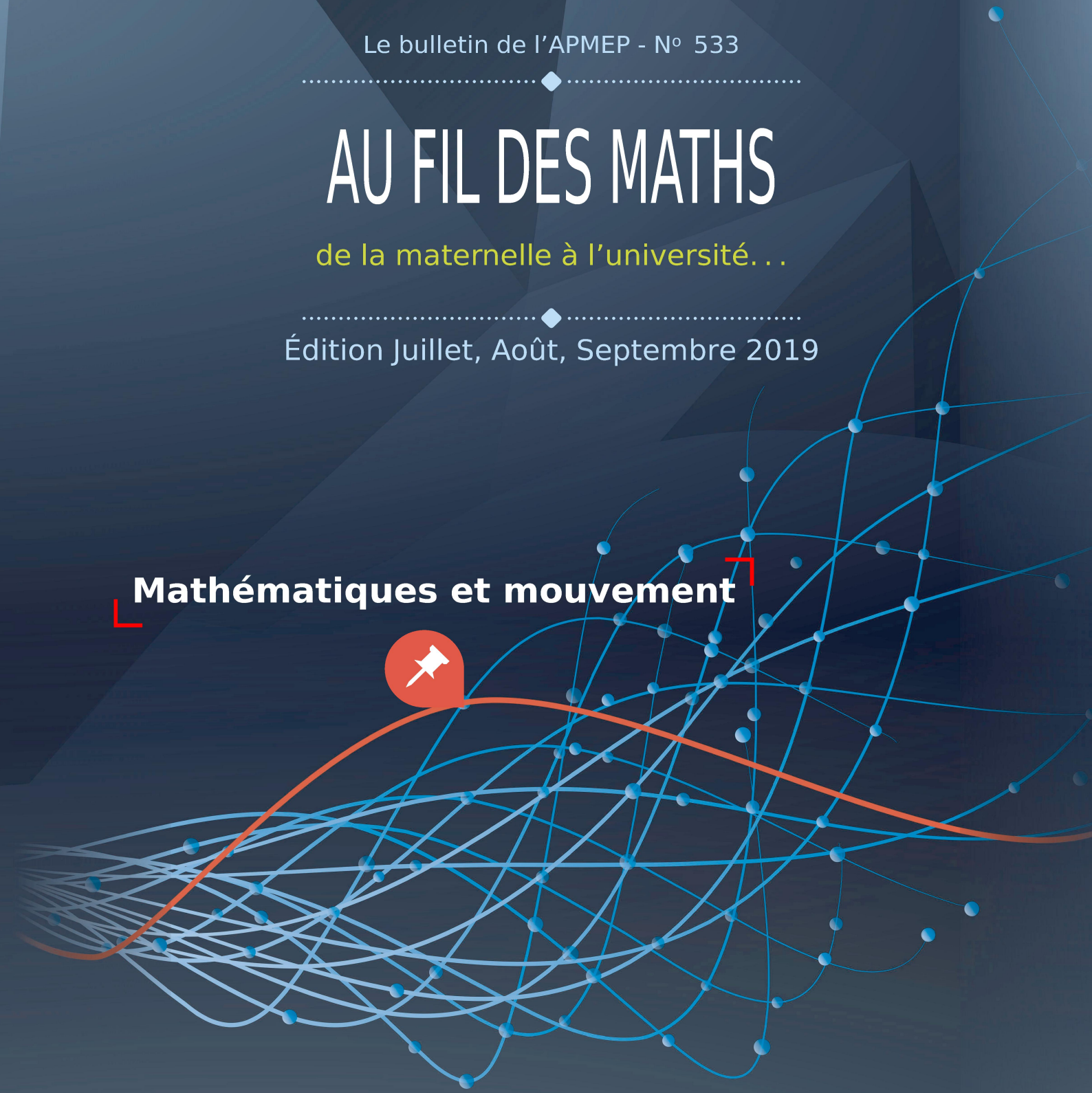
Le bulletin de l'APMEP - N° 533

AU FIL DES MATHS

de la maternelle à l'université. . .

Édition Juillet, Août, Septembre 2019

Mathématiques et mouvement



APMEP

Association des Professeurs de Mathématiques de l'Enseignement Public

ASSOCIATION
DES PROFESSEURS DE MATHÉMATIQUES
DE L'ENSEIGNEMENT PUBLIC

26 rue Duméril, 75013 Paris
Tél. : 01 43 31 34 05 - Fax : 01 42 17 08 77
Courriel : secretariat-apmep@orange.fr - Site : https://www.apmep.fr
Présidente d'honneur : Christiane Zehren

Au fil des maths, c'est aussi une revue numérique augmentée :
https://afdm.apmep.fr
version réservée aux adhérents. Pour y accéder connectez-vous à votre compte *via* l'onglet *Au fil des maths* (page d'accueil du site) ou *via* le QRcode, ou suivez les logos.

Si vous désirez rejoindre l'équipe d'*Au fil des maths* ou bien proposer un article, écrivez à aufildesmaths@apmep.fr

Annonceurs : pour toute demande de publicité, contactez Mireille Génin mcgenin@wanadoo.fr

À ce numéro est jointe la plaquette
Visages 2019-2020 de l'APMEP.

ÉQUIPE DE RÉDACTION

Directeur de publication : Sébastien Planchenault..

Responsable coordinateur de l'équipe : Lise Malrieu..

Rédacteurs : Vincent Beck, Marie-Astrid Bézard, François Boucher, Richard Cabassut, Séverine Chassagne-Lambert, Frédéric De Ligt, Mireille Génin, Cécile Kerboul, Valérie Larose, Lise Malrieu, Jean-Marie Martin, Daniel Vagost, Thomas Villemonteix, Christine Zelty..

« Fils rouges » numériques : Gwenaëlle Clément, Nada Dragovic, Laure Étévez, Marianne Fabre, Robert Ferréol, Adrien Guinemer, Christophe Romero, Jacques Vallois..

Illustrateurs : Pol Le Gall, Olivier Longuet, Jean-Sébastien Masset..

Équipe TEXnique : François Couturier, Isabelle Flavier, Anne Héam, François Pétiard, Olivier Reboux, Guillaume Seguin, Sébastien Soucaze, Michel Suquet..

Maquette : *Olivier Reboux*.

Votre adhésion à l'APMEP vous abonne automatiquement à *Au fil des maths*.
Pour les établissements, le prix de l'abonnement est de 60 € par an.
La revue peut être achetée au numéro au prix de 15 € sur la boutique en ligne de l'APMEP.

Mise en page : François Pétiard
Dépôt légal : Juillet, Août, Septembre 2019
Impression : Imprimerie Corlet
ZI, rue Maximilien Vox BP 86, 14110 Condé-sur-Noireau ISSN : 2608-9297

AU FIL DES MATHS

Algorithmique débranchée

L'informatique est passée en 40 ans d'un apprentissage sélectif concernant peu d'élèves à un apprentissage scolaire de masse. De la notion d'algorithme depuis 2010 aux concepts de programmation en 2017, l'informatique est désormais intégrée à l'enseignement des mathématiques au lycée. Cet article propose deux activités en « débranché » pour travailler spécifiquement l'algorithmique.

Cyrille Kirch & Olivier Jutand (groupe Lycée de l'IREM de Poitiers)

Dans les années 1970, l'informatique naissante occupait quelques passionnés autodidactes. Depuis, elle a pris une importance de plus en plus grande dans nos vies personnelle et professionnelle : utilisation de logiciels, d'objets connectés ou programmes pour agir, navigation sur l'internet, etc. L'enseignement de SNT (Science Numérique et Technologie) en Seconde à partir de la rentrée 2019, ainsi que la mise en place d'une certification obligatoire en fin de Troisième et de Terminale via la plateforme PIX dès 2020 permettront de révéler à un plus grand nombre les dessous des outils utilisés.

L'informatique est un domaine d'activité scientifique, technique et industriel qui concerne le traitement automatique de l'information par des machines exécutant des listes d'instructions, les programmes. Elle englobe un large champ de compétences, en passant de l'utilisation d'interfaces homme-machine accessible à un large public, à une partie plus théorique, voire abstraite, jusqu'alors destinée à des professionnels : la programmation, qui consiste à coder des instructions dans un langage spécifique/informatique compréhensible par la machine.

Si la démarche algorithmique est, depuis les origines, une composante essentielle de l'activité mathématique, ce n'est que depuis 2010 que la programmation est réellement entrée dans les programmes scolaires. Les collégiens de cycle 4 apprennent maintenant à écrire, mettre au point et exécuter un programme simple avec Scratch, un logiciel par blocs. Les lycéens, depuis 2017, doivent programmer leurs algorithmes avec un logiciel textuel, le langage Python est imposé dans les programmes de 2019.

Une réflexion didactique doit être menée pour rendre efficient pour tous ce nouvel enseignement, et en particulier la programmation, à savoir la « production d'un texte dans un langage informatique » d'après le programme de Seconde. En premier lieu, notons que la transition Collège-Lycée dans ce domaine ne va pas de soi. Les collégiens ayant utilisé un système de programmation par blocs (Scratch) n'en sont pas devenus pour autant des programmeurs experts. En effet, la pratique de Scratch (ou de tout autre système de programmation par blocs) à ce niveau fait apparaître des insuffisances :

- des structures limitées (compteur non apparent dans la boucle « `Repeat` ») ;
- des blocs masquant des structures algorithmiques élémentaires dont les élèves ne peuvent pas percevoir l'utilité, ni en comprendre le fonctionnement (nécessité, par exemple d'expliciter la fin d'une boucle dans un langage textuel).

Certaines des activités que nous proposons ont pu être menées dès le cycle 3 car elles font à juste titre partie des « classiques » de l'algorithmique débranchée. Ce n'est pas grave : tous auront ici l'opportunité de réellement travailler et synthétiser sur la structure d'algorithme.

Il nous apparaît donc important en début de Seconde de travailler spécifiquement l'algorithmique avant d'aborder la traduction dans un langage textuel, afin de distinguer les difficultés de l'algorithmique de celles liées au langage de programmation.

En algorithmique, on rencontre souvent des difficultés d'ordre mathématique :

- rigueur et structuration d'une démarche (d'un raisonnement) ;
- conceptualisation de la notion de boucle liée à la notion d'itération ;
- utilisation des booléens (dans les instructions conditionnelles).

En programmation, d'autres obstacles interviennent :

- affectation et gestion des variables
  * différence entre `:=` et `=` pour certains langages ;
- syntaxe propre à chaque langage
  * nécessité et utilité des indentations dans les boucles (Python),
  * `for i in range (1,17)` pour une boucle de 1 à 16 (Python)[1],
  * le signe == pour un test d'égalité (Algobox, Python, ...),
  * axe des ordonnées orienté vers le bas de l'écran pour la gestion des pixels.

> La mise en évidence (de manière non exhaustive) des difficultés répertoriées ci-dessus nous amène à penser qu'avant de placer les élèves de Seconde face à un logiciel de programmation, il est souhaitable de leur faire travailler la nature même d'un algorithme.

Les problèmes rencontrés lors de l'apprentissage de la syntaxe d'un langage de programmation risquent de masquer les difficultés spécifiquement algorithmiques notamment la conceptualisation de structures telles qu'affectation, répétition, boucle et compteur de boucle, condition. Une solution envisagée pour contourner cet obstacle est de se libérer de l'outil informatique et de commencer par un travail qualifié de « débranché » c'est-à-dire sans la moindre machine.

Dans cette optique, le groupe Lycée de l'IREM de Poitiers a construit deux situations permettant un enseignement de l'algorithmique « débranchée » :

- l'une proposant de reproduire une figure sur un quadrillage simulant un écran informatique composé de (millions de) pixels ;
- l'autre menée parallèlement à un travail sur les fonctions.

Pour chacune de ces situations, présentées succinctement dans la suite de cet article, nous expliquons nos choix, exposons les observations faites lors de leur utilisation avec les élèves et dressons un bilan dans la brochure[2] [1].

1. On peut aussi utiliser `for i in range (16)` qui fait 16 fois parce que le compteur a débuté à 0.
2. La brochure filigranée est téléchargeable gratuitement sur le site de l'IREM de Poitiers. L'achat de la brochure papier permet d'obtenir des identifiants pour télécharger notamment les planches de cartes évoquées dans l'article.

Dessiner avec un jeu de cartes

Structuration

Nous avons imaginé un jeu de cartes permettant de travailler la notion d'algorithme avec un nombre minimum d'instructions de base mais sans limitation du nombre de cartes.

Ce travail ne nécessite pas de salle informatique mais est sûrement plus productif et confortable pendant les heures en demi-classe, où il est plus facile d'observer et de comprendre le comportement des élèves.

Par groupe de deux, ils sont amenés à écrire un algorithme (avec les cartes), qui sera lu et exécuté par un autre groupe d'élèves. Ainsi chaque production est contrôlée par des pairs et entraîne des discussions en cas d'ambiguïtés. Trois instructions élémentaires nous sont apparues nécessaires pour réaliser des constructions basiques sur un quadrillage :

.

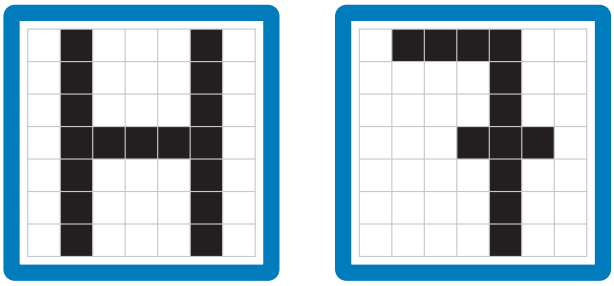
Étape 1 : elle consiste à définir sans ambiguïté l'action représentée par chacune des trois cartes et l'ordre de lecture des instructions. Laissés seuls face à un exemple[3], les élèves sont vite capables de définir chacune des trois cartes, mais quelques questions légitimes se posent sur le sens de lecture de la liste d'instructions.

-

Avancer d'une case dans la direction du curseur en noircissant ;
- Pivoter le curseur sur place d'un quart de tour dans le sens des aiguilles d'une montre ;
- Pivoter le curseur sur place d'un quart de tour dans le sens inverse des aiguilles d'une montre.

Étape 2 : une fois les trois instructions de base définies, chaque groupe dispose d'un jeu de cartes comportant 25 cartes

, 10 cartes et 10 cartes , et doit lister et ordonner les instructions nécessaires pour la reproduction d'une figure qui lui est donnée (lettre, chiffre, forme géométrique simple. . .) telles celles ci-dessous.



Une fois la liste de cartes « instructions » posées sur la table, les groupes changent de place et tentent de suivre l'algorithme proposé par un autre groupe. Les élèves perçoivent alors l'importance d'être rigoureux et la nécessité d'un pseudo-langage (règles communes).

3. On donne le dessin de la lettre T et la liste des instructions correspondantes ordonnées en une colonne permettant de reproduire le dessin à partir d'une position initiale du curseur.

Cette activité entraîne de nombreuses discussions. En effet, les cartes ayant été parfaitement définies avant :

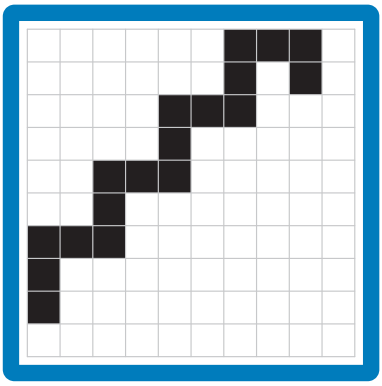
1. on noircit quand on avance (si on ne veut pas, il faudra inventer une autre carte) ;
2. le point de départ est donné par celui qui propose l'algorithme grâce à un pointeur sur un quadrillage vierge (et demandé d'ailleurs par les autres) ;
3. le sens initial est donné en même temps que la case initiale.

On verra plus tard qu'avec Python, ces paramètres (position initiale et sens) sont définis par défaut.

On peut alors définir clairement un algorithme comme une liste ordonnée d'instructions « simples » et non ambiguës ayant pour but, dans cette première approche, de reproduire un dessin.

Répétitions

Le but à atteindre dans un second temps est la compréhension et l'apprentissage de la structure de boucle, fondamentale en informatique. On propose donc aux élèves de reprendre le même travail que précédemment avec des dessins comportant des motifs « répétitifs » tels que celui ci-contre.
Les nouveaux dessins proposés obligent les élèves à utiliser un grand nombre de cartes (ce qui rend l'écriture et la lecture de l'algorithme fastidieuse). Certains groupes sont même limités par le nombre de cartes dont ils disposent.



Face à ces nouvelles contraintes, les élèves n'ont pas d'autre solution que de créer de nouvelles cartes :

- s'apparentant, pour certaines, à des procédures informatiques ;
- permettant, pour les autres, de réitérer des instructions (les boucles).

C'est à cela qu'on veut les amener.

Pour la seconde catégorie, c'est une carte « `Répéter... fois` » qui est envisagée[4]. Certains l'utilisent après la liste d'instructions à répéter (cf. l'annexe p. 51), comme le sens commun du mot (on ne répète que ce qui a été fait déjà une fois) ; d'autres le placent avant la liste d'instructions, comme pour « `Faire... fois` ».

Après discussion et pour remédier au problème de vocabulaire lié au mot « répéter » tout en se servant de leur idée naturelle, il sera décidé de créer une nouvelle carte `Faire ... fois` avec la possibilité d'écrire le nombre de répétitions voulu à l'intérieur. Celle-ci devra alors être placée avant les cartes à répéter, listées en dessous avec un léger décalage[5].

On aurait pu créer une carte « `Pour` i `allant de 1 à` n » se rapprochant de la syntaxe informatique, mais celle-ci s'éloigne de la proposition naturelle des élèves. De plus, la notion de variable n'ayant pas encore été abordée, l'utilisation du compteur i dans cette instruction serait « forcée » et constituerait ici davantage un obstacle qu'une avancée dans la compréhension de ce qu'est une boucle.

Mais avant d'aborder ces nouvelles difficultés, il faut passer de l'algorithmique débranchée à la programmation.

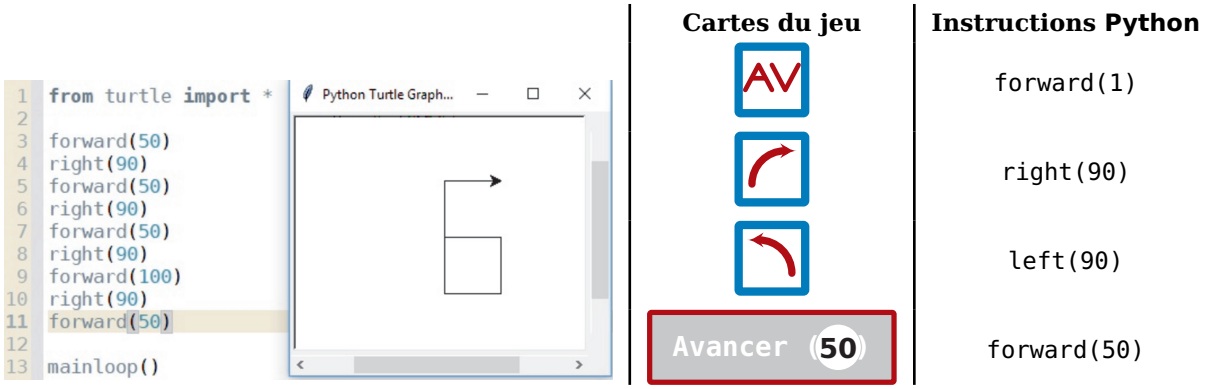
4. Les élèves sont peut-être influencés par leur travail avec Scratch.
5. Pour rendre bien lisible le début et la fin de la structure boucle et donc bien identifier les instructions à répéter.

Premières traductions en Python

Le temps passé à utiliser le jeu de cartes a toujours eu pour but d'écrire des algorithmes à implémenter dans un langage textuel.

Ce moment n'est alors qu'une simple traduction des instructions représentées jusqu'alors par des cartes. Si les élèves ont bien compris l'usage de ces cartes, le passage à un langage textuel tel que Python ne doit pas amener de difficultés importantes. Il faut bien sûr donner la syntaxe de ce langage et la correspondance avec les cartes.

Le lien entre les cartes et les instructions de Python est fait par les élèves eux-mêmes en observant, sur un algorithme simple, l'effet des instructions utilisées. Ils seront amenés également à se rendre compte d'un implicite important dans Python qui consiste à placer le curseur au milieu de l'écran et orienté horizontalement vers la droite.



Cet algorithme ne contient pas de boucle, mais cette notion a déjà été travaillée avec le jeu de cartes. L'instruction « `for i in range(1,6):` » sera donnée à la suite de cet exemple, à la demande des élèves.

Variables et affectations

Avec le travail réalisé « en débranché », les élèves se sont familiarisés avec la structure d'un algorithme. Mais seules des situations de construction de motifs leur ont été proposées. La notion de variable n'a donc jamais été évoquée. C'est, là encore, une difficulté qu'il ne faut pas négliger. On peut sans trop de difficulté prendre l'image d'une « case mémoire »[6] (dans l'ordinateur) dans laquelle on stocke une valeur. La difficulté intervient quand on veut affecter une valeur à cette variable, et surtout prendre la valeur d'une variable, lui faire subir un calcul avant de remettre le résultat dans cette même variable. En effet, dans un algorithme, il y a ambiguïté entre la case mémoire et la valeur qui y est stockée.

En langage naturel	En langage informatique (Python)
variable prend la valeur variable $\times$ 10	`variable` = `variable` $\times$ 10

Il est important d'être bien clair sur la différence entre la valeur et le nom de la variable dans laquelle elle va être rangée. La difficulté est accentuée dans Python puisque l'affectation se fait avec le signe = qui a alors un statut différent de ceux du signe = en mathématique connus des élèves (équation, définition, identité).

6. La meilleure image serait sûrement celle de valeur étiquetée avec un nom (un post-it sur une valeur) mais cela nous semblait plus compliqué pour des débutants.

Un jeu de rôle pour comprendre le principe d'affectation

Pour permettre aux élèves de bien différencier le concept difficile de variable et ces deux aspects, l'IREM de Poitiers propose d'utiliser un jeu de rôle. Les élèves vont se mettre à tour de rôle dans la peau d'un banquier chargé de calculer chaque année la nouvelle somme sur un compte épargne rémunéré à 2 % par an, auquel on ajoute une somme fixe (200 €) à chaque anniversaire de l'ouverture du compte.

Le professeur propose l'ouverture du compte bancaire d'un nouveau-né et le simule à l'aide d'une enveloppe sur laquelle il écrit « Épargne » (c'est le nom de la **variable**) dans laquelle il dépose un papier avec la somme initiale de 1 000 € dessus (première opération d'**affectation**). Chaque élève (mis par groupe de six pour estimer la somme acquise au bout de six ans) doit alors, chacun à son tour :

- ouvrir l'enveloppe, prendre le papier et lire la valeur écrite dessus
  (cela revient à prendre la valeur de la variable Épargne) ;
- et le remplacer par un autre papier sur lequel figure le montant du compte après un an de placement : Épargne $\leftarrow$ Épargne $\times 1{,}02 + 200$
  (cela revient à affecter une nouvelle valeur à la variable).

Une fois le jeu de rôle terminé, un point est réalisé en grand groupe, sur les différentes étapes effectuées et une traduction possible dans Python. Le parallèle entre l'expérimentation et le programme informatique est alors possible afin que les élèves implémentent l'algorithme dans l'ordinateur.

Algorithme	**Étapes**	**Traduction en Python**
Ouvrir le compte et déposer 1 000 €	Initialisation	`Epargne = 1000`
Faire six fois de suite Calculer la nouvelle somme après un an	Affectations	`for i in range(1,7):` `  Epargne = Epargne*1.02+200`
Regarder la somme disponible	Sortie	`print(Epargne)` [7]

Quand le professeur demande ensuite de déterminer la somme disponible sur le compte épargne à la majorité de l'enfant, les élèves n'ont aucune difficulté ; il suffit en effet d'effectuer un changement de la valeur maximale du paramètre de la boucle « `for` » dans le programme.

Introduire la boucle « `While` » et consolider la notion de variable

Pour introduire la boucle « `tant que...` », encore plus complexe que la boucle « `for` », la même situation peut servir. Les élèves n'ont jusqu'alors utilisé qu'une boucle « `for` » connaissant au préalable le nombre d'itérations à effectuer. Pour atteindre une certaine somme, sans connaître le nombre d'itérations nécessaires, les élèves vont devoir réfléchir au moyen de compter le nombre d'affectations effectuées.

La simulation peut alors recommencer avec trois enveloppes (pour trois comptes différents) passant aléatoirement dans un groupe de 18 élèves (demi-classe). Il est alors difficile de compter le nombre de « banquiers » qui voient passer un « compte ».

7. L'instruction de sortie (`print`) n'est plus préconisée ; depuis deux ans, on nous invite à utiliser des fonctions (avec `return`). Le choix de ne pas introduire la notion de fonction est volontaire et assumé. Le premier objectif est de comprendre les affectations, les répétitions et, bien sûr, la structure même d'un algorithme avant de se lancer dans l'utilisation d'une fonction dans un algorithme.

À la fin de l'expérimentation, les élèves doivent répondre à la question initiale : « Combien d'années ont été nécessaires pour atteindre 5 000 € sur chacun des trois comptes ? »

Vu le nombre de transactions et le cumul d'enveloppes, ils ne peuvent pas être certains de la réponse. Ils sont alors amenés à comprendre la nécessité d'introduire une seconde variable (à augmenter d'une unité à chaque transaction), représentée par une enveloppe « Année », pour compter le nombre de transactions.

Une fois le jeu terminé, on peut faire le point sur la structure algorithmique à utiliser puis sa traduction en Python.

Les élèves proposent de manière assez naturelle une structure de boucle sous la forme « **faire... jusqu'à Épargne $\geqslant$ 5 000** ». Ce type de boucle n'étant pas disponible dans Python, le professeur indique la syntaxe d'une boucle équivalente : « **tant que Épargne $<$ 5 000 faire...** ». Il est alors nécessaire de noter que la condition d'arrêt de cette boucle est la négation de celle proposée par les élèves.

Une fois ce problème éclairci, le professeur peut alors donner la syntaxe de la nouvelle instruction.

Algorithme	**Étapes**	**Traduction en Python**
Ouvrir le compte et déposer une somme Initialiser l'enveloppe Année	Initialisation	`Epargne = 2000` `Annee = 0`

Faire jusqu'à obtenir 5 000 €
Calculer la nouvelle somme après un an
Augmenter la variable Année de un
	Affectations itérées	`While Epargne < 5000:`
		`Epargne = Epargne*1.02+200`
		`Annee = Annee+1`

Regarder la somme disponible Regarder le nombre d'années passées	Sortie	`print(Epargne)` `print(Annee)`

Comme la première fois, le passage à Python n'est alors qu'une affaire de traduction et de syntaxe.

Un jeu pour tout réinvestir

Dans toute utilisation d'un langage informatique textuel, il est difficile pour un élève de travailler directement sur machine. Pour comprendre la nécessité d'une boucle, d'une variable et de son utilisation, il est important de réfléchir à l'algorithme sur papier avant de se lancer sur la machine.

Afin de réinvestir tout ce qui a été étudié précédemment, on demande aux élèves d'écrire un programme Python permettant à un utilisateur de jouer contre l'ordinateur à un jeu que l'on appellera « le juste prix ».

Pour trouver un nombre entier compris entre 1 et 1 000 choisi au hasard par l'ordinateur, le joueur devra faire des propositions successives en tenant compte des réponses de l'ordinateur du type « trop grand » ou « trop petit ». Pour bien comprendre les étapes importantes de ce jeu, l'idée est de faire jouer le rôle de l'ordinateur par un élève et le rôle de l'utilisateur par un autre. Un troisième élève observe le jeu de l'extérieur et doit être capable d'expliciter les différentes étapes ; c'est un rôle difficile car l'objectif n'est pas d'analyser la stratégie du joueur mais bien les étapes du jeu.

Les élèves énoncent facilement les étapes telles que le choix au hasard d'un nombre (étape 1), la proposition du joueur (étape 2) puis la réponse de l'ordinateur (étape 3) et la répétition des deux

dernières. Cependant, il est nécessaire de leur poser des questions sur le rapport entre l'étape 2 et l'étape 3. Les élèves découvrent alors la nécessité d'une condition permettant à l'ordinateur de donner une réponse plutôt que l'autre.

Comme pour le jeu précédent, les élèves voient naturellement une répétition du type « **faire... jusqu'à ce que...** », qu'ils ont déjà été amenés à transformer en une boucle « **tant que... faire...** » et à réfléchir à la condition d'arrêt.

Les différentes étapes du jeu étant énoncées, les élèves sont chargés d'écrire en langage naturel l'algorithme permettant d'y jouer. À la demande des élèves, le professeur traduit alors les instructions qu'ils n'ont pas déjà rencontrées ; cinq questions sont alors formulées :

- *comment faire choisir un nombre au hasard par l'ordinateur ?*
  L'instruction `random.randint(1,1000)` nécessite d'importer le module `random` ; cela ne gêne pas les élèves qui ont déjà utilisé précédemment le module de construction `turtle` ;
- *comment demander une valeur à l'utilisateur ?*
  L'instruction `int(input("Donner un nombre:"))` se décompose en :
  * `input("Donner un nombre")` qui écrit la question sur l'écran et attend une réponse de l'utilisateur,
  * `int(...)` qui donne le statut d'entier à cette réponse ;
- *quel est le symbole « différent » pour la condition d'arrêt de la boucle ?*
  `!=`
- *comment introduire une condition ?*
  On utilise l'instruction `if «condition:»` suivie d'un retour à la ligne avec une indentation ;
- *comment faire afficher un résultat dans la console (fenêtre de discussion entre Python et l'utilisateur) ?*
  La commande `print("...")` permet d'afficher dans la console un texte.

Cette étape ne peut être réalisée que pour les groupes d'élèves qui ont su bien analyser le jeu de rôle et transcrire en langage naturel les différentes étapes du jeu. Certains ont eu besoin d'aide pour surmonter les difficultés : boucle, conditions et affectation.

Conclusion

Cette initiation à l'algorithmique et à la programmation en langage Python aura nécessité six ou sept heures selon les classes.

Elle aura permis d'introduire les boucles bornées, les variables et les conditions mais aussi de mettre en évidence les difficultés rencontrées par les élèves et de conforter un certain nombre d'*a priori* des expérimentateurs :

- la programmation dans un langage textuel n'est pas seulement affaire de codage. Plus que la connaissance de la syntaxe du langage, il apparaît indispensable de maîtriser les différentes structures algorithmiques mises en jeu pour l'obtention du but visé par le programme ;
- un travail de réflexion sur papier s'avère nécessaire avant tout passage sur machine.

L'utilisation du jeu de cartes et la pratique des jeux de rôle ont donné aux élèves la possibilité de se construire un savoir algorithmique stable, à partir de leurs représentations et de leurs intuitions. L'écriture en langage textuel n'est plus alors qu'une affaire de traduction.

Pour tous les élèves, y compris ceux ayant travaillé avec Scratch au collège, ce travail de fond n'a pas été inutile. Néanmoins, il faudra observer dans les années à venir comment évolueront les connaissances algorithmiques des collégiens pour adapter ce type d'initiation en début de classe de Seconde.

Référence

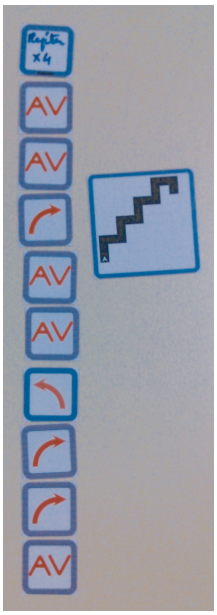
[1] Groupe Lycée. « Algorithmique et programmation en Seconde ». In : *Mathématiques vivantes au lycée*. IREM de Poitiers, septembre 2018. ISBN : 978-2-85954-100-2.

Cyrille Kirch et Olivier Jutand sont professeurs au lycée du Bois d'Amour de Poitiers (86) et membres du groupe Lycée de l'IREM de Poitiers.

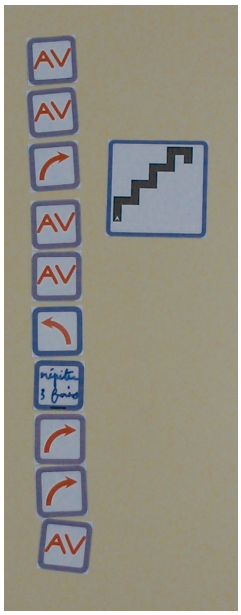
olivier.jutand@ac-poitiers.fr
cyrille.kirch@ac-poitiers.fr

© APMEP *Septembre 2019*

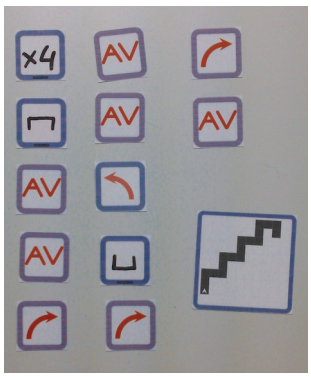
Annexe : productions d'élèves



« `Répéter` » avant une liste d'instructions non délimitée.



« `Répéter` » après la liste d'instructions non délimitée.



« `Faire 4 fois` » une liste précise et claire d'instructions.

Journées nationales de l'APMEP

La Saveur des Mathématiques

De la maternelle à l'université

les 19-20-21-22 octobre 2019

DIJON

infos: www.apmep.fr

design graphique©Muriel Carpentier

Sommaire du n° 533

Mathématiques et mouvement

Éditorial 1

Opinions 3

Des pistes pour sortir de la crise de l'enseignement des sciences — Gilles Dowek 3

Les labos de maths — Valérie Larose 6

L'Observatoire EVAPM, une aventure de l'APMEP — Antoine Bodin 8

Avec les élèves 16

Mouvement mathématique en Bretagne — Claudie Asselain-Missenard 16

Coup de cœur pour une appli — Isabelle Audra 21

Sprint ! — Romain Estampes 23

Histoire de ~~boules~~ Boole — Agnès Veyron 27

Mesure du flux de muons cosmiques — Luca Agostino 33

Les 6e ne manquent pas d'aire ! — Anne Dusson & Nathalie Lecouturier 39

Algorithmique débranchée — Cyrille Kirch & Olivier Jutand (groupe Lycée de l'IREM de Poitiers) 43

Ouvertures 52

Mat'les ressources : un journal pour des ressources — Vincent Bansaye, Alain Camanes & Daphné Giorgi 52

Le transport optimal numérique — Gabriel Peyré 55

Sauver Walu, une aventure ! — Dominique Cambrésy 65

Variations autour d'une formule — Attila Máder & Zoltán Matos 69

Mathématiques du jonglage — Vincent Pantaloni 74

Récréations 83

Au fil des problèmes — Frédéric de Ligt 83

La coupe du monde de rugby — Michel Soufflet 85

Au fil du temps 88

Matériaux pour une documentation 88

Anniversaires — Dominique Cambrésy 94

CultureMATH

APMEP

www.apmep.fr