1) Afficher toutes les parties du jeu qui sont statiques (qui ne changent pas d'un état de jeu à l'autre):
    a) l'image du joueur;
    b) l'image de l'ordinateur;
    c) les carrés qui vont contenir les animations des attaques;
    d) le titre du jeu;
    e) le pointage du joueur et de l'ordinateur.
2) Dans la méthode `on_key_press`:
    a) C'est ici que le joueur fait changer l'état du jeu avec la touche 'espace'
        i) De l'état `NOT_STARTED`, on passe à `ROUND_ACTIVE`;
        ii) De l'état `ROUND_DONE`, on revient à `ROUND_ACTIVE`.
            (1) N'oubliez pas de remettre à faux ou 0 toutes variables qui sert à la validation;
        iii) De l'état `GAME_OVER`, on revient à `ROUND_ACTIVE`:
            (1) Lors de ce changement, c'est comme si on démarrait le jeu pour la première fois. Il faut donc s'assurer de remettre tous les "flags" à `False` ainsi que le pointage à zéro.
3) Dans la méthode `on_mouse_press`:
    a) Il faut valider si le joueur clique sur un des Sprites. Il faut utiliser la méthode `collides_with_point`. Par exemple, si le Sprite de la roche se nomme 'rock', on effectue la validation suivante:

```
if self.rock.collides_with_point((x, y)):
      pass
```

    b) Comment faire en sorte que notre méthode `on_update` puisse connaître le type d'attaque qui a été sélectionné dans la méthode `on_mouse_press`?
        i) Il faut avoir une variable dans la classe MyGame pour l'indiquer. On peut logiquement la nommer `player_attack_type`.
4) Ensuite, il faut créer le Enum `AttackType` dans le fichier `attack_animation`.
    a) Voir dans le document théorie TP6 dans la section animation.

b) Avec l'exemple ci-haut, on assigne la variable comme ça:

```
self.player_attack_type = AttackType.ROCK
```

c) Maintenant, il faut aussi avoir une seconde variable pour indiquer à notre jeu que le joueur à cliqué sur un Sprite d'attaque. C'est ce que je nomme un 'flag'. C'est une variable de type booléenne. Il y a donc deux seules possibilités: `True` ou `False`. Nommez-la de façon logique pour que ça représente l'action faite. Il faut y assigner une valeur True lorsque la méthode `collides_with_point()` retourne vraie.

5) Valider la victoire:
   a) C'est dans la méthode `on_update` que la logique de votre programme s'effectue.
   b) Avant de procéder, il faut s'assurer que l'état du jeu est `ROUND_ACTIVE` et que le "flag" créer au point 4-c soit vrai.
   c) Ensuite, il faut générer l'attaque de l'ordinateur.
      i) Il faut donc générer un chiffre aléatoire entre 0 et 2. Ensuite passer ce chiffre en paramètre du constructeur de la classe `AttackType`.

```
pc_attack = randint(0, 2)
if pc_attack == 0:
    self.computer_attack_type = AttackType.ROCK
elif pc_attack == 1:
    self.computer_attack_type = AttackType.PAPER
else:
    self.computer_attack_type = AttackType.SCISSORS
```

   d) Une fois l'attaque de l'ordinateur générée, il suffit de valider qui gagne la ronde. Vous connaissez les règles du jeu alors je vous laisse faire cette partie.
   e) Il faut aussi s'assurer que le pointage soit modifié en lien avec l'étape 5-d.
   f) Il faut aussi regarder si un des deux a obtenu un pointage de 3. Si c'est le cas, il faut passer à l'état `GAME_OVER`.

6) Dans la méthode `on_draw`, il faut s'assurer d'afficher toutes les informations pertinentes en lien avec l'état du jeu courant.

7) Faire la dernière partie du document théorique pour l'animation des sprites.
8) **En dernier lieu, assurez-vous d'avoir écrit votre entête de fichier ainsi que les commentaires sous forme de "docstring" pour toutes les méthodes et classes de votre jeu.**