Exercice 1
Donner la décomposition en binaire de 213.

Exercice 2
Un CAN (convertisseur analogique-numérique, qui transforme un signal physique en une suite de bit), dans une situation pratique non définie ici, doit pouvoir distinguer entre 1000 valeurs de la quantité mesurée (il y a potentiellement 1000 valeurs différentes que l'on souhaite pouvoir obtenir, on considère que ce sont les entiers entre 0 et 999).

1. Combien de bits faut-il utiliser au minimum pour stocker une mesure ?
2. On effectue une série de mesure à 1000 Hz pendant 10 secondes. Quelle est la taille, en KO, de l'ensemble des données produites ? On pourra donner le résultat sous forme de fraction.

Exercice 3
On considère la fonction suivante, où **L** est une liste de nombres entiers

```
def f(L):
    s = 1
    for i in range(len(L)):
        for j in range(i + 1, len(L)):
            if L[j] < L[i]:
                s = -s
    return s
```

1. Quelles sont les valeurs de retour possibles pour **f** ?
2. On note n la longueur de la liste **L**. Exprimer en fonction de n le nombre de fois où l'instruction **if** est exécutée.

Exercice 4
Ecrire une fonction **somme(n)** qui prend comme argument un entier positif n (ne pas le vérifier dans la fonction) et retourne $\sum_{k=1}^{n} k^3 = 1^3 + 2^3 + \cdots + n^3$.
Rappel : a^3 s'écrit en python `a**3`.

Exercice 5
Ecrire une fonction **occurrences(s, c)** qui prend comme argument une chaîne de caractère **s** et un caractère **c** et retourne le nombre de fois où le caractère **c** est présent dans la chaîne. Expliquer rapidement le rôle de chaque variable locale.

Exercice 6
On dispose d'une liste **L** dont chaque élément est une liste de la forme **[x, y1, y2]**.

1. Donner les instructions pour construire la liste **X** qui contient toutes les valeurs **x** présent en tant que premier élément de chaque élément de **L**.
2. On considère que l'on construit de même les listes **Y1** et **Y2**. Donner les instructions permettant d'afficher les courbes **Y1** en fonction de **X** et **Y2** en fonction de **X**. Les bibliothèques utiles sont considérées comme étant déjà importées, avec leurs noms usuels.

Exercice 7
On considère une liste $\mathbf{LV} = [v_0, \ldots, v_{n-1}]$ qui contient $n > 0$ nombres représentant les vitesses d'un objets aux instants $t_0, \ldots, t_{n-1}$. On note **T** la liste dont les éléments sont $t_0, \ldots, t_{n-1}$.

1. Écrire les instructions qui stockent dans la variable **t_vmax** l'instant où la vitesse maximale est atteinte. Ici, une fonction n'est pas demandée, seulement une suite d'instruction, en considérant les variables **LV** et **T** déjà définies.
2. Ecrire une suite d'instruction permettant de construire **LA**, la liste des accélérations aux instants $t_1, \ldots, t_{n-1}$.
   On considère qu'à l'instant t_i, l'accélération a_i vaut $a_i = \frac{v_i - v_{i-1}}{t_i - t_{i-1}}$. Expliquer rapidement quelle est l'approximation utilisée ici.
3. On considère que notre objet se déplace sur un axe, à partir de la position 0. Écrire les instructions permettant de construire **LP** la liste des positions sur l'axe aux instants $t_0, \ldots, t_{n-1}$.
   On utilisera la même approximation qu'à la question précédente pour traduire que la vitesse est la dérivée de la position.

Exercice 8
1. Écrire une fonction `puissance(x, n)` qui prend comme argument `x` un nombre et `n` un entier naturel et retourne x^n.
2. On considère maintenant le code suivant

```
def f(x, n):
    if x == 0:
        return 1
    elif n % 2 == 0:
        a = f(x, n // 2)
        return a * a
    else:
        return x * f(x, n - 1)
```

   (a) Expliquer l'exécution de `f(2, 3)`. On précisera en particulier quels sont les arguments passés aux différents appels à `f`.
   (b) On note $n = \overline{b_p \ldots b_0}_2$ l'écriture binaire de n. Dans le cas n pair, que peut-on dire de b_0 ? et dans le cas n impair ?
   (c) Donner l'écriture binaire de $n//2$ dans le cas n pair, et celle de $n-1$ dans le cas n impair.
   (d) Exprimer en fonction de p le nombre d'appel maximal à la fonction `f` pour calculer `f(x, n)`.