```python
import numpy as np
import random as rd
import copy

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]]


def vide():
    return [[0]*7 for i in range(6)]


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

#création du tableau des points pour calculer h(p)
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]]

def h(p):
    #cas où la position est gagnante pour J1 ou J2
    if gagne(p,1):
        return np.inf
    if gagne(p,2):
        return -np.inf

    #cas où ce n'est pas une position terminale gagnante pour l'un des joueurs
    somme=0
    for i in range(len(p)):
        for j in range(len(p[0])):
            if p[i][j]!=0:
                somme+=points[i][j]*(-2*p[i][j]+3) #(-2*p[i][j]+3=1 si joueur1 =-1 si joueur 2
    return somme

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

    #détection des alignement horizontaux
    for i in range(len(p)):
        for j in range(4):
            if p[i][j:j+4]==g:
                return True

    #détection des alignement verticaux
    for j in range(len(p[0])):

        for i in range(3):
            liste=[]
            for k in range(i,i+4):
                liste.append(p[k][j])
            if liste==g:
                return True
```

```
69
70      #détection des alignement diagonaux haut->bas
71      for i0 in range(3):
72          for j0 in range(4): #les points de départ
73              liste=[]
74              for k in range(4):
75                  liste.append(p[i0+k][j0+k])
76              if liste==g:
77                  return True
78
79
80
81      #détection des alignement diagonaux bas->haut
82      for i0 in range(3,6):
83          for j0 in range(4): #les points de départ
84              liste=[]
85              for k in range(4):
86                  liste.append(p[i0-k][j0+k])
87              if liste==g:
88                  return True
89
90      return False
91
92
93  ##Algorithme Min-Max
94
95  def maxiJ1(p,n): #on cherche à maximiser l'heuristique
96                       #concerne le joueur 1
97      if n==0:
98          return (h(p),None)
99      if gagne(p,1):
100         return (np.inf,None) #le jeu s'arrête dans ce cas
101 #     if gagne(p,2):
102 #         return (-np.inf,None) #le jeu s'arrête dans ce cas
103     coups=coups_possibles(p)
104     if coups==[]: #la grille est pleine, la récursivité s'arrête
105         return (h(p),None)
106     #si la partie peut se poursuivre:
107     maxi=-np.inf
108     c_maxi=coups[0]
109     for c in coups:
110         pos=jouer(p,c,1) #la position suivante si J1 joue le coup c
111         valeur=miniJ2(pos,n-1)[0] #J2 va chercher à minimiser la valeur
112         if valeur>=maxi:
113             maxi=valeur
114             c_maxi=c
115     return (maxi,c_maxi)
116
117 def miniJ2(p,n): #on cherche à minimiser l'heuristique
118                      #concerne le joueur 2
119     if n==0:
120         return (h(p),None)
121     if gagne(p,2):
122         return (-np.inf,None) #le jeu s'arrête dans ce cas
123 #     if gagne(p,1):
124 #         return (np.inf,None) #le jeu s'arrête dans ce cas
125     coups=coups_possibles(p)
126     if coups==[]:  #la grille est pleine, la récursivité s'arrête
127         return (h(p),None)
128     #si la partie peut se poursuivre:
129     mini=np.inf
130     c_maxi=coups[0]
131     for c in coups:
132         pos=jouer(p,c,2) #la position suivante si J2 joue le coup c
133         valeur=maxiJ1(pos,n-1)[0] #J1 va chercher à maximiser la valeur
134         if valeur<=mini:
135             mini=valeur
136             c_mini=c
137     return (mini,c_mini)
```