```
''' 3 - Intelligence artificielle'''

'''3-1 - Algorithme KNN - Reconnaissance de panneaux'''

'''Dossier partagé images: https://www.dropbox.com/sh/2zce39bxxtfcl58/AADVz2oW9QxT-leD-0

## 3-2 - Algorithme KNN - Reconnaissance de panneaux

def Distance_uv(u,v):
    n = len(u)
    Dst = 0
    for i in range(n):
        di = u[i]-v[i]
        Dst += di**2
    Dst = Dst**(1/2)
    return Dst

def Distance(u,Lv):
    Ld = []
    for i in range(len(Lv)):
        v = Lv[i]
        Dst = Distance_uv(v,u)
        Res = [Dst,i]
        Ld.append(Res)
    return Ld

def Proches(u,Lv,k):
    Ld = Distance(u,Lv)
    Ld.sort()
    Res = Ld[:k]
    return Res

## Lecture des images

# Affichage

import matplotlib.pyplot as plt
plt.close('all')

def Affiche(image):
    plt.figure()
    plt.imshow(image)
    plt.axis('off')
    plt.show()
    plt.pause(0.00001)

# Lecture des images

def Lecture(Chemin):
    Image = plt.imread(Chemin)
    return Image


## Fonctions d'analyse des images

def Analyse(Image):
    Nl,Nc = Image.shape[0:2]
    N = Nl * Nc
    L_RGB = []
    for ligne in Image:
        for pixel in ligne:
```

```
            R,G,B = pixel
            R = float(R)
            G = float(G)
            B = float(B)
            L_RGB += [R,G,B]
    return L_RGB

''' Vérification
len(Analyse(Lecture('Sources\\0\\0.bmp')))
'''

def Analyse_Globale(L_Chemin):
    Res = []
    N = len(L_Chemin)
    for i in range(len(L_Chemin)):
        Chemin = L_Chemin[i]
        print("Apprentissage image",i+1,"sur",N)
        Image = Lecture(Chemin)
        Image = Image[:,:,:3] # Si RGBA
        Analyse_im = Analyse(Image)
        Res.append(Analyse_im)
    return Res

## Création de la base des données

# Ouverture des images sources

Sources = "Sources\\"
Dossiers = [0,1,2,3,4,5,6,7]
Nb_Dossiers = len(Dossiers)
Nb_Images_Dossiers = [5,5,5,5,5,5,5,5]

Liste_Chemin = []
Liste_Dossier = []
Liste_Num = []
for d in Dossiers:
    Nb_Im = Nb_Images_Dossiers[d]
    for im in range(Nb_Im):
        Chemin = Sources + str(d) + "\\" + str(im) + ".bmp"
        Liste_Chemin.append(Chemin)
        Liste_Dossier.append(d)
        Liste_Num.append(im)

# Analyse des images

Donnees = Analyse_Globale(Liste_Chemin)

## Reconnaissance automatique

# Fermeture des images

plt.close('all')

# Ouverture et analyse de l'image recherchée

Recherche = "Recherche\\"
Im_Cherchee_Chemin = Recherche + "11.bmp"
Im_Cherchee = Lecture(Im_Cherchee_Chemin)
Affiche(Im_Cherchee)
Im_Cherchee_Infos = Analyse(Im_Cherchee)
```

```
# Recherche des k plus proches voisins

k = 5
Resultat_Proches = Proches(Im_Cherchee_Infos,Donnees,k)
Resultat_Ind = [Resultat_Proches[i][1] for i in range(k)]
Resultat_Dossiers = [Liste_Dossier[ind] for ind in Resultat_Ind]
Resultat_Num = [Liste_Num[ind] for ind in Resultat_Ind]
print("Dossiers trouvés: ",Resultat_Dossiers)
print("Numéros des images: ",Resultat_Num)

# Selection du résultat

def Max_Occurences(L,n):
    Occ = [0 for _ in range(n)]
    for t in L:
        Occ[t] += 1
    Max = max(Occ)
    for t in L:
        if Occ[t]==Max: # Remarque
            return t

# Remarque: Ne pas rechercher Occ[i]==Max. En cas d'exaeqo, cela renvoie le premier doss

Dossiers_final = Max_Occurences(Resultat_Dossiers,Nb_Dossiers)
print("Dossier: ",Dossiers_final)

# Affichage du résultat

Im_Trouvee_Chemin = Sources + str(Dossiers_final) + "\\0" + ".bmp"
Im_Trouvee = Lecture(Im_Trouvee_Chemin)
Affiche(Im_Trouvee)

## Matrice de confusion

'''
Cette partie est adaptée aux dossiers numérotés de 0 à 7 et à la bonne organisation des
'''

# M(1,0)=1

'''
On a recherché une image 11 ou 12 de sens interdit
L'algorithme a renvoyé le résultat du dossier 0: sens prioritaire inverse
'''

# Resolution

def Resolution(N,k):
    Recherche = "Recherche\\"
    Im_Cherchee_Chemin = Recherche + N + ".bmp"
    Im_Cherchee = Lecture(Im_Cherchee_Chemin)
    Im_Cherchee_Infos = Analyse(Im_Cherchee)
    Resultat_Proches = Proches(Im_Cherchee_Infos,Donnees,k)
    Resultat_Ind = [Resultat_Proches[i][1] for i in range(k)]
    Resultat_Dossiers = [Liste_Dossier[ind] for ind in Resultat_Ind]
    Resultat_Num = [Liste_Num[ind] for ind in Resultat_Ind]
    Dossiers_final = Max_Occurences(Resultat_Dossiers,Nb_Dossiers)
    return Dossiers_final

N = "11"
k = 5
```

```
Resolution(N,k)

# Liste LR

LR = ["01","11","12","21","22","31","41","42","51","52","61","71"]

# Etude k

import numpy as np

def Etude(k):
    A = np.zeros([Nb_Dossiers,Nb_Dossiers])
    for R in LR:
        l = int(R[0])
        c = Resolution(R,k)
        A[l,c] += 1
    return A

M1 = Etude(1)
print(M1)

# Matrices pour k=1 à 5

for k in range(1,41):
    Mk = Etude(1)
    print('k=',k)
    print(Mk)

''' Quel que soit k de 1 à 40
[[1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 2. 0. 0. 0. 0. 0. 0.]
 [0. 0. 2. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 2. 0. 0. 0.]
 [0. 0. 0. 0. 0. 2. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]]
'''

'''
On a de la chance, ici l'algorithme fonctionne très bien.
Il faudrait essayer d'agandir notre base de données avec des panneaux plus ressemblants
'''
```