Correction du DS 2

Informatique pour tous, première année

Julien Reichert

Exercice 1

```
def produit_f(f,n):
    p = 1
    for i in range(n+1):
        p = p * f(i)
    return p

def fact(n):
    def fonction_triviale(m): # Qu'on peut définir localement
        return m+2 # Surtout ne pas multiplier par 0, et pas besoin de multiplier par 1
    return produit_f(fonction_triviale,n-2)
```

Exercice 2

Il est possible d'utiliser des tranches ici aussi.

```
def tous_differents(l):
    for i in range(len(l)):
        if l[i] in l[i+1:]:
            return False
    return True
```

Sinon, une version plus classique :

```
def tous_differents(l):
    for i in range(len(l)):
        for j in range(i+1,len(l)):
            if l[i] == l[j]
                return False
    return True
```

On notera que l'opérateur `in` teste l'appartenance d'un élément égal dans la liste. En particulier, `0.0 in [0]` est vrai.

Exercice 3

`seq1` contient les éléments de `s` aux indices `3` à `6` (exclu). Pour rappel, les indices commençant à `0`, l'élément à l'indice `3` est le quatrième. On note aussi que la taille de la chaîne obtenue est `6-3`, comme pour `range`.

La réponse est donc `"L E"`.

`seq2` contient les éléments de `l` aux indices multiples de `5` entre `50` et `80` (exclu), en notant que `l` est la liste des carrés des entiers de `0` à `99`. Le calcul des carrés des multiples de `5` est aisé (penser aux identités remarquables).

On trouve rapidement `[2500, 3025, 3600, 4225, 4900, 5625]`.

`seq3` contient les éléments de `s` aux indices `4` à `-5`, c'est-à-dire que le cinquième caractère en partant de la fin sera le premier caractère exclu. Sans qu'on ait besoin de se fatiguer à calculer la taille de `s`, on déduit la réponse.

Il s'agit de `" EST LE PRODUIT DE SIX PAR"`.

`seq4` est la fusion de deux tranches : d'une part celle qui contient les éléments de `l` aux indices `-1` (c'est-à-dire `99`) à `1` par pas de `-25`, donc les indices `99`, `74`, `49` et `24` ; d'autre part celle qui contient les éléments d'indice `1` à `1000` par pas de `100`, où seul l'indice `1` est dans le bon intervalle.

Le résultat final est donc `[9801, 5476, 2401, 576, 1]`.

Exercice 4

```
def extrait(seq,i,j,n):
    if n == 0:
        raise ValueError("le pas de la tranche ne peut pas être nul")
    if i < 0:
        i = len(seq)+i # Attention, i étant négatif, il ne faut pas le soustraire
    if j < 0:
        j = len(seq)+j
    if (i >= len(seq) and j >= len(seq)) or (i < 0 and j < 0) :
        return [] # Double débordement dans le même sens : résultat vide
    i = max(0,min(len(seq)-1,i)) # Simple débordement : on ramène au bord inclus
    j = max(-1,min(len(seq),j)) # Simple débordement : on ramène au bord exclu
    l = []
    while i != j and (i < j) == (n > 0): # i < j si n est positif, i > j sinon
    # En particulier, si par exemple i < j et n < 0, on n'entre jamais dans la boucle
        l.append(seq[i])
        i = i + n
    return l
```

La dernière partie est sans doute plus simple en utilisant un `if`, ce qui donne :

```
    if n > 0:
        while i < j:
            l.append(seq[i])
            i = i + n
    else:
        while i > j:
            l.append(seq[i])
            i = i + n
```