IFT 1015 – Programmation I

TP 2

- À faire en groupe de **deux** étudiants.
- Remise : Le 28.04.2024 à **23:59** au plus tard

1. Objectifs du TP2

Dans ce projet, vous aurez l'occasion de pratiquer les concepts suivants:

- Boucles
- Tableaux
- Fonctions
- Décomposition fonctionnelle
- Traitement d'événements
- Programmation web

Le code que vous devez écrire (fichier « **`tp2.py`** ») implante un jeu qui s'exécute dans l'environnement du navigateur web. Une bonne partie du jeu peut se développer avec **`codeBoot`**. Cependant, il faut comprendre qu'on vise à déployer le programme comme une application web standard (les détails sont expliquées ci-dessous). Vous pouvez utiliser le code qui a été montré dans le cours, mais vous ne devez pas utiliser de code provenant d'ailleurs, que ce soit du web ou d'une autre personne.

2. Description générale

Ce travail pratique consiste à développer un programme web pour jouer au jeu « **Cueillette de sous »**. C'est un jeu d'un seul joueur. Le jeu consiste à trouver les pièces cachées sur une carte quadrillée. La carte indique les emplacements de sous indirectement. Sur la carte, il y a des cases avec les nombres. Ces cases sont vides. En revanche, la valeur numérique dans ces cases indique le nombre de pièces se trouvant dans les cases voisines (pour un maximum de huit), qu'elles se touchent par un côté ou par un coin. Il n'y a plus d'une pièce par case.
Le joueur gagne lorsque toutes les pièces sont retrouvées avec maximum 3 ouvertures erronées.

3. Éléments du jeu

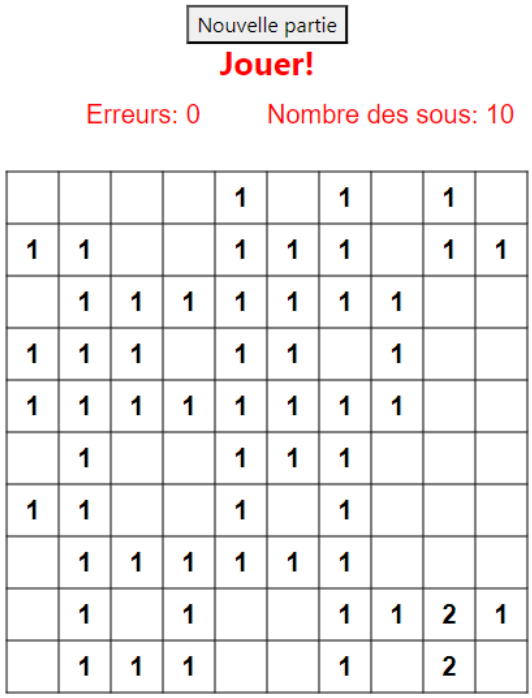
La carte du jeu est une grille de 10 rangées et 10 colonnes avec les valeurs placées dans certaines cases. Au-dessus de la grille, il y a un bouton de démarrage d'une nouvelle partie, un espace de message, un compteur d'ouverture erronées **`Erreurs`** et un compteur des sous cachés.

Étape Initialisation :

1. Générer un nombre des pièces cachées, ce nombre doit varier de 15 à 20 % du nombre total des cases dans la grille de jeu (100 pour ce jeu).
2. Générer les indices de placement des sous. Il faudra respecter une règle de placement : si une pièce est placée, une nouvelle pièce ne peut pas être cachée dans ses cases voisines.
3. Après avoir placé des sous, il faut calculer les valeurs dans les cases voisines des pièces de monnaie cachées.

4. Former le code html qui représente le contenu de la grille formée. Dans les cases avec des sous, il faudra mettre les images avec attribut **`hidden="hidden"`**. Dans les cases voisines des pièces cachées, il faudra afficher les nombres indicateurs.

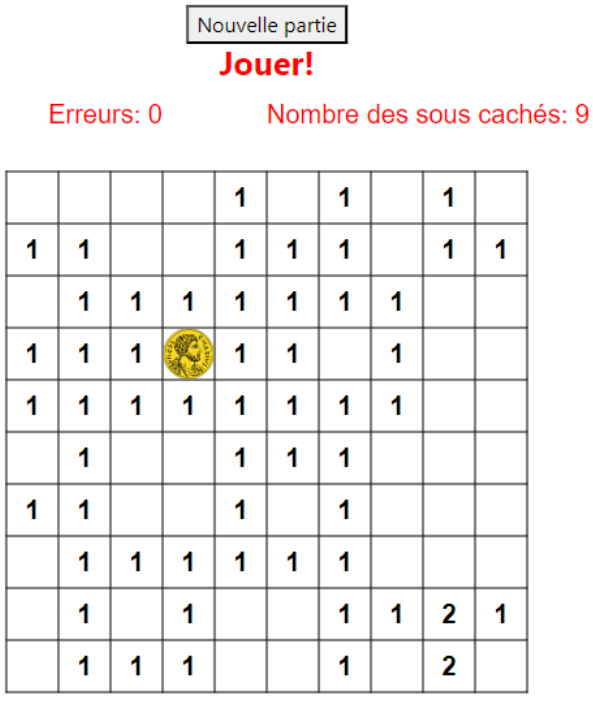
   Voici à quoi ressemble la fenêtre au début d'une partie :



				1		1		1	
1	1			1	1	1		1	1
	1	1	1	1	1	1	1		
1	1	1		1	1		1		
1	1	1	1	1	1	1	1		
	1			1	1	1			
1	1			1		1			
	1	1	1	1	1	1			
	1		1			1	1	2	1
	1	1	1			1		2	

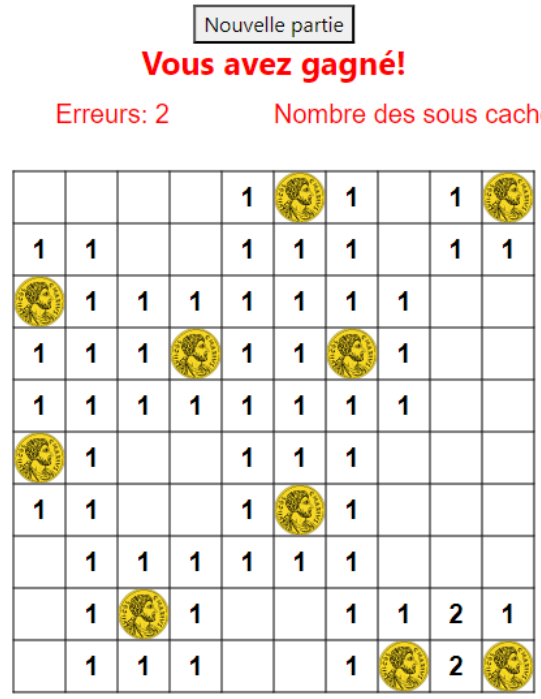
Étape Jeu :

1. Si on trouve correctement un sous caché, il faudra afficher l'image de monnai dans la case et decrementé le compteur des pièces cachées.

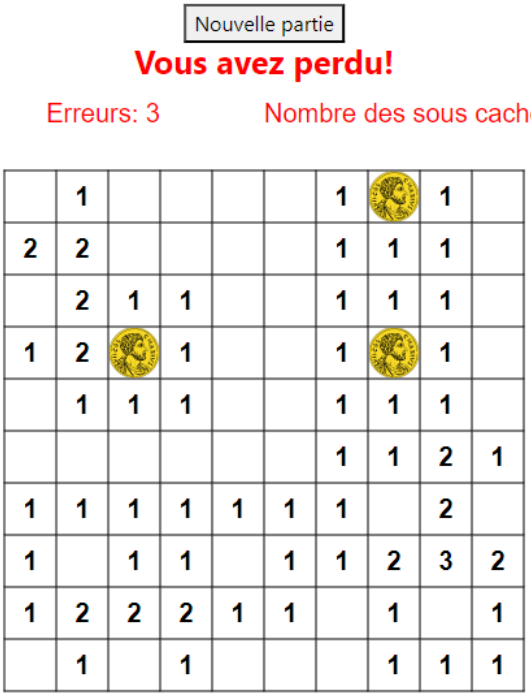


				1		1		1	
1	1			1	1	1		1	1
	1	1	1	1	1	1	1		
1	1	1		1	1		1		
1	1	1	1	1	1	1	1		
	1			1	1	1			
1	1			1		1			
	1	1	1	1	1	1			
	1		1			1	1	2	1
	1	1	1			1		2	

2. Si on trouve tous les sous avec le nombre d'ouvertures erronées inferieur à 3 – vous gagnez la partie. Il faudra afficher le message correspondant, preserver l'image du jeu 10 secondes et redemarer le jeu.



3. Si le nombre d'ouvertures erronées depasse 3, vous perdez la partie. Dans ce cas, il faudra afficher le message corespondant, preserver l'image 10 secondes et redemarrer le jeu.



5. **Détails**

Pour vous démarrer dans la bonne direction, vous trouverez sur Studium le **`fichier serveur-web.py`** et le fichier **`documents.zip`**. Le fichier **`documents.zip`** contient le répertoire avec les documents qui

doivent être utilisés dans votre **tp**. Le sous répertoire **symboles** contient l'image du symbole en format **SVG**, « Scalable Vector Graphics ». Il faut sauvegarder ces fichiers dans votre système de fichiers local.

Pour exécuter le code, il faudra démarrer le serveur avec la commande **python3 serveur-web.py** (système d'exploitation Linux, MacOS), **python serveur-web.py** (Windows) et lancer la requête **http://localhost:8000/tp2.html** dans votre navigateur. Les fichiers **tp2.html**, **tp2.py**, **tp2.css**, **codeboot.bundle.js**, **codeboot.bundle.css** et **index.html** sont également présents dans le répertoire **documents**. Vous ne devez pas changer cette organisation de fichiers ni changer le contenu des fichiers sauf **tp2.py** et **tp2.css** que vous devez compléter avec votre code. Le fichier **tp2.html** contient des directives pour inclure les fichiers **codeboot.bundle.js**, **codeboot.bundle.css**, le code Python **tp2.py** et le fichier **css** avec styles. Le corps du document est le suivant :

```
<body onload="init()">
<div id="main"></div>
</body>
```

Le traiteur d'événement **onload** du corps fait un appel à la fonction **init** définie dans **tp2.py** pour démarrer l'exécution du code Python au moment du chargement du fichier **tp2.html**. La fonction **init** doit créer le contenu HTML qui sera mis dans l'élément ** **. On pourrait, par exemple, définir la fonction **init** comme suit pour afficher une grille 2 x 3 centrée et un bouton Nouvelle partie avec la disposition par default :

```
def init():
    main = document.querySelector("#main")
    main.innerHTML = """
      <button onclick="init()">Nouvelle partie</button>
      <div id="jeu" class="centered">
      <table>
          <tr>
            <td id="case0"></td>
            <td id="case1"></td>
            <td id="case2"></td>
          </tr>
          <tr>
            <td id="case6"></td>
            <td id="case7"></td>
            <td id="case8"></td>
          </tr>
        </table>
      </div>"""
```

Évidemment, pour une grille de grande taille, il serait mieux de créer le HTML avec des boucles pour éviter des répétitions de code. Pour changer les attributs de style de l'élément, on peut utiliser les méthodes **setAttribute/removeAttribute.**

Le développement du programme peut se faire de cette manière :
Éditer directement le fichier **tp2.py** avec un éditeur de code et utiliser le navigateur pour rafraichir la page **http://localhost:8000/tp2.html** (ce qui va exécuter votre programme **tp2.py** à nouveau). Un clic avec le bouton de droite vous permet d'afficher dans une fenêtre flottante l'état du programme, ce qui est utile s'il y a bogues ou des appels à **breakpoint()** dans votre code.

Note :

1. Pour le système d'exploitation Windows, il se peut qu'il y ait un problème avec l'utilisation du répertoire avec le nom **documents**. Utilisez le nom **documents_tp2** ou autre variante de ce nom en faisant l'ajustement du nom de répertoire dans le fichier **serveur-web.py** :

```
def obtenirDocument(path):
    #print(path)
    return readFile('documents_tp2' + path)
```

2. Explications de la classe css **centered**

```
.centered {
  position: absolute;
  top: 60%;
  left: 50%;
  transform: translate(-50%, -50%);
}
```

La classe css **centered** définit les propriétés suivantes :
position: absolute - position absolue
top: 60% - affecte la position verticale d'élément positionné ;
left: 50%; - position à gauche ;

Les attributs **top**, **left** placent l'élément à la position en % par rapport à la taille de l'élément conteneur ;

transform: translate(-50%, -50%); - déplace un élément en permettant de le centrer par rapport d'élément conteneur ;

6. **Évaluation**

Ce travail compte pour 15 points dans la note finale du cours. Indiquez vos noms clairement dans les commentaires au début de votre code.
Vous devez remettre deux fichiers tp2.py et tp2.css. Chaque fonction devrait avoir un bref commentaire pour dire ce qu'elle fait, il devrait y avoir des lignes blanches pour que le code ne soit pas trop dense, les identificateurs doivent être bien choisis pour être compréhensibles et respecter le standard CamelCase. Il faudra écrire les tests unitaires pour les fonctions de calcul.