Enseignement des fonctions algorithmiques en classe de seconde

Mélanie Cornillac, Sébastien Detroyat

Année universitaire 2018-2019

Master *Métiers de l'enseignement, de l'éducation et de la formation*
Mention *Second degré mathématiques*

Enseignement des fonctions algorithmiques en classe de seconde

Présenté par Mélanie Cornillac,
en collaboration avec Sébastien Detroyat

Mémoire de M2 encadré par Hamid Chaachoua

Table des matières

Partie I. Partie théorique

1. Introduction

Depuis son entrée dans les programmes, l'algorithmique prend une part de plus en plus importante dans le cadre de la classe de mathématiques en seconde générale. La transmission aux élèves de la pensée informatique devient un enjeu pour les professeur de mathématiques. Les élèves découvrent un nouvel "outil avec lequel penser" (Buteau, Muller, Sacristàn & Mgombelo, 2018) ce qui est l'un des changements majeurs de ce début de XXIème siècle dans l'enseignement des mathématiques.

La partie algorithmique et programmation du programme de seconde est articulée autours de différents points, l'un d'entre eux étant la fonction en algorithmique.

Nous nous sommes interrogés dans cette étude sur la manière dont cela est introduit aux élèves de classe de seconde. Les manuels traitent cela grâce à une analogie avec les fonctions mathématiques, et nous nous sommes demandés si une approche purement algorithmique permettrait aux élèves de s'approprier cette notion, dont les enjeux sont sensiblement différents de ceux des fonctions mathématiques.

Pour cela, nous avons dans un premier temps effectué une étude des programmes et manuels, ainsi qu'une étude épistémologique des fonctions pour en saisir tous les enjeux tant en mathématiques qu'en algorithmique. Après un comparatif, nous présentons la mise en place d'un séquence avec des élèves de seconde ciblant les enjeux algorithmiques des fonctions: la structuration et la factorisation d'un programme. Enfin, nous présentons les résultats issus de notre expérience et ce que nous en retenons pour notre enseignement.

2. Etat de l'art

2.1.Etude épistémologique de la notion de fonction

2.1.1 En mathématiques

2.1.1.1 Définition actuelle

En mathématiques, une fonction est définie comme étant une relation liant deux ensembles, chaque élément de l'ensemble de départ étant lié à au plus un élément de l'ensemble d'arrivée. Nous n'aborderons pas dans ce document la notion de fonction multivaluée.

2.1.1.2 Historique

Les informations de cette section sont majoritairement tirées de la page Wikipédia Fonction et de la page sur l'histoire des fonctions repérée sur le site Maths93.

Premières notions de fonctions

Nous avons retrouvé chez les babyloniens des tables de calculs (carrés, cubes, racines cubiques, multiplications et autres). Dans la grèce antique, des tables de sinus ont aussi été retrouvées dans l'almageste de Ptolémée. De plus, la fraternité pythagoricienne cherchait déjà des relations entre la hauteur des sons émis par des cordes pincées et la longueur de ces cordes.

Néanmoins, la relation permettant de passer d'une colonne à l'autre de ces tables n'est jamais considérée, ce ne sont que des tables de valeurs numériques utilisées dans divers calculs. A ce titre, nous ne pouvons pas parler de fonctions à proprement parler. La question de quantités variables ou de loi de variations n'est pas non plus abordée.

Les écoles de philosophie naturelle d'Oxford et de Paris au XIVème siècle, à travers l'étude des

mouvements de solides, proposent de modéliser les phénomènes physiques et mettent en évidence des relations entre vitesse, force, temps et résistance de façon géométrique.

Des fonctions du temps apparaissent, ainsi que les concepts de relations et de quantités variables.

C'est le mathématicien français Viète qui fait apparaître les formules en se ramenant au calcul littéral. Alors que les fonctions étaient uniquement associées à des courbes, elles sont désormais liées à des formules. Un exemple serait la célèbre formule de Galilée en 1623 qui propose ses lois sur la chute des corps:

$$z(t) = -\frac{1}{2}gt^2$$

Les fonctions ne servent pour l'instant qu'à modéliser des trajectoires de points en mouvement.

Premières définitions

René Descartes propose la première véritable définition de l'objet fonction:

"*Est fonctionnelle pour Descartes, une relation qui permet de faire correspondre à une longueur donnée, une autre longueur déduite de la première par un nombre fini d'opérations algébriques*".

Jules Vuillemin (1920-2001)

Grâce à cela, il définit une classification des courbes en deux types: les courbes géométriques, où les coordonnées sont reliées par une équation polynomiale, et les courbes mécaniques (ou transcendantes), comme par exemple le logarithme, que Descartes rejette. Une fonction est donc associée et définie à ce moment là par une équation, malgré le problème que pose les courbes transcendantes. Ses successeurs, par la découverte les séries de puissances infinies, élargissent cette définition et permettent de représenter par exemple le logarithme.

C'est la définition de James Gregory que nous retiendrons du 17ème siècle:

"Une fonction est définie comme une quantité obtenue à partir d'autres quantités par une succession d'opérations algébriques ou par n'importe quelle opération imaginable."

Vera circuli et hyperbolae quadratura, 1667.

Introduction du terme de fonction par Leibnitz

C'est au mathématicien allemand Gottfried Wilhelm Leibniz à qui l'on doit l'apparition du terme fonction:

"J'appelle fonctions toutes les portions des lignes droites qu'on fait en menant des droites indéfinies qui répondent au point fixe et aux points de la courbe; comme sont les abscisse, ordonnée, corde, tangente, perpendiculaire, sous-tangente ...et une infinité d'autres d'une construction plus composée, qu'on ne peut figurer." La Méthode inverse des tangentes ou à propos des fonctions, 1673 Gottfried Wilhelm Leibniz

On note que cette définie se base uniquement sur une approche des fonctions par leur courbe représentative, en liant à chaque point un objet (tangente, perpendiculaire…).

Redéfinitions par Bernoulli et Euler et introduction de la notation Φx

Jean Bernoulli et Euler, en identifiant chaque point de la courbe avec son ordonnée, redéfinissent le terme de fonction pour décrire une expression composée d'une variable et d'éventuels paramètres constants. Bernoulli propose la définition suivante:

"*On appelle fonction d'une grandeur variable une quantité composée, de quelque manière que ce soit, de cette grandeur variable et de constante*." 1718, Jean Bernoulli

C'est aussi à ce moment là que l'on voit apparaître la notation Φx.

Leonhard Euler propose quand à lui une 3ème définition de l'objet fonction:

"Une fonction est une expression analytique composé d'une manière quelconque de cette quantité variable et de nombres ou de quantités constantes."

Introductio in analysin infinitorum, Marcum-Michaelem Bousquet & socios, 1748, Leonhard Euler

Pour Euler, une fonction est donc obtenue par combinaison d'opérations et de méthodes de calculs applicables aux nombres. Il en déduit une classification des fonctions en deux catégories:

- Les fonctions algébriques, obtenues par combinaison d'opérations algébriques.

- Les fonction transcendantes, obtenues grâces à des opérations répétées à l'infini (trigonométrie, logarithmes, exponentielles, intégrales, puissances irrationnelles).

Définition explicite

De par des contre-exemples, cette définition va être mise en défaut. Gustav Peter Lejeune Dirichlet donne par exemple la fonction caractéristique des irrationnels (valant 0 si x est rationnel, 1 sinon). Apparaît alors le besoin d'un définition plus générale: la correspondance arbitraire entre deux nombres.

2.1.2 En algorithmique

2.1.2.1 Définition

Une fonction est une suite ordonnée d'instructions qui retourne une valeur (bloc d'instructions nommé et paramétré).

2.1.2.2 Historique

Cette section s'appuie principalement sur la page du site Scriptol: Histoire et évolution des langages de programmation, ainsi que sur le support de cours de Tisseau: Initiation à l'algorithmique, procédure et fonctions, 1. Définition d'une fonction.

Les premières fonctions algorithmiques sont apparues en 1954 dans le langage Fortran (FORmula TRANslator system), language dédié aux calculs mathématiques. La récursivité (appel d'une fonction dans son corps) a été introduite en 1956 dans le langage IPL (Information Processing Language). Les premières méthodes sont apparues dans le langages Simula 67 en 1962, cela permet de lier des fonctions à des objets.

Dans certains langages, on distingue procédures et fonctions. La différence, c'est que la fonction retourne un résultat au programme appelant alors que la procédure non, c'est un sous programme qui remplit une tâche précise.

2.1.2.3 Qu'est-ce qu'un algorithme?

Nous avons défini plus haut la notion de fonction mathématique. Une fonction est calculable s'il existe un algorithme qui peut la calculer. Reste à donner une définition mathématique précise de ce qu'est un algorithme, c'est ce qu'étudie Rapaport (2004, p. 236 à 245).

Avant même que quiconque tente de définir la notion d'algorithme, de nombreux algorithmes étaient utilisés par les mathématiciens, comme les procédures d'Euclide pour construire à la règle et au compas des objets géométriques, l'algorithme d'Euclide pour calculer le plus grand commun diviseur de deux entiers. Les algorithmes étaient aussi utilisés par les personnes ordinaires comme les babyloniens avec les opérations arithmétiques basiques.

Différentes caractérisations précises de la notion d'algorithme ont été proposées récemment, qui peuvent aussi bien s'appliquer à la notion de fonction algorithmique puisque une fonction est un algorithme. Voici celle proposée par Donald Knuth (1973). Il dit qu'un algorithme est "un ensemble fini de règles qui donne une séquence d'opérations pour résoudre un type spécifique de problème" (traduction par nos soins), avec "cinq caractéristiques importantes":

1. "Finitude. Un algorithme doit toujours se terminer après un nombre fini de pas."
2. "Clarté. Chaque pas doit être précisément défini, les actions à réaliser doivent être spécifiées de façon rigoureuse et non ambiguë."
3. "Entrée. Un algorithme a zéro ou plus d'entrées."
4. "Sortie. Un algorithme a une ou plusieurs sorties."
5. "Efficacité. Cela signifie que toutes les opérations à réaliser dans l'algorithme doivent être suffisamment basiques pour qu'elles puissent en principe être réalisées exactement et en un temps fini par un homme utilisant un crayon et une feuille."

2.2. Etude des programmes et manuels scolaires

Dans cette section, nous étudions la façon dont la notion de fonction mathématique et algorithmique sont introduites dans les programmes officiels du secondaire et dans quelques manuels. Nous étudions aussi plus largement comment est traitée la partie Algorithmique et Programmation dans laquelle s'inscrit la notion de fonction algorithmique. Ceci afin de situer où

en sont les élèves de seconde avec ces différentes notions quand nous abordons avec eux la notion de fonction algorithmique.

2.2.1. La notion de fonction au cycle 4

2.2.1.1. La notion de fonction mathématique dans le programme de cycle 4

Nous étudions le programme de cycle 4 en vigueur à partir de l'année 2018-2019. Le programme de mathématiques est structuré selon cinq thèmes : nombres et calculs ; organisation et gestion de données, fonctions ; grandeurs et mesures ; espace et géométrie ; algorithmique et programmation qui entre dans le cadre d'un enseignement de l'informatique dispensé conjointement en mathématiques et en technologie.

La notion de fonction mathématique est introduite dans le thème *Organisation et gestion de données, fonctions*. Les attendus de fin de cycle sont de comprendre et utiliser une fonction.
Sont ensuite détaillées les connaissances et compétences associées (voir figure 1).

Connaissances

- vocabulaire : variable, fonction, antécédent, image ;
- différents modes de représentation d'une fonction (expression symbolique, tableau de valeurs, représentation graphique, programme de calcul) ;
- notations $f(x)$ et $x \mapsto f(x)$;
- fonction linéaire, fonction affine.

Compétences associées

- passer d'un mode de représentation d'une fonction à un autre ;
- déterminer, à partir d'un mode de représentation, l'image ou un antécédent d'un nombre par une fonction ;
- représenter graphiquement une fonction linéaire, une fonction affine ;
- modéliser un phénomène continu par une fonction ;
- modéliser une situation de proportionnalité à l'aide d'une fonction linéaire ;
- résoudre des problèmes modélisés par des fonctions.

Figure 1: Connaissances et compétences du thème Organisation et gestion de données, fonctions (Extrait du programme de cycle 4 applicable à la rentrée 2018-2019)

En 5ème, la rencontre de relations de dépendance entre grandeurs mesurables, ainsi que leurs représentations graphiques, permet d'introduire la notion de fonction qui est stabilisée en 3ème, avec le vocabulaire et notations correspondantes.

Dans le thème *Algorithmique et programmation*, il est indiqué qu'en s'initiant à la programmation les élèves revisitent les notions de variables et de fonctions sous une forme différente. L'introduction des fonctions algorithmiques n'est pas prévue dans le programme de Cycle 4, elle est tout de même abordée dans les manuels sous forme de bloc d'instructions ou même de fonction comme nous le détaillons dans la partie suivante.

2.2.1.2. La notion de fonction mathématique dans les manuels de cycle 4

La notion de relation de dépendance entre valeurs mesurables est rencontrée en 5ème, ce qui permet de formaliser la notion de fonction en 3ème, nous avons étudié ce qui est proposé dans deux manuels sur les fonctions en 3ème.

Nous étudions d'abord le manuel Myriade Cycle 4 programme 2016, livret de cours. Dans le thème Expressions littérales - Fonctions, une définition de la notion de fonction est proposée: "le processus qui, à un nombre, fait correspondre un autre nombre unique s'appelle une fonction".

Pour faire comprendre la notion de fonction dans la partie exercices, l'élève doit écrire une fonction à partir d'un programme de calcul (ce qui rapproche la notion de fonction mathématique de celle d'algorithme), puis exprimer l'aire d'un carré de côté x. Les notations et la représentation graphique d'une fonction sont ensuite introduites Puis les notions d'image et d'antécédent sont formalisées et les techniques pour les calculer dans différents cadres sont introduites (à partir de la courbe représentative de la fonction, d'un tableau de valeurs, de son expression algébrique).

Etudions maintenant le manuel Sésamaths cycle 4 programme 2016. La définition proposée dans le cours est très proche du premier manuel: "Une fonction est un procédé qui, à un nombre donné, associe un unique nombre. On peut donner le procédé sous la forme d'une expression littérale."

Dans les exercices, il est là aussi proposé à l'élève de passer d'un programme de calcul à l'expression algébrique d'une fonction et vice versa (changement de cadre), et de calculer le

périmètre d'un rectangle ou d'un carré par exemple en guise d'exercices de découverte des fonctions.

2.2.1.3. La notion de fonction algorithmique dans les manuels de cycle 4

Bien que le programme de cycle 4 n'évoque pas l'introduction de la notion de fonction algorithmique, elle est naturellement introduite dans deux manuels que nous étudions.

Dans Myriade Cycle 4, une des cinq séquences du livret d'algorithmique et programmation travaille spécifiquement sur l'utilisation d'un bloc d'instructions, "un bloc permet de mémoriser un petit algorithme intervenant dans un algorithme plus élaboré, éventuellement plusieurs fois, ce qui facilite également la compréhension de l'algorithme principal". Il s'agit bien de la notion de fonction informatique bien qu'elle ne soit pas nommée ainsi. La notion est introduite en débranché puis avec Scratch, par exemple avec des constructions de motifs géométriques. La notion de bloc est ensuite utilisée dans le projet "chiffre de César" avec un bloc Coder qui code une lettre.

Dans Sésamaths, la dernière partie du thème Algorithmique et programmation s'intitule "Les fonctions et procédures". Une définition est proposée "Les fonctions et procédures sont des "morceaux de programme" que l'on peut appeler en leur indiquant des paramètres. On réutilise souvent la même partie d'un programme. Au lieu de réécrire cette partie, on en fait une fonction ou une procédure." Les auteurs semblent vouloir introduire la nuance entre fonction (qui renvoie une valeur) et procédure (qui ne renvoie pas de valeur), nous trouvons cela très prématuré en Cycle 4, surtout que cette nuance n'est pas illustrée dans les exemples proposés. Les exercices d'entraînement proposent pour la plupart de tracer des forme géométriques (afficher 10 carrés,...). Dans les projets, la notion de fonction est utilisée dans le projet cryptographie de César (codage et décodage).

Pour vérifier si ce choix d'introduire les fonctions algorithmiques en Cycle 4 est généralisé, nous observons d'autres manuels. Ce n'est pas le cas ni dans le manuel maths cycle 4, collection Kiwi, programme 2016, ni dans Delta mathématiques cycle 4 programme 2016, la notion de fonction algorithmique n'y est abordée sous aucune forme, même dans les exercices avancés de programmation.

Donc dans les cours de seconde, le professeur doit considérer que la notion de fonction algorithmique est nouvelle pour les élèves.

2.2.2. L'algorithmique et programmation en seconde

2.2.2.1 L'algorithmique et programmation dans les programmes de seconde

Dans les programme de mathématiques de seconde de juillet 2009, sont donnés pour l'algorithmique des objectifs pour le lycée, il n'y a pas de programme par année. Les capacités attendues en algorithmique sont considérées comme transversales, au même titre que celles attendues en raisonnement, et doivent être développées à l'intérieur de chacune des trois parties du programme de mathématiques : Fonctions, Géométrie, Statistiques et probabilités.

Dans les aménagements de programme d'enseignement de mathématiques de seconde générale et technologique, applicable à compter de la rentrée 2017, l'algorithmique et la programmation devient une partie à part entière du programme de mathématiques. Il est toujours indiqué que le travail sur l'algorithmique et la programmation doit être réinvesti dans les trois autres parties (Fonctions, Géométrie, Statistiques et probabilité).

Dans les futurs programmes de mathématiques applicables à la rentrée 2019 (parus en janvier 2019), l'algorithmique et programmation reste une partie à part entière du programme. La classe de seconde a pour objectif de consolider les acquis du cycle 4 autour de deux idées essentielles :
- la notion de fonction,
- la programmation comme production d'un texte dans un langage informatique.

Pour la plupart des chapitres du programme, des exemples d'algorithmes à aborder avec les élèves sont indiqués.
Et si nous regardons la suite du programme de lycée, nous constatons que l'algorithmique est là encore réinvestie dans les différents chapitres de mathématiques (le futur programme de terminale n'est pas encore paru):

« L'enseignement de spécialité de mathématiques de classe de première vise la consolidation des notions de variable, d'instruction conditionnelle et de boucle ainsi que l'utilisation des fonctions. La seule notion nouvelle est celle de liste qui trouve naturellement sa place dans de nombreuses parties du programme et aide à la compréhension de notions mathématiques telles que les suites numériques, les tableaux de valeurs, les séries statistiques...

[...]

L'accent est mis sur la programmation modulaire qui permet de découper une tâche complexe en tâches plus simples» (extrait du BO de janvier 2019)

2.2.2.2. Traitement de l'algorithmique et programmation dans des manuels de seconde

Nous observons comment est traitée cette partie dans les manuels mettant en œuvre les aménagements de programme de 2017 où une partie Algorithmique et programmation est créée. De nombreux éditeurs ont aussi édité à cette occasion des cahiers d'algorithmique.

Nous étudions 2 manuels: Hyperbole 2de édition 2017 et Indice 2de édition 2017.

Les 2 manuels proposent un chapitre complet dédié à l'algorithmique et programmation, voir l'index des ces chapitres dans la figure 2.

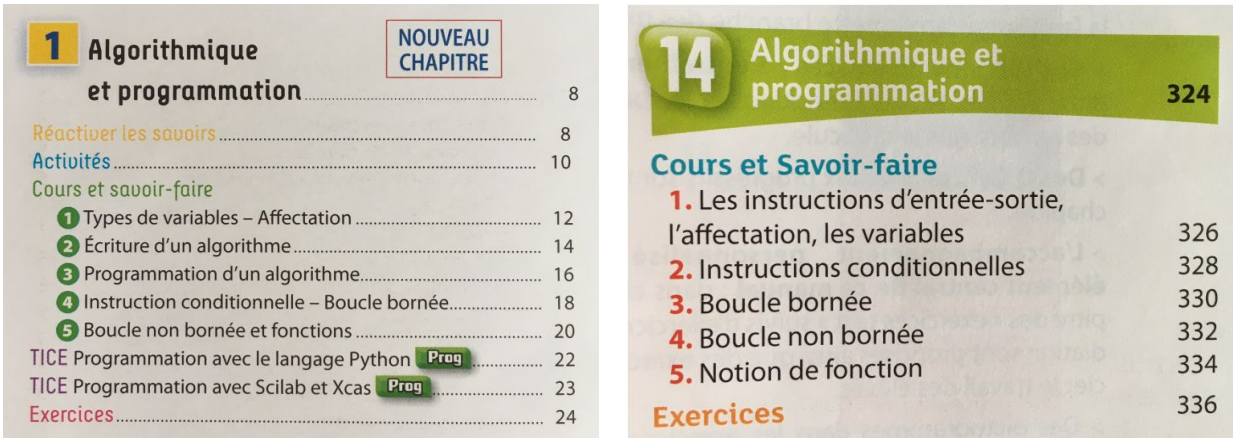


Figure 2: Index du chapitre Algorithmique et programmation dans le sommaire du manuel Indice 2de 2017 (à gauche) et Hyperbole 2de 2017 (à droite)

Le manuel Hyperbole propose aussi une progression annuelle qui alterne les chapitres relatifs aux fonctions, à la géométrie et aux probabilités et statistiques. Le chapitre 14 Algorithmique et Programmation ne figure pas dans cette progression car au sens des auteurs “il n’est pas à traiter d’un seul bloc mais sert plutôt de fil directeur: dans chaque chapitre, pour les exercices relevant de l’algorithmique, un renvoi indique la partie utile du chapitre 14.”, ce qui correspond aux préconisations du programme 2017.

Les cahiers d’algorithmique de seconde se structurent soit autour des notions de bases en programmation Python (variables, affectations, instructions conditionnelles, boucles bornées, boucles non bornées, fonctions), soit autour des chapitres de mathématiques du programme de seconde en proposant des exercices d’algorithmique et programmation en lien avec ces chapitre.

Le cahier d’algorithmique Indice 2de 2017 propose une structuration mixte, avec d’abord les notions de base d’algorithmique puis des exercices liés aux parties mathématiques du programme. Les compétences signalées dans les exercices recoupent celles proposées par Lac et More (2017): **comprendre** un algorithme; **modifier** un algorithme; **compléter** un algorithme; **analyser** une situation; **écrire** un algorithme; **programmer** un algorithme.

2.2.3. La notion de fonction en seconde

2.2.3.1. La notion de fonction mathématique dans le programme de seconde

Le programme 2009 de seconde est divisé en trois partie: Fonctions, Géométrie, Statistiques et probabilités. La notion de fonction est donc très centrale en seconde. Le programme applicable à la prochaine rentrée garde cette place centrale pour les fonctions, et ajoute comme pour les autres parties du programme une partie histoire des mathématiques: “On peut évoquer la très lente élaboration de la notion de fonction, depuis l’Antiquité jusqu’à la codification actuelle par Dirichlet, en mettant en évidence quelques étapes importantes: Newton, Leibniz, Euler. On souligne alors l’importance de la notation algébrique.“ Les objectifs de la classe de secondes sont de consolider la notion de fonction, comme exprimant la dépendance d’une variable par rapport à une autre; exploiter divers registres, notamment le registre algébrique et le registre graphique; étendre la panoplie des fonctions de référence; étudier les notions liées aux variations et aux extremums des fonctions.

2.2.3.2. La notion de fonction mathématique dans les manuels de seconde

Dans Indice 2017, la définition de la notion de fonction numérique donnée est:
“Soit D un sous-ensemble de R. On définit une fonction f sur D en associant à chaque nombre réel x de D un nombre réel et un seul noté f(x).” En s’appuyant sur la formalisation déjà effectuée au cycle 4, on peut directement proposer aux élèves cette définition qui va un peu plus loin dans le formalisme. C’est ici la notion d’intervalle qui est nouvelle et détaillée auparavant.
Dans Hyperbole 2017, une définition similaire est proposée “À tout nombre x d’une partie D de R, une fonction f associe un nombre et un seul que l’on note f(x).”

2.2.3.3. L’introduction de la notion de fonction en algorithmique dans les manuels de seconde

Dans le manuel Hyperbole 2017, l’introduction des fonctions algorithmiques est faite en se basant sur les fonctions mathématiques, la première fonction présentée s’appelle f(x) et renvoie une expression algébrique calculée à partir de x. Cela nous semble peu pertinent car cela entretient la confusion entre fonction mathématique et fonction algorithmique.
Le premier exercice propose lui aussi de simplement implémenter une fonction définie par morceaux dans un programme.
Il y a ensuite des exercices avec de vraies fonctions au sens algorithmique :
- maximum de 3 nombres réels,
- distance entre 2 points d’une droite graduée,
- fonction qui à partir des 3 longueurs d’un triangle renvoie « Vrai » si le triangle est rectangle, « Faux » sinon (ce qui permet de s’abstraire des difficultés liées à la logique booléenne),
- fonction qui associe à une température en degré Celsius sa valeur en degré Fahrenheit,
- fonction qui dénombre les diviseurs d’un entier.

Dans le manuel Indice 2017, là encore les fonctions en programmation sont introduites en évoquant les fonctions numériques étudiées en mathématiques.
Les exercices proposés pour démarrer :
- somme des n premiers entiers naturels,
- volume d’un cylindre à partir de son rayon et sa hauteur,

Ensuite des petits exercices purement algorithmiques (sans mathématiques) ayant pour objectif la compréhension des notions utilisées dans les fonctions sont proposés (questions à partir de fonctions données aux élèves) :

- quels sont les arguments de cette fonction ?
- Que renvoie f(3,1,-2) ?
- Quel est le type de la variable S ?
- Cette fonction comprend 2 instructions return, est-ce correct ? (il y a un return dans un if et un return en dernière instruction)

Cette façon de présenter les fonctions nous semble plus adaptée car centrée sur les notions inhérentes aux fonctions de programmation, en arrivant à s'extraire des mathématiques.

Ensuite, des exercices plus complexes sont proposés (87 à 92 p.33, 99 p.33), plus liés aux autres chapitres de mathématiques maintenant que les bases des fonctions en algorithmique sont maîtrisées :

- somme des cubes des n premiers entiers naturels,
- fonction qui calcule le montant des impôts à partir des revenus (fonction affine),
- découverte de l'algorithme de la division euclidienne : le programme est donné, il est demandé de trouver ce qu'il permet de calculer,
- fonction qui calcule le prix à payer pour une location de skis (avec 2 tarifs selon le nombre de jours de location)
- fonction qui renvoie le minimum de 2 (donnée avec seulement la condition à compléter), puis 4 nombres (en faisant appel à la première fonction),
- moyenne de tous les entiers compris entre deux entiers a et b (squelette à compléter).

L'exercice corrigé 90 sort du lot, il permet de travailler sur les fonctions sans argument, le traitement est purement algorithmique, sans travail sur des notions de mathématiques.

2.3. Synthèse sur les différences entre fonctions mathématiques et fonctions algorithmiques

Nous résumons dans cette partie des différences entre les fonctions mathématiques et algorithmiques.

2.3.1 Différences épistémologique

Le fonctions mathématiques et algorithmiques possèdent des différences de statut, d'origine et de construction et représentation.

Différences de statut

Nous considérons ici la distinction outils/objet discutée par Amra (2003, p.43-46) et introduite par Douady (1986):

"Ainsi, nous disons qu'un concept est outil lorsque nous focalisons notre intérêt sur l'usage qui en est fait pour résoudre un problème. Un même outil peut être adapté à plusieurs problèmes, plusieurs outils peuvent être adaptés à un même problème. Par objet, nous entendons l'objet culturel ayant sa place dans un édifice plus large qui est le savoir savant à un moment donné, reconnu socialement." (Douady, 1986, p. 9).

En mathématiques, les fonctions sont des objets dont on étudie les propriétés (variations, signes…) servant à lier les éléments de deux ensembles entre eux. Elles sont aussi utilisées comme outils pour modéliser et résoudre des problèmes.

En algorithmique, les fonctions sont des outils (aspects problème et effectivité (Modeste, 2012)) permettant :

- De réutiliser un algorithme sans avoir besoin de le réécrire (factorisation du code),
- Une structuration du code: lisibilité et découpage d'un problème en sous-problèmes.

Sa conception comme un objet est établie dans les cas d'études de complexité, d'étude de correction ou de terminaison (preuve), ou de modèle théorique (Modeste, 2012). Cette étude de complexité n'a pas d'équivalent en mathématiques.

Différences d'origine

- En algorithmique, les routines sont décomposées en deux sous-catégories: les fonctions, retournant une unique valeur, et les procédures n'en retournant pas.
- Les fonctions en mathématiques lient chaque élément d'un ensemble de départ à au plus élément d'un ensemble d'arrivée. C'est un cas particulier de relation.

Différences de construction et de représentation

- La récursivité (un algorithme faisant appel à lui même) est un outil répandu en informatique, contrairement aux mathématiques (principalement pour la définition récursive de suites).
- L'instruction d'affectation, essentielle en algorithmique, n'est pas présente en mathématique.
- En mathématiques, une fonction admet plusieurs représentations (expression algébrique, tableau de valeurs, tableau de variations, représentation graphique, programme de calcul) contrairement à l'algorithmique où nous avons seulement le code de la fonction comme représentation.

2.3.2 Différences dans l'enseignement secondaire

Les fonctions mathématiques abordées dans le secondaire sont le cas particulier des fonctions numériques, et sont définies par leurs représentations (Coppe, Dorier, Yavuz, 2006) . Elle sont considérées en tant qu'objets (étude de signes, de variations…) mais aussi en tant qu'outils pour résoudre des problèmes (modélisation, mise en équation, optimisation...).

En algorithmique, la notion de suite d'instructions est plus présente, ce qui permet, contrairement aux mathématiques, de lier des types d'objets différents (un nombre et une chaîne de caractères par exemple). Cela implique aussi la possibilité d'avoir un appel de fonction sans paramètre en algorithmique, contrairement aux fonctions mathématiques. L'étude des fonctions algorithmiques en tant qu'objets (étude de complexité, de terminaison, ...) n'est pas du tout abordée en classe de seconde.

3. Problématique

D'après notre étude de manuels, les fonctions algorithmiques sont abordées dans le secondaires en faisant une analogie avec les fonctions en mathématiques. Or, d'après notre étude épistémologique, le but des fonctions algorithmique n'est pas de lier deux entités comme le font les fonctions mathématiques, mais de pouvoir réutiliser un algorithme et de structurer un programme. Cela nous mène à la problématique suivante:

"Une approche purement algorithmique permet-elle aux élèves de comprendre l'utilisation des fonctions et leur intérêts dans le cadre de l'écriture d'un programme?"

Nous pensons que l'utilisation de nombreuses fonctions, ainsi que la création de certaines d'entre-elles dans le cadre d'une séquence permettra aux élèves de comprendre le fonctionnement (définition puis appel) et l'intérêt de cet outil (factorisation, structuration).
Pour cela, notre séquence aura 3 caractéristiques majeures:
1/ L'écriture d'un programme utilisant de nombreux appels de fonctions pour permettre aux élèves de comprendre la fonctionnement de cet outils.
2/ L'écriture d'un programme nécessitant l'utilisation de nombreux appels de fonctions permettant de facilement factoriser le code produit, et l'utilisation de cette factorisation pour modifier un code existant (une seule modification à faire pour toutes les répétitions) ,pour permettre aux élèves d'intégrer cet enjeu.
3/ Un travail de travail en groupe permettant aux élèves de comprendre l'intérêt d'utiliser les fonctions pour structurer un programme:

- Le découpage en sous-tâches est facilité par la création de fonctions ayant chacune un but défini, et permet aux membres du groupe n'ayant pas travailler sur un certain morceau du code de comprendre et utiliser le travail réalisé par les autres.
- L'exercice de la modification d'un code existant est facilité par la structuration correcte de celui-ci (maintenabilité).

Partie II. Méthode

Pour présenter la notion de fonction algorithmique aux élèves de seconde, différentes approches sont envisageable. Dans les manuels, nous avons souvent trouvé des leçons qui s'appuient sur les connaissances des fonctions mathématiques des élèves. Nous faisons le choix d'une approche purement algorithmique.

Nous proposons une séquence sur trois séances pour travailler sur la notion de fonction algorithmique. Ces trois séances sont réalisées la même semaine pour immerger les élèves dans l'algorithmique. Le thème de notre séquence est la création d'une librairie de fonctions graphiques avec le module Turtle de Python, puis l'utilisation de ces fonctions pour tracer une figure complexe: un robot.

Le premier objectif qui a guidé notre conception de la séquence est de faire travailler les élèves sur la structuration: programme comportant un main simple avec différents appels de fonctions, répartition du travail entre les élèves d'un même groupe. Le second objectif est de faire travailler les élèves sur la factorisation de code: plusieurs appels à la même fonction.

Nous listons les tâches-objectifs visés pour les élèves avec notre séquence:

- lire, comprendre et écrire une fonction sans argument ni valeur de retour,
- lire, comprendre et écrire une fonction avec argument et sans valeur de retour,
- lire, comprendre et écrire une fonction avec argument et valeur de retour,
- utiliser les fonctions pour factoriser du code,
- utiliser les fonctions pour structurer un programme.

Nous allons maintenant décrire en détail la séquence que nous avons construite pour répondre à ces objectifs.

1. Participants

Les classes avec lesquelles nous réalisons notre expérimentation sont trois classes de seconde: deux classes du lycée Emmanuel Mounier à Grenoble (23 élèves par classe) pour Mélanie Cornillac et une classe du lycée Philibert Delorme à l'Isle d'Abeau (28 élèves) pour Sébastien Detroyat.

Les élèves du lycée Mounier ont des niveaux très hétérogènes. Depuis le début de l'année nous avons travaillé régulièrement sur la programmation Python, notamment en utilisant le tutoriel France IOI en début d'année scolaire, puis depuis février en proposant des séances d'algorithmique sur la notion de variable, d'instruction conditionnelle puis de boucle. Nous abordons aussi sur chaque chapitre étudié un ou deux exercices d'algorithmique. La maîtrise des notions abordées n'est cependant pas acquise par tous les élèves.

Les élèves du lycée Philibert Delorme ont eux aussi des niveaux très hétérogènes. L'algorithmique a été introduite en début d'année en classe sur les notions d'entrée, de sorties et de variables, pour être poursuivie régulièrement en autonomie sur le tutoriel France IOI, et ponctuellement en classe. Les élèves ont vu les notions de boucles bornées et d'instructions conditionnelles mais leur maîtrise est très variable d'un élève à l'autre.

Nous choisissons de faire travailler les élèves par groupes hétérogènes de 3 ou 4 élèves, composés par le professeur. Les groupes resteront identiques au fil des trois séances.

2. Mise en oeuvre et déroulement

La séquence que nous proposons sur les fonctions vient en fin d'année scolaire quand les élèves ont déjà abordé les variables, les instructions conditionnelles et les boucles. Notre séquence est composée de trois séances, la première en débranché pour introduire la notion de fonction, puis les deux suivantes en salle informatique pour réaliser un mini projet mobilisant les fonctions "Dessine moi un robot".

2.1. Séance 1: Introduction de la notion de fonction

Voir en Annexe 1 le document de travail fourni aux élèves et en Annexe 3 une correction pour le professeur.

Pour introduire la notion de fonction aux élèves, nous choisissons une séance d'informatique débranchée, donc en salle de classe habituelle et non en salle machine et commençons par un petit cours au tableau sur la notion de fonction en plus de la trace écrite du document. Les fonctions sont présentées comme une façon de factoriser le code, au même titre que les boucles que les élèves connaissent déjà.

La fiche d'exercices proposée aux élèves commence par deux petits exercices pour fixer le vocabulaire (argument) et le principe de définition et appel de fonction en analysant des fonctions données.

Ensuite, un exercice propose de définir une fonction qui affiche une ligne de 10 caractères "+", puis de rajouter un paramètre permettant de choisir le nombre de caractères affichés, et enfin un second paramètre permettant de choisir le caractère affiché. La difficulté est croissante, avec d'abord un code à compléter, puis une fonction à modifier (ajouter un argument à la fonction précédente pour qu'elle affiche n fois "x" et non 10 fois "+"), et enfin une seconde modification de la fonction, pour ajouter un deuxième argument permettant de choisir le caractère affiché. La dernière question demande la suite d'appels à la fonction avec les bons arguments pour afficher trois lignes de caractères.

Le dernier exercice est plus difficile puisque le code est entièrement à écrire par les élèves. Il s'agit d'un algorithme très classique de détermination du minimum de deux nombres, dont les élèves comprennent facilement le principe et doivent le retranscrire en algorithme. La seconde question demande de définir une fonction qui détermine le minimum de 4 nombres en utilisant la fonction précédente. Nous n'attendons pas des élèves la version avec le corps de la fonction entièrement factorisée en une ligne `return min2(min2(a,b),min2(c,d))`, mais une version avec instruction conditionnelle.

Déroulement

Les élèves prennent connaissance de leur groupe, déplacent le mobilier pour créer des îlots et s'installent par groupe. Le professeur leur explique l'objectif de la séquence et de la séance, le travail attendu et le mode d'évaluation, il commence par un petit cours introductif au tableau sur les fonctions. Il distribue ensuite un document par élève, et une fiche d'évaluation par groupe. Les élèves commencent par lire le cours avant de commencer les exercices. Le professeur circule de groupe en groupe pour répondre aux questions et cocher l'avancée du travail sur la fiche d'évaluation, il répond à toute la classe au tableau pour certaines questions qui reviendraient plusieurs fois. En fin de séance, les élèves sont invités à remettre les tables en place et de vérifier que le professeur a pu valider tout leur travail sur la fiche évaluation qu'ils rendent. Ils ont pour consigne de finir la fiche d'exercices à la maison avant la prochaine séance.

2.2. Séance 2 et 3: Mini projet sur les fonctions "Dessine moi un robot"

Voir en Annexe 2 le document de travail fourni aux élèves, et Annexe 4 une correction pour le professeur.

L'objectif est de créer une petite librairie de fonctions graphiques qui vont utiliser la librairie Python turtle pour tracer des figures géométriques, et utiliser cette librairie pour tracer une figure complexe: un robot. Ces deux séances ont lieu en salle informatique et en voici le déroulement prévu. Le document commence par un mémo sur les 6 fonctions turtle que les élèves manipulent dans ce projet:

`forward(X)` Fait avancer la tortue de X pixels	`left(X)` Fait tourner la tortue de X° vers la gauche	`up()` Lève le crayon pour ne plus afficher la trace de la tortue
`goto(x,y)` Déplace la tortue jusqu'au point de coordonnées (x,y)	`right(X)` Fait tourner la tortue de X° vers la droite	`down()` Baisse le crayon pour afficher la trace de la tortue.

Puis les élèves sont guidés pour créer leurs primitives graphiques avant de créer leur robot.

A. Créer des primitives graphiques

Pour commencer, les élèves doivent recopier et exécuter une suite d'instruction turtle et reconnaître la figure géométrique tracée (un parallélogramme), ceci afin de les familiariser avec les fonctions de la librairie turtle.

Dans le deuxième exercice, les élèves écrivent une fonction `Triangle()` qui trace un triangle équilatéral de côté 20 pixels, puis la modifient pour ajouter en argument la longueur du côté et enfin l'utilisent pour tracer une figure à trois triangles ce qui permet de manipuler la fonction `goto(x,y)` pour se déplacer dans la fenêtre graphique entre deux tracés.

Les élèves écrivent ensuite les fonctions `Carre(c)`, `Rectangle(l,L)`, `Polygone(nb_cotes,longueur)` et `Etoile(l)` qui tracent respectivement un carré de côté c, un rectangle de largeur l et de longueur L, un polygone régulier à n côtés, une étoile à cinq branches de longueur l. Les élèves sont guidés pour le polygone et l'étoile qui sont plus complexes.

B. Dessiner un robot

En utilisant les fonctions précédemment écrites, les élèves sont maintenant invités à dessiner un robot.

Nous les guidons d'abord pour dessiner la figure du robot avec un nez au milieu, en leur introduisant les coordonnées relatives permettant de placer leurs figures les unes par rapport aux autres pour composer un robot. Puis ils sont libres de choisir le robot complet qu'ils veulent tracer.

Une proposition d'extension est faite pour les groupes les plus avancés: ajouter de la couleur à leurs figures (remplissage des figures avec une couleur).

Déroulement

Chaque groupe s'installe avec deux ordinateurs pour pouvoir éventuellement se répartir les tâches. Le professeur explique l'objectif des deux séances et montre un exemple de robot pour illustrer cet objectif (voir figure 3).

Le professeur distribue ensuite à chaque élève un document, et à chaque groupe la petite fiche d'évaluation de la séance 2. La fiche d'évaluation de la séance 3 sera distribuée en début de séance 3. Chaque fiche est ramassée en fin de séance.

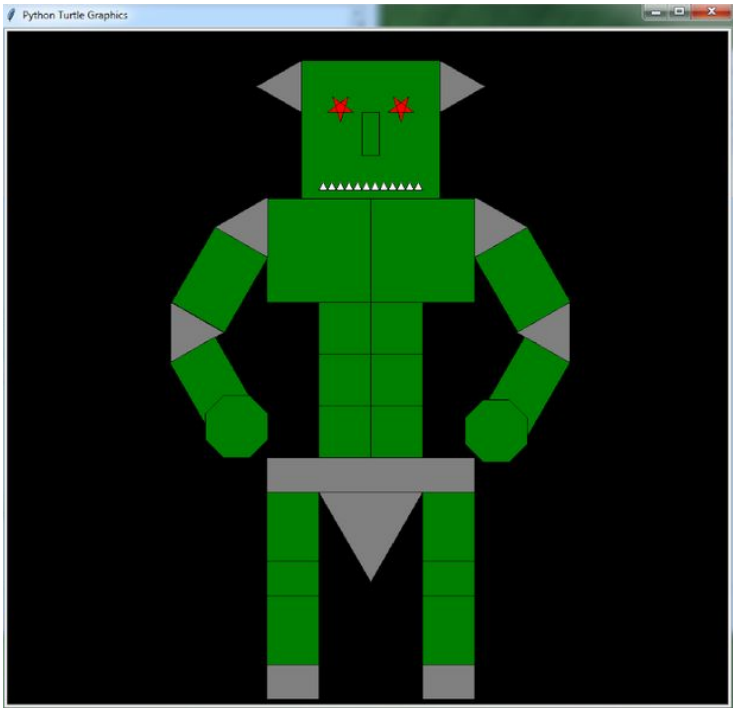


Figure 3: un exemple de robot dessiné avec des primitives géométriques

Le professeur passe ensuite d'un groupe à l'autre pour répondre aux questions pendant la séance et donne éventuellement des consignes à la classe entière si besoin.

Dans le découpage des 2 séances sur le robot, le professeur propose aux élèves de passer directement au robot en deuxième heure, pour que tous les élèves dessinent un robot, ils ont donc plus ou moins de fonctions à leur disposition selon leur avancement dans la première heure.

2.3. Evaluation des élèves

Voir en Annexe 5 les petites fiches distribuées à chaque groupe pour leur évaluation.

Pour la séance 1, nous remettons à chaque groupe une petite fiche sur laquelle nous indiquons leur avancée. Leur implication et le fonctionnement du groupe font aussi partie de l'évaluation, cette note pourra être différenciée pour chaque élève. Avec cette façon d'évaluer, l'avantage est aussi de ne pas demander de rendu en fin de séance, les élèves gardent ainsi chacun leur feuille d'exercices et nous leur demandons de finir les exercices à la maison avant la séance 2.

Pour la séance 2, nous procédons de la même façon, et notons sur une petite feuille pour chaque groupe les primitives géométriques réalisées par les élèves.

Pour la séance 3 (utilisation des primitives pour dessiner une figure complexe: un robot), nous procédons différemment et leur demandons de nous rendre le code en fin de séance, nous leur distribuons une petite fiche donnant les critères d'évaluation de leur code et le professeur note après la séance en regardant le code déposé par les élèves et le robot produit par son exécution.

Nous choisissons trois critères d'évaluation:

- Le code est factorisé: les élèves ont utilisé des boucles et fonctions plutôt que de répéter des parties de code identiques.
- Le code est structuré: les élèves ont découpé leur travail en fonctions pour plus de lisibilité
- Complexité du robot dessiné: nombre et variété des figures géométriques le composant, coloration, détails...

2.4 Critères et indicateurs

Pour évaluer notre séquence, nous mettons en place des critères et indicateurs (voir tableau 1). Les trois critères évalués correspondent aux trois objectifs énoncés dans notre problématique. Pour chaque critère, nous proposons quatre indicateurs que nous noterons Niveau 1, 2, 3, 4 par la suite dans nos tableaux de résultats, le niveau 4 correspondant au niveau le plus élevé de maîtrise des notions par les élèves.

	Niveau 1	Niveau 2	Niveau 3	Niveau 4
Les élèves ont compris comment utiliser une fonction	Les élèves ne savent pas appeler une fonction déjà définie	Les élèves savent appeler une fonction déjà définie	Les élèves savent écrire une fonction sans paramètre et l'appeler	Les élèves savent écrire une fonction paramétrée et l'appeler
Les élèves ont compris que les fonctions permettent de factoriser leur code	Les élèves écrivent à chaque fois les instructions nécessaires sans jamais utiliser de fonctions	Les élèves utilisent des fonctions déjà écrites pour ne pas réécrire du code	Les élèves créent des fonctions sans arguments pour ne pas réécrire plusieurs fois le même code	Les élèves créent des fonctions paramétrées pour factoriser leur code au maximum.
Les élèves ont compris comment les fonctions peuvent aider à structurer leur code	Tout le code est à la suite, sans utilisation de fonctions	Les élèves utilisent quelques fonctions mais les noms sont mal choisis et tout le groupe travaille sur la même chose	Les élèves utilisent des fonctions avec des noms bien choisis et le travail est réparti entre les membres du groupe	Les élèves utilisent des fonctions avec des noms bien choisis et le travail réalisé par un élève est utilisé par un autre membre du groupe.

Tableau 1: Critères et indicateurs pour évaluer notre séquence

Partie III. Résultats

Nous avons expérimenté notre séquence en classe avec nos élèves de seconde la semaine du 8 avril 2019 (deux classes pour Mélanie, une classe pour Sébastien, dénommée Classes 1, 2, 3 voir le tableau 2).

Classe 1	23 élèves	Lycée Emmanuel Mounier	6 groupes
Classe 2	23 élèves	Lycée Emmanuel Mounier	5 groupes
Classe 3	28 élèves	Lycée Philibert Delorme	7 groupes

Tableau 2: Les 3 classes de notre expérimentation

La première séance (en débranché) a été bien vécue par les élèves, ils sont volontiers entré dans les tâches proposées et tous les groupes ont fini les 3 premiers exercices, les groupes les plus avancés (78% des groupes) ont aussi écrit la fonction qui calcule le minimum de 2 nombres, et aucun n'a écrit la fonction qui calcule le minimum de 4 nombres (voir le tableau 3). Dans leurs traces écrites, nous voyons que l'écriture d'un algorithme en respectant l'indentation n'est pas acquis par tous. L'évaluation en faisant valider chaque exercice par le professeur a été bien accueillie, voir les cases être cochées petit à petit les a motivés à s'investir pour avancer rapidement et arriver au bout des exercices avant la fin de la séance.

	Classe 1 (6 groupes)	Classe 2 (6 groupes)	Classe 3 (7 groupes)	Total (% des 19 groupes)
Exercice 1	6	6	7	100%
Exercice 2	6	6	7	100%
Exercice 3	6	6	7	100%
Exercice 4 1)	6	3	6	78%
Exercice 4 2)	0 (2 essais)	0 (0 essai)	0 (2 essais)	0% (22% essais)

Tableau 3: Exercices validés par les groupes d'élèves lors de la première séance

Difficultés fréquemment rencontrées par les élèves:

- Non respect de l'indentation, bien qu'ils aient un modèle,
- Confusion dans les consignes entre la signature de la fonction et l'exemple d'appel,
- Compréhension de l'instruction conditionnelle,
- Ne pensent pas à réutiliser la fonction qu'ils viennent d'écrire quand on leur demande les instructions pour afficher 3 lignes de caractères différents, ils réécrivent trois fois le corps du code de leur fonction `ligneCaracteres` avec des valeurs différentes,
- Incompréhension de la différence entre l'utilisation d'une variable dans le corps de la fonction pour les arguments, et le remplacement par une valeur lors de son appel.

Le bilan est plus nuancé pour la partie sur ordinateur (séances 2 et 3). Dans nos trois classes, obtenir l'écriture de la première fonction qui trace un triangle et son exécution a pris la quasi totalité de la séance 2, alors que nous pensions qu'ils pourraient réaliser plusieurs fonctions.
Nous avons donc adapté nos consignes et en séance 3 les élèves ont continué à rédiger les fonctions de primitives graphiques, car nous avons constaté que la première séance n'avait pas suffit pour qu'ils maîtrisent les compétences attendues. Nous avons dû beaucoup les guider en séance 2 pour qu'ils arrivent à écrire leur première fonction et l'appeler. Peu de groupes ont donc eu le temps de dessiner un robot en classe (4 groupes sur 19). Voir le tableau 4 pour des données chiffrées sur leur avancement dans l'écriture des primitives graphiques à la fin de la séance 3. La figure 4 donne l'évaluation par critères et indicateurs du travail sur le robot réalisé par 7 groupes (les 7 groupes de la classe 3 ont en effet terminé cette tâche à la maison). Nous avons choisi de ne pas évaluer par critères/indicateurs le travail sur le robot des classes 1 et 2 car seuls 3 groupes ont commencé à travailler sur le robot, et ils n'ont eu qu'une quinzaine de minutes sur cette tâche.
En fin de séance 3, près de la moitié de groupes ont terminé les fonctions jusqu'à l'étoile (47%) et surtout ils ont en grande majorité acquis des automatisme et bien compris le principe de définition et appel de fonction. Quand les élèves ont une fonction polygone rédigée, nous les invitons à l'appeler avec un grand nombre de côtés (50 suffit), en leur demandant de conjecturer quelle figure ils vont obtenir (cela tend vers un cercle).

	Classe 1 (6 groupes)	Classe 2 (6 groupes)	Classe 3 (7 groupes)	Total (% des 19 groupes)
`Triangle(c)`	6	6	7	100%
`Triforce()`	6	5	7	95%
`Carre()`	6	3	5	74%
`Rectangle(l,L)`	6	3	5	74%
`Polygone(nb_cotes,longueur)`	5	1	3	47%
`Etoile(l)`	5	0	4	47%

Tableau 4: Primitives graphiques écrites par les groupes d'élèves lors des séances 2 et 3

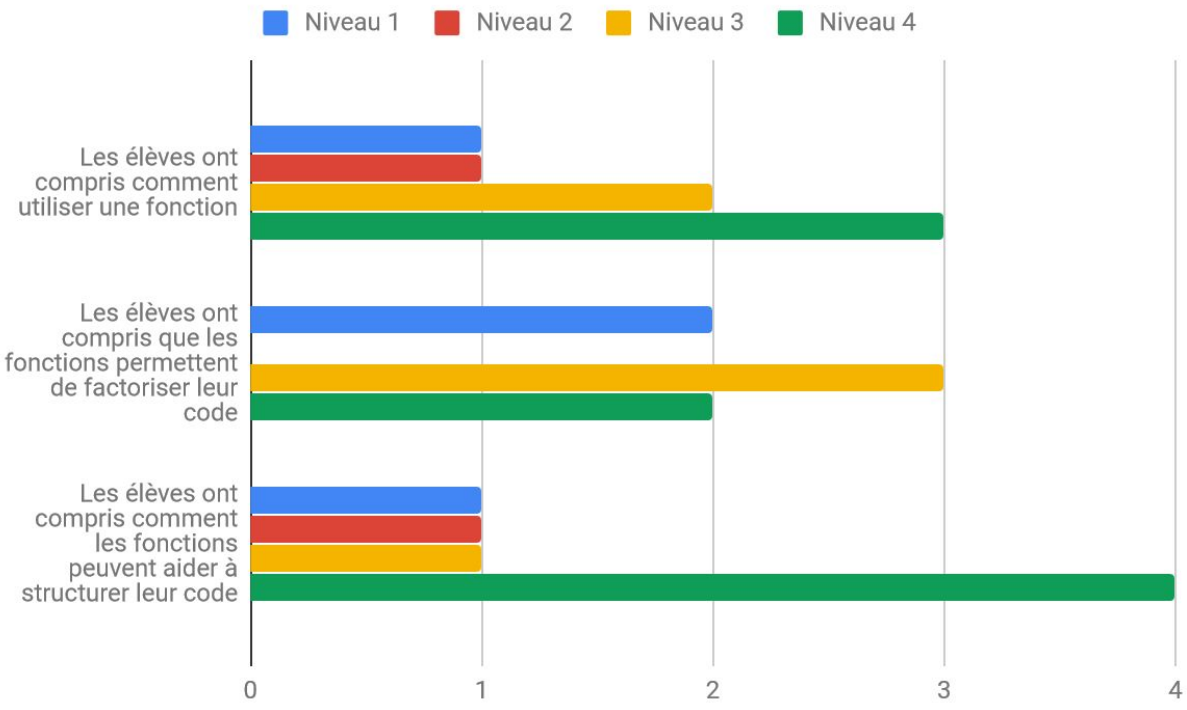


Figure 4: Classement par critères et indicateurs effectué sur les 7 groupes

Difficultés fréquemment rencontrées par les élèves et certaines remédiations que nous leur proposons:

- Confusion entre la définition et l'appel de la fonction,
- Utilisation des arguments: certains appellent la fonction avec l'argument au lieu d'une valeur (nous leur avons alors parlé de la portée des variables), ou définissent la fonction avec une valeur au lieu d'un argument, ou encore mettent bien un argument dans la signature de la fonction mais ne l'utilisent pas dans le corps de la fonction (nous leur demandons alors d'appeler leur fonction pour tracer deux figures de tailles différentes pour qu'ils réalisent quel est le problème en voyant les deux figures identiques).
- Difficulté à interpréter les messages d'erreur de l'interpréteur car c'est en anglais. Un élève particulièrement éloigné des apprentissages en cours de mathématiques et très à l'aise en anglais lit les messages d'erreur et comprends immédiatement la modification de code à effectuer.
- Indentation
- Syntaxe: oubli du caractère ":" après une définition de fonction, un if, un else ou un for.
- Gestion du fichier: certains élèves ont effacé leur travail et ont dû réécrire certaines fonctions s'ils n'arrivaient pas à les récupérer à l'aide de la combinaison de touches "Ctrl+Z" comme nous leur avons conseillé.
- Fonction `goto(x,y)` utilisée avec un point-virgule au lieu d'une virgule pour séparer les arguments car les élèves ont fait l'analogie avec la notation mathématique des coordonnées d'un point.

L'écriture de programmes Python demande une grande rigueur (notamment sur la syntaxe) qui est difficile à aborder pour nos élèves, ils ont pourtant tous déjà fait du Python cette année, mais cette rigueur n'est toujours pas un automatisme pour eux. Autre difficulté majeure pour eux: l'anglais. Les mots clés Python, et aussi les messages d'erreur de l'interpréteur sont en anglais, et certains élèves sont gênés par cela. Au collège, ils ont fait du Scratch en français, le passage à Python ajoute ces deux difficultés: la rigueur dans la syntaxe et l'anglais, bien qu'ils aient acquis

pour certains la logique algorithmique, ces deux nouveaux aspect sont vécus comme des obstacles par beaucoup d'élèves.

Un exemple de robot obtenu par un des groupes les plus avancés (voir la figure 5), voir le code associé en Annexe 6. Les élèves ont découpé leur code en fonctions (`tete()`, `bras()`, `corps()`, `robot()`) et bien manipulé les coordonnées relatives. Néanmoins, une nouvelle fonction a été créée pour chaque couleur, plutôt que de passer la couleur voulue en paramètre.

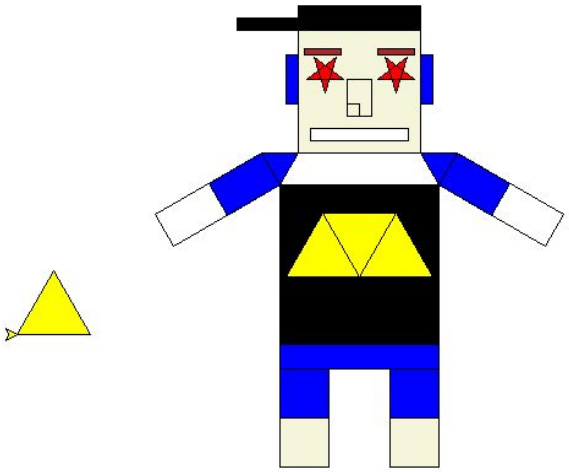


Figure 5: Robot obtenu par un groupe d'élèves avancé

Retour en classe après la séquence

Sébastien demande à ses élèves de finaliser leur robot à la maison. Mélanie choisit de ne pas le faire car elle a des problématiques d'accès à un ordinateur à la maison pour certains de ses élèves.

A la séance qui suit la 3ème séance de la séquence, un petit retour est prévu en classe. Dans notre expérimentation les vacances de printemps séparent cette séance des trois séances de la séquence. Mélanie propose une correction des fonctions `min2` et `min4`(qui utilise `min2`) calculant le minimum de respectivement 2 et 4 nombres. Sébastien propose une synthèse de ce qui a été fait et un retour sur les points les plus importants sur la notion de fonctions: distinction définition/appel d'une fonction, utilisation des arguments.

Partie IV. Discussion et conclusion

Nous avons conçu une séquence sur trois séances pour introduire la notion de fonction algorithmique à nos élèves de seconde. L'objectif était de leur présenter comme un outil de factorisation du code, plutôt que de nous appuyer sur la notion de fonction mathématique comme c'est beaucoup fait dans les manuels. Pour cela, nous leur avons proposé une réalisation avec un objectif motivant et visuel: dessiner un robot grâce à la librairie Turtle de Python. C'était aussi l'occasion de leur faire faire trois heures de Python dans la même semaine et donc de leur laisser du temps pour s'investir dans cette partie du programme que nous abordons habituellement par petites touches (un exercice ou une heure à la fois seulement).

Notre hypothèse de travail dans ce mémoire était que les élèves allaient bien intégrer la notion de fonction algorithmique avec cette approche purement algorithmique et nous avons pu vérifier notre hypothèse lors de l'expérimentation de notre séquence. A la fin des trois séances, la majorité (voir tableau 4 et figure 7) de nos élèves avaient bien compris le principe de définition et appel de fonction avec ou sans arguments, et aussi l'appel d'une fonction dans une autre fonction, ce qui valide notre première hypothèse.

Nous remarquons toutefois que pour aider les élèves à comprendre la distinction entre la définition et l'appel d'une fonction, nous avons naturellement fait le parallèle avec la fonction mathématique sur laquelle ils ont déjà des automatismes, ou avec une recette de cuisine.

De nouvelles fonctions ont été introduites par les élèves dans 5 groupes sur 7, permettant d'éviter les répétitions dans le code (voir figure 7). Nous considérons à ce titre que l'utilisation de fonctions pour factoriser leur programme a été acquis, ce qui valide notre seconde hypothèse.

Le travail a été réparti entre les différents membres de chaque groupe par eux-mêmes dans 5 groupes sur 7, et lors de la construction du robot final, les fonctions écrites par certains membres ont été utilisées par d'autre. Cela valide notre troisième hypothèse concernant l'utilisation de fonctions pour structurer un programme (voir figure 7).

Dans la perspective de reconduire ce mini projet dans le futur, nous modifierions certains points, en particulier sur la séance 2, en guidant plus les élèves au début, par exemple en proposant la première primitive graphique `Triangle()` déjà rédigée avec seulement la longueur du côté et le nombre d'itérations de la boucle à compléter. Les élèves auraient ensuite ce modèle sur lequel s'appuyer pour créer les primitives suivantes. Nous n'avions pas mesuré à quel point cela allait être difficile pour eux de passer d'une suite d'instructions à la fonction. Les élèves étaient en effet très rapides à écrire la suite d'instructions correcte pour tracer la figure avec trois triangles, et ont mis beaucoup de temps à passer à l'utilisation de fonctions pour faire la même figure, ce nouveau concept n'était pas évident pour eux.
Ensuite, le moment choisi pour mettre en oeuvre notre expérimentation n'était pas très propice. Avec nos multiples contraintes de l'année de stage, et les délais pour rendre notre mémoire, nous avons réalisé ce projet la semaine avant les vacances de printemps, en majorité le vendredi après-midi de cette semaine là (après un devoir surveillé le vendredi matin pour les classes de Mélanie), les élèves n'étaient donc pas au maximum de leurs capacités de concentration et d'apprentissage, un autre moment de l'année aurait été plus propice.

En conclusion, nous avons eu beaucoup de plaisir à préparer cette séquence d'algorithmique pour nos élèves et voir en fin de séquence leur avancée dans la maîtrise des concepts présentés. L'introduction de l'algorithmique et programmation est récente dans les programmes du secondaire, et la littérature en didactique de l'informatique n'est pas encore très nombreuse. Les manuels rédigés par des professeurs de mathématiques ont tendance à principalement présenter l'informatique comme un outil au service des mathématiques. Nous pensons qu'il serait intéressant de passer du temps à présenter les notions d'algorithmique comme des notions indépendantes des mathématiques, ce qui n'empêcherait pas de les utiliser ensuite comme un outil pour résoudre des problèmes mathématiques.

Bibliographie

Amra Nadia, La transposition didactique du concept de fonction : comparaison entre les systèmes d'enseignement français et palestiniens. Thèse de doctorat, sciences humaines et sociales Paris 7, 2003. tel.archives-ouvertes.fr/tel-01256635/

Buteau Chantal, Muller Éric, Sacristàn Ana Isabel, Mgombelo Joyce, Pensée informatique en enseignement des mathématiques post-secondaires: un cadre théorique. EMF2018 –SPE5 Mathématiques et Informatique, 2018. .../emf2018_spe5_buteau-revised.pdf

Coppe Sylvie, Dorier Jean-Luc, Yavuz Ilyas, Elements d'analyse sur le programme de 2000 concernant l'enseignement des fonctions en seconde, Petit x 71, 29-60, 2006.

Douady R (1986), Jeux de cadres et dialectique outil/objet, Recherches en Didactique des mathématiques vol 7 n°2, 5-32.

Lac Philippe et More Malika, L'évaluation de l'algorithmique dans l'enseignement des mathématiques au lycée. REPERES - IREM. N°106, janvier 2017

Modeste Simon, Enseigner l'algorithmique, pourquoi ? Quels apports pour l'apprentissage de la preuve ? Quelles nouvelles questions pour les mathématiques ? Thèse de doctorat Mathématiques Grenoble, 2012. tel.archives-ouvertes.fr/tel-00783294/

Page du site Math93: Histoire de la notion de fonction. www.math93.com/histoire-des-maths/…

Page du site Scriptol: Sureau Denis, Histoire et évolution des langages de programmation. www.scriptol.fr/.../histoire-langages.php

Page Wikipedia: Fonction (mathématiques) www.wikipedia.org//Fonction_(mathematiques)

Tisseau Jacques, Initiation à l'algorithmique, procédure et fonctions, 1. Définition d'une fonction (diaporama). www.enib.fr/.../fonctions-1Slide.pdf

Rapaport William J., Philosophy of Computer Science, *Teaching Philosophy* 28 (4):319-341 (2005)

Manuels scolaires de mathématiques

Delta Mathématiques cycle 4, Editions Belin Education, 2016.

Hyperbole Mathématiques 2de, Directeur : J. Malaval, Auteurs : J.-L. Bousseyroux, P.-A. Desrousseaux, F. Destruhaut, H. Lample, I. Lericque, J.-M. Lécole, J. Ternoy, M. Védrine, Editions Nathan, 2017.

Indice Mathématiques 2de, Vleudrin Denis et Poncy Michel, Editions Bordas, 2017.

Indice Algorithmique 2de, Cahier d'activités, Lebert Catherine, Vleudrin Denis et Poncy Michel, Editions Bordas, 2017.

Myriade Cycle 4, Directeur: Marc Boullis, Editions Bordas, 2016.

Sésamaths Cycle 4, Editions Magnard, 2016.

Kiwi mathématiques cycle 4, Beltramone Jean-Paul, Paulou Florian, Candeloro Audrey, Tabourin Dominique, Henry Fabienne, Laboudigue Geoffroy, Galand Nicolas, Stupfler Laurence, Editions Hachette, 2016.

Programmes de mathématiques

Programme de mathématiques de la classe de seconde, BO n°30 du 23 juillet 2009 mathématiques seconde 2009

Aménagement du programme de mathématiques de la classe de seconde, 2017. mathématiques seconde 2017

Programme d'enseignement de mathématiques de la classe de seconde générale et technologique, BO spécial n°1 du 22 janvier 2019
mathématiques seconde 2019

Programme d'enseignement de spécialité de mathématiques de la classe de première de la voie générale, BO spécial n°1 du 22 janvier 2019
mathématiques première 2019

Programme du cycle 4 (programme de mathématiques pages 146-155), BO spécial n° 11 du 26 novembre 2015, et nouvelles dispositions publiées au BO n°30 du 26 juillet 2018.
mathématiques cycle 4 2018

Annexes

Annexe 1 : Mini projet algorithmique sur la notion de fonction (3 séances)

Séance 1 - Introduction sur la notion de fonction

Définition: Une **fonction** est un bloc d'instructions qui a reçu un nom, dont le fonctionnement dépend d'un certain nombre de paramètres (les **arguments** de la fonction) et qui renvoie un résultat (au moyen de l'instruction **`return`**).

Exemple: on peut définir la fonction `somtrois` qui, à partir de la donnée de trois nombres `a, b, c` renvoie comme résultat la somme de ces nombres.

```
def somtrois(a,b,c):
    somme = a+b+c
    return (somme)
```

Ainsi, `somtrois(2,5,-1)` renvoie la valeur 6.

Les noms que l'on peut utiliser comme identifiants de fonctions sont les mêmes que ceux possibles pour les identifiants de variables. Comme pour le choix des noms de variables, il est important de bien choisir les noms des fonctions, pour qu'ils expriment très clairement ce que fait la fonction.

Appeler une fonction revient à remplacer dans le programme, cet appel par le bloc d'instructions correspondant à l'identifiant de la fonction.

Remarques:

- Les entrées d'une fonction se font en donnant des valeurs aux arguments.
- Une fonction peut n'avoir aucun argument.
- Une fonction peut ne pas avoir de valeur retournée, on dit alors que c'est une procédure.
- On peut aussi utiliser l'instruction `print` dans une fonction. Le programme continue après `print`, alors qu'il s'arrête après avoir rencontré `return`.
- Une fonction peut être appelée dans un programme ou dans une autre fonction.

Exercice 1

Soit la fonction `bob` programmée ci-contre en Python.

```
def bob(a,b)
   return (a+b+a*b)
```

1) Quels sont les arguments de cette fonction? ……
2) Que renvoie `bob(2,3)` ?……
3) Que renvoie `bob(-1,1)` ?……

Exercice 2

Soit la fonction `f` programmée ci-contre en langage Python.

```
def f(x,y,z):
   if x>0:
      t=x+y+z
   else:
      t=x*y*z
   return(t)
```

1) Que renvoie `f(3,1,-2)` ? …….
2) Que renvoie `f(-3,-1,2)` ? …….

Exercice 3

1) Compléter cette fonction (renseigner les zones en pointillés) pour qu'elle affiche une ligne de 10 caractères "+" .

L'appel à `lignePlus()` affichera:

`++++++++++`

```
def lignePlus():
   for i in range(...):
       print("...", end = "")
   print()
```

2) Ajouter un argument *n* à cette fonction pour qu'on puisse choisir le nombre de caractères "+" affichés.

Par exemple, l'appel à `lignePlus(4)` affichera:

`++++`

```
def lignePlus(n):
```

……

……

……

3) Ecrire une nouvelle fonction nommée `ligneCaracteres` qui prend en argument le caractère `car` à afficher sur la ligne et le nombre *n* de caractères sur la ligne, et affiche

une ligne de *n* caractères choisis.

Par exemple l'appel à `ligneCaracteres("X", 5)` affichera:

```
XXXXX
```

```
def ligneCaracteres(...,...):
```

……

……

……

4) Donner la suite d'instructions permettant d'obtenir l'affichage suivant:

```
$$$$$$$$
#####
((((
```

……

……

……

Exercice 4

1) Écrire une fonction nommée `min2`, qui prend deux entiers `a` et `b` en arguments et retourne le plus petit.

……

……

……

……

……

2) Proposer le programme d'une fonction qui retourne le plus petit de quatre nombres donnés, en faisant appel à la fonction `min2`.

……

……

……

Annexe 2: Mini projet algorithmique sur la notion de fonction (3 séances)

Séances 2 et 3 - Dessine moi un robot!

Le langage Python propose un module nommé “turtle” permettant de diriger à l’écran une tortue, et d’afficher le tracé de ses mouvements.
Pour l’utiliser, il faut écrire au début du fichier de notre code:

```
from turtle import *
```

Cela nous donne accès à 6 fonctions pour contrôler la tortue:

`forward(X)` Fait avancer la tortue de X pixels	`left(X)` Fait tourner la tortue de X° vers la gauche	`up()` Lève le crayon pour ne plus afficher la trace de la tortue
`goto(x,y)` Déplace la tortue jusqu’au point de coordonnées (x,y)	`right(X)` Fait tourner la tortue de X° vers la droite	`down()` Baisse le crayon pour afficher la trace de la tortue.

Préambule

- Ouvrir l'interpréteur Python, créer un nouveau fichier nommé `robot_prenom.py` (avec un prénom d’un des membres du groupe).
- Taper dans la première ligne: `from turtle import *`
- Taper dans le seconde ligne `up()` pour relever le crayon
- Ecrire l’instruction: `print(“hello world”)`
- Enregistrer puis exécuter le fichier en vérifiant que la console affiche bien le message `“hello world”`

Pour chaque question, vous écrirez le programme d’abord sur votre feuille avant de le tester avec l’interpréteur Python.

Exercice 1: Exécution d’un premier programme

1) Recopier et exécuter le code suivant :

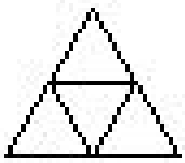
```
down()
forward(40)
left(45)
forward(20)
```

```
left(135)
forward(40)
left(45)
forward(20)
up()
```

2) Qu'affiche-t-il?
   ……

Exercice 2: Ecriture d'un premier programme

1) En s'aidant du programme de l'Exercice 1, écrire une fonction `Triangle()` qui trace un triangle équilatéral de côté 20.
2) Modifier cette fonction pour qu'elle prenne un argument `c` indiquant la longueur du côté du triangle.
3) Ecrire une fonction `Triforce()` appelant notre fonction `Triangle()` 3 fois pour tracer la figure ci dessous
   Rappel: pour se déplacer dans la fenêtre, on utilise la fonction: `goto(x,y)`



Exercice 3: Ajout de nouvelles fonctions

1) Ecrire une fonction `Carre(c)` prenant en argument la longueur `c` d'un côté et traçant un carré de côté c. Tester cette fonction.
2) Ecrire une fonction `Rectangle(l,L)` qui trace un rectangle de largeur `l` et de longueur `L`. Tester cette fonction.
3) Dessiner un polygone régulier à n côtés:
   a) Combien d'angles a un polygone à n côtés? ……....
   b) De combien de degrés la tortue doit-elle tourner pour faire un tour complet sur elle-même? ……

c) En déduire de combien de degrés la tortue doit tourner à chaque angle pour tracer un polygone à n côtés. …….

d) Compléter la fonction suivante qui trace un polygone régulier à n côtés.

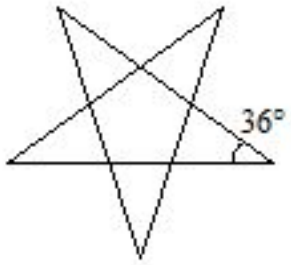
```
def polygone(nb_cote, longueur):
    down()
    for i in range(...):
        forward(...)
        left(...)
    up()
```

e) Essayer cette fonction pour tracer un octogone (8 côtés), un icosagone (20 cotés)...etc.

f) Que remarquez-vous en appelant `polygone(3,20)` ?

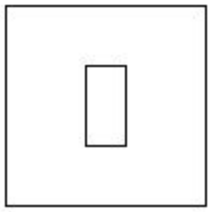
4) Ecrire une fonction `Etoile(l)` qui trace une étoile à 5 branches de longueur `l`

Indication:



Exercice 5: Dessine moi un robot!

<u>Partie 1: Un nez au milieu de la figure</u>



1) Ecrire une fonction `Tete()` dessinant une tête avec un nez au milieu, grâce aux fonctions `Carre` et `Rectangle`
2) Placer cette tête aux coordonnées (100,100). Que se passe-t-il?

…….

3) Pour remédier à cela, il faut indiquer les coordonnées du nez par rapport à celles de la tête. Ainsi, si on avait placé le nez aux coordonnées (30,40), il faut

désormais le placer aux coordonnées (100+30, 100+40). Pour cela, `turtle` nous donne les fonctions `xcor()` et `ycor()`, renvoyant respectivement l'abscisse et l'ordonnée de la tortue. Si on récupère ces coordonnées au début de la fonction `Tete()`, il suffira de les ajouter à celles du nez.

```
def Tete():
    x=xcor()
    y=ycor()
    Carre(100)
    goto(x+40,x+30)
    Rectangle(20,40)
```

4) En s'appuyant sur l'exemple avec le nez, modifier la fonction `Tete()` pour dessiner des yeux (on utilisera une forme géométrique au choix).

Partie 2: Un robot complet

A l'aide des fonctions que vous avez implémentées, écrire un programme qui dessine un robot. Attention, ce travail est long, pensez à vous répartir les tâches!
Faites d'abord une figure sur papier avec les coordonnées des différents éléments, puis vous pourrez travailler sur deux ordinateurs pour créer les différentes parties, que vous décomposerez en fonctions (`JambeDroite()`, `JambeGauche()`, …).

Pour aller plus loin: Dessine moi un robot coloré!

Turtle nous permet aussi de colorier (remplir avec une couleur) nos figures.
Pour cela, nous avons besoin de trois nouvelles fonctions du module turtle:

- `fillcolor(couleur)` prend une couleur en argument et définit la couleur de remplissage. Attention, il faut écrire le nom de la couleur en anglais, entre guillemets: `"red", "blue", "green", "yellow", "black", "white", "pink", "grey"`,...
- `begin_fill()` précise le début du remplissage. Cela fonctionne comme la fonction `down()`

- `end_fill()` précise la fin du remplissage. Cela fonctionne comme la fonction `up()`

Exemple: Voici une fonction qui trace un triangle rempli en rouge:

```
def Triangle_rouge(c):
    fillcolor("red")
    begin_fill()
    triangle(c)
    end_fill()
```

1) En s'aidant de cet exemple, créer une fonction `Triangle_colore(c,couleur)` qui trace un carré de côté c de la couleur donnée en argument.
2) Ajouter les fonction `Carre_colore(c,couleur)`, `Rectangle_colore(l,L,couleur)`, `Etoile_coloree(l,couleur)`.
3) Utiliser ces fonctions pour dessiner un robot coloré.

Annexe 3: Correction de la Séance 1 sur les fonctions algorithmiques

Exercice 1

Soit la fonction `bob` programmée ci-contre en Python.

```
def bob(a,b)
   return (a+b+a*b)
```

1) Quels sont les arguments de cette fonction? **a et b**
2) Que renvoie `bob(2,3)` ? **11**
3) Que renvoie `bob(-1,1)` ? **-1**

Exercice 2

Soit la fonction `f` programmée ci-contre en langage Python.

```
def f(x,y,z):
   if x>0:
      t=x+y+z
   else:
      t=x*y*z
   return(t)
```

1) Que renvoie `f(3,1,-2)` ? **2**
2) Que renvoie `f(-3,-1,2)` ? **6**

Exercice 3

1) Compléter cette fonction (renseigner les zones en pointillés) pour qu'elle affiche une ligne de 10 caractères "+" .

L'appel à `lignePlus()` affichera:

```
++++++++++
```

```
def lignePlus():
    for i in range(10):
        print("+", end = "")
    print()
```

2) Ajouter un argument *n* à cette fonction pour qu'on puisse choisir le nombre de caractères "+" affichés.

Par exemple, l'appel à `lignePlus(4)` affichera:

```
++++
```

```
def lignePlus(n):
    for i in range(n):
        print("+", end = "")
    print()
```

3) Ecrire une nouvelle fonction nommée `ligneCaracteres` qui prend en argument le caractère `car` à afficher sur la ligne et le nombre *n* de caractères sur la ligne, et affiche

une ligne de *n* caractères choisis.

Par exemple l'appel à `ligneCaracteres("X", 5)` affichera:

```
XXXXX
```

```
def ligneCaracteres(car,n):
    for i in range(n):
        print(car, end = "")
    print()
```

4) Donner la suite d'instructions permettant d'obtenir l'affichage suivant:

```
$$$$$$$$

#####

((((
```

```
ligneCaracteres("$",8)
ligneCaracteres("#",5)
ligneCaracteres("(",4)
```

Exercice 4

1) Écrire une fonction nommée `min2`, qui prend deux entiers `a` et `b` en arguments et retourne le plus petit.

```
def min2(a,b):
    if a<b:
        return a
    else:
        return b
```

2) Proposer le programme d'une fonction qui retourne le plus petit de quatre nombres donnés, en faisant appel à la fonction `min2`.

Version attendue:

```
def min4(a,b,c,d):
    e = min2(a,b)
    f = min2(c,d)
    if e<f:
        return e
    else:
        return f
```

Version factorisée:

```
def min4(a,b,c,d):
    return min2(min2(a,b), min2(c,d))
```

Annexe 4: Proposition de correction pour les séances 2 et 3 “Dessine moi un robot!”

```
from turtle import *
from math import *


def triangle(c):
    down()
    for i in range(3):
        forward(c)
        left(120)
    up()

def filled_triangle(c, color):
    fillcolor(color)
    begin_fill()
    triangle(c)
    end_fill()

def rectangle(l,L):
    down()
    for i in range(2):
        forward(l)
        left(90)
        forward(L)
        left(90)
    up()

def filled_rectangle(l,L,color):
    fillcolor(color)
    begin_fill()
    rectangle(l,L)
    end_fill()

def carre(c):
    down()
    for i in range(4):
        forward(c)
        left(90)
    up()

def filled_carre(c,color):
    fillcolor(color)
    begin_fill()
    carre(c)
    end_fill()

def carre_angle(c,a):
    left(a)
    carre(c)

def polygone(nb_cote, longueur):
    down()
    for i in range(nb_cote):
        forward(longueur)
        left(360/nb_cote)
    up()
```

```
def filled_polygone(nb_cote, longueur,color):
    fillcolor(color)
    begin_fill()
    polygone(nb_cote, longueur)
    end_fill()

def etoile(l):
    down()
    for i in range(5):
        forward(l)
        left(144)
    up()

def filled_etoile(l,color):
    fillcolor(color)
    begin_fill()
    etoile(l)
    end_fill()

def reset_angle():
    right(heading())


def Triforce():
   Triangle()
   y=sqrt(20**2-10**2)
   goto(-10,-y)
   Triangle()
   goto(10,-y)
   Triangle()
```

```
def tete():
    x=xcor()
    y=ycor()
    filled_carre(160,"green")
    goto(x+30,y+100)
    filled_etoile(30,"red")
    goto(x+100,y+100)
    filled_etoile(30,"red")
    goto(x+70,y+50)
    filled_rectangle(20,50,"yellow")
    goto(x+20,y+10)
    for i in range(12):
        filled_triangle(10,"white")
        forward(10)
    goto(x,y+160-60)
    left(90)
    filled_triangle(60,"grey")
    goto(x+160,y+160)
    right(180)
    filled_triangle(60,"grey")

def corps():
    x=xcor()
    y=ycor()
    filled_carre(120,"green")
    goto(x+120,y)
    filled_carre(120,"green")
    goto(x+60,y-60)
    filled_carre(60,"green")
    goto(x+120,y-60)
    filled_carre(60,"green")
    goto(x+60,y-120)
    filled_carre(60,"green")
    goto(x+120,y-120)
    filled_carre(60,"green")
    goto(x+60,y-180)
    filled_carre(60,"green")
    goto(x+120,y-180)
    filled_carre(60,"green")
    goto(x,y-220)
    filled_rectangle(240,40,"grey")
    goto(x+180,y-220)
    left(180)
    filled_triangle(120,"grey")
    reset_angle()

def jambe():
    x=xcor()
    y=ycor()
    filled_rectangle(60,200,"green")
    goto(x,y+80)
    filled_rectangle(60,40,"green")
    goto(x,y-40)
    filled_rectangle(60,40,"green")
```

Annexe 5 - Les trois fiches d'évaluation des groupes pour les 3 séances.

Prénoms :

Demandez au professeur de valider chaque exercice pendant la séance.

Implication, travail en groupe	/3
Exercice 1	/1
Exercice 2	/1
Exercice 3 1)	/1
2)	/1
3)	/1
4)	/1
Exercice 4 1)	/1
2)	/1
Note séance 1	/11

Prénoms :

Demandez à la professeure de valider vos fonctions pendant la séance 2.

Implication, travail en groupe	/2,5
Triangle(c)	/2
Triforce()	/2
Carre(c)	/2
Rectangle(l,L)	/2
Polygone(nb_cotes,longueur)	/2
Etoile(l)	/2
Note séance 2	/14,5

Prénoms :

Rendez votre code à la fin de la séance 3 (envoyez un message Pronote à la professeure avec le fichier robot_prenom.py en pièce jointe).

Implication, travail en groupe	/2,5
Le code est factorisé (le code identique n'est pas répété, utilisation d'une boucle ou d'une fonction quand c'est nécessaire)	/4
Le code est structuré	/4
Complexité du robot dessiné (nombre et variété des figures géométriques le composant, coloration, détails...)	/4
Rendre votre fichier Python en fin de séance 3	
Note séance 3	/14,5

Note globale du projet sur les fonctions algorithmiques (3 séances)	/40

Annexe 6 - Code Python produit par un groupe d'élèves

```
from turtle import*
up()
def triangle(c):
    down()
    for i in range(3):
        forward(c)
        left(120)
    up()

def triangle_bleu(c):
    fillcolor("blue")
    begin_fill()
    triangle(c)
    end_fill()

def etoile_rouge(l):
    fillcolor("red")
    begin_fill()
    Etoile(l)
    end_fill()

def rectangle_noire(l,L):
    fillcolor("black")
    begin_fill()
    rectangle(l,L)
    end_fill()

def rectangle_bleu(l,L):
    fillcolor("blue")
    begin_fill()
    rectangle(l,L)
    end_fill()

def rectangle_blanc(l,L):
    fillcolor("white")
    begin_fill()
    rectangle(l,L)
    end_fill()

def rectangle_marron(l,L):
    fillcolor("brown")
    begin_fill()
    rectangle(l,L)
    end_fill()

def Triforce():
    for i in range(3):
        triangle(60)
        left(60)
    up()
    left(180)
    goto(-30,52)
    down()
    triangle(60)
```

```
def Triforce_jaune():
    fillcolor("yellow")
    begin_fill()
    Triforce()
    end_fill()



def Etoile(l):
    down()
    for i in range(5):
        forward(l)
        left(144)

def carre(c):
    down()
    for i in range(4):
        forward (c)
        left (90)
    up()

def carre_beige(c):
    fillcolor("beige")
    begin_fill()
    carre(c)
    end_fill()

def rectangle(l,L):
    down()
    for i in range (2):
        forward (l)
        left (90)
        forward (L)
        left (90)
    up()

def polygone(nb_cote, longueur):
    down()
    for i in range(19):
        forward(longueur)
        left(nb_cote)
    up()
```

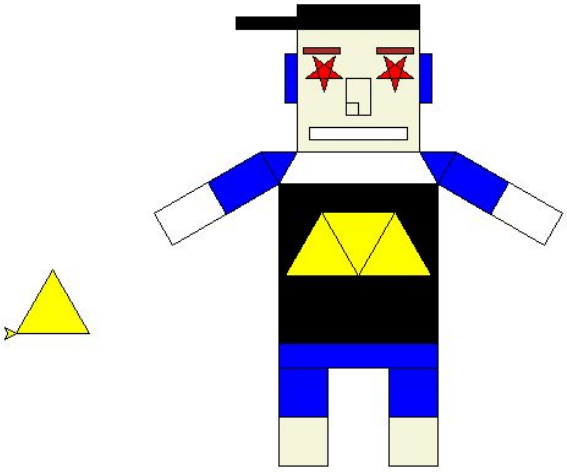
```
def Tete():
    x=xcor()
    y=ycor()
    carre_beige(100)
    goto(x+40,x+30)
    carre(10)
    rectangle (20,30)
    goto(x+7,x+60)
    etoile_rouge(30)
    up()
    goto(x+65,x+60)
    down()
    etoile_rouge(30)
    up()
    goto(x+10,x+10)
    down()
    rectangle_blanc(80,10)
    up()
    goto(x+0,x+100)
    down()
    rectangle_noire(100,20)
    rectangle_noire(-50,10)
    up()
    goto(x+5,x+80)
    down()
    rectangle_marron(30,5)
    up()
    goto(x+65,x+80)
    down()
    rectangle_marron(30,5)
    up()
    goto(x+0,x+40)
    down()
    rectangle_bleu(-10,40)
    up()
    goto(x+100,x+40)
    down()
    rectangle_bleu(10,40)
```

```
def bras():
    x=xcor()
    y=ycor()
    goto(x+0,x+0)
    triangle_bleu(-30)
    up()
    goto(x+130,x+0)
    down()
    triangle_bleu(-30)
    up()
    goto(x-15,x-25.98)
    down()
    left(120)
    rectangle_bleu(30,50)
    rectangle(30,100)
    up()
    goto(x+115,x-25.98)
    down()
    left(-240)
    rectangle_bleu(-30,50)
    rectangle(-30,100)


def corp():
    x=xcor()
    y=ycor()
    goto(x-15,x-25.98)
    left(120)
    rectangle_noire(130,-150)
    up()
    goto(x-15,x-175.98)
    down()
    rectangle_bleu(130,20)
    rectangle_bleu(40,-80)
    up()
    goto(x+115,x-175.98)
    down()
    rectangle_bleu(-40,-80)
    up()
    goto(x-15,x-255.98)
    down()
    carre_beige(40)
    up()
    goto(x+75,x-255.98)
    down()
    carre_beige(40)
    up()
    goto(x+50,x-100.98)
    down()
    Triforce_jaune()
```

```
def robot():
    goto(200,200)
    Tete()
    goto(200,200)
    bras()
    goto(200,200)
    corp()


robot()
```



Année universitaire 2018-2019

Master 2 *Métiers de l'enseignement, de l'éducation et de la formation*

Mention Second degré mathématiques

Titre du mémoire : Enseignement des fonctions algorithmiques en classe de seconde.

Auteurs : Mélanie Cornillac et Sébastien Detroyat

Résumé: Depuis son entrée dans les programmes, l'algorithmique prend une part grandissante dans la classe de mathématiques en seconde. La transmission aux élèves de la pensée informatique devient un enjeu dans l'enseignement des mathématiques. Nous nous intéressons à l'enseignement de la fonction algorithmique en seconde. L'objectif est de leur présenter comme un outil de factorisation et structuration du code, plutôt que de nous appuyer sur la notion de fonction mathématique comme c'est souvent fait dans les manuels. Nous proposons une séquence de trois séances "Dessine moi un robot" où les élèves sont amenés à coder en Python une librairie graphique. La majorité des élèves parvient à utiliser et saisir l'intérêt des fonctions algorithmiques à l'issue de la séquence. Et nous concluons que notre proposition d'approche purement algorithmique dans l'enseignement des fonctions algorithmiques est pertinente.

Mots-clés: enseignement des mathématiques, 2de, algorithmique, langage de programmation Python, fonction.

Abstract: Since algorithms have appeared in the programs, they have taken on a growing part in mathematics in the class of "seconde". The teaching of the computer thought process has become a defy in the teaching of mathematics. We focus on teaching algorithmic functions in seconde. The objective is to introduce them as a tool for factorizing rather than relying on a mathematical function as often done in textbooks. We suggest a sequence of three sessions "Draw me a robot" where students get to code in Python a graphic library. By the end of the sequence the majority of students manage to use the algorithmic functions as well as understand their advantages. We have concluded that our purely algorithmic approach to the teaching of algorithmic functions is pertinent.

Key-words: mathematics teaching, 2de, algorithms, Python programming language, function.