
Devoir sur table

Récurtivité

Durée de l'épreuve : 1h

L'usage de la calculatrice n'est pas autorisé.

Exercice 1 (2 points)

La somme des n premiers entiers

Pour définir la somme des n premiers entiers, on a l'habitude d'écrire la formule :

$$0+1+2+3+4+\dots+n$$

Vous allez écrire la fonction récursive `somme(n)` qui calcule la somme des n premiers entiers.

Consigne : Recopiez le code suivant et complétez sur votre copie :

```
def somme(n) :  
    if n == ....:  
        return 0  
    else :  
        return n + ...
```

Correction:

```
def somme(n) :  
    if n == 0 :  
        return 0  
    else :  
        return n + somme(n-1)
```

Exercice 2 (2 points)

Le calcul de la factorielle d'un nombre est une opération mathématique qui consiste à multiplier ce nombre par tous les entiers positifs inférieurs ou égaux à lui. La factorielle d'un nombre entier positif n est notée $n!$

La **factorielle** d'un nombre n , notée $n!$, est le produit des entiers de 1 à n . Par exemple :

- $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$
- $0! = 1$ (par définition)

Nous pouvons donc utiliser la définition récursive suivante :

- Si $n = 0$, alors $n! = 1$ (cas de base).
- Sinon, $n! = n \times (n - 1)!$ (cas récursif).

Vous allez écrire la fonction récursive appelée factorielle(n) qui prend un entier positif n en paramètre et retourne $n!$

Consigne : Recopiez le code suivant et complétez sur votre copie :

```
def factorielle(n):  
    if n == 0:  
        return ....  
    else:  
        return n * ....
```

Correction:

```
def factorielle(n):  
    if n == 0:  
        return 1 # Cas de base  
    else:  
        return n * factorielle(n - 1) # Appel récursif
```

Exercice 3 (4 points)

1. Voici une fonction codée en Python :

```
def f(n):
    if n == 0:
        print("Partez!")
    else:
        print(n)
        f(n-1)
```

- Qu'affiche la commande `f(5)` ?
- Pourquoi dit-on de cette fonction qu'elle est récursive ?

2. On rappelle qu'en python l'opérateur `+` a le comportement suivant sur les chaînes de caractères :

```
>>> S = 'a'+'bc'
>>> S
'abc'
```

Et le comportement suivant sur les listes :

```
>>> L = ['a'] + ['b', 'c']
>>> L
['a', 'b', 'c']
```

On a besoin pour les questions suivantes de pouvoir ajouter une chaîne de caractères `s` en préfixe à chaque chaîne de caractères de la liste `liste`. On appellera cette fonction `ajouter`.

Par exemple, `ajouter("a", ["b", "c"])` doit retourner `["ab", "ac"]`.

- Recopiez le code suivant et complétez `.....` sur votre copie :

```
def ajouter(s, liste):
    res = []
    for m in liste:
        res.....
    return res
```

- Que renvoie la commande `ajouter("b", ["a", "b", "c"])` ?
- Que renvoie la commande `ajouter("a", [])` ?

3. On s'intéresse ici à la fonction suivante écrite en Python où s est une chaîne de caractères et n un entier naturel.

```
def produit(s, n):  
    if n == 0:  
        return []  
    else:  
        res = []  
        for i in range(len(s)):  
            res = res + ajouter(s[i], produit(s, n-1))  
        return res
```

- a. Que renvoie la commande `produit("ab", 0)` ? Le résultat est-il une liste vide ?
- b. Que renvoie la commande `produit("ab", 1)` ?
- c. Que renvoie la commande `produit("ab", 2)` ?

Q1.a

5

4

3

2

1

Partez !

Q1.b

elle s'appelle elle-même

Q2.a

```
def ajouter(s, liste) :  
    res = []  
    for m in liste :  
        res.append(s + m)  
    return res
```

Q2.b

["ba", "bb", "bc"]

Q2.c

["a"]

Q3.a

""

Q3.b

["a", "b"]

Q3.c

["aa", "ab", "ba", "bb"]

Exercice 4 (2 points)

Les deux fonctions f1_rec et f2_rec ci-dessous font les mêmes opérations, mais l'instruction print() est placée différemment.

Qu'affiche f(4) ?

```
def f1_rec(i):
    if (i == 8):
        print('cas de base:', i)
    else:
        print(i)
        f1_rec(i+1)

def f2_rec(i):
    if (i == 8):
        print('cas de base:', i)
    else:
        f2_rec(i+1)
        print(i)

def f(n):
    f1_rec(n)
    print('----')
    f2_rec(n)
```

Correction:

```
1 def f1_rec(i):
2     if (i == 8):
3         print('cas de base:', i)
4     else:
5         print(i)
6         f1_rec(i+1)
7
8 def f2_rec(i):
9     if (i == 8):
10        print('cas de base:', i)
11    else:
12        f2_rec(i+1)
13        print(i)
14
15 def f(n):
16     f1_rec(n)
17     print('----')
18     f2_rec(n)
19
20 f(4)
```

Console ×

```
>>> %Run -c $EDITOR_CONTENT
```

```
4
5
6
7
cas de base: 8
----
cas de base: 8
7
6
5
4
```

```
>>>
```