> D'après 2023, sujet zéro B
> Auteurs : Franck Chambon, Sébastien Hoarau

Fonctions récursives

> D'après 2023, Sujet 0.b, Ex. 2

Cet exercice est consacré à l'analyse et à l'écriture de programmes récursifs.

1.a) Expliquer en quelques mots ce qu'est une fonction récursive.

> ✓ **"Réponse"**
>
> Une fonction récursive est une fonction qui possède un appel à elle-même dans son code source.

1.b) On considère la fonction Python suivante :

```
def compte_rebours(n):
    """ n est un entier positif ou nul """
    if n >= 0:
        print(n)
        compte_rebours(n - 1)
```

L'appel `compte_rebours(3)` affiche successivement les nombres `3`, `2`, `1` et `0`.
Expliquer pourquoi le programme s'arrête après l'affichage du nombre `0`.

> ✓ **"Réponse"**
>
> Une fois l'affichage de `0` effectué, il y a un appel récursif `compte_rebours(0 - 1)`.
>
> Lors de cet appel récursif, `n` vaut `-1`, on ne rentre pas donc dans la structure conditionnelle.
>
> La pile d'appel récursif se vide sans qu'il ait d'autres instructions effectuées.
>
> Ainsi le programme s'arrête après avoir affiché `0` et vidé la pile d'appels récursifs.

2. En mathématiques, la factorielle d'un entier naturel n est le produit des nombres entiers strictement positifs inférieurs ou égaux à n. Par convention, la factorielle de 0 est 1. Par exemple :

- la factorielle de 1 est 1
- la factorielle de 2 est $2 \times 1 = 2$
- la factorielle de 3 est $3 \times 2 \times 1 = 6$
- la factorielle de 4 est $4 \times 3 \times 2 \times 1 = 24$

Recopier et compléter sur votre copie le programme donné ci-dessous afin que la fonction récursive `fact` renvoie la factorielle de l'entier passé en paramètre de cette fonction.

Exemple : `#!py fact(4)` renvoie `24`.

```
def fact(n):
    """ Renvoie le produit des entiers strictement positifs
         et inférieurs ou égaux à n.
    """
    if n == 0:
        return ...  # À compléter
    else:
        return ...  # À compléter
```

"Réponse"

```
def fact(n):
    """ Renvoie le produit des entiers strictement positifs
        et inférieurs ou égaux à n.
    """
    if n == 0:
        return 1
    else:
        return n * fact(n - 1)
```

3. La fonction `somme_entiers_rec` ci-dessous permet de calculer la somme des entiers, de `0` à l'entier naturel `n` passé en paramètre.

Par exemple :

- Pour `#!py n = 0`, la fonction renvoie la valeur `0`.

- Pour `#!py n = 1` , la fonction renvoie la valeur `0 + 1 = 1` .
- ...
- Pour `#!py n = 4` , la fonction renvoie la valeur `0 + 1 + 2 + 3 + 4 = 10` .

```
def somme_entiers_rec(n):
    """ Renvoie, de manière récursive,
     la somme des entiers de 0 à l'entier naturel n.
    """
    if n == 0:
        return 0
    else:
        print(n)  # pour vérification
        return n + somme_entiers(n - 1)
```

L'instruction `#!py print(n)` de la ligne 7 dans le code précédent a été insérée afin de mettre en évidence le mécanisme en œuvre au niveau des appels récursifs.

3.a) Écrire ce qui sera affiché dans la console après l'exécution de la ligne suivante :

```
>>> res = somme_entiers_rec(3)
```

"Réponse"

```
>>> res = somme_entiers_rec(3)
3
2
1
>>>
```

- L'appel `somme_entiers_rec(3)` affiche `3` puis appelle `somme_entiers_rec(2)`
- L'appel `somme_entiers_rec(2)` affiche `2` puis appelle `somme_entiers_rec(1)`
- L'appel `somme_entiers_rec(1)` affiche `3` puis appelle `somme_entiers_rec(0)`
- L'appel `somme_entiers_rec(0)` n'affiche rien.

3.b) Quelle valeur sera alors affectée à la variable `res` ?

"Réponse"

```
>>> res = somme_entiers_rec(3)
3
2
1
>>> res
6
```

La valeur `6` est affectée à la variable `res`, la somme $3 + 2 + 1 + 0$.

4. Écrire en Python une fonction `somme_entiers` non récursive : cette fonction devra prendre en argument un entier naturel `n` et renvoyer la somme des entiers de `0` à `n` compris. Elle devra donc renvoyer le même résultat que la fonction `somme_entiers_rec` définie à la question 3.

Exemple : `#!py somme_entiers(4)` renvoie `10`.

✓ **"Réponse"**

Il y a plusieurs solutions, par exemple :

```python
def somme_entiers(n):
    # style itératif
    somme = 0
    for x in range(n + 1):
        somme += x
    return somme


def somme_entiers(n):
    # style fonctionnel
    return sum(range(n + 1))
```