

Devoir n°1

3 heures

Exercice :

1. On donne le code python suivant :

```
def mystere(chaine):  
    n = len(chaine)  
    for i in range(n):  
        if chaine[i] != chaine[n-i-1]:  
            return False  
    return True
```

- (a) Exécuter cette fonction `mystere` sur la chaîne 'tout' ainsi que sur la chaîne 'ressasser'. On représentera l'évolution des quantités `i`, `n-1-i`, `chaine[i]` et `chaine[n-1-i]` dans un tableau (classiquement).
 - (b) Que fait cette fonction `mystere` ? Quelle est sa complexité ?
 - (c) Optimiser la fonction `mystere`.
2. On souhaite écrire une fonction python qui détermine l'écriture en binaire d'un entier n . On rappelle que 10 en binaire s'écrit 1010 puisque $10 = 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0$.

- (a) Quelle est l'écriture binaire de 24 ? 32 ? 255 ?
- (b) Un algorithme possible se base sur les divisions successives par 2. Par exemple, pour obtenir l'écriture binaire de 100, on écrit :
 $100 = 2 \times 50 + 0$
 $50 = 2 \times 25 + 0$
 $25 = 2 \times 12 + 1$
 $12 = 2 \times 6 + 0$
 $6 = 2 \times 3 + 0$
 $3 = 2 \times 1 + 1$
 $1 = 2 \times 0 + 1$: le dernier quotient est nul, on s'arrête.
Alors, l'écriture de 100 en base 2 est 1100100 (liste des restes à l'envers), on a bien $100 = 1.2^6 + 1.2^5 + 0.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 0.2^0 = 64 + 32 + 4$.
Compléter le code qui suit :

```
def binaire(n):
    chaine = "" # initialisation de l'écriture binaire
    q = n # nombre courant
    while q != ??? :
        r = ??? # reste
        q = ??? # quotient, nouveau nombre courant
        chaine = ???
    if chaine == ??? # cas particulier
        return "0"
    return ???
```

- (c) Pour vérification, exécuter manuellement le code précédent avec $n = 91$ en représentant l'exécution par un tableau de l'évolution des variables q , r et $chaine$.

Problème : base de données

On dispose d'une base de données composée de quatre tables :

- la table **Continents** constituée des champs suivants :
 - **nom** : nom du continent (chaîne de caractères) ;
 - **surface** : surface du continent en kilomètres carrés (entier).
- la table **Pays** constituée des champs suivants :
 - **nom** : nom du pays (chaîne de caractères) ;
 - **code_pays** : identifiant unique du pays (chaîne de caractères) ;
 - **capitale** : capitale administrative du pays (chaîne de caractères) ;
 - **population** : nombre d'habitants du pays (entier).
- la table **Inclusion** constituée des champs suivants :
 - **code_pays** : identifiant unique du pays (chaîne de caractères) ;
 - **continent** : nom du continent auquel appartient le pays (chaîne de caractères).
- la table **Frontieres** constituée des champs suivants :
 - **code_pays1** : identifiant unique du premier pays (chaîne de caractères) ;
 - **code_pays2** : identifiant unique du second pays (chaîne de caractères) ;
 - **longueur** : longueur en kilomètres de la frontière entre **pays1** et **pays2** (nombre flottant strictement positif).

On a toujours $code_pays1 < code_pays2$ pour l'ordre lexicographique, ce qui assure que chaque frontière n'apparaît qu'une fois dans la table **Frontieres**.

Table **Continents**

nom	surface
Asie	44579000
Europe	9938000
...	...

Table **Pays**

nom	code_pays	capitale	population
Albanie	AL	Tirane	30883385
Algérie	DZ	Alger	44487616
...	...	...	...

Table Inclusion	
code_pays	continent
AL	Europe
DZ	Afrique
...	...

Table Frontieres		
code_pays1	code_pays2	longueur
AL	GR	282.8
AL	MK	151.5
...	...	...

Les requêtes demandées dans cette partie sont à écrire en langage SQL.

1. Combien d'attributs possède la table **Pays** ?
2. Expliquer le type de l'attribut **population** et son domaine.
3. L'attribut **code_pays** de la table **Inclusion** est-il une clé primaire et pourquoi ?
4. L'attribut **code_pays1** de la table **Frontiere** est-il une clé primaire et pourquoi ?
5. Quelles sont les clés primaires de la table **Pays** ?
6. Comment désigne t-on l'attribut **code_pays2** de la table **Frontiere** et pourquoi ?
7. La table **Frontiere** possède-t-elle une clé primaire ? Préciser et apporter éventuellement des solutions.
8. Dans cette base de données, combien y-a-t-il d'entités et d'associations ?
9. De quel type est l'association entre les tables **Pays** et **Continents** ? Détaillez l'association et ses deux couples de cardinalité.
10. Quel est le rôle de la table **Frontiere** ? On précisera le cas échéant l'association et les cardinalités.
11. Que faudrait modifier dans la base de données pour que la table **Inclusion** ne soit plus nécessaire ? On optimisera (minimisera) les modifications dans l'esprit des bases de données.
12. Écrire une requête SQL qui détermine le code du pays nommé 'France'.
13. Écrire une requête SQL qui liste les capitales de l'Europe.
14. Écrire une requête SQL permettant d'obtenir les noms des pays appartenant au continent 'Europe'.
15. Écrire une requête SQL qui liste les pays de l'Europe dont la première lettre du nom de la capitale appartient à $A, \dots, H$ ou à $P, \dots, X$. On rappelle que l'on peut comparer des chaînes de caractères et on suppose que le logiciel SQL est paramétré de sorte qu'il soit insensible à la casse (majuscule-minuscule) pour les valeurs des attributs.
16. Écrire une requête SQL qui détermine la liste des noms des dix pays les plus peuplés classés par population décroissante.
17. Écrire une requête SQL qui donne la liste des continents avec leur population totale classés par population croissante et ayant une population supérieure à 1 milliard.

18. Écrire une requête SQL qui donne la liste des continents avec le nombre de pays le constituant.
19. Écrire une requête SQL permettant de récupérer les noms des pays frontaliers du pays nommé 'France' (on ne connaît pas le code pays de la France).
20. Écrire une requête SQL permettant de récupérer la longueur de la frontière du pays nommé 'France' sachant que l'on sait que le code de pays nommé 'France' est 'F' et que l'on peut utiliser cette information.



Exercice :

1. Palindromes.

(a) Pour la chaîne 'tout' :

i	0	1
$3 - i$	3	2
$chaîne[i]$	t	o
$chaîne[3 - i]$	t	u

et pour la chaîne 'ressasser' :

i	0	1	2	3	4	5	6	7	8
$8 - i$	8	7	6	5	4	3	2	1	0
$chaîne[i]$	r	e	s	s	a	s	s	e	r
$chaîne[8 - i]$	r	e	s	s	a	s	s	e	r

(b) La fonction mystère teste si une chaîne est un palindrome (même chaîne lue à l'envers) et renvoie une réponse en booléen. Nous avons une boucle de taille la longueur de la chaîne, et ainsi la complexité est en $\mathcal{O}(n)$, n la longueur de la chaîne. Dans le meilleur des cas, le test est faux à la première étape de boucle, complexité $\mathcal{O}(1)$, dans le pire des cas, nous avons un palindrome et la boucle est complète, complexité en $\mathcal{O}(n)$.

(c) On peut optimiser la fonction `mystere` en remarquant qu'il suffit d'effectuer les comparaisons sur la moitié de la chaîne, et que dans le cas d'une chaîne de longueur impaire, le caractère du milieu se compare avec lui-même, ce qui est inutile, d'où la simplification :

```
def mystere(chaîne):
    n = len(chaîne)
    # si n pair, n=2p, n//2=p, p comparaisons, cf tout
    # si n impair, n=2p+1, n//2=p, p comparaisons, cf ressasser
    for i in range(n//2): # p étapes
        if chaîne[i] != chaîne[n-i-1]:
            return False
    return True
```

2. On souhaite écrire une fonction python qui détermine l'écriture en binaire d'un entier n . On rappelle que 10 en binaire s'écrit 1010 puisque $10 = 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0$.

(a) Nous avons directement $24 = 16 + 8 = 2^4 + 2^3 = 1.2^4 + 1.2^3 + 0.2^2 + 0.2^1 + 0.2^0 = \overline{11000}$, $32 = \overline{100000}$, $255 = 256 - 1 = 2^8 - 1 = 1 + 2 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 = \overline{11111111}$. On peut aussi poser les divisions successives, comme ci-dessous.

(b) Voici le code complété :

```
def binaire(n):
    chaine = "" # initialisation de l'écriture binaire
    q = n       # nombre courant
    while q != 0:
        r = q%2 # reste
        q = q//2 # quotient, nouveau nombre courant
        chaine = str(r) + chaine
    if chaine == "": # cas particulier
        return "0"
    return chaine
```

- (c) Pour vérification, exécuter manuellement le code précédent avec $n = 91$ en représentant l'exécution par un tableau de l'évolution des variables q , r et $chaine$.

q	91	45	22	11	5	2	1	0
r		1	1	0	1	1	0	1
$chaine$	"	'1'	'11'	'011'	'1011'	'11011'	'011011'	'1011011'

- Combien d'attributs possède la table **Pays** ?

La table **Pays** possède 4 attributs : **nom**, **code_pays**, **capitale** et **population**.

- Expliquer le type de l'attribut **population**.

L'attribut **population** est de type entier, de domaine $\mathbb{N}$.

- L'attribut **code_pays** de la table **Inclusion** est-il une clé primaire et pourquoi ?

Oui, il s'agit d'une clé primaire, il est unique pour chaque pays.

- L'attribut **code_pays1** de la table **Frontiere** est-il une clé primaire et pourquoi ?

Non, en effet, un pays et donc son code pays peut apparaître plusieurs fois dans la colonne **code_pays1** s'il possède plusieurs pays voisins : par exemple la France.

- Quelles sont les clés primaires de la table **Pays** ?

Les attributs uniques **nom**, **code_pays**, sans doute **capitale** sont chacun des clés primaires.

Il est probable que l'attribut **population** aussi, mais ce n'est pas raisonnable.

On peut aussi utiliser des couples d'attributs, mais cela est aussi peu raisonnable.

- Comment désigne-t-on l'attribut **code_pays2** de la table **Frontiere** et pourquoi ?

Il s'agit d'une clé secondaire, ou clé étrangère. Elle fait référence à l'attribut **code_pays** clé primaire de la table **Pays**.

7. La table **Frontiere** possède-t-elle une clé primaire ? Préciser et apporter éventuellement des solutions.

Il n'y a pas de clé primaire à attribut unique (**longueur** n'est encore une fois pas raisonnable). Par contre, le couple (**code_pays1**, **code_pays2**) est une clé primaire.
On pourrait ajouter une clé primaire par un attribut unique **id_frontiere** entier.

8. Dans cette base de données, combien y-a-t-il d'entités et d'associations ?

Nous avons 4 entités (tables) : **Continent**, **Pays**, **Inclusion** et **Frontiere**.

Nous avons deux associations :

- la table **Inclusion** relie des tables **Pays** et **Continents**.
- la table **Frontière** associe la table **Pays** avec elle-même.

9. De quel type est l'association entre les tables **Pays** et **Continents** ? Détaillez l'association et ses deux couples de cardinalité.

Un continent contient 1 à plusieurs pays, de type 1 – * ;
un pays appartient (principalement) à un seul continent, de type 1 – 1 ou de type 1 – * selon la construction de la base de données.

10. Quel est le rôle de la table **Frontiere** ? On précisera le cas échéant l'association et les cardinalités.

La table frontière relie la table **pays** avec elle même, de type * – *.
Un pays a de 0 à 1 frontière (si considérée dans sa globalité) ou de 0 à plusieurs frontières (si séparées) avec un autre pays. Nous avons une cardinalité symétrique de type 0 – 1 ou 0 – *.

11. Que faudrait modifier dans la base de données pour que la table **Inclusion** ne soit plus nécessaire ?
On optimisera (minimisera) les modifications dans l'esprit des bases de données.

On peut par exemple ajouter un attribut **continent** à la table **Pays**. L'idéal, serait que les tables possèdent des attributs entiers **id** pour utiliser des références numériques (on identifierait alors le continent par son identifiant, et non par son nom).

12. Écrire une requête qui détermine le code du pays nommé 'France'.

```
SELECT code_pays FROM Pays WHERE nom = 'France'
```

13. Écrire une requête SQL qui liste les capitales de l'Europe.

```
SELECT capital FROM Pays  
JOIN Inclusion ON Pays.code_pays = Inclusion.code_pays  
WHERE continent = 'Europe'
```

14. Écrire une requête SQL permettant d'obtenir les noms des pays appartenant au continent 'Europe'.

```
SELECT nom FROM Pays
JOIN Inclusion ON Pays.code_pays = Inclusion.code_pays
WHERE continent = 'Europe'
```

15. Écrire une requête SQL qui liste les pays de l'Europe dont la première lettre du nom de la capitale appartient à $A, \dots, H$ ou $P, \dots, X$.

```
SELECT nom FROM Pays
JOIN Inclusion ON Pays.code_pays = Inclusion.code_pays
WHERE continent = 'Europe' AND
(capitale < 'I' OR (capitale >= 'P' AND capitale < 'Y'))
```

16. Écrire une requête SQL qui détermine la liste des noms des dix pays les plus peuplés classés par population décroissante.

```
SELECT nom FROM Pays
ORDER BY population DESC LIMIT 10
```

17. Écrire une requête SQL qui détermine donne la liste des continents avec leur population totale classés par population croissante et ayant une population supérieure à 1 milliard.

```
SELECT continent, SUM(population) AS population_totale
FROM Inclusion
JOIN Pays AS Pays.code_pays = Inclusion.code_pays
GROUP BY continent HAVING population_totale >= 10^9
ORDER BY population_totale
```

18. Écrire une requête SQL qui donne la liste des continents avec le nombre de pays les constituant.

```
SELECT continent, COUNT(*) AS nombre_pays FROM Inclusion
GROUP BY continent
```

19. Écrire une requête SQL permettant de récupérer les noms des pays frontaliers du pays nommé 'France'.

```
SELECT nom FROM Pays
JOIN Frontiere ON ( code_pays1 = code_pays
AND code_pays2=(SELECT code_pays from Pays WHERE nom='France'))
OR ( code_pays2 = code_pays
AND code_pays1=(SELECT code_pays from Pays WHERE nom='France'))
```

20. Écrire une requête SQL permettant de récupérer la longueur de la frontière du pays nommé 'France' sachant que l'on sait que le code de pays nommé 'France' est 'F' et que l'on peut utiliser cette information.

```
SELECT SUM(longueur) FROM Frontieres  
WHERE code_pays1 = 'F' OR code_pays2 = 'F'
```