Centres étrangers – 2026 – sujet1

Exercice 3 (8 points)

Cet exercice porte sur la programmation orientée objet, la récursivité et les bases de données relationnelles.

Le champ de mines du démineur est représenté par une grille.

Chaque case de cette grille peut cacher une mine, ou être vide.

Le déroulement du jeu :

- Si on choisit sur une case libre qui a au moins une mine dans une case voisine, un chiffre apparaît. Ce chiffre indique combien de mines se trouvent dans les cases voisines. Une case a au maximum 8 cases voisines.
- Si on choisit une case vide dont toutes les cases voisines sont vides, la case s'affiche vide. Ensuite, le jeu dévoile automatiquement toutes les cases vides voisines, et ainsi de suite, jusqu'à ce qu'il rencontre des cases avec des chiffres.
- La fin de la partie : Si on choisit sur une mine, la partie est perdue. Si on découvre toutes les cases libres, la partie est gagnée.

	0	1	2	3	4	5	6
0	1	-1	2	1	2	1	1
1	1	2	3	-1	2	-1	1
2	1	2	-1	2	2	1	1
3	2	-1	2	2	1	1	0
4	-1	2	1	1	-1	2	1
5	1	1	0	1	1	2	-1

Figure 1. Une grille 6x7 du démineur avec les informations du jeu.

La figure 1 représente une grille du jeu du démineur de 6 lignes : hauteur et de 7 colonnes : largeur, où l'on a dévoilé la position des 8 mines de la grille représentées par l'entier relatif -1 ainsi que les informations utiles au joueur qui lui seront dévoilées au fur et à mesure du jeu.

- La case de coordonnées (1, 2) contient le chiffre 3 car elle a trois de ses cases voisines qui contiennent une mine : les cases (0, 1), (2, 2) et (1, 3).
- La case (3, 6) contient le chiffre 0 car aucune case qui lui est voisine ne contient de mine.

Partie A : La classe Demineur

On propose de programmer le jeu du démineur à l'aide du langage de programmation Python. On utilise alors le paradigme de programmation orienté objet.
On donne un extrait de la classe `Demineur` suivante.

```
1       class Demineur :
2           def __init__(self, hauteur, largeur, pourcentage_mines):
3               '''
4               Pour un pourcentage de 15,6 on donnera au paramètre
5               pourcentage_mines la valeur de 0.156.
6               '''
7               assert ..., 'Le pourcentage de mines doit être compris
                entre 10% et 30%.'
8               ...hauteur = hauteur
9               ...largeur = largeur
10              ...pourcentage_mines = pourcentage_mines
```

1. Recopier et compléter la ligne 7 de l'assertion pour que le pourcentage de mines respecte la précondition recommandée.

2. Recopier et compléter les lignes 8, 9 et 10 et compléter les pointillés avec le code manquant.

On donne les pourcentages suivants, conçus pour offrir un bon équilibre entre challenge et jouabilité.

- Débutant : Grille de 8x8 avec ce qui représente 10 mines, environ 15,6% de mines.
- Intermédiaire : Grille de 16x16 cases avec 40 mines, ce qui représente environ 15,6% de mines.
- Expert : Grille de 30x16 cases avec 99 mines, ce qui représente environ 20,6% de mines.

3. Créer une instance `demineur_intermediaire`, de la classe `Demineur` en respectant les recommandations pour le niveau de jeu intermédiaire pour le jeu du démineur.

PARTIE B : CRÉATION DE LA GRILLE DU DÉMINEUR

La grille du jeu du démineur peut être modélisée par un objet Python de type `list`. Par exemple une grille vide d'un démineur de hauteur 3 et de largeur 5 peut-être représentée par l'objet Python suivant :

```
grille_vide = [[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0,
0]]
```

4. Recopier et compléter la méthode `grille_demineur_vide` de la classe `Demineur` qui permet d'initialiser une grille vide du jeu.

```
def grille_demineur_vide(self):
    return [[0 for _ in range(...)] for _ in range(...)]
```

On ajoute dans le constructeur de la classe `Demineur` l'attribut `grille_demineur` qui va permettre d'initialiser une grille vide.

```
self.grille_demineur = self.grille_demineur_vide()
```

Les mines doivent être placées de façon aléatoire dans la grille et seront représentées par l'entier relatif -1.

On donne alors une méthode `placer_mines`, incomplète, de la classe `Demineur`.

```
1     def placer_mines(self):
2         '''
3         Cette fonction doit placer de façon aléatoire les mines dans la
4         grille.
5                 Une mine sera représentée par le nombre -1.
6                 La méthode permet de mettre à jour l'attribut
          grille_demineur
7         '''
8         compteur_mines = 0 # Nombre de mines placées dans la grille.
9         # nombre_bombes contient le nombre de mines à placer dans la
          grille.
10        nombre_bombes = self.largeur*self.hauteur*self.pourcentage_mines
11
12        while  compteur_mines < nombre_bombes:
13            ligne = randint(0, self.hauteur - 1)
14            colonne = randint(0, ...)
15            if self.grille_demineur[...][...] == 0:
16                self.grille_demineur[...][...] = -1

17                compteur_mines  = ...
```

La documentation Python donne les informations suivantes :

```
>>> from random import randint
>>> help(randint)
Help on method randint in module random:

randint(a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
```

5. Recopier et compléter les lignes 14, 15, 16 et 17 de la méthode `placer_mines` de la classe `demineur`.

On dispose des deux méthodes suivantes.

```
    def voisines(self, coordonnees_case):
        '''
        La méthode voisines renvoie les coordonnées des cases voisines de
        la case de la grille dont les coordonnées sont données par le
        tuple coordonnees_case passé en paramètre.
        Cette méthode renvoie une liste de tuple.

        >>> demineur_1 = Demineur(3, 5,0.20)
        >>> demineur_1.voisines((0, 0))
        [(1, 1), (1, 0), (0, 1)]
        >>> demineur_1.voisines((1, 2))
        [(0, 1), (0, 3), (0, 2), (2, 1), (2, 3), (2, 2), (1,
       1), (1, 3)]
        '''
```

```
def nombre_voisines_avec_mines(self, coordonnees_case):
        '''
         La méthode nombre_voisine_avec_mines renvoie le nombre de cases
         voisines contenant une mine
         à la case dont les coordonnées sont passés en
         paramètre(coordonnees_case) à la fonction.
        '''
```

6. Écrire le code Python de la méthode `nombre_voisines_avec_mines`. On ne demande pas d'écrire le code de la méthode `voisines`.

7. Écrire une méthode `generer_demineur` qui permet de mettre à jour l'attribut `grille_demineur` pour que chaque cellule ne contenant pas de mine, contienne le nombre de mines qui lui sont voisines.

   Par exemple :

```
>>> demineur_nsi = demineur(6, 6, 0.278)
>>> demineur_nsi.generer_demineur()
 >>> demineur_nsi.grille_demineur
    [[2, -1, 2, 1, 2, 1], [2, -1, 4, -1, 2, -1], [2, 3, -1, 2, 2, 1],
    [2, -1, 2, 2, 1, 1], [-1, 3, 2, 2, -1, 2], [1, 2, 1, 2, 2, -1]]
```

	0	1	2	3	4	5
0	2	-1	2	1	2	1
1	2	-1	4	-1	2	-1
2	2	3	-1	2	2	1
3	2	-1	2	2	1	1
4	-1	3	2	2	-1	2
5	1	2	-1	2	2	-1

Figure 2. Représentation de la grille 6x6 du démineur de l'exemple précédent.

PARTIE C : L'INTERFACE UTILISATEUR DU JEU DU DÉMINEUR

Les parties précédentes ont permis la création d'une grille du démineur. L'utilisateur se verra afficher une grille où les informations de la grille ne seront pas apparentes. Le joueur fait le choix d'une case et en fonction de ce choix des cases seront découvertes.

- Si la case contient une mine toutes les cases sont alors dévoilées.
- Dans le cas contraire une ou plusieurs cases seront dévoilées.

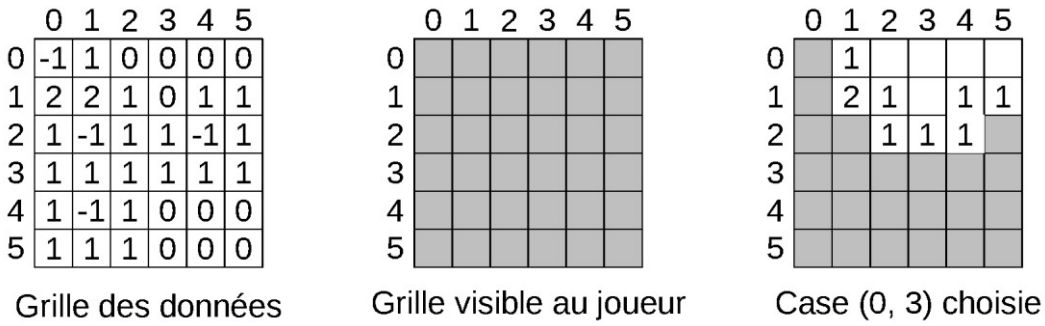

Figure 3. Gérer le visuel du joueur.

Sur la figure 3 on peut lire:

- sur la grille de gauche, les informations du démineur.
- sur la grille au centre, le visuel présenté au joueur, aucune information n'est visible au départ du jeu.
- sur la grille de droite, les informations révélées quand le joueur a choisi la case de coordonnées (0, 3).

On propose de modéliser la visibilité des cellules pour l'utilisateur de la façon suivante.

	0	1	2	3	4	5
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
5	False	False	False	False	False	False

Initialisation de la grille de visibilité

	0	1	2	3	4	5
0	False	True	True	True	True	True
1	False	True	True	True	True	True
2	False	False	True	True	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
5	False	False	False	False	False	False

Mise à jour de la grille de visibilité

Figure 4. Évolution de la grille de visibilité.

Une grille est utilisée pour connaître l'état de visibilité de l'information des cases.

- `False` : L'information de la case n'est pas visible par l'utilisateur.
- `True` : L'information de la case est visible à l'utilisateur.

Si on considère la figure 4, le constructeur de la classe `Demineur` initialisera l'attribut `grille_visibilite` avec `[[False for_ in range(6)] for _ in range(6)]`.

8. Écrire une méthode `visibilite` de la classe `Demineur` qui permettrait de mettre à jour l'attribut `grille_visibilite` en fonction du choix de l'utilisateur d'une case du démineur. On pourra utiliser la récursivité.

PARTIE D : JOUER EN LIGNE AU DÉMINEUR

Dans cette partie, on pourra utiliser les clauses du langage SQL pour :

- construire des requêtes d'interrogation à l'aide de `SELECT`, `FROM`, `WHERE` (avec les opérateurs logiques `AND`, `OR`) et `JOIN ... ON` ;
- construire des requêtes d'insertion et de mise à jour à l'aide de `UPDATE`, `INSERT` et `DELETE`
- affiner les recherches à l'aide de `DISTINCT` et `ORDER BY`.

On donne la possibilité aux joueurs d'enregistrer les scores en ligne pour le jeu du démineur.

On dispose d'une base de données relationnelles avec les tables suivantes.

- `Joueur` : enregistrement du pseudo et du mot de passe de connexion du joueur.

Joueur

id_joueur	pseudo	mot_de_passe
1	Grimdal	EgxGB6a3bRinllon
2	Raptor	J17NtfMS3Dudjjln
3	PetiteFée	UuukBSvj01VrGoGD
4	Kirna	7NcDFPNl0Xy1MEPb

- `Meilleur_score` : enregistrement du meilleur score d'un joueur pour chaque niveau joué.

Meilleur_score			
joueur	niveau	score	temps
1	facile	1200	72
2	expert	1196	366
3	intermédiaire	997	230
2	intermédiaire	997	200
4	expert	1097	400

- `Demineur` : enregistrement des caractéristiques de chaque niveau du démineur.

Demineur		
niveau	dimension	pourcentage mines
facile	8x8	15.6
intermédiaire	16x16	15.6
expert	30x16	20.6

9. Donner les clés étrangères de la table `Meilleur_score` en précisant pour chacune d'elle à quelle clé primaire elle fait référence.

Une requête envoyée à la base de données a renvoyé la réponse suivante.

niveau	score
expert	1196
intermédiaire	997

10. Donner une requête qui permet d'obtenir les informations du tableau précédent.

Kirna change son mot de passe par cGhhxDE4 en place de 7NcDFPNl0Xy1MEPb.

11. Écrire la requête qui permet de mettre à jour la table `Joueur`.

On donne la requête suivante.

```
SELECT pseudo FROM Joueur
JOIN Meilleur_score
ON Joueur.id_joueur = Meilleur_score.joueur
WHERE niveau = 'expert' AND score > 1000 AND temps < 400;
```

12. Donner le résultat de cette requête.

Une erreur a été faite à la création de la table `Demineur`. Le niveau facile doit être désigné par le mot débutant.

Écrire les requêtes qui permettent de corriger la base de données