```
##Couleurs et joueurs

# On prend comme convention 0 pour une case vide, 1 pour une case occupée
par le joueur bleu, et 2 pour une case occupée par le joueur rouge. Par
convention, le joueur 1 sera le bleu, et commencera donc, et le joueur 2
sera donc le rouge.
VIDE = 0
BLEU = 1
ROUGE = 2

##Cases valides autours
#Définis si une cellule est présente dans le tableau
def estDans(tableau, cellule):
    abs = cellule[0]
    ord = cellule[1]
    l = len(tableau)-1
    if 0 > abs or abs > l or 0 > ord or ord > l :
        return False
    return True

#retourne une liste contenant les cellules autours d'une cellule donnée

def cellProxy (plateau, cellule):
    l = len(plateau)-1
    n = cellule[0]
    k = cellule[1]
    s = []
    for x in range (n-1, n+2) :
        for y in range (k-1, k+2):
            if estDans(plateau, [x,y]) :
                if [x,y] != [n-1, k-1] and [x,y] != [n+1, k+1] and [x,y] !=
cellule :
                    s.append([x,y])
    return s


##Valeur d'une cellule
#retourne la couleur d'une cellule du plateau

def valeur(plateau, cellule):
    return plateau[cellule[0]][cellule[1]]

##CellMemeCouleurs
#retourne les cellules de même couleur autours d'une certaine cellule

def cellMemeCouleur (plateau, cellule):
    couleur = valeur(plateau, cellule)
    sortie = []
    T = cellProxy(plateau, cellule)
    for x in T:
        if valeur(plateau, x) == couleur :
            sortie.append(x)
    return sortie


##Plateau plein

#Renvoie un booléen selon si le plateau est plein ou non
def estPlein (Plateau):
```

```
    n = len(Plateau)
    for x in range(n):
        for y in range(n):
            if Plateau[x][y] == 0 :
                return False
    return True

## Liste des cases non vides

def listeCasesNonVides (Plateau): #renvoie la liste des cases non vides du
plateau
    n = len(Plateau)
    Sortie = []
    for x in range(n):
        for y in range (n):
            if Plateau[x][y] == 0 :
                Sortie.append([x,y])
    return Sortie

## générer plateau

#Fonction pratique pour générer un plateau vide de taille c

def platGen (c):
    plateau=[ [ 0 for _ in range (c)] for _ in range (c)]
    return plateau


##

def autreCouleur (couleur):

    if couleur == BLEU :

        return ROUGE


    return BLEU


##
#Renvoie la première case non vide

def premCaseNonVide (plateau):

    for x in range (len(plateau)):

        for y in range (len(plateau)):

            if plateau[x][y] == VIDE:

                return [x, y]


##Positions de départ

#retourne la liste des cellules de départ pour la recherche d'un chemin

def posDeparts (plateau, couleur):
    L = len(plateau)
```

```
    Sortie = []
    if couleur == ROUGE:
        for k in range (L):
            if plateau[0][k]==ROUGE:
                Sortie.append([0,k])
    if couleur == BLEU:
        for n in range (L):
            if plateau[n][0]==BLEU:
                Sortie.append([n,0])
    return Sortie

## Cases valides finales

#retourne la liste des cellules d'arrivée

def casesArivee (plateau, couleur):
    Sortie = []
    L = len(plateau)-1
    if couleur == BLEU :
        for n in range (L+1):
            if plateau[n][L] == BLEU :
                Sortie.append([n,L])
    if couleur == ROUGE :
        for k in range (L+1):
            if plateau[L][k] == ROUGE:
                Sortie.append([L,k])
    return Sortie

##Parcours en profondeur

#initialiste la fonction posGagnante

def __posGagnante (plateau, couleur, tableau, cellule):
    L = len(plateau)
    for k in cellMemeCouleur(plateau, cellule):
        if valeur(tableau, k) == 0 :
            tableau[k[0]][k[1]] = 1
            if k in casesArivee(plateau, couleur):
                return True
            if __posGagnante(plateau, couleur, tableau, k):
                return True
    return False


# retourne si la couleur 'couleur' est gagnante sur le plateau
def posGagnante(plateau, couleur):
    L = len(plateau)
    tableau = [ [0 for _ in range (L)] for _ in range (L)]
    for posDepart in posDeparts(plateau, couleur):

        #On regarde pour chaque position de départ s'il y a un chemin pour
la relier à une position finale

        if __posGagnante(plateau, couleur, tableau, posDepart):
            return True

    return False
```

```
## couleur Gagnante

#Définis si une couleur donnée est gagnante sur le plateau
def couleurGagnante(plateau):

    if posGagnante(plateau,BLEU):

        return BLEU

    if posGagnante(plateau, ROUGE):

        return ROUGE

    return False
```