♦ TP3 – Aliasing, portée de variable et effet de bord

Exercice 1. Déterminer le ou les affichages obtenu(s) lors de l'exécution de chacun des programmes suivants.

```
# Programme 1
a = 3
b = 3
c = a
a = 5
print(a,b,c)
```

```
# Programme 2
L = [1, 2, 3]
M = [1, 2, 3]
L[0] = 'a'
print(L)
print(M)
```

```
# Programme 3
L = [4, 'c', True]
N = L
N[1] = 3.14
print(L)
print(N)
```

```
# Programme 4
M = [0, 0, 0]
N = 4*[M]
print(M, N)
N[1][0] = 1
print(M, N)
N[2] = [2, 3, 4]
print(M, N)
N[3][2] = 11
print(M, N)
N[2][1] = 'n'
print(M, N)
```

```
# Programme 5
N = 4*[3*[0]]
print(N)
N[1][0] = 1
print(N)
N[2] = [2, 3, 4]
print(N)
N[3][2] = 11
print(N)
N[2][1]= 'n'
print(N)
```

```
# Programme 6
N = [[0 for i in
   range(3)] for j
   in range(4)]
print(N)
N[1][0] = 1
print(N)
N[2] = [2, 3, 4]
print(N)
N[3][2] = 11
print(N)
N[2][1] = 'n'
print(N)
```

Solution.

- Pour le programme1, on obtient `5, 3, 3`.
- Pour le programme2, on obtient `['a', 2, 3]` et `[1, 2, 3]`.
- Pour le programme3, il y a un effet d'aliasing et on obtient à l'affichage `[4, 3.14, True]` et `[4, 3.14, True]`.
- Pour le programme4, il y a un effet d'aliasing dans la création de `N`. On obtient à l'affichage successivement :

```
[0, 0, 0], [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
[1, 0, 0], [[1, 0, 0], [1, 0, 0], [1, 0, 0], [1, 0, 0]]
[1, 0, 0], [[1, 0, 0], [1, 0, 0], [2, 3, 4], [1, 0, 0]]
[1, 0, 11], [[1, 0, 11], [1, 0, 11], [2, 3, 4], [1, 0, 11]]
[1, 0, 11], [[1, 0, 11], [1, 0, 11], [2, 'n', 4], [1, 0, 11]]
```

- Pour le programme 5, c'est la même chose que pour le programme 4 et on obtient :

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
[[1, 0, 0], [1, 0, 0], [1, 0, 0], [1, 0, 0]]
[[1, 0, 0], [1, 0, 0], [2, 3, 4], [1, 0, 0]]
[[1, 0, 11], [1, 0, 11], [2, 3, 4], [1, 0, 11]]
[[1, 0, 11], [1, 0, 11], [2, 'n', 4], [1, 0, 11]]
```

- Pour le programme 6, il n'y a pas d'effet d'aliasing et on obtient :

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
[[0, 0, 0], [1, 0, 0], [0, 0, 0], [0, 0, 0]]
[[0, 0, 0], [1, 0, 0], [2, 3, 4], [0, 0, 0]]
[[0, 0, 0], [1, 0, 0], [2, 3, 4], [0, 0, 11]]
[[0, 0, 0], [1, 0, 0], [2, 'n', 4], [0, 0, 11]]
```

Exercice 2. Déterminer la valeur renvoyée par la fonction ainsi que la valeur de la variable `a` à la fin de l'exécution de chacun des programmes suivants.

```
# Programme 1
a = 2
def carre(x):
  a = x**2
  return a
```

```
# Programme 2
a = 2
def puissance_a(x):
  b = x**a
  return b
```

```
# Programme 3
a = 2
def puissance_a(x):
  a = x**a
  return a
```

Solution.

Les programmes 1 et 2 renvoient le la carré de la valeur de x passée en argument et, à la fin de l'exécution, `a` vaut 2.

Le programme 3 renvoie une erreur car l'affectation `a = x**a` crée une variable locale à la fonction qui n'est pas encore définie et qui empêche le calcul de `x**a`.

Exercice 3. Réécrire le programme 4 de l'exercice 1 afin d'obtenir les mêmes affichages pour la liste `N` mais sans modifier la liste `M` initiale. (Le programme doit fonctionner quelle que soit la liste `M` initiale.)

Solution.

```
import copy

M = [0, 0, 0]
P = copy.deepcopy(M)
N = 4*[P]
print(M, N)
N[1][0] = 1
print(M, N)
N[2] = [2, 3, 4]
print(M, N)
N[3][2] = 11
print(M, N)
N[2][1] = 'n'
print(M, N)
```

Exercice 4. On reprend le programme 3 de l'exercice 1.

1. On remplace, dans `L`, `'c'` par la liste `[1,2,3]` et on remplace `N[1] = 3.14` par `N[1][1] = 3.14`. Qu'obtient-on à l'affichage ?
2. Réécrire le programme ainsi modifié pour qu'il affiche la même valeur de `N` mais sans modifier la liste `L`.

Solution.

1. Le programme devient

```
L = [4, [1,2,3], True]
N = L
N[1][1] = 3.14
print(L)
print(N)
```

Il y a un effet d'aliasing donc on obtient à l'affichage `[4, [1,3.14,3], True]` et `[4, [1,3.14,3], True]`.

2.

```
import copy

L = [4, [1,2,3], True]
N = copy.deepcopy(L)
N[1][1] = 3.14
print(L)
print(N)
```

Exercice 5. La fonction de l'exercice 26 du TP2 a-t-elle un effet de bord ? Si oui, écrire une fonction `valeur_absolue_liste_2` effectuant le même travail mais sans effet de bord.

Solution. La fonction `??` modifie la liste passée en argument donc il y a un effet de bord. Comme la liste passée en argument contient des réels (et ne contient donc pas de listes), on peut se contenter d'une copie de surface.

```
def valeur_absolue_2(L):
  M = [e for e in L]
  for i in range(len(M)):
    if M[i]<0:
      M[i]=-M[i]
  return M
```

Exercice 6. Écrire, sans effet de bord,

1. une fonction `ajout` qui prend en argument une liste `L` et un objet `o` et qui ajoute `o` à la liste `L` en dernière position.
2. une fonction `suppr` qui prend en argument une liste `L` et un objet `o` et qui supprime toutes les occurrences de `o` dans la liste `L`.

Solution.

1.

```
def ajout(L,o):
  M = [e for e in L]
  M.append(o)
  return M
```

2.

```
def suppr(L,o):
  M = []
  for e in L:
    if e != o:
      M.append(e)
  return M
```