OUTIL - LA PROGRAMMATION EXPLIQUÉE AUX ENFANTS

CRÉATION DE CONTENU > 3.4 PROGRAMMATION

CONVIENT POUR	AGE	NIVEAU DE COMPÉTENCE	FORMAT	DROITS D'AUTEUR	LANGUE(S)
Animateurs	N/A	Niveau 1	Fiche d'activité	Creative Commons (BY-SA)	Français

Cette activité basée sur la discussion propose une approche des notions de base de la programmation.

Objectif général	Connaissances
Temps de préparation pour l'animateur	moins d'une 1 heure
Domaine de compétence	3 – Création de contenu
Nom de l'auteur	BSF Belgique
Ressource originellement créée	Français

DÉROULÉ

1 Introduction

Cette fiche vous apporte des explications simplifiées sur une série de notions sur et autour de la programmation. Elle vise à donner des bases aux animateur.ices qui ne seraient pas familiarisé.e.s avec ces sujets, pour les aider à mener des activités.

Vous pouvez explorer ces notions avec votre public de différentes façons, par exemple :

- en tant que liant entre différentes activités (les fiches concernées seront données en lien)
- avec un public déjà un peu initié à ces notions, via une discussion où les notions sont apportées par le public, puis complétées par vous
- avec un public non initié mais plus âgé (>=12 ans), via une discussion à base de questions guidées
- avec un public non initié et plus jeune (<12 ans), via des explications plus ex-cathedra mais où vous maintiendrez de l'interactivité en leur demandant des exemples et laisserez un espace de questions

2 Est-ce qu'un ordinateur est intelligent ?

Des activités comme le Robot idiot ou Scratch Jr permettent de faire comprendre une notion essentielle : l'ordinateur ne **réfléchit** pas. Il ne pense pas. Il n'est pas intelligent. Il ne peut pas deviner. Il ne peut pas comprendre.

Il ne fait qu'une chose : **exécuter** les instructions qu'on lui a données. Si on lui donne de mauvaises instructions, il va les suivre malgré tout ! Où alors est son avantage ? Dans la **complexité** de ce qu'il peut réaliser (faire des calculs très difficiles), dans la **vitesse** d'exécution (en 2020, un simple ordinateur de bureau peut faire plus de dix milliards d'opérations par seconde !), et surtout, il ne s'épuise jamais (contrairement à un humain).

Puisqu'il ne réfléchit pas, il faut réfléchir pour lui, par exemple en trouvant des solutions pour trier les choses efficacement. Ensuite, on lui donne les **instructions** qui lui disent quoi faire, et il le fera, sans réfléchir, et très vite. Il est donc important de lui donner des instructions très claires, très précises, et surtout sans erreur, sinon on obtient un **bug** : le résultat obtenu n'est pas celui prévu, il y a une erreur dans le **programme**.

3 Vous avez dit programme ?

Un **programme**, c'est un ensemble d'instructions, que l'ordinateur va suivre l'une après l'autre jusqu'à arriver au bout. En faisant cela, il va réaliser une tâche ou résoudre un problème. La tâche peut être très variée : ouvrir un site internet, jouer une musique, envoyer un document, réagir à un clic de souris, faire bouger un personnage dans un jeu vidéo, chercher un mot dans un texte, trier des données, calculer des trajectoires de navette spatiale, ...

La personne qui écrit ces instructions est un(e) **programmeur**(se), ou **développeur**(se). Ne pas confondre avec programmateur, qui est la personne qui s'occupe d'un programme dans un évènement culturel.

Les instructions sont rédigées en utilisant un ou plusieurs **langages** informatiques.

4 Qu'est-ce qu'un langage informatique ?

Les ordinateurs ne peuvent pas comprendre de langage humain; ils ne comprennent que le **binaire** (une façon de présenter l'information avec des 0 et des 1, grâce à l'électricité). Les humains pour leur part ne peuvent que difficilement « parler binaire » (et ce n'est pas pratique du tout). On a donc inventé des **langages de programmation**, faciles à manipuler pour les humains, et qui sont traduits en binaire pour être compris par l'ordinateur.

Un langage de programmation est donc un moyen de faire le **lien** entre le/la programmeur/euse (personne qui créé des programmes) et l'ordinateur. Il permet de donner des ordres, qu'on appelle **instructions**, à l'ordinateur, pour lui dire quoi faire (jouer une vidéo, ouvrir un site internet,...) et comment réagir (à un clic de souris, à une touche,...).

Les langues parlées par les humains ont évolué au fil des changements de pratique : création de nouveaux mots, mots qui disparaissent, langues qui se mélangent,... Les langages de programmation eux, ont évolué avec les **progrès** des technologies. Différents langages informatiques permettent de faire **différentes** choses : gérer des sites internet, faire une app pour smartphone, piloter des robots, etc.

En général, les langages de programmation utilisent des mots anglais pour exprimer des choses très simples, très précises. Plus un langage est proche du binaire, plus il est de « **bas niveau**« . Plus un langage est éloigné du binaire et permet d'exprimer des concepts abstraits, plus il est de « **haut niveau** « . Dans Scratch, on peut choisir la langue humaine dans laquelle on va programmer, et il s'agit d'un langage de haut niveau.

5 Pour aller plus loin avec les langages de programmation

Les langues humaines sont regroupées en « familles » : langues germaniques, langues romanes, etc., en fonction de leurs origines et de leurs caractéristiques communes. Les langages de programmation eux, ont des « familles » basées sur des **paradigmes** : des façons de fonctionner, qui permettent de programmer dans des styles différents. On représente les choses et on les manipule de façon différente, un petit peu comme certaines langues humaines, avec l'allemand qui utilise des déclinaisons alors que le français non, ou bien le néerlandais qui met parfois ses verbes à la fin des phrases...

Par exemple, dans la famille des langages **par bloc**, au lieu d'écrire les instructions mot par mot, on utilise des blocs pré-faits, qu'on assemble bout à bout. C'est le cas de Scratch et Blockly par exemple. Ce paradigme a été inventé pour simplifier l'apprentissage de la programmation, et s'adresse surtout aux enfants.

Dans la famille des langages **orientés objet**, on peut représenter des objets et des concepts, qui peuvent ensuite être manipulés. C'est le cas de Java, Python, C++ par exemple. Ce paradigme facilite la

représentation des processus et sert dans énormément de domaines, depuis les jeux vidéos jusqu'aux logiciels de gestion d'entreprise.

Dans la famille des langages **pour le web**, on trouve une série de langages très différents qui servent à créer et faire fonctionner des sites internet. Le plus basique, HTML, fonctionne avec balises qui décrivent où se trouvent les choses (texte, images, cadres, liens,...) sur une page internet. Tous les sites l'utilisent ! Souvent, on y rajoute le CSS, qui permet de mettre en forme les choses (changer les couleurs, les polices,...). Et de nos jours, presque tous les sites y ajoutent également du Javascript, qui est un langage orienté objet, qui permet de manipuler les choses (faire bouger des images, intégrer de la vidéo, faire des animations,...) pour rendre les sites plus dynamiques et leur apporter des fonctionnalités.

6 Et les algorithmes, alors ?

La majorité des activités visant à enseigner la programmation commence d'abord par enseigner ce qu'est un algorithme. Et pour cause : les programmes sont des ensembles d'algorithmes, parfois très complexes. Mais tous les algorithmes ne sont pas des programmes...

En effet, un **algorithme** est une suite d'étapes ordonnées, qui permettent l'accomplissement d'une tâche. On peut en trouver de diverses formes, la plus commune étant celles des recettes de cuisine. Une recette établit en effet une série de paramètres (les ingrédients), renseigne une série d'étapes (couper, malaxer, cuire,...), et permet de réaliser une tâche (préparer un plat).

Les programmes informatiques ne sont qu'une façon de rédiger, transmettre, utiliser des algorithmes. Ceux-là servent à réaliser des tâches très variées : trier, chercher, transmettre des données, etc.

Et le **code** ? Coder, c'est stocker l'information d'une certaine façon (par exemple, le présent texte est codé en français). Lorsqu'on utilise une méthode connue uniquement d'un petit nombre de personnes, ça devient un code secret. En informatique, on stocke, manipule, code et décode l'information sans arrêt, y compris les instructions dans les programmes informatiques. C'est pourquoi on appelle parfois les programmes « code » et les programmeurs des « codeurs ».

7 Quelques notions de programmation

Il existe une série de notions qui sont communes à pratiquement tous les langages de programmation, et que vous découvrirez au fil des activités Scratch. Il s'agit des **structures de contrôle** : des éléments de code qui permettent de contrôler, affecter, le déroulement des instructions.

Les conditions

Très souvent, nos actions dépendent de plusieurs paramètres : s'il fait beau, alors je ne prends pas de pull. Vérifier une condition (« si…alors… ») permet de n'exécuter une partie du code que lorsque la condition ou les conditions sont respectées. Il est aussi possible d'offrir un code alternatif, qui sera exécuté lorsque les conditions ne seront pas utilisées (« si…alors…sinon… »).

Les boucles

Vous le verrez vite dans les réalisations Scratch ou dans certaines activités débranchées, on se retrouve à avoir besoin de répéter la même instruction plusieurs fois. Pour ne pas devoir écrire le code plusieurs fois, on peut recourir à une boucle, qui encapsule une partie de code et va l'exécuter plusieurs fois (« répéter… »).

Le nombre de fois que ce code sera exécuté peut dépendre d'une condition (« répéter… tant que… »), être fixe (« répéter x fois »), être infini (« répéter indéfiniment »).

8 Pour aller plus loin

Conseil médiation

Pour aller plus plus loin sur le sujet, nous vous conseillons de vous référer à la fiche outil « Guide de présentation de Scratch«