La pensée computationnelle

Cadre pédagogique

2018

let's talk science | parlons sciences

Remerciements

Fondé sur des études, de la documentation et des idées provenant du monde entier, le présent cadre pédagogique était considéré comme étant à jour en date du mois d'octobre 2018. À mesure que la réflexion sur l'enseignement et l'apprentissage de la pensée computationnelle évoluera, on y apportera les ajustements requis afin qu'il continue de fournir de l'information pertinente aux éducateurs.

Parlons sciences aimerait remercier Mme Lisa Floyd, directrice de la recherche et de l'investigation, et M. Derek Tangredi, directeur de l'éducation en STIAM intégrée, tous deux de Fair Chance Learning, d'avoir partagé leurs connaissances et leur expérience dans le domaine de la pensée computationnelle. Leur apport a été grandement apprécié. Parlons sciences aimerait aussi remercier Mme Kim Taylor, spécialiste en éducation, pour l'élaboration du présent cadre, de même que tous les membres du personnel qui ont mis leur expertise à profit pour en effectuer la relecture. Pour obtenir plus de renseignements sur Parlons sciences, n'hésitez pas à vous rendre à l'adresse suivante : parlonssciences.ca

Parlons sciences
1510 Woodcock Street, Unité 12
London, Ontario, Canada
N6H 5S1

www.parlonssciences.ca

1.1 La pensée computationnelle: introduction

Presque toutes les sphères de nos vies personnelles et professionnelles sont dorénavant imprégnées de technologie numérique. Les plus jeunes d'entre nous n'ont même jamais connu un monde sans ordinateurs ni téléphones intelligents. Mais, bien qu'ils soient de grands consommateurs de technologie, on se demande de plus en plus s'ils sont bien préparés à en devenir des créateurs. Les chercheurs Kafai et Margolis (Washington Post en ligne, 2014 [traduction]) décrivent ainsi la situation :

> « Aujourd'hui, être natif de l'ère du numérique ne se résume pas à naviguer sur le Web, à utiliser de la technologie pour communiquer ou à faire partie de réseaux de jeux en ligne. Cela implique en effet de savoir comment les choses sont faites, de décomposer et de résoudre des problèmes, de concevoir des systèmes, de fournir un apport créatif et de comprendre les ramifications sociales et éthiques de nos gestes. Les ordinateurs sous toutes leurs formes font dorénavant partie intégrante de notre vie collective; ils nous permettent non seulement d'interagir avec le monde, mais aussi de nous y sentir utiles. »

À mesure que la technologie avancera, il deviendra de plus en plus important de préparer les jeunes à s'investir dans leur environnement numérique. Cela implique notamment de les rendre aptes à interagir de façon constructive avec autrui, à intégrer des milieux de travail où les ordinateurs jouent un rôle de plus en plus prépondérant, à décider comment la technologie pourrait façonner la société et à trouver des solutions à des problèmes auxquels nous sommes tous confrontés.

Les enseignants auront besoin de soutien pour déterminer ce qui doit être transmis aux jeunes afin qu'ils puissent jouer un rôle actif au sein d'un monde numérique en constante évolution. Selon nous, pour aujourd'hui comme pour demain, ils devront apprendre à penser « informatiquement » en acquérant les compétences, les connaissances et les habitudes mentales associées à la pensée computationnelle (PC).

Le terme « pensée computationnelle » commence à faire son apparition au sein des systèmes d'éducation du monde entier, y compris celui du Canada, de la maternelle à la fin du secondaire. Comme il s'agit d'une nouvelle notion, il importe que les enseignants en comprennent le sens avant de pouvoir en favoriser l'apprentissage par les élèves. C'est pourquoi nous nous sommes penchés sur de nombreux documents provenant d'un bout à l'autre de la planète (programmes scolaires, littérature didactique, ressources à l'intention d'éducateurs, sites Web d'organismes offrant des programmes en PC, etc.) pour y trouver des définitions et des explications.

> Un milieu où la PC s'épanouit en est un où les utilisateurs peuvent développer les compétences requises pour devenir des créateurs.
> (Lee et coll., 2011, p. 35 [traduction])

Pour obtenir de l'information, lire des définitions et connaître les sources citées, veuillez consulter le document (anglais seulement) Let's Talk Science Computational Thinking Literature and Curriculum Review (Parlons sciences, 2018).

1.2 Où la pensée computationnelle s'insère-t-elle?

À maints égards, la PC s'insère dans ce qu'on appelle la culture ou **littératie numérique** (LN). Selon le Conseil des technologies de l'information et des communications (CTIC) la LN se définit par « la capacité de repérer, de classer, de comprendre, d'évaluer et de générer de l'information en utilisant la technologie numérique pour établir une société fondée sur le savoir » (2012, p. 2). Le Brookfield Institute for Innovation + Entrepreneurship la caractérise quant à lui comme « la capacité, fondée sur une habileté à comprendre de manière critique le contenu et les instruments numériques, d'utiliser des outils technologiques pour résoudre des problèmes. Cela peut aussi inclure l'aptitude plus avancée de créer de nouveaux outils, produits et services technologiques » (2017, p. 11 [traduction]). Il importe de noter que certains aspects de la PC peuvent être développés sans la technologie numérique; c'est pourquoi les losanges de la LN et de la PC à la figure 1 ne se chevauchent pas complètement.

Les **études informatiques (ÉI)** sont moins « axées sur l'utilisation d'ordinateurs » et « vont au-delà de simples notions de programmation » (ministère de l'Éducation de l'Ontario, 2008, p. 3 [traduction]). Elles se situent là où la LN et la PC se chevauchent, comme on peut le voir à la figure 1. La programmation, parfois désignée par le terme « codage », s'inscrit parfaitement dans la LN, les ÉI et la PC. Elle consiste à dire quoi faire aux ordinateurs. Dans la plupart des cas, les gens utilisent un langage particulier qui définit la manière d'écrire le code pour que les appareils le comprennent. La programmation peut aider les élèves à développer de nombreux aspects de la PC.

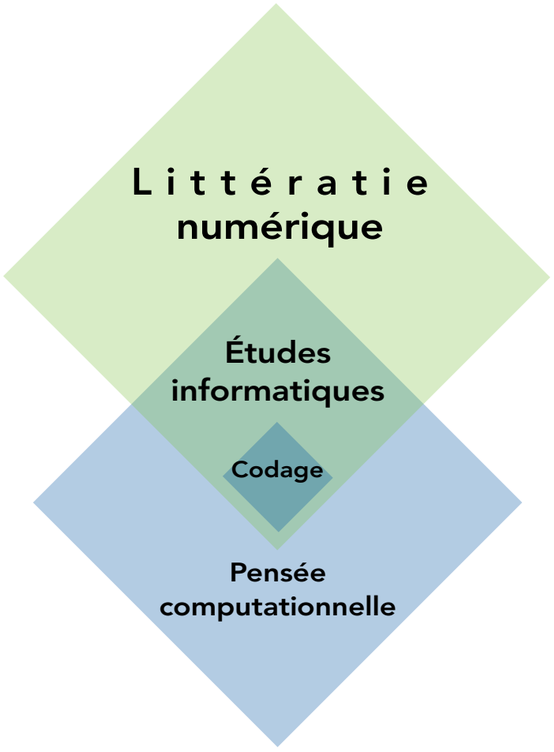


Figure 1 : *Intersections de la pensée computationnelle (PC) avec d'autres domaines, y compris la littératie numérique (LN) et les études informatiques (ÉI).*

1.3 Qu'est-ce que la pensée computationnelle?

Il n'y a actuellement pas de définition universellement acceptée pour la PC. Celles qui nous sont données sont très semblables, mais expliquées de manières différentes selon les intérêts et opinions des chercheurs, des éducateurs et des organisations (en sciences informatiques, en sciences tout court, en mathématiques, etc.) qui les ont formulées.

Dans de nombreux cas, au lieu de définir franchement la PC, on l'explique du point de vue des connaissances et des compétences que les élèves doivent acquérir pour la développer. Certains résultats d'apprentissage sont conceptuels (la notion d'algorithmes, par exemple), certains sont axés sur des compétences (le raisonnement algorithmique, par exemple). Certaines sources décrivent également des activités d'apprentissage fondées sur la PC qu'elles désignent par des termes comme « stratégies », « méthodes », « approches », « processus » ou « pratiques ».

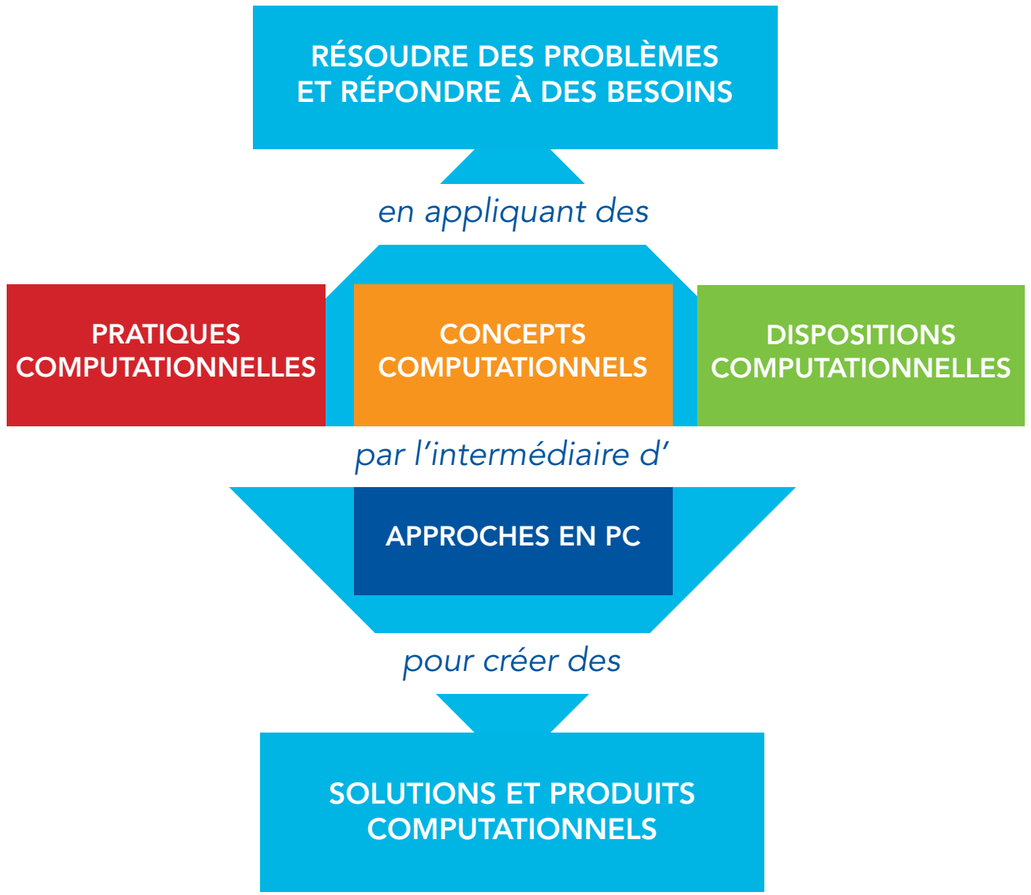
Bien qu'il semble y avoir peu de consensus quant à la terminologie, les chercheurs ont toutefois exprimé des idées récurrentes.

La pensée computationnelle…

implique des processus intellectuels comme le raisonnement logique.	est associée de manière non exclusive aux processus de détermination, de compréhension et de résolution de problèmes.	prend appui sur les outils, techniques et concepts informatiques (c.-à-d. la décomposition, l'abstraction et le raisonnement algorithmique).
implique une structuration systématique et logique de procédures (algorithmes) pour générer des solutions automatisables (programmables).	permet la création de solutions qui peuvent être efficacement réalisées par un appareil de traitement de l'information, comme un ordinateur par exemple.	implique souvent la collecte, l'organisation (le tri, le groupement) et l'analyse de données (recherche de régularités, de dépendances et de relations).
peut mener à la création de produits, de processus et de systèmes numériques.	permet de développer et de maintenir des compétences en raisonnement de plus haut niveau, comme l'analyse, la synthèse et l'évaluation.	favorise le développement d'habiletés et de compétences globales/du 21e siècle comme la pensée créative, la pensée critique, la collaboration et la communication.

1.4 Le Cadre pédagogique en pensée computationnelle de Parlons sciences

Le ***Cadre pédagogique en pensée computationnelle*** de Parlons sciences présente une synthèse des nombreuses définitions et explications proposées à ce jour. Son élaboration s'appuie sur des travaux effectués par la Computer Science Teachers Association (É.-U.), l'International Society for Technology in Education, Computing at School (R.-U.), le Brookfield Institute (Canada), Karen Brennan et Mitchell Resnick (MIT Media Lab), ainsi que Peter J. Denning. Il est conçu pour illustrer comment les diverses facettes de la PC permettent aux élèves d'acquérir les compétences, les connaissances et les habitudes mentales dont ils ont besoin pour créer des solutions et résoudre des problèmes au sein d'un monde numérique.



Ce qui suit donne plus de détails sur les attentes de chaque catégorie à observer chez les élèves alors qu'ils développent la pensée computationnelle.

Pratiques computationnelles (Compétences)

- *Décomposition*
- *Abstraction*
- *Reconnaissance de régularités*
- *Raisonnement algorithmique*
- *Test et évaluation*
- *Pensée logique (implique le raisonnement)*
- *Débogage*
- *Collecte et analyse de données*
- *Représentation de données*

Les définitions se trouvent à la section 1.5

Concepts computationnels (Connaissances)

- *Séquences*
- *Répétition (boucles)*
- *Conditionnelles*
- *Données*
- *Variables*
- *Événements*
- *Opérateurs*
- *Fonctions*
- *Entrées et sorties*

Les définitions se trouvent à la section 1.6

Dispositions computationnelles (Aptitudes mentales)

- *Persévérance*
- *Être à l'aise à travailler en équipe*
- *Être à l'aise avec l'essai et l'erreur*
- *Flexibilité mentale*
- *Créativité*
- *Tolérerance à l'ambiguité*
- *Habilité à gérer des problèmes ouverts*
- *Confiance à gérer la complexité*
- *Curiosité*

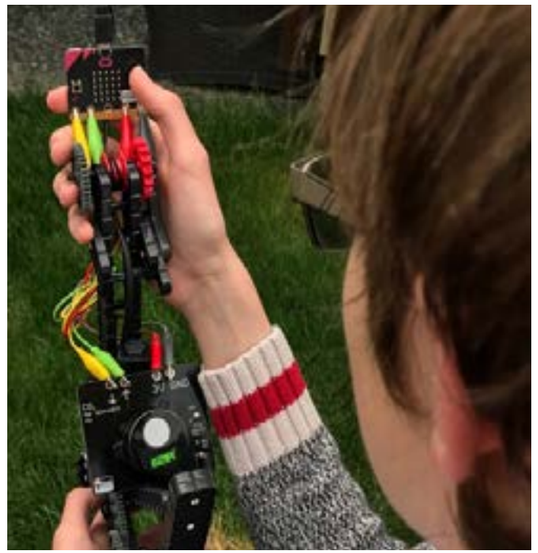
Approches computationnelles

- *Débranché (sans appareils informatiques)*
- *Apprentissage exploratoire*
- *Réutilisation et remixage*
- *Fabrication*

Les définitions se trouvent à la section 1.7

Exemples de produits et solutions computationnels

- *Objets physiques contrôlés numériquement*
- *Applications mobiles et sites web*
- *Modèles et simulations informatiques*
- *Algorithmes et programmes informatiques*
- *Modélisations, visualisations et structures de données*



1.5 Définitions : Pratiques en pensée computationnelle

Décomposition

Définition :	Exemples :	Compétences liées :
La **décomposition** consiste à diviser un problème en parties ou sous-problèmes. Travailler avec une portion d'un problème sert à réduire la complexité globale d'un problème. La décomposition peut aussi comprendre la compréhension d'un produit informatique (logiciel, site web, etc.) en termes de ces composantes (ex. : graphiques, données, interface, etc.). La pensée computationnelle implique non seulement la décomposition d'un problème, mais aussi l'application des connaissances acquises lors de la résolution de problèmes antérieurs et l'identification d'une solution au problème après que tous les sous-problèmes aient été réglés.	• *Identifier toutes les étapes nécessaires afin de faire un gâteau.* • *Expliquer comment les parties du corps d'une libellule lui procurent un avantage afin d'être un prédateur efficace.* • *Créer un jeu vidéo avec vos amis. Comment pourriez-vous diviser les tâches?* • *Créer un programme qui convertira une valeur de température en degrés Fahrenheit, en degrés Celsius. Quelles sont les étapes et processus importants?*	• Décomposer • Diviser • Trier • Classifier

Abstraction

Définition :	Exemples :	Compétences liées :
L'**abstraction** consiste en la simplification d'un problème ou d'une tâche en mettant l'accent sur ce qui est important, en extrayant les informations pertinentes et ignorant les détails superflus. L'abstraction est aussi utilisée lorsqu'un item est sélectionné pour représenter un groupe d'item, ou lorsqu'un mot représente une action (qui elle-même comporte plusieurs étapes). Les modèles et simulations peuvent également être considérés comme des abstractions.	• *Identifier les compétences les plus importantes pour un gardien de but de hockey.* • *Représenter les actions courir, arrêter et sauter avec des images.* • *Coder une simulation d'un volcan en éruption.* • *Identifier la formule appropriée à utiliser pour résoudre un problème de physique.*	• Simplifier • Recentrer • Généraliser • Trier

Observation de régularités

Définition :	Exemples :	Compétences liées :
L'**observation de régularités** implique la capacité de reconnaître, identifier et utiliser des régularités pour décrire et représenter les séquences dans des données ou processus. En identifiant ces régularités, il est possible de prédire comment fonctionneront certains éléments ou ce qui peut arriver sous certaines conditions. L'observation de régularités permet la création de règles, comme lorsque des actions peuvent être répétées automatiquement. L'observation de régularités permet aussi d'appliquer une méthode reconnue comme précédemment valable et de l'appliquer à une nouvelle problématique, ce qui améliore grandement l'efficacité du processus de résolution de problèmes.	• *Quelle régularité remarquez-vous lorsque vous dessinez un carré? Comment pourriez-vous appliquer cette connaissance au dessin d'un pentagone?* • *Quelles notes sont répétées dans la musique? Quand est-ce que ces parties sont répétées?* • *Au baseball, qu'arrive-t-il après deux prises? Et après trois prises?*	• Observer • Prédire • Comparer • Généraliser

Raisonnement algorithmique

Définition :	Exemples :	Compétences liées :
Le **raisonnement algorithmique** est la compétence nécessaire à la création d'un algorithme. Un algorithme est une série de règles ou instructions ordonnées, logiques et précises nécessaires à la résolution d'un problème ou à l'atteinte d'un objectif déterminé. En identifiant les étapes qui peuvent être transmises comme instructions (verbales ou écrites), ou codes ou programmes à d'autres personnes ou appareils informatiques, le raisonnement algorithmique est la compétence utilisée.	• *Quelles étapes sont nécessaires afin de faire votre sandwich idéal? Est-ce que l'ordre des étapes est important? Pourquoi ou pourquoi pas?* • *Créer un algorithme qui pourrait vous aider à décider ce que vous porterez comme vêtements chaque jour.* • *Créer un algorithme qui peut vous aider à identifier les oiseaux que vous pouvez voir dans votre quartier.* • *Créer un algorithme pour enseigner à un jeune élève comment faire une boucle avec ces lacets de chaussures.*	• Planifier • Organiser • Ordonner • Classifier • Trier

Raisonnement logique

Définition :	Exemples :	Compétences liées :
Le **raisonnement logique** consiste à penser de façon logique. Ce raisonnement implique de débuter avec une prémisse (information que l'on sait vraie), puis d'utiliser sa logique pour terminer avec une conclusion qui est cohérente avec cette prémisse. Nous appliquons le raisonnement logique lorsque nous utilisons l'abstraction, le raisonnement algorithmique et l'observation de régularités.	• ***Raisonnement déductif :*** lorsqu'on déduit, on débute avec une prémisse précise qui nous conduit à un conclusion exacte et précise(ex. : Tous les carrés sont des rectangles. Tous les rectangles ont quatre côtés; donc, tous les carrés ont quatre côtés.). • ***Raisonnement inductif :*** lorsqu'on induit, on utilise une prémisse précise qui conduit à une conclusion générale, parfois même imprécise (ex. : Les élèves de ma classe 6e année aiment l'éducation physique; donc, tous les élèves de 6e année aiment l'éducation physique.).	• Classifier • Analyser • Généraliser

Tests et évaluation

Définition :	Exemples :	Compétences liées :
Tester implique l'essai de quelque chose et l'observation du résultat. **L'évaluation** implique l'utilisation de la pensée critique et du bon sens afin de déterminer si une liste de critères sont respectés. Si ce n'est pas le cas, alors des corrections ou améliorations seront peut-être nécessaires. La programmation informatique est bien souvent un procédé interactif pendant lequel la phase de test et d'évaluation est utilisée de façon récurrente.	• *Est-ce que le personnage animé bouge de la façon prévue et espérée?* • *Qu'arrive-t-il lorsque vous testez le robot? Était-il capable d'atteindre le panier à chaque lancer?* • *Qu'arrive-t-il lorsque vous téléchargez le code pour faire défiler votre nom sur l'écran du micro:bit? Est-ce que les lettres de votre nom défilent dans le bon ordre?*	• Observer • Comparer • Analyser

Débogage

Définition :	Exemples :	Compétences liées :
Quand un programme informatique ne fait pas ce que vous pensiez qu'il ferait, il pourrait y avoir des erreurs. Les erreurs dans un ou des programmes informatiques sont appelées « bogues ». **Déboguer** un problème consiste à identifier et corriger les bogues trouvés. Les bogues peuvent inclure des erreurs de syntaxe ou d'orthographe, les erreurs de logique et autres types d'erreurs. Le débogage nécessite le raisonnement logique.	• *Pourquoi est-ce que l'ordinateur indique que le code ne peut se compiler, est-ce dû à une faute d'orthographe?* • *Pourquoi est-ce que votre programme vous donne un résultat aussi éloigné de ce qui était prévu, est-ce dû à une erreur de logique?* • *Quels tests simples pourriez-vous effectuer afin de vous aider à trouver le bogue dans ce programme?*	• Observer • Tester • Analyser • Interpréter

Collecte et analyse de données

Définition :	Exemples :	Compétences liées :
Collecte de données : méthode planifiée qui consiste à sélectionner et collecter l'information nécessaire à la résolution d'un problème ou pour répondre à une question précise. Lors qu'on définit les méthodes de consignation et d'organisation des données, il est important de considérer les types de données collectées (ex.: qualitatives, quantitatives) afin de définir la meilleure façon de les documenter et organiser, en considérant l'objectif que les données doivent être utilisées pour l'analyse et l'interprétation. **L'analyse de données** implique l'observation des régularités pouvant en être extraites, telles des résultats répétitifs, des tendances à la baisse ou la hausse, etc. L'analyse inclut aussi l'exercice d'expliquer les régularités et tendances, autant que les écarts ou données irrégulières.	• *La nuit, combien d'animaux traversent la route à cet endroit? Quelles données devrait-on collecter? Comment collecter ces données?* • *Qu'observez-vous sur la germiantion des semences au cours des deux dernières semaines?* • *Quelle est l'espèce d'oiseau la plus commune venant se nourrir à la mangeoire? Quelles données devrait-on collecter? Comment collecter ces données?*	• Observer • Comparer • Analyser • Interpréter • Tirer des conclusions

Représentation des données

Définition :	Exemples :	Compétences liées :
Une fois que les données sont collectées, elles sont souvent conservées dans une base de données informatique. Parfois, les utilisateurs voudront analyser toutes les données, parfois ils ne seront concernés que par une portion ou sous-groupe des données. Afin de pouvoir adéquatement analyser et communiquer ces résultats, il est souvent pratique de **représenter les données** de façon plus visuelle et conviviale, tel que dans un graphique, un tableau, une infographie, etc.	• *Quelles villes du Canada ont le plus haut taux de CO_2 dans leurs classes? Quelle est la meilleure façon de représenter ces données?* • *Comment organiseriez-vous les données sur les précipitations quotidiennes des grandes villes afin d'aider les chercheurs à comparer les futures données, que ce soit entre différentes périodes ou différents endroits?* • *Quelle est la méthode la plus utile pour représenter mes notes en mathématiques afin de m'aider à analyser mon progrès?*	• Organiser • Trier • Interpréter

1.6 Définitions : Concepts en pensée computationnelle

Séquences

Définition :	Exemples :
Les étapes d'un algorithme se produisent toujours selon une séquence déterminée. Il faut toutefois noter qu'une séquence est rarement linéaire et qu'elle peut impliquer des tâches répétitives ou s'exécuter seulement sous certaines conditions. Un diagramme de séquence est un bon outil pour comprendre et représenter les séquences. Les séquences simples et linéaires font partie des permiers concepts à apprendre en programmation.	• *Quelles sont les étapes nécessaires à ……? (ex. : changer un pneu, brosser ses dents, se faire un sandwich). Y a-t-il des étapes que vous répétez?* • *Déterminer les règles d'un jeu (ex. : qu'arrive-t-il au tout début, ensuite, pour finir une partie?).* • *Décrire le trajet pour se rendre à l'école à l'aide d'une carte routière (ex. : où débuter, tourner? Où sont les points de départ, d'arrivée?).* • *Expliquer à quelqu'un comment dessiner un carré avec un périmètre déterminé.*

Répétitions (boucles)

Définition :	Exemples :
La **répétition** dans un algorithme consiste à répéter une étape un certain nombre de fois jusqu'à ce que le point de terminaison soit atteint. Les tâches répétitives sont très communes en programmation informatique et programmer des actions afin qu'elles se répètent automatiquement (boucle) peut épargner bien du temps. Les boucles aident à mieux organiser et abréger les programmes en diminuant la quantité de code qui doit être écrite.	• *Expliquez à quelqu'un d'autre la façon la plus efficace de faire 100 sandwiches.* • *Programmer une lampe afin q'elle s'allume et s'éteigne à des moments précis de la journées.* • *Programmer un jeu de voitures pour que la course s'arrête automatiquement après 10 tours de piste.* • *Créer un oeuvre d'art qui implique une répétition, comme une mosaïque, une frise, un mandala ou une figure fractale.*

Expressions conditionnelles

Définition :	Exemples :
Parfois, dans un algorithme, il est nécessaire de sélectionner une action parmi une liste de possibilités, de façon similaire à la nécessité de prendre une décision quand à la voie à emprunter lorsqu'on arrivons à un embranchement de la route. Les expressions conditionnelles donnent des règles précises pour contrôler la séquence des actions, tel... **si** quelque chose est vrai, **alors** quelques chose se produit ou **sinon** quelque chose d'autre se produit. Les expressions conditionnelles permettent à un programme de prendre des décisions et contrôler la séquence d'actions, sans l'intervention humaine.	• *S'il pleut à l'extérieur, vous aurez alors besoin de porter des bottes de pluie, sinon vous pouvez mettre vos chaussures sport.* • *Si le capteur indique une température supérieure à 25°C, alors afficher un icône sourire, sinon le capteur indiquant une température inférieure à 25°C, alors afficher un icône triste.* • *Si vous arrivez à une intersection dans le labyrinthe, alors tournez à gauche, sinon continuez en ligne droite.* • *Si (heure < 18:00) {salutation = "Bonjour";} sinon {salutation = "Bonsoir":}.*

Données

Définition :	Exemples :
Les ordinateurs et autres systèmes de communication fonctionnent avec différents types de données. Les **données** sont des valeurs que les ordinateurs peuvent conserver et récupérer. **Types de données** : les différents types de données sont propres aux différents langages informatiques, mais comprennent normalement des types tels: les caractères (lettres, ponctuation, espaces), chaînes (séquences de caractères), nombres entiers, nombres à virgule fixe ou flottante et données de type booléen, souvent utilisées dans les expressions conditionnelles.	• ***Caractère*** (ex.: A); **Chaîne** (e.g., Allo!); **Nombre entier (**ex.: 12). • ***Nombre à virgule flottante*** (ex.: 15,2203829) - peut avoir un nombre variable de chiffres après la virgule décimale. • **Condition booléenne** *(valeur vraie ou fausse seulement) - peut être interprétée comme activé/désactivé ou actif/inactif.* • *Si vous voulez suivre le nombre de buts qui ont été comptés, quel type de données utiliseriez-vous? (nombre entier).* • *Si vous voulez conservez l'information sur l'état d'une lampe (allumée ou éteinte), quel type de données utiliseriez-vous? (de type booléen)*

Variables

Définition :	Exemples :
Les **variables** sont des items où l'on peut conserver et récupérer des données. Les variables ont des noms qui représentent les données qu'elles conservent, ce qui rend une variable un concept abstrait puisqu'une chose (nom de variable) représente quelque chose d'autre (donnée). Les variables possèdent aussi un type de données (le type de données que l'on peut conserver) et une valeur (représentant ce qui est conservé dans la variable). Dans les langages de programmation, les variables sont sensibles à la casse, en un seul mot, sans espace.	• *var a = 10; var b = 5; a et b sont les noms des variables et le = vous indique quelle donnée est conservée dans chaque variable.* • *La variable "âge" pourrait représenter l'âge des gens (ex. : var âge;).* • *La variable "tauxdecroissance" pourrait être utilisée pour suivre la croissance d'une plante sur une certaine période de temps (ex. : var tauxdecroissance;).* • *La variable "temperature" pourrait être utilisée pour conserver et récupérer les lectures de température d'un capteur.* • *La variable "Nomdeleleve" pourrait être utilisée pour conserver le nom de l'élève et l'insérer automatiquement dans un courriel personalisé.*

Événements

Définition :	Exemples :
Un **événement** implique une action qui en déclenche une autre, comme lorsqu'un ordinateur répond à une action de son utilisateur. Les entrées d'un utilisateur incluent des actions telles le clic d'un bouton de souris, taper une touche du clavier ou toucher un écran tactile à un endroit précis.	• *Agrandir une carte lorsque quelqu'un double-clique dessus.* • *Défiler les images d'un téléphone lorsque quelqu'un glisse son doigt sur l'écran.* • *Ajuster le volume d'un video lorsque quelqu'un presse un des boutons + ou - .*

Opérateurs

Définition :	Exemples :
Les **opérateurs** sont des caractères qui demande à l'ordinateur d'exécuter une fonction précise, soit mathématique ou logique. Les opérateurs utilisés dépendent du langage de programmation mais peuvent inclure des opérateurs pour additionner (+), soustraire (-), multiplier (*), diviser (/), moins que (<), plus que (>) et égal à (==), aussi bien que des opérateurs logiques tels ET (&&), OU (II) et NON (!).	• *Quelle est la somme des âges de tous les membres de ma famille? (âgeMoi + âgeFrère + âgeMaman + âgePapa)* • *Est-ce que ton frère est plus âgé que toi? (âgeFrère > âgeMoi)* • *Les opérateurs Plus que et Moins que fournissent des résultats booléens (vrai ou faux) dans la plupart des langages de programmation.*

Fonctions

Définition :	Exemples :
Une **fonction** regroupe toutes les étapes d'une action complexe en une seule commande (ex. : brosser ses dents). Les fonctions sont particulièrement utiles pour définir une séquence de commandes qui peuvent être reproduites, par exemple comment faire tourner un robot à droite, à un angle de 90 degrés. La création de fonctions se base sur l'observation de régularités, l'abstraction et le raisonnement logique.	• *Brosser ses dents (implique ouvrir le robinet, mettre de la pâte à dents sur sa brosse à dents, etc.).* • *Fonction : Dessiner un carré (tracer une ligne droite de 3 cm, tournez à 90°, tracer une ligne droite de 3 cm, tournez à 90°, tracer une ligne droite de 3 cm, tournez à 90°,tracer une ligne droite de 3 cm, tournez à 90°.).* • *Fonction : Style de l'entête <h1 style="color:Gray;">*

Entrées & Sorties

Définition :	Exemples :
Les **entrées** et **sorties** (parfois abbréviées E/S mais plus souvent I/O pour l'anglais, Inputs et Outputs) est la forme de communication entre humains et ordinateurs, entre ordinateurs, à l'intérieur même d'un processus informatique, ou entre un ordinateurs/robot et son environnement. Les humains interagissent avec les ordinateurs grâce à des composants qu'on appelle périphériques. Lorsque les ordinateurs interagissent avec d'autres ordinateurs, c'est souvent à travers des réseaux comme l'internet. Les entrées et sorties sont impliquées à tous les niveaux opérationnels d'un ordinateur.	• *Les humains fournissent des entrées aux ordinateurs grâce à des périphériques comme les claviers, souris, caméras, microphones, etc.* • *Les ordinateurs fournissent des sorties grâce à des haut-parleurs, un écran, imprimantes, etc.* • *Les robots utilisent des dispositifs d'entrée comme des capteurs (capteur de lumière, de température, etc.), boutons, etc.* • *Les dispositifs de sortie des robots incluent des composants tels moteurs, haut-parleurs, actionneurs, témoins lumineux, etc.*

1.7 Définitions : Approches en pensée computationnelle

Débranché

Définition :	Exemples :
Utiliser la pensée computationnelle ne nécessite pas nécessairement un ordinateur! Plusieurs activités qui développent les compétences de pensée computationnelle peuvent être faites de façon « **débranchée** », signifiant sans ordinateurs. Jouer à un jeu avec des règles, faire un casse-tête de logique and créer et suivre des recettes sont différents façons de faire des exercices en pensée computationnelle de façon « débranchée ».	• *Programmation sur papier* • *Dessiner des cartes routières ou la complétion de labyrinthes.* • *Jouer à des jeux où une personne doit dire à une autre ce qu'il faut faire ou où aller.* • *Compléter une table de vérité, aussi connu comme le jeu de « vérité ou mensonge ».*

Apprentissage exploratoire

Définition :	Exemples :
Dans le contexte de la pensée computationnelle, l'**apprentissage exploratoire** (tinkering en anglais) signifie explorer la programmation de façon ludique et grâce des essais et erreurs. L'apprentissage exploratoire inclut changer certaines parties du code d'un tiers pour voir ce qui s'ensuit. L'apprentissage exploratoire est important parce qu'il développe la pensée créative et la confiance de l'utilisateur qui prend un risque en essayant quelque chose de nouveau. C'est aussi une méthode efficace de commencer à apprendre les bases de la programmation, via les relations de cause à effet. L'apprentissage exploratoire concorde avec la phase d'**UTILISATION** de la progression des apprentissages en pensée computationnelle (Use-Modify-Create) selon Lee et al.	• *Modifier les couleurs de vêtements d'un lutin (personnage) dans Scratch.* • *Ajuster le code pour afficher vos initiales sur l'écran matriciel dans MakeCode.* • *Modifier le code pour qu'un robot puisse se déplacer différemment.*

2 filles en mode « d'apprentissage exploratoire » dans l'environnement de programmation en blocs MakeCode.

Réutilisation et remixage

Définition :	Exemples :
La **réutilisation** signifie prendre une partie d'un code créée par d'autres et de l'utiliser pour résoudre un problème, au lieu de le créer à partir de zéro. Le **remixage** implique d'assembler ou « fusionner » du code pour créer un vidéo, un fichier sonore, du texte, etc. créés par d'autres afin de produire quelque chose de nouveau et unique. La réutilisation et le remixage concordent avec la phase de **MODIFICATION** de la progression des apprentissages en pensée computationnelle (Use-Modify-Create) selon Lee et al.	• *Utiliser les images existantes dans la bibliothèque de Scratch.* • *Prendre un projet que quelqu'un d'autre a fait dans Scratch et le changer (remixer) afin de lui donner sa touche personnelle.* • *Réutiliser une fonction que quelqu'un d'autre a créé (ex. : Au démarrage, afficher temp.).*

Fabrication

Définition :	Exemples :
Dans le contexte de la pensée computationnelle, la fabrication pure implique l'écriture d'un code de A à Z, sans débuter avec un code existant. Ceci est habituellement possible lorsqu'on est familier avec un langage de programmation et qu'on possède de l'expérience avec des approches telles l'apprentissage exploratoire et le remixage. La fabrication concorde avec la phase de **CRÉATION** de la progression des apprentissages en pensée computationnelle (Use-Modify-Create) selon Lee et al. (2011).	• *Créer son propre avatar et le dessiner avec Scratch.* • *Concevoir et construire un robot LEGO™ Mindstorms les différentes pièces incluses dans l'ensemble.* • *Créer et programmer un bandeau avec capteurs en utilisant un micro:bit.*

let's talk science | parlons sciences