Python 15 - Portée des variables locales

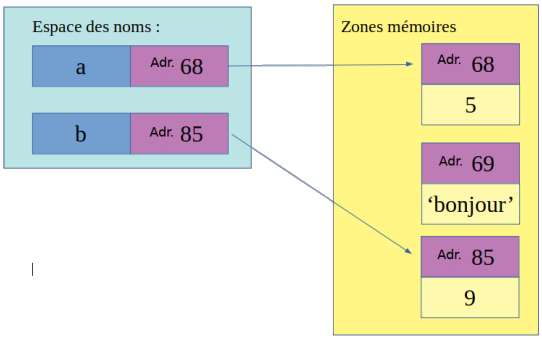
I – Identifiants des variables

1.1 Espace des noms

Les variables Python ne peuvent pas être vues comme de simples boîtes.

L'interpréteur dispose pour le **programme principal** et pour **chaque lancement de fonction** des tableaux qu'on nommera **ESPACES DES NOMS**.

Ces tableaux associent un nom de variable à un identifiant permettant de localiser une zone mémoire.

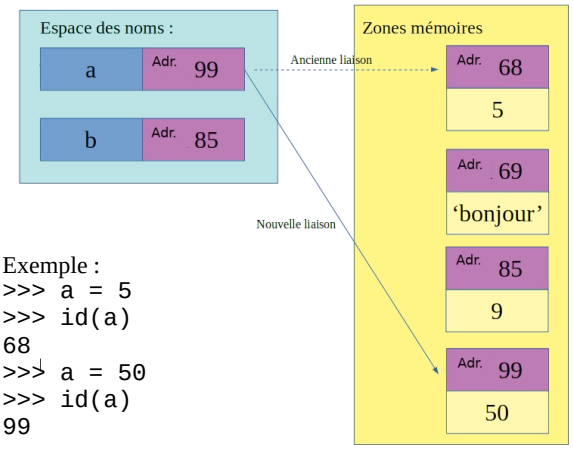


1.2 Fonction native id()

```
>>> a = 5
>>> id(a)
68          # Valeur imaginaire ci-dessus
```

1.3 Affectation

Une affectation correspond donc **en Python** à la création ou la modifcation d'une des lignes dans l'un des tableaux.



Exemple :

```
>>> a = 5
>>> id(a)
68
>>> a = 50
>>> id(a)
99
```

ATTENTION :

- à chaque nouvelle affectation l'id de la variable change
- **b = a** ne fait que créer une variable **b** qui fait référence à la **même zone mémoire** que **a**. Ce n'est pas une copie mais un alias.

II – Espace des noms : un par fonction

```
1    def gentil_prof(note_gp):
2        if note_gp < 10:
3            note_gp = note_gp + 4
4        return note_gp
5
6    def mechant_prof(note_mp):
7        if note_mp > 10:
8            note_mp = 10 + (note_mp-10) // 2
9        else:
10           note_mp = note_mp // 2
11       return note_mp
12
13   note_pp = 8
14   y = gentil_prof(note_pp)
15   z = mechant_prof(note_pp)
```

Sur cet exemple, j'ai rajouté des indications derrière les variables notées **note** mais elles ne sont pas nécessaires. On peut noter juste **note** à chaque fois sans que l'interpréteur Python ne se trompe.

Il suffit de se souvenir que le programme principal ainsi que chaque lancement de fonction a son propre espace des noms.

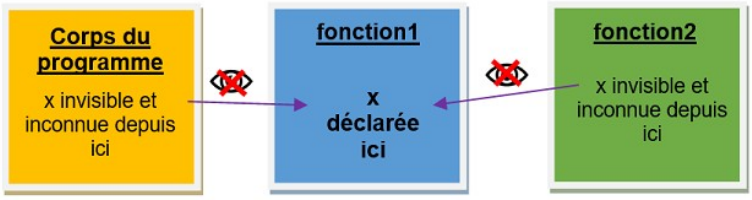
Ainsi la variable **note** des lignes 1, 2, 3 et 4 est la variable **note** de la fonction **gentil_prof.**

Et elle n'a rien à voir avec le **note** des lignes 13, 14, 15 qui correspond à la variable **note** du programme principal.

III – Portée des variables locales

Définition : une **variable locale** Python est une variable déclarée à l'intérieur d'une fonction.

Une variable locale est **lisible et modifiable** uniquement à l'**intérieur de la fonction** où elle est déclarée.



Une variable locale est détruite une fois qu'on sort de la fonction avec return : seule la réponse va rester en mémoire SI on la stocke.