Chapitre 10 : Ecriture d'un programme

Un programme est composé d'instructions et de commentaires.

I- Retour sur l'instruction affectation :

a) L'affectation

L'affectation est un exemple d'instruction. Une affectation est utilisée pour lier un nom à une valeur ou pour modifier un élément d'un objet mutable comme une liste. L'affectation est représentée par le signe =.

Lors d'une opération d'affectation, il se produit 3 évènements :

$var = 2$	
1 : On prend un objet (ici un entier) dans l'espace des objets	2 : On crée une variable var (espace des variables)
3 : On affecte var à cet objet via le signe = Plus exactement on dit que la variable référence l'objet (c'est une référence à une adresse de la mémoire).	
$var = var + 2$ Le typage dynamique est possible avec une nouvelle affectation la variable ne référence plus à la même adresse.	

b) Effets de bords sur les listes

- Effet de bord lors de l'exécution d'une fonction

```
a=2

L=[1,2,3]

def f(x):

    a=4

    L.append(a*x)

print(a)#a=2
f(1)
print(L)#L=[1,2,3,4]  (effet de bord)
print(a)#a=2  (pas d'effet de bord)
```

En dehors du corps de la fonction, c'est l'espace des variables globales. Si ces variables sont associées à des entiers, floats, des chaînes ou des tuples alors l'exécution d'une fonction ne peut pas les modifier (on parle de mutable ou muable). En revanche, une liste est mutable : l'exécution d'une fonction peut la modifier : on parle d'effet de bord

Les arguments et les variables définies dans le corps de la fonction sont des variables locales. Elles sont modifiables et accessibles qu'au moment de l'exécution de la fonction

- Effet de bord lors d'une copie par référence

```
L1=[1,2,3,4]
L2=L1
L2.append(5)
print(L1)#L1=[1,2,3,4,5]
```

L2 fait référence à la même adresse que L1. Donc modifier L2, modifieL1

Rq : L3=L1[:] permet une copie (superficielle) évitant l'effet de bord ci-dessus (en faisant une copie de l'adresse mémoire de chaque élément)

II- Commentaires

a) Spécification d'une fonction

Une spécification permet d'informer l'utilisateur de la tâche effectuée par la fonction et même des contraintes imposées sur ses variables. Elle peut être inscrite au début du corps d'une fonction entre triples guillemets :

```
def permutation(liste):
    """liste est de type list la fonction permute le 1e
    et le dernier élt et renvoie une nouvelle liste
    Exemple : permute([1,2,3,4]) renvoie [4,2,3,1]"""
    copie=liste[:]#commentaire : copie superficielle
    copie[0],copie[-1]=copie[-1],copie[0]#permutation
    return copie
```

b) Assertion

Prenons un exemple montrant la nécessité d'introduire des assertions. La fonction ci-dessous est destinée à faire la somme de 2 nombres, cependant, elle s'exécute sans aucun souci si les arguments sont de type str.

```
def calcul(x,y):
    return (x+y)
print(calcul(2,3))#=>5
print(calcul("a","b"))#=>"ab"
```

Si le résultat obtenu doit être utilisé pour un autre calcul, il est capital que ce soit bien un nombre et donc de détecter ce problème de typage. On peut rajouter une instruction qui va arrêter le programme en cas de mauvaise utilisation (assert condition).

```
def calcul(x,y):
    """x et y sont des nombres
    la fonction renvoie la somme de ces deux nombres"""
    assert type(x)==int and type(y)==int#le programme se
poursuit si vrai
    return (x+y)
```

Commenté [AM1]: Une instruction est un morceau de code qui produit un effet sur la machine lorsqu'elle est exécutée
Un expression est la notation d'un calcul à réaliser
Une instruction simple peut s'écrire sur une seule ligne. On peut en écrire plusieurs séparées par ; sur une seule ligne.
Autres exemples d'instructions simples
-Renvoyer le résultat (ou les résultats sous forme d'un tuple) d'une fonction avec *return*
-Arrêter une boucle avec *break*
-Importer des modules avec *import*
Exemple d'instructions composées :
Une instruction composée est une instruction sur une ligne terminée par deux-points suivi d'une ou plusieurs instructions indentées. On les retrouve typiquement dans les fonctions avec le mot clé def, boucles $for, while$ dans les instructions conditionnelles $if, else, elif$.

Commenté [AM4]: On retrouve donc un effet de bord si L1 est une liste de liste. Il reste la possibilité de faire une copie sans aucun effet de bord avec $L_4 = copy.deepcopy(L_1)$

Commenté [AM2]: Ainsi la variable var n'est pas directement associée à la valeur 2. var est associé à une adresse mémoire dans laquelle est stockée la valeur 2

Commenté [AM5]: Préciser les entrées et sortie définit la signature de la fonction

Commenté [AM3]: A noter l'absence d'effet de bord si on crée une nouvelle liste :
a=2
L=[1,2,3]
def f(L,x):
  a=4
  L=L+[a]