```python
import numpy as np
import random as rd
import copy
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import time

pos_ex=[[0, 0 ,0, 0, 0, 0, 0], [0, 0, 2, 0, 0, 0, 0], [0, 0, 1 ,0, 0 ,0, 0], [0, 0 ,2, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0 ,1], [0, 2 ,1, 0, 1, 0 ,2]]

##pour afficher un joli jeu

def affiche_grille(grille):
    #pions jaunes = joueur1
    ylim,xlim = 6,7
    xoffset = 0.5
    yoffset = 0.5
    decal = 0.5
    fig,ax = plt.subplots()
    ax.set(xlim = (0.5*xoffset,xlim+1.5*xoffset), ylim = (0.5*yoffset,ylim+1.5*yoffset), aspect = "equal")
    ax.add_artist(mpatches.Rectangle((xoffset,yoffset), xlim, ylim))

    for i in range(0,xlim) :
        for j in range(0,ylim):
            if grille[j][i] == 0 :
                ax.add_artist(mpatches.Circle(xy = (xoffset+i+decal,ylim-(yoffset+j)+decal), radius = 0.4, color = (1,1,1)))
            if grille[j][i] == 2 :
                ax.add_artist(mpatches.Circle(xy = (xoffset+i+decal,ylim-(yoffset+j)+decal), radius = 0.4, color = (1,0,0)))
            if grille[j][i]== 1 :
                ax.add_artist(mpatches.Circle(xy = (xoffset+i+decal,ylim-(yoffset+j)+decal), radius = 0.4, color = (1,1,0)))
    plt.show()
    return None

affiche_grille(pos_ex)

def vide():
    return [[0 for j in range(7)] for i in range(6)]
#ATTENTION ne pas faire [[0]*7]*6
# car un changement d'un coefficient se duplique sur toute la colonne!!!

def pleine(p,c):
    '''vérifie si la colonne c est pleine'''
    return p[0][c]!=0

def coups_possibles(p):
    liste_col=[]
    for c in range(7):
        if not pleine(p,c):
            liste_col.append(c)
    return liste_col

def jouer(p,c,i):
    '''renvoie la nouvelle position de jeu si le joueur i joue la colonne c'''
    newp=copy.deepcopy(p) #ATTENTION copy ne suffit pas
    for j in range(-1,-7,-1): #on cherche le coefficient nul le plus bas possible dans la colonne c
        if p[j][c]==0:
            newp[j][c]=i
            return newp
    return None #cas où la colonne i est déjà pleine


def gagne(p,i):
    #print('la position p de gagne: ',p)
    g=[i]*4 #la liste gagnante

    #détection des alignements horizontaux
    for i in range(len(p)):
```

```
67          for j in range(4):
68              if p[i][j:j+4]==g:
69                  #si la grille est un tableau numpy, faire list(p[i][j:j+4])
70                  #car sinon le test d'égalité ne fonctionnera pas
71                  return True
72
73      #détection des alignement verticaux
74      for j in range(len(p[0])):
75
76          for i in range(3):
77              liste=[]
78              for k in range(i,i+4):
79                  liste.append(p[k][j])
80              if liste==g:
81                  return True
82
83      #détection des alignement diagonaux haut->bas
84      for i0 in range(3):
85          for j0 in range(4): #les points de départ
86              liste=[]
87              for k in range(4):
88                  liste.append(p[i0+k][j0+k])
89              if liste==g:
90                  return True
91
92
93
94      #détection des alignement diagonaux bas->haut
95      for i0 in range(3,6):
96          for j0 in range(4): #les points de départ
97              liste=[]
98              for k in range(4):
99                  liste.append(p[i0-k][j0+k])
100             if liste==g:
101                 return True
102
103     return False
104
105 #création du tableau des points pour calculer h(p)
106 points=[[3,4,5,7,5,4,3],[4,6,8,10,8,6,4],[5,8,11,13,11,8,5],[5,8,11,13,11,8,5],[4,6,8,
    10,8,6,4],[3,4,5,7,5,4,3]]
107
108 def h(p):
109     #cas où la position contient un alignement gagnant pour J1 ou J2
110     if gagne(p,1):
111         return np.inf
112     if gagne(p,2):
113         return -np.inf
114
115     #cas où ce n'est pas une position terminale gagnante pour l'un des joueurs
116     somme=0
117     for i in range(len(p)):
118         for j in range(len(p[0])):
119             if p[i][j]!=0:
120                 somme+=points[i][j]*(-2*p[i][j]+3) #(-2*p[i][j]+3=1 si joueur1 =-1 si
                    joueur 2
121     return somme
122
123
124
125 ##Algorithme Min-Max
126
127 def maxiJ1(p,n): #on cherche à maximiser l'heuristique
128                     #concerne le joueur 1
129     if n==0:
130         return (h(p),None)
131     if gagne(p,1):
132         return (np.inf,None) #le jeu s'arrête dans ce cas
133
134     coups=coups_possibles(p)
135     if coups==[]: #la grille est pleine, la récursivité s'arrête
136         return (h(p),None)
```

```
137
138         #si la partie peut se poursuivre:
139         maxi=-np.inf
140         c_maxi=coups[0]
141         for c in coups:
142             pos=jouer(p,c,1) #la position suivante si J1 joue le coup c
143             valeur=miniJ2(pos,n-1)[0] #J2 va chercher à minimiser la valeur
144             if valeur>=maxi:
145                 maxi=valeur
146                 c_maxi=c
147         return (maxi,c_maxi)
148
149     def miniJ2(p,n): #on cherche à minimiser l'heuristique
150                         #concerne le joueur 2
151         if n==0:
152             return (h(p),None)
153         if gagne(p,2):
154             return (-np.inf,None) #le jeu s'arrête dans ce cas
155
156         coups=coups_possibles(p)
157         if coups==[]:  #la grille est pleine, la récursivité s'arrête
158             return (h(p),None)
159
160         #si la partie peut se poursuivre:
161         mini=np.inf
162         c_maxi=coups[0]
163         for c in coups:
164             pos=jouer(p,c,2) #la position suivante si J2 joue le coup c
165             valeur=maxiJ1(pos,n-1)[0] #J1 va chercher à maximiser la valeur
166             if valeur<=mini:
167                 mini=valeur
168                 c_mini=c
169         return (mini,c_mini)
170
171     def simulation_random():
172         '''simule une partie où chaque joueur joue au hasard'''
173         p=vide()
174         i=1 #le joueur qui commence à jouer
175         Fin=False #pour vérifier que la grille n'est pas pleine
176         while  gagne(p,1)==False and gagne(p,2)==False and not Fin :
177
178             colonnes_dispos=coups_possibles(p)
179             if colonnes_dispos!=[]:
180                 c=rd.choice(colonnes_dispos)
181                 p=jouer(p,c,i)
182                 affiche_grille(p)
183                 time.sleep(0.5)
184                 i=-i+3  #on change de joueur
185             else:
186                 Fin=True #le jeu est fini car la grille est pleine
187
188         if gagne(p,1):
189             print('le joueur 1 a gagné')
190             #return 1
191         elif gagne(p,2):
192             print('le joueur 2 a gagné')
193             #return 2
194         else:
195             print('match nul')
196             #return 0
197
198     def stat_random(N):#on lance N simulations de jeu
199         res=[0,0,0]
200         for i in range(N):
201             res[simulation_random()]+=1
202         return res
203
204     #quand on joue au hasard, très très peu de matchs nuls, le joueur1 a un peu plus de
        chance de gagner que joueur 2
205
206     #-----------------------------------------------------------------------
207     #on fait jouer les joueurs avec l'algo minmax
```

```
208
209 def simulation_joueur1et2(n): #le joueur1 prévoit n coups d'avance
210     #le joueur2 prévoit AUSSI n coups d'avance
211     p=vide()
212     i=1 #le joueur qui commence à jouer
213     Fin=False #pour
214     while gagne(p,1)==False and gagne(p,2)==False and not Fin:
215
216
217         colonnes_dispos=coups_possibles(p)
218         if colonnes_dispos!=[]:
219             if i==1:
220                 c=maxiJ1(p,n)[1]
221
222                 p=jouer(p,c,i)
223
224                 i=2  #on change de joueur
225             elif i==2: #ATTENTION à ne pas mettre seulement un if
226                  c=miniJ2(p,n)[1]
227                  p=jouer(p,c,i)
228                  i=1
229             affiche_grille(p)
230             time.sleep(0.5)
231         else:
232             Fin=True #le jeu est fini car tout est plein
233
234     #print(np.array(pos))
235     if gagne(p,1):
236         print('le joueur 1 a gagné')
237         #return 1
238     elif gagne(p,2):
239         print('le joueur 2 a gagné')
240         #return 2
241     else:
242         print('match nul')
243         #return 0
244
245
```