La pensée computationnelle

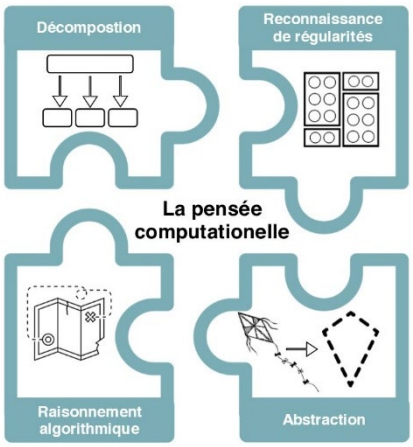
Il n'y a pas de bonne ou de mauvaise façon d'animer un atelier de programmation. Certains animateurs choisissent de commencer en présentant aux participants un logiciel de programmation et en les laissant s'amuser et explorer les fonctions, favorisant ainsi la méthode d'apprentissage « au fur et à mesure » grâce à une démarche de questionnement. D'autres préfèrent commencer en présentant les notions fondamentales de la programmation à l'aide d'exemples, ou en proposant des jeux « non connectés » qui illustrent ces notions de façon interactive. Nous recommandons les deux formules, dans l'ordre qui convient le mieux à votre groupe. Toutes ces activités aideront les participants à mieux comprendre ce qu'est la pensée computationnelle.

Les quatre formes de pensée computationnelle

Si un code de programmation peut être résumé à sa plus simple expression comme étant de « dire à un ordinateur de faire une tâche particulière », la pensée computationnelle peut être définie comme étant « la façon de décrire la tâche pour que l'ordinateur puisse la comprendre ». Les données de sortie d'un ordinateur sont fonction des données d'entrée. Si le résultat n'est pas celui auquel on s'attendait, c'est que ce qui a été demandé – le code écrit par l'utilisateur – n'était pas exact.

Les codes de programmation sont comme une discussion entre un utilisateur et un ordinateur. Comme toute conversation entre deux personnes, il vaut mieux éviter les erreurs de communication! Avant de se lancer dans un dialogue avec une machine, il est préférable de mettre de l'ordre dans les idées – c'est là que la pensée computationnelle entre en jeu. Ci-dessous sont expliquées les quatre principales formes de pensée computationnelle, toutes étant aussi importantes les unes que les autres. Les activités « non connectées » de la section suivante permettent à vos élèves de mettre en pratique chacune de ces compétences.

Décomposition
Les participants utilisent la décomposition pour morceler un problème complexe en problèmes plus petits, plus simples et plus faciles à résoudre.



Reconnaissance de régularités
Le fait de pouvoir reconnaître les régularités qui émergent aide les participants à faire des associations et à voir des similitudes dans des problèmes ou des expériences.

Abstraction
En utilisant l'abstraction, les participants se concentrent uniquement sur le cœur du problème – en écartant les détails non pertinents ou sans lien ni importance.

Raisonnement algorithmique
Les participants créent une série d'étapes à suivre pour résoudre un problème.