

DS : Requêtes SQL / Dictionnaires

CORRECTION

1 Exercice 1 : Tables de hachage

1. **Transformer une chaîne en entier** On souhaite calculer :

$$f(ch) = \sum_{k=0}^{n-1} \text{ord}(ch[k]) 256^k.$$

On parcourt donc les caractères de la chaîne et on ajoute à une variable la valeur de chaque caractère multipliée par la puissance de 256 correspondant à sa position.

Correction :

```
def chaineversentier(ch):  
    resultat = 0  
    for k in range(len(ch)):  
        resultat += ord(ch[k]) * 256**k  
    return resultat
```

Une autre écriture possible consiste à faire évoluer directement la puissance de 256 :

```
def chaineversentier(ch):  
    resultat = 0  
    puissance = 1  
    for k in range(len(ch)):  
        resultat += ord(ch[k]) * puissance  
        puissance *= 256  
    return resultat
```

2. **Calculer la valeur associée à une chaîne** Pour la chaîne 'ABC', on a :

$$\begin{aligned} f('ABC') &= \text{ord}('A') + \text{ord}('B') 256 + \text{ord}('C') 256^2 \\ &= 65 + 66 \times 256 + 67 \times 256^2. \end{aligned}$$

Or :

$$256^2 = 65536.$$

Donc :

$$\begin{aligned} f('ABC') &= 65 + 66 \times 256 + 67 \times 65536 \\ &= 65 + 16896 + 4390912 \\ &= \boxed{4407873}. \end{aligned}$$

3. **Construire la fonction de hachage** La fonction de hachage doit calculer le reste de la division euclidienne de la valeur obtenue à la question précédente par m .

Correction :

```
def chaineversentiermodulo(ch, m):
    return chaineversentier(ch) % m
```

Le reste d'une division euclidienne par m est nécessairement compris entre 0 et $m - 1$. Ainsi :

$$0 \leq \text{chaineversentiermodulo}(ch, m) < m$$

La valeur obtenue correspond donc toujours à un numéro de case valide de la table.

4. **Étudier une petite table de hachage** Pour une chaîne de deux caractères ch , on a :

$$f(ch) = \text{ord}(ch[0]) + \text{ord}(ch[1]) \bmod 256.$$

Comme :

$$256 \equiv 0 \pmod{4},$$

on obtient :

$$f(ch) \equiv \text{ord}(ch[0]) \pmod{4}.$$

Il suffit donc de connaître le code ASCII du premier caractère. On utilise :

Caractère	A	B	C	D
ord	65	66	67	68
ord (mod 4)	1	2	3	0

On obtient donc :

Chaîne	Case
AB	1
AC	1
AD	1
BA	2
BB	2
BC	2
CA	3
CB	3
CC	3
DA	0
DB	0
DC	0

La répartition est donc :

```
{ 0: 3, 1: 3, 2: 3, 3: 3 }
```

La distribution est ici parfaitement uniforme.

5. **Générer des mots et étudier leur répartition** Chaque caractère doit être choisi aléatoirement parmi les codes ASCII compris entre 65 et 90. La ligne :

```
mot += chr(randint(65, 90))
```

permet donc d'ajouter une lettre majuscule aléatoire au mot. La fonction complète est :

```
from random import randint
def repartition(N, m):
    repartition = {}
    for i in range(N):
        # Generation d'un mot de longueur 7
        mot = ""
        for j in range(7):
            mot += chr(randint(65, 90))
            # Calcul de la case correspondante
            case = chaineversentiermodulo(mot, m)
            #Comptage du nombre de mots par case
            if case in repartition:
                repartition[case] += 1
            else:
                repartition[case] = 1
        return repartition
```

La fonction génère bien N mots de longueur 7, puis associe chaque mot à une case de la table et compte les occurrences dans le dictionnaire. On a nécessairement :

$$\sum_{\text{case}} \text{repartition}[\text{case}] = N.$$

6. Tester l'uniformité

- (a) L'effectif théorique d'une case est :

$$\frac{N}{m} = \frac{1000}{12} = \frac{250}{3} \approx 83,33.$$

Il n'est donc pas possible d'obtenir exactement le même nombre de mots dans chaque case puisque cet effectif n'est pas entier. Une répartition aussi uniforme que possible comporterait donc principalement des cases contenant 83 mots et certaines contenant 84 mots.

- (b) On parcourt le dictionnaire et on compare chaque effectif à l'effectif théorique N/m . Par exemple, si une case contient 91 mots, son écart à l'effectif théorique est :

$$\left| 91 - \frac{1000}{12} \right| \approx 7,67.$$

On effectue ce calcul pour chacune des cases.

- (c) On cherche ensuite le plus grand de ces écarts. Une fonction possible est :

```
def ecart_maximal(repartition, N, m):
    moyenne = N / m
    ecart = 0
    for case in repartition:
        difference = abs(repartition[case] - moyenne)
        if difference > ecart:
            ecart = difference
    return ecart
```

La valeur renvoyée mesure donc l'écart entre la case la plus éloignée de l'effectif théorique et cet effectif théorique. Plus cette valeur est petite, plus la distribution est proche d'une distribution uniforme.

2 Exercice 2 : Requêtes SQL

On considère une base de données permettant de gérer des étudiants, des formations et les inscriptions des étudiants aux formations.

Structure de la base

Table ETUDIANT

NumEtudiant	Nom	Prenom	Ville
1	Martin	Alice	Marseille
2	Dupont	Hugo	Lyon
3	Bernard	Emma	Marseille
4	Petit	Lucas	Nice
5	Robert	Chloé	Lyon

Table FORMATION

NumFormation	Intitule	Responsable	Ville
101	Informatique	Durand	Marseille
102	Mathématiques	Morel	Lyon
103	Physique	Durand	Nice

Table INSCRIPTION

NumEtudiant	NumFormation	Annee	Note
1	101	2025	15
1	102	2025	12
2	101	2025	14
2	103	2025	11
3	101	2025	17
3	102	2025	16
4	103	2025	13
5	102	2025	10

1. Afficher le nom, le prénom et la ville de tous les étudiants.

```
SELECT Nom, Prenom, Ville  
FROM ETUDIANT;
```

2. Afficher le nom et le prénom des étudiants habitant à Marseille.

```
SELECT Nom, Prenom  
FROM ETUDIANT  
WHERE Ville = 'Marseille';
```

3. Afficher les différentes villes dans lesquelles sont situées les formations, sans répétition.

```
SELECT DISTINCT Ville  
FROM FORMATION;
```

4. Afficher le nom et le prénom de chaque étudiant ainsi que sa note obtenue. La requête doit utiliser les tables ETUDIANT et INSCRIPTION.

```
SELECT ETUDIANT.Nom, ETUDIANT.Prenom, INSCRIPTION.Note  
FROM ETUDIANT JOIN INSCRIPTION  
ON ETUDIANT.NumEtudiant = INSCRIPTION.NumEtudiant;
```

5. Afficher le nom et le prénom des étudiants ayant obtenu une note supérieure ou égale à 15. La requête doit utiliser les tables ETUDIANT et INSCRIPTION.

```
SELECT ETUDIANT.Nom, ETUDIANT.Prenom  
FROM ETUDIANT JOIN INSCRIPTION  
ON ETUDIANT.NumEtudiant = INSCRIPTION.NumEtudiant  
WHERE INSCRIPTION.Note >= 15;
```

6. Afficher le nom de l'étudiant et le numéro de la formation pour chaque inscription. La requête doit utiliser uniquement les tables ETUDIANT et INSCRIPTION.

```
SELECT ETUDIANT.Nom, INSCRIPTION.NumFormation  
FROM ETUDIANT JOIN INSCRIPTION  
ON ETUDIANT.NumEtudiant = INSCRIPTION.NumEtudiant;
```

7. Afficher, pour chaque étudiant, son numéro, son nom et sa moyenne obtenue aux différentes formations auxquelles il est inscrit. La requête doit utiliser uniquement les tables ETUDIANT et INSCRIPTION.

```
SELECT ETUDIANT.NumEtudiant, ETUDIANT.Nom, AVG(INSCRIPTION.Note) AS Moyenne  
FROM ETUDIANT JOIN INSCRIPTION  
ON ETUDIANT.NumEtudiant = INSCRIPTION.NumEtudiant  
GROUP BY ETUDIANT.NumEtudiant, ETUDIANT.Nom;
```