**Projet Final:**

**Rapport de Sprint #1**

Michael Morin

Département d’Informatique et de Mathématique

Université du Québec à Chicoutimi

8INF892 – Apprentissage Profond

Dr. Kevin Bouchard

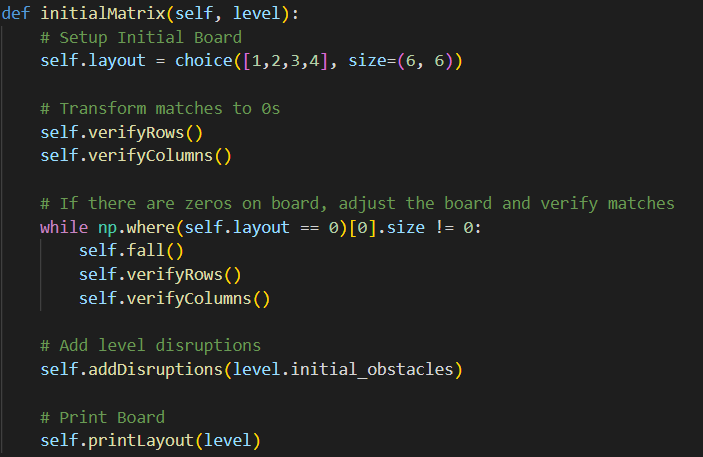
24 Mars 2023

**Rapport de Sprint #1**

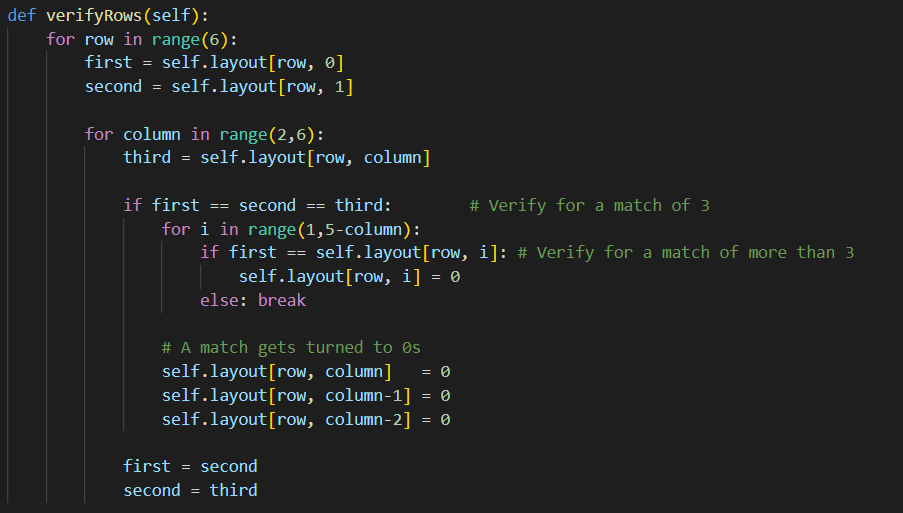
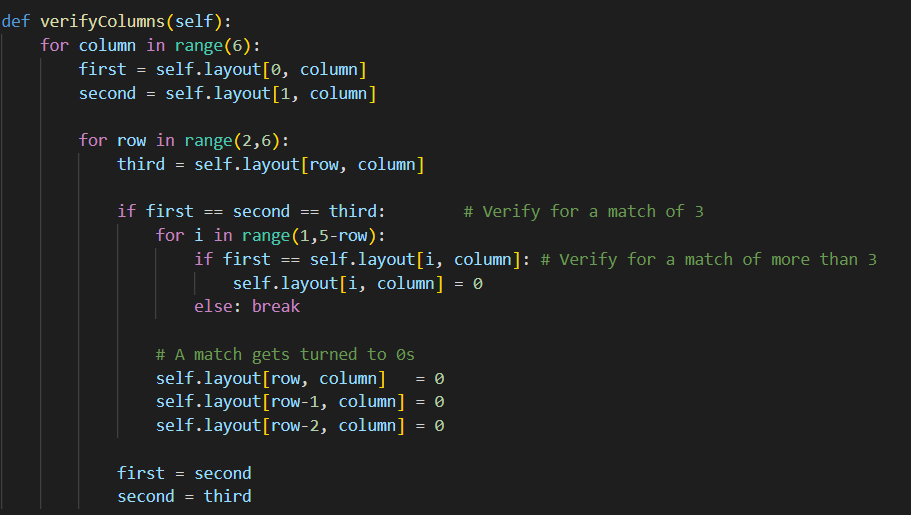
**Problématique à adresser**

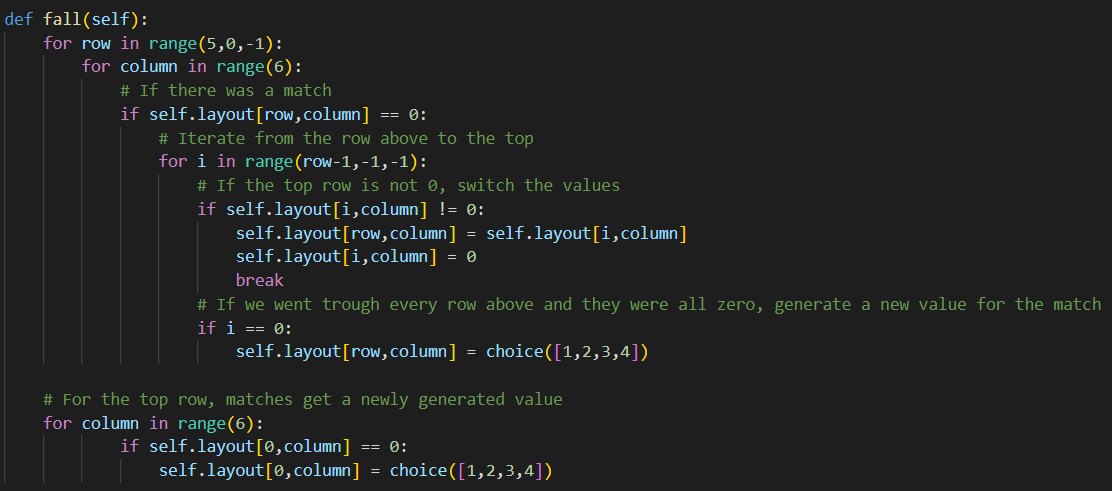
Le projet final consiste à utiliser un agent apprenant par renforcement pour compléter de manière efficace les différents tableaux de *Pokémon Shuffle*. Pour ce faire, une mini version du jeu doit être construite, puis le tableau de jeux est acheminé à un réseau de neurones pour transformer le tableau graphique en tableau de chiffre qui sera utiliser pas l’agent apprenant pour trouver la meilleure solution. Pour le premier sprint, les efforts ont donc été déployer dans la construction de la mini version du jeu. Pour l’ensemble du projet, le langage de programmation Python sera utilisé. En plus de ce rapport, le code source sera joint si possible.

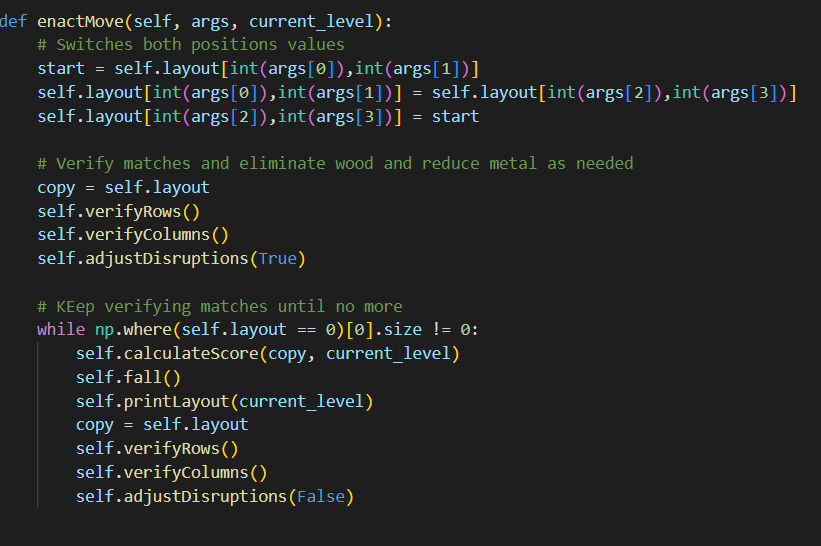
**Travail réalisé**

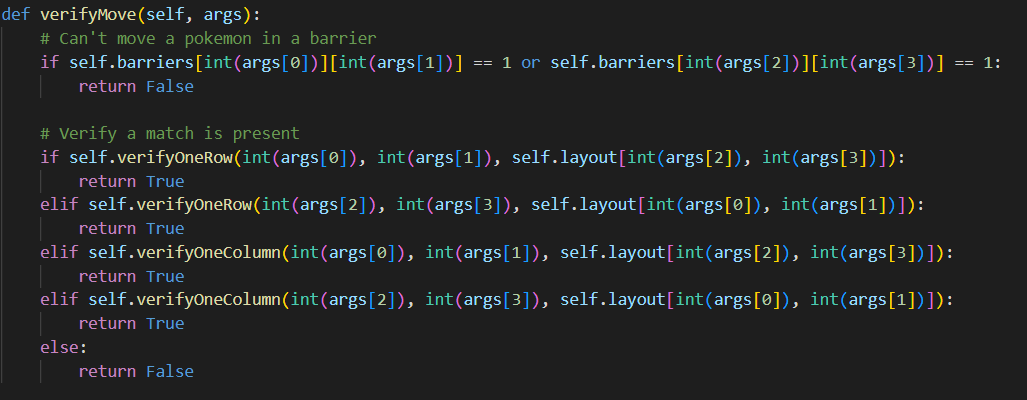
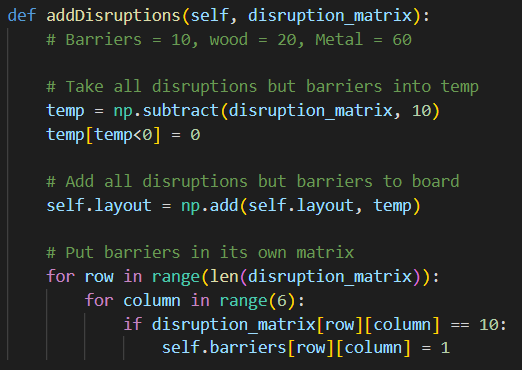
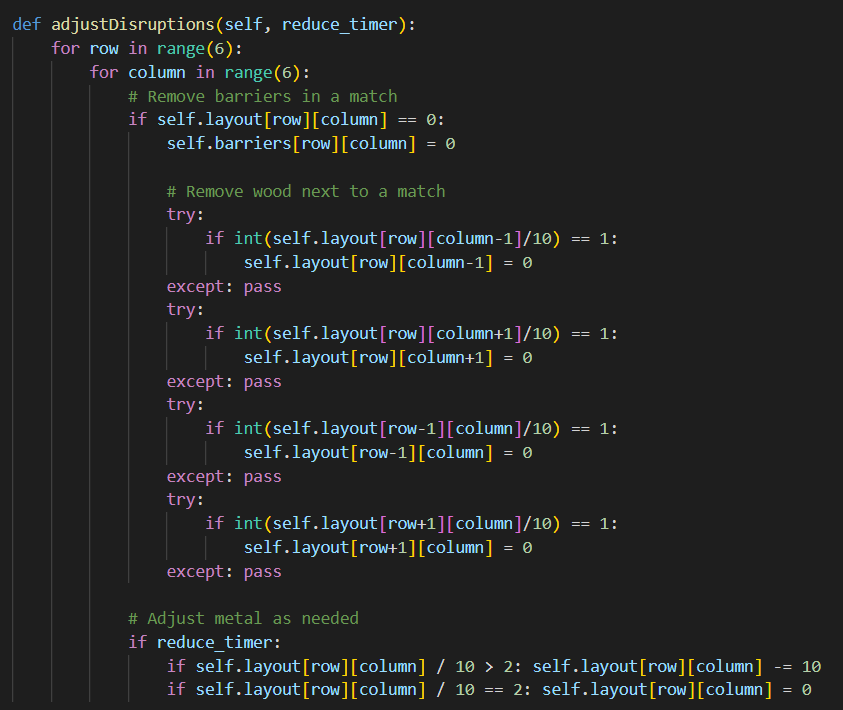
 *Pokémon Shuffle* est un jeu similaire au style de jeu match-3 dans le sens ou le minimum de trois icones identiques doivent être alignés chaque tour. Comme tous les opus de cette série, les créatures sont omniprésentes, que ce soit avec les icones ou dans les niveaux. Chaque niveau est donc représenté par un espèce de Pokémon ayant une limite de points de vie et un nombre maximal de tours. Chaque tour, l’utilisateur est confronté à une matrice d’icones de six par six et il doit choisir deux icones qui seront échangés dans le but d’aligner les Pokémons identique.

Pour notre jeu, la première étape fût de préparer une matrice n’ayant pas 3 icones identiques sur la même ligne ou colonne.

Pour verifier l’absence d’un match-3, les fonctions verifyrows et verify columns ont été utilisé. Puisqu’elle change les matchs en zeros, une boucle utilise ces fonctions et la fonction fall pour obtenir la matrice voulu.

 En fait, à chaque fois qu’un match-3 est fait par l’utilisateur ou par accident, la fonction fall est appelé pour imiter la gravité sur le tableau.

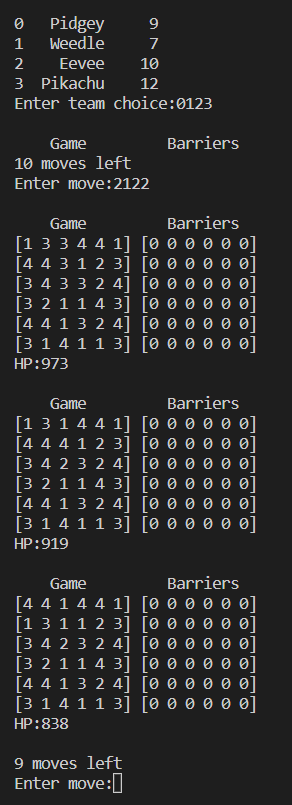
Pour l’utilisateur, la première étape est de choisir sont équipe. Initialement, il doit choisir quatre pokémons et il n’y en a que quatre, mais chaque niveau passé en débloque un. Suivant ce choix, le tableau est affiché et l’utilisateur choisit de déplacer deux icones. Le logiciel vérifie la validité du mouvement avant de l’effectuer.

En ce qui concerne les fonctions verifyOneRow et verifyOneColumn, elles ne font qu’itérer sur la ligne en vérifiant si trois icones identiques sont côte-à-côte. Cependant, cette image fait aussi mention de barrières, ce qui nous amène à une particularité de ce jeu par rapport à d’autre: les obstacles. Au début du jeu et après un nombre de mouvement pour chaque Pokémon, des obstacles peuvent faire leur apparition sur le tableau. Trois variétés existent, soit les barrières qui empêche une icone de bouger et disparait lors d’un match-3, soit le bois qui disparait lors d’un match-3 près de lui, soit le métal qui disparait après 4 mouvements. Dans notre programme, la fonction addDisruption permet d’ajouter les obstacles alors que adjustDisruption permet de les enlevés.

Avec ces fonctions, la plupart des fonctionnalités sont couvertes. Cependant, plusieurs autres composés sont utilisés mais ils ne seront pas discutés ici.

**Bilan et Prochain Sprint**

Au final, ce sprint n’a pas touché à l’apprentissage profond, mais l’utilisation des matrices, Numpy and Pandas sont, pour moi, de nouveaux outils de programmation. De plus, cette étape était primordiale pour l’entrainement du réseau de neurones et l’apprentissage de l’agent puisque ce jeu n’a pas de version pour l’ordinateur. Pour confirmer le bon fonctionnement du programme, voici une trace d’exécution :



Lors du prochain sprint, le réseau de neurones sera implémenté et le programme verra une interface graphique. Cette interface permettra de transformer la matrice du tableau en tableau graphique. De plus, les icones seront indirectement transféré au réseau de neurones pour qu’il retrouve la matrice numérique initiale. Plus précisément, le programme va envoyer une version plus ou moins complète de l’icône pour le déchiffrage.