Nombre de participant·e·s maximal pour l'atelier	16	Nombre de sessions de 45mn que vous pouvez réaliser	2
Matériel nécessaire (vidéo-projection, grande salle, autre ...)	Vidéo-projecteur, connexion Wifi pour les participants		

Modèles et instruments pour l'enseignement et l'apprentissage de la programmation en Python

Sébastien Hoarau et Christophe Declercq

Laboratoire d'Informatique et de Mathématiques, IREMI, Université de La Réunion

RÉSUMÉ *Cet atelier propose de manipuler deux modèles mémoires ou « machines notionnelles », dédiés à l'apprentissage du langage Python : le modèle d'ardoise et le modèle de références. Les participant.e.s seront invité.e.s à utiliser les modèles sur des exemples de tâches de programmation au programme de la spécialité NSI au lycée. Un générateur d'ardoise leur permettra de créer des activités débranchées d'évaluation de programmes simples, dans le cadre du modèle d'ardoise. Ensuite, les participant.e.s seront invité.e.s à utiliser le modèle de références pour expliquer l'exécution de programmes utilisant des mutables ou des objets en Python.*

MOTS-CLÉS • *NSI, programmation, Python, machine notionnelle, modèle mémoire.*

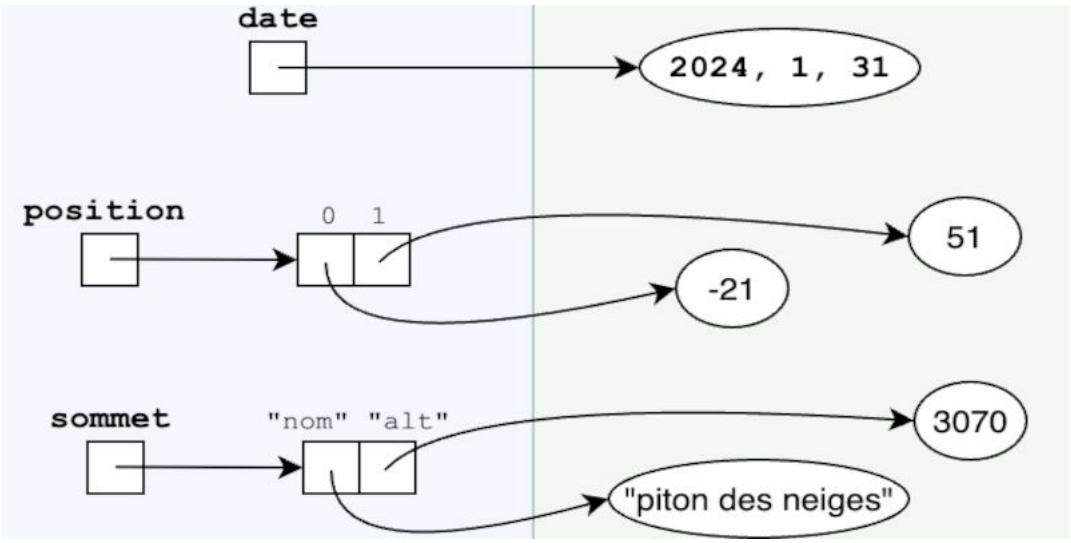

Figure 1: Modèle de références pour quelques structures Python

1. Déroulement de l'atelier

L'atelier est principalement destiné aux enseignant·e·s d'informatique au lycée et à leurs formateurs.

Après la présentation des deux modèles, les participant·e·s seront invité·e·s à utiliser les modèles sur des exemples de tâches de programmation au programme de la spécialité NSI au lycée. Les participant.e.s expérimenteront en particulier les modèles mémoire sur des tâches déjà expérimentées en classe au lycée et jugées problématiques à cause d'une difficulté pour les élèves à évaluer leurs programmes.

Concernant le modèle d'ardoises, les participant·e·s travailleront successivement sur des exemples simples, puis sur des exemples avec appels de fonctions y compris récursives.

Ensuite, les participant·e·s seront invité·e·s à utiliser le modèle de références pour expliquer l'exécution de programmes utilisant des mutables ou des objets en Python.

Une mise en commun permettra de discuter l'intérêt de chaque modèle par rapport aux tâches de programmation étudiées et au niveau de classe concerné.

2. Description des ressources présentées dans l'atelier

Dans le modèle d'ardoise, une ardoise comporte un ensemble de cases - les variables - nommées chacune par un nom. Une ardoise permet de représenter à un instant donné l'état de la machine, déterminé par les valeurs des variables du programme en cours d'exécution (voir figure 2). Ce modèle est suffisant pour représenter l'état d'exécution d'un programme comportant tout type de variable simple (entier, flottant, chaîne ou booléen) ainsi que des tableaux à condition qu'ils soient de taille statique. L'usage en classe de ces ardoises consiste à en distribuer une à chaque élève avec la consigne d'exécuter le programme pas à pas en modifiant au fur et à mesure les valeurs des variables.

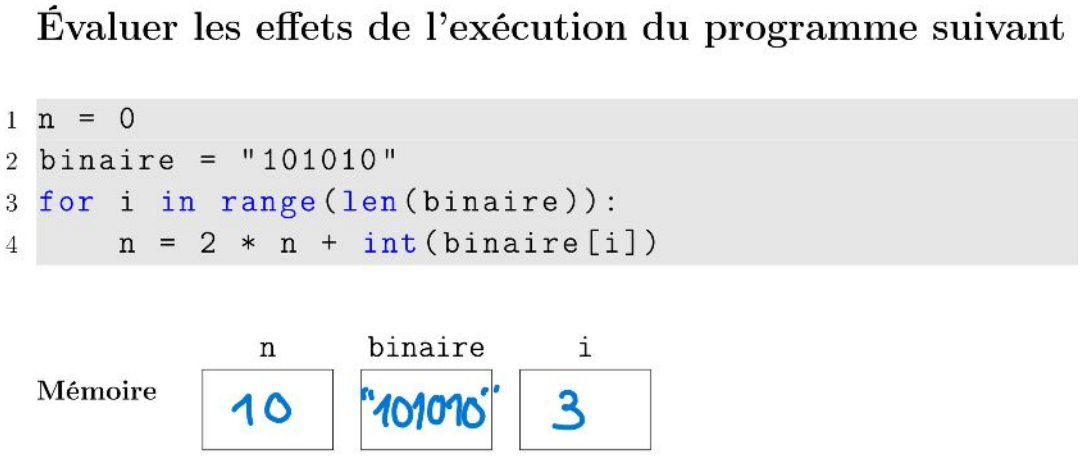

Figure 2: Ardoise pour un programme simple : état en cours d'exécution

Le modèle de multi-ardoises permet d'interpréter des appels de fonctions. Dans ce modèle on utilise une ardoise par fonction, avec pour chacune la visualisation de ses variables locales et paramètres, et une ardoise pour le programme principal. L'activité en classe peut alors être organisée en groupes en distribuant une ardoise différente à chaque élève d'un groupe.

Une variante utile du modèle d'ardoise est le tableau d'évolution des variables qui permet de représenter l'historique des valeurs des variables. Au lieu d'effacer une valeur pour la remplacer par la nouvelle, l'élève doit alors inscrire la nouvelle valeur sur une nouvelle ligne du tableau.

Le tableau d'évolution a sur l'ardoise l'avantage de pouvoir être vérifié plus facilement par l'enseignant et de constituer une trace écrite du travail d'évaluation d'un programme.

Un générateur d'ardoises et tableaux d'évolution

Pour institutionnaliser la pratique d'évaluation de programmes Python à l'aide d'ardoises ou de tableaux d'évolution, avec une présentation homogène, on propose un générateur permettant à l'enseignant de réaliser simplement ses ardoises ou tableaux d'évolution, accessible en ligne à l'adresse :

https://iremi974.gitlab.io/python-de-la-fournaise/ardoise.html

Un formulaire permet de saisir le programme à évaluer et de choisir de générer soit une ardoise pour avoir seulement une case par variable, soit un tableau pour pouvoir écrire l'historique des valeurs des variables. Le formulaire permet aussi d'indiquer si le programme comporte des entrées et/ou des sorties, et alors d'activer le clavier et/ou l'écran et de choisir leurs nombres de lignes. Il permet enfin de préciser les noms des variables et pour un tableau le nombre de lignes à afficher.

Un lien permet alors de télécharger un fichier latex ardoise.tex, prêt à imprimer, plastifier puis à faire remplir par l'élève pour apprendre à évaluer les effets de l'exécution d'un programme. Les exemples ci-dessus ont été réalisés grâce à ce générateur.

Un deuxième générateur, spécialisé pour les fonctions, permet de saisir le texte d'une fonction, ses paramètres et variables locales et de générer l'ardoise ou le tableau d'évolution de la fonction.

Présentation du modèle de références

Le modèle de références repose sur l'existence d'une mémoire dans laquelle sont stockées à la fois les variables et les valeurs. Comme pour le modèle d'ardoise, une variable est une case avec un nom. La différence se situe au niveau du contenu : dans le modèle de références la case contient une référence (flèche).

Les objets de type simple (`int`, `float`, `bool` et `str`) ainsi que les tuple sont modélisés par leurs valeurs. Les structures mutables `que` sont les list, dict et autres objets au sens de la POO sont des collections de cases étiquetées par des entiers dans le cas des list et par des noms dans le cas des autres structures.

La figure 1 montre le modèle sur différents objets de Python.

```
date = (2024, 1, 31)
position = [-21, 51]
sommet = {"alt": 3070, "nom": "piton des neiges"}
```

Toute référence est nommable et modifiable. Par exemple `position[0]` est la référence vers la valeur `-21` et `sommet` est une référence vers l'enregistrement.

La sémantique de l'affectation (`variable = expression`) consiste à évaluer l'expression, à créer la valeur obtenue en mémoire, puis à créer ou modifier la référence associant la variable à cette valeur. Lorsque l'expression est réduite à une variable, la référence se fait sur la valeur déjà référencée, il s'agit du concept d'aliasing.

```
x = 10
y = 10
alias = x

t = [10, 20, 30]
t2 = t
t2[2] = -1
```

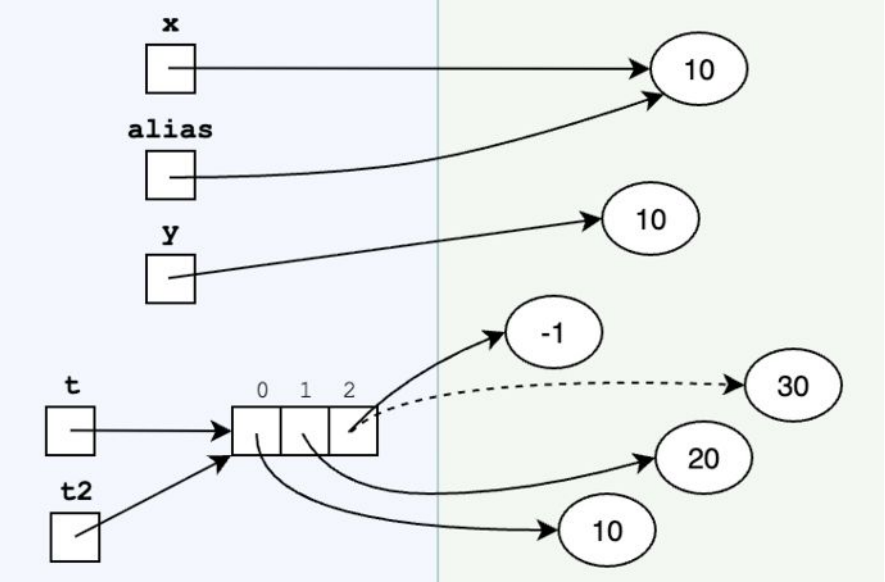

Figure 3: Modèle de références pour l'affectation

La figure 3 donne une visualisation du modèle pour une séquence d'affectations. Les trois premières concernent des entiers ; la variable `alias` référence la même valeur que la variable `x`. La variable `y` ne référence pas la même valeur, même si mathématiquement les entiers sont effectivement égaux.

L'affectation de la variable `t` crée une référence vers le tableau `[10, 20, 30]`, celle de `t2` crée une référence vers ce même tableau. La dernière affectation modifie la référence de la dernière case du tableau, l'ancienne référence est visualisée en pointillé. La modification est visible depuis les deux références `t` et `t2`.

Un instrument en ligne permet de générer un fichier à imprimer et plastifier avec un programme et un espace mémoire à compléter :

`https://iremi974.gitlab.io/python-de-la-fournaise/references.html`

3. Expérimentations réalisées

Les outils présentés ont été expérimentés en formation d'enseignant·e·s et en classe de lycée. Les expérimentations sont relatées dans la communication à Didapro 2026 (Hoarau, Declercq & Ethève, 2026).

4. Liens avec la recherche

L'explication détaillée des modèles et les résultats des expérimentations ont été présentés à Didapro 2026 (Hoarau, Declercq & Ethève, 2026). Les modèles mémoire sont des machines notionnelles au sens de Du Boulay (Du Boulay, 1981).

5. Liens vers les ressources et point de contact

- L'ensemble des ressources sur les modèles mémoire, incluant les protocoles d'expérimentation et divers exemples utilisés en formation, est disponible à l'adresse : `https://iremi.univ-reunion.fr/?p=1670`

- Point de contact pour plus de renseignement : sebastien.hoarau@univ-reunion.fr et christophe.declercq@univ-reunion.fr

6. Références

Du Boulay, B., O'Shea, T., Monk, J. : The black box inside the glass box : presenting computing concepts to novices. International Journal of man-machine studies 14(3), 237–249 (1981)

Hoarau, S., Declercq, C. : De l'intérêt de machines notionnelles adaptées à l'apprentissage de la programmation en Python. In : Broisin, J., Declercq, C., Fluckiger, C., Jolivet, S., Parmentier, Y., Peter, Y., Secq, Y. (eds.) Actes de l'Atelier APIMU. pp. 39–49. Villeneuve d'Ascq, France (2025), https ://hal.science/hal-05084735

Hoarau, S., Declercq, C., Ethève A. : Modèles, langages et instruments pour l'enseignement et l'apprentissage de la programmation en Python, Expérimentations en formation d'enseignants et au lycée, Didapro 11, Grenoble, Mars 2026.