Python : une deuxième opportunité pour les élèves de comprendre la notion de variable

Geneviève Davion

Année universitaire 2017-2018

Master MEEF

Mention 2nd degré- parcours FSTG

2ème année

Python : Une deuxième opportunité pour les élèves de comprendre la notion de variable

Mots Clefs : Variable informatique, Transition Scratch-Python, Algorithmique, Programmation

Présenté par : Geneviève Davion

Encadré par : Katia Odiot

École Supérieure du Professorat et de l'Éducation de l'académie de Paris
10 rue Molitor, 75016 PARIS – tél. 01 40 50 25 92 – fax. 01 42 88 79 74
www.espe-paris.fr

Table des matières

Introduction

L'algorithmique prend une place de plus en plus importante dans les programmes de mathématiques du cycle 4 et du lycée. L'aménagement du programme de seconde de cette année stipule qu'il faut mettre en place : « une consolidation des acquis du cycle 4 autour de deux idées essentielles :

- la notion de fonction d'une part, et
- la programmation comme production d'un texte dans un langage informatique d'autre part. »

L'utilisation d'un langage informatique est une nouveauté de l'aménagement du programme de seconde et le langage qui nous est recommandé est Python. Eduscol a conçu un document ressource pour les professeurs dans lequel sont expliquées les raisons de ce choix (voir [Edu]) : « Python [a été] choisi pour la concision et la simplicité de sa syntaxe, la taille de la communauté d'utilisateurs (en particulier dans le cadre éducatif), ainsi que la richesse des ressources disponibles. » . Les élèves arrivant en seconde ont étudié Scratch, et la question naturelle que l'ont peut se poser est « Comment faciliter la transition entre Scratch et Python ? ».

Ma première réponse à cette interrogation a été de travailler Scratch avec mes élèves en début d'année, cela étant lié au fait que beaucoup d'entre eux n'avait que peu (voire pas) travaillé Scratch au collège et commencer à programmer avec un langage aussi formel que Python est susceptible, selon moi, de rebuter les élèves. Cela m'a permis de m'apercevoir que la première difficulté que rencontraient les élèves était la notion de variable : ceux qui n'ont pas compris cette notion ont été en difficulté tandis que les autres ont globalement bien réussi les exercices proposés. C'est suite à cette constatation que j'ai décidé d'insister sur la notion de variable lorsque j'ai commencé à travailler sur Python avec les élèves, en faisant la supposition que, dès lors que les élèves maîtriseraient la notion de variable et la manière dont Python traite les variables, ces derniers n'auraient que peu de difficultés par la suite.

Ainsi, la question qui sous-tend cet écrit est : « Comment permettre aux élèves de redécouvrir la notion de variable lors de l'apprentissage de Python ? »

Pour répondre à cette question, je commencerai par étudier des textes. Le premier est le premier chapitre de la thèse de Simon Modeste ([Mod12]) portant sur le concept d'algorithme et le second est un article de Jean-Baptiste Lagrange et Janine Rogalski sur les premiers apprentissages en programmation et en algorithmique ([LR17]) résumant des avancées sur la recherche dans ce domaine depuis l'introduction d'une option informatique au lycée dans les années 80. Cela sera suivi d'une partie sur les variables qui commencera par un résumé des définitions et propriétés et finira par une étude de la vision des élèves. Enfin, dans la troisième partie seront détaillées deux expérimentations faites en classe, la première sur l'algorithme d'échange de deux variables et la deuxième sur le rôle de la variable dans le cadre de l'instruction conditionnelle.

I Résumés de textes

I.1 Autour du concept d'algorithme, thèse de Simon Modeste

Simon Modeste a étudié l'enseignement de l'algorithmique dans sa thèse (voir [Mod12]). Il a posé 9 questions auxquelles il a répondu. Ici, nous nous pencherons sur la réponse donnée à la première question : « Quels sont les éléments constitutifs spécifiques du concept d'algorithme ? Quels sont l'objet et les préoccupations fondamentales de la discipline algorithmique ? Quel lien entretiennent algorithmique et preuve ? », réponse qui est formulée dans le premier chapitre.

Ce premier chapitre commence par la recherche d'une définition d'algorithme. La première caractérisation est celle donnée par Knuth : un algorithme a 5 caractéristiques fondamentales qui sont :

- « Finiteness » : Un algorithme doit toujours se terminer en un nombre fini d'étapes.
- « Definiteness » : Chaque étape de l'algorithme doit être précisément définie et ne comporter aucun ambiguïté.
- « Input » : Un algorithme peut avoir des entrées c'est-à-dire des quantités qui sont spécifiées avant qu'il ne commence.
- « Output » : Les sorties de l'algorithme sont des quantités qui dépendent des entrées. Il y en a au moins une.
- « Effectiveness » : Chaque opération de base effectuée par un algorithme doit pouvoir être effectuée (en théorie) par un humain muni d'un crayon.

Les algorithmes sont aussi vus comme « des méthodes de résolution de problèmes, adaptés à une implémentation sur ordinateur » (par Sedgewick). L'accent est mis sur la différence entre algorithme et programme, le programme étant l'incarnation d'un algorithme dans un langage donné.

Un algorithme peut d'ailleurs prendre différentes formes, à l'aide de langages (informatiques ou non) ou de diagrammes par exemple. Dans la plupart des langages de programmations, on retrouve l'instruction conditionnelle et les boules « while » et « for ».

L'autre aspect des algorithmes qui est mis en valeur est celui de preuve : il faut prouver qu'un algorithme fait ce qu'on attend de lui et termine mais les algorithmes peuvent aussi faire l'office de preuves en mathématiques (on parle de « preuves algorithmiques »).

Après cette étude, une définition d'algorithme est proposée : « Un algorithme est une procédure de résolution de problème, s'appliquant à une famille d'instances du problème et produisant, en un nombre fini d'étapes constructives, effectives, non-ambigües et organisées, la réponse au problème pour toute instance de cette famille. ».

La notion d'algorithme a été formalisée au cours du XX$^{\text{ème}}$ siècle et a donné naissance à des modèles théoriques comme celui de la machine de Turing.

L'algorithmique est « la science dont l'objet d'étude est l'algorithme ». Un point important de l'algorithmique est d'être capable de créer un algorithme adapté à la résolution d'un problème. L'étude d'algorithmes fait entrer en jeux de nouvelles notions :

- Une **variable informatique** « modélise un emplacement dans la mémoire et son contenu peut changer ».
- La **complexité** d'un algorithme est caractérisée par « l'étude du comportement de l'algorithme en fonction de ses entrées ». La complexité en temps prend en compte le

nombre d'opérations élémentaires effectuées par l'algorithme en fonction de la taille de l'entrée tandis que la complexité en espace est liée à la quantité de mémoire nécessaire.

- La **représentation des instances** influe sur l'écriture de l'algorithme et son efficacité, surtout lorsqu'on utilise un ordinateur.
- Il est possible de **comparer** des algorithmes ce qui introduit la notion de meilleur algorithme.
- Trouver le meilleur algorithme permet de parler de la **complexité** d'un problème.
- « Algorithmique » est vu par Knuth comme synonyme d'« **informatique** » et même si ce n'est pas la seule conception, il est certain que c'est ce qui lie l'ensemble des sous-branches de l'informatique.

Trois exemples caractéristiques sont alors données : celui de l'algorithme d'Euclide, celui pour trouver un cycle eulérien dans un graphe (ou dire qu'il n'en existe pas) et celui du tri rapide. Ces exemples permettent d'illustrer plusieurs des points détaillés précédemment.

L'étude précédente permet de donner ce que Simon Modeste appelle les « aspects » de l'algorithme. Il s'agit de l'aspect Problème, l'aspect Effectivité, l'aspect Preuve, l'aspect Complexité et l'aspect Modèles Théoriques. Ces cinq aspects fournissent une réponse à la première question de la thèse.

Le lien est fait avec la dialectique outil-objet : lorsque nous voyons l'algorithme comme un outil (c'est-à-dire quand nous nous concentrons sur l'usage qu'on en fait pour résoudre un problème), nous faisons appelle aux aspects Problème et Effectivité mais quand nous étudions l'algorithme pour lui-même, quand l'algorithme est vu comme un objet, les aspects qui s'y réfèrent sont alors les aspects Preuve, Complexité et Modèles Théoriques.

I.2 L'apprentissage de l'algorithmique, article de Lagrange et Rogalski

Dans [LR17], Lagrange et Rogalski étudient l'enseignement de l'algorithmique et de la programmation en France et visent à faire un bilan sur la recherche dans ce domaine. Dans la première partie, ils s'appuient sur des recherches pour étudier les difficultés des élèves liées à la programmation.

Dans les années 80, une option informatique était proposée aux lycéens. Elle a disparu dans les années 90, laissant la place à l'utilisation de l'outil informatique (tableur par exemple). Depuis 2009, l'algorithmique est à nouveau au programme du lycée. Les notions d'algorithmique et de programmation ne sont pas véritablement dissociées. Pour programmer, les élèves ont besoin de la notion de variable. Les variables sont caractérisées par leur **type**, dans l'article, les types considérés sont « nombre », « chaîne » (assemblage de caractères) et « booléens » (qui prend deux valeurs : vrai et faux). Les langages informatiques étudiés sont des langages impératifs qui ressemblent au langage naturel. La thèse développée ici est que cela est initialement une aide mais peut « constituer un obstacle à la compréhension de propriétés générales des langages informatiques ».

Un premier exemple d'exercice vient de la thèse de Briant (2003). Les élèves doivent écrire un programme qui prend trois nombres a, b et c en entrée et renvoie la solution de l'équation $ax + b = c$. La réponse attendue est de renvoyer $\frac{c-b}{a}$ (quand a est non nul). Plusieurs élèves ne s'extraient pas du cadre papier-crayon et donnent à l'ordinateur une suite d'instruction de calcul similaire à ce qu'ils font naturellement (c prend la valeur $c-b$ puis c prend la valeur c/a puis X prend la valeur c). Les difficultés des élèves pourraient

ne pas être purement liées au langage informatique ce qui n'est pas le cas dans le deuxième exercice étudié. Cet exercice traite des chaînes de caractères. La première question est de comprendre ce que fait un programme (il écrit la première et la dernière lettre d'une chaîne à la suite) et la seconde demande d'écrire un programme qui place le premier caractère d'une chaîne à la fin de celle-ci. Dans ces deux exemples, on observe que les élèves ne comprennent pas que le programme fonctionne en effectuant des modifications successives de variables. On trouve des problèmes d'expression dans le premier exemple et, dans le deuxième, on remarque les élèves ne comprennent pas la signification des objets qu'ils utilisent et des opérations effectuées sur ces objets.

Une autre difficulté des élèves est liée au type booléen. Un exemple illustrant cela est tiré du mémoire de Guy (2013). Les élèves doivent utiliser une boucle tant que qui se termine quand est nombre est égal à 0 ou à 495. Nier cette condition pose problème aux élèves, ce qui est lié à l'utilisation du « ou » dans le langage courant. Un exercice de Lagrange (1991) montre que les élèves voient « les conditions booléennes comme des conditions telles qu'elles s'expriment dans le langage usuel, plutôt que comme un calcul sur des valeurs logiques ». Certains élèves utilisent des chaînes (oui et non) ou des des nombres (0 et 1) à la place des booléens.

La difficulté de voir les variables comme des objets habituels mais avec un fonctionnement différent est proche des difficultés qu'ont les élèves en algèbre (vu comme « outil de modélisation du réel »). Le type de la variable modifie le sens que lui donnent les élèves. Les problèmes liés à l'utilisation des booléens sont des problèmes liés au fait de passer de la logique usuelle à la logique mathématique.

La question de trouver de bonnes situations didactiques pour les premiers apprentissages se pose alors. Pour la structure itérative, les auteurs se basent sur les travaux de Nguyen qui fait le choix d'une situation « a-didactique » opposée à un apprentissage par « monstration », dans laquelle l'élève ressent la nécessité d'une structure itérative.

Pour trouver des situations appropriées, il faut avoir conscience des difficultés des élèves pour construire un algorithme : un expert part d'un plan qu'il décompose en étapes, appelées sous-objectifs (approche descendante) ou, s'il a des difficultés, utilise des schémas qu'il connaît déjà (approche ascendante). Le débutant, lui, n'arrive pas à définir les sous-objectifs de façon adaptée à un dispositif informatique et ne connaît pas suffisamment de schémas. Les « savoirs sous-jacents » sont alors définis comme la capacité à concevoir ces deux approches, à déterminer les variables adéquates et à vérifier qu'elles sont bien gérées au cours de la complétion de chaque sous-objectif.

Après avoir traité ces exemples, les auteurs résument les acquis sur l'activité de programmation. Tout d'abord, l'informatique a deux faces : « science et pratique », ces deux aspects sont à prendre en compte. Programmer est une activité qui a ses propres spécificités. Certains concepts-clés en informatique n'ont pas de « précurseurs » tandis que d'autres ont des précurseurs qui à la fois facilitent la compréhension de l'élève et lui rendent la tâche plus difficile, par exemple l'utilisation d'un langage impératif qui ressemble au langage naturel aide d'abord l'élève mais est un obstacle à la compréhension des spécificités des langages informatiques.

Les difficultés d'ordre cognitif dans l'apprentissage de la programmation informatique sont les suivantes :

- lorsque les élèves écrivent un programme, il délègue le contrôle à la machine, cela nécessite de d'avoir une représentation du fonctionnement de la machine ;
- les élèves doivent créer une procédure générique à partir d'une « connaissance en actes » et se questionner sur la validité de ce qu'ils écrivent ;

- il faut connaître les objets et les actions possibles dans le cadre de la programmation ;
- voir le côté fonctionnel de la récursivité est difficile ;
- il faut contrôler la correction logique des structures conditionnelles ;
- exprimer correctement les entrées et sorties est compliqué.

Deux autres difficultés sont enfin soulevées : la notion d'invariant de boucle dans le cadre d'une programmation récursive et la notion de variable qui est détaillée dans la suite de l'article.

La variable informatique peut être considérée comme une fonction car « sa valeur dépend du moment de l'exécution du programme ». Les variables dans un programme dépendent aussi des entrées. En s'appuyant sur les propos de Knuth, on remarque qu'une différence fondamentale entre variables mathématiques et informatiques est l'opération d'affectation et ainsi la « notion d'état d'un processus ». La notion de variable mathématique peut tout de même servir de précurseur à celle de la variable informatique bien que le fait qu'elle ne change pas de valeur est « réducteur ». On peut remédier à cela en développant la notion de fonction (au sens mathématiques) chez les élèves. Samurçay a distingué quatre rôles de la variable :

- donnée ;
- compteur ;
- accumulateur ;
- intermédiaire pour la programmation.

Il est à noter que, lorsqu'une variable joue le rôle d'intermédiaire dans la programmation, il est plus difficile pour les élèves de comprendre le rôle de cette dernière quand elle n'existe que pour répondre à une exigence du système informatique. En outre, les compteurs et accumulateurs sont plus faciles à utiliser lorsqu'ils suivent les étapes qu'on ferait « à la main ».

Expliciter le rôle joué par les variables permet aux élèves de faire davantage de progrès.

Un point important est que « les élèves (et étudiants) débutants rencontrent des difficultés différentes selon les activités cognitives portant sur la variable informatique ». Se représenter les différentes opérations que l'on peut effectuer sur les variables (entre autre) peut être facilitée par une étape de déclaration des variables et de leur type.

Ainsi, les points importants sont l'opposition entre ce qui doit être écrit pour être compris par un ordinateur et l'écriture « papier-crayon » et le lien entre le problème à modéliser et la forme que doit prendre la modélisation.

Lors de leur conclusion, les auteurs insistent sur le fait que la création d'un algorithme ou d'un programme doit être vu comme le fait d'organiser l'évolution des valeurs de différentes variables.

II La notion de variable

II.1 Variables : place dans le programme et définition

La notion de variable informatique est présente dans le programme du cycle 4 et on la retrouve dans l'aménagement du programme de seconde :

CONTENUS	CAPACITÉS ATTENDUES	COMMENTAIRES
Variables et instructions élémentaires	• choisir ou déterminer le type d'une variable (entier, flottant ou chaîne de caractères) ; • concevoir et écrire des affectations à des variables ; • écrire une formule permettant un calcul combinant des variables.	On commence par consolider les notions de variables, de boucles et d'instructions conditionnelles introduites au cycle 4 en complétant la programmation par blocs par l'utilisation d'un langage de programmation textuel.

On remarque que les seuls types au programme sont entier, flottant et chaîne de caractère. Le type booléen n'y figure pas.

Comme indiquée dans [Mod12], une variable symbolise un emplacement mémoire. On dit qu'on affecte une valeur à une variable lorsqu'on « dépose » cette valeur dans l'emplacement mémoire désigné par la variable. Le type d'une variable est la nature des valeurs que l'on peut affecter à la variable. Ceux qui sont le plus utilisés sont le type booléen (valeurs vrai ou faux), le type entier (pour des nombres entiers), le type flottant (qui correspond aux nombres réels) et le type chaîne de caractères.

La borchure IREM [dlCL17] donne quatre propriétés des variables :

- **Nommage :** une variable représente une « adresse » (au sens informatique du terme) et cette adresse est accessible via le nom que l'on a donné à la variable. Il convient de choisir des noms explicites.
- **Durée de vie :** Une variable existe pendant l'exécution du programme et disparaît après la fin du programme (voir avant).
- **Unicité :** Une variable ne peut pas contenir plus d'une valeur à un moment donné.
- **Modification :** Au cours du programme, la valeur d'une variable peut être modifiée.

Il convient d'insister avec les élèves sur cette dernière propriété : une variable informatique est une « boîte » qui contient une valeur, que l'on peut changer. Comme on l'a vu dans [LR17], c'est la différence principale entre les notions de variable informatique et de variable mathématique qui sont distinctes, ce qui n'est pas toujours clair dans l'esprit des élèves. D'ailleurs, pour les élèves, la notion de variable mathématique est floue, certain la conçoivent comme un élément de l'ensemble de départ d'une fonction, certains élèves identifient même les notions de variables mathématiques et d'antécédents de fonctions.

Bien que les élèves n'aient pas une idée précise de ce que sont les variables informatiques, ils arrivent en général à les utiliser pour écrire des programmes simples avec Scratch. Lors du passage vers Python, il convient d'être conscient des différences et des similitudes en ce qui concerne la notion de variable entre Scratch et Python.

Tout d'abord, pour utiliser Scratch, il faut emboîter des blocs représentant des instructions. Cette méthode permet à l'utilisateur de ne pas avoir de problèmes syntaxiques. C'est en partie cela qui rend Scratch facile d'utilisation. Avec Python, il faut taper ses programmes et utiliser le formalisme propre à la programmation : il faut respecter précisément

la syntaxe et faire preuve de précision, par exemple lorsqu'on indente certaines lignes de code. C'est une difficulté supplémentaire pour les élèves.

Avec Scratch, il faut créer une variable dans le menu « Données » avant de pouvoir l'utiliser. Pour affecter une valeur à une variable, il faut utiliser le bloc « Mettre **variable** à **valeur** ». Avec Python, la variable est créée quand on lui affecte une valeur : la commande « = » dans l'instruction « x= 3 » va créer une variable x (si elle n'existe pas encore) et lui affecter la valeur 3. Il est à noter qu'un bloc contenant le symbole « = » existe dans Scratch, dans le menu opérateur, et qu'il sert à tester l'égalité de deux variables. Pour tester l'égalité dans Python, on utilise « == ». Ce point est important dans le cadre de l'instruction conditionnel. Il est à l'origine d'une des deux expérimentation en classe détaillée dans la partie III.2.

Enfin, une autre différence réside dans le nom des variables : on peut utiliser des espaces avec Scratch, ce qui est impossible avec Python. C'est une différence mineure mais qui peut être à l'origine de bugs lors de l'exécution d'un programme écrit en Python que les élèves auraient du mal à repérer.

L'une des similitudes entre Scratch et Python est liée au fait que les types des variables ne sont pas primordiaux lors de leur déclaration et affectation et que les variables peuvent changer de type au cours de l'exécution d'un programme. Il est tout de même pertinent de noter que le type d'une variable intervient avec Python lors de l'utilisation de la commande input. Si on écrit une des instruction Python suivantes :

```
1 V=input('Donner une valeur à affecter à V')
2 V=int(input('Donner une valeur à affecter à V'))
3 V=float(input('Donner une valeur à affecter à V'))
```

qu'on rentre « 36 » puis qu'on demande quelle est la valeur de la variable V à l'ordinateur, celui-ci répondra respectivement : '36' (chaîne de caractères), 36 (nombre entier) et 36.0 (nombre flottant). Autant les nombres entiers et flottants peuvent être utilisés pour les calculs autant les chaînes de caractères ne s'y prêtent pas du tout : dans le cas où on utilise la commande int ou float, à l'instruction V+V, l'ordinateur renverra 72 (ou 72.0) mais dans le premier cas, l'ordinateur renverra '3636'.

II.2 La vision des élèves a priori

Pour commencer la transition Scratch-Python avec les élèves, je me suis inspirée d'un document de l'académie de Bordeaux intitulé « De Scratch à Python ». (Voir l'annexe A). Cet exercice utilise le passage par le langage naturel : un programme est écrit en Scratch puis en langage naturel puis en Python. J'ai décidé de faire cet exercice en salle de classe et non pas sur les ordinateurs. En effet, l'interface utilisée pour Python (Idle) n'est pas facile d'usage pour la majorité des élèves (bien qu'on puisse faire une analogie avec les deux fenêtres de Scratch) et il vaut mieux traiter les difficultés potentielles une après l'autre.

En ce qui concerne la première partie, elle a été plutôt bien réussi. Certains élèves (minoritaire) ont rencontré des difficultés. La première était liée aux conditions à écrire au début des instructions conditionnelles : par exemple, certains ont écrit « si 10<moyenne et moyenne < 12 ou moyenne = 11 ». Pour remédier à cela, j'ai poussé les élèves à tester des cas particuliers, voire à leur dire de voir ce qui se passait quand la moyenne était égale à 10 ou 12. La seoncde difficulté était liée à la valeur à affecter à la variable moyenne, certains écrivant « $\frac{n1+n2+n3}{3}$ » et même $\frac{1+2+3}{3}$ ». On peut penser que les élèves n'auraient pas fait cela si l'activité avait été faite sur ordinateur.

La deuxième partie n'a pas été traitée par tous les élèves, certains n'ayant pas vu la question, d'autre la trouvant trop rébarbative. Certains ont réussi à traiter la troisième

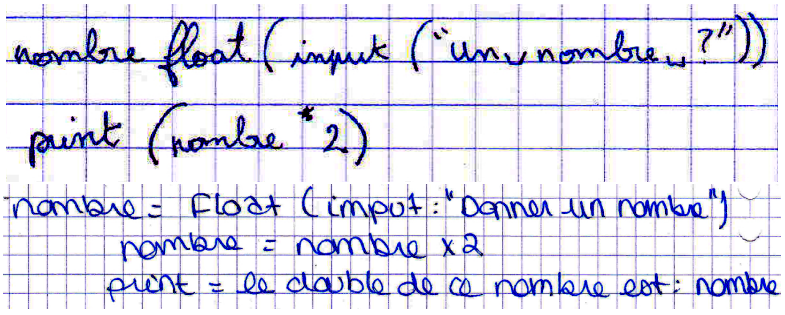
partie sans traiter la deuxième, d'autres ont eu des difficultés. Les élèves qui ont traité cette partie ont eu tendance à copier ce qui était fait avec Scratch : « Quand « drapeau » est cliqué, », « Si la condition « $12 \leq$ moyenne et moyenne < 14 » est vérifiée » (il aurait été plus naturel d'écrire « Si $12 \leq$ moyenne < 14 »). Bien que ce ne soit pas ce qui est attendu, je ne pense pas que cela ait des répercussions négatives. L'utilité de passer par le langage naturel est de formuler ce qu'on veut faire. La plupart des élèves commencent à écrire un programme sans avoir de vision d'ensemble ce qui les pénalise. Leur donner l'habitude de verbaliser leurs objectifs a priori facilitera leur progression.

La première question de la troisième partie a été bien réussie. On peut penser que les élèves ont recopié les lignes de codes à chaque étape sans comprendre en profondeur ce qu'ils faisaient (certains ont oublié de définir la variable « note3 »).

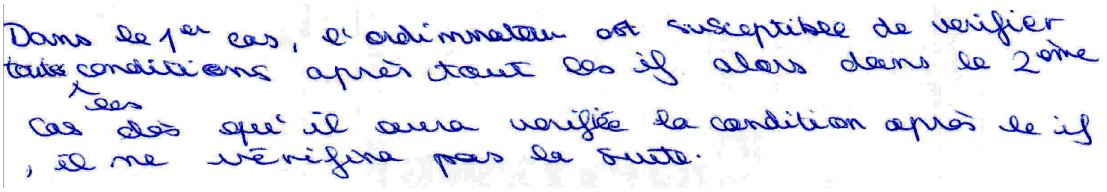
Pour essayer d'évaluer si les élèves avaient compris ou non, j'ai posé en contrôle (en bonus) l'exercice suivant : « Écrire en Python un programme qui demande un nombre et renvoie le double de ce nombre. **Indication :** On pourra utiliser « input », « print » et « float ». »

Seuls 5 élèves ont répondu. Leurs réponses montrait qu'ils comprenait ce qui était attendu d'eux mais la syntaxe de Python n'était pas respectée (ce qui est naturel). On peut penser qu'il y aurait eu plus de réponses si les élèves avaient eu plus de temps.

Voici deux extraits de copies :



Enfin, les dernières questions de cette activité d'introduction étaient destinées aux élèves les plus rapides. Quelques élèves ont traité la question 4 et seulement deux ont eu suffisamment de temps pour se pencher sur les questions 4 et 5. Pour la question 4, on attendait une utilisation de « elif ». Plusieurs ont réussi à l'utiliser correctement mais un seul a compris la différence entre if et elif au niveau de l'ordinateur, même si sa réponse a été formulée maladroitement :



À la suite de cet exercice d'introduction, j'ai demandé aux élèves ce qu'était pour eux une variable, 5 élèves (sur 35) ont donné une réponse qui montre qu'ils ont une bonne vision de cette notion. Par exemple :

- « Elles permettent de contenir des valeurs » ;
- « Elles permettent de stocker des données et des valeurs (informations) » ;

- « Les variables prennent la valeur indiquée [par l'utilisateur] [...]. Ensuite, elles vont être utilisées pour le calcul. »

D'autres ont donné des réponses qui laissent penser qu'ils ont compris certains des aspects de la notion, mais pas tous, en particulier l'aspect « boîte où l'on stocke des valeurs » est absent :

- « Valeurs qui sont répertoriées et traitée en fonction de l'utilisation voulue dans une formule. »
- « Elle varie en fonction de l'utilisateur » ;
- « Une variable est un bloc où l'on peut mettre n'importe quelle information dedans. C'est une écriture simplifiée dans ensemble d'informations ».

Plusieurs élèves affirment qu'on peut « mettre plusieurs choses » dans une variable, il est difficile de savoir s'ils ont bien compris qu'une variable ne prend qu'une seule valeur à la fois. Il est possible qu'ils pensent au fait qu'on peut affecter successivement des valeurs différentes à une variable.

Certains sont sensibles à la proximité sémantique entre « variable » et « varier » : « [ce sont des] nombres variant par rapport à plusieurs choix et possibilités en gardant la même signification. ».

Enfin, un élève fait la liste de ce qu'évoque chez lui la notion de variable :

« • variable (x,y)
- Rapport avec les fonctions ?
- Equation ? (x= ?)
- valeur positive/négative ?
- Repère "OIJ" ? »

Ici, on constate que l'élève ne fait pas la différence entre variables informatiques et mathématiques.

On peut ainsi conclure cette partie en observant que les connaissances qu'ont les élèves sont suffisantes pour réussir des exercices simples mais ne s'appuient pas sur une réelle compréhension de la notion de variable. Il était donc important de travailler davantage cette notion, ce que j'ai fait dans les exercices décrits dans la partie suivante.

III Expérimentation en classe

III.1 Une première séance dédiée à la notion de variable : échange de variables

III.1.1 Description de la séance et analyse a priori

J'ai d'abord fait le TP de l'annexe B avec les élèves. La première questions est la plus importante, la seconde devait être traitée par la majorité des élèves, les deux dernières s'adressent aux élèves les plus rapides, la dernière n'ayant pas de lien avec l'échange de variables mais nécessitant de choisir des entrées appropriées. J'ai fait ce TP en demi-groupe dans la salle informatique.

Ce TP a pour but d'amener les élèves a trouver un algorithme pour échanger les valeurs de deux variables puis d'implémenter ce dernier avec Scratch puis en Python. L'algorithme que je veux voir mis en œuvre se code de la manère suivante (avec Python) :

```
1 nombre1=float(input("Premier nombre"))
2 nombre2=float(input("Deuxième nombre"))
3 enveloppe=nombre2
4 nombre2=nombre1
5 nombre1=enveloppe
6 print("nombre1 vaut ",nombre1," et nombre2 ",nombre2)
```

La difficulté de cet exercice consiste en la création d'une troisième variable. Le premier algorithme auquel penseront les élèves est représenté par le programme suivant :

```
1 nombre1=float(input("Premier nombre"))
2 nombre2=float(input("Deuxième nombre"))
3 nombre1=nombre2
4 nombre2=nombre1
5 print("nombre1 vaut ",nombre1," et nombre2 ",nombre2)
```

Lorsqu'on exécute le programme précédent, les deux variables (nombre1 et nombre2) ont la même valeur à la fin du programme ce qui ne répond pas au problème posé.

La difficulté ici tient au fait que l'introduction d'une troisième variable n'est nécessaire que parce que l'on utilise un système informatique : lorsqu'on échange des valeurs sur le papier, les anciennes valeurs sont toujours accessibles, et donc l'utilisation d'un nouvel emplacement mémoire est superflu. Cela illustre ce qui a été écrit dans [LR17] au sujet de la variable comme intermédiaire dans la programmation.

L'objectif de ce TP est de faire prendre conscience aux élèves qu'une variable peut être assimilée une boîte (ou une enveloppe, qui est une représentation plus facile à utiliser en classe) dans laquelle on « range » une valeur.

Les élèves travaillent individuellement. Dans un premier temps, j'avais prévu de laisser les élèves chercher en autonomie. Sachant que la majorité d'entre eux n'allait pas trouver, j'ai préparé trois enveloppes avant la séance pour représenter des variables : j'ai écrit « a » sur la première, « b » sur la deuxième et « nouvelle variable » sur la troisième. J'ai pris deux

petites feuilles sur lesquelles j'ai écrit des valeurs (1 et 5). Le but était, après la première phase de recherche des élèves de leur montrer les enveloppes « a » et « b » (la troisième étant utile seulement à la fin de la séance pour montrer la réponses aux élèves qui n'auraient pas trouvé), de glisser les feuilles avec les valeurs dans les enveloppes et de manipuler devant les élèves pour leur faire comprendre cette modélisation du fonctionnement d'un ordinateur. J'ai préparé des feuilles à disposition des élèves pour qu'ils puissent en faire des enveloppes et des papiers sur lesquelles écrire les valeurs dans but que les élèves qui n'avaient pas trouvé puissent manipuler ces objets du monde réel et trouver une solution.

Durant la séance, les élèves ont à disposition un ordinateur (pour chaque élève), leur cahier (en particulier l'exercice décrit dans la partie précédente) et leur matériel scolaire. Je m'attendais à ce que les élèves n'utilisent l'ordinateur qu'après avoir trouvé un algorithme à la main.

Pour les deux premières questions (qui constituent le cœur du TP), aucune connaissance mathématique particulière n'est attendue des élèves (pour la première partie du moins). Au niveau des connaissances informatiques, les élèves doivent savoir utiliser Scratch : il faut écrire un script, créer des variables et modifier leurs valeurs, ce sont des connaissances disponibles. Les élèves sont également amenés à utiliser Python, c'est le premier contact sur machine, il faut donc qu'ils découvrent son utilisation et utilisent l'exercice fait précédemment (annexe A) en recopiant et modifiant les lignes de code qui sont pertinentes dans cet exercice.

Les élèves peuvent savoir si leur algorithme fonctionne grâce à la réponse de l'ordinateur une fois que l'algorithme est transformé en programme. J'avais aussi prévu de passer dans les rangs.

Deux remarques sont à formuler :

- Dans la représentation des variables avec des enveloppes, on « enlève » les valeurs contenues dans les enveloppes représentant les variables au cours de l'opération alors qu'un ordinateur va copier les valeurs (et ne pas les effacer). Je pense que tenter d'expliquer cette nuance aux élèves risque de leur compliquer la tâche pour un bénéfice négligeable.

- Dans cette algorithme, les variables ont le rôle de données et d'intermédiaire pour la programmation (en s'appuyant sur les quatre rôles définis dans [LR17]).

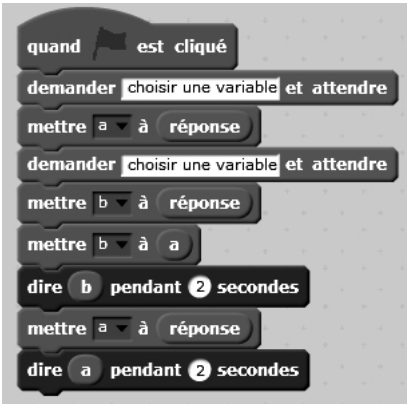
III.1.2 Réponses des élèves

Le TP s'est globalement passé comme prévu. 4 élèves (sur les 35) suivent l'enseignement d'exploration ICN et n'ont eu aucune difficulté (un a d'abord écrit un algorithme basé sur les sommes et différences de valeurs). Parmi les autres, un seul a trouvé l'algorithme sans manipuler les feuilles de papier. Lorsque j'ai proposé aux élèves la représentation des variables par des enveloppes, la grande majorité a réussi à trouver une solution en manipulant.

Certains élèves ont eu besoin d'un exemple (oral) pour comprendre la consigne.

Très peu d'élèves sont passé par une phase papier/crayon et la majorité a tenté d'écrire un programme avec Scratch directement. J'attribue cela au fait que les élèves n'ont pas l'habitude d'écrire un algorithme à la main et, au premier abord, cette question leur semble facile : ils pensent immédiatement au deuxième algorithme décrit au paragraphe précédent. L'avantage est que l'ordinateur invalide leur réponse.

Voici un exemple de tentative d'élève pour échanger les variables :



Ici, le programme est le résultat de plusieurs tentatives. L'élève a utilisé la variable « réponse » qui est définie par défaut comme troisième variable. On peut pensé qu'il n'a pas pleinement conscience de cela. Cette démarche n'est possible qu'avec Scratch et ne peut pas se transposer telle quelle en Python. Écrire un algorithme en langage naturel aurait permis à l'élève de voir que sa démarche n'était pas idéale.

Plusieurs élèves ont donné une solution faisant intervenir deux variables intermédiaires :

Cette solution marche parfaitement et, à un niveau seconde, on ne peut qu'être satisfait. Pour aller plus loin, on peut préciser aux élèves que ce programme utilise plus de place dans la mémoire de l'ordinateur qu'un algorithme qui serait plus efficace.

Les élèves qui ont eu le temps d'écrire le programme en Python ont écrit des programmes qui fonctionnaient bien. On retrouve ce qu'ont fait les élèves avec Scratch, et le même nom des variables. Par exemple, un élève a repris la méthode à 2 variables temporaires. Python n'a pas posé de grandes difficultés, il est à noter qu'un élève n'a pas utilisé la commande « input », on peut penser que c'est lié à Python :

```
1  a=4
2  b=6
3  c=b
4  b=a
5  a=c
6  print("a=",a)
7  print("b=",b)
```

Un élève a proposé la réponse suivante à la troisième question :

```
x=float(input("Valeur_de_x_?"))
y=float(input("Valeur_de_y_?"))
z=y
y=x
x=z
print("on_inverse_les_valeurs")
if y>x :
    print("la_valeur_la_plus_petite_est_x_égale_à",x)
elif x>y :
        print("la_valeur_la_plus_petite_est_y_égale_à",y)
```

Cette réponse serait parfaitement satisfaisante si l'élève avait traité le cas où $x = y$. Ici, l'ordinateur ne renverra rien. Cette difficulté fait partie de celle mentionnées dans [LR17] : le contrôle des structures conditionnelles n'est pas naturel, vérifier qu'on a traité tous les cas en nécessaire en informatique.

On peut considérer les noms que les élèves ont donné aux variables. Tout d'abord, certains ont donné des noms sans faire la différence entre les différents rôles des variables (deux jouent le rôle de données et une joue le rôle d'intermédiaire à la programmation) :

- A B C
- a b c
- x1 x2 x3
- x y z w
- x y z t

Certains les ont différenciées mais de façon peu visible :

- g h n m
- g h b b'
- nombre a nombre b NOMBRE C

Enfin, certains ont donné des noms explicites :

- x y y rappel
- nombre a nombre b autre
- a b valeur de b au début du script

Il convient d'encourager les élèves à donner des noms explicites (comme cela a été écrit dans [dlCL17]) pour rendre leurs programmes plus lisibles.

Globalement, cet exercice a été bien réussi et on peut espérer que les élèves ont une meilleure perception de la notion de variable. On peut insister sur l'aide qu'a apporté la manipulation physique aux élèves.

Afin de savoir à quel point les élèves avaient compris ce qui a été fait en TP, j'ai donnée l'exercice suivant en DS, en prévenant les élèves précisément ce sur quoi ils allaient être interrogés :

1. Expliquer (en français, avec un dessin...) comment échanger les valeurs de deux variables.
2. Écrire un programme en Python qui échange les valeurs de deux variables.

Rappel : on peut utiliser les instruction "float" "input" et "print".

Sur 34 élèves, 9 n'ont pas traité l'exercice, 4 n'ont pas abordé la question sur Python, 1 n'a pas du tout réussi à utiliser la syntaxe propre à Python, 5 ont réussi la question sur Python mais n'ont pas réussi (ou pas traité) la première question et enfin 12 ont parfaitement répondu (dont 7 qui ont eu des réponses très proches de la correction du TP). Parmi les élèves qui n'ont pas traité l'exercice, quatre sont en grande difficulté. On peut penser que les autres ont manqué de temps. La première question était révélatrice de la compréhension qu'avaient les élèves tandis que la seconde évaluait leur capacité à restituer un algorithme écrit en Python. À part pour un élève, la syntaxe était bien respectée, il y avait seulement quelques problèmes d'indentation pour lesquels les élèves n'ont pas été pénalisés. Ainsi, exactement la moitié de la classe a bien répondu à la première question, parmi ceux qui n'ont pas réussi, certains n'avaient pas évoqué l'existence de la troisième variable, intermédiaire de programmation (tout l'utilisant dans leur programme) et un n'avait pas compris la modélisation variable/enveloppe et valeur/papier dans l'enveloppe. Les résultats de la première question montrent que les élèves ont progressé (seul un élève ne suivant pas l'enseignement d'exploration ICN a réussi sans enveloppe). Les résultats de la seconde montrent que la moitié des élèves sont capables de restituer un programme qu'ils ont appris sans problème de syntaxe. Des extraits de copies sont en annexe C.

Ces résultats montrent que les élèves ont progressé et qu'il faut poursuivre les efforts et continuer d'insister sur la notion de variable.

III.2 Une deuxième séance : l'utilisation des variables dans le cadre de l'instruction conditionnelle

III.2.1 Description de la séance et analyse a priori

L'exercice de la partie précédente a permis aux élèves d'avoir une meilleure vision de la notion de variable. Cependant, il est nécessaire d'apprendre à utiliser cette notion, en particulier lors de la programmation en Python. L'instruction conditionnelle est une notion assez bien maîtrisée par les élèves d'après ce que j'ai pu constater pendant les TP Scratch, j'ai donc décidé de commencer par là et ai donné aux élèves l'exercice de l'annexe D. La première partie consiste à demander aux élèves d'écrire un programme qui demande à l'utilisateur combien fait 7×7 et qui renvoie une réponse différente selon qu'il ait raison ou non. Cela a déjà été fait en Scratch, la difficulté est donc centrée sur la variable lors de la programmation en Python et sur le test d'égalité. Ici, la variable a le rôle de donnée.

Les deuxièmes et troisièmes parties s'adressent aux élèves les plus rapides. Dans la deuxième partie, on utilise une variable qui a le rôle de compteur (pour compter les points) et la troisième partie utilise une boucle for et est destinée aux élèves avancés. Compter les points nécessite d'initialiser la variable compteur.

Comme précédemment, les élèves sont en demi-groupe dans la salle informatique (un ordinateur par élève).

La première partie est le cœur du TP. Le programme attendu est le suivant :

```
1 x=int(input("Combien fait 7*7 ? "))
2 if x==49 :
3     print("Bravo !")
4 else :
5     print("Non, ce n'est pas juste.")
```

Ici, les élèves n'ont besoin d'aucune connaissance mathématique particulière. Ils doivent avoir compris l'utilisation de input (incluant le « int » ou « float » le précédent) et la notion « crayon-papier » d'instruction conditionnelle, notion disponible pour beaucoup d'élèves. L'objectif est d'apprendre aux élèves à coder l'instruction conditionnelle en Python. Cela peut se décomposer en deux tâches :

- Imiter la syntaxe donnée dans l'énoncé : en particulier, il faut écrire « : » à la fin des lignes et indenter correctement.
- Réussi à écrire convenablement la condition à tester. Pour cela, il faut avoir réussi les deux premières questions qui différencient le « = » réservé à l'affectation et le « == » réservé au test d'égalité.

La validation des deux premières questions est effectuée par le professeur, la validation de la troisième est effectuée par l'ordinateur.

III.2.2 Réponses des élèves

La réponse de la majorité des élèves aux deux premières questions indique qu'ils ont compris la différence entre « = » et « == » mais les formulations sont quelque peu maladroites :

- « == » veut dire égal
- En Python, « == » veut dire « un nombre = un nombre ».

7 élèves ont donné une réponse satisfaisante, par exemple :

- « = » signifie « la variable prend la valeur de ... ». « == » signifie « la valeur ... est égale à la valeur ... Vrai ou Faux ? ».
- « = » peut servir à donner une donnée à quelque chose. Ex : a=2 la variable a a pour info le chiffre 2. « == » sert à prouver si deux choses sont égales. Si les deux données placées au début et à la fin de « == » sont égales, la réponse sera « True » et si les deux données ne sont pas égales, la réponse sera « False ».

Ce qui rend ces réponses satisfaisantes est le fait que les élèves ont compris (sans probablement être capable de le formuler) que l'instruction « == » renvoyait un booléen.

Pour la troisième question, les élèves ont rencontré les deux difficultés attendues.

Tout d'abord, plusieurs ont eu du mal avec la syntaxe de Python :

```
x=float(input("donner un nombre"))
x=float(input("résoudre 7*7"))
if  la valeur donner et 49 alors dire félicitation")
else la valeur donner n'est pas 49 alors faux")
```

Ici, l'élève a écrit le passage conditionnel en langage naturel, après en avoir parlé avec lui, il s'agissait d'une lecture trop rapide de l'énoncé.

```
nombre=float(input("Combien_fait_7*7_?"))
if 7*7==49 :
    "Félicitations"
else :
    "C'est_faux"
```

Ici, l'élève n'a pas utilisé la commande print.

```
print("combien_fait_7x7")
x=float(input("Donner_un_nombre")
        if("7^7==x")
        print("oui")
        else
        print("non")
```

Ici, l'élève n'a pas indenté convenablement et a oublié « : » à deux reprises.
D'autres élèves ont rencontré des problèmes logiques :

```
x=float(input("Valeur de 7*7"))
if x==49 :
    print("bravo")
```

Ici, l'élève n'a pas traité le cas où la réponse n'était pas la bonne. On peut penser que c'est lié à une mauvaise compréhension de « else ». En effet, deux autres élèves n'ont pas utilisé else et ont tenté de traiter les mauvaises réponses de la manière suivante :

```
x=float(input("donner un nombre"))
if x==49 :
          print("oui")
if x==2  :
          print("non")
```

```
6
7  if x==39 :
8              print("non")
9  if x==45 :
10             print("non")
```

Ici, l'élève a listé des mauvaises réponses possibles mais n'a pas pu remédie au manque d'exhaustivité de son programme. Je ne m'attendais pas à cette difficulté, les élèves arrivant globalement tous à utiliser le bloc « si ... sinon ... » de Scratch. On peut faire l'hypothèse que cette erreur repose sur une mauvaise lecture de l'énoncé qui précisait que « else » signifie « sinon ».

```
1  a=49
2  a==float(input("résoudre 7x7"))
3  if a>=49:
4              print("Faux")
5  else:
6              print("juste")
```

Ici, c'est la condition devant le « if » qui pose problème à l'élève. On remarque aussi qu'il affecte 49 à la variable a au début du programme, ce qui n'est pas utile. Cette affectation est due au fait que le programme a été modifié plusieurs fois : la deuxième ligne a été rajoutée après coup et on peut penser que l'élève a oublié d'effacer la première ou n'a pas compris qu'elle ne jouait plus aucun rôle.

Cela illustre la difficulté liée au contrôle de la correction des instructions conditionnelles mentionnée dans [LR17] lors de l'énumération des difficultés cognitives propres à la programmation.

On peut clore cette partie en mentionnant les problèmes des élèves qui ont traité la question 4. Trois programmes figurent en annexe E. Certains n'ont pas créé de compteur de points et leur programme commande l'affichage d'une phrase indiquant que l'utilisateur a un point supplémentaire. Parmi ceux qui ont pris créé une variable compteur, l'un d'entre eux ne l'a pas initialisée dans le cas ou la première réponse était fausse. Il a également eu des difficultés à utiliser la commande input au cours du programme alors que c'était bien fait dans la première ligne. Cela permet de supposer que l'élève n'a pas eu le temps de tester son programme et de corriger les erreurs. Un élève n'a pas utilisé de compteur et a utilisé 6 instructions conditionnelles à la suite à la fin de son programme semblables à celle-ci :

```
1  if x==49 and g==36:
2      print("2points")
```

La manière de contourner la difficulté est intelligente et utilise « and » qui n'a pas été vu dans le contexte de Python mais seulement dans celui de Scratch. L'élève aurait pu

utiliser « or » pour diminuer le nombre de lignes de code. Cet élève a aussi oublié de traiter le cas ou toutes les réponses étaient fausses, ce qui attribuait 0 points à l'utilisateur.

Enfin, la plupart des élèves qui ont eu le temps de chercher cette question ont trouvé un programme qui répond à la question. Il est intéressant de noter qu'une nouvelle variable réponse a été créée à chaque question, les élèves n'ont pas utilisé la même. S'ils avaient eu le temps d'aborder la dernière partie, cela aurait été un frein à l'utilisation de la boucle for.

Conclusion

La notion de variable est une notion peu maîtrisée par les élèves de seconde. Certes, la notion de variable mathématique est un précurseur à celle de variable informatique. Cependant, lors de l'apprentissage de l'algorithmique et de la programmation, le fait que la valeur de la variable informatique puisse changer est quelque chose de nouveau pour les élèves mais d'absolument fondamental.

Les élèves en début de seconde arrivent à créer des programmes simples dans lesquels la variable joue le rôle de donnée mais dès que le rôle de la variable change, cela génère des difficultés : les élèves ont du mal à créer une variable jouant le rôle d'intermédiaire dans la programmation et dont l'existence n'est nécessaire qu'à cause du système informatique, ils ont également du mal quand la variable a le rôle de compteur et qu'il faut penser à l'initialiser. On peut penser que cela est dû au fait que les élèves n'ont pas une bonne représentation de ce qu'est une variable et de la manière dont le système informatique fonctionne. En ce qui concerne la représentation de la variable, on peut proposer la représentation boîte/enveloppe aux élèves qui a l'avantage de lier l'algorithmique à la manipulation pratique. En ce qui concerne le fonctionnement du système informatique, une remédiation que l'on peut proposer est de détailler avec les élèves chaque étape effectuée lors de la mise en œuvre d'un programme informatique par un ordinateur.

Les élèves ont des difficultés à appréhender la notion de variable et la retravailler spécifiquement en classe de seconde semble être nécessaire pour garantir aux élèves des bases solides en algorithmique et programmation. L'occasion pour cela est l'apprentissage de Python. Mais il ne faut pas négliger le fait qu'écrire un programme en Python nécessite un apprentissage de la syntaxe surtout lorsqu'il faut l'implémenter sur ordinateur. Les difficultés propres à Pyton en ce qui concerne la notion de variable que les élèves vont rencontrer en premier sont l'utilisation du signe = pour l'affectation (et == pour le test d'égalité) et la commande input. En effet, pour que celle-ci fonctionne, il faut avoir conscience du type de variable dont on a besoin.

Dans la suite de mon cours, j'aborderai les boucles avec les élèves. Les difficultés auxquelles ont peu s'attendre sont le passage du « jusqu'à » de Scratch au « while » (tant que) de Python. Cela nécessitera de nier une condition ce qui n'est pas simple pour les élèves (comme cela a été détaillé dans [LR17]). En outre, on peut avoir besoin d'une variable compteur dans les boucles (par exemple pour sortir d'une boucle while), rôle qui n'est pas naturel pour les élèves. Les élèves ont eu beaucoup de difficulté avec la boucle for avec Scratch, il faudra veiller à détailler cette notion avec Python, en particulier à cause de la variable qui va être incrémentée à chaque étape. Enfin, la notion de fonction sera à aborder. Tout comme variables mathématiques et informatiques, il faudra insister sur la différence entre fonctions mathématiques et informatiques. On peut penser que les fonctions mathématiques jouent le rôle de précurseur des fonctions informatiques, les deux notions étant très proches. Les différences étant en autre autre qu'une fonction informatique peut ne pas avoir d'argument (alors qu'une fonction mathématique est toujours appliquée à un élément, hors cas pathologique). De plus, une fonction informatique a ce qu'on appelle des effets de bord (elle peut par exemple afficher des messages). Enfin, elle ne renvoie pas toujours les mêmes valeur à arguments fixés (on peut penser à une fonction qui donne la date par exemple).

Je n'ai pas fait le choix de faire de cours à mes élèves mais de leur faire découvrir les notions à travers des exercices. On peut se demander si faire un cours récapitulatif (ou un polycopié) dans lequel toutes les notions seraient définies précisément aurait un impact positif non négligeable. Il sera également intéressant de demander à nouveau à mes élèves ce

que représente une variable pour eux et comment on peut l'utiliser afin de voir l'évolution de leur appréhension de la notion de variable.

L'informatique prend une place de plus en plus importante dans notre société, il est nécessaire d'initier les élèves à cette science Cela requiert un investissement pédagogique pour permettre à tous les élèves de progresser dans cette discipline.

Bibliographie

[BO] Aménagement du programme de mathématiques. Bulletin officiel numéro 18 du 4 mai 2017.

[dB17] Académie de Bordeaux. De Scratch à Python, 2017.

[dlCL17] Groupe de la CII Lycée. Algorithmique et programmation au cycle 4, commentaires et recommandations du groupe informatique de la cii lycée, 2017. Imprimé par l'IREM de Paris - Université Paris Diderot Paris 7.

[Edu] Eduscol. Algorithmique et programmation, ressources pour le lycée.

[LR17] Jean-Baptiste Lagrange and Janine Rogalski. Savoirs, concepts et situations dans les premiers apprentissages en programmation et en algorithmique. *Annales de didacique et de sciences cognitives*, pages 119–158, 2017.

[Mod12] Simon Modeste. *Enseigner l'algorithmique, pourquoi ? Quels apports pour l'apprentissage de la preuve ? Quelles nouvelles questions pour les mathématiques ?* Thèse de doctorat, Université de Grenoble, 2012.

Annexes

A Premier exercice

Nom, Prénom : .....
Seconde Année 2017-2018

TP 6 : Découverte de Python

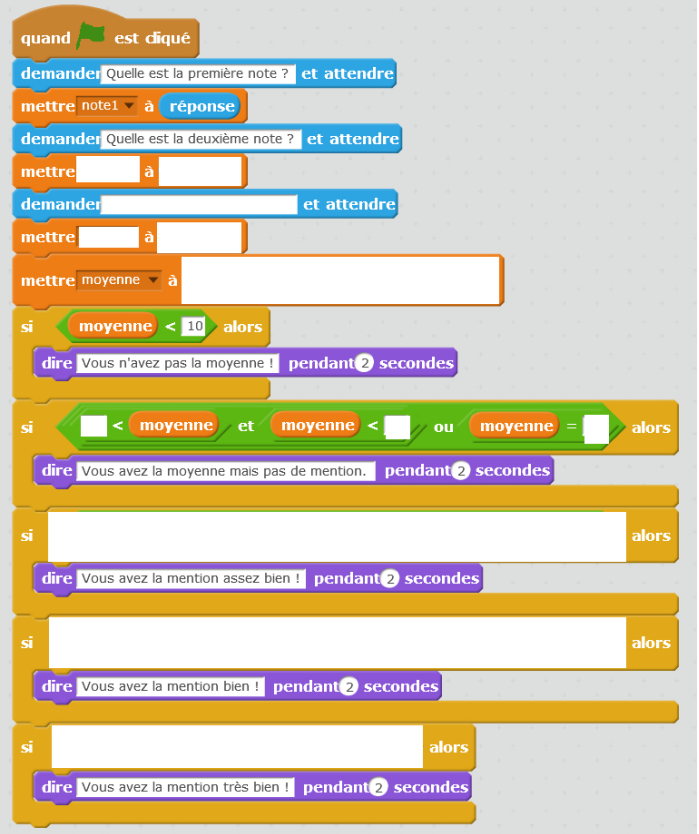
Le but est d'utiliser l'ordinateur pour calculer la moyenne d'un élève ayant eu trois notes (de même coefficient) et renvoyer la mention qu'il aurait s'il s'agissait du bac.

On rappelle que :

- en dessous de 10 (strictement), on rate le bac ;
- entre 10 (inclus) et 12 (exclus), on n'a pas de mention ;
- entre 12 (inclus) et 14 (exclus), on a la mention assez bien ;
- entre 14 (inclus) et 16 (exclus), on a la mention bien ;
- au-dessus de 16 (16 inclus), on a la mention très bien.

I Rappel sur Scratch

1. Compléter le script suivant pour qu'il réponde au problème posé :



1

II De Scratch vers le français

2. En utilisant le tableau joint, écrire cet algorithme en français.

III Du français à Python

3. En utilisant le tableau ci-joint et vos réponses précédentes, compléter le programme écrit en Python :

```
note1=float(input("Quelle est la première note ?"))
note2 = ..................................................
.........................................................
moyenne = ...............................
if moyenne<10:
    print("Vous n'avez pas la moyenne")
if 10<=moyenne<12:
    print ..................................................
if  ....................................................
    print("Vous avez la mention assez bien !")
if  ....................................................
    .....................................................
.........................................................
    .....................................................
```

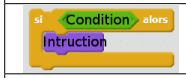
4. En vous aidant du tableau, trouver une autre manière d'écrire ce programme (très proche de la précédente). Comparer ce que fera l'ordinateur pour chacune des deux méthodes.

IV Pour aller plus loin

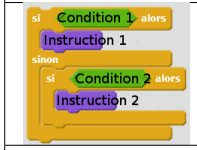
5. Modifier le programme (en Scratch, en français puis en Python) pour que l'utilisateur puisse donner des coefficients.

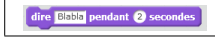
6. Modifier le programme (en Scratch, en français puis en Python) pour que l'utilisateur puisse rentrer ses 2 première notes et la mention qu'il veut obtenir (aucune, AB, B, TB) et que l'ordinateur lui renvoie la note minimale qu'il doit avoir à la troisième évaluation pour avoir la mention qu'il veut.

7. (Difficile) Modifier le programme (en Scratch, en français puis en Python) pour que l'utilisateur puisse rentrer le nombre de notes qu'il veut (après avoir précisé ce nombre à l'ordinateur).

Scratch	Français	Python
	$\frac{2+3}{4}$	`(2+3)/4`
	$x < 5$	`x<5`
	$x \leq 5$	`x<=5`
	La variable x prend la valeur 5	`x=5`
	Afficher « Valeur de x ? » et affecter à x la valeur donnée par l'utilisateur.	`x=input("Valeur de x ?")` [a]
	Si la condition est vérifiée alors suivre les instructions	`if condition:` `  instructions`

	Si la condition est vérifiée alors suivre l'instruction 1 sinon suivre l'instruction 2	`if condition:`
		`  instruction 1`
		`else:`
		`  instruction 2`

	Si la condition 1 est vérifiée alors suivre l'instruction 1 sinon, si la condition 2 est vérifiée alors suivre l'instruction 2	`if condition 1:`
		`  instruction 1`
		`elif condition 2:`
		`  instruction 2`

	Afficher « Blabla »	`print("Blabla")`

a. Lorsque l'utilisateur répond, Python pense qu'il s'agit d'une chaîne de caractère (des lettres, des mots...). Il faut lui expliquer qu'il s'agit d'un nombre entier : x=int(input("Valeur de x ?")) ou d'un nombre pas forcément entier (un nombre « flottant ») : x=float(input("Valeur de x ?"))

B Séance sur l'échange de variables

Seconde Année 2017-2018

TP 7 : Premiers pas avec Python

Tous les programmes devront être enregistrés et devront m'être envoyés par Educ'Horus.

I Échanger les valeurs de deux variables

Le but de cette partie est d'échanger les valeurs de deux variables.

1. Écrire en français un algorithme permettant d'échanger les valeurs de deux variables (après qu'un utilisateur les a choisies). Le tester avec Scratch.

2. Une fois que l'algorithme fonctionne bien avec Scratch, l'écrire en Python.

   Remarque : On utilisera IDLE pour programmer en Python. Lancer IDLE. La première fenêtre que l'on voit est similaire à celle dans laquelle se trouve le chat dans Scratch. Ouvrir un nouveau document. La fenêtre qui s'affiche est similaire à celle dans laquelle on emboîte les blocs dans Scratch. Taper le programme dans cette fenêtre et cliquer sur « Run » puis « Run module » (ou appuyer sur F5) pour tester le programme.

3. Modifier ce programme pour que ce dernier tri les valeurs dans l'ordre croissant : le programme doit renvoyer une phrase comme « La plus petite valeur est 1 et la plus grande valeur est 3. »

   Remarque : Pour utiliser la commande « print » en Python, on met le texte entre crochets, si on veut afficher la valeur affectée à une variable, on écrit simplement la variable et si on veut afficher plusieurs éléments différents, on les sépare par une virgule. Par exemple, si la variable x a la valeur 2, la commande

   ```
   print("On a affecté à la variable x la valeur",x)
   ```

   affiche « On a affecté à la variable x la valeur 2 » .

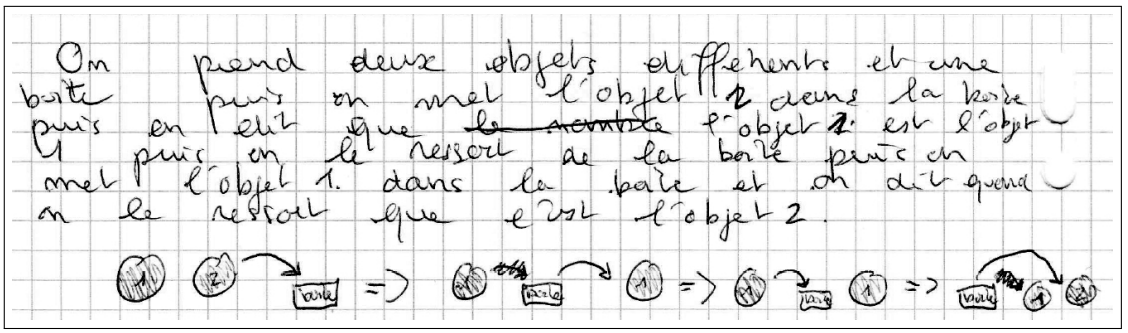
II Pour réviser le cours sur les droites

4. Écrire un programme qui vérifie qu'un point appartient à une droite. L'utilisateur choisit l'équation de la droite et les coordonnées du point. Commencer par décrire l'algorithme en français puis l'écrire en Python.

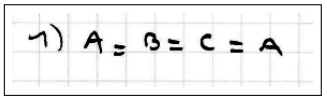
C Copies d'élèves sur l'échange de variables

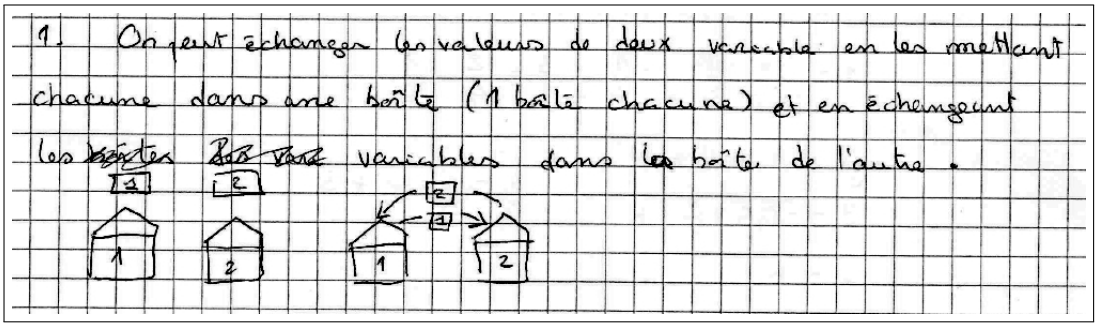
Question 1

Un élève n'a pas du tout compris la différence entre valeur et variable :

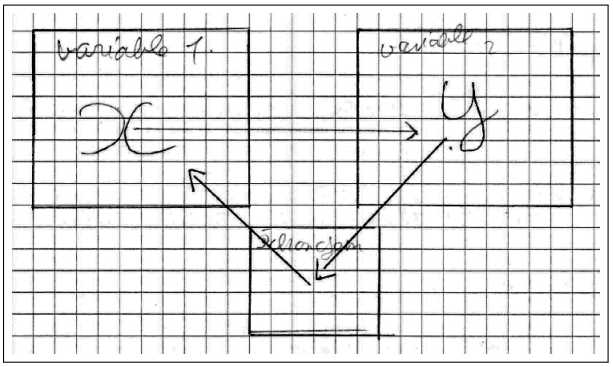


Plusieurs élèves n'ont pas précisé l'ordre des étapes d'affectation :

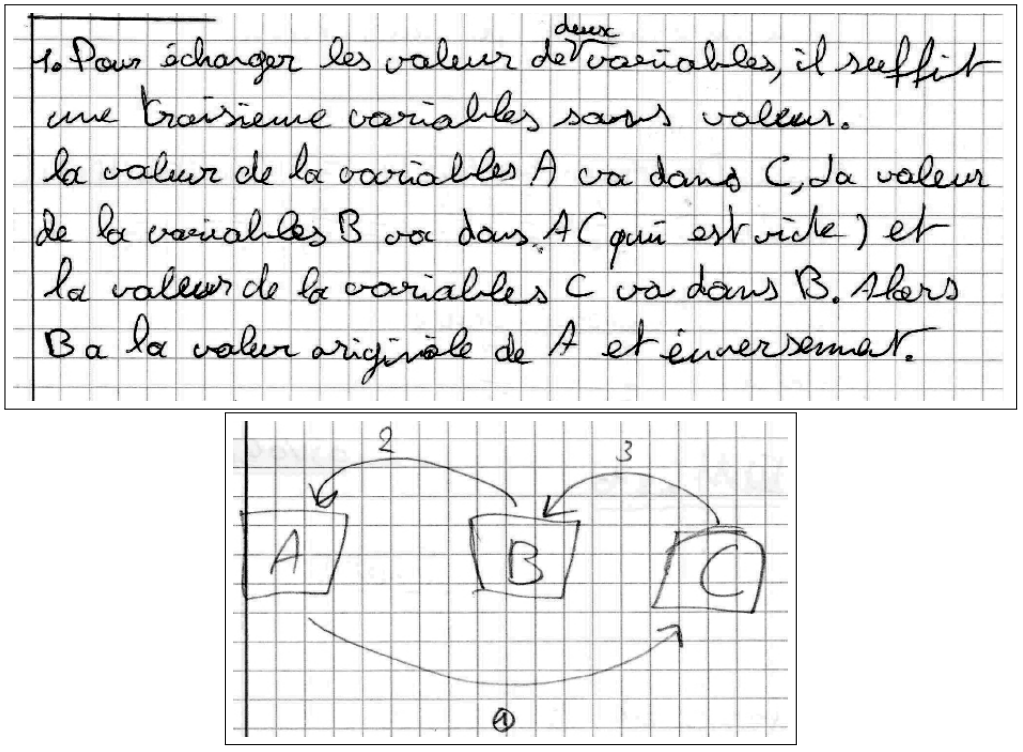




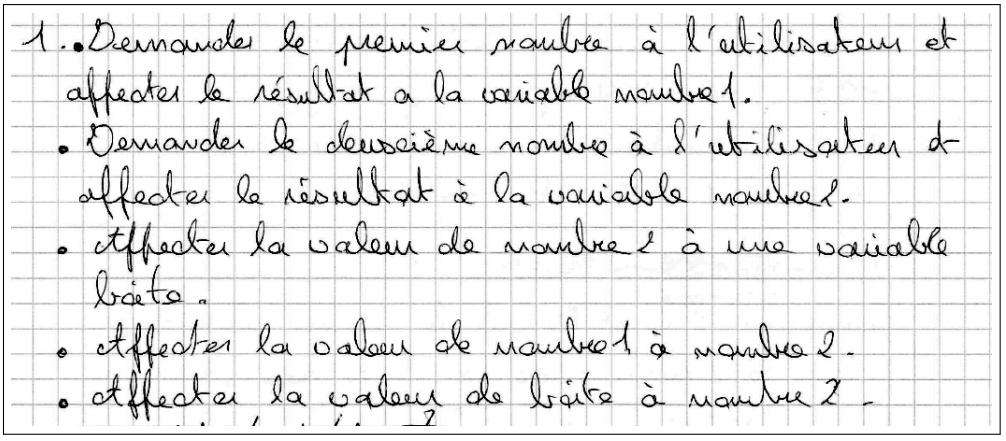
Un élève utilise le nom « échangeur » pour la troisième variable, nom très cohérent :



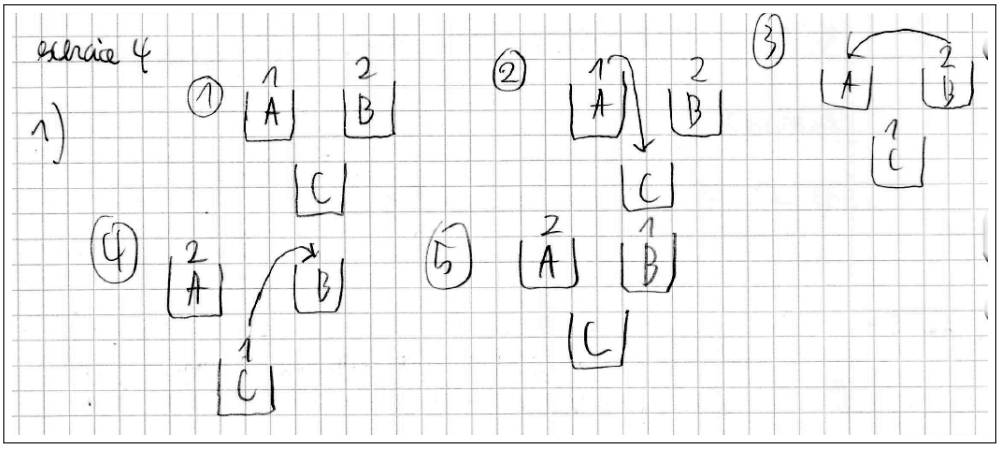
L'analogie variable/enveloppe pousse un élève à croire que des variables peuvent être « vides » :



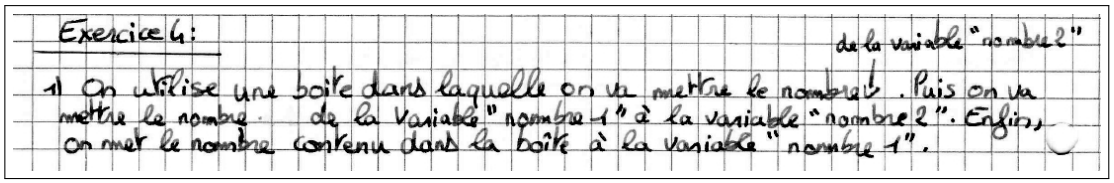
Parmi les réponses satisfaisantes, on retrouve des solutions proches du corrigé de TP :



Un élève fait un schéma dans lequel on retrouve la dichotomie variable/valeur :

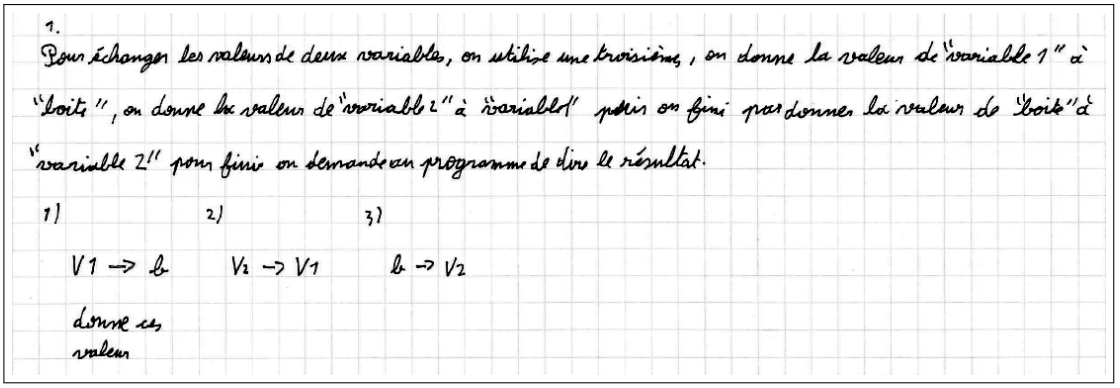


Un élève assimile boîte et variable :


Exercice 4 :

1) On utilise une boite dans laquelle on va mettre le nombre 1. Puis on va mettre le nombre de la variable "nombre 1" à la variable "nombre 2". Enfin, on met le nombre contenu dans la boîte à la variable "nombre 1".

Un élève schématise la situation à l'aide d'une flèche dont il définit le sens (« donne ses valeurs ») :


1.

Pour échanger les valeurs de deux variable, on utilise une troisième, on donne la valeur de "variable 1" à "boite", on donne la valeur de "variable 2" à "variable1" puis on finit par donner la valeur de "boite" à "variable 2" pour finir on demande au programme de dire le résultat.

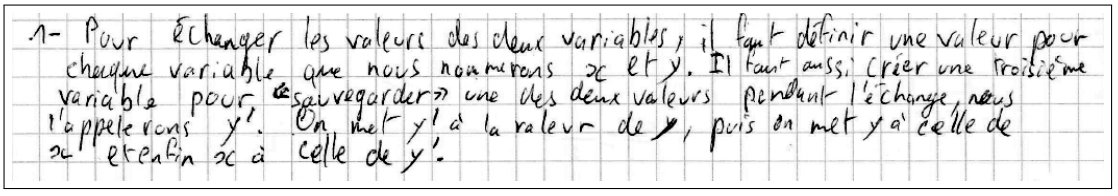
1) V1 -> b

2) V2 -> V1

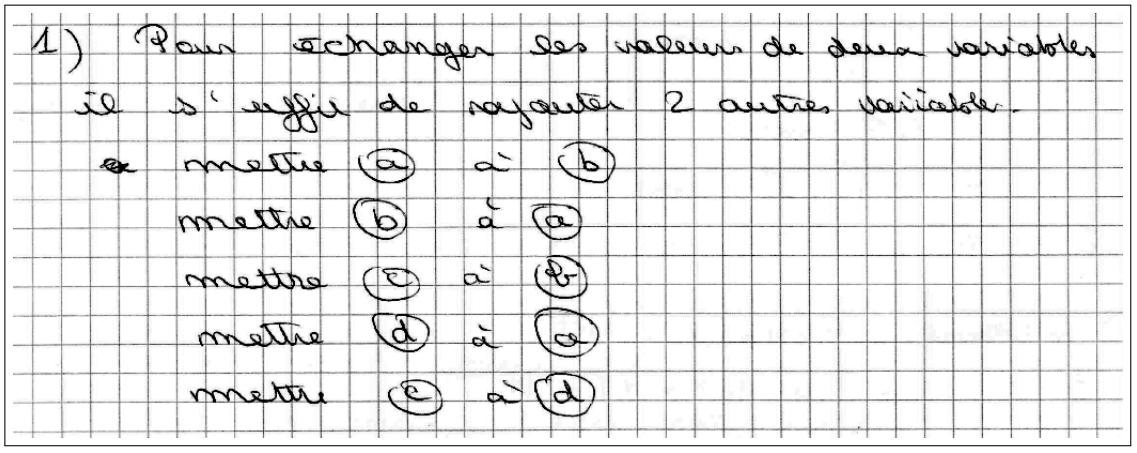
3) b -> V2

donne ses valeur

Une solution d'un élève justifiant l'utilisation d'une troisième variable :


1- Pour échanger les valeurs des deux variables, il faut définir une valeur pour chacune variable que nous nommerons x et y. Il faut aussi créer une troisième variable pour sauvegarder une des deux valeurs pendant l'échange, nous l'appelerons y'. On met y' à la valeur de y, puis on met y à celle de x et enfin x à celle de y'.

On retrouve la solution à 4 variables d'un élève :


1) Pour échanger les valeurs de deux variables il s'agit de rajouter 2 autres variable.

mettre (a) à (b)

mettre (b) à (a)

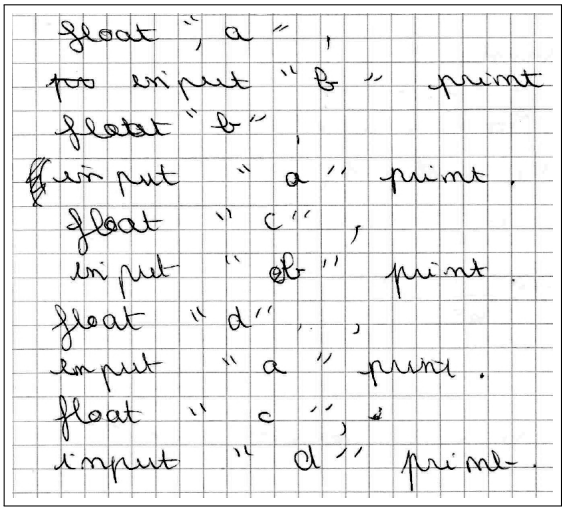
mettre (c) à (b)

mettre (d) à (a)

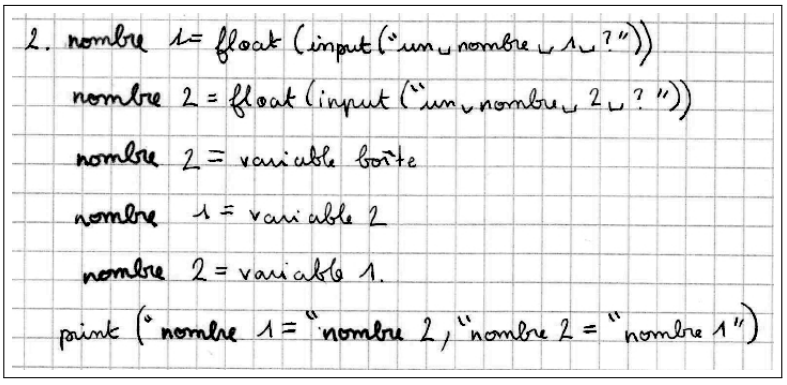
mettre (c) à (d)

Question 2

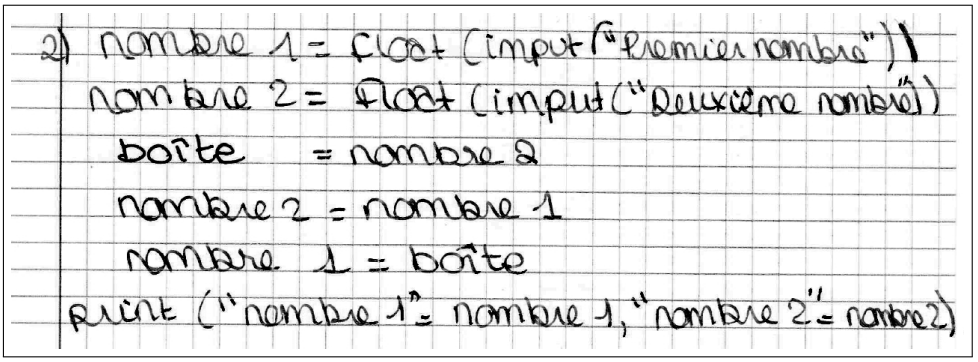
Voici la solution de l'élève qui n'a pas du tout assimilé la syntaxe de Python :

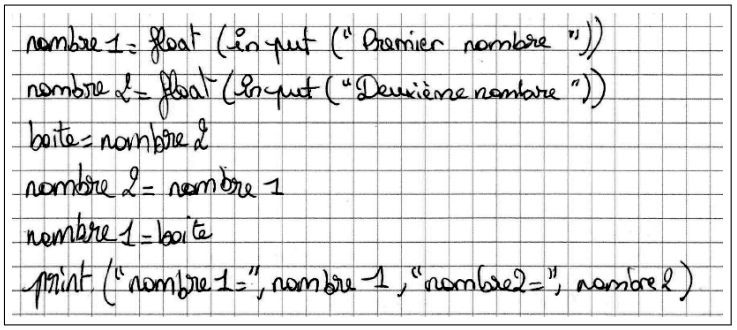


Un élève n'a pas compris qu'en informatique on désignait une valeur par la variable dans laquelle elle est stockée et n'a pas utilisé la boîte dans l'avant dernière ligne (étourderie probablement, la question 1 était bien traitée) :



La majorité des élèves qui a traité cette question l'a bien réussi :





D Séance sur l'instruction conditionnelle

Seconde Année 2017-2018

TP 8 : L'instruction conditionnelle

Pour programmer en Python, ouvrir IDLE.

I Découvrir l'instruction conditionnelle avec Python

1. Que signifie « = » en Python ?
2. Taper « 2==4 » dans la fenêtre shell (valider en appuyant sur entrée) puis « 4==4 ». Que signifie « == » en Python ?
3. Programmer ce qui suit : l'ordinateur demande à l'utilisateur combien fait 7×7. Il félicite l'utilisateur si ce dernier a trouvé et lui dit qu'il a faux sinon.

   Remarque : L'instruction conditionnelle s'écrit comme cela en Python :

```
1 if condition:
2     suivre cette instruction
3 else:
4     suivre cette instruction-là
```

   (« if » signifie « si » et « else », « sinon ».)

II Compter les bonnes réponses

4. Modifier le programme précédent pour qu'il pose 3 questions. Chaque bonne réponse donne un point à l'utilisateur. L'ordinateur doit renvoyer le nombre de bonnes réponses à la fin.
5. Maintenant, on veut que l'ordinateur donne un multiplication à effectuer en choisissant les deux facteurs au hasard. **Remarque :** La commande « randint(0,2) » donne un nombre entier au hasard entre 0 et 2. Pour pouvoir l'utiliser, il faut écrire « from random import randint » au début du programme.

III Pour aller plus loin

6. Modifier le programme pour qu'il pose un grand nombre de questions (au moins 10). **Remarque :** Utiliser une boucle for. Vous pouvez recopier les deux lignes suivantes pour comprendre comment cette boucle fonctionne en Python.

```
1 for i in range(1,6):
2     print("Bonjour !")
```

E Réponses des élèves à la question 3 du TP8

Programme 1 : L'élève ne compte pas les points.

```
x=int(input("Donner 7*7"))
if x==49:
    print("Tu as un point!")
else:
    print("Faux")

y=int(input("Donner 7*5"))
if y==35:
    print("Tu As un point!")
else:
    print("Faux")

z=int(input("Donner 7*8"))
if z==56:
    print("Tu as un point")
else:
    print("Faux")
```

Programme 2 : L'élève n'initialise pas son compteur et a des difficultés avec la syntaxe.

```
x=float(input("7_*_7"))
if x==49:
   print ("bravo tu as un point de plus")
   p=float("1")
else:
   print ("raté")

z=print(float("combien fait 11*11"))

if z==111:

    print ("bravo tu as un point de plus" )
    p=p+1
else:

    print("raté")

y=print(float(" combien fait 2*2"))

if y==4:

    print("bravo tu as un point de plus")
```

```
23          p=p+1
24      else:
25          print("raté")
26      print("tu as" , p, " points")
```

Programme 3 : L'élève ne crée pas de compteur mais y remédie avec une instruction conditionnelle.

```
1   x=float(input("Combien_fait_7x7"))
2   if x==49:
3       print("félicitation")
4   else:
5       print("non")
6
7
8   x=float(input("Combien_fait_7x7"))
9   if x==49:
10      print("1+point")
11  else:
12      print("non")
13  g=float(input("Combien_fait_6x6"))
14  if g==36:
15      print("1+point")
16  else:
17      print("non")
18  r=float(input("Combien_fait_2x2"))
19  if r==4:
20      print("+1point")
21  else:
22      print("non")
23  if x==49 and g==36 and r==4:
24      print("3points")
25  if x==49 and g==36:
26      print("2points")
27  if x==49:
28      print("1point")
29  if g==36 and r==4:
30      print("2points")
31  if g==36:
32      print("1point")
33  if r==4:
34      print("1point")
```

Programme 4 : Une des réponses satisfaisantes.

```
1  total=0
2  reponse1=float(input("Combien font 7*7"))
3  if reponse1==49:
4      print("Félicitation")
5      total=total+1
6  else:
7      print("Faux")
8
9  reponse2=float(input("Combien font 1*1"))
10 if reponse2==1:
11     print("Félicitation")
12     total=total+1
13 else:
14     print("Faux")
15
16 reponse3=float(input("Combien font 2*50"))
17 if reponse3==100:
18     print("Félicitation")
19     total=total+1
20 else:
21     print("Faux")
22
23 print("vous avez",total," bonnes réponses.")
```