TigerJython et suivi de classe : vers une classe inversée en cours d'informatique

Cédric Donner[1]

[1] Haute École Pédagogique Vaud, UER MT, 1007 Lausanne, Suisse
cedric.donner@hepl.ch

Résumé. L'utilisation de méthodes pédagogiques actives s'impose comme une évidence dans l'enseignement de la programmation et de la pensée informatique puisqu'il s'agit de développer chez les apprenants une pensée créative de résolution de problèmes. Cependant, étant donnée l'hétérogénéité des apprenants dans une même classe, en termes de motivation, de développement cognitif et de compétences en résolution de problèmes, il devient vite très difficile de gérer la classe et d'intervenir de manière pertinente pour répondre aux divers besoins. Cet article présente plusieurs outils ayant favorisé l'implémentation d'une classe inversée dans un cours d'option complémentaire informatique dans un lycée suisse (Collège du Sud, Bulle). L'article présente notamment divers outils de suivi de classe ainsi qu'un aperçu de l'IDE pédagogique TigerJython qui fournit aux élèves des messages d'erreur en français et d'une pertinence souvent bien meilleure que ceux de l'interpréteur Python standard.

Mots-clés. Pensée informatique, Python, TigerJython, classe inversée, suivi et gestion des élèves

Abstract. It is no secret that computer science courses are particularly well suited to be taught using active pedagogical methods as computational thinking requires the development of creative thinking and problem-solving abilities. However, the heterogeneity of the student's motivation as well as their cognitive and problem-solving abilities is not without posing several difficulties for the teacher willing to optimally manage the class and bring relevant feedback to those that are stuck. This article presents several tools supporting a flipped classroom in an optional computer science course at a swiss college (Collège du Sud, Bulle). Among those, this article presents an overview of the TigerJython IDE which integrates a customized Python parser able to give more relevant and comprehensible error messages in the student's mother tongue. Other tools allowing a certain degree of student tracking are also mentioned.

Keywords. Computational Thinking, Python, TigerJython, flipped classroom, student progress tracking and management

1 Introduction

Tout enseignant qui s'est essayé aux méthodes pédagogiques actives sait qu'elles impliquent un sacrifice au niveau de la maîtrise des éléments au sein de sa classe. Ceci n'est pas nécessairement un problème puisqu'une telle situation peut favoriser certains objectifs essentiels de la formation tels que l'autonomie, la confiance en soi, les compétences sociales, de communication et de résolution de problèmes. Cependant, dans le cadre d'un cours d'informatique, il arrive souvent que les étudiants soient bloqués pour de nombreuses raisons largement discutées dans la littérature, notamment [1–6]. Ces difficultés proviennent le plus souvent des erreurs de syntaxe ou d'exécution dans les programmes, d'une compréhension insuffisante ou erronée d'un concept fondamental ou du modèle d'exécution sous-jacent, de problèmes techniques ou du syndrome de la feuille blanche face à un problème à résoudre.

L'enseignant, de son côté, est souvent démuni pour intervenir de manière pertinente auprès des apprenants qui en auraient le plus besoin car il ne dispose pas d'une vue d'ensemble de la progression de ses étudiants. Il doit donc souvent courir d'un étudiant bloqué à l'autre, chose impensable dans une grande classe. L'introduction du cours d'informatique obligatoire va rendre cette difficulté d'autant plus grande que les étudiants seront généralement moins motivés et moins matures que dans un cours à option.

L'IDE TigerJython permet de soulager l'enseignant de devoir intervenir auprès de ses étudiants pour une bonne partie des erreurs de syntaxe et d'exécution. Il permet en effet aux apprenants d'être plus autonomes dans leur démarche de correction d'erreurs. Le débogueur visuel intégré permet en particulier de comprendre et corriger les erreurs liées à une mauvaise compréhension du modèle d'exécution sous-jacent. D'autres outils, tels que les sites de France IOI, Runestone Interactive, REPL.it ou une feuille de calcul Google Sheet peuvent d'un autre côté permettre de garder un certain suivi des élèves au sein de la classe pour intervenir de manière plus pertinente.

2 Présentation de l'IDE TigerJython

L'IDE TigerJython a été récemment développé par T. Kohn dans le cadre de sa thèse [7] menée à l'École Polytechnique Fédérale de Zürich. Cette thèse a permis de classifier de nombreuses erreurs souvent commises par les débutants en programmation Python et d'y apporter une remédiation efficace dans un grand nombre de cas à l'aide d'un analyseur syntaxique Python personnalisé intégré à l'IDE. Les erreurs signalées de manière pertinente ne se limitent pas à celles considérées par la grammaire du langage Python comme étant des erreurs de syntaxe mais également à toute une quantité de motifs de programmation erronés souvent rencontrés dans les programmes de débutants et causant des erreurs sémantiques ou d'exécution.

L'interface utilisateur de l'IDE TigerJython est particulièrement adaptée à une utilisation par des débutants car elle est très épurée mais néanmoins très puissante. L'IDE comporte notamment (1) un éditeur puissant doté de complétion automatique (2) un débogueur pas-à-pas permettant de visualiser à tout instant l'état de la mémoire de même que l'instruction qui va être exécutée au prochain pas et (3) un espace réservé aux messages d'erreur qui sont, pour la plupart, reportés dans la langue maternelle de

l'apprenant et souvent bien plus compréhensibles que ceux retournés par l'interpréteur Python standard. De plus, pour les programmes utilisant le module `gturtle`, une fenêtre (4) permet de constater les mouvements de la tortue et ainsi de bien visualiser l'exécution du programme.

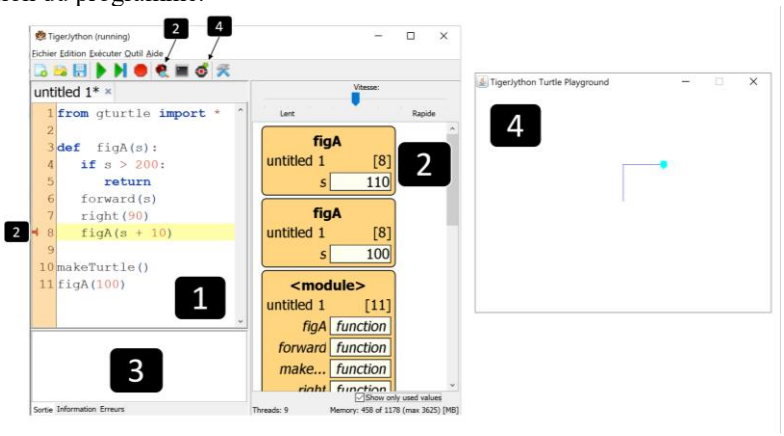

Fig. 1. Interface utilisateur de l'IDE TigerJython montrant (1) l'éditeur de code, (2) le débogueur pas-à-pas, (3) la fenêtre de notifications, (4) le terrain de jeu de la tortue permettant de visualiser l'exécution du programme selon la philosophie LOGO.

2.1 Messages d'erreur pertinents et compréhensibles

De nombreux travaux ainsi que l'expérience montrent que les débutants ont beaucoup de peine à prendre le temps de lire les messages d'erreur fournis par l'environnement d'exécution de leur programme. Dans de nombreux cas, les messages sont en effet cryptiques et ne mettent pas le doigt sur le véritable problème. À cela s'ajoute le fait que si le message est rédigé dans une langue étrangère et affiché dans une autre fenêtre que celle où se trouve le code, il y a vraiment peu de chances que le jeune apprenant, de moins en moins aguerri à la lecture, prenne le temps pour bien le comprendre et en tirer profit pour corriger son programme. Pour remédier à ce problème, l'IDE TigerJython intègre un parseur personnalisé qui effectue une analyse du code Python avant de le soumettre à l'interpréteur Jython comme le montre la figure 2.

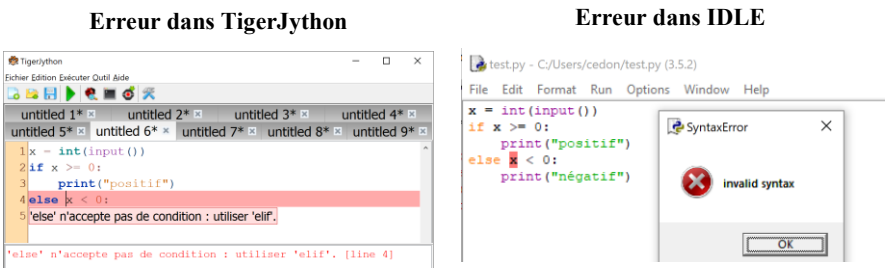

Fig. 2. Comparaison entre le message d'erreur fourni par TigerJython (à gauche) et celui, bien moins pertinent pour l'apprenant, fourni par l'interpréteur Python standard (à droite).

2.2 Pertinence des messages d'erreur de TigerJython

Le tableau 1 montre le gain de pertinence apporté par les messages d'erreur de TigerJython par rapport aux messages d'erreur fournis par l'interpréteur Python standard.

Tableau 1. Comparatif des messages d'erreur entre TigerJython et l'interpréteur Python.

Programme erroné	TigerJython	Python

`x = 5`
`if x < 3 == True:`
`  print("x is less than 3")`
	Dans ce cas, la comparaison avec `'True'` est superflue. [line 2]	-

`def return_two_vars(a, b):` `  return a and b`	Pour combiner plusieurs valeurs, il faut utiliser des virgules et non 'and'. [line 2]	-

`if x = 0:`
`  print("x est nul")`
	Pour tester l'égalité, il faut utiliser l'opérateur `'=='`. [line 1]	`if x = 0:`
		`  ^`
		`SyntaxError: invalid syntax`

`x = int(input())`
`if x >= 0:`
`  print("positif")`
`else x < 0:`
`  print("négatif")`
	'else' n'accepte pas de condition : utiliser 'elif'. [line 4]	`else x < 0:`
		`  ^`
		`SyntaxError: invalid syntax`

`x = input()`
`if x = = 0:`
`  print("nul")`
	Il y a un espace en trop. [line 3]	`if x = = 0:`
		`  ^`
		`SyntaxError: invalid syntax`

`word = 'chocolat'`
`count = 0`
`for "c" in word:`
`  count += 1`
	Le boucle 'for' nécessite une variable de contrôle. [line 4]	`for "c" in word:`
		`  ^`
		`SyntaxError: can't assign to literal`

`123 * 5 = variable`	La destination d'une assignation doit se trouve à gauche de l'opérateur d'assignation et non à droite. [line 1]	`123 * 5 = variable`
		`  ^`
		`SyntaxError: can't assign to operator`

2.3 Le revers de la médaille

L'étude présentée dans [7] aux pages 111 à 116 montre que, même si les messages d'erreur sont traduits dans la langue maternelle et rendus plus compréhensibles, les apprenants ne vont pas pour autant leur prêter une attention suffisante bien qu'ils apprécient de manière générale leur meilleure intelligibilité. Cette étude souligne également qu'il est extrêmement délicat de vérifier expérimentalement si ces messages personnalisés font une différence significative pour les apprenants. Quoi qu'il en soit, le fait qu'ils soient écrits dans la langue maternelle ôte la mauvaise excuse que se donnent parfois ceux qui sont faibles en anglais de ne pas parvenir à les lire. Au pire, ces messages d'erreur pourraient induire l'apprenant en erreur et mener à des fausses conceptions sur le langage de programmation. Ils pourraient également avoir pour effet discutable de fournir trop vite aux étudiants une interprétation aboutie de l'erreur, les

dispensant ainsi d'une certaine manière de la nécessité de mieux comprendre par eux-mêmes la logique fondamentale du langage et de leur programme.

Afin d'éviter ces écueils, il pourrait être pertinent de fournir, dans un premier temps, uniquement le message d'erreur du Python standard et, si ce dernier ne permettait pas à l'apprenant de déterminer rapidement les causes du problème, de lui fournir, dans un deuxième temps seulement, le message traduit et plus compréhensible.

Une autre solution, mise en place dans la classe en question, a consisté à explicitement valoriser l'effort de rédaction d'un rapport d'erreur détaillé contenant une interprétation correcte du message d'erreur, une description des causes du problème ainsi qu'une proposition de correction. Cela a permis d'encourager la mise en place, dans la pratique du programmeur débutant, d'une démarche d'investigation clairement documentée, lui permettant de devenir de plus en plus autonome. Il faut cependant passablement de patience pour qu'une telle démarche intellectuelle devienne une habitude naturelle chez la plupart des apprenants. En effet, lors de la première épreuve où les étudiants de cette classe disposaient de la possibilité de récupérer des points en exposant de manière détaillée la démarche mise en place pour identifier et régler les problèmes, un seul étudiant sur les neuf que comptait la classe en a vraiment profité. Les autres se sont en général contentés de décrire le comportement erroné de leur programme et d'émettre des hypothèses parfois hasardeuses quant à ses causes présumées, sans toutefois mettre en place une démarche systématique de débogage.

3 Enseigner la pensée informatique en classe inversée avec le manuel TigerJython

L'IDE TigerJython a été conçu en parallèle du support de cours [8], originellement rédigé en allemand ([9]). Il a la particularité de mettre l'apprentissage de la programmation au service du développement de la pensée informatique et non l'inverse, ce qui se traduit bien dans la structure et les intitulés des chapitres. En effet, ce ne sont pas les concepts de programmation de bas niveau qui sont centraux (variables, boucles, conditions, listes, chaînes de caractères, …) mais les notions informatiques de haut niveau et leurs applications. Le manuel fait en particulier la part belle à l'application des concepts informatiques au sciences naturelles et aux mathématiques au travers de simulations et d'expériences, sans oublier la robotique ou la conception orientée objets. Il aborde également, de manière accessible, des sujets centraux tels que la théorie de la complexité, la cryptographie, la programmation parallèle ou la théorie de l'information.

Pour permettre une bonne compréhension des concepts de base de la programmation et du modèle d'exécution d'un programme Python, le manuel contient un grand nombre d'exemples de programmes intéressants touchant à divers domaines d'application de l'informatique. L'approche suivant la philosophie LOGO ainsi que le débogueur intégré permettent de bien saisir les notions de base de la programmation et de développer une bonne compréhension du modèle d'exécution d'un programme Python grâce à la visualisation de l'état des variables en mémoire.

Les possibilités de l'IDE TigerJython en matière de débogage et ses messages d'erreur plus compréhensibles permettent d'envisager de manière réaliste un dispositif de classe inversée dans lequel il est demandé aux étudiants d'étudier de manière autonome,

à domicile, les notions et les exemples de programmes fournis dans le manuel. Un tel dispositif peut se justifier en cours d'informatique puisque, comme le souligne [4], les environnements de programmation donnent de nombreux retours de bas niveau tel que les erreurs de syntaxe mais très peu de retours pertinents de haut niveau concernant la manière de concevoir les programmes et d'agencer habilement les éléments du langage. Un des avantages d'une telle classe inversée est de pouvoir bénéficier de plus de temps en classe pour échanger avec les étudiants à ce sujet. Ce dispositif se heurte cependant à différents problèmes relevés dans la littérature, notamment l'adhésion des étudiants, étudiée dans [10]. Une des manières dont cette difficulté se traduit en pratique est que certains étudiants ont tendance à ne pas réaliser de manière consciencieuse ce travail de préparation en autonomie, étant plutôt habitués à ce que le travail leur soit prémâché en classe. D'autre part, certains étudiants n'ont pas nécessairement la maturité pour évaluer leur degré de compréhension du texte et peuvent passer à côté de notions importantes sans s'en rendre compte.

Pour pallier ces difficultés, deux stratégies de contrôle ont été mises en place. D'une part, les étudiants devaient noter le temps passé sur chaque activité proposée en autonomie dans un document Google Sheet permettant à l'enseignant d'avoir un aperçu rapide du travail fourni par chacun. Ils avaient également la possibilité de poser des questions ou signaler des incompréhensions par messagerie électronique. Ce moyen de contrôle a porté ses fruits et permis de constater par exemple, sur une période de deux semaines, une baisse d'engagement et de motivation de la part des étudiants en partie due à une surcharge de travail dans d'autres disciplines. Il a permis également de constater que certaines activités étaient trop difficiles à réaliser en autonomie puisque plusieurs étudiants n'y étaient pas parvenus ou y avaient consacré un temps très important. D'autre part, ils avaient reçu pour consigne de se mettre à la place de l'enseignant et de noter, dans un document Google Docs partagé avec toute la classe, les questions de compréhension qui pourraient être posées dans une épreuve concernant les notions de programmation ou les exemples de code fournis dans la lecture. Cette deuxième stratégie s'est révélée payante également puisque, la plupart du temps, la classe était parvenue à identifier, par les questions posées, les concepts essentiels présentés. Lorsque la classe n'a pas été en mesure d'identifier les concepts importants, c'est l'enseignant qui s'est chargé de poser des questions pertinentes pour qu'elles puissent être discutées et développées en classe.

Malheureusement, l'environnement TigerJython ne permet pas de récolter automatiquement les programmes réalisés par les étudiants et il est de ce fait encore difficile d'avoir un bon suivi des difficultés rencontrées par les étudiants.[1] De plus, le manuel [8] étant essentiellement basé sur des exemples à étudier, il souffre d'un manque d'exercices pratiques permettant d'entraîner les compétences de base de programmation. De ce fait, le cours est complété par l'usage de plateformes d'apprentissage de la programmation telles que France IOI[2], Runestone Academy[3] ou REPL.it[4] qui permettent de pallier ce manque et fournissent un outil de suivi de classe avec tableau de bord intégré.

[1] Cette possibilité existe en réalité puisque le travail de classification des erreurs des débutants en Python a été rendue possible par une étude systématique des programmes collectés. Cette fonctionnalité n'est cependant pas accessible à l'enseignant lambda.

[2] http://www.france-ioi.org/

[3] https://runestone.academy/

[4] https://repl.it/

La plateforme France IOI propose notamment un parcours d'exercices et de théorie extrêmement bien étudié et progressif permettant à l'apprenant d'avoir un retour immédiat sur sa solution à l'aide d'un système d'évaluation automatique et de corrigés lui permettant de juger de la qualité de sa solution. Cette plateforme met également à disposition, pour les premiers niveaux touchant les compétences attendues dans un cours de base, à un tableau de bord permettant à l'enseignant de connaître pour chaque élève les problèmes résolus, le nombre de tentatives de soumission erronées ainsi que le temps passé sur chacun des problèmes. La plateforme serait cependant encore plus utile si elle permettait à l'enseignant d'avoir accès aux programmes corrects ou incorrects soumis par ses étudiants. Ce désavantage peut être contourné en demandant aux étudiants de maintenir un dépôt GitHub avec les problèmes réalisés. Un autre inconvénient de France IOI est qu'il n'est pas possible d'y créer soi-même des problèmes et que le parcours est figé et non personnalisable. La plateforme REPL.it peut venir combler cette lacune puisqu'elle permet de définir des problèmes personnalisés, y compris des tests permettant de valider automatiquement les solutions des étudiants. Elle permet en outre d'accéder aux codes soumis par les étudiants.

La plateforme Runestone Academy, malheureusement uniquement anglophone, propose quant à elle de textes explicatifs de grande qualité ainsi que des exercices interactifs, notamment sous forme de questionnaires à choix multiple ou de petits programmes à réaliser ou à compléter. Elle fournit également un tableau de bord permettant à l'enseignant d'effectuer un suivi de classe des lectures effectuées par chaque étudiant et des scores réalisés aux différents questionnaires ou exercices de codage intégrés.

4 Conclusion

L'utilisation de l'environnement de programmation TigerJython permet aux débutants en programmation de gagner en autonomie grâce aux messages d'erreur pertinents qui évitent des blocages trop fréquents et au débogueur adapté leur permettant de chercher par eux-mêmes les causes des erreurs de programmation plus difficiles à saisir. Cela libère en partie l'enseignant de devoir assister les apprenants pour de broutilles et lui permet de les encourager à gagner en autonomie. Ce gain de disponibilité lui permet d'intervenir davantage pour les difficultés de haut niveau qui nécessitent une discussion approfondie.

Les tableaux de bord fournis par les plateformes de programmation telles que France IOI, Runestone Academy ou REPL.it permettent en outre de garder un œil sur la progression de chacun dans la classe, ce qui est indispensable dans un contexte où le travail autonome et actif des apprenants est encouragé.

Il faut cependant noter qu'il n'est pas agréable, autant pour les apprenants que pour l'enseignant, de devoir jongler avec toute une panoplie de plateformes et tableaux de bord différents. Une plateforme unifiée réunissant toutes ces fonctionnalités et permettant de récolter encore davantage de données sur la progression des étudiants serait d'une grande aide pour les cours d'informatique obligatoires qui vont voir le jour dans les années à venir, notamment en Suisse à partir de 2022. À ce titre, la plateforme Learnflow [11], développée à la HEPL (Haute École Pédagogique de Lausanne) pourrait être prometteuse. Il s'agit d'une plateforme généraliste non dédiée à la

programmation permettant de créer des parcours d'apprentissage composés d'activités intégrant du contenu de n'importe quel autre site Web. Cela permet de créer un support pédagogique unique et cohérent intégrant néanmoins la grande variété et richesse de contenus présents sur le Web. Le grand intérêt de Learnflow est de fournir un unique tableau de bord très puissant permettant de suivre la progression de chaque étudiant, de voir le temps passé sur chacune des activités qui peut être validée soit par l'étudiant lui-même, soit par un autre étudiant (évaluation par pair) soit par l'enseignant. Il n'est pour le moment pas possible d'utiliser les fonctionnalités de correction automatique fournies par des plateformes telles que France IOI pour valider les tâches. Learnflow permet également à l'étudiant de valider son activité en soumettant des remarques, ce qui pourrait permettre de collecter facilement le code source des problèmes de programmation. La plateforme Pathwright [12] semble également intéressante et fournir des fonctionnalités même plus avancées que celles de Learnflow en matière de création de cours et de suivi de classe.

Références

1. Boulay, B.D.: Some Difficulties of Learning to Program. J. Educ. Comput. Res. 2, 57–73 (1986).
2. Tan, P.H., Ting, C.Y., Ling, S.W.: Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception. In: 2009 International Conference on Computer Technology and Development. pp. 42–46 (2009).
3. Piteira, M., Costa, C.: Learning Computer Programming: Study of Difficulties in Learning Programming. In: Proceedings of the 2013 International Conference on Information Systems and Design of Communication. pp. 75–80. ACM, New York, NY, USA (2013).
4. Butler, M., Morgan, M.: Learning challenges faced by novice programming students studying high level and low feedback concepts. (2007).
5. Mow, I.T.C.: Issues and Difficulties in Teaching Novice Computer Programming. In: Innovative Techniques in Instruction Technology, E-learning, E-assessment, and Education. pp. 199–204. Springer, Dordrecht (2008).
6. Lahtinen, E., Ala-Mutka, K., Järvinen, H.-M.: A Study of the Difficulties of Novice Programmers. In: Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education. pp. 14–18. ACM, New York, NY, USA (2005).
7. Kohn, T.: Teaching Python Programming to Novices: Addressing Misconceptions and Creating a Development Environment, https://www.research-collection.ethz.ch/handle/20.500.11850/129666, (2017).
8. Plüss, A., Arnold, J., Kohn, T.: Concepts de programmation avec Python et l'environnement de programmation TigerJython, http://www.tigerjython.fr.
9. Plüss, A., Arnold, J., Kohn, T.: Programmierkonzepte mit Python und der Lernumgebung TigerJython, http://www.tigerjython.ch.
10. Thobois-Jacob, L., Christoffel, E., Marquet, P.: L'adhésion des étudiants à la classe inversée : une approche par le style d'apprentissage. 24, (2018).
11. Learnflow, http://learnflow.ch/.
12. Create online courses the better way., https://www.pathwright.com/.