Exercices - La récursivité

EXERCICE 1 :

La fonction *s()* donnée ci-contre a comme paramètre un entier n. Elle retourne la somme suivante :

$$s(n) = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-1} + \frac{1}{n}$$

```
def s(n) :
    s = 1
    for i in range(1,n+1) :
        s = s + 1/i
    return s
```

```
>>> s(3)
1.8333333333333333
```

On donne en exemple l'exécution donnée ci-contre, à gauche.

⇨ Ecrire **une version récursive** de la fonction *s()* et compléter les 2 tableaux ci-dessous pour l'exécution s(3)

Empilement dans la Pile d'exécution	*Dépilement* – Affichage dans la console
$s(1) = 1$	
$s(2) = s(1) + \frac{1}{2}$	$s(2) = 1 + \frac{1}{2} = \frac{3}{2}$
$s(3) = s(2) + \frac{1}{3}$	$s(3) = \frac{3}{2} + \frac{1}{3} = \frac{11}{6} \approx 1.83$

Corrigé

```
def s(n):
    if n == 1 : return 1
    else :
        return s(n-1) + 1/n
```

EXERCICE 2 :

La fonction *nombre()* donnée ci-contre a comme paramètre une liste ℓ de nombres. Elle retourne le nombre de valeurs impaires contenues dans cette liste.

```
def nombre(l):
    nb = 0
    for val in l:
        if val%2 == 1 : nb = nb + 1
    return nb
```

```
>>> nombre([1,4,7])
2
```

On donne en exemple l'exécution donnée ci-contre, à gauche.

⇨ Ecrire **une version récursive** de la fonction nombre*()* et compléter les 2 tableaux ci-dessous pour l'exécution *nombre([1,4,7])*

Empilement dans la Pile d'exécution	*Dépilement* – Affichage dans la console
$nombre([\,]) = 0$	
$nombre([1]) = nombre([\,]) + 1\%2$	$nombre([1]) = 0 + 1$
$nombre([1,4]) = nombre([1]) + 4\%2$	$nombre([1,4]) = 1 + 0 = 1$
$nombre([1,4,7]) = nombre([1,4]) + 7\%2$	$nombre([1,4,7]) = 1 + 1 = 2$

Corrigé avec analyse du dernier élément

```
def nombre(l):
    if l==[]: return 0
    else : return nombre(l[:-1]) + l[-1]%2
```

Ou

Corrigé avec analyse du 1er élément

```
def nombre(l):
    if l==[]: return 0
    else : return nombre(l[1:]) + l[0]%2
```

EXERCICE 3 :

En mathématiques, la suite de Fibonacci est une suite de nombres entiers dans laquelle chaque nombre est la somme des deux nombres qui le précèdent. Elle commence par les nombres 0 et 1 puis se poursuit avec 1, 2, 3, 5, 8, ….

$fibo(0) = 0$

$fibo(1) = 1$

$fibo(2) = fibo(1) + fibo(0) = 0 + 1 = 1$

$fibo(3) = fibo(2) + fibo(1) = 1 + 1 = 2$

$fibo(4) = fibo(3) + fibo(2) = 2 + 1 = 3$

$fibo(5) = fibo(4) + fibo(3) = 3 + 2 = 5$

$fibo(6) = fibo(5) + fibo(4) = 5 + 3 = 8$

$fibo(7) = fibo(6) + fibo(5) = 8 + 5 = 13$

$fibo(8) = fibo(7) + fibo(6) = 13 + 8 = 21$, etc ….

```
def fibo(n) :
    if n==0 : return 0
    if n==1 : return 1
    f_2 = 0
    f_1 = 1
    for i in range(2,n+1) :
        f = f_2 + f_1
        f_2 = f_1
        f_1 = f
    return f
```

On donne ci-dessus une version **itérative** de la fonction *fibo()* avec ci-contre, un exemple l'exécution :

```
>>> fibo(3)
2
```

⇨ Ecrire **une version récursive** de la fonction *fibo()* et compléter les 2 tableaux ci-dessous pour l'exécution fibo(3)

Empilement dans la Pile d'exécution	*Dépilement* – Affichage dans la console
$fibo(1) = 1$ et $fibo(0) = 0$	
$fibo(2) = fibo(1) + fibo(0)$	$fibo(2) = 1 + 0 = 1$
$fibo(3) = fibo(2) + fibo(1)$	$fibo(3) = 1 + 1 = 2$

Corrigé

```
def fibo(n) :
    if n <= 1 : return n
    else :
        return fibo(n-1)+fibo(n-2)
```

EXERCICE 4 :

La fonction *envers()* donnée ci-contre a comme paramètre un string. Elle retourne les caractères de ce string à l'envers.

```
>>> envers("ylnarb")
'branly'
```

On donne en exemple l'exécution donnée ci-contre, à gauche.

```
def envers(mot):
    new = ""
    for c in mot :
        new = c + new
    return new
```

⇨ Ecrire **une version récursive** de la fonction envers*()* et compléter les 2 tableaux ci-dessous pour l'exécution *envers('ylnarb')*

Empilement dans la Pile d'exécution	*Dépilement* – Affichage dans la console
$envers('') = ''$	
$envers('y') = 'y' + envers('')$	$envers('y') =' y'+''= 'y'$
$envers('yl') = 'l' + envers('y')$	$envers('yl') =' l'+'y' = 'ly'$
$envers('yln') = 'n' + envers('yl')$	$envers('yln') =' n'+'ly' = 'nly'$
$envers('ylna') = 'a' + envers('yln')$	$envers('ylna') =' a'+'lny' = 'anly'$
$envers('ylnar') = 'r' + envers('ylna')$	$envers('ylnar') =' r'+'anly' = 'ranly'$
$envers('ylnarb') = 'b' + envers('ylnar')$	$envers('ylnarb') =' b'+'ranly' = 'branly'$

Corrigé

```
def envers(mot) :
    if mot == "" : return ""
    else :
        return mot[-1] + envers(mot[:-1])
```