Une approche de la notion de récursivité à l'aide d'outils de visualisation graphique

Charles Poulmaire & Pascal Remy

Ministère de l'Education Nationale, France

Numéro thématique 4 / 2024

RÉSUMÉ *L'introduction de l'Informatique dans les programmes de l'Enseignement Secondaire en France date de 2012 avec l'apparition de l'option ISN en Terminale. A l'issue de la réforme du lycée général en 2019, celle-ci est devenue une discipline à part entière dont l'enseignement s'étale sur les deux années du cycle terminal (classes de Première et Terminale). L'un des points clés de cet enseignement est la récursivité, introduite en classe de Terminale, après seulement un an de formation initiale. Ce point est souvent difficile à aborder avec les élèves, en raison de l'abstraction sous-jacente à cette notion qui peut entraîner de nombreuses difficultés de compréhension. Dans cet atelier, nous proposons, sous la forme de notebooks Jupyter, deux ressources permettant une meilleure appréhension par les élèves des notions de base de la récursivité (appels récursifs, cas de base et piles d'exécution) en nous appuyant sur une stratégie liée à la manipulation de trois outils de visualisation graphiques : Python Tutor, Code Puzzle et Recursion Visualizer.*

MOTS-CLÉS *• enseignement de l'informatique • fonction • récursivité • appels récursifs • cas de base • piles d'exécution • arbre d'exécution • visualisation graphique*

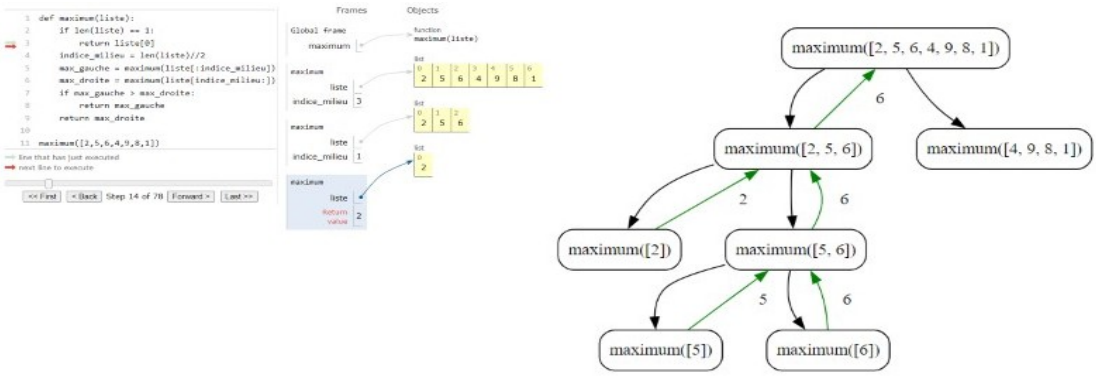

Figure 1 : Python Tutor et RecursionVisualizer

Objectif de l'atelier

Cet atelier a pour but de présenter aux collègues des ressources permettant d'introduire la notion de récursivité en classe de Terminale NSI[1]. Nous axons en particulier notre présentation

1 https://eduscol.education.fr/2068/programmes-et-ressources-en-numerique-et-sciences-informatiques-voie-g

sur l'utilisation d'outils numériques libres de visualisation graphique favorisant une bonne compréhension des concepts de la récursivité :

1. Définir la récursivité comme une méthode où une fonction se fait appel à elle-même pour résoudre un problème.
2. Comprendre comment les appels récursifs se produisent en analysant les étapes d'exécution et en identifiant les conditions d'arrêt (ou cas de base) qui permettent de terminer le processus récursif.
3. Reconnaître des situations où la récursivité est induite naturellement par le problème étudié, tels que les calculs factoriels, les suites de Fibonacci, les fractales, les arbres, etc.
4. Écrire des fonctions récursives en utilisant les concepts de base tels que l'appel récursif, la condition de base et la gestion des paramètres.
5. Identifier et résoudre les erreurs courantes liées à la récursivité, telles que les appels récursifs infinis, les appels récursifs mal définis ou les cas de base incorrects.

Nous présenterons également la manière dont les *notebooks* ont été pensés en vue de gérer l'autonomie des élèves et l'interactivité entre l'enseignant et eux afin de *permettre à tous les élèves d'apprendre à apprendre, seuls ou collectivement, en classe ou en dehors*[2].

Description des ressources

Motivation. L'enseignement classique *papier-crayon* de la récursivité conduisait à ce que la plupart des élèves n'arrivait pas à en appréhender convenablement l'ensemble des concepts fondamentaux. Pour dépasser les limites de cet enseignement, il semble naturel de se tourner vers des outils numériques utilisés à bon escient et de manière réfléchie.

Prérequis. Travail préalable sur la notion de fonctions et une première utilisation des outils numériques de visualisation graphique.

Les outils utilisés

Capytale : cette plateforme[3] fournit, dans un cadre institutionnel, un environnement de travail standardisé conçu pour l'enseignement scolaire. C'est un service accessible sans installation avec un simple navigateur web qui possède une bibliothèque d'activités pédagogiques partagées entre enseignants. Parmi toutes les activités proposées par cette plateforme, nous avons choisi d'utiliser des *notebooks jupyter* qui permettent de combiner, dans un même fichier, du code, des commentaires, des fichiers multimédias et des visualisations sous forme de documents interactifs.

RecursionVisualizer : cet outil en ligne permet de visualiser pas à pas l'arbre des appels effectués lors de l'exécution d'un code *Python*.

Python Tutor : cet outil en ligne permet de visualiser l'exécution pas à pas d'un code *Python* et affiche les différents contextes d'exécution.

2 Référence au Socle commun de Septembre 2016

3 Une alternative possible à cette plateforme est le site basthon.fr, ou des environnements de développement comme EduPyter ou EduPython, par exemple, qui fournissent des serveurs Jupyter intégrés.

CodePuzzle : cet outil permet de créer des problèmes de Parsons qui sont une forme d'évaluation objective dans laquelle les répondants sont invités à choisir parmi une sélection de fragments de code, dont certains sous-ensembles constituent la solution du problème.

Choix et description des exercices

Notebook 1 : ce *notebook* permet d'aborder, à l'aide d'exercices progressifs s'inscrivant dans le temps, les concepts fondamentaux de la récursivité.

1. On demande tout d'abord aux élèves de compléter un programme implémentant le calcul de la factorielle d'un entier, la formule de récurrence étant donnée; puis de décrire à la main les différents appels récursifs pour le calcul de `4!`. L'utilisation de *RecursionVisualizer* permet de se corriger. L'idée est ici de faire réfléchir les élèves sur la notion d'appels récursifs et de cas de base. On notera, par ailleurs, que l'on demande aux élèves de **commencer** par écrire sur papier, avant de passer à l'évaluation via l'application *RecursionVisualizer.* Ceci a la visée pédagogique suivante : forcer les élèves à rentrer dans le code et à voir comment fonctionnent les différents appels récursifs entre eux, ce qui apporte une plus-value non négligeable en terme de compréhension. En effet, nous avons pu tester avec une autre classe l'approche *évaluation puis écriture à la main*, et il s'est avéré au final que ces élèves avaient moins bien compris la notion d'appels récursifs, ceux-ci se contentant juste d'observer le schéma qui se construisait dans *RecursionVisualizer* sans faire le lien avec l'exécution du code.
2. On donne une fonction `mystère`. A l'aide de *RecursionVisualizer* et *Python Tutor*, on demande de répondre à des questions permettant aux élèves de bien comprendre l'intérêt du cas de base. On introduit également la notion de *fonction récursive bien formée*.
3. On donne la définition d'une suite avec sa relation de récurrence et son premier terme. On demande aux élèves de compléter un code permettant de calculer de façon itérative le terme de rang `n` de cette suite. Des tests sont proposés pour vérifier la réponse. Dans un second temps, on demande aux élèves d'écrire une version récursive du code précédent à l'aide de *CodePuzzle* afin de présenter deux approches différentes de ce problème.En particulier on teste ici leurs capacités à choisir les blocs justes parmi tous les blocs proposés, certains correspondant à des erreurs classiques.
4. On donne une fonction récursive permettant de déterminer le maximum dans une liste. L'utilisation des outils numériques permet ici de gérer les multiples appels et de déterminer dans quel ordre sont obtenus les différents calculs intermédiaires de maximum. Ceci permet de se focaliser seulement sur les appels, ce qui apporte aux élèves une meilleure compréhension de la notion de pile d'appels.

Notebook 2 : le but de ce second *notebook* est de réinvestir et approfondir les différentes notions de base de la récursivité introduites dans le notebook précédent. La philosophie générale reste la même : exercices sous la forme de questions et utilisation des outils numériques précédents.

Ceci permet au professeur·e de mettre en place une stratégie d'individualisation.

Les différents exercices proposés ici, étant de nature plus complexes que ceux du *notebook* précédent, il est préférable de les inscrire dans la durée. Ceci aura aussi l'avantage de permettre une meilleure mémorisation des notions en y revenant régulièrement.

Expérimentations réalisées ou envisageables (public, descriptif ressource)

Ces deux ressources ont été expérimentées dans trois classes de Terminale NSI. A l'issue de cette expérimentation, nous avons pu en tirer les conclusions suivantes sur les plus-values pédagogiques (qu'elles soient du point de vue des enseignants ou des élèves), ainsi que sur les freins et les leviers inhérents à ce type d'activités :

Les plus-values pédagogiques

- autonomie des élèves ;
- différenciation pédagogique par une individualisation des remarques et conseils ;
- gestion du groupe.

Les freins

- les contraintes de temps des exercices des activités peuvent limiter l'impact des objectifs fixés ;
- les activités nécessitent des ressources spécifiques, telles que du matériel, une connexion internet et un accès aux différentes plateformes en ligne ;
- les activités peuvent être plus difficiles à gérer dans de grandes classes où le nombre d'élèves est élevé. Il peut être plus complexe de fournir un suivi individuel et d'assurer la participation active de tous les élèves ;
- certains élèves peuvent être réticents à sortir de leur zone de confort et à participer activement à ces activités dont l'approche est plus expérimentale et participative, préférant des méthodes d'enseignement plus traditionnelles.

Les leviers

- engagement actif : ces activités favorisent l'engagement actif des élèves dans le processus d'apprentissage. Ils sont encouragés à participer activement, à réfléchir, à résoudre des problèmes et à interagir avec les autres élèves, ce qui renforce leur motivation et leur implication ;
- apprentissage expérientiel : les activités pédagogiques présentées dans ces deux *notebooks* offrent des opportunités concrètes aux apprenants d'expérimenter, de manipuler, de mettre en pratique leurs connaissances théoriques et de développer des compétences pratiques. Cela favorise une meilleure compréhension des concepts et permet aux apprenants de se familiariser avec les situations réelles ;
- collaboration et travail d'équipe : ces activités pédagogiques favorisent la collaboration et le travail d'équipe entre les apprenants. Ils sont encouragés à partager leurs idées, à résoudre des problèmes ensemble, à échanger des connaissances et à apprendre les uns des autres. Cela développe leurs compétences en communication, en collaboration et en pensée critique.

Liens avec la recherche

J. Arsac, *Itération et récursivité*, Troisième Rencontre Francophone de Didactique de l'Informatique, Sion, Suisse, 1992

A. Busser, F. Jean-Albert et S.Hoarau, *Récursivité et difficultés,* IREMI de La Réunion, https://iremi.univ-reunion.fr/spip.php?article1090

A.Esbelin et M.More *Fonctions et récursivité STS SIO,* IREM Clermont-Ferrand, http://www.irem.univ-bpclermont.fr/IMG/pdf/2FonctionsRecursivite-2.pdf

P. Lac et M.More, *Récursivité,* IREM Clermont-Ferrand, http://www.irem.univ-bpclermont.-fr/IMG/pdf/recursivite.pdf

N. Leon et S. Modeste, *Récurrence et récusivité à l'interface des mathématiques et de l'informatique*, IREM, 119, 2020

Liens vers les ressources et point de contact

- Dépôt des ressources
- Capytale
- Python Tutor
- RecursionVisualizer
- Code Puzzle

N'hésitez pas à contacter Charles Poulmaire et Pascal Rémy pour plus d'information.

Références

Balabonski T., Conchon S., Filliâtre J-C., & Nguyen K. (2020) *Spécialité Numérique et sciences informatiques : 24 leçons avec exercices corrigés - Terminale - Nouveaux programmes.* Ellipses.

Cormen T. H., Leiserson, C., Rivest, R., Stein, C. (2010) *Algorithmique*, 3ème édition, Sciences Sup, Dunod.

Dowek, G. (2011). *Introduction à la science informatique*, RPA, Scérén, CNDP-CRDP