# EXERCICE X (8 points)

*Cet exercice porte sur la programmation Python (dictionnaire), la programmation orientée objet, les bases de données relationnelles et les requêtes SQL.*

*Cet exercice est composé de 3 parties indépendantes.*

On veut créer une application permettant de stocker et de traiter des informations sur des livres de science-fiction. On désire stocker les informations suivantes :

* l’identifiant du livre (id) ;
* le titre (titre) ;
* le nom de l’auteur (nom\_auteur) ;
* l’année de première publication (ann\_pub) ;
* une note sur 10 (note).

Voici un extrait des informations que l’on cherche à stocker :

| Livres de science fiction | | | | | |
| --- | --- | --- | --- | --- |
| id | titre | auteur | ann\_pub | note |
| 1 | 1984 | Orwell | 1949 | 10 |
| 2 | Dune | Herbert | 1965 | 8 |
| 14 | Fondation | Asimov | 1951 | 9 |
| 4 | Ubik | K.Dick | 1953 | 9 |
| 8 | Blade Runner | K.Dick | 1968 | 8 |
| 7 | Les Robots | Asimov | 1950 | 10 |
| 15 | Ravage | Barjavel | 1943 | 6 |
| 17 | Chroniques martiennes | Bradbury | 1950 | 7 |
| 9 | Dragon déchu | Hamilton | 2003 | 8 |
| 10 | Fahrenheit 451 | Bradbury | 1953 | 8 |

## Partie A

Dans cette première partie, on utilise un dictionnaire Python. On considère le programme suivant :

1 dico\_livres = {  
 'id' : [1, 2, 14, 4, 5, 8, 7, 15, 9, 10],  
 'titre' : ['1984', 'Dune', 'Fondation', 'Ubik', 'Blade Runner',  
 'Les Robots', 'Ravage', 'Chroniques martiennes',  
 'Dragon déchu', 'Fahrenheit 451'],  
 'auteur' : ['Orwell', 'Herbert', 'Asimov',  
 'K.Dick', 'K.Dick', 'Asimov', 'Barjavel',  
 'Bradbury', 'Hamilton', 'Bradbury'],  
 'ann\_pub' : [1949, 1965, 1951, 1953,1968,   
 1950, 1943, 1950, 2003, 1953],  
 'note' : [10, 8, 9, 9, 8, 10, 6, 7, 8, 8]  
 }  
  
2 a = dico\_livres['note']  
3 b = dico\_livres['titre'][2]

1. Déterminer les valeurs des variables a et b après l’exécution de ce programme.

La fonction titre\_livre prend en paramètre un dictionnaire (de même structure que dico\_livres) et un identifiant, et renvoie le titre du livre qui correspond à cet identifiant. Dans le cas où l’identifiant passé en paramètre n’est pas présent dans le dictionnaire, la fonction renvoie None.

1 def titre\_livre(dico, id\_livre):  
2 for i in range(len(dico['id'])):  
3 if dico['id'][i] == ... :  
4 return dico['titre'][...]  
5 return ...

1. Recopier et compléter la fonction titre\_livre.
2. Écrire une fonction note\_maxi qui prend en paramètre un dictionnaire dico (de même structure que dico\_livres) et qui renvoie la note maximale.
3. Écrire une fonction livres\_note qui prend en paramètres un dictionnaire dico (de même structure que dico\_livres) et une note n, et qui renvoie la liste des titres des livres ayant obtenu la note n (on rappelle que t.append(a) permet de rajouter l’élément a à la fin de la liste t).
4. Écrire une fonction livre\_note\_maxi qui prend en paramètre un dictionnaire dico (de même structure que dico\_livres) et qui renvoie la liste des titres des livres ayant obtenu la meilleure note sous la forme d’une liste Python.

## Partie B

Dans cette partie, on utilise le paradigme orientée objet (POO). On propose deux classes : Livre et Bibliotheque.

1 class Livre:  
2 def \_\_init\_\_(self, id\_livre, titre, auteur, ann\_pub, note):  
3 self.id = id\_livre  
4 self.titre = titre  
5 self.auteur = auteur  
6 self.ann\_pub = ann\_pub  
7 self.note = note  
8 def get\_id(self):  
9 return self.id  
10 def get\_titre(self):  
11 return self.titre  
12 def get\_auteur(self):  
13 return self.auteur  
14 def get\_ann\_pub(self):  
15 return self.ann\_pub  
16   
17 class Bibliotheque:  
18 def \_\_init\_\_(self):  
19 self.liste\_livre = []  
20 def ajout\_livre(self, livre):  
21 self.liste\_livre.append(livre)  
22 def titre\_livre(self, id\_livre):  
23 for livre in self.liste\_livre :  
24 if ... == id\_livre :  
25 return ...  
26 return ...

1. Citer un attribut et une méthode de la classe Livre
2. Écrire la méthode get\_note de la classe Livre. Cette méthode devra renvoyer la note d’un livre.
3. Écrire le programme permettant d’ajouter le livre Blade Runner à la fin de la “bibliothèque” en utilisant la classe Livre et la classe Bibliotheque (voir le tableau en début d’exercice).
4. Recopier et compléter la méthode titre\_livre de la classe Bibliotheque. Cette méthode prend en paramètre l’identifiant d’un livre et renvoie le titre du livre si l’identifiant existe, ou None si l’identifiant n’existe pas.

## Partie C

On utilise maintenant une base de données relationnelle. Les commandes nécessaires ont été exécutées afin de créer une table livre. Cette table livre contient toutes les données sur les livres, vous allez donc retrouver la table suivante :

| Table livre | | | | | |
| --- | --- | --- | --- | --- |
| id | titre | auteur | ann\_pub | note |
| 1 | 1984 | Orwell | 1949 | 10 |
| 2 | Dune | Herbert | 1965 | 8 |
| 14 | Fondation | Asimov | 1951 | 9 |
| 4 | Ubik | K.Dick | 1953 | 9 |
| 8 | Blade Runner | K.Dick | 1968 | 8 |
| 7 | Les Robots | Asimov | 1950 | 10 |
| 15 | Ravage | Barjavel | 1943 | 6 |
| 17 | Chroniques martiennes | Bradbury | 1950 | 7 |
| 9 | Dragon déchu | Hamilton | 2003 | 8 |
| 10 | Fahrenheit 451 | Bradbury | 1953 | 8 |

L’attribut id est la clé primaire pour la table livre.

1. Expliquer pourquoi l’attribut auteur ne peut pas être choisi comme clé primaire.
2. Donner le résultat renvoyé par la requête SQL suivante :

SELECT titre  
FROM livre  
WHERE auteur = ‘K.Dick’;

1. Écrire une requête SQL permettant d’obtenir les titres des livres écrits par Asimov publiés après 1950.
2. Écrire une requête SQL permettant de modifier la note du livre Ubik en la passant de 9/10 à 10/10.

On souhaite proposer plus d’informations sur les auteurs des livres. Pour cela, on crée une deuxième table auteur avec les attributs suivants :

* id de type INT ;
* nom de type TEXT ;
* prenom de type TEXT ;
* annee\_naissance de type INT (année de naissance).

| Table auteur | | | | | | |
| --- | --- | --- | --- | --- | --- |
| id | nom |  |  |  |  |
| 1 | Orwell | George | 1903 | |  |
| 2 | Herbert | Franck | 1920 | |  |
| 3 | Asimov | Isaac | 1920 | |  |
| 4 | K.Dick | Philip | 1928 | |  |
| 5 | Bradbury | Ray | 1920 | |  |
| 6 | Barjavel | René | 1911 | |  |
| 7 | Hamilton | Peter | 1960 | |  |

La table livre est aussi modifiée comme suit :

| Table livre | | | | | |
| --- | --- | --- | --- | --- |
| id | titre | id\_auteur | ann\_pub | note |
| 1 | 1984 | 1 | 1949 | 10 |
| 2 | Dune | 2 | 1965 | 8 |
| 14 | Fondation | 3 | 1951 | 9 |
| 4 | Ubik | 4 | 1953 | 9 |
| 8 | Blade Runner | 4 | 1968 | 8 |
| 7 | Les Robots | 3 | 1950 | 10 |
| 15 | Ravage | 6 | 1943 | 6 |
| 17 | Chroniques martiennes | 5 | 1950 | 7 |
| 9 | Dragon déchu | 7 | 2003 | 8 |
| 10 | Fahrenheit 451 | 5 | 1953 | 8 |

1. Expliquer l’intérêt d’utiliser deux tables (livre et auteur) au lieu de regrouper toutes les informations dans une seule table.
2. Expliquer le rôle de l’attribut id\_auteur de la table livres.
3. Écrire une requête SQL qui renvoie le nom et le prénom des auteurs des livres publiés après 1960.
4. Décrire par une phrase en français le résultat de la requête SQL suivante :

SELECT titre  
FROM livres  
JOIN auteurs ON id\_auteur = auteur.id  
WHERE ann\_pub - annee\_naissance < 30;

Un élève décide de créer une application d’annuaire pour sa classe. On pourra retrouver, grâce à cette application, différentes informations sur les élèves de la classe : nom, prénom, date de naissance, numéro de téléphone, adresse email, etc.

1. Expliquer en quoi la réalisation de ce projet pourrait être problématique.