SÉQUENCE D'APPRENTISSAGE

SUR BASE DU PROGRAMME DU TRONC COMMUN

SECONDAIRE S1

NUMÉRIQUE

SÉQUENCE 4 : ROBOTIQUE ET ALGORITHME, LA LOGIQUE DERRIÈRE LA MACHINE

WALLONIE-BRUXELLES ENSEIGNEMENT

Tronc commun

Robotique et algorithmique : la logique derrière la machine

1. Généralités

Titre de la séquence	Robotique et algorithmique : la logique derrière la machine
Année	S1
Discipline	FMTTN – Volet numérique
Champs	Informations et données – Création de contenus – Intelligence artificielle
Temps estimé	5 périodes de 50 minutes
Matériel	Ordinateurs ou tablettes avec accès Internet, cartes micro:bit, câbles USB ou connexion Bluetooth, plateforme MakeCode

2. Le dispositif amène les élèves à…

Cette séquence amène les élèves à découvrir les principes fondamentaux de l'algorithmique et de la programmation à travers des activités concrètes de robotique éducative. Ils apprennent progressivement à comprendre comment les ordinateurs, les robots et certains systèmes d'intelligence artificielle exécutent des instructions conçues par des humains.

À travers l'utilisation du micro:bit et de l'environnement MakeCode, les élèves développent leur pensée computationnelle en apprenant à décomposer un problème, organiser une suite d'instructions, utiliser des boucles, mettre en place des conditions et tester différentes solutions.

La séquence vise également à développer la persévérance, l'autonomie et la collaboration. Les élèves découvrent que l'erreur fait partie intégrante du processus de programmation. Ils apprennent à observer le comportement d'un programme, repérer une erreur, formuler une hypothèse, corriger et améliorer leur production.

Enfin, cette activité permet de mieux comprendre les mécanismes qui se cachent derrière les objets numériques du quotidien. Les élèves découvrent que les systèmes automatisés, les robots et les outils d'intelligence artificielle ne fonctionnent pas de manière magique : ils s'appuient sur des instructions, des règles, des données et des choix humains.

a. **Tableaux des contenus / attendus**

Contenu / Savoir	Attendu
Vocabulaire spécifique au hardware	Distinguer hardware de software, mémoire vive de mémoire morte, stockage interne de stockage externe. Utiliser, adéquatement et en contexte, le terme périphérique.
Programmation et logigrammes	Expliquer le concept de variable.
Fonctionnement de l'IA	Expliquer le fonctionnement de base de l'intelligence artificielle et du Machine Learning.
Fonctionnement de l'IA	Expliquer les limites de l'utilisation de l'IA : biais, hallucinations, stéréotypes…
Vocabulaire spécifique à l'IA	Distinguer « IA » et « IA générative ».

Contenu / Savoir-faire	Attendu
Sélectionner un outil, une application, un logiciel	Sélectionner un outil, une application, un logiciel adéquat, en fonction de l'intention.
Utiliser les fonctions principales d'un outil de création de contenus	Utiliser les fonctions principales d'un outil de création de contenus.
Lire un algorithme simple	Verbaliser un logigramme intégrant une condition, une boucle et une variable.
Écrire un algorithme simple	Écrire un logigramme intégrant une condition, une boucle et une variable.
Lire un programme simple	Lire un programme intégrant une condition, une boucle et une variable.
Écrire un programme simple	Traduire un logigramme intégrant une condition, une boucle et une variable en langage de programmation.

Contenu / Compétence	Attendu
Concevoir un algorithme pour résoudre un problème simple	Concevoir un logigramme intégrant une condition, une boucle et une variable.
Concevoir un programme pour résoudre un problème	Traduire un logigramme intégrant une boucle, une condition et une variable, en langage de programmation ; le tester, le déboguer et l'optimiser.

Interagir / communiquer	Interagir et communiquer de manière orale et écrite, en sélectionnant et en utilisant des outils numériques adéquats.
Porter un regard critique sur l'usage de l'IA	S'interroger sur la pertinence du recours à l'IA dans différents contextes numériques : créativité, résolution de problème, collaboration…

b. Renvoi aux référentiels

Cette séquence s'inscrit principalement dans le champ Création de contenus du volet numérique du référentiel FMTTN, à travers les apprentissages liés à l'algorithmique, à la programmation et à la résolution de problèmes.

Les élèves sont amenés à découvrir les principes fondamentaux de la pensée computationnelle en construisant, lisant et modifiant des algorithmes ainsi qu'en développant des programmes simples répondant à une situation donnée. Les activités proposées mobilisent également la démarche de test, de débogage et d'amélioration progressive d'un programme.

La séquence fait aussi référence au champ Informations et données à travers la distinction entre hardware et software, la découverte du rôle des périphériques et la compréhension du fonctionnement général des systèmes numériques.

Elle établit enfin un lien avec le champ Intelligence artificielle, en permettant aux élèves de comprendre que les systèmes automatisés et certains outils d'IA reposent sur des algorithmes conçus par des humains. Cette ouverture contribue à distinguer intelligence artificielle et intelligence artificielle générative, tout en introduisant les notions de fonctionnement, de limites et d'esprit critique face aux technologies numériques.

3. Lectures associées à cette séquence

a. Renvoi aux repères méthodologiques généraux

Cette séquence mobilise prioritairement les repères suivants : expliciter les démarches et les apprentissages, créer un climat de classe engageant, développer les autonomies, différencier pour soutenir les apprentissages, garder une trace des apprentissages et évaluer pour soutenir les apprentissages.

L'explicitation occupe une place importante, car les concepts de programmation peuvent rester abstraits pour les élèves : algorithme, séquence, boucle, condition, variable, programme, test, débogage. L'enseignant veille à rendre visibles les démarches : comprendre le problème, décomposer la tâche, organiser les étapes, programmer, tester, corriger et améliorer.

La séquence valorise également l'essai et l'erreur. En programmation, une erreur n'est pas un échec : elle constitue une information qui permet de comprendre ce que la machine exécute réellement. Les phases de test et de débogage sont donc pleinement intégrées aux apprentissages.

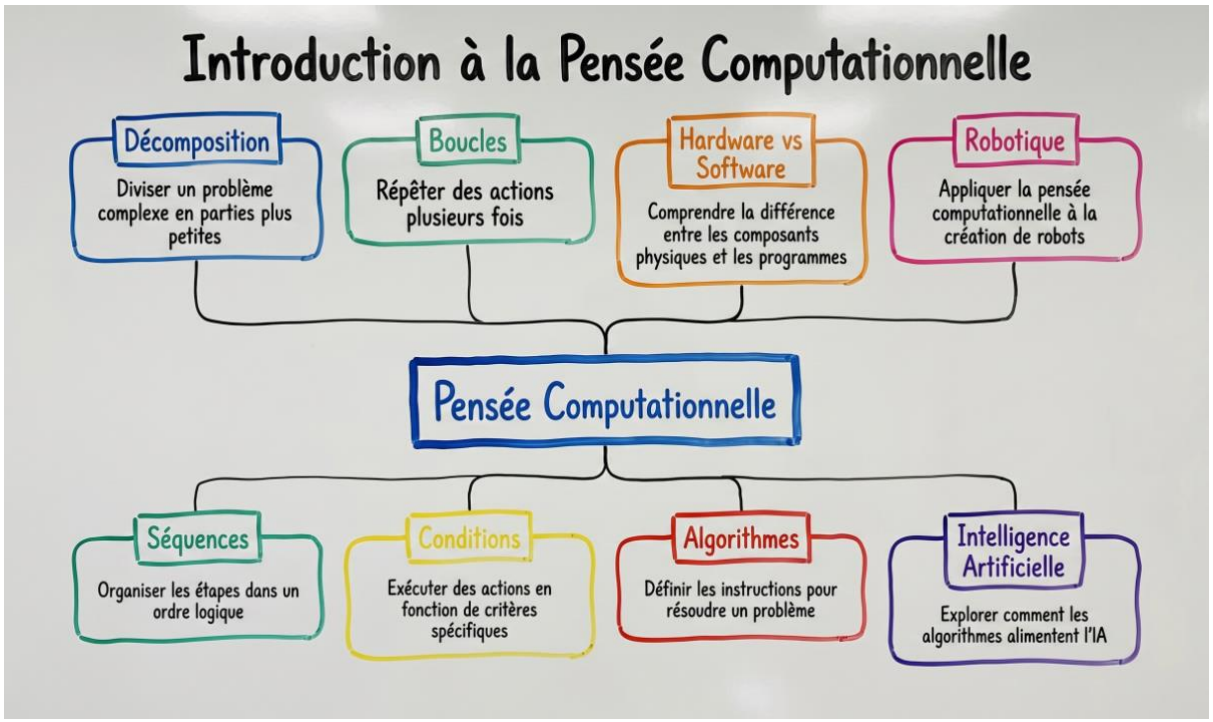
b. Renvoi aux repères méthodologiques disciplinaires

Dans le volet numérique de la FMTTN, cette séquence privilégie une approche concrète de la programmation. L'utilisation d'un environnement visuel par blocs permet aux élèves de se concentrer sur les concepts fondamentaux sans être freinés par la syntaxe d'un langage informatique complexe. Les activités reposent sur des défis progressifs qui amènent les élèves à découvrir, comprendre et réinvestir les notions de séquence, de boucle, de condition et de variable. Cette progression soutient le développement de la pensée computationnelle : décomposer un problème, identifier les actions nécessaires, organiser les étapes et anticiper le comportement attendu de la machine.

Une attention particulière est portée à l'observation du programme en fonctionnement. Les élèves apprennent à comparer le résultat attendu et le résultat obtenu, à formuler une hypothèse sur l'erreur, à modifier leur programme puis à tester à nouveau.

La séquence permet enfin d'établir des liens entre programmation, robotique et intelligence artificielle. Les élèves découvrent que les technologies numériques qui les entourent reposent sur des instructions, des règles et des choix humains.

4. Mise en contexte



Les élèves utilisent quotidiennement des objets numériques, des applications et des systèmes automatisés sans toujours comprendre comment ceux-ci fonctionnent. Les assistants vocaux, les jeux vidéo, les réseaux sociaux, les objets connectés, les robots ou encore les outils d'intelligence artificielle reposent sur des algorithmes et des programmes conçus par des humains.

Cette séquence propose une première découverte de ces mécanismes à travers la programmation d'un micro:bit. Les élèves apprennent à construire des algorithmes simples, à programmer des comportements automatisés et à comprendre les liens entre les instructions qu'ils créent et les actions réalisées par la machine.

L'enjeu est de faire passer les élèves d'une posture d'utilisateurs à une posture de concepteurs. Ils ne se contentent pas d'observer un objet numérique : ils construisent eux-mêmes une suite d'instructions, testent son fonctionnement et l'améliorent.

La séquence permet aussi d'introduire la pensée computationnelle : décomposer un problème, organiser les étapes, repérer les répétitions, prévoir des conditions, utiliser une variable et corriger les erreurs. Ces démarches sont transférables à d'autres situations de résolution de problèmes, au-delà de la programmation.

Moment possible dans l'année

Cette séquence peut être menée après une première approche du vocabulaire hardware/software ou après une activité de découverte des objets numériques. Elle peut également servir d'entrée dans la programmation, car l'environnement MakeCode permet une prise en main progressive.

Elle peut être placée avant une séquence sur l'intelligence artificielle afin d'aider les élèves à comprendre que les systèmes numériques automatisés reposent sur des instructions, des modèles, des données et des choix humains.

Prérequis possibles

Les élèves devraient idéalement avoir déjà rencontré quelques gestes simples :
- utiliser un ordinateur ou une tablette ;
- ouvrir un navigateur ;
- accéder à une plateforme en ligne ;
- déplacer des éléments à l'écran ;
- enregistrer ou retrouver un fichier ;
- travailler en binôme ;
- lire une consigne étape par étape.

Ces prérequis peuvent être très variables selon les élèves. Il est donc utile de prévoir une vérification rapide des acquis et un temps de prise en main de l'environnement MakeCode.

Matériel nécessaire

- Ordinateurs ou tablettes avec accès Internet
- Cartes micro:bit, idéalement une pour deux élèves
- Câbles USB ou connexion Bluetooth
- Plateforme MakeCode : https://makecode.microbit.org/
- Fiches de défis
- Fiche de vocabulaire
- Grille de test et de débogage
- Vidéoprojecteur ou écran pour les démonstrations
- Éventuellement : haut-parleur, piles, capteurs ou accessoires disponibles

Points de vigilance généraux

L'enseignant veille à ne pas réduire la séquence à une simple manipulation technique. Les élèves doivent comprendre ce qu'ils font : quelle instruction est donnée, dans quel ordre, avec quel effet attendu et quel résultat obtenu.

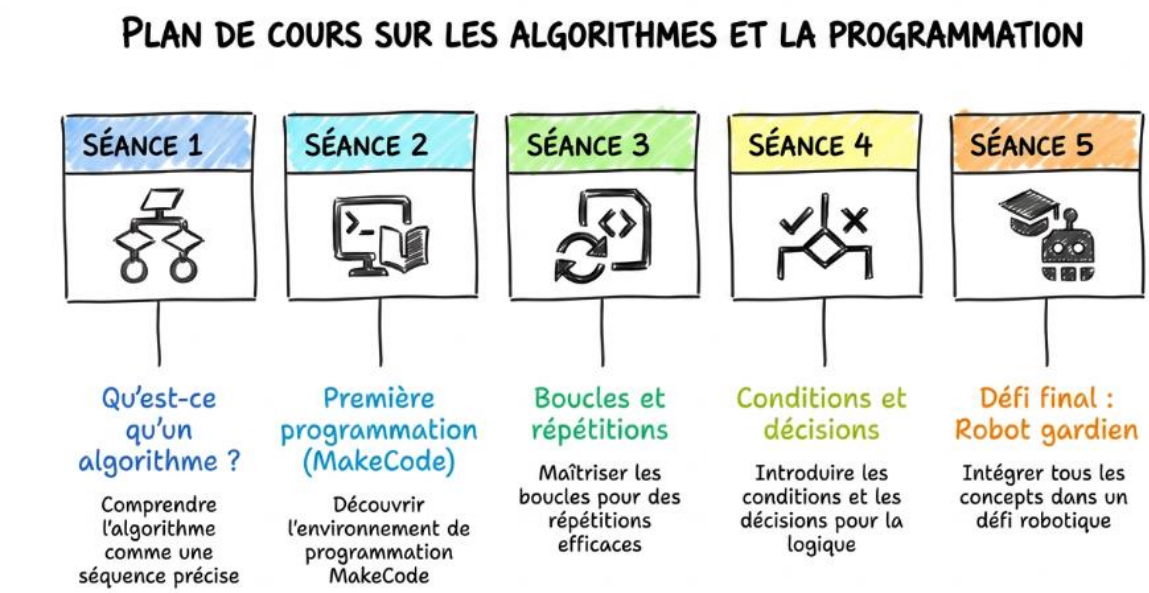
Les erreurs doivent être valorisées comme des étapes normales du travail de programmation. Il est utile d'installer un vocabulaire commun : tester, observer, déboguer, corriger, améliorer.

L'usage du micro:bit peut générer des difficultés matérielles : connexion USB, téléversement, batterie, câble défectueux, lenteur de la plateforme, appareil non reconnu. Ces obstacles doivent être anticipés afin de ne pas masquer les apprentissages visés.

Enfin, l'enseignant veille à adapter la complexité des défis. Les élèves en difficulté peuvent travailler à partir de blocs déjà partiellement assemblés, tandis que les élèves plus avancés peuvent enrichir les programmes avec des capteurs, des variables ou des conditions supplémentaires.

5. Aperçu des étapes méthodologiques-clés de l'activité

Cette rubrique reprend de manière synthétique les étapes didactiques de la séquence. Elle ne détaille pas les modalités pratiques afin de permettre une prise en main rapide de l'activité.



Étape	Intention pédagogique	Ce que les élèves construisent
1. Comprendre ce qu'est un algorithme	Découvrir qu'un algorithme est une suite d'instructions précises et ordonnées.	Les élèves décomposent une tâche simple en étapes et observent l'importance de la précision.
2. Programmer une première action	Découvrir l'environnement MakeCode et la	Les élèves créent un premier programme

	programmation par blocs.	simple et le testent sur micro:bit.
3. Utiliser des boucles	Comprendre comment automatiser une action répétitive.	Les élèves simplifient un programme en utilisant une répétition.
4. Programmer des conditions	Comprendre comment une machine peut réagir différemment selon une situation.	Les élèves programment une réaction liée à un événement ou à un bouton.
5. Relever un défi de programmation	Mobiliser les notions travaillées dans une situation-problème.	Les élèves conçoivent, testent, déboguent et présentent un programme répondant à un défi.

6. Intention de l'activité

a. Intention rédigée à destination des enseignants

L'intention de cette séquence est d'amener les élèves à comprendre les principes fondamentaux de l'algorithmique et de la programmation en concevant des programmes simples à l'aide d'un environnement de programmation par blocs.

La séquence vise à développer la pensée computationnelle des élèves : décomposer une tâche, organiser des instructions, utiliser une boucle, programmer une condition, mobiliser une variable simple, tester un programme, identifier une erreur, le déboguer et l'améliorer.

Elle vise également à construire une première compréhension de la logique qui se cache derrière les systèmes automatisés et certains outils d'intelligence artificielle : une machine exécute des instructions, selon des règles définies, à partir de données ou d'événements.

b. Annonce de l'objectif aux élèves

Nous allons apprendre à comprendre comment une machine exécute des instructions. Pour y arriver, vous allez construire des algorithmes, programmer un micro:bit, utiliser des boucles et des conditions, tester vos programmes, corriger les erreurs et relever un défi de programmation.

7. Déroulement proposé de la séquence

Séance 1 – Qu'est-ce qu'un algorithme ?

Objectif de la séance

Amener les élèves à comprendre qu'un algorithme est une suite d'instructions précises, ordonnées et compréhensibles par celui ou celle qui doit les exécuter.

Activité principale

Les élèves réalisent une activité débranchée à partir d'une situation simple, comme une recette ou une tâche quotidienne. Ils doivent décomposer l'action en étapes précises. Un pair exécute ensuite les instructions sans interpréter ce qui n'est pas écrit.

Consignes aux élèves

- Choisissez une tâche simple à réaliser
- Découpez cette tâche en étapes
- Écrivez les instructions dans l'ordre
- Soyez précis dans chaque action
- Faites exécuter votre algorithme par un autre élève
- Repérez les étapes manquantes ou imprécises
- Corrigez votre algorithme

Trace d'apprentissage

Les élèves conservent :
- un algorithme simple rédigé ;
- une correction après test ;
- une définition construite collectivement : un algorithme est une suite d'instructions précises et ordonnées permettant de réaliser une tâche.

Point d'attention pour l'enseignant

L'analogie avec la recette fonctionne bien si l'on insiste sur l'exécution stricte des instructions. L'élève qui exécute ne doit pas interpréter ce qui manque. Cela permet de montrer qu'une machine ne devine pas l'intention : elle exécute ce qui est programmé.

Séance 2 – Première programmation avec MakeCode

Objectif de la séance

Amener les élèves à découvrir un environnement de programmation par blocs et à créer un premier programme sur micro:bit.

Activité principale

Les élèves découvrent MakeCode et créent un premier programme simple : afficher une icône, un texte ou un motif au démarrage. Ils téléversent ensuite le programme sur le micro:bit et observent le résultat.

Consignes aux élèves

- Ouvrez MakeCode micro:bit
- Créez un nouveau projet
- Choisissez un bloc de démarrage

- Ajoutez une instruction d'affichage
- Vérifiez l'ordre des blocs
- Téléversez le programme sur le micro:bit
- Testez le programme
- Notez ce qui fonctionne et ce qui doit être corrigé

Trace d'apprentissage

Les élèves conservent :
- une capture ou une copie du programme créé ;
- une courte explication du rôle des blocs utilisés ;
- une observation du test réalisé : résultat attendu, résultat obtenu, correction éventuelle.

Point d'attention pour l'enseignant

Les élèves peuvent se concentrer sur le branchement ou le téléversement au détriment du raisonnement. Il est important de rappeler régulièrement le lien entre les blocs choisis, l'ordre des instructions et le comportement observé sur le micro:bit.

Séance 3 – Boucles et répétitions

Objectif de la séance

Amener les élèves à comprendre comment une boucle permet de répéter une action sans réécrire plusieurs fois la même instruction.

Activité principale

Les élèves programment une action répétitive, par exemple faire clignoter une LED, afficher plusieurs fois une icône ou répéter un message. Ils comparent ensuite un programme avec instructions répétées et un programme utilisant une boucle.

Consignes aux élèves

- Programmez une action qui se répète.
- Observez le nombre de blocs utilisés.
- Remplacez les répétitions par une boucle.
- Modifiez le nombre de répétitions.
- Testez le programme.
- Comparez les deux versions.
- Expliquez pourquoi la boucle simplifie le programme.

Trace d'apprentissage

Les élèves conservent :
- un programme avec répétition ;
- un programme utilisant une boucle ;
- une phrase de synthèse : une boucle permet de répéter plusieurs fois une ou plusieurs instructions.

Point d'attention pour l'enseignant

La boucle infinie doit être abordée avec prudence. Elle peut être utile, mais elle peut aussi bloquer le programme ou rendre son arrêt moins évident. Il est nécessaire de montrer comment arrêter, réinitialiser ou modifier le programme.

Séance 4 – Conditions et décisions

Objectif de la séance

Amener les élèves à comprendre qu'une condition permet à un programme de réagir différemment selon une situation ou un événement.

Activité principale

Les élèves programment une réaction conditionnelle sur le micro:bit. Par exemple : si le bouton A est pressé, afficher un sourire ; si le bouton B est pressé, afficher une autre icône ; si le micro:bit est secoué, afficher un message.

Consignes aux élèves

- Choisissez un événement : bouton A, bouton B ou mouvement
- Ajoutez une condition
- Définissez l'action à réaliser si la condition est vraie
- Testez le programme
- Modifiez la condition ou l'action
- Expliquez ce que le micro:bit vérifie avant d'agir

Trace d'apprentissage

Les élèves conservent :
- un programme utilisant une condition ;
- une phrase de type « Si…, alors… » ;
- une courte explication du fonctionnement : le programme vérifie une condition avant d'exécuter une action.

Point d'attention pour l'enseignant

La structure « Si…, alors… » doit être verbalisée à plusieurs reprises. Les élèves peuvent confondre l'événement, la condition et l'action. Il est utile de leur faire formuler leur programme en langage naturel avant de le construire dans MakeCode.

Séance 5 – Défi final : robot gardien

Objectif de la séance

Amener les élèves à mobiliser les notions travaillées dans un défi concret de programmation.

Activité principale

Les élèves conçoivent un programme de « robot gardien ». Le micro:bit doit surveiller une situation et réagir lorsqu'un événement se produit. Par exemple : s'il est secoué, il affiche « Stop ! » ; s'il détecte un mouvement, il affiche une alerte ; si un bouton est pressé, il change d'état.

Consignes aux élèves

- Lisez le défi proposé
- Identifiez ce que le micro:bit doit détecter
- Écrivez la logique du programme en langage naturel
- Construisez le programme dans MakeCode
- Utilisez au moins une condition
- Utilisez une répétition si elle est utile
- Testez votre programme
- Corrigez les erreurs
- Présentez votre solution au groupe

Trace d'apprentissage

Chaque binôme conserve :
- la logique du programme rédigée en langage naturel ;
- le programme réalisé ;
- une trace du test ;
- une correction ou amélioration apportée ;
- une courte justification des choix effectués.

Point d'attention pour l'enseignant

La production finale ne doit pas être évaluée uniquement sur le fait que le programme fonctionne immédiatement. L'important est la démarche : concevoir, tester, observer, déboguer, améliorer et expliquer.

8. Points de vigilance liés aux repères méthodologiques généraux

Créer un climat de classe engageant

La programmation peut provoquer de la frustration lorsque le résultat attendu n'apparait pas. L'enseignant veille à installer un climat dans lequel l'erreur est considérée comme normale et utile. Les élèves doivent pouvoir essayer, échouer, chercher, corriger et recommencer.

Expliciter les démarches et les apprentissages

Les démarches doivent être rendues visibles : décomposer un problème, organiser les instructions, choisir les blocs, tester, comparer le résultat attendu et le résultat obtenu, déboguer. L'enseignant peut régulièrement demander aux élèves de formuler leur programme en langage naturel avant de le construire avec des blocs.

Différencier pour soutenir les apprentissages

Certains élèves peuvent recevoir des blocs préassemblés, une fiche procédure, un défi simplifié ou un programme à compléter. Les élèves plus avancés peuvent ajouter un capteur, intégrer une variable, combiner plusieurs conditions ou proposer une amélioration du défi.

Développer les autonomies

L'autonomie se construit progressivement. Les premières manipulations sont guidées collectivement, puis les élèves sont amenés à tester et corriger eux-mêmes leurs programmes. Le travail en binôme permet de favoriser l'entraide et la verbalisation.

Garder une trace des apprentissages

Les élèves conservent des traces variées : algorithme en langage naturel, programme MakeCode, capture d'écran, fiche de vocabulaire, grille de test, correction apportée, justification du programme. Ces traces permettent de rendre visibles les apprentissages et de soutenir le transfert vers d'autres situations.

Évaluer pour soutenir les apprentissages

L'évaluation accompagne la séquence et permet de réguler les apprentissages. Elle porte sur la compréhension des notions, mais aussi sur la démarche : lire un programme, expliquer une boucle, formuler une condition, tester, déboguer et améliorer.

Points matériels à anticiper

L'enseignant veille à anticiper les difficultés techniques : connexion du micro:bit, câbles, téléversement, batterie, accès à MakeCode, compatibilité des appareils. Il peut être utile de prévoir une version hors ligne de MakeCode ou des binômes lorsque le matériel est limité.

9. Propositions de pratiques d'évaluation formative

L'évaluation formative accompagne l'ensemble de la séquence. Elle porte sur la compréhension progressive des notions d'algorithme, de programme, de boucle, de condition, de variable et de débogage.

Elle permet d'observer :
- la capacité à décomposer une tâche en étapes ;
- l'utilisation d'un vocabulaire adapté : algorithme, programme, instruction, boucle, condition, variable, test, débogage ;
- la capacité à lire un programme simple ;
- la capacité à écrire ou compléter un algorithme simple ;
- l'utilisation correcte d'une boucle ;
- l'utilisation correcte d'une condition ;
- la capacité à tester un programme ;
- la capacité à identifier une erreur et à proposer une correction ;
- la capacité à expliquer le fonctionnement d'un programme à l'oral ou à l'écrit.

Modalités possibles

L'évaluation peut prendre plusieurs formes :
- observation des binômes pendant les défis ;
- questionnement oral lors des phases de test ;
- vérification des algorithmes rédigés ;
- lecture commentée d'un programme MakeCode ;
- grille de débogage ;
- autoévaluation en fin de séance ;
- présentation du défi final.

10. Liens avec les focus WBE

FLSco – Langue de scolarisation

La séquence permet de travailler un vocabulaire spécifique : algorithme, instruction, programme, séquence, boucle, condition, variable, capteur, périphérique, test, débogage, optimisation. Les élèves sont amenés à utiliser ce vocabulaire pour expliquer leurs démarches et présenter leurs programmes.

Pensée critique et complexe

Les élèves apprennent à analyser un problème, à le décomposer, à organiser des étapes et à vérifier si la solution fonctionne. Ils comprennent qu'une machine exécute des instructions et que la qualité du résultat dépend des choix réalisés par le programmeur.

Citoyenneté numérique

La séquence permet de questionner les conséquences des programmes conçus par des humains. Un algorithme mal pensé ou mal testé peut produire des erreurs ou des effets non souhaités. Cette réflexion ouvre vers les enjeux des systèmes automatisés dans la société : objets connectés, robots, recommandations, véhicules autonomes ou outils d'IA.

Éducation aux médias

La séquence peut faire le lien avec les logiques invisibles des plateformes numériques. Les élèves peuvent comprendre que les contenus proposés par certaines applications reposent sur des règles de sélection, des données et des algorithmes. Cette ouverture doit rester simple et adaptée à leur âge.

Éducation relative à l'environnement et au développement durable

La séquence peut ouvrir une réflexion sur la consommation énergétique des objets connectés et des systèmes automatisés. Les élèves peuvent, par exemple, programmer un micro:bit qui s'éteint ou réduit son affichage après un temps donné, afin de relier programmation et usage responsable des ressources.

Prolongements possibles

La séquence peut être prolongée par un défi plus ouvert : créer un mini-jeu, programmer une alarme, utiliser un capteur de température, concevoir un compteur de pas ou imaginer un objet connecté utile dans la classe. Ces prolongements doivent permettre de réinvestir les notions travaillées sans alourdir les attendus de base.

11. Ressources digitales

Le Digipad associé aux séquences rassemble des ressources complémentaires destinées à soutenir leur mise en œuvre en classe : liens utiles, supports pédagogiques, exemples d'activités, outils numériques, vidéos, documents de référence et pistes d'approfondissement.

Il permet de centraliser les ressources dans un espace unique et facilement accessible

Son intérêt est aussi d'offrir un support évolutif. Dans le domaine du numérique, les outils, les usages, les plateformes, les ressources liées à l'IA ou à l'éducation aux médias évoluent rapidement. Le Digipad pourra donc être enrichi, adapté ou mis à jour au fil du temps, sans modifier l'ensemble des séquences.

Il constitue ainsi un espace d'appui pour les enseignants : chacun pourra y puiser des ressources en fonction de son matériel, de son groupe-classe, du temps disponible et des besoins observés chez les élèves.