ces blocs s'emboîtent comme les pièces d'un puzzle, créant une séquence logique d'instructions. La forme et la couleur de chaque bloc sont significatives : les blocs d'une même catégorie (mouvement, son, contrôle, etc.) partagent généralement la même couleur, tandis que leur forme indique comment ils peuvent être combinés avec d'autres blocs.

Cette approche visuelle permet une compréhension intuitive de la structure du code avant même d'explorer les concepts de programmation plus abstraits – concepts que vous avez naturellement découverts grâce à l'activité débranchée « Mode Danse ! » :

- **Séquence** : En programmation, la séquence permet d'organiser les actions dans un ordre logique et chronologique. À l'instar d'une recette de cuisine où l'ordre des étapes est crucial pour la réussite du plat, la séquence garantit que chaque instruction est exécutée au bon moment et dans le bon ordre.



Par exemple, on ne peut pas glacer un gâteau avant de l'avoir cuit ! De même, un programme doit suivre une séquence d'actions précise pour fonctionner correctement. Ce concept est essentiel car il permet de structurer les programmes de manière cohérente et prévisible.



- **Boucles :** Les boucles sont des structures de contrôle qui permettent de répéter une série d'instructions. Elles sont essentielles en programmation car elles évitent d'avoir à écrire plusieurs fois les mêmes instructions.



Par exemple, un robot de surveillance devant patrouiller en continu dans un bâtiment utilisera une boucle infinie pour répéter indéfiniment son itinéraire. Son fonctionnement est similaire à celui d'un battement de cœur – excellent exemple de boucle biologique – qui répète continuellement le même cycle de contraction et de relaxation. Ce concept est fondamental car il permet d'automatiser les tâches répétitives et de créer des programmes plus efficaces.

- **Déclencheurs ou événements :** Les déclencheurs (ou événements) sont des mécanismes qui permettent à un système de réagir automatiquement, selon des modalités prédéfinies, à des stimuli externes ou internes. Ils fonctionnent comme un signal qui, lorsqu'il est détecté, déclenche automatiquement l'exécution d'une séquence d'actions. Pour illustrer ce concept, prenons l'exemple d'un thermostat qui active le chauffage lorsque la température descend en dessous d'un seuil défini (par exemple 19 °C) et l'arrête automatiquement lorsqu'elle dépasse un autre seuil (21 °C). Ce système simple mais efficace maintient une température confortable sans intervention humaine constante. Autre exemple naturel : nos réflexes. Lorsque notre main touche une surface très chaude, notre système nerveux déclenche automatiquement un réflexe de retrait, avant même que notre cerveau n'ait analysé consciemment la situation.

- **Conditions :** Les conditions sont des structures qui permettent à un programme de prendre des décisions en fonction de situations spécifiques. Elles fonctionnent comme un aiguillage qui oriente le programme vers différentes actions selon que certaines conditions sont remplies ou non.





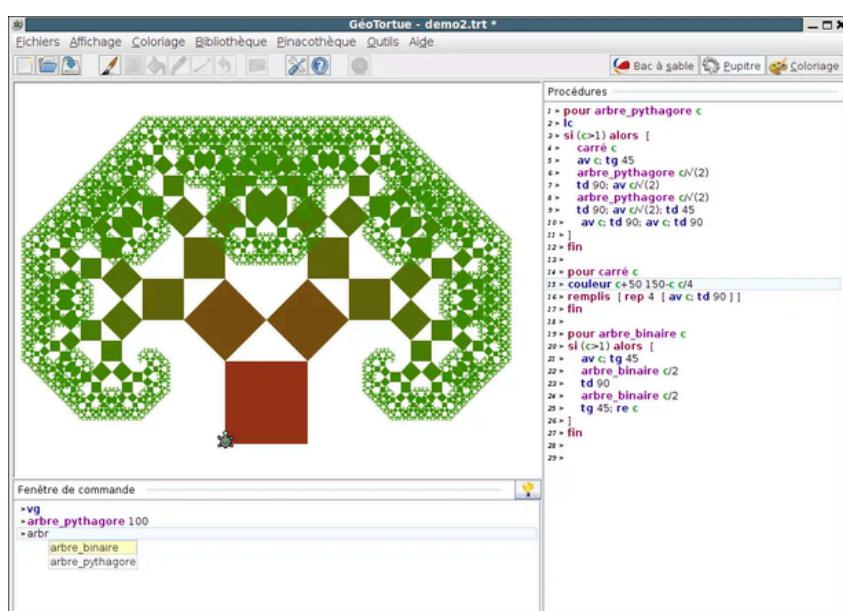
Par exemple, dans la nature, les plantes s'orientent vers la lumière (phototropisme) : si la lumière provient d'un côté, alors la plante pousse dans cette direction. En programmation, ce principe permet de créer des programmes adaptatifs qui réagissent différemment selon les situations, comme un thermostat qui active le chauffage si la température descend en dessous d'un certain seuil.

Approche constructiviste de la programmation : explorer le codage à travers les objets programmables

À l'instar de l'exploration débranchée de la programmation par blocs, la robotique éducative peut servir de support complémentaire pour expérimenter le fonctionnement d'un programme informatique. Les robots au sol, en particulier, rendent l'intangible tangible en offrant une compréhension visuelle et concrète des commandes de programmation.

Cette approche s'inscrit dans la lignée des travaux de Seymour Papert qui, dans les années 1960, a développé le langage LOGO et sa célèbre « tortue » programmable – précurseur des robots éducatifs modernes. La tortue LOGO permettait déjà aux enfants d'explorer la programmation de manière concrète et visuelle, illustrant parfaitement la théorie constructiviste de Papert selon laquelle l'apprentissage est plus efficace lorsque les élèves construisent activement leur compréhension par le biais d'expériences tangibles.

Le principe de base de LOGO est simple : à l'écran, une petite tortue se contrôle à l'aide de commandes élémentaires. Elle peut avancer ou reculer d'un nombre défini de pas, ou pivoter à gauche ou à droite d'un angle précis (en degrés). La tortue trace une ligne continue lorsqu'elle se déplace, permettant ainsi aux enfants de réaliser des dessins en lui donnant les bonnes instructions.





Cette approche est particulièrement intéressante car elle encourage naturellement la curiosité intellectuelle de l'élève et enrichit sa perception des figures géométriques. De plus, les concepts mathématiques émergent en réponse à des besoins concrets lorsque les élèves réalisent leurs dessins. Par exemple, la notion de variable devient rapidement un outil essentiel pour créer efficacement des motifs complexes.

À l'instar de la tortue LOGO, les robots physiques permettent une approche concrète et expérimentale de la programmation. Le Thymio, par exemple, permet aux élèves de visualiser directement les résultats de leurs instructions par le biais de dessins et de mouvements.

Quelques exemples :

- Le robot **Thymio** peut illustrer le concept de boucle en dessinant des cercles répétitifs.
- Le **Blue-Bot**, avec sa forme d'abeille et son interface simple, est idéal pour enseigner l'abstraction et les séquences de base.
- Le **LEGO WeDo** excelle dans l'enseignement de la modularité grâce à sa nature constructive, permettant aux élèves de comprendre la réutilisation des composants.

Cette approche rejoint celle de LOGO en ce qu'elle permet une expérimentation directe et visuelle des concepts de programmation et encourage naturellement une démarche d'investigation chez l'élève. Les concepts mathématiques et de programmation émergent naturellement des besoins concrets lors de la réalisation des projets.

Une diversité de moyens, pas une obligation d'équipement

Bien que chaque solution robotique offre des avantages pédagogiques uniques, il est important de souligner qu'aucune n'est parfaite ni indispensable. Le coût parfois élevé de ces robots peut constituer un obstacle majeur pour de nombreux établissements scolaires.

Heureusement, l'apprentissage de la programmation ne requiert pas forcément l'utilisation de robots physiques. Des alternatives gratuites ou peu coûteuses — simulateurs en ligne (Scratch, Code.org) ou activités débranchées — peuvent tout aussi efficacement développer la pensée informatique et la logique de programmation. L'essentiel est de privilégier une approche pédagogique diversifiée et exploratoire, combinant différentes méthodes d'apprentissage.



Bibliographie

BBC Bitesize. Séquençage dans les algorithmes - Séquençage - Révision d'informatique pour le cycle 3. BBC. <https://www.bbc.co.uk/bitesize/guides/zsf8d2p/revision/1>

CS Unplugged. <https://www.csunplugged.org/>

Papert, S. (1983). Jaillissement de l'esprit : ordinateurs et apprentissage. Revue française de pédagogie, 62(1), 94–95. https://www.persee.fr/doc/rfp_0556-7807_1983_num_62_1_2294_t1_0094_0000_2

Stöckel, A., et al. (2024). Un environnement de programmation par blocs pour l'enseignement du calcul bas niveau (Document de discussion). ResearchGate. https://www.researchgate.net/publication/378013283_A_Block-Based_Programming_Environment_for_Teaching_Low-Level_Computing_Discussion_Paper

Fondation Raspberry Pi. (30 juillet 2019). De l'inspiration à l'innovation : blocs de programmation pratiques. Raspberry Pi. <https://www.raspberrypi.org/blog/hands-on-coding-marcos-navas-picademy/>

Algo.be. Logo : Introduction à la programmation Logo. <https://www.algo.be/logo1/logo-primer-fr.html>

IREM Paris Nord. GéoTortue – Programmer en Logo. https://www-irem.univ-paris13.fr/site_spip/spip.php?rubrique1

Robogenie. La programmation informatique : code et langage. <https://robogenie.fr/avantages-apprentissage-robotique/programmation-informatique-code-langage/>

IREM Paris Nord. LOGO, ordinateurs et apprentissages. https://www-irem.univ-paris13.fr/site_spip/spip.php?article32

Ultimate Coders. (14 septembre 2022). Programmation par blocs pour enfants. <https://www.ultimatecoders.ca/blog/block-coding-for-kids>

Robogenie. (2021, December 7). L'héritage de Seymour Papert : 4 dates clés dans la vie d'un visionnaire. <https://robogenie.fr/heritage-seymour-papert-4-dates-cles/>



Bibliographie

Contributeurs de Wikipédia. (22 août 2024). Seymour Papert. Dans Wikipédia, l'encyclopédie gratuite https://en.wikipedia.org/wiki/Seymour_Papert

HAL. (2018). L'algorithme débranchée : enseigner l'informatique sans ordinateur. <https://hal.science/hal-01716631/document>

Common Sense Education. (s.d.). Les meilleurs outils de programmation par blocs pour débutants. <https://www.commonsense.org/education/best-in-class/the-best-block-based-coding-tools-for-beginners>



Activités et conseils pour explorer la pensée informatique

Dans sa phase pratique, ce guide explore comment les enseignants peuvent adopter de manière autonome une approche constructiviste de leur apprentissage, en établissant une relation positive avec la pensée informatique. Par une construction progressive, les enseignants-apprenants découvrent par l'action et l'expérimentation comment ces outils peuvent donner du sens à l'apprentissage.

Cette session ne propose donc pas de solutions toutes faites, mais des conseils, des outils et des ressources permettant de manipuler, d'expérimenter et de s'approprier l'approche informatique. Grâce à ces découvertes, les enseignants développent leurs compétences, nouent une relation positive avec la programmation et consolident les bases de leur enseignement futur.

Étape 1 – Expérimenter l'approche débranchée

- **Objectif de l'étape : Démystifier la programmation ; développer une compréhension intuitive des algorithmes, des séquences, des boucles et des conditions ; développer la pensée logique et la décomposition des problèmes ; maîtriser le vocabulaire de base de la programmation par le biais d'une expérience ludique.**
- **Transposition pédagogique : Identifier les stratégies pertinentes de résolution de problèmes ; reconnaître les moments de blocage et développer des stratégies pour les surmonter ; développer de l'empathie pour les difficultés d'apprentissage des élèves.**

L'approche déconnectée de l'apprentissage de la programmation repose en grande partie sur le constructivisme. À travers des défis et des jeux, et grâce à des règles simples, les apprenants peuvent découvrir par eux-mêmes des concepts complexes en résolvant des problèmes.

Activité 1 – Autoformation : Jouez, jouez, jouez !

Les activités débranchées ont une dimension fondamentalement ludique. Apprendre à utiliser la pensée informatique par le jeu permet d'explorer un sujet potentiellement nouveau pour les apprenants, tout en réduisant le sentiment de risque. Ils peuvent ainsi se concentrer sur le contenu, encadré par des règles explicites et claires.

La documentation comprend de nombreuses ressources pertinentes et facilement accessibles dans lesquelles les enseignants peuvent puiser pour créer leur propre parcours d'autoformation, avec lequel ils se sentent à l'aise.



La ressource la plus connue et reconnue est CS Unplugged (csunplugged.org), qui vise à promouvoir l'informatique auprès des jeunes. Les apprenants y trouveront un ensemble d'activités gratuites sans ordinateur, conçues pour sensibiliser et populariser la pensée informatique. Le principal atout de cette ressource réside dans la variété des activités proposées ainsi que dans son approche multilingue, ce qui la rend accessible à un large public.



Nous vous encourageons à explorer les différentes activités, à les imprimer et à les tester en groupe, entre amis ou avec vos collègues. Organisez votre propre soirée jeux dédiée aux activités débranchées ! L'essentiel : s'amuser à découvrir de nouvelles ressources, sans risque ni contrainte ! En partageant ces ressources de manière collaborative et ludique, vous pourrez observer les réactions de vos pairs et adopter une approche positive de cet apprentissage. Vous pouvez également explorer les jeux débranchés de l'univers Unplugged Quest : comptage binaire, grille de parité, Tours de Hanoï... un ensemble de jeux et d'activités débranchées contextualisés pour répondre aux enjeux de la citoyenneté et du développement durable. → [Découvrir les jeux Unplugged Quest - https://github.com/unplugged-quest/games](https://github.com/unplugged-quest/games)

Activité 2 – Formation en présentiel : Explorer les robots à travers des séances artistiques

Dans le cadre du projet « Robots Meet Arts », quatre fiches de séance introduisent l'approche débranchée en classe, en lien avec les sciences humaines. Ces quatre activités peuvent être utilisées lors d'ateliers en présentiel pour explorer la pensée informatique par le jeu.

Attention : Nous déconseillons d'utiliser ces ressources dans le cadre d'une autoformation. Les fiches « Robots Meet Arts » ont été conçues comme l'aboutissement de la phase de formation. Elles intègrent des approches pédagogiques plus complexes et supposent une connaissance préalable des démarches proposées (débranchée, sur ordinateur ou robotique). Bien que leur exploration puisse offrir une vision plus complète de l'impact pédagogique et de l'intérêt de la pensée informatique lors de la phase de formation guidée, elles ne conviennent pas à une introduction autonome à l'approche débranchée sans connaissances préalables.



« Le Jeu du Mode Danse ! »

Cette activité, qui sollicite l'expression corporelle, vise à traduire le langage informatique en mouvements et à découvrir l'univers de Scratch. Elle permet d'expérimenter comment les concepts de programmation peuvent s'incarner par le mouvement et d'apprendre à créer un cadre d'apprentissage progressif, de la compréhension kinesthésique à la programmation numérique. Elle met également en lumière les multiples représentations possibles d'un même algorithme et aide à comprendre comment rendre concrets et attrayants des concepts de programmation abstraits.

« Code Collage Craze »

Inspirée par l'art de Matisse, cette activité développe la perception spatiale et la capacité à donner des instructions étape par étape de manière logique. Elle aide les élèves à comprendre l'importance de fournir des instructions claires et précises, tout en établissant un lien direct avec la rigueur requise en programmation et en algorithmique. En décomposant une tâche artistique complexe en étapes simples, elle illustre la relation entre créativité et logique algorithmique et montre comment la programmation peut être intégrée aux arts visuels.

« Robot Chef : Mission Petit-déjeuner ! »

Ancrée dans le quotidien, cette activité encourage la pensée logique et la pratique des algorithmes sans ordinateur. Elle met en lumière l'importance de l'enchaînement des actions pour atteindre un objectif, tout en soulignant le rôle crucial d'instructions claires et précises. Les apprenants expérimentent la traduction d'actions quotidiennes en algorithmes et acquièrent une compréhension concrète de l'importance de l'ordre des instructions. Cette activité révèle les difficultés d'une communication précise, développe l'empathie envers les difficultés rencontrées par les débutants et aide à identifier les éventuels obstacles à l'apprentissage algorithmique.

« La quête du Curry »

Sous forme de récit interactif, cette activité introduit les instructions conditionnelles en programmation. Elle invite les participants à transposer des processus de décision analogiques en instructions conditionnelles numériques et à modéliser des récits complexes sous forme d'algorithmes. Elle démontre comment les approches débranchées peuvent servir de tremplin à l'apprentissage numérique, tout en illustrant la puissance des structures conditionnelles par le biais de la narration. Elle aide les élèves à comprendre comment les choix et leurs conséquences peuvent être modélisés algorithmiquement, développe la visualisation de flux logiques complexes et établit des liens étroits entre narration et programmation.



Étape 2 – Du débranché au branché : premiers pas dans le codage

- **Objectif de l'étape :** Transférer les concepts découverts vers les environnements de programmation visuelle ; comprendre que la même logique s'applique dans différents contextes.
- **Transposition pédagogique :** Expérimenter l'étaillage progressif comme stratégie d'enseignement ; identifier les points de rupture potentiels dans la transition ; développer des stratégies pour accompagner ce passage délicat.

À travers le contenu et les activités proposés dans ce guide, il apparaît clairement que la pensée informatique va bien au-delà de la simple programmation. Bien qu'elle repose principalement sur les défis de la résolution créative de problèmes et du raisonnement logique, il est intéressant de la relier à la programmation, ce qui permet de concrétiser et d'expliciter les apprentissages acquis grâce à l'approche débranchée. En codant, l'apprenant transmet à l'ordinateur sa compréhension du fonctionnement des concepts, ce qui revient à valider ou à approfondir ses connaissances.

Activité 1 – Reconnecter les activités débranchées

Les apprenants peuvent consulter les guides « Plugging it in » proposés par CS Unplugged (<https://www.csunplugged.org/en/plugging-it-in/>). Ils y trouveront plusieurs propositions guidées et variées pour prolonger des activités débranchées sur des outils tels que Scratch ou Blockly. Cela leur permettra de valider leurs acquis de la phase débranchée et de découvrir progressivement des outils pertinents pour leurs propres élèves.

Scratch : un outil de référence

Développé par le groupe de recherche Lifelong Kindergarten (MIT Media Lab), Scratch est un outil de programmation par blocs permettant de créer des histoires interactives, des compositions musicales, des simulations numériques et bien plus encore. Son interface ludique, colorée et simple en fait un outil idéal pour l'apprentissage de la programmation à l'école primaire. Il est fréquemment utilisé (mais pas exclusivement) dans les séances proposées par le programme « Robots Meet Arts ».

Scratch offre une première approche simple de la programmation. Cependant, il est nécessaire de découvrir d'autres écosystèmes – notamment pour les enseignants dont les classes se situent à la charnière entre le primaire et le début du secondaire – afin de comprendre la pluralité des outils disponibles pour l'enseignement de la programmation et de proposer une approche diversifiée.



Activité 2 – S'amuser avec la programmation : utiliser des outils créatifs

L'un des aspects de la programmation souvent négligé dans l'imaginaire collectif est son potentiel créatif. Au sein du programme « Robots Meet Arts », nous visons à maximiser ce lien émotionnel entre la pensée informatique et la création.

Activité 2.1 – Programmation avec MakeCode Arcade

MakeCode Arcade (arcade.makecode.com) propose un environnement de programmation par blocs spécialement conçu pour la création de jeux vidéo rétro. Sa particularité réside dans sa capacité à rendre immédiatement visibles les résultats du code : chaque bloc ajouté transforme instantanément le jeu, offrant un retour gratifiant qui maintient l'intérêt de l'apprenant. L'outil intègre harmonieusement les concepts de boucles, de conditions et de variables dans des contextes ludiques et pertinents.

Pour débuter sur MakeCode Arcade, les apprenants peuvent explorer les Skill Maps (cartes de compétences). Ces cartes guident pas à pas l'apprentissage de la programmation à travers la création de jeux. Elles décomposent les concepts en petits niveaux progressifs, permettant aux élèves de suivre leurs progrès et de découvrir les bases — boucles, variables, conditions — tout en explorant la conception, la narration et les règles du jeu.

Comment commencer :

1. Ouvrez MakeCode Arcade et repérez la section « Beginner Skillmaps » sous le bouton « Nouveau projet ».
2. Choisissez une carte adaptée, comme « Full of Stories », idéale pour découvrir les blocs de construction les plus utilisés.
3. Chaque carte de compétences est composée de niveaux intermédiaires clairs et stimulants. En cliquant sur une carte, vous découvrirez ses objectifs d'apprentissage, le nombre d'étapes et les compétences visées (programmation, narration, effets visuels ou sonores).

Chaque étape est accompagnée d'un tutoriel interactif expliquant quels blocs utiliser et leur fonction (événements, boucles, etc.). Les élèves apprennent par la pratique et peuvent ensuite créer leurs propres projets. La progression est ludique : des récompenses et même un certificat de réussite sont délivrés à la fin du parcours, ce qui renforce la motivation.

Enfin, si une carte d'introduction comme « Full of Stories » vous semble trop simple, vous pouvez explorer les cartes de compétences de niveau supérieur proposées sur l'écran d'accueil. Les créations peuvent être partagées avec la communauté, les enseignants ou les parents, favorisant ainsi la collaboration et la créativité.



Activité 2.2 – Donner vie à ses idées avec OctoStudio

OctoStudio, développé par l'équipe de Scratch, pousse la créativité encore plus loin en permettant la création d'histoires interactives, d'animations artistiques et de simulations. Son interface intuitive encourage l'expérimentation libre tout en introduisant progressivement des concepts plus avancés tels que la gestion d'événements, la synchronisation et les interactions complexes.

OctoStudio offre une expérience mobile optimisée pour la programmation créative. Si des logiciels comme Scratch sont bien connus, OctoStudio se distingue par plusieurs aspects importants :

- **Ressources d'apprentissage complètes** : Démarrez rapidement grâce à une riche bibliothèque de tutoriels et de démonstrations vidéo disponibles sur la chaîne YouTube d'OctoStudio.
- **Conçu pour le mobile, pensé pour la flexibilité** : OctoStudio est spécialement conçu pour les appareils mobiles, permettant de créer des projets à tout moment et n'importe où, sans ordinateur. Donnez vie à vos projets grâce à des fonctionnalités interactives qui réagissent au toucher, aux secousses, à l'inclinaison et aux autres capteurs de votre appareil. Intégrez facilement des photos, des vidéos et des sons de votre environnement à vos projets, et stimulez votre créativité au-delà de l'écran.
- **Programmation par blocs intuitive et gratuite** : Grâce à son interface conviviale de type « glisser-déposer », OctoStudio permet aux enfants de 7 à 14 ans d'apprendre facilement les concepts de la programmation sans se soucier de la syntaxe. Son utilisation est gratuite et, de l'avis de nombreux utilisateurs, il est plus intuitif que Scratch.
- **Fonctionne sans Internet** : OctoStudio fonctionne hors ligne, ce qui en fait un outil idéal pour les zones où l'accès à Internet est limité ou inexistant.
- **Multilingue et accessible** : Disponible dans plus de 20 langues et doté de fonctionnalités d'accessibilité (compatibilité avec les lecteurs d'écran, agrandissement des images), OctoStudio est véritablement inclusif pour tous les utilisateurs.

L'objectif de cette exploration n'est pas de maîtriser parfaitement ces outils, mais de découvrir par soi-même comment la programmation peut devenir un moyen d'expression personnelle. Nous encourageons les apprenants à se fixer des défis créatifs simples : « Et si je créais un petit jeu sur mon passe-temps ? », « Comment pourrais-je raconter une histoire de mon cours par l'animation ? » Cette approche par projet personnel génère un engagement authentique et une motivation intrinsèque difficiles à reproduire avec des exercices imposés.



Activité 3 – Explorer librement les logiciels d'apprentissage de la programmation

Cette phase d'exploration libre constitue une étape supplémentaire pour les enseignants-apprenants souhaitant approfondir leur découverte des environnements de programmation. Le principal défi n'est pas de multiplier les compétences techniques, mais de trouver l'environnement dans lequel chaque enseignant se sent à l'aise et en confiance. Les préférences d'apprentissage étant propres à chacun, certains se sentiront plus à l'aise avec des interfaces visuelles colorées, d'autres préféreront des environnements plus épurés, et d'autres encore seront attirés par des approches plus textuelles. La diversité des outils disponibles peut parfois engendrer une forme de paralysie décisionnelle. Il est important de se rappeler que l'outil n'est qu'un moyen au service de l'apprentissage de la pensée informatique. Un enseignant qui maîtrise parfaitement les concepts peut utiliser efficacement n'importe quel environnement, tandis qu'une multiplication des outils sans compréhension approfondie peut créer de la confusion.

Nous vous encourageons à une démarche exploratoire : testez chaque outil pendant 20 à 30 minutes, notez vos impressions et identifiez ce qui vous plaît et ce qui vous déplaît. Cette approche comparative vous permettra de mieux comprendre les spécificités de chaque environnement et de déterminer ceux qui pourraient convenir à vos futurs projets pédagogiques. N'hésitez pas à revenir plusieurs fois au même outil si une première exploration ne vous convainc pas.

Scratch

Scratch (<https://scratch.mit.edu/>) est un environnement de programmation visuelle créé par le MIT et destiné aux apprenants de 8 à 16 ans. Il permet de créer des contenus interactifs tels que des histoires, des animations et des jeux. La plateforme est gratuite, mais nécessite une période d'apprentissage initiale pour se familiariser avec son interface et son fonctionnement. Scratch encourage la créativité tout en initiant aux concepts clés de la pensée informatique.



- **Chaîne YouTube officielle de Scratch Team** → [Accéder à la chaîne](#)
- **Playlist de tutoriels Scratch rapides et ludiques (Griftpatch)** → [Voir la playlist](#)
- **Coding for Kids – Tutoriels d'initiation à Scratch** → [Voir la playlist](#)
- **Projets Scratch inspirants (DenshiVideo)** → [Voir la vidéo](#)



MakeCode et MakeCode Arcade

MakeCode (<https://makecode.microbit.org/>) est un environnement de programmation par blocs développé par Microsoft pour les apprenants de 8 à 16 ans. Il propose des extensions spécifiques, notamment pour la programmation des robots micro:bit (voir : <https://makecode.microbit.org/pkg/microsoft/microbit-robot>). Gratuit, il nécessite une brève prise en main, mais permet de programmer pour micro:bit, Arcade (<http://arcade.makecode.com/>) et Minecraft, ce qui en fait un outil polyvalent pour les projets créatifs.



- **Tutoriels officiels MakeCode → Accéder aux tutoriels**
- **Introduction à MakeCode Arcade → Accéder aux tutoriels**
- **Projet « Cœur clignotant » → Voir le projet**
- **Projet « Smiley Buttons » → Voir le projet**
- **Projet « Pierre, feuille, ciseaux » → Voir le projet**
- **Tutoriels d'introduction à MakeCode Arcade (YouTube) → Voir la playlist**
- **Première carte de compétences – « Full of Stories » (Arcade) → Accéder à la Skillmap**
- **« Bubble Pop » (Arcade) → Voir le tutoriel**
- **« Maze Game » (Arcade) → Voir le tutoriel**

Tynker

Tynker (<https://www.tynker.com/>) est une plateforme de programmation créative conçue pour les apprenants de 5 à 18 ans. Elle propose des projets progressifs alliant créativité et programmation. L'accès à la plateforme nécessite un abonnement et une connexion internet. Le logiciel est disponible à l'achat ou par abonnement mensuel (à partir de 312 € par an). Tynker est reconnu pour son approche ludique qui combine jeu, narration et programmation.



- **Premiers pas avec Tynker (tutoriel vidéo) → Voir la vidéo**
- **Chaîne YouTube officielle de Tynker → Accéder à la chaîne**
- **Tutoriels Tynker pour débutants → Voir la playlist**



Code.org

Code.org (<https://code.org/en-US>) est une plateforme en ligne progressive d'apprentissage de la programmation, conçue pour les apprenants de 4 à 18 ans. Elle propose des activités guidées et des tutoriels structurés adaptés aux différents groupes d'âge. La plateforme est gratuite, mais nécessite une connexion internet stable.



- Portail élèves Code.org pour le primaire → [Accéder au portail](#)
- Chaîne YouTube officielle de Code.org → [Accéder à la chaîne](#)
- Exemple de projet : Comment créer son propre jeu avec Sprite Lab → [Voir la vidéo](#)
- Exemple de projet : « Attrape le papillon » avec Game Lab → [Voir la vidéo](#)

Kodable

Kodable (<https://www.kodable.com/>) est une plateforme de programmation ludique destinée aux enfants de 4 à 10 ans. Elle utilise des activités structurées et ludiques pour enseigner les bases de la programmation. Bien qu'un abonnement soit requis, une version gratuite est disponible pour les enseignants. Kodable allie un gameplay amusant à une progression structurée, ce qui la rend accessible aux plus jeunes.



- Chaîne YouTube officielle de Kodable → [Accéder à la chaîne](#)
- Playlist d'activités et de tutoriels Kodable → [Voir la playlist](#)
- Comment débuter avec Kodable (guide de l'enseignant) → [Voir la vidéo](#)



codeSpark Academy

codeSpark Academy (<https://codespark.com/play/>) est une plateforme d'apprentissage du code sans texte, conçue pour les enfants de 5 à 9 ans. Son approche visuelle et interactive unique permet même aux enfants qui ne savent pas encore lire d'apprendre les concepts de la programmation. La plateforme nécessite un abonnement et une connexion internet ; l'achat du logiciel ou un abonnement mensuel coûte environ 190 €. Elle privilégie un apprentissage intuitif et ludique.



- **Présentation de codeSpark Academy – Comment utiliser « Create A Foo »** → [Voir la vidéo](#)
- **Initier les enfants au codage avec « The Foos »** → [Voir la vidéo](#)

LearnToMod

LearnToMod (<https://www.learntomod.com/>) est une plateforme d'apprentissage du code axée sur Minecraft, destinée aux jeunes de 8 à 16 ans. Elle enseigne les concepts de la programmation à travers la création de mods personnalisés pour Minecraft. La plateforme est payante ; les tarifs sont disponibles sur demande pour les établissements scolaires. Bien que limitée à l'environnement Minecraft, elle offre une expérience d'apprentissage très interactive et captivante.



- **Chaîne YouTube officielle de LearnToMod** → [Accéder à la chaîne](#)
- **Apprendre aux enfants à programmer avec LearnToMod (Henrik Kniberg)** → [Voir la vidéo](#)



Étape 3 – Découvrir le monde de la robotique éducative sans pression

- **Objectif de l'étape :** Découvrir le monde de la robotique éducative ; comprendre les possibilités d'intégration en classe ; identifier les outils qui correspondent à vos besoins et contraintes.
- **Transposition pédagogique :** Développer une vision réaliste des investissements nécessaires ; savoir poser les bonnes questions aux experts ; constituer son propre répertoire d'outils robotiques adaptés à son contexte d'enseignement.

L'exploration de la robotique éducative représente un prolongement naturel de la démarche constructiviste : après avoir expérimenté des concepts de manière débranchée puis numérique, la découverte de la manière dont ces apprentissages peuvent s'incarner dans des objets physiques ouvre de nouvelles perspectives éducatives.

Cependant, la robotique éducative implique souvent des investissements financiers importants. Plutôt que de se précipiter sur un achat, cette séance pratique propose une approche exploratoire qui vous permet de découvrir l'écosystème robotique sans engagement financier, vous préparant ainsi efficacement aux échanges avec des experts lors de formations en présentiel.

Activité 1 – Exploration documentaire : comprendre l'offre en robotique grâce aux ressources en ligne

Ces dernières années, l'intégration d'outils robotiques dans le contexte éducatif s'est considérablement développée. Les plateformes de streaming proposent une multitude de démonstrations illustrant ces outils en action, offrant ainsi aux enseignants la possibilité d'explorer leurs fonctionnalités. En observant comment les séquences pédagogiques sont conçues à l'aide de ces outils, les enseignants peuvent mieux appréhender leur potentiel pédagogique et comprendre les contraintes liées à leur utilisation. De plus, lorsque des simulateurs ou des environnements similaires sont disponibles, ils permettent de se faire une idée concrète du fonctionnement de ces robots dans un contexte réel, enrichissant ainsi la compréhension de leurs capacités et de leurs limites.

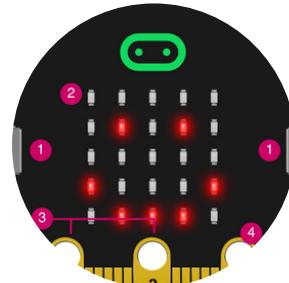
Cette exploration préliminaire permet d'identifier les outils qui correspondent à vos objectifs pédagogiques.

Voici une liste non exhaustive d'outils susceptibles de répondre aux besoins des enseignants du primaire. Chaque contexte pédagogique présente ses spécificités : budget disponible, niveau des élèves, infrastructure technique et temps de préparation. Explorer ces options permet d'anticiper ces aspects pratiques en observant comment d'autres enseignants intègrent ces outils dans leurs pratiques.



Micro:bit

Micro:bit (<https://microbit.org/fr/>) est un microcontrôleur programmable destiné aux apprenants à partir de 8 ans. Il prend en charge la programmation par blocs, JavaScript et Python. Compatible avec de nombreux composants électroniques, il est idéal pour les projets scientifiques, technologiques, d'ingénierie, artistiques et mathématiques (STEAM). Son principal inconvénient réside dans l'investissement initial et la maintenance. Les prix démarrent à 18,60 € pour la carte seule, 21 € pour un kit avec câble et pile, et 177 € pour un lot de 10 kits.



- **Simulateur intégré MakeCode (tester les blocs, le code JavaScript et Python en ligne)**
→ [Accéder au simulateur](#)
- Circuits Tinkercad avec micro:bit → [Explorer Tinkercad](#)
- Playlist officielle des projets micro:bit → [Voir la playlist](#)
- Chaîne éducative micro:bit → [Accéder à la chaîne](#)
- Exemple de projet : « Réactions micro:bit » → [Voir la vidéo](#)
- Exemple de projet : « Dé micro:bit » → [Voir la vidéo](#)
- Exemple de projet : « Compteur de pas micro:bit » → [Voir la vidéo](#)

LEGO® Education SPIKE™

Le coffret LEGO® Education SPIKE™ est destiné aux enfants de 7 ans et plus. Il combine la construction LEGO et la programmation par blocs pour créer des robots, des dispositifs dynamiques et des modèles interactifs. Malgré sa grande polyvalence, il nécessite un investissement et un entretien importants, pour un prix d'environ 500 €. → [Découvrir SPIKE™](#)



- Tutoriel officiel avec simulateur de construction → [Accéder au tutoriel](#)
- Chaîne officielle LEGO Education → [Accéder à la chaîne](#)
- Playlist des tutoriels SPIKE Prime → [Voir la playlist](#)
- Playlist d'activités de construction et de programmation → [Voir la playlist](#)



Bee-Bot

Bee-Bot est un robot programmable simple, conçu pour les enfants à partir de 4 ans. Il utilise des boutons physiques pour des activités de programmation de base et fonctionne parfaitement avec des tapis de jeu thématiques. Bien que ses fonctionnalités soient limitées, il reste abordable (généralement entre 70 et 90 €).



- **Émulateur en ligne Bee-Bot (Terrapin)** → [Accéder à l'émulateur](#)
- **Projets Bee-Bot pour les écoles** → [Voir la vidéo](#)
- **Activités Bee-Bot en classe** → [Voir la vidéo](#)
- **Jeux éducatifs Bee-Bot** → [Voir la vidéo](#)

Cubetto

Cubetto (<https://primotoys.com/>) est conçu pour les très jeunes enfants (dès 3 ans) et initie à la programmation grâce à des blocs physiques sur un plateau. Sans écran et accessible, il est cependant assez coûteux (219,60 €) et peu adapté aux élèves plus âgés. Les tapis ou packs de blocs supplémentaires coûtent environ 30 €.



- **Salle de jeux Cubetto – Activités interactives pour explorer de nouvelles façons de jouer et d'apprendre avec Cubetto** → [Accéder à la Playroom](#)
- **Chaîne officielle de Primo Toys** → [Accéder à la chaîne](#)
- **Cubetto en action** → [Voir la vidéo](#)
- **Démonstration Cubetto en classe** → [Voir la vidéo](#)

Alternative numérique : Vous pouvez également utiliser LightBot (<https://lightbot.lu/>). Ce jeu de puzzle en ligne enseigne des concepts de programmation tels que les séquences, les boucles et les procédures. Bien qu'il ne s'agisse pas d'une simulation directe de Cubetto, il offre un équivalent numérique indirect à la programmation concrète de Cubetto. Les apprenants déplacent un robot en enchaînant des commandes, renforçant ainsi le raisonnement logique introduit par les blocs physiques de Cubetto.



Thymio

Thymio (<https://www.thymio.org/>) est un robot programmable polyvalent destiné aux apprenants de 6 à 14 ans. Compatible avec plusieurs langages de programmation, il est équipé de capteurs. Son prix est d'environ 157,20 €, et des kits de 4, 6 ou 10 robots sont disponibles. Il nécessite un investissement initial et de la maintenance, mais son utilisation s'adapte facilement à la progression de l'apprenant.



- **Simulateur Thymio en ligne → [Accéder au simulateur](#)**
- **Vidéo d'introduction à Thymio → [Voir la vidéo](#)**
- **Activités en classe avec Thymio → [Voir la vidéo](#)**
- **Programmation de Thymio avec VPL et Scratch → [Voir la vidéo](#)**
- **Démonstration de Thymio dans les écoles → [Voir la vidéo](#)**
- **Chaîne officielle Thymio → [Accéder à la chaîne](#)**

Photon

Photon (<https://photon.education/>) est un robot interactif doté de 10 capteurs et proposant des activités thématiques, conçu pour les enfants dès 5 ans. Relativement coûteux, son prix de base avoisine les 250 € ; il est disponible en kit. Ses atouts résident dans son adaptabilité et la variété de ses activités.



- **Chaîne officielle de Photon Education → [Accéder à la chaîne](#)**
- **Présentation de Photon Classroom → [Voir la vidéo](#)**
- **Projets scolaires Photon → [Voir la vidéo](#)**



Sphero Indi

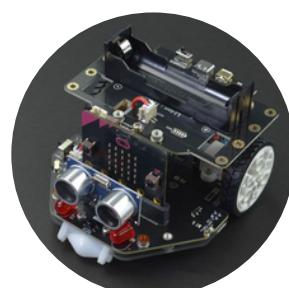
Sphero Indi (<https://sphero.com/pages/sphero-indi>) est un robot destiné aux enfants de 4 ans et plus, programmable sans écran grâce à des cartes de couleurs. Il privilégie la logique et l'exploration, mais son prix est élevé (environ 150 €), des kits plus complets étant disponibles pour les classes.



- **Chaîne officielle Sphero → [Accéder à la chaîne](#)**
- **Playlist « Indi Classroom » → [Voir la playlist](#)**
- **Vidéo de démonstration d'Indi → [Voir la vidéo](#)**

DFRobot Maqueen V2+

Le **DFRobot Maqueen V2+** (<https://www.dfrobot.com/search-maqueen-plus-v2.html>) est un robot éducatif abordable, compatible avec micro:bit et équipé de capteurs, destiné aux enfants de 8 ans et plus. Son prix est d'environ 50 € pour le robot seul et de 130 € pour un kit comprenant une carte micro:bit, une caméra HUSKYLENS et les câbles. La carte micro:bit est vendue séparément.



- **Tutoriel Maqueen V2+ → [Voir la vidéo](#)**
- **Prise en main du Maqueen V2+ → [Voir la vidéo](#)**
- **Exemple d'utilisation en classe → [Voir la vidéo](#)**
- **Démonstration Maqueen avec HUSKYLENS → [Voir la vidéo](#)**

Alternative numérique : Vous pouvez également utiliser les extensions MakeCode pour micro:bit afin de simuler la programmation du Maqueen (<https://makecode.microbit.org/>). Bien que les capteurs et les moteurs du robot ne puissent pas être reproduits physiquement dans le navigateur, le simulateur permet aux apprenants de tester et de déboguer leurs programmes (commandes de mouvement, contrôle des LED, logique de prise de décision) avant de les transférer sur le véritable Maqueen. Cela offre un environnement sécurisé pour s'exercer à la programmation.



ElecFreaks Cutebot

Cutebot (<https://shop.elecfreaks.com/products/elecfreaks-micro-bit-smart-cutebot-kit-without-micro-bit-board>) est un robot abordable compatible avec micro:bit, conçu pour les enfants de 8 ans et plus. Il est doté de la fonction de suivi de ligne, de capteurs ultrasoniques et de LED RGB. Prix : 29,90 € (robot seul) ou 54,80 € (avec carte micro:bit).



- **Démonstration Cutebot → [Voir la vidéo](#)**
- **Présentation de Cutebot en classe → [Voir la vidéo](#)**
- **Chaîne officielle Elecfreaks → [Accéder à la chaîne](#)**

Alternative numérique : Vous pouvez également utiliser les extensions MakeCode pour micro:bit afin de simuler la programmation de Cutebot (<https://makecode.microbit.org/>). Ce simulateur permet aux apprenants d'expérimenter les commandes de mouvement, la logique des capteurs et les sorties LED dans un environnement virtuel. Bien qu'il ne puisse pas émuler entièrement les capteurs physiques de Cutebot, il constitue un outil précieux pour tester les algorithmes avant de les appliquer au robot réel.

KittenBot TabbyBot

TabbyBot (<https://www.kittenbot.cc/products/kittenbot-tabbybot-programming-robotics-kit>) est un robot éducatif compatible avec micro:bit, destiné aux apprenants de 8 ans et plus. Il permet de construire différents modèles et offre des possibilités d'extension, pour une grande flexibilité dans la réalisation de projets. La documentation disponible en dehors de l'anglais est toutefois limitée. Prix : 59,90 €, carte micro:bit vendue séparément.



- **Exemple de projet TabbyBot → [Voir la vidéo](#)**
- **Chaîne officielle de KittenBot → [Accéder à la chaîne](#)**
- **Utilisation de TabbyBot en classe → [Voir la vidéo](#)**

Alternative numérique : Vous pouvez également utiliser les extensions MakeCode pour micro:bit afin de simuler la programmation de TabbyBot (<https://makecode.microbit.org/>). Ce simulateur permet aux apprenants de s'exercer à coder des structures (boucles, conditions et événements) qui contrôlent TabbyBot. Bien qu'il ne puisse pas reproduire le fonctionnement physique du robot ni ses extensions, il permet de tester la logique et de déboguer le code avant de l'appliquer au matériel.



Activité 2 – Préparer ses questions pour les séances en présentiel

En explorant les différents outils présentés ci-dessus, nous recommandons aux apprenants de dresser une liste de questions, de remarques et d'idées d'application à approfondir lors des phases de formation en présentiel. Les échanges avec les formateurs n'en seront que plus productifs, en se concentrant sur les aspects pratiques et les retours d'expérience plutôt que sur des observations générales.

**Si vous ne participez pas aux phases de formation en présentiel du consortium « Robots Meet Arts », n'hésitez pas à nous envoyer vos questions par courriel à :
contact@labaixbidouille.com.**



L'inclusion appliquée à cette exploration

Introduction aux défis de l'inclusion

L'intégration de la robotique et de la programmation éducatives dans l'enseignement pose d'importants défis en matière d'inclusion.

Accessibilité technologique

L'accessibilité technologique constitue un enjeu majeur, notamment concernant l'utilisation des interfaces matérielles traditionnelles. La manipulation des souris, des claviers et des composants électroniques exige une motricité fine qui peut s'avérer problématique pour certains élèves. Par ailleurs, les écrans standards ne répondent pas toujours aux besoins des élèves malvoyants.

Accessibilité logicielle

Du point de vue logiciel, la complexité inhérente aux interfaces de programmation par blocs peut constituer un obstacle majeur. Les élèves présentant des troubles de l'attention peuvent être submergés par la densité d'informations visuelles, tandis que la navigation dans les menus et la manipulation des blocs requièrent une coordination œil-main qui n'est pas universellement accessible. De plus, les retours visuels et sonores des applications ne sont pas systématiquement adaptés aux différents profils d'apprentissage. Barrières linguistiques, culturelles et socio-économiques

Outre ces défis techniques, il existe des barrières linguistiques et culturelles, ainsi que des disparités socio-économiques qui peuvent restreindre l'accès aux ressources nécessaires. Pour développer une éducation inclusive, il est utile d'adopter une approche qui tienne compte de la diversité de ces défis. Cela implique de considérer les aspects techniques de l'accessibilité tout en prêtant attention aux dimensions sociales et culturelles de l'apprentissage.



Liste de pratiques faciles à mettre en œuvre pour réduire les problèmes d'inclusion

Voici une liste de questions fondamentales à prendre en compte lors de la conception d'activités technologiques inclusives en classe. Vous pouvez utiliser cette liste de contrôle pour vous assurer que vos activités utilisant des outils technologiques tiennent compte de ces points.

Avez-vous réfléchi aux difficultés que pourraient rencontrer les élèves ayant des besoins particuliers pour accéder aux technologies ? Avez-vous réfléchi aux difficultés qu'ils pourraient avoir à comprendre l'objectif et ce qui est attendu d'eux lors d'activités utilisant des outils numériques ou robotiques ?

- Envisager des interfaces offrant des options de contraste élevé et des polices sans empattement pour les élèves malvoyants.
- Fournir des outils ou des plateformes compatibles avec les lecteurs d'écran et les afficheurs braille numériques.
- Proposer des solutions d'audiodescription pour les interfaces graphiques complexes et le retour visuel des robots.
- Utiliser des diagrammes et des maquettes en relief (par exemple, des impressions 3D) pour rendre les concepts visuels accessibles aux élèves aveugles.
- Mettre en place des outils permettant la personnalisation des palettes de couleurs dans les environnements de programmation afin d'inclure les élèves daltoniens.
- Proposer des robots ou des interfaces à commandes tactiles ou vocales pour pallier les limitations de la motricité fine.

Avez-vous réfléchi à la manière dont les groupes sous-représentés (filles, minorités culturelles, élèves issus de milieux défavorisés) pourraient se sentir exclus des activités technologiques ? Avez-vous envisagé des moyens de rendre ces activités culturellement pertinentes et accessibles à tous les élèves ?

- Autoriser les élèves à intégrer des éléments culturels spécifiques à leurs projets : motifs, récits ou personnages issus de leur propre patrimoine.
- Fournir des environnements et des interfaces visuelles utilisant des symboles clairs et universels pour surmonter les barrières linguistiques.
- Garantir une représentation culturelle et linguistique dans les exemples et les contextes proposés pour les projets.



Avez-vous pris des mesures pour améliorer vos activités afin qu'elles soient conformes aux principes de la Conception Universelle de l'Apprentissage (CUA) ?

- Concevoir des activités de robotique permettant de multiples options d'interaction : tactile, gestuelle, vocale ou via des claviers adaptatifs.
 - Fournir des options visuelles et auditives pour chaque retour d'information produit par les outils (par exemple, un signal sonore associé à un message visuel pour les erreurs de programmation).
 - Proposer des plans de travail et des fiches simplifiés avec des étapes numérotées et illustrées, intégrant des supports en gros caractères.
 - Mettre à disposition des tutoriels interactifs adaptés aux élèves malvoyants ou présentant des troubles de l'attention, avec des fonctionnalités telles que la pause et la relecture étape par étape.
-

Avez-vous pris en compte les implications pratiques de l'accès aux ressources pour les élèves issus de milieux défavorisés ou ne disposant pas d'équipement personnel ?

- Créer des kits de robotique en utilisant des matériaux accessibles, peu coûteux ou recyclables, et fournir des instructions pour un assemblage facile.
- Prévoir des environnements de programmation accessibles sur des appareils basiques tels que les smartphones ou tablettes de génération précédente.
- Fournir des licences gratuites pour les logiciels nécessaires aux activités et organiser des ateliers communautaires pour partager le matériel.
- Mettre à disposition des élèves n'ayant pas accès régulièrement à un ordinateur ou à Internet des fiches d'activités imprimables.



Bibliographie

BBC MakeCode. Accessibilité. <https://makecode.com/accessibility>

Hernandez, A. (18 novembre 2021). Apprendre à coder sans ouvrir les yeux. freeCodeCamp. <https://www.freecodecamp.org/news/a-vision-of-coding/>

Code Jumper. Code Jumper. <https://codejumper.com/>

Franklin, D., Penuel, W. et Lee, S. (mars 2024). Accessible à qui ? Rendre les blocs accessibles. Dans : Actes du 55e symposium technique de l'ACM sur l'enseignement de l'informatique (SIGCSE '24), vol. 1 (p. 127-133). ACM. <https://dl.acm.org/doi/10.1145/3626252.3630770>

Fondation éducative Micro:bit. (3 décembre 2021). micro:bit LIVE 2021 | Accès et équité dans MS MakeCode/micro:bit [Vidéo]. YouTube. <https://youtu.be/DmAWpH8zI7Y>

Réseau Canopé. Outils numériques de lecture adaptés au handicap visuel. <https://www.reseau-canope.fr/agence-des-usages/outils-numeriques-de-lecture-adaptes-au-handicap-visuel.html>

Fondation nationale pour la science. (2021). Dépôt des résultats de recherche. <https://par.nsf.gov/servlets/purl/10302001>

School Outlet. (11 janvier 2022). Concevoir une salle de classe adaptée aux daltoniens : conseils et stratégies pour un apprentissage inclusif. https://www.schooloutlet.com/blogs/news/designing-a-color-blind-friendly-classroom-tips-and-strategies-for-inclusive-learning?srsltid=AfmBOoxGJX7r1kA_TdLmfMJ1hxj2nbmJh2WewL9YtQ9P_E7r3k_gsmP

Letssteam. (s.d.). Ressources pour l'accès et l'équité. GitHub. https://github.com/letssteam/Resources/tree/main/4_AS_Equity/EN



Validation des acquis d'apprentissage

Pour valider les acquis d'apprentissage, nous vous invitons à créer une activité débranchée, transposable en programmation par blocs selon la logique « du débranché au branché ».

L'objectif est de concevoir une activité pratique et stimulante qui présente d'abord les concepts de la programmation sans ordinateur, puis montre comment ils s'articulent avec la programmation numérique grâce à une approche par blocs. Cette double approche permet aux élèves d'appréhender concrètement les concepts fondamentaux de la programmation avant d'être confrontés à un environnement numérique, rendant ainsi leur apprentissage plus accessible et plus pertinent.

Les enseignants devront élaborer leur activité en utilisant le modèle fourni afin de permettre la reproduction et la transférabilité des résultats de leur apprentissage.

Idées de projets

- **Créer une activité de danse ou de mouvement sans appareil électronique, qui pourra ensuite être programmée à l'aide de blocs MakeCode ou Scratch.**
- **Concevoir un jeu physique d'association mots-émotions pour des textes littéraires, qui se transforme en un programme numérique interactif.**
- **Concevoir une activité au sol basée sur des motifs géométriques, utilisant des objets physiques pouvant être recréés avec les robots Thymio.**
- **Créer un jeu physique « Livre dont vous êtes le héros » à l'aide de cartes ou d'un plateau, qui pourra ensuite être programmé comme une histoire numérique.**
- **Concevoir une activité de cartographie et d'orientation sans appareil électronique, permettant ensuite de programmer un Bee-Bot pour qu'il parcoure des chemins similaires.**
- **Créer un jeu physique avec des niveaux représentant les Objectifs de Développement Durable (ODD), qui pourra ensuite être transformé en un jeu MakeCode Arcade.**
- **Créer des motifs musicaux acoustiques en utilisant des objets du quotidien, programmables ensuite comme instruments numériques.**
- **Concevoir une activité de représentation environnementale basée sur la couleur, qui puisse être transformée en une œuvre d'art numérique interactive.**

Accéder au modèle →

[https://www.canva.com/design/DAGhaGHDeug/oBG61Vf8rlrxAHbjEIPZaQ/view?
utm_content=DAGhaGHDeug&utm_campaign=designshare&utm_medium=link&utm_source=publishsharelink&mode=preview](https://www.canva.com/design/DAGhaGHDeug/oBG61Vf8rlrxAHbjEIPZaQ/view?utm_content=DAGhaGHDeug&utm_campaign=designshare&utm_medium=link&utm_source=publishsharelink&mode=preview)

Aperçu du modèle

Titre de l'activité

Matière principale : Thématiques :

Pratiques informatiques : Décomposition, reconnaissance de motifs, algorithmes, résolution de problèmes...

Points du programme abordés

Point du programme #1
Matière - Niveau scolaire
Description de la contribution de l'activité

Point du programme #2
Matière - Niveau scolaire
Description de la contribution de l'activité

Point du programme #3
Matière - Niveau scolaire
Description de la contribution de l'activité

Point du programme #4
Matière - Niveau scolaire
Description de la contribution de l'activité

Matière nécessaire

-
-
-
-
-

Activité unplugged

Découverte sans écran

Introduction

Nam non ex et nulla sagittis interdum. Sed a nulla purus. Proin mattis laculis scelerisque. Etiam pellentesque erat in mi pharetra, non ultricies turpis gravida. Aenean aliquet elit vitae finibus pellentesque. Curabitur et laus sed nunc ultricies laoreet vitae id velit. Curabitur consequat id ante ac ultricies.

Présentation - Comment introduire l'activité aux élèves

Locutus est amet, consectetur adipiscing elit. Sed sed nulla ac sem sodales sodales vitae maximus risus. Fusce eget enim aliquet, blandit ligula, blandit. Vestibulum eu modis ultricies vestibulum quam. In ultricies, tellus ut fermentum ultricies, sem pulvinar leo, ut maximus est quam a urna. Morbi nec erci eu risus molestie maximus nec a purus. Nunc laoreet quam sollicitudin, pharetra ligula at, hendrerit risus. Nam viverra risus sit amet ligula gravida mollis. Quisque mi felis, ullamcorper eget lectus at, auctor eusimod massa. Phasellus placerat gravida mattis. Nulla vehicula nisi in felis sollicitudin, mattis ullamcorper quam tristique. Nam auctor justo in nisl vestibulum, quis rutrum arcu fringilla.

Déroulement - Étapes à suivre et instructions pour les élèves

Étape 1 - Titre

- Description de l'étape
- Instructions pour les élèves

Étape 2 - Titre

- Description de l'étape
- Instructions pour les élèves

Étape 3 - Titre

- Description de l'étape
- Instructions pour les élèves

Étape 4 - Titre

- Description de l'étape
- Instructions pour les élèves

Liens utiles

-
-
-
-
-

Reconnecter l'activité

Connexion avec la programmation par blocs

Introduction

Blocs de programmation - Capture d'écran de la version réalisée par l'enseignant

Parallèles clés à établir avec les élèves

- [Élément unplugged] → [Équivalent en programmation par blocs]
- [Élément unplugged] → [Équivalent en programmation par blocs]
- [Élément unplugged] → [Équivalent en programmation par blocs]

Déroulement - Étapes de la transposition

Étape 1 - Titre

- Description de l'étape
- Instructions pour les élèves

Étape 2 - Titre

- Description de l'étape
- Instructions pour les élèves

Étape 3 - Titre

- Description de l'étape
- Instructions pour les élèves

Étape 4 - Titre

- Description de l'étape
- Instructions pour les élèves

Conclure sur la séquence

- Astuce pour aider les élèves à faire le lien entre le physique et le numérique :
-
-
-
-
-
-

• Ce que les élèves doivent comprendre et les compétences acquises en conclusion de la séquence :

-
-
-
-
-
-

Fin de la séquence débranchée

Fin de la séquence branchée



Critères d'évaluation des concepts et des compétences acquis

- **Compréhension des concepts fondamentaux :** Les enseignants doivent démontrer une compréhension claire des principes de la programmation par blocs : événements, boucles, séquençage et conditions.
- **Compétences pratiques en programmation :** Les enseignants doivent être capables de concevoir et de mettre en œuvre des activités de programmation de base à l'aide d'outils de programmation par blocs.
- **Sélection des outils pour les activités éducatives :** Les enseignants doivent être capables d'identifier et de sélectionner les outils de robotique et de programmation éducatifs appropriés pour différentes activités.
- **Pensée algorithmique et logique :** Les enseignants doivent démontrer leur capacité à créer des solutions étape par étape aux problèmes en utilisant la programmation par blocs.
- **Créativité informatique :** Les enseignants doivent utiliser la programmation pour créer des projets créatifs et interactifs.
- **Pratiques de codage inclusives :** Les enseignants doivent identifier et traiter les questions d'inclusion dans leurs activités de codage.



Auto-évaluation

1. Qu'est-ce que la programmation par blocs et pourquoi est-elle bénéfique pour les débutants ?

- a) Un langage de programmation qui nécessite une connaissance approfondie de la syntaxe informatique.
- b) Un langage de programmation visuel où les instructions sont représentées par des blocs colorés qui s'emboîtent, rendant l'apprentissage plus intuitif et moins abstrait.
- c) Une méthode de programmation simplifiée qui ne permet pas de créer des programmes complexes.

2. Quel est le but principal des boucles dans la programmation par blocs ?

- a) Répéter une action un nombre fixe de fois seulement.
- b) Créer des variables qui stockent des informations.
- c) Répéter des instructions soit un nombre défini de fois, soit tant qu'une condition est vraie, permettant l'automatisation des tâches répétitives.

3. Comment le séquençage contribue-t-il à la création de programmes ?

- a) Il sert uniquement à organiser visuellement le code.
- b) Il définit l'ordre précis dans lequel les instructions doivent être exécutées pour obtenir le résultat souhaité.
- c) Il est exclusivement utilisé pour la gestion des erreurs dans le programme.

4. Quelle est la fonction principale des conditions (if-else) ?

- a) Elles servent uniquement à détecter les erreurs dans le programme.
- b) Elles permettent au programme de prendre des décisions automatiques en fonction de conditions spécifiques, ce qui le rend plus adaptable.
- c) Elles servent à stocker des données temporaires.

5. Pourquoi l'inclusion est-elle essentielle dans l'enseignement de la programmation ?

- a) Pour créer un environnement d'apprentissage équitable où chaque élève, quelles que soient ses caractéristiques ou son origine, peut développer ses compétences en programmation.
- b) Pour simplifier le contenu des cours de programmation.
- c) Uniquement pour satisfaire aux exigences administratives des établissements.



Auto-réflexion sur le parcours d'apprentissage

Ces questions de conclusion visent à guider la réflexion sur le parcours d'apprentissage réalisé. Elles mettent en lumière le niveau de compréhension des concepts clés, l'utilité des ressources et l'impact des activités sur le développement des compétences.

Vous pouvez nous faire part de vos réflexions, commentaires ou suggestions à l'adresse : contact@labaixbidouille.com.

- Comment décririez-vous votre compréhension globale de la programmation par blocs maintenant, par rapport au début de votre exploration ?
- Quels principes de la programmation par blocs vous semblent les plus clairs, et lesquels soulèvent encore des questions ?
- Dans quelle mesure vous sentez-vous capable de concevoir et d'animer des activités de codage avec la programmation par blocs dans votre propre contexte ?
- Parmi les ressources proposées (vidéos, lectures, activités), lesquelles vous ont été les plus utiles, et pourquoi ?
- Comment ce guide a-t-il influencé votre façon d'envisager l'inclusion dans le codage et l'éducation numérique ?
- En considérant les compétences transversales telles que la pensée logique, la résolution de problèmes, la pratique du codage et la créativité, dans quel domaine estimez-vous avoir le plus progressé ?
- Comment les travaux pratiques et les activités vous ont-ils aidé à approfondir votre compréhension ?
- Si vous pouviez modifier ou améliorer un aspect du guide pour les futurs enseignants en formation, quel serait-il ?