enseignants en programmation

COFFRE À OUTILS NUMÉRIQUE :

Guide de démarrage

Un programme éducatif de
CANADA EN PROGRAMMATION

Table des matières

ANNEXE : ENSEIGNER AVEC LA TECHNOLOGIE

Consultez le site Web d'Enseignants en programmation pour découvrir d'autres activités hors ligne et des plans de cours :
https://www.canadalearningcode.ca/fr/plans-de-cours/

Bienvenue au Guide de démarrage du Coffre à outils numériques!

Qu'est-ce que le Coffre à outils numérique?

Le Coffre à outils numérique d'Enseignants en programmation comprend des guides conçus pour aider le personnel enseignant à développer les connaissances générales, le savoir et l'assurance nécessaires pour initier les élèves à la programmation et à la technologie.

Dans ces guides, vous trouverez des ressources pour vous familiariser avec les compétences numériques ainsi que des conseils pour les enseigner. De plus, vous aurez accès à des plans de leçons qui vous permettront d'outiller la prochaine génération de technologues du Canada.

Guide de démarrage

Le document que vous lisez en ce moment est le Guide de démarrage. Il comprend des informations générales qui vous prépareront à enseigner la programmation en classe. Ce guide est conçu pour être utilisé conjointement avec nos autres guides pratiques propres à un outil ou à une expérience. Ces derniers contiennent des notions de base sur des outils de programmation et des ressources technologiques recommandées ainsi que des leçons prêtes à être enseignées en classe.

Qui peut utiliser les guides du Coffre à outils numérique?

Les guides du Coffre à outils numérique ont été spécialement conçus pour le personnel enseignant de la maternelle jusqu'à la fin du secondaire. Néanmoins, ils sont faciles à adapter et conviennent aux personnes enseignant dans un autre milieu, comme les responsables des programmes dans les centres communautaires, les parents faisant l'école à la maison, les jeunes Guides, etc.

Quels sont les objectifs du Coffre à outils numérique?

En enseignant l'informatique, nous souhaitons inspirer la population du Canada à devenir des citoyens numériques bien outillés pouvant comprendre et transformer la société et y contribuer en créant et en innovant avec la technologie.

Objectifs d'apprentissage

Après avoir lu le Guide de démarrage du Coffre à outils numérique d'Enseignants en programmation, le personnel enseignant pourra...

- Reconnaître les compétences et les aptitudes développées en enseignant l'informatique et la programmation
- Définir des concepts de programmation de base et les expliquer à l'aide d'exemples concrets
- Déterminer et modifier (ou « remixer ») des composantes des leçons de Canada en programmation
- Appliquer quatre stratégies pour modifier le contenu afin de répondre aux besoins des élèves
- Soutenir les élèves durant le débogage et la résolution de problèmes dans des projets de technologie à l'aide de questions d'orientation et de la liste de vérification pour le débogage
- Appliquer sept stratégies d'enseignement pour les ateliers de technologie

La programmation pour les débutants

Qu'est-ce que le code?

L'explication la plus simple est la suivante : un ensemble de consignes données à un ordinateur pour exécuter une tâche. Quand nous mettons les consignes dans un ordre précis, nous créons un algorithme (un ensemble de consignes à suivre par étape pour résoudre un problème).

Les ordinateurs suivent nos directives à la lettre. Par conséquent, les consignes dans les algorithmes doivent être très précises. Par exemple, en lisant une recette pour préparer un sandwich au beurre d'arachides et à la confiture (un algorithme), un humain interpréterait « mettre la confiture sur le pain » comme une consigne d'étaler de la confiture sur une tranche de pain. Par contre, un ordinateur interpréterait cette directive différemment. Le robot doit-il mettre le pot de confiture sur le sac de pain? Pour les humains, il s'agit d'une directive facile à comprendre, mais les ordinateurs interprètent tout littéralement. Afin que l'ordinateur comprenne, nous devons lui donner les consignes suivantes :

1. Tourner le couvercle du pot de confiture en sens horaire jusqu'à ce que le couvercle soit complètement dévissé
2. Soulever le couvercle du pot de confiture
3. Placer le couvercle à côté du pot de confiture

...et ainsi de suite. Voyez une démonstration de ce concept : **bit.ly/pbandj-challenge**.[1]

Le code est extrêmement polyvalent. On utilise des lignes de code pour donner des consignes aux ordinateurs et aux robots, créer des pages Web, concevoir des jeux vidéo, analyser de gros ensembles de données, et plus encore. Nous utilisons des langages informatiques différents pour réaliser des tâches différentes (p. ex. le HTML et le CSS pour créer des sites, le R pour analyser des données, Scratch pour créer des jeux, MakeCode et Python pour manipuler un micro:bit et le JavaScript pour faire voler des robots). Malgré leurs différences, ces langages ont le même but, c'est-à-dire communiquer des consignes, des algorithmes, à un ordinateur.

Pourquoi enseigner l'informatique et la programmation?

En plus d'amener les élèves à acquérir des compétences et des habiletés importantes, un enseignement de l'informatique complet les aidera à développer les aptitudes et les attitudes nécessaires pour participer à la société du futur.

Découverte : L'informatique enseigne aux élèves à résoudre des problèmes en mettant l'accent sur la curiosité et la découverte. Ils seront poussés à vivre de nouvelles expériences, à persévérer en réalisant des tâches et à apprendre de leurs erreurs en acquérant des compétences.

Pensée critique : En comprenant le fonctionnement des ordinateurs, les élèves seront bien préparés à penser aux aspects sociaux, juridiques, éthiques et politiques de la technologie de manière critique.

Travail d'équipe : L'informatique amène les élèves à travailler en équipe, à réaliser des projets, à communiquer efficacement, à gérer des tâches, à développer leur empathie et à comprendre le point de vue d'autrui. De plus, les élèves apprennent à faire des commentaires constructifs, à accepter les rétroactions et à s'entraider.

Citoyenneté : L'enseignement de l'informatique développe la capacité des élèves à comprendre les effets de la technologie sur la société. Ils seront prêts à agir à titre d'intendants de la technologie et à exploiter celle-ci pour améliorer le monde qui les entoure.

Résilience : En créant des artéfacts numériques, les élèves gagneront en confiance et n'auront pas peur d'essayer de nouvelles choses, de faire des erreurs et d'apprendre de leur expérience. L'enseignement de l'informatique souligne l'importance de l'apprentissage continu et imprévu pour développer la résilience des élèves et les pousser à voir les échecs comme une occasion d'apprendre.

Créativité : L'enseignement de l'informatique encourage les élèves à faire preuve de créativité, à penser autrement et à développer des solutions novatrices pour régler des problèmes qui les touchent personnellement ou qui touchent leur communauté ou la planète entière.

Pensée informatique : L'informatique aide les élèves à appliquer des stratégies de résolution de problèmes tant dans le monde numérique que dans le monde réel. Consultez la section « Pensée informatique » pour en savoir plus.

Concepts de programmation clés : L'enseignement de l'informatique amène les élèves à acquérir des concepts de programmation communs à des centaines de langages informatiques distincts. Les élèves pourront apprendre différents langages sans problème s'ils comprennent les concepts clés. Consultez la section « Concepts de programmation clés et terminologie » pour en savoir plus.

Pensée informatique

La pensée informatique est un ensemble de processus et de stratégies servant à déterminer la nature d'un problème et à préparer une solution qui sera comprise par un ordinateur (ou un autre humain). Il s'agit de diviser de gros problèmes en petits problèmes et de penser aux étapes nécessaires pour les surmonter. Les concepts de pensée informatique ne sont pas propres à la programmation. Vous remarquerez que ce sont des principes de résolution de problème auxquels nous faisons appel dans la vie de tous les jours.

Voici les étapes principales du processus de pensée informatique :

I. Décomposition

La décomposition est la division d'un gros problème en petits problèmes faciles à gérer. Le processus ressemble à la division d'un compte-rendu de livre en différentes sections.

II. Reconnaissance des formes

La reconnaissance des formes est la détermination des caractéristiques communes aux petits problèmes pour faire des prédictions, établir des règles et résoudre des problèmes généralement.

III. Abstraction

L'abstraction est l'isolation des différences entre les petits problèmes pour trouver une solution qui fonctionne avec plusieurs problèmes. En gros, nous déterminons les éléments les plus importants et les plus négligeables des problèmes. L'abstraction nous permet de gérer la complexité d'un problème, comme quand nous choisissons les informations essentielles pour la résolution d'un problème en maths (une équation ou un problème textuel).

IV. Algorithmes

Un algorithme est un ensemble de consignes à suivre par étape pour résoudre un problème. Les algorithmes sont partout dans notre vie : un plan de cours est un algorithme pour une leçon tandis qu'une recette est un algorithme pour préparer notre plat préféré. En écrivant des consignes avec des phrases, nous faisons ce que l'on appelle du « pseudo code ».

Concepts de programmation et terminologie

Il y a des centaines de langages informatiques. Bien que ceux-ci ne se ressemblent parfois pas du tout, ils ont tous des concepts communs.

Comme vous le constaterez, la plupart des concepts de programmation sont des concepts généraux que nous rencontrons au quotidien. Il est conseillé d'utiliser les analogies pour expliquer les concepts plus difficiles aux élèves!

Concept / terme	Définition	Exemple concret ou analogie
Algorithme	Ensemble de consignes à suivre pour résoudre un problème.	Nous suivons tous un algorithme semblable en nous brossant les dents. Nous mettons du dentifrice sur notre brosse à dents, frottons nos dents avec la brosse et rinçons notre bouche.
Tableau	Variable spéciale enregistrant plus d'une valeur à la fois. Les éléments sont numérotés afin que nous puissions nous y référer plus tard. Par exemple, un tableau appelé « chiens » pourrait contenir les éléments suivants : caniche, chihuahua et labrador.	Un tableau est comme un album contenant des photos de nos dernières vacances en famille. Un album nous permet de stocker plusieurs souvenirs (photos) sur un même thème à un seul endroit. Quand nous avons envie de nous replonger dans ces souvenirs, nous ouvrons notre album photo.
Logique booléenne	Calcul algébrique faisant appel à des opérateurs comme « et », « ou » ou « non ». Les valeurs comprises dans le calcul doivent être fausses ou vraies (p. ex. « SI j'ai chaud ET que je suis au frais, ALORS, je me sens bien. »)	Nous pouvons utiliser la logique booléenne pour classer les chats selon la couleur de leur pelage. Groupe 1 : Chats noirs ET blancs Groupe 2 : Chats complètement noirs OU blancs Groupe 3 : Chats seulement noirs, NON blancs
Bogue	Erreur empêchant un programme d'être exécuté normalement.	Quand nous raccordons un tuyau au robinet pour remplir notre piscine, nous nous attendons à voir de l'eau sortir de l'embout. Si le tuyau est plié ou percé, le débit de l'eau est ralenti et le tuyau ne fonctionne pas comme prévu. En fait, les plis et les trous sont des bogues!
Langage informatique	Moyen utilisé par les humains pour communiquer avec un ordinateur (p. ex. HTML, CSS, Python, Scratch, JavaScript, etc.).	Nous parlons différentes langues entre nous, selon notre compréhension. Ce guide est en français, mais vous et votre famille parlez peut-être une autre langue. Tout comme les humains parlent différentes langues entre eux, les ordinateurs utilisent ou interprètent différents langages informatiques.
Commande	Consigne donnée à un ordinateur.	Nous pouvons commander un ordinateur comme nous entraînons un chien à faire des actions!
Expression conditionnelle	Expression permettant aux ordinateurs de prendre des décisions selon certaines conditions. Les expressions conditionnelles contiennent souvent les mots « si / sinon ».	Nous utilisons des expressions conditionnelles pour prendre des décisions au quotidien! Par exemple : « S'il pleut, je vais sortir mon parapluie. S'il ne pleut pas, je ne vais pas le sortir! »

Concepts de programmation et terminologie

Concept / terme	Définition	Exemple concret ou analogie
Débogage	Trouver des problèmes dans du code et les résoudre.	Dans notre exemple de tuyau qui fonctionne mal, le processus de débogage serait le suivant : 1) examiner le tuyau sur toute sa longueur pour trouver des problèmes (un trou ou un pli) 2) mettre du ruban adhésif pour boucher le trou ou appuyer sur le tuyau pour le déplier.
Événement	Action qui entraîne une autre action.	Quand l'horloge affiche 12 h, nous allons dîner. Quand elle affiche 15 h, nous avons fini l'école.
Fonction	Procédure exécutant une action distincte. Il existe des fonctions préfabriquées, comme le bloc « Rebondir si le bord est atteint », qui comprennent un ensemble de commandes à exécuter.	Voyez les fonctions comme le refrain d'une chanson. En écrivant, nous n'avons qu'à composer le refrain qu'une fois, car il se répète. Quand le refrain est composé et nommé, il suffit d'écrire « refrain » dans les paroles pour que tout le monde comprenne la référence. En faisant cela, nous économisons du temps.
Index	Emplacement précis d'un objet dans un tableau. Le premier objet dans un tableau a un index de 0.	Quand une enseignante prend les présences, elle lit une liste (ou un tableau) contenant le nom de toutes les personnes dans le groupe. Le nom des élèves est indiqué sur la liste à une certaine position (un numéro d'index) selon les premières lettres du nom de famille.
Entrée	Informations fournies par l'utilisateur ou détectées par l'ordinateur dans son environnement.	Quand nous voulons acheter un sac de croustilles précis dans une machine distributrice, nous appuyons sur un bouton avec une lettre et un nombre (p. ex. A5). Ces informations sont l'entrée.
Boucle	Élément qui exécute une séquence plusieurs fois, tant qu'une condition est satisfaite.	« Si l'assiette est encore sale, je vais continuer à la laver. Je n'arrêterai que quand elle sera complètement propre! » Certaines boucles n'ont pas de fin! Un exemple concret de boucle est le temps. Le temps est... infini! Il ne s'arrête jamais.
Modularisation	Exploration des liens entre les parties et le tout.	Dans du code modulaire, un programme de grande taille est divisé en petites parties pour faciliter la gestion et la lisibilité. Chaque petite partie a son propre rôle dans le programme. À titre d'analogie, pensez à une équipe qui travaille sur un projet scolaire. Si une seule personne travaille sur le projet, la tâche est plus grande, donc plus difficile à gérer. En divisant le projet en petites tâches, les membres de l'équipe peuvent réaliser le projet plus facilement et efficacement (tant que les membres communiquent). Nous faisons aussi de la modularisation en solvant des équations mathématiques comme =4×7+5÷2. Pour résoudre ce problème, nous allons le diviser en petites parties, d'abord (4×7), puis (5÷2).

Concepts de programmation et terminologie

Concept / terme	Définition	Exemple concret ou analogie
Opérateurs	Expressions mathématiques et logiques (p. ex. le bloc « X et X » dans Scratch). Les opérateurs relationnels sont utilisés dans les comparaisons (p. ex. == (égale), != (n'égale pas), >= (plus grand que ou égal à), etc.). Les opérateurs arithmétiques sont utilisés dans les calculs (p. ex. + (addition), - (soustraction), etc.).	Les symboles dictent nos actions tous les jours! Quand nous voyons un panneau d'arrêt, nous arrêtons. Les panneaux « SORTIE » nous indiquent le chemin vers la sortie d'un immeuble. Nous connaissons le sens des symboles + et - dans les équations mathématiques et les ordinateurs utilisent les mêmes symboles (ou des symboles très semblables).
Sortie	Informations fournies par l'ordinateur à l'utilisateur.	Quand nous appuyons sur un bouton avec une lettre et un nombre (p. ex. A5), la machine distributrice fait sortir notre choix de collation comme sortie.
Parallélisme	Caractère d'actions se déroulant en même temps.	Pouvez-vous toucher votre tête et frotter votre ventre en même temps? C'est un exemple concret de parallélisme : notre cerveau dit à nos bras de faire deux actions différentes en même temps!
Programme	Algorithme programmé pour être	exécuté par une machine; le programme est écrit dans un langage informatique (synonyme de « code » et de « logiciel »). Les programmes sont comme une recette secrète qui se transmet de génération à génération. La recette est l'algorithme. Comme un programme, elle a été écrite par quelqu'un afin qu'une autre personne puisse la faire. (Par contre, n'oubliez pas que les programmes sont seulement exécutés par les ordinateurs, pas par les humains!)
Remixage	Création d'une nouvelle idée ou d'un nouveau projet en modifiant une idée ou un projet qui existe déjà.	Les DJs remixent des chansons populaires en changeant le rythme, en ajoutant des sons, en intégrant des extraits d'autres chansons, etc

Concepts de programmation et terminologie

Concept / terme	Définition	Exemple concret ou analogie
Séquence	Série de commandes essentielles afin de réaliser une tâche. Les ordinateurs lisent et exécutent des commandes dans un ordre précis, de haut en bas.	Pensez à l'exemple d'algorithme de brossage des dents. Avez-vous déjà essayé de brosser vos dents en rinçant votre bouche d'abord et en frottant vos dents avec du dentifrice par la suite? La séquence (l'ordre des actions) est très importante!
État	Caractéristique d'un élément. Le sens en programmation est le même que dans les autres domaines (p. ex. : la télé est allumée ou éteinte). Certaines variables ont des états, d'autres n'en ont pas. Par exemple, 42 veut dire 42, et c'est tout.	L'état d'un téléviseur ou d'un interrupteur peut être « en marche » ou « en arrêt ».
Chaîne	Type de données représentant du texte.	Toute ligne de texte peut être une chaîne : un mot, une phrase, des lettres au hasard, etc.
Syntaxe	Ensemble de règles pour écrire un langage informatique afin que l'ordinateur puisse le comprendre. Dans les langages à blocs comme Scratch, il n'y a pas de syntaxe, car les blocs constituent la structure.	Les différents langages humains (p. ex. le français, l'anglais et l'espagnol) ont des expressions et un vocabulaire différents et des règles de grammaire et d'orthographe à suivre. Si nous ne suivons pas les règles, les personnes à qui nous parlons peuvent avoir de la difficulté à nous comprendre. Il est important de respecter la syntaxe des langages informatiques afin que l'ordinateur puisse comprendre nos commandes!
Variable	Éléments enregistrant une information qui change au fil du temps (p. ex. : le pointage d'un joueur mis à jour dans un jeu).	Nous pouvons comparer une variable à une tirelire ou à un portefeuille avec de l'argent. Le montant d'argent que nous avons change selon nos actions, par exemple, si nous avons payé le loyer ou magasiné. Pour savoir combien d'argent nous avons, il suffit de regarder dans notre tirelire ou notre portefeuille.

Stratégies de débogage

Quand les élèves ont de la difficulté et sont mécontents, il est tentant de résoudre les problèmes à leur place. Toutefois, la résolution de problèmes et les erreurs sont essentielles au processus d'apprentissage.

En cas de problème, restez calme et profitez-en pour enseigner des stratégies de débogage et de résolution de problèmes. Définissez vos attentes! Dites aux élèves que les blocs de code qui ne fonctionnent pas font partie du processus de programmation. En fait, le débogage est de la programmation!!

Questions d'orientation

Nous vous recommandons de poser ces questions pour orienter et motiver les élèves qui ont du mal à déboguer [2] :

« Nous allons diviser le problème en petites étapes. Quelle est la première étape? »

« Si je suis l'utilisatrice, comment puis-je interagir avec le projet? »

« Lisez le code du projet comme si vous lisiez un texte. Que remarquez-vous? Avez-vous oublié quelque chose? »

« Qu'essayez-vous de faire? À votre avis, quels éléments ne fonctionnent pas? Qu'avez-vous essayé pour régler les problèmes? »

« Allez [ici]. Voyez-vous quelque chose qui pourrait vous aider à résoudre votre problème? »

Liste de vérification

Demandez aux élèves d'utiliser la liste de vérification suivante pendant le débogage [3,4,5]:

Avez-vous...

☐ **Lu votre code à voix haute?**
La lecture du code à voix haute, comme s'il s'agissait d'un texte, peut aider les élèves à trouver des erreurs de logique ou des éléments manquants.

☐ **Demandé à un partenaire de regarder votre problème? Ou travaillé en équipe pour résoudre le problème?**
Parfois, il suffit d'avoir une deuxième paire d'yeux pour résoudre des problèmes!

☐ **Isolé et essayé le code?**
Retournez à une vieille version fonctionnelle de votre code. Ensuite, insérez les nouvelles lignes du code actuel en enlevant une commande à la fois. Remettez une commande à la fois dans le code jusqu'à ce que vous ayez reproduit le problème.

☐ **Pensé comme un ordinateur?**
Vos étapes sont-elles assez précises? N'oubliez pas que les ordinateurs ne sont pas très intelligents. Il faut leur dire exactement quoi faire!

☐ **Cherché sur Google?**
Posez la question à notre ami Google! Si vous avez des problèmes, que ce soit avec le Wi-Fi, votre ordinateur, votre clavier ou votre code, tapez votre problème dans Google. La plupart du temps, vous trouverez une solution.

Si Google n'a pas de solution pour vous, consultez la section « Aide » ou « Documentation » des outils numériques que vous utilisez.

☐ **Essayé autre chose?**
En cas de doute, essayez quelque chose! Si cela ne fonctionne pas, vous n'avez qu'à appuyer sur les boutons « Défaire » et « Refaire ».

☐ **Fait une pause?**
Quand nous ressentons trop de frustration, nous n'avons pas les idées claires.

☐ **Redémarré l'appareil?**
Ce n'est peut-être pas efficace pour la programmation, mais si vous travaillez avec de la robotique ou du matériel informatique, le redémarrage est parfois la solution.

Planification des cours

Il y a plusieurs façons d'enseigner la programmation! Vous pourriez démarrer une activité parascolaire ou intégrer la programmation à vos cours. Par exemple, plutôt que de demander aux élèves d'écrire un compte-rendu d'un événement ou le résumé de la vie d'un personnage historique, vous pourriez leur demander de créer un site Web sur le sujet! N'hésitez pas à faire preuve de créativité.

Exigences minimales

Vous aurez besoin de :

- ☐ **Personnel enseignant**
  Le personnel enseignant supervisera les élèves et animera les séances et les exercices de programmation.

- ☐ **Mentors bénévoles (facultatif, mais recommandé)**
  Les mentors répondront aux questions des élèves pendant que le personnel enseignant donne le cours. Nous vous conseillons de confier ce rôle à des élèves avancés.

- ☐ **Contenu**
  Utilisez les plans de leçons du Coffre à outils numérique ou de notre site :https://www.canadalearningcode.ca/fr/plans-de-cours/.

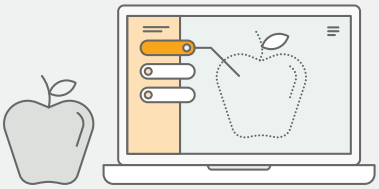
- ☐ **Accès au matériel**
  - ☐ Ordinateur portable ou de bureau (au moins un ordinateur pour deux élèves)
  - ☐ Rallonges et multiprises
  - ☐ Projecteur, écran de projection et clés matérielles pour brancher vos ordinateurs (Vous pouvez aussi trouver une autre façon d'afficher votre écran à l'avant de la classe.)
  - ☐ Matériel (comme indiqué dans le plan de leçon ou le guide pratique)

- ☐ **Connexion Wi-Fi (facultatif, mais recommandé)**
  Certains outils peuvent être utilisés sans Internet, mais d'autres exigent une connexion en tout temps.

- ☐ **Disposition du local**
  Le local doit comprendre un ordinateur pour le personnel enseignant ainsi qu'un projecteur ou un écran pour afficher les diapositives à l'avant de la classe. Les ordinateurs des élèves doivent être disposés de façon à ce que l'écran à l'avant soit visible en tout temps.

Structure de leçon suggérée

Nous vous recommandons d'avoir les sections suivantes dans chaque leçon :



1. Introduction

L'introduction annonce la couleur de la leçon. Commencez par faire une activité brise-glace et définir vos attentes. Chez Canada en programmation, nous définissons nos attentes avec le Code des programmeuses et des programmeurs (consultez la section « Gestion de classe » pour en savoir plus). Ensuite, posez des questions essentielles liées au sujet de l'atelier et donnez des informations supplémentaires au besoin. Terminez cette section en montrant un exemple du projet final aux élèves!

2. Programmation en groupe

Pendant la programmation en groupe, présentez (ou révisez) les fonctions de l'outil principal en réalisant une activité d'apprentissage exploratoire. Donnez aux élèves l'occasion d'explorer l'outil et d'essayer des fonctions de manière structurée à l'aide d'exercices. Montrez aux élèves comment résoudre des problèmes et chercher des ressources dans le but de trouver des solutions. Pour ce faire, demandez aux élèves de trouver comment faire certaines actions (plutôt que de leur donner la réponse). Verbalisez votre raisonnement et votre démarche du point A au point B et redirigez les questions vers d'autres élèves ou vers la référence.

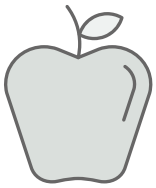
3. Séance de travail

Commencez la séance de travail en demandant aux élèves de vous aider à créer le projet principal. Posez-leur des questions et accompagnez-les pendant qu'ils font des exercices. À ce point-ci, la séance de travail ressemble à l'activité de programmation en groupe. Quand vous sentez que les élèves sont prêts, laissez-les travailler de manière autonome pour terminer le projet et le personnaliser (le remixer). N'oubliez pas de faire le tour de la classe pendant que les élèves travaillent. Si les élèves finissent tôt, proposez-leur des exercices ou des activités de prolongement tirées du plan de leçon.

4. Conclusion

Durant la conclusion, donnez l'occasion aux élèves de présenter leurs projets au reste du groupe! Vous pourriez présenter les projets sous forme de « galerie d'art ». Les élèves laisseront leur ordinateur ouvert à l'écran de leur projet, rangeront leur chaise et se lèveront pour faire tour de la classe et admirer les créations de tout le monde. Avant de conclure, n'oubliez pas de réviser les concepts de programmation abordés et de souligner les objectifs d'apprentissage atteints!

Listes de vérification pour l'enseignement



Avant chaque leçon...

Préparation : 2 heures

- ☐ Lisez le plan de leçon et les ressources d'accompagnement (p. ex. diapositives, exemple de projet, corrigé, documents à distribuer, etc.).
    - ☐ Révisez les concepts de programmation et de codage et exercez-vous à les expliquer au groupe avec assurance.
    - ☐ Créez ou programmez (s'il y a lieu) le projet principal. Assurez-vous que vous comprenez bien toutes les étapes de création. (Partagez votre projet avec le groupe à titre d'exemple!)
    - ☐ Réalisez la leçon pour déterminer les problèmes potentiels avec la technologie.
- ☐ Créez des comptes sur les applications et sites utilisés (s'il y a lieu).
- ☐ Ayez un plan d'urgence en cas de panne d'Internet.
    - ☐ Imprimez un exemplaire du plan de leçon et du corrigé.
    - ☐ Téléchargez les logiciels pour les utiliser hors ligne et transférez les fichiers sur une clé USB.
- ☐ Imprimez les ressources d'apprentissage (s'il y a lieu).
- ☐ Trouvez les outils technologiques et procurez-vous le matériel.
- ☐ Installez et configurez le matériel et les logiciels.
- ☐ Écrivez le Code des programmeuses et des programmeurs, le mot de passe du Wi-Fi et les autres identifiants requis à un endroit visible par tous (p. ex. tableau blanc, carton, etc.).

Après chaque leçon...

- ☐ Rangez votre matériel.
- ☐ Prenez le temps de faire un retour sur l'expérience :
    1. Avez-vous rencontré des problèmes ou des obstacles durant cette leçon? Comment avez-vous résolu ces problèmes?
    2. Selon vous, quels obstacles les élèves rencontreront-ils plus tard? Comment pourriez-vous vous préparer à soutenir les élèves et favoriser leur réussite?
- ☐ Obtenez des rétroactions des élèves. Quels éléments ont-ils aimés? Quels éléments pourraient être améliorés?

Modification des plans de leçons

Quand est-il recommandé de modifier une leçon?
Il est recommandé de modifier légèrement le contenu si vos élèves ont de la difficulté à réaliser la leçon dans les délais indiqués. Vous pouvez également personnaliser le contenu pour l'intégrer à une autre matière. Il est aussi possible d'utiliser un autre outil ou de créer un nouveau projet.

Comment modifier une leçon?
Il y a différentes façons de modifier une leçon pour répondre aux besoins de votre groupe.

Si vos élèves sont avancés, nous vous recommandons les modifications et stratégies suivantes :

1. Prolongement et exercices supplémentaires
Les activités de prolongement et les exercices supplémentaires sont des tâches à proposer aux élèves ayant terminé le projet principal. En plus de meubler du temps, ces tâches permettent aux élèves avancés de continuer à développer leurs compétences. Toutes nos leçons comprennent des activités de prolongement ou des exercices supplémentaires. Toutefois, nous vous invitons à créer vos propres activités et exercices!

2. Remixage
Après chaque activité de programmation en groupe, vous ferez une séance de travail individuel pendant laquelle les élèves sont invités à personnaliser le projet principal. Nous appelons cela du « remixage » de projet! Si les élèves finissent l'activité de programmation en groupe rapidement, vous pouvez prolonger la séance de travail et encourager les élèves à créer. Posez des questions pour aider les élèves à trouver de nouvelles idées en groupe. Par exemple : « Comment pourrions-nous rendre notre projet plus [amusant, difficile, drôle, etc.]? » ou « Que manque-t-il dans le projet? ».“

Si vos élèves prennent du retard, nous vous recommandons les modifications et stratégies suivantes :

1. Programmation à deux
La programmation à deux est une technique de développement logiciel où deux personnes travaillent ensemble sur un projet. Formez des groupes avec des élèves de même niveau ou jumelez un élève avancé avec un élève qui a pris du retard. Dans tous les cas, le travail d'équipe permettra aux élèves de réaliser les tâches plus rapidement et les encouragera à résoudre des problèmes à deux. Consultez la section « Stratégies d'enseignement » pour en savoir plus sur la programmation à deux.

2. Prolongation des ateliers
Si vous n'avez pas le temps de terminer le projet principal, vous pouvez toujours prolonger (raisonnablement) les séances. Assurez-vous de prévoir des pauses pendant les ateliers prolongés! Vous pourriez aussi réaliser un atelier sur plusieurs jours pour mieux répondre aux besoins de votre groupe. Quand vous divisez vos ateliers, prenez le temps de réviser les concepts abordés précédemment au début de chaque séance. Vous pourriez expliquer ces concepts vous-même ou demander aux élèves de vous rappeler les notions et d'expliquer des concepts qu'ils ont appris au reste du groupe.

Finalement, en remixant ou en personnalisant du contenu, pensez au langage, au programme ou à l'outil que les élèves utiliseront pour présenter leurs acquis dans une matière. Pour vous aider, la majorité des leçons de Canada en programmation (**https://www.canadalearningcode.ca/fr/plans-de-cours/**) peuvent être divisées en trois composantes : le sujet, le projet, et le langage ou l'outil. Par exemple, dans la leçon originale « Paysages sonores de la faune », les élèves découvrent l'écosystème de la forêt boréale (le sujet) en créant un paysage sonore numérique (le projet) avec Scratch (l'outil).

Pour personnaliser ce contenu, il suffit de remplacer une de ces composantes. Vous pourriez remixer le projet en demandant aux élèves de présenter leurs acquis sur un écosystème en créant une histoire animée avec Scratch. Une autre solution serait de remixer le sujet de la leçon en demandant aux élèves de faire une recherche sur l'histoire du Canada et de créer un paysage sonore numérique avec Scratch qui représente le style et les sons d'une époque.

Servez-vous des plans de leçons comme point de départ. N'ayez pas peur de les modifier et de laisser libre cours à votre imagination! Les possibilités sont infinies.

Conseils pour enseigner la programmation en s'amusant

1. Assurez-vous d'être à l'aise avec l'outil, mais ne vous inquiétez pas si vous ne le maîtrisez pas à la perfection. Laissez les élèves s'entraider!

2. Ayez une idée claire de ce que vous souhaitez accomplir et trouvez un collègue qui enseignera à vos côtés. La clé est d'intégrer l'apprentissage de la programmation à votre enseignement.

3. Concentrez-vous sur la croissance et l'apprentissage par l'échec. Rappelez au groupe que les compétences ne s'acquièrent pas seulement avec du talent, mais en travaillant dur et en faisant preuve de persévérance.

4. Invitez des spécialistes, des conférenciers et des bénévoles de votre région qui animeront de petites activités et offriront un soutien supplémentaire à l'apprentissage de la programmation.

5. Laissez la créativité des élèves vous guider. Que souhaitent-ils explorer et apprendre?

Stratégies d'enseignement

Enseignerez-vous pour la première fois? Avez-vous des doutes? Pour vous donner un coup de pouce, voici quelques stratégies d'enseignement :

1. Programmation à deux

La programmation à deux est une technique de développement logiciel où deux personnes travaillent ensemble sur un projet. À tout moment, une personne écrit le code (le « conducteur ») tandis que l'autre vérifie le code pour trouver des erreurs (le « navigateur »). Les deux personnes inverseront leurs rôles plusieurs fois, ce qui favorisera le développement égal de leurs compétences.

Plusieurs études démontrent que les élèves d'un certain groupe d'âge ont de bons résultats lorsqu'ils travaillent en équipe sur une même tâche.[6,7,8] Vous remarquerez qu'ils réaliseront les activités plus rapidement et que la résolution de problèmes prendra moins de temps, car ils corrigeront les bogues au fur et à mesure. La nature collaborative de la programmation à deux permet aux élèves d'être exposés à des points de vue différents et de moins dépendre des adultes quand ils rencontrent un problème. De plus, la programmation à deux brise les stéréotypes quant aux emplois liés au développement, qui sont perçus comme « antisociaux ». Cette stratégie peut donc servir à faire la promotion de la diversité dans le domaine.

2. Étayage

L'étayage sert à rendre les élèves plus autonomes au fil du processus d'apprentissage. Au début, l'enseignante enseigne à l'élève. Ensuite, il enseigne avec la participation de l'élève. Finalement, l'élève apprend de manière autonome (par et pour lui-même).

La structure recommandée de nos leçons suit ce modèle d'étayage. Par exemple, durant l'introduction, l'enseignante montre le projet final aux élèves (enseignement à l'élève). Durant la programmation en groupe, l'enseignante montre les outils et leur fonctionnement et collabore avec les élèves pour réaliser des exercices (enseignement avec la participation de l'élève). Finalement, durant la séance de travail, les élèves travaillent individuellement (apprentissage par et pour eux-mêmes). Pendant la leçon avec l'enseignante, ils ont acquis les compétences nécessaires pour ajouter des éléments ou essayer de nouvelles choses eux-mêmes.

3. Méthode à l'envers

La méthode à l'envers est une stratégie d'enseignement pratique pour amener les élèves à passer du « quoi » au « comment » et au « pourquoi ». Par exemple, nous voulons enseigner aux élèves que le concept des boucles (p. ex. les blocs « répéter indéfiniment » et « répéter _ fois ») ne s'arrête pas à la répétition de commandes. Notre but est de leur faire différencier les deux blocs afin qu'ils comprennent pourquoi un bloc convient mieux à une situation que l'autre.

Les tutoriels de programmation nous montrent comment programmer un programme étape par étape, du début jusqu'à la fin. En suivant la méthode à l'envers, nous commençons par des blocs au milieu du code et posons des questions aux élèves pour qu'ils déterminent les étapes à suivre pour compléter le code. Ainsi, les élèves ne suivent pas des consignes sans comprendre ce qu'ils font et découvrent l'importance des séquences, un concept fondamental en programmation. Ils ont l'occasion de faire des erreurs dans l'ordre des blocs, de résoudre des problèmes, de faire appel à leur créativité et de prendre des décisions pour comprendre le fonctionnement du programme (le « comment » et le « pourquoi »).

Stratégies d'enseignement

4. Exploration par les élèves

Notre but est de stimuler la curiosité des élèves. Nous voulons qu'ils brisent des choses et essaient de nouvelles méthodes pour voir si elles fonctionnent. Nous cherchons à leur faire accepter les erreurs et les changements de cap. Pour engager les élèves dans le processus d'apprentissage, posez-leur des questions et demandez-leur des solutions et des conseils pendant l'atelier. (Suivez-les, même si vous savez qu'il ne s'agit pas d'une bonne solution!)

Si un élève ne sait pas quoi faire, demandez-lui de travailler avec une personne assise à côté de lui ou à sa table. S'il doute, encouragez-le à essayer des choses! Après tout, s'il y a des problèmes, on peut toujours appuyer sur le bouton « Défaire ».

5. Enseignement en équipe

Comme le nom le dit, l'enseignement en équipe est un type d'enseignement mené par au moins deux personnes collaborant pour aider les élèves à réaliser les activités. Pendant une séance animée par deux personnes, vous pourriez enseigner à deux ou désigner une personne qui enseignera tandis que l'autre fera du mentorat.

Peu importe la configuration, l'enseignement en équipe permet aux élèves de découvrir différentes façons de résoudre des problèmes et d'obtenir du soutien personnalisé plus fréquemment. L'enseignement en équipe est recommandé aux personnes qui commencent tout juste à enseigner et qui souhaitent avoir le soutien d'un collègue!

6. Coanimation et mentorat par les pairs

Parfois, les personnes qui enseignent le mieux ne sont pas des spécialistes des technologies, mais des gens qui ont une petite longueur d'avance sur les élèves! Elles ont conscience des obstacles potentiels et peuvent aider les élèves, car elles ont elles-mêmes rencontré ces obstacles récemment.

Si vous avez des élèves avancés, demandez-leur s'ils souhaiteraient faire du mentorat auprès d'autres élèves ou coanimer pendant une partie de l'atelier. Le mentorat et la coanimation ont une influence positive sur plusieurs personnes dans un groupe. Les élèves avancés peuvent développer leurs compétences en direction et renforcer leur compréhension des concepts de base. Si vos élèves ont des niveaux d'expérience variés, le mentorat par les pairs contribue à mettre le groupe sur un même pied d'égalité et vous enlève un poids sur les épaules quant au soutien.

7. Autonomie

Autant que possible, nous vous conseillons de ne pas toucher aux appareils des élèves. Plutôt que de prendre leurs appareils pour résoudre des problèmes à leur place, amenez-les à résoudre des problèmes par eux-mêmes. Cela ne veut pas dire que vous ne devez pas aider les élèves! Le but est de ne pas résoudre des problèmes que les élèves pourraient résoudre de manière autonome. Si les élèves éprouvent beaucoup de difficulté, vous pouvez les orienter verbalement vers la solution sans toucher vous-même à l'écran, au clavier, à la souris ou au pavé tactile. Ainsi, même si vous les avez aidés, ils auront fait la démarche par eux-mêmes!

Gestion de classe avec la technologie

Si vous avez déjà vu des jeunes utiliser la technologie, vous savez qu'il peut être difficile de capter leur attention quand ils ont les yeux rivés sur un écran!

1. Code des programmeuses et programmeurs

Le Code des programmeuses et programmeurs est une entente collaborative décrivant les comportements attendus de toutes les personnes qui participent à une leçon.

Il est conseillé de créer votre propre code en collaboration avec les élèves au début de l'atelier. Demandez aux élèves de communiquer leurs besoins afin qu'ils puissent s'amuser et apprendre dans un environnement agréable. De quoi ont-ils besoin? De quoi avez-vous besoin?

Exemple de code

Nous allons...

- Écouter les personnes qui ont le droit de parole;
- Faire preuve de patience et de gentillesse envers les autres;
- Poser des questions si nous avons des problèmes;
- Soutenir les gens qui ont des problèmes;
- Faire de notre mieux pour faire chaque tâche!

En grand groupe, déterminez quatre ou cinq attentes qui constitueront le code. Écrivez-les à un endroit visible par tous (p. ex. sur un grand carton, un tableau blanc, etc.) afin de vous y référer en cas de problème de comportement (en gros, si quelqu'un ne répond pas aux attentes établies par le groupe). Le code peut être modifié à votre guise et selon les besoins des élèves pendant l'atelier!

2. Mode « politesse »[9]

Pour empêcher les élèves d'être distraits par leur écran pendant que vous leur parlez, pensez à utiliser le mode « politesse ». En mode « politesse », les ordinateurs portables sont à moitié fermés et les autres écrans et les moniteurs des ordinateurs de bureau sont éteints pour empêcher les élèves de voir l'écran. Cette stratégie est idéale à appliquer quand vous expliquez de nouvelles étapes à l'avant de la classe. Les élèves allumeront leur écran ou ouvriront leur ordinateur portable quand ils réaliseront les étapes à leur tour!

3. Transparence et temps

Créez un environnement prévisible en écrivant l'horaire de chaque séance (y compris les pauses) à un endroit visible. Vous pouvez aussi utiliser un chronomètre pendant les périodes de travail individuel afin que les élèves puissent gérer leur temps. Nous vous conseillons l'outil **timer-tab.com**, un chronomètre numérique à afficher sur l'écran ou le projecteur de la classe pour que le groupe s'y réfère![10]

4. Appel et réponse

Si vous avez perdu l'attention de votre groupe, ne montez pas la voix. Utilisez plutôt la technique d'appel et de réponse. Dites le début d'une expression connue et laissez le groupe la compléter!

Exemples d'appel et réponse :

Enseignante : « Vers l'infini... ». Élèves : « Et plus loin encore! »

Enseignante : « On arrête tout! ». Élèves : « Je collabore et j'écoute! »

Enseignante : « Si vous m'entendez... »

- « Saluez-moi de la main! »
- « Faites des oreilles de lapin à votre partenaire! »
- « Levez la main gauche, la main droite... et tapez des mains! »
- « Claquez des doigts une fois. Claquez des doigts deux fois. Claquez des doigts dix fois! »

Quand vous travaillez avec la technologie, il est avantageux d'avoir des objets à manipuler pour empêcher les élèves de toucher au clavier pendant que vous enseignez quelque chose d'important.

5. Leçons animées par les élèves

Évitez les cours magistraux où vous ne faites que parler aux élèves. Posez-leur des questions, laissez-les vous orienter pendant les activités et invitez-les à vous donner des solutions!

Exemples :

- **Réfléchir, échanger, présenter**
  Quand vous voulez obtenir des rétroactions des élèves, utilisez la stratégie « Réfléchir, échanger, présenter » pour leur donner l'occasion de s'exprimer. D'abord, accordez-leur 30 secondes pour réfléchir à leur réponse. Ensuite, dites-leur de prendre 1 ou 2 minutes pour échanger avec 2 ou 3 personnes près d'eux. Finalement, demandez à quelques personnes de présenter les idées soulevées pendant la discussion en petits groupes. En faisant cela, les élèves auront moins peur de s'exprimer devant la classe, car ils auront déjà réfléchi à leurs réponses et discuté de leurs idées.

- **Vote**
  Devez-vous prendre une décision? Un truc : les élèves peuvent voter en levant ou en baissant le pouce!

- **Présentation**
  Ask learners to come to the front and demonstrate their solutions for the class using the educator's computer. Did another learner approach the same problem in a different way? Ask them to show the class their approach! Did someone figure out something really cool? Have them share with the group!

6. Modélisation des bons comportements

Certains élèves pourraient ressentir de la frustration si leur ordinateur ne fonctionne pas comme ils le souhaitent. Pour les aider à gérer leurs émotions, modélisez des comportements à adopter en cas de problème. Montrez-leur à réagir positivement aux problèmes techniques et à verbaliser leurs pensées pendant le débogage. Consultez la section « Stratégies de débogage » pour en savoir plus.

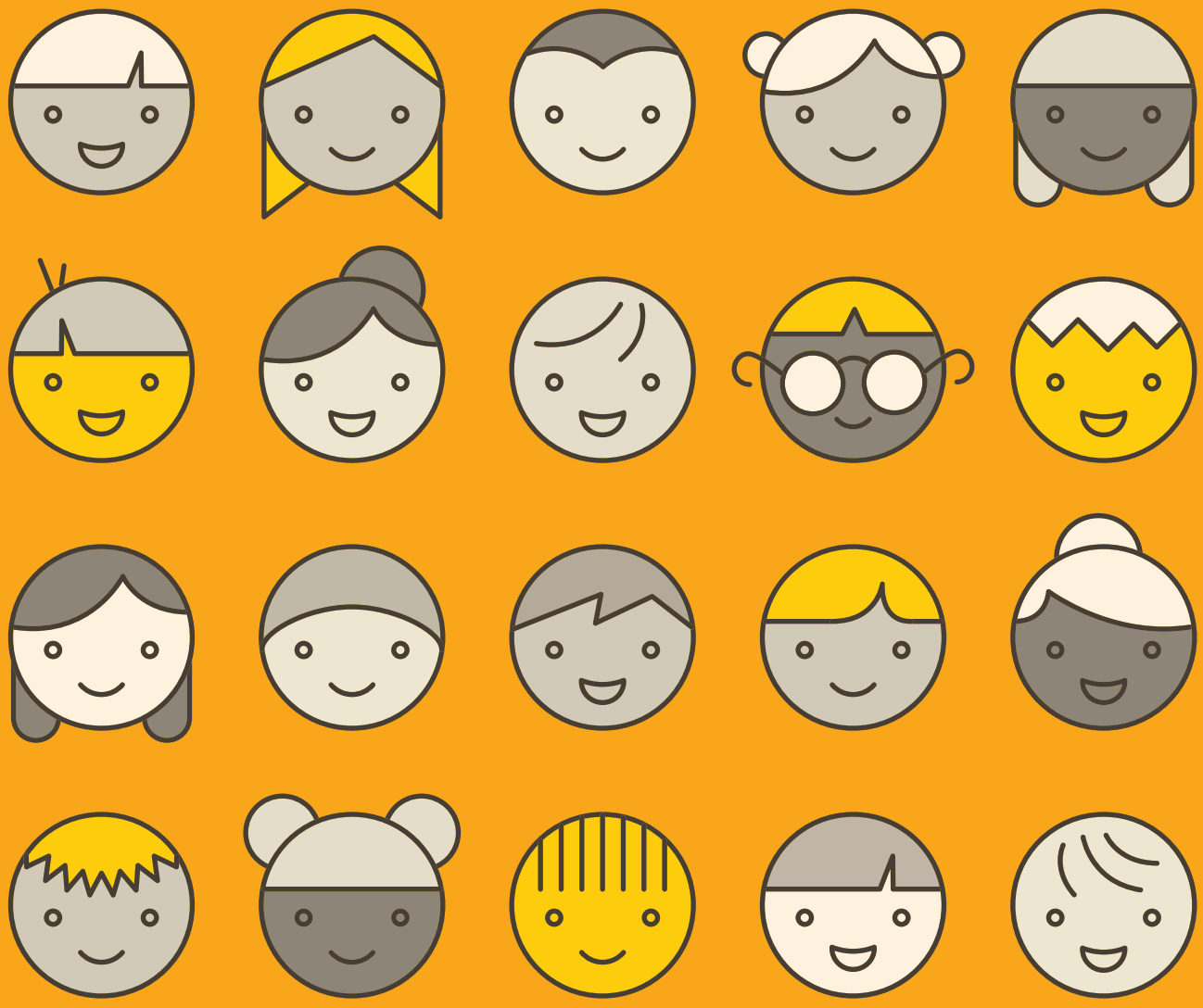
7. Exploration par les élèves

Les activités de programmation en groupe de chaque atelier comprennent des concepts de programmation et des exercices bien définis. Par contre, les séances de travail sont le moment parfait pour laisser les élèves explorer par eux-mêmes et concrétiser leurs idées. Les élèves seront plus motivés s'ils ont la liberté de remixer leurs projets. Consultez la section « Stratégies d'enseignement » pour en savoir plus sur l'exploration par les élèves.

Ressources

1. « Exact Instructions Challenge PB&J Classroom Friendly | Josh Darnit. » YouTube, vidéo téléversée par Josh Darnit, 18 avril 2017, https://www.youtube.com/watch?v=FN2RM-CHkul. Repéré le 11 mars 2020.
2. Gustafson, Ingrid. « Overheard In the Classroom. » ScratchEd, 20 juillet 2016, http://scratched.gse.harvard.edu/resources/overheard-classroom-ingrid-gustafson. Repéré le 10 mars 2020.
3. Roach, Emily. « Celebrating Mistakes and Embracing Process. » ScratchEd, 3 mars 2017, http://scratched.gse.harvard.edu/stories/see-inside-classroom-emily-roach. Repéré le 10 mars 2020.
4. ScratchEd Team. « Debugging in Scratch: Resources and Strategies. » ScratchEd, 18 avril 2017, https://scratched.gse.harvard.edu/resources/debugging-scratch-resources-and-strategies.html. Repéré le 10 mars 2020.
5. Tanguay-Carel, Matt. « Frustrations of Programming & How to Avoid Them. » Codementor, 30 novembre 2016, https://www.codementor.io/@matstc/avoid-frustration-as-programmers-ge54ddszr. Repéré le 17 juillet 2017.
6. Goel, Sanjay, et Vanshi Kathuria. « A Novel Approach for Collaborative Pair Programming Executive Summary. » Journal of Information Technology Education, vol. 9, 2010.
7. Lye, Sze Yee, et Joyce Hwee Ling Koh. « Review on Teaching and Learning of Computational Thinking through Programming: What Is next for K-12? » Computers in Human Behavior, vol. 41, 2014, p. 51–61, doi:10.1016/j.chb.2014.09.012.
8. McDowell, Charlie, et al. « Pair Programming Improves Student Retention, Confidence, and Program Quality. » Communications of the ACM, vol. 49, août 2006, p. 90–95, doi:10.1145/1145287.1145293.
9. Wilson, Greg. Data Scientist and Professional Educator at RStudio, Inc. LinkedIn. https://www.linkedin.com/in/greg-wilson-a26510b6/?originalSubdomain=ca. Repéré 13 mars 2020.
10. Brillout, Romuald. Timer Tab. https://www.timer-tab.com/. Repéré le 7 mars 2020.

DATE:



COMMUNIQUE AVEC NOUS

30 St Patrick St
Toronto, ON M5T 3A3

info@canadalearningcode.ca
canadalearningcode.ca

※enseignants en programmation