Modèles de mémoire pour l'enseignement de la programmation en Python

Sébastien Hoarau, Christophe Declercq

4 décembre 2024

Plan de la présentation

- Première expérimentation
- Deuxième expérimentation
- Retour sur première expérimentation : vers un modèle pour les débutants
- Retour sur deuxième expérimentation : vers un modèle pour initiés
- Conclusions et perspectives

Première expérimentation

Etude des modèles de représentation

Objectif : recenser les pratiques enseignantes pour expliquer le comportement d'un programme à des élèves de première NSI.

Consigne : dessiner le tableau de la classe après explication aux élèves de l'exécution du programme suivant :

```
def etage(n):
    t = 0
    for i in range(1, n+1):
        t = t + i
    return (t)
```

```
taille = 3
total = 0
for i in range(1, taille+1):
    total = total + etage(i)
```

Figure 1: Consigne première expérimentation

Étude des modèles de représentation, partie 2

- **Objectif** : recenser les pratiques enseignantes pour expliquer le comportement d'un programme à des élèves de Terminale NSI.
- **Consigne** : En vous aidant des points de contrôle, dessiner les différents tableaux (comme si vous n'effaciez pas le tableau) de votre explication à un élève qui ne comprend pas pourquoi son programme P1 ne fonctionne pas alors que le programme P2 de son amie fonctionne bien.

```
#P1
def double(valeurs):
    # point de contrôle 1
    for elt in valeurs:
        # point de contrôle 2A
        elt = 2 * elt
    # point de contrôle 2B

t = [10, 30, 50]
double(t)
#point de contrôle 3
```

```
#P2
def double(valeurs):
    # point de contrôle 1
    for i in range(len(valeurs)):
        # point de contrôle 2A
        valeurs[i] = 2 * valeurs[i]
    # point de contrôle 2B

t = [10, 30, 50]
double(t)
#point de contrôle 3
```

Figure 2: Consigne deuxième expérimentation

Réflexion sur les modèles de mémoire

Cadre théorique

- L'informatique repose sur 4 piliers : des algorithmes, des langages, des machines et des informations [G. Dowek].
- On s'intéresse à l'adéquation langage <-> machine et plus précisément à la question : quelle image de la machine pour les élèves, selon le niveau de langage à apprendre ?
- Voir la notion de "machines notionnelles" [Du Boulay]

Contextes

- Enseignement de Python en seconde, classe de mathématiques et de SNT (*)
- Enseignement de Python en première NSI (*)
- Enseignement de Python en terminale NSI ou en premier cycle à l'université

Objectifs

Identifier des couplages cohérents

- un sous-ensemble du langage Python
- un modèle de machine
- la “sémantique” (sens) d’une construction du langage doit pouvoir être expliquée sur la machine

Proposer des environnements d’apprentissage cohérents

- Des outils pour l’enseignant : représentations de la machine (partie de) au tableau
- Des instruments pour l’élève : IDE (environnement de programmation)

Retour sur première expérimentation

Un modèle pour les débutants

Un sous-ensemble de Python très réduit

- Memento 2nde
- expressions sur types simples (int, str, float, bool)
- affectations et instructions composées (séquence, if, while, for)
- fonctions (définition et appel)

Une machine à mémoire

- la **mémoire** associe à des noms, des valeurs
- le sens d'une expression, c'est son **évaluation**, connaissant l'état de la mémoire
- le sens d'une instruction, c'est comment son **exécution** modifie la mémoire
- le sens d'une **affectation**, c'est évaluer l'expression en partie droite, puis associer cette valeur au nom figurant en partie gauche
- le sens d'un **appel de fonction**, c'est déléguer à une autre machine, l'exécution du code avec pour mémoire l'affectation des paramètres aux valeurs données à l'appel

Représentation des variables

- une variable, c'est une **case mémoire** qui a un nom
- métaphore de la variable : l'ardoise effaçable
- l'ensemble des variables représente l'**état** de la machine à un instant donné

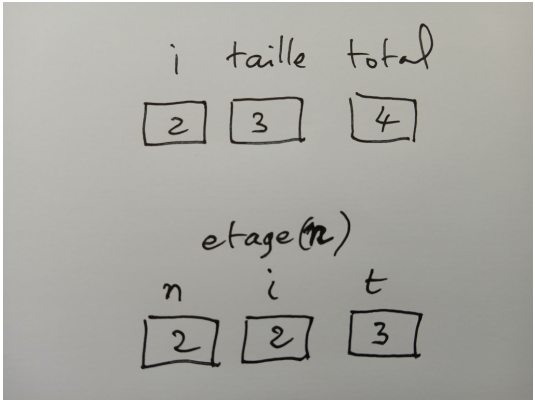

Figure 3: Etat de la mémoire

Représentation de l'historique

- tableau d'évolution : une colonne par variable
- le tableau représente les états observés de la machine au cours de l'exécution

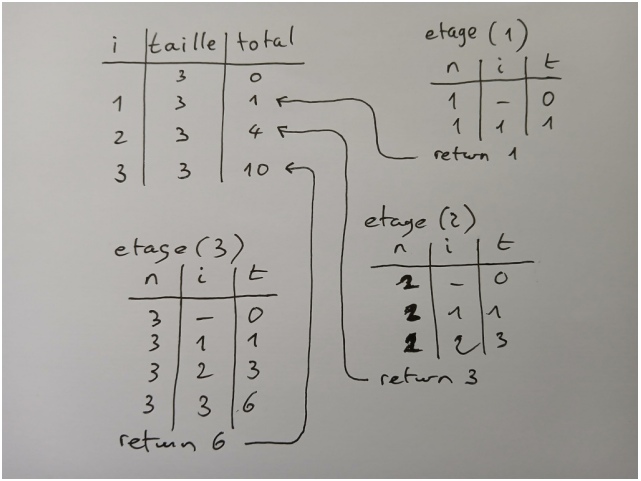

Figure 4: Historique des états mémoire

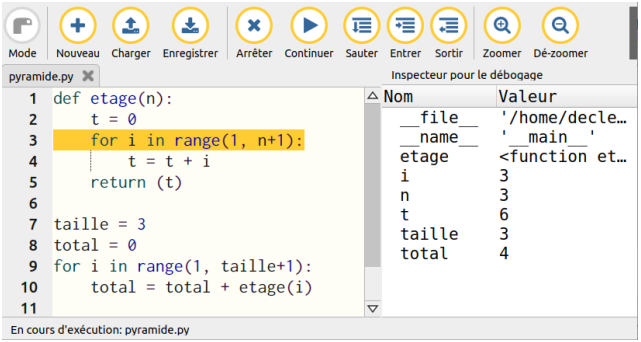

Figure 5: Exécution en mode pas à pas avec Mu

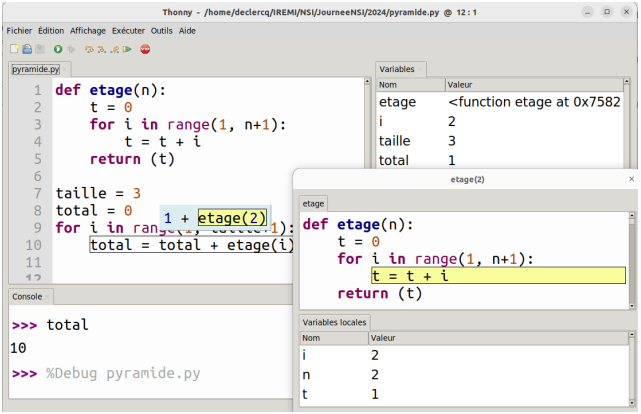

Figure 6: Exécution en mode pas à pas avec Thonny

Extension possible aux tableaux et structures de données

Un sous-ensemble de Python augmenté

- Memento 1ere
- instructions de création
- expression d'accès
- instructions de modification pour les tableaux et les enregistrements

Une machine à mémoire "structurée"

- des cases contigues nommées par un indice (tableaux, tuples) ou un nom (enregistrements)
- change le format des noms de variables, la mémoire reste une association : noms -> valeurs.

Restrictions au niveau langage

- pas d'affectation globale de tableau (après la construction initiale)
- pas de tableau en **paramètre** d'une fonction

Retour sur deuxième expérimentation

Vers un modèle pour initiés

ce qui change...

- on appelle **référence** l'association variable $\rightarrow$ valeur
- lorsqu'en partie droite d'une affectation on a une unique variable on parle d'*aliasing* (un objet référencé par plusieurs variables)
- une instruction qui modifie la mémoire, crée ou change une référence
- le sens d'un **appel de fonction**, c'est d'effectuer les affectations entre des *variables paramètres* créées temporairement pour l'exécution de la fonction et les valeurs d'appel, à la fin, une variable particulière est créée et référence la valeur de retour
- un tableau est un **ensemble de cases mémoire contigües** contenant des références

Conclusions et perspectives