

Exercices -

La récursivité

EXERCICE 1 :

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

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

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

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

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

⇒ 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

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.

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

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

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

⇒ 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

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, ...

$$fibonacci(0) = 0$$

$$fibonacci(1) = 1$$

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

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

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

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

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

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

$$fibonacci(8) = fibonacci(7) + fibonacci(6) = 13 + 8 = 21, \text{ etc ...}$$

```
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 *fibonacci()* avec ci-contre, un exemple l'exécution :

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

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

Empilement dans la Pile d'exécution

Dépilement – Affichage dans la console

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')*

