Épreuve écrite de contrôle des acquis
(connaissances et compréhension des notions)

Exercice 3 – Ce sujet traite des bases de données, de la programmation objet et de la programmation générale en Python..

On cherche à programmer le jeu suivant : une série de 9 mots est tirée au hasard, l'un de ces mots est affiché et les autres sont masqués ; en cliquant sur l'un des mots masqués on l'affiche ; si les deux mots affichés sont identiques c'est gagné sinon le mot sur lequel on a cliqué redevient masqué. Plusieurs joueurs tentent leur chance à tour de rôle. Celui qui affichent les deux mots identiques augmente son score de la valeur inscrite à côté du mot à trouver.

Pour programmer ce jeu on construit d'abord une base de données comportant les quatre tables suivantes :

joueur

id *int* PK	**nom** *str*	**score** *int*
25	Alain	15
26	Julie	25
28	Marc	22
29	Esther	14

mot_cache

id *int* PK	**mot** *str*
36	JAUNES
37	JAUGES
39	JATTES
40	JAPPES

mot_a_trouver

id *int* PK	**mot** *str*	**point** *int*
85	JOUETS	0
86	JOULES	35
88	JOUTES	15
89	JONGLE	20

tour

id *int* PK	**id_j** *int* FK	**id_mc** *int* FK
120	32	28
121	33	32
122	35	35
123	34	31

id : identifiant id_j : identifiant_joueur id_mc : identifiant_motcache
PK : clé primaire FK : clé étrangère

Les quatre lignes qui figurent dans ces tables constituent un extrait des différents enregistrements que comporte chacune d'elles.

A chaque tour de jeu on ajoute une ligne dans la table 'tour' qui possède un identifiant (id) unique, l'identifiant d'un joueur (id_j) et l'identifiant du mot caché 'id_mc' sur lequel le joueur a cliqué.

<u>Première partie</u>

Question 1 – On veut afficher la liste des joueurs qui ont un score supérieur à 100. Écrire la requête qui permet d'obtenir cette liste.

Question 2 – On veut ajouter à partir de l'identifiant 154 les mots ADJUGE, ADJURE et AJONCS dans la table 'mot-cache'. Écrire la requête qui permet de réaliser cet ajout en une seule fois.

Question 3 – On veut modifier le nombre de points pour le mot à trouver 'JOUETS'. La nouvelle valeur est 30. Écrire la requête qui permet de réaliser cette mise à jour.

Question 4 – On veut connaître le nom de tous les joueurs qui ont cliqué sur le mot JATTES. Si un même joueur a cliqué plusieurs fois sur ce mot son nom ne doit apparaître qu'une seule fois. Les noms des joueurs seront triés par ordre alphabétique croissant. Écrire la requête qui permet d'obtenir ce résultat.

Question 5 – On veut ajouter dans la table 'tour' une ligne ayant pour 'id' 145, et 27 pour la valeur de l'id_j et 39 pour la valeur de id_mc. Bien que la requête écrite ne comporte aucune erreur de syntaxe, cet ajout est refusé. Proposer une explication.

Deuxième partie

Pour programmer ce jeu on écrit ensuite des classes en langage Python. L'une d'entre elles est la classe 'Joueur'. Voici un extrait de cette classe :

```
1   class Joueur:
2
3       def __init__(self, idt, nom, score=0):
4           self.__idt = idt  # identifiant du joueur
5           self.__nom = nom  # nom du joueur
6           self.__sco = score # score du joueur
```

Question 1 – Écrire une méthode 'getIdtNom' qui renvoie un tuple comportant l'identifiant et le nom du joueur.

Par exemple :

j1 = Joueur(25, 'Alain', 15)

j1.getIdtNom() renvoie (25, 'Alain')

Question 2 – Écrire une méthode 'setScore' qui permet de modifier le score d'un joueur.

Dans le programme principal du jeu on dispose de fonctions qui permettent d'interroger la base de données.

L'une de ces fonctions s'appelle 'selectJoueur'. Elle prend en argument un nombre entier positif et renvoie un dictionnaire dont les clés sont les noms de colonnes de la table 'joueur' et les valeurs, celles de la ligne correspondant à l'identifiant dont la valeur est l'entier passé en argument.

Par exemple selectJoueur(28) renvoie {'id': 22, 'nom':'Marc', 'score': 22}

On affecte à la variable 'j' ce que renvoie la fonction 'selectJoueur(n)' sachant qu'il existe bien une ligne dans la table 'joueur' pour laquelle 'id' vaut 'n'.

Question 3 – Écrire l'instruction qui permet d'instancier la classe Joueur à partir de ce qui sera affecté à j.

On crée également une classe 'Mot' dont voici un extrait :

```
20  from random import randint
21  from random import shuffle
22
23  class Mot:
24
25      pts = [p for p in range(5, 55, 5)]
26
27      def __init__(self, liste):
28          self.mots_caches = liste  # liste de 9 mots
29          self.mot_a_trouver = liste[randint(0,100) % 9]
30          self.points = Mot.pts[randint(0,100) % 10]
```

Dans la console on écrit les instructions suivantes :

```
In [2]: LM = Mot(['RAJOUT', 'REJETE', 'REJOUE', 'REJOUI', 'REJUGE',
   ...:           'SEJOUR', 'SUJETS', 'SURJET', 'TRAJET'])

In [3]: LM.mot_a_trouver
Out[3]: 'REJOUE'

In [4]: LM.points
Out[4]: 5
```

Question 4 – Que va-t-il se passer si on exécute une deuxième fois l'instruction LM = Mot(['RAJOUT', 'REJETE', 'REJOUE', 'REJOUI', 'REJUGE', 'SEJOUR', 'SUJETS', 'SURJET', 'TRAJET']) ?

On veut ajouter une méthode 'marque' qui prend en argument un mot caché sur lequel un joueur a cliqué et qui renvoie la valeur '0' (zéro) si ce mot est différent du mot à trouver ou qui renvoie le nombre de points attribué au mot à trouver si les deux mots sont identiques.

Par exemple pour la situation affichée juste avant, LM.marque('SEJOUR') renvoie 0, et LM.marque('REJOUE') renvoie 5.

Question 5 - Écrire cette méthode.

On veut ajouter une méthode 'grilleJeu' qui renvoie une grille de trois lignes et trois colonnes contenant les mots cachés.

Par exemple pour la situation affichée juste avant, LM.grilleJeu() renvoie

```
In [5]: LM.grilleJeu()
Out[5]:
[['REJOUI', 'REJUGE', 'SUJETS'],
 ['REJETE', 'SEJOUR', 'SURJET'],
 ['RAJOUT', 'REJOUE', 'TRAJET']]
```

Voici une partie du code écrit en Python de cette méthode.

```
47      def grilleJeu(self):
48          copie = self.mots_caches.copy()
49          shuffle(copie)
50          grille = [...]
51          for ligne in range(0,3):
52              for colonne in range (0,3):
53                  grille[ligne][colonne] = copie[...]
54          return grille
```

La ligne 48 permet de réaliser une copie de la liste des mots cachés et la ligne 49 permet de les mélanger.

Question 6 – Recopier et compléter la ligne 50 qui permet de construire par compréhension la liste appelée 'grille' selon le schéma suivant : trois sous-listes de trois chaînes de caractères vides chacune.

```
In [7]: grille
Out[7]: [['', '', ''], ['', '', ''], ['', '', '']]
```

Question 7 – Recopier et compléter compléter la ligne 53 qui permet de remplir 'grille' avec les mots de 'copie'.