Block2Py, un éditeur de blocs pour l'apprentissage du langage Python

Christophe Declercq, Florence Nény

Block2Py, un éditeur de blocs pour l'apprentissage du langage Python

Christophe Declercq[1] et Florence Nény[2]

[1] Centre de Recherche en Education de Nantes, Université de Nantes
[2] IREM de Marseille

Résumé La transition d'un langage de programmation par blocs à un langage textuel constitue pour les élèves un obstacle qui mérite d'être outillé. Plusieurs environnements de programmation par blocs fondés sur Blockly - PyBlock[2], SofusPy[4] - proposent une traduction automatique vers le langage Python, pour tenter de simplifier ce passage. Cette approche à l'avantage de familiariser l'élève à la lecture de programmes textuels mais a pour défaut de ne permettre qu'une maîtrise indirecte du code Python généré.

On propose avec l'environnement de programmation Block2Py une approche différente, dans laquelle l'élève manipule directement des blocs correspondant aux constructions du langage Python. On postule que l'usage de cet environnement a un impact positif sur l'apprentissage du langage tant au niveau syntaxique que sémantique. On conclut en commentant les premières expérimentations, en formation d'enseignants puis en classe, de ce nouvel environnement.

Mots-clé : Programmation par blocs · Langage Python · Environnement de programmation

1 Introduction

Le premier obstacle pour les élèves, lors de la transition d'un langage de programmation par blocs à un langage textuel comme Python, réside dans l'apprentissage de la syntaxe. Alors que les programmes par blocs sont corrects par construction, la bonne imbrication d'un programme Python dépend du bon placement des mots clés du langage, de l'indentation et de quelques éléments lexicaux dont le deux-points. Le second obstacle, particulièrement avec un langage à typage dynamique, est d'apprendre à combiner les expressions et instructions du langage de manière cohérente au niveau des types.

Parmi les environnements existants, pouvant faciliter cette transition, on distingue tous ceux utilisant la bibliothèque de programmation par blocs Blockly [3]. Blockly n'est pas un langage de programmation, dans la mesure où la sémantique des blocs n'est définie que par la traduction vers un autre langage. C'est une bibliothèque permettant aux développeurs de créer des environnements. Dans l'exemple de PyBlock, la sémantique est donnée à la fois par traduction en Javascript pour la simulation, et en Python pour donner à voir aux élèves un

code dont l'exécution est supposée fournir un résultat équivalent. Des règles de traduction sont prédéfinies dans la bibliothèque et peuvent être modifiées. Par exemple le bloc `repeter n fois` est traduit en `for i in range(n)`. Ceci permet aux élèves de lire des programmes Python corrects et correspondant à leur saisie par blocs.

Cependant, sachant qu'il est difficile de faire coïncider les deux sémantiques, les systèmes automatiques de traduction de blocs en Python utilisent des schémas de traduction particuliers pour contourner les difficultés. Par exemple la traduction standard du bloc `demande un nombre` est `float(input())` qui enchaîne directement la lecture d'une chaîne et sa conversion en nombre flottant. Dans l'environnement PyBlock, le bloc `sinus` est traduit par `math.sin(math.radians())`. Le code généré automatiquement ne peut donc pas être complètement maîtrisé par l'élève. Il est parfois illisible ou contient des instructions (`try .. except`) que l'enseignant aurait préféré ne pas devoir présenter.

Notre hypothèse de départ concernant l'alphabétisation au langage Python, est de dire qu'il est utile d'apprendre à lire pour apprendre à écrire mais que cela n'est pas suffisant. Dans une approche résolument constructiviste, nous proposons à l'élève un environnement exploratoire dans lequel il peut manipuler directement des blocs Python, essayer de les imbriquer, constater que les expressions ne s'emboitent les unes dans les autres que si les valeurs des opérandes sont cohérentes avec l'opérateur ou la fonction utilisée, et où les imbrications d'instructions ne peuvent produire que des programmes syntaxiquement corrects.

2 Proposition d'un éditeur de blocs Python

L'environnement Block2Py - construit avec la bibliothèque Blockly - est un éditeur par blocs, où les blocs disponibles sont tous des constructions standard du langage Python, limité à un fragment choisi pour l'initiation au langage.

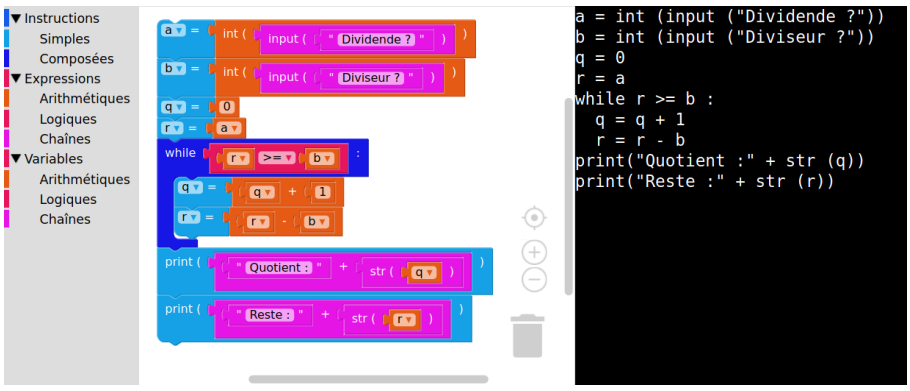

Figure 1. Edition d'un programme avec Block2Py

On a ainsi fait le choix de limiter les expressions aux entiers, chaînes de caractères et booléens, de limiter les constructions aux boucles usuelles - `for`, `while` - et aux conditionnelles et alternatives et de ne proposer comme instructions de base que l'affectation et l'impression - `print`.

L'interface contraint de plus l'utilisateur à choisir le type de ses variables au moment de leur création. Ce milieu didactique n'est pas imposé par le langage Python, mais est choisi pour guider l'élève vers l'écriture de programmes corrects par construction aussi bien au niveau de la syntaxe qu'au niveau des types.

On a volontairement séparé dans l'interface instructions et expressions pour permettre la construction par les élèves de la spécificité de ces deux concepts.

— Les instructions, qui s'emboitent avec la précédente ou la suivante dans une séquence, ou s'imbriquent dans une des constructions `while`, `for`, `if`, `if else`, sont les blocs avec des connecteurs en haut et en bas.
— Les expressions, qui ont une valeur, sont les blocs avec un connecteur à gauche, chargé de transmettre cette valeur à l'expression englobante.

L'usage de l'éditeur syntaxique, consiste simplement à prendre successivement des blocs dans le bandeau de gauche, à les déposer sur l'espace de travail au milieu et à les connecter ou imbriquer selon les règles de connexion prédéfinies.

L'élève peut ainsi se concentrer sur son intention, par exemple "imbriquer une affectation dans une boucle", sans devoir se préoccuper de la position des deux-points ni de l'indentation des lignes de code.

La partie droite de l'application est une zone de texte généré automatiquement, qui contient à tout moment le code Python correspondant au programme par blocs. Sa génération est particulièrement simple, car elle ne retient que les éléments textuels du langage sans rien ajouter qui n'ait été programmé par l'élève, à l'exclusion des parenthèses.

En effet la décompilation des expressions imbriquées a été programmée pour inclure les seules parenthèses indispensables pour que l'évaluation de l'expression Python corresponde à la structure des blocs. Ainsi si l'élève construit une expression algébrique : 

, il obtient son écriture parenthésée : `(x + 1) * (x - 1)`.

La saisie d'un programme par blocs est terminée quand il ne reste plus d'espace libre dans un bloc. On peut alors assurer que le programme construit par l'élève est un programme Python, correct par construction au niveau de la syntaxe et au niveau des types. Les seules vérifications qui ne sont pas effectuées concernent les propriétés de sémantique dynamique, en particulier la bonne initialisation des variables avant leur usage ou l'absence d'opérations illégales comme la division par 0.

Pour tester son programme, il suffit à l'élève de copier le texte du programme Python, pour le coller dans l'interprète Python de son choix. On privilégie biensûr les environnements dédiés à l'apprentissage comme l'éditeur Mu[1].

Block2Py n'est donc qu'un éditeur par blocs de programmes Python. On n'a volontairement pas inclus d'interprétation des blocs pour éviter tout conflit de sémantique. La seule sémantique d'un programme Python est celle mise en oeuvre par un interprète conforme à la norme du langage.

3 Analyse de situations d'apprentissage

Les situations d'apprentissages que l'on a proposées et étudiées sont toutes des activités constructives dont l'objectif est de permettre à l'élève de construire des schèmes pertinents d'usage des constructions du langage.

Les premières activités de découverte de la programmation Python permettent de construire des programmes comportant lectures de données, calculs, affectations et écriture de résultats. Ces activités amènent progressivement l'élève à intégrer les contraintes de type lors de l'usage des fonctions de lecture et d'écriture et induisent l'usage des fonctions de conversion de type `int` et `str`.

Les activités de programmation itérative permettent de découvrir les règles d'usage des boucles en Python à savoir l'usage d'une variable dans une boucle `for` et celui de conditions booléennes dans une boucle `while`.

Les activités nécessitant des réponses dépendant des valeurs des données, induisent l'usage des conditionnelles ou des alternatives et permettent à l'élève de découvrir la nature des expressions pouvant figurer dans une condition.

Les activités de programmation plus complexes, amènent à réfléchir globalement à la structure de la solution à apporter, en construisant un programme où sont imbriquées correctement de nombreuses constructions.

Les différentes situations d'apprentissage proposées ont fait l'objet d'un atelier lors d'une formation d'enseignants consacrée à la transition collège - lycée. L'analyse a priori laisse entrevoir les possibles apports de cette approche pour la première initiation au langage Python.

Il convient maintenant de mener des expérimentations en classe avec les enseignants formés.

4 Conclusion

Les perspectives futures concernent l'étude comparative de l'approche didactique proposée avec Block2Py avec celle proposée avec SofusPy ou PyBlock. En réalité ces deux types d'approches pourraient ne pas être directement concurrentes mais plutôt complémentaires.

Après avoir appris à *lire* du code Python avec PyBlocks, on pourrait proposer aux élèves d'apprendre à en *écrire* avec Block2Py.

Références

1. H.Tollervey, N. : Mu editor, https ://codewith.mu/
2. Joly, V. : PyBlock, http ://mathematiques-medias.discipline.ac-lille.fr/PyBlock/
3. Pasternak, E., Fenichel, R., Marshall, A.N. : Tips for creating a block language with Blockly. In : IEEE Blocks and Beyond Workshop (2017)
4. Raffinat, P. : Activités pédagogiques Python ou Blockly avec SofusPy. Revue MathémaTICE **63** (2019)