

# Comprendre la logique informatique avec les projets WeDo 2.0

Dans ce chapitre, vous découvrirez comment vous pouvez utiliser WeDo 2.0 pour développer des compétences en matière de logique informatique dans un contexte scientifique.





## La logique informatique avec les projets LEGO® Education WeDo 2.0

Les projets LEGO® Education WeDo 2.0 ont été conçus pour une utilisation à l'école primaire et au collège afin de développer notamment les compétences des élèves en matière de logique informatique.

La logique informatique est un ensemble de compétences que tout le monde peut utiliser pour résoudre différents problèmes de la vie de tous les jours. Dans WeDo 2.0, ces compétences sont développées à chaque phase des projets. Différentes modalités de mise en oeuvre sont présentées dans chacun des projets : retenez celles qui sont les plus pertinentes pour vos élèves et vous.

Chaque projet dans WeDo 2.0 associe l'utilisation des briques LEGO à un langage de programmation utilisant des icônes, permettant à vos élèves de trouver des solutions aux problèmes tout en étant initiés aux principes de la programmation.

WeDo 2.0 aborde la logique informatique par le biais d'activités de codage qui donnent vie aux créations des élèves, leur donnant le sourire et l'envie d'en savoir plus.





## Informatique, logique informatique, codage

Si les domaines des sciences et de la technologie ont vu le jour dès l'aube de l'humanité, l'informatique a une histoire bien plus récente. Néanmoins, cette jeune discipline a influencé non seulement la façon dont nous abordons les sciences et la technologie, mais aussi notre mode de vie.

L'informatique est une discipline technique, partageant des caractéristiques avec les sciences, la technologie et les mathématiques.

Toutes les disciplines scientifiques et techniques permettent de développer un état d'esprit et des pratiques utiles tout au long de la vie. Parmi celles-ci, citons par exemple la possibilité de susciter des questions, de concevoir des solutions ou encore de communiquer des résultats.

La logique informatique fait partie de ces pratiques. Elle est une manière de réfléchir et une démarche grâce à laquelle chacun peut résoudre des problèmes.

La logique informatique peut être décrite comme un ensemble de compétences qui englobe notamment la pensée algorithmique. Les termes « code » ou « codage » peuvent être utilisés pour décrire l'action de créer un algorithme.

Le codage est donc un moyen par lequel on développe la logique informatique dans un contexte scientifique et technique.

## Disciplines scientifiques et techniques

Sciences, Technologie, Mathématiques,  
Informatique

**Développer un état d'esprit et des pratiques utiles tout au long de la vie.**

1. Rechercher.
2. Organiser.
3. Concevoir.
4. Planifier.
5. Produire.
6. Concevoir un programme.

- a. Décomposer
- b. Conceptualiser
- c. Penser de manière algorithmique (code)
- d. Évaluer
- e. Généraliser

7. Simuler.
8. Modifier.



## Qu'est-ce que la logique informatique ?

L'expression « logique informatique », également appelée « pensée computationnelle », a été utilisée pour la première fois par Seymour Papert, mais le professeur Jeannette Wing est connue pour avoir popularisé le concept. Elle a défini la logique informatique comme :

« les processus de réflexion utilisés pour formuler des problèmes et leurs solutions de façon à ce que les solutions soient représentées sous une forme qui puisse être effectivement réalisée par un agent de traitement des informations » (Wing, 2011).

La logique informatique est utilisée dans divers domaines et situations, et nous l'utilisons dans notre vie quotidienne. Les compétences en matière de logique informatique sont présentes dans les sciences, la technologie et les mathématiques. Ces compétences peuvent être définies comme suit :

### Décomposition

La décomposition est la capacité à simplifier un problème en parties plus petites afin de faciliter le processus de recherche de solution. En faisant cela, le problème devient plus facile à expliquer à une autre personne, ou plus facile à diviser en tâches. La décomposition entraîne souvent la généralisation.

Exemple : Lorsque l'on part en vacances, la préparation (ou le projet) peut être séparée en sous-tâches : réserver les billets d'avion, réserver un hôtel, faire sa valise, etc.

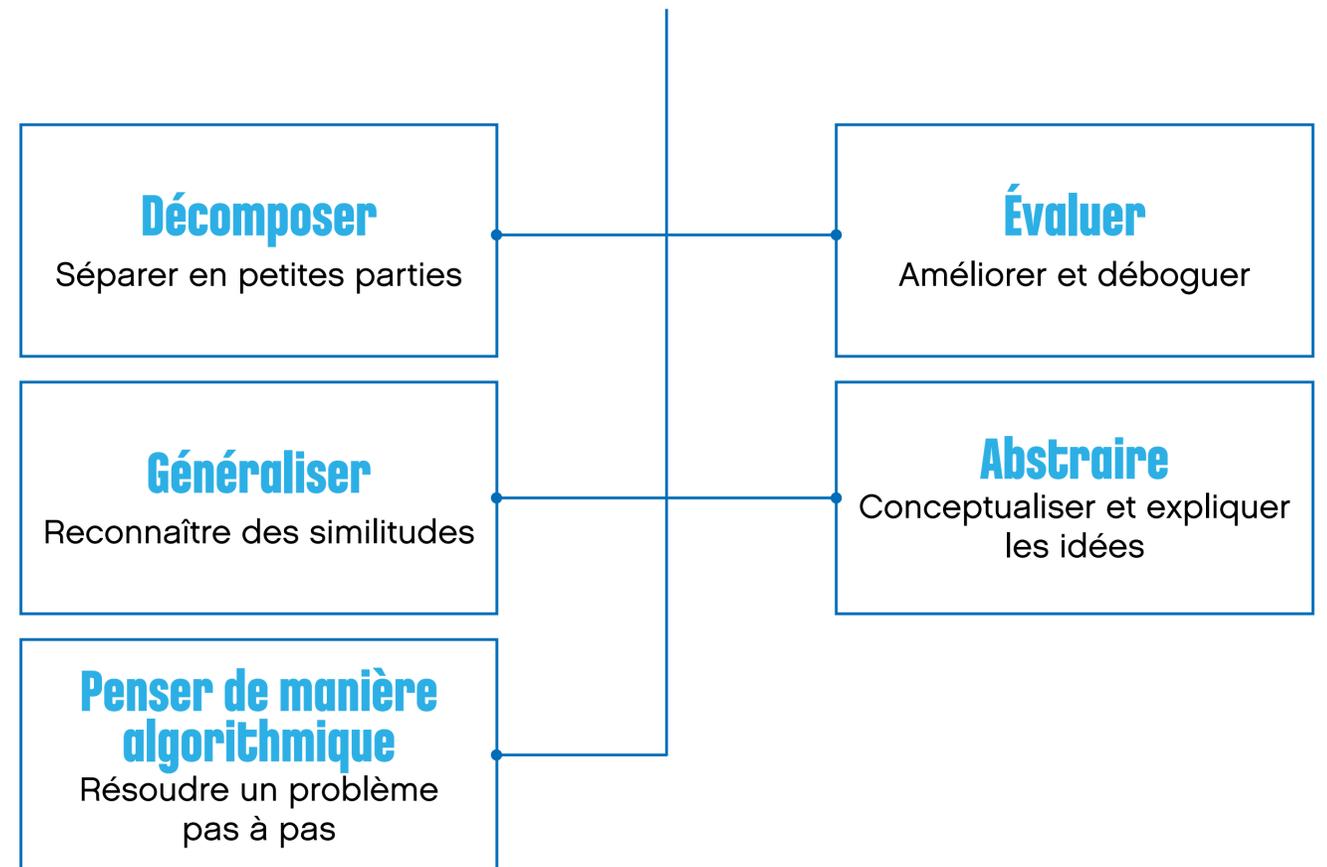
### Généralisation (reconnaissance des similitudes)

La généralisation est la capacité à reconnaître les parties d'une tâche qui sont connues ou ont été vues ailleurs. Cela permet souvent de trouver des manières plus faciles de concevoir des algorithmes.

Exemple : Les feux de signalisation fonctionnent en répétant les mêmes séries d'actions indéfiniment.

## Logique informatique

Nos manières de résoudre les problèmes





## Qu'est-ce que la logique informatique ?

### Logique algorithmique

La logique algorithmique est la capacité à créer une série ordonnée d'étapes en vue de résoudre un problème.

Exemple 1 : lorsqu'on fait une recette, on suit une série d'étapes afin de préparer un repas.

Exemple 2 : lorsqu'on joue avec des ordinateurs, on peut coder une séquence d'actions qui indiquent à l'ordinateur ce qu'il doit faire.

### Évaluation ou débogage

C'est la capacité de vérifier si un prototype fonctionne comme prévu, et si ce n'est pas le cas, la capacité à identifier les points à améliorer. C'est également le processus par lequel passe un programmeur informatique pour détecter et corriger les erreurs dans un programme.

Exemple 1 : lorsqu'on cuisine, on goûte régulièrement le plat pour vérifier s'il est correctement assaisonné.

Exemple 2 : lorsqu'on vérifie les fautes d'orthographe et de ponctuation dans un travail écrit, on débogue celui-ci pour qu'il puisse être lu correctement.

### Abstraction

L'abstraction est la capacité d'expliquer un problème ou une solution en supprimant les détails qui n'ont pas d'importance. En d'autres termes, être capable de conceptualiser une idée.

Exemple : Lorsqu'on décrit un vélo, on utilise simplement quelques détails pour le décrire. On peut mentionner son type et sa couleur, et ajouter d'autres détails pour quelqu'un qui s'intéresse particulièrement aux vélos.



## Un processus de développement des compétences en matière de logique informatique

### Utiliser un processus d'étude de conception

Lorsqu'ils cherchent des solutions à un problème, les ingénieurs utilisent un processus de conception. Ils passent par une série de phases qui les guident vers une solution. Lors de chacune de ces phases, certaines de leurs compétences sont utilisées ou développées. Ce sont ces compétences que nous appelons « compétences en matière de logique informatique ».

Dans WeDo 2.0, les élèves suivent un processus similaire :

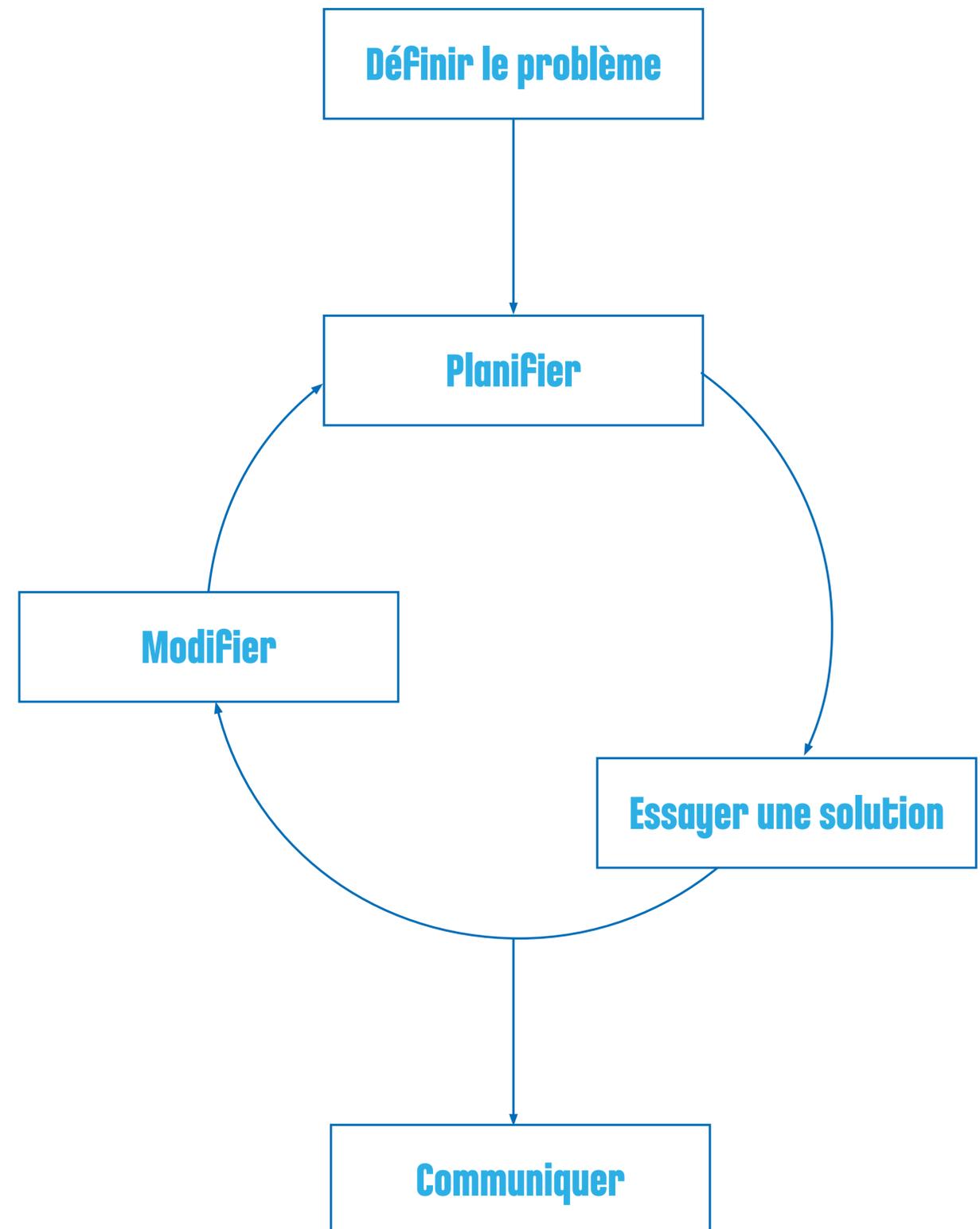
### Définir le problème

On présente aux élèves un sujet qui les guide vers un problème ou une situation qu'ils doivent améliorer. Parfois, un problème peut présenter de nombreux détails. Pour faciliter sa résolution, le problème peut être décomposé en parties plus petites.

En définissant le problème d'une manière simple et en identifiant des critères de succès, les élèves développeront une compétence appelée « décomposition ».

En d'autres termes :

- Est-ce que l'élève est capable d'expliquer le problème tout seul ?
- Est-ce que l'élève est capable de décrire la manière dont il évaluera s'il a réussi à résoudre le problème ou non ?
- Est-ce que l'élève est en mesure de décomposer le problème en parties plus petites et plus gérables ?





## Un processus de développement des compétences en matière de logique informatique

### Planification

Les élèves doivent passer un certain temps à imaginer différentes solutions à un problème, puis à établir un plan détaillé pour mettre en pratique l'une de leurs idées. Ils définiront les étapes par lesquelles ils doivent passer pour trouver la solution. En identifiant les parties de la tâche qu'ils sont susceptibles d'avoir déjà vues, ils développeront une compétence appelée « généralisation ».

En d'autres termes :

- Est-ce que l'élève est capable d'établir une liste d'actions à programmer ?
- Est-ce que l'élève est capable d'identifier des parties de programmes qu'il pourrait utiliser ?
- Est-ce que l'élève est capable de réutiliser les parties de programmes ?

### Essai

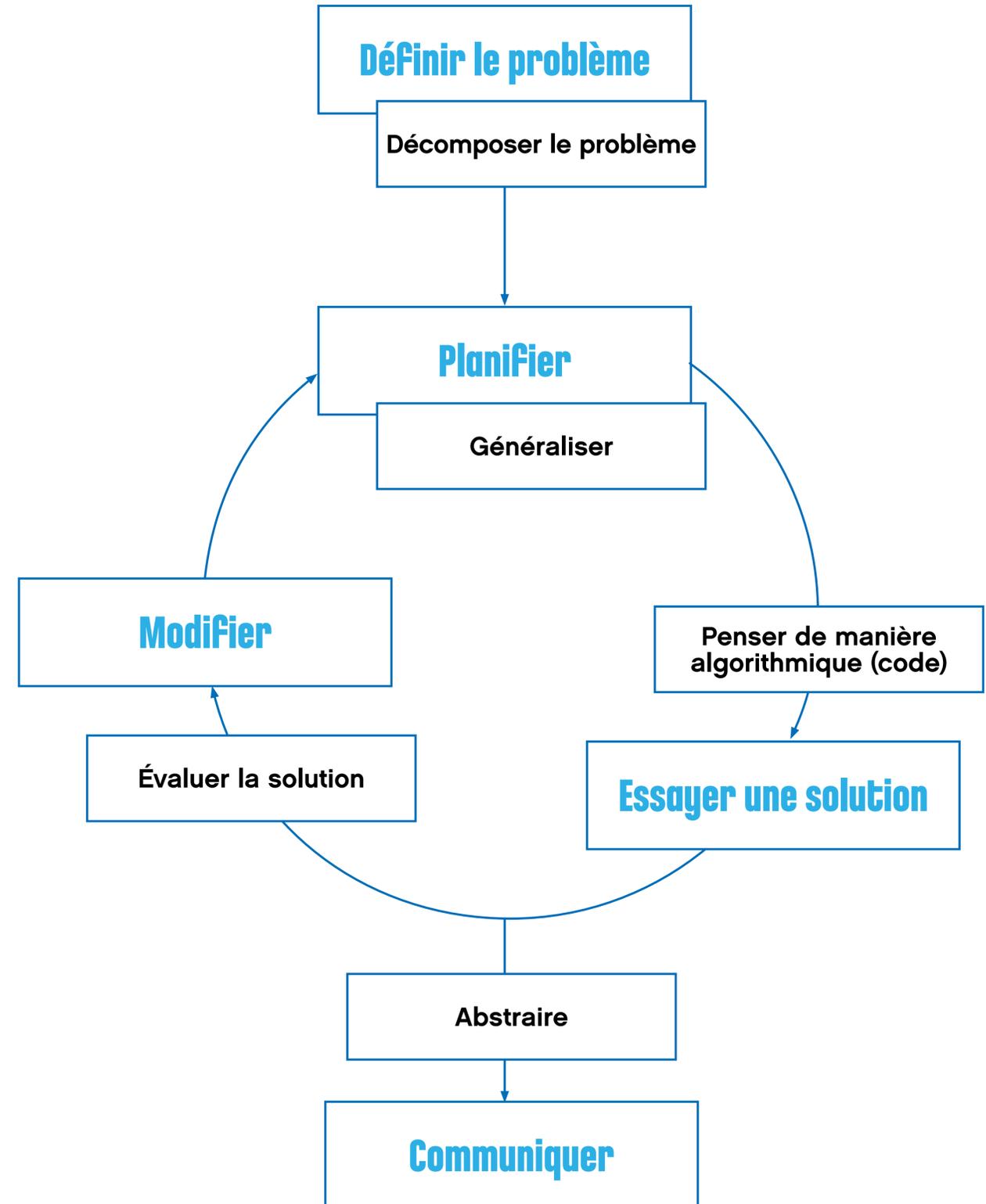
Chaque élève est ensuite chargé de créer la version finale de sa solution. Dans cette phase du processus, ils utilisent un langage de programmation par icônes pour activer leurs modèles LEGO®. Lorsque les élèves programment leurs idées, ils développent leurs compétences en matière de logique algorithmique.

En d'autres termes :

- Est-ce que l'élève est capable de programmer une solution à un problème ?
- Est-ce que l'élève est capable d'utiliser une séquence, des boucles, des instructions conditionnelles ?

### Modification

Les élèves évalueront leur solution en fonction des critères de succès établis au préalable. En utilisant leurs compétences en matière d'évaluation, ils détermineront s'ils doivent modifier, réparer, déboguer ou améliorer une partie de leur programme.





## Un processus de développement des compétences en matière de logique informatique

En d'autres termes :

- Est-ce que l'élève fait des itérations de son programme ?
- Est-ce que l'élève résout les problèmes dans son programme ?
- Est-ce que l'élève est capable de juger si la solution est liée au problème ?

### Communication

Les élèves présenteront la version finale de leur solution à la classe, expliquant dans quelle mesure elle répond aux critères de succès. En expliquant leur solution avec le bon niveau de détails, ils développeront leurs compétences en matière d'abstraction et de communication.

En d'autres termes :

- Est-ce que l'élève explique correctement sa solution ?
- Est-ce que l'élève donne suffisamment de détails pour améliorer la compréhension ?
- Est-ce que l'élève explique en quoi sa solution répond aux critères de succès ?





## Développement de la logique informatique par le biais du codage

Afin de développer leurs compétences en matière de pensée algorithmique, les élèves seront initiés à certains principes de programmation. Lorsqu'ils développeront leurs solutions, ils organiseront une série d'actions qui donneront vie à leurs modèles.

Les principes de programmation WeDo 2.0 les plus courants que les élèves utiliseront sont :

### 1. Valeur de sortie

La valeur de sortie est une variable contrôlée par le programme écrit par les enfants. Par exemple, les valeurs de sortie de WeDo 2.0 sont les sons, les voyants, l'affichage, l'allumage et l'extinction des moteurs.

### 2. Valeur d'entrée

La valeur d'entrée est une information reçue par un ordinateur ou un appareil. Elle peut être saisie en utilisant des capteurs sous forme de valeur numérique ou textuelle. Par exemple, un capteur qui détecte ou mesure quelque chose (comme une distance) convertit cette valeur en un signal d'entrée numérique pour qu'il puisse être utilisé dans un programme.

### 3. Événements (attendre)

Les élèves peuvent dire à leur programme d'attendre qu'un événement se produise avant de continuer à exécuter la séquence d'actions. Les programmes peuvent attendre pendant une durée spécifique ou attendre qu'un événement soit détecté par un capteur.

### 4. Boucle

Les élèves peuvent programmer des actions à répéter soit indéfiniment, soit pendant une durée spécifique.

### 5. Fonctions

Les fonctions sont un groupe d'actions qui doivent être utilisées conjointement dans des situations spécifiques.

Par exemple, le groupe de blocs qui pourrait être utilisé pour faire clignoter un voyant serait appelé « la fonction clignoter ».

### 6. Conditions

Les conditions sont utilisées par les élèves pour programmer des actions qui ne doivent être exécutées que dans certains cas. Créer des conditions dans un programme signifie qu'une partie du programme ne sera jamais exécutée si la condition n'est jamais remplie. Par exemple, si le détecteur d'inclinaison est incliné à gauche, le moteur démarrera, et si le détecteur est incliné à droite, le moteur s'arrêtera ; si le détecteur d'inclinaison n'est jamais incliné à gauche, le moteur ne démarrera jamais et s'il n'est jamais incliné à droite, alors le moteur ne s'arrêtera jamais.

