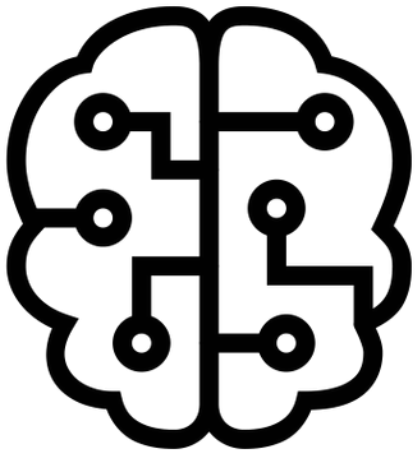
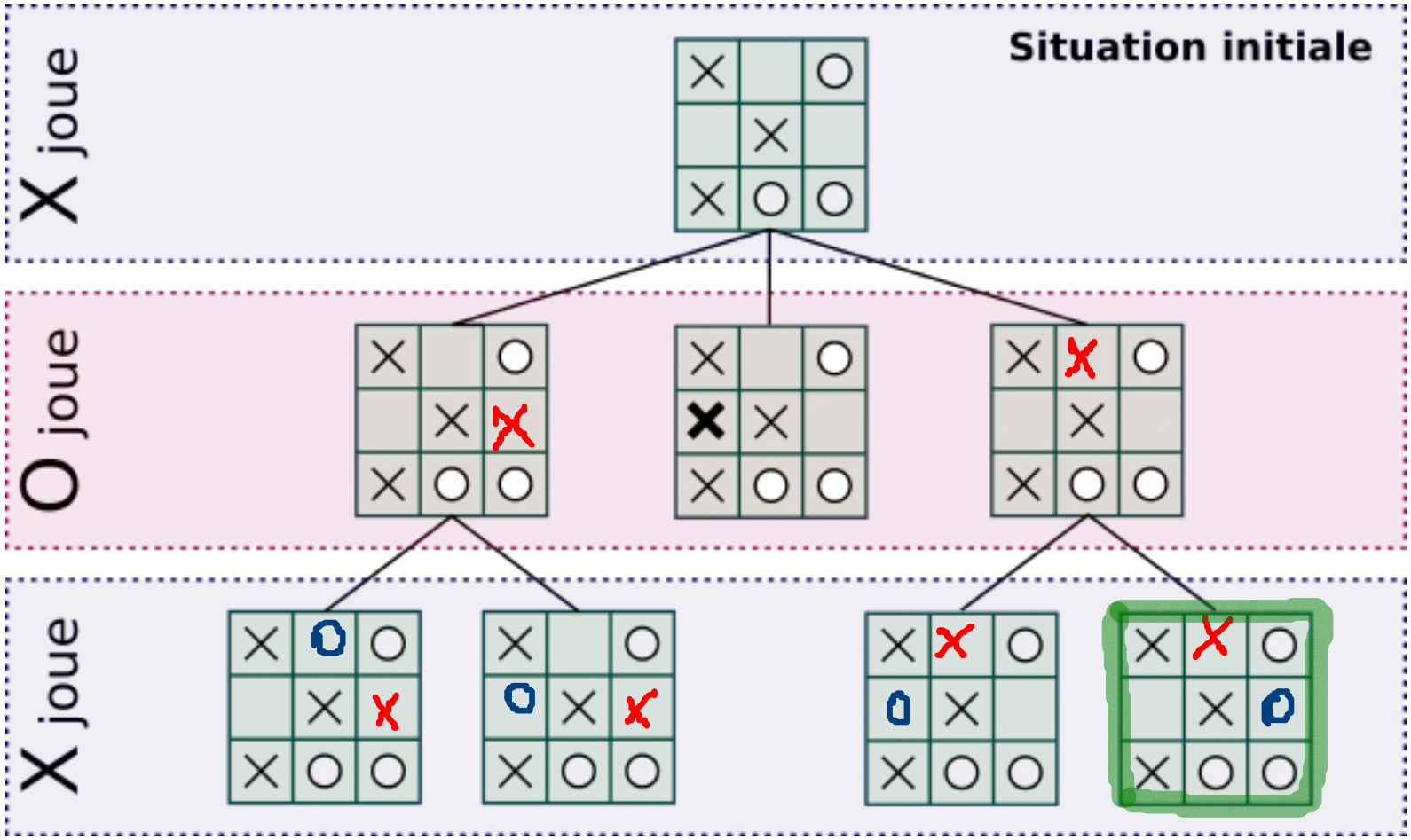


L’intelligence d’un morpion

Correction de l'algorithme Min Max

A partir de la situation de jeu suivante **compléter** l'arbre de jeu en plaçant les croix/ronds manquants jusqu'à atteindre la fin de partie. **Encadrer** les plateaux en situation gagnante.



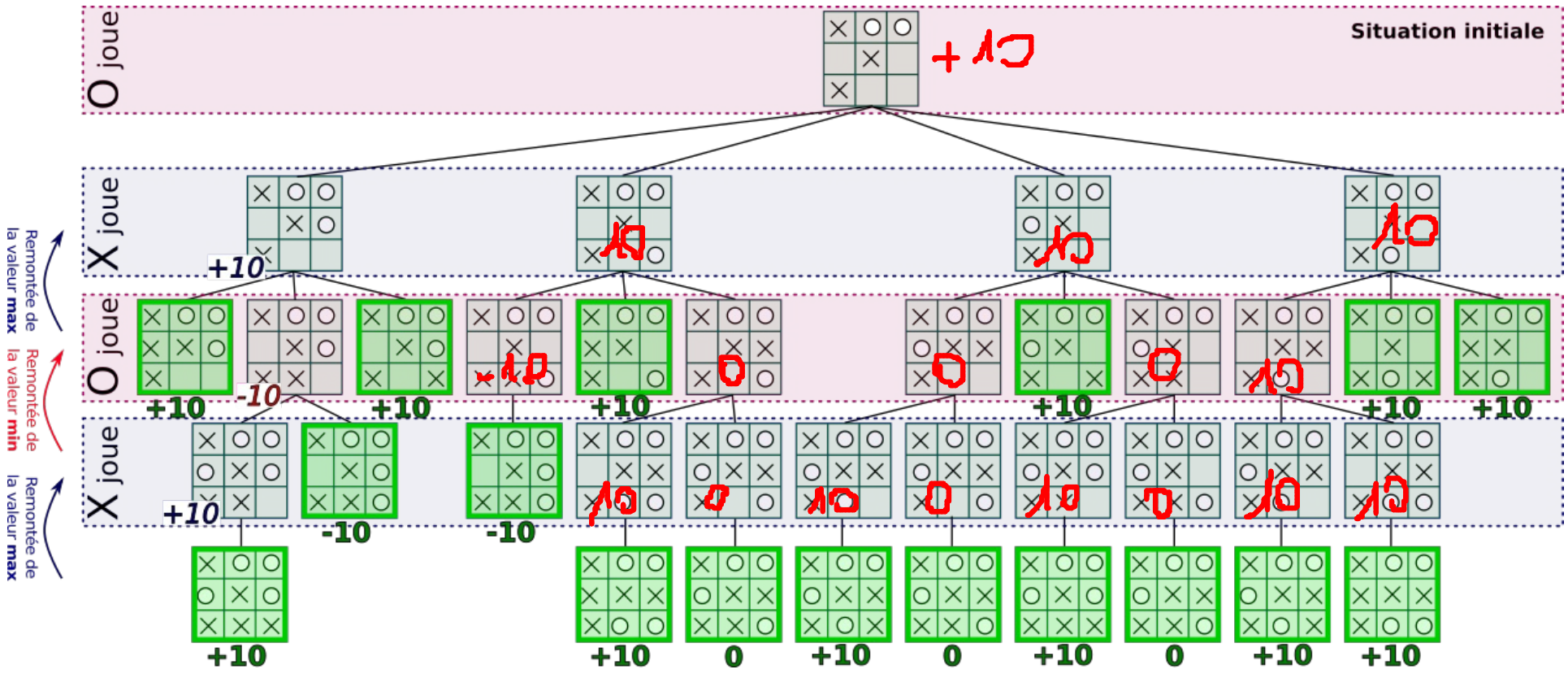
Q4. **Compléter** la méthode *evaluationStatique()* de la classe MorpionIA en respectant sa spécification. **Tester** cette méthode avec le jeux de test présent dans la spécification.

```
def evaluationStatique(self):
    '''Evalue la situation du plateau (+10,-10,0, None)

    ** Test **
    >>>obj.setPlateau(['x','.','.','x','.','.','x','.','.'])
    >>>obj.evaluationStatique()
    10
    >>>obj.setPlateau(['o','o','o','x','x','.','x','.','.'])
    >>>obj.evaluationStatique()
    -10
    >>>obj.setPlateau(['o','x','x','x','o','o','o','o','x'])
    >>>obj.evaluationStatique()
    0
    >>>obj.setPlateau(['o','x','.','.','.','.','.','.','o'])
    >>>obj.evaluationStatique()
    >>>
    '''
    gagne,gagnant=self.analyserPlateau()

    if gagnant=='x': return 10
    elif gagnant=='o' : return -10
    elif self.plateauComplet()==True: return 0
    else : return None
```

L'algorithme MINMAX



A cette situation 'O' joue et le score est -10

→ 'O' va perdre si 'X' joue bien

Q7. **Justifier** que cet algorithme est récursif

Cette fonction minmax() s'appelle elle même -> Fonction récursive.

Q8. **Déterminer** la profondeur maximale de récursivité que peut atteindre cet algorithme lorsqu'il est appliqué au jeu du morpion. **Justifier** que l'utilisation de cet algorithme récursif n'engendrera pas de dépassement mémoire s'il est codé en Python.

Nous sommes en présence d'un parcours en profondeur de l'arbre de jeu.
Le nombre d'appels récursifs sera donc égal, au maximum, à la profondeur de l'arbre -> 8.
Ce nombre d'appel (8) est très inférieur à la limite fixée par défaut dans Python (1000). Le langage sera donc capable d'exécuter ces fonctions récursives.

```python
def minmax(self,joueur):
    '''Identifie le meilleur coup (algo Min_Max)
    Parametre : joueur (str) : 'x', 'o'
    retour : position du meilleur coup (int)

    ** Test **
    >>>obj.setPlateau(['x','x','o','.','o','.','x','.','.'])
    >>>obj.minmax('o')
    (0,3)
    >>>obj.setPlateau(['x','x','o','o','o','.','x','.','.'])
    >>>obj.minmax('x')
    (0,5)


    '''
scoreBranches = [] #Liste des scores de chaque branche
for coup in self.coupsRestants():
    score=self.evaluerCoup(joueur,coup)

    if score==None:
        score,_=self.minmax(self.adversaire(joueur) )

    scoreBranches.append((score,coup))
    self.jouer('.', coup)  # efface le coup joué

if joueur=='x' : return max(scoreBranches)
else : return min(scoreBranches)
```