Informatique PCSI
TP 9a : écriture d'un programme

Exercice 1

La fonction définie ci-dessous a-t-elle un effet de bord ? Autrement dit, est-ce que la valeur de la variable `m` est modifiée après l'exécution du code ? Répondre avant de tester le programme.

```
m = [[0, 1], [2, 3]]

def f(liste):
    liste = 2 * liste + m
    n = len(liste)
    liste[n-1][0] = 4
    return liste

f([0])
```

Exercice 2

On définit la suite de Collatz ou suite de Syracuse de la manière suivante. Le premier terme est un entier naturel non nul n. Si un terme est pair, le suivant vaut sa moitié, si un terme est impair, le suivant vaut le triple plus un. Mathématiquement, on écrit $u_0 = n$ et $u_{k+1} = u_k/2$ si u_k est pair et $u_{k+1} = 3u_k + 1$ si u_k est impair.

La fonction qui suit prend en paramètre un entier naturel `n` et renvoie la liste des termes de la suite.

```
def syracuse(n):
    liste = [n]
    while n != 1:
        if n % 2:
            n = 3 * n + 1
        else:
            n = n // 2
        liste.append(n)
    return liste
```

Ajouter une première assertion permettant de vérifier que `n` est strictement positif.

Il peut y avoir un problème si la fonction est appelée avec un flottant dont la valeur n'est pas entière. Proposer deux manières d'éviter ce problème, soit par une assertion, soit en modifiant une ligne (unique) du programme.

Exercice 3

La fonction écrite à la suite calcule le n-ième nombre de Fibonacci. Ces nombres notés u_n sont définis par $u_0 = 0$, $u_1 = 1$ et $u_n = u_{n-1} + u_{n-2}$ pour tout entier $n > 1$.

Cette fonction a-t-elle un effet de bord ?

```
dico = {0: 0, 1: 1}

def fibo(n):
    for i in range(2, n+1):
        dico[i] = dico[i-2] + dico[i-1]
    return dico[n]
```

Exercice 4

Les deux codes qui suivent sont censés calculer le n-ième nombre de Fibonacci (ces nombres sont définis dans l'exercice précédent).

Commenter ces codes. Écrire un code sans effet de bord qui fournit un résultat correct.

Les deux listes `liste1` et `liste2` sont des variables globales donc modifiables par une fonction.

```
liste1 = [0, 1]
def fibo1(n):
    for i in range(2, n+1):
        liste1.append(liste1[i-2] + liste1[i-1])
    return liste1[n]

print(fibo1(3), fibo1(5))

liste2 = [0, 1]
def fibo2(n):
    for i in range(2, n+1):
        liste2.append(liste2[i-2] + liste2[i-1])
    return liste2[len(liste2)-1]

print(fibo2(3), fibo2(5))
```

Exercice 5

Écrire une fonction `ajoute` qui prend en paramètres deux listes de nombres de même longueur et renvoie une nouvelle liste construite en effectuant la somme terme à terme des éléments des deux listes. Utiliser l'instruction `assert` pour vérifier les longueurs des listes.