Documentation et mise au point de programmes - Exercices - CORRECTION

Documenter son programme

Exercice 1

Donnez un meilleur nom et une chaîne de documentation à la fonction suivante.

```
def f(a, b):
    return a + b
```

Correction

- Meilleur nom : `somme`
- Chaîne de documentation possible :

```
''' Renvoie la somme des deux nombres entrés en paramètres '''
```

Exercice 2

Donnez un meilleur nom et une chaîne de documentation à la fonction suivante, le paramètre `t` étant un tableau.

```
def f(t):
    s = 0
    for i in range(len(t)):
        s = s + t[i]
    return s / len(t)
```

Correction

En observant le code de la fonction, on se rend compte qu'à la fin de la boucle `for` la variable `s` contient la somme des éléments de `t`. La valeur renvoyée est donc la somme des éléments de `t` divisé par le nombre d'éléments de `t`.

- Meilleur nom : moyenne
- Chaîne de documentation possible :

```
'''
Renvoie la moyenne des éléments du tableau t.
t étant un tableau de nombres non vide.
'''
```

Vous noterez la nécessité d'indiquer que le tableau doit être non vide sinon `len(t)` vaut zéro et il y a une division par zéro. Il faut aussi que la variable `s` soit un nombre pour pouvoir faire le calcul `s / len(t)` donc on a précisé que `t` doit contenir des nombres.

Programmation défensive

Exercice 3

On considère la fonction `indice_maxi_tab(T)` suivante. A l'aide de la construction `assert`, proposez un test vérifiant si la précondition sur le tableau T est validée (*on ne cherchera pas à écrire la fonction*).

```
def indice_maxi_tab(T):
    """
    Renvoie l'indice de la première occurence de la valeur
    maximale du tableau T. T est supposé non vide.
    """
    # TEST A ECRIRE ICI
```

Correction

Il faut vérifier que `T` est non vide donc on peut par exemple vérifier que la longueur de `T` n'est pas nulle avec le test suivant :

```
assert len(T) != 0
```

Exercice 4

On considère la fonction `quotient(a, b)` suivante. A l'aide de la construction `assert`, proposez un test vérifiant si les préconditions sont validées.

```
def quotient(a, b):
    '''
    Renvoie la valeur du quotient de a par b, b étant non nul.
    '''
    # TEST A ECRIRE ICI
```

Correction

Il faut vérifier que `b` n'est pas nul. Le test suivant permet de le faire :

```
assert b != 0
```

Tester ses programmes

Exercice 5

1. En utilisant `assert`, donnez un jeu de tests de qualité pour la fonction suivante.
2. Faites ensuite passer vos tests à la fonction.

```
def multiplication(a, b):
    '''
    Renvoie le produit de a par b,
    où a et b sont deux nombres quelconques.
    '''
    return a * b
```

Correction

On peut proposer le jeu de tests suivant en prenant soin de tester les cas particuliers.

```
assert multiplication(2, 4) == 8  # test classique
assert multiplication(0, 3) == 0  # multiplication par zéro
assert multiplication(5, -6) == -30 # produit de nombres de signes contraires
assert multiplication(-5, -1) == 5  # produit de deux nombres négatifs
```

Vous remarquerez que l'on n'a pas testé les multiplications de deux nombres réels car il faudait alors comparer deux nombres réels, ce qu'il faut absolument éviter à cause de leur représentation approximative. Par exemple, le test suivant n'est pas validé alors qu'il devrait l'être :

```
assert multiplication(0.1, 0.2) == 0.02
```

```
---------------------------------------------------------------------------

AssertionError                            Traceback (most recent call last)

<ipython-input-13-6925c4341a2c> in <module>
----> 1 assert multiplication(0.1, 0.2) == 0.02


AssertionError:
```

Références :

- Documents ressources du DIU EIL Nantes, C. DECLERCQ.
- Numérique et Sciences Informatiques, 1re, T. BALABONSKI, S. CONCHON, J.-C. FILLIATRE, K.

NGUYEN, éditions ELLIPSES : Site du livre

Germain BECKER & Sébastien POINT, Lycée Mounier, ANGERS