Chapitre 9 : Ecriture d'une fonction

Une fonction est composée d'instructions (morceaux de code), de spécifications et de commentaires.

I- Effet de bord :

On dit qu'une fonction a un effet de bord si son exécution modifie une variables globales mutables (typiquement une liste) :

Avec effet de bord	Sans effet de bord

`def ajoute1(liste,x):`
`  liste.append(x)`
`L=[0,1,2]`
`ajoute1(L,3)`
`print(L)#[0,1,2,3]`
	`def ajoute2(liste,x):`
	`  liste=liste+[x]`
	`  return liste`
	`L=[0,1,2]`
	`ajoute2(L,3)`
	`print(L)#[0,1,2]`

`def multiplie1(liste,x):`
`  res=liste#copie par référence`
`  for i in range(len(res)):`
`  res[i]=res[i]*x`
`  return res`
`L=[0,1,2]`
`multiplie1(L,2)`
`print(L)#[0,2,4]effet de bord`
	`def multiplie2(liste,x):`
	`  res=liste[:]#copie superficielle`
	`  for i in range(len(res)):`
	`  res[i]=res[i]*x`
	`  return res`
	`L=[0,1,2]`
	`multiplie2(L,2)`
	`print(L)#[0,1,2]pas effet de bord`

II- Spécifications, commentaires et assertion

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 est inscrite au début du corps d'une fonction entre triples guillemets. Des commentaires sur chaque ligne peuvent être apportés pour aider à la relecture du programme.

```
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 entiers"""
    assert type(x)==int and type(y)==int#le programme se
poursuit si vrai
    return x+y
```

Commenté [AM1]: On peut aussi modifier des tableaux numpy, des valeurs de dictionnaires

Commenté [AM2]: Une variable globale est une variable définie en dehors du corps de la fonction. Cette variable, en étant mis en argument ou appelée dans le corps de la fonction peut être modifiée si elle est mutable.

Commenté [AM3]: Une instruction est un morceau de code qui produit un effet sur la machine lorsqu'elle est exécutée
Une instruction simple peut s'écrire sur une seule ligne.
Exemples d'instructions simples
-L'affectation
-Importer des modules avec *import*
-Renvoyer le résultat (ou les résultats sous forme d'un tuple) d'une fonction avec *return*
-Arrêter une boucle avec *break*
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 *for, while* dans les instructions conditionnelles *if, else, elif*.

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

Commenté [AM4]: Cette copie par référence permet une économie de la mémoire. La copie superficielle n'est cependant pas suffisante pour éviter les effets de bord dans le cas d'une liste de listes

```
L=[[1,1],[2,2]]
L2=L[:]
L2[0][1]=2
print(L)#[[1, 2], [2, 2]]

import copy
L=[[1,1],[2,2]]
L2=copy.deepcopy(L)
L2[0][1]=2
print(L)#[[1, 1], [2, 2]]
```