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 5000 valeurs de la quantité mesurée (il y a potentiellement 5000 valeurs différentes que l'on souhaite pouvoir obtenir).
Quelle profondeur déchantillonnage doit-il utiliser ? (ou encore, combien de bits, au minimum, doit-on utiliser pour pouvoir coder toutes les valeurs ?)

Exercice 3
Ecrire une fonction **factorielle(n)** qui prend comme argument un entier positif n (ne pas le vérifier dans la fonction) et retourne $n! = 1 \times 2 \times \cdots \times n$.

Exercice 4
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 5
On considère la fonction suivante, où **L** est une liste de nombres entiers

```python
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 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 usuelles.

Exercice 7
On considère une liste **LV** 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 $t_1, \ldots, t_{n-1}$. Préciser l'approximation utilisée.
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 $t_0, \ldots, t_{n-1}$. Préciser l'approximation utilisée.

Exercice 8
On considère une base de données comportant deux tables :

- La table **clients** dont les champs sont :
  - id (clé primaire, valeurs entières)
  - nom (valeur texte)
  - prenom (valeurs texte)
  - email (valeur texte)
  - adresse (valeur texte)
- La table **commandes**, dont les champs sont :
  - id (clé primaire, entier)
  - client_id (entier)
  - montant (entier, exprimé en centimes)
  - envoyee (entier, valant 0 ou 1, 1 quand la commande a été envoyée)

1. Donner la requête SQL permettant de trouver les commandes (on veut obtenir les id, client_id et montant) des commandes non-envoyées.
2. Écrire la requête donnant les nom, prenom, email et adresse des clients ayant passé une commande d'un montant supérieur ou égal à 100€.
3. Trouver le nombre de commande déjà envoyées, via une requête.
4. Écrire la requête donnant le nombre de commande par client.

Exercice 9
Écrire une fonction **est_triee(L)** qui prend en argument une liste **L** et retourne un booléen indiquant si **L** est triée dans l'ordre croissant.
Vous porterez une attention particulière à l'explication du code.

Tourner la page svp.

Exercice 10
On considère la fonction :

```
def f(x, n):
    e = n
    acc = x
    res = 1
    while e > 0:
        r = e % 2
        if r == 1:
           res = res * acc
        e = e // 2
        acc = acc * acc
    return res
```

1. Éventuellement à l'aide d'un tableau, expliciter les valeurs prises par les variables locales lors de l'exécution de **f(2, 3)**.
2. On note $\mathrm{C}(n)$ le nombre d'opérations élémentaires (addition, multiplication, division) pour calculer **x, n** par cette fonction. Pour $k \in \mathbb{N}$ évaluer $\mathrm{C}(2^k)$. On note $n = 2^k$. Calculer $\mathrm{C}(n)$ en fonction de n.
3. Expliciter les valeurs successives prises par les variables **r** et **acc** lors du calcul de **f(x, 213)**.
4. Expliquer pourquoi **f(x, n)** retourne x^n.
5. Proposer une autre fonction calculant x^n, et évaluer le nombres d'opérations élémentaires en fonction de n.