Correction du DS 2

Julien Reichert

Exercice 1

```
def compte(n):
    l = []
    for i in range(1, n+1):
        for j in range(i): # ou l.extend([i]*i)
            l.append(i)    # sans boucle et en une ligne
    return l
```

Exercice 2

```
def deux_egaux(l):
    for i in range(len(l)):
        for j in range(i): # ou range(i+1, len(l)), en tout cas garantir i différent de j
            if l[i] == l[j]: # ou boucles normales et tester i != j and ...
                return True
    return False
```

Exercice 3

En raison du `range(n)` dans la fonction `dbi`, la valeur `n` ne sera pas considérée, ce qui fait un tour de moins dans la boucle extérieure (et de même dans chaque tour de la boucle intérieure). Les fonctions ne sont donc pas équivalentes, mais elles le seraient par exemple avec des opérateurs de comparaison stricts dans `dbc`. On note que dans cette fonction, l'initialisation de `i` comme de `j` pourraient se faire à la valeur `1` sans impact sur le résultat.

En outre, on constate que le nombre `j` est additionné à `s` exactement `n-j+1` fois, c'est-à-dire pour tous les `i` allant de `j` à `n`, et donc mathématiquement la valeur retournée, en tant que somme, est :

$$\sum_{j=1}^{n} j(n-j+1) = \frac{n(n+1)}{2}(n+1) - \frac{n(n+1)(2n+1)}{6} = \frac{n(n+1)(n+2)}{6}$$

et ceci correspond (mais ce n'est pas un hasard) à $\binom{n+2}{3}$. On pourra consulter à ce sujet un très bel exercice de colle en combinatoire.

Exercice 4

```
def minmax(l):
    mini = l[0]
    maxi = l[0]
    for i in range(len(l)):
        if l[i] > maxi:
            maxi = l[i]
        if l[i] < mini:
            mini = l[i]
    return (mini, maxi)
```

Exercice 5

```
def liste_carres(l): # peut se faire en une ligne grâce aux listes en compréhension
    rep = []
    for element in l:
        rep.append(element**2)
    return rep
```

Exercice 6

Par des divisions successives par 2, on écrit les restes de droite à gauche et on obtient $\overline{1010111}^2$. Ceci tient sur 7 bits et quelle que soit la méthode de représentation on aura donc sur 8 bits 01010111 pour 87, et en complément à deux, par l'algorithme consistant à échanger les 1 et les 0 puis à ajouter 1, on aura 10101001 pour -87.

Exercice 7

```
def ppe():
     k = 0
     dpk = 1
     while 1 + dpk != 1:
         k = k - 1
         dpk = dpk / 2
     return k + 1
```

Le `k + 1` à la fin se justifie par le fait que `k` est la plus grande valeur pour laquelle la condition de la boucle n'est plus remplie. La réponse est 52, car c'est pour cette valeur que le dernier bit de la mantisse est alors à 1, justifiant une différence avec la valeur 0. Au-delà, en raison de l'arrondi au plus proche, la valeur est confondue avec 1.