TP 1 - Espaces de représentation des couleurs

Le test d'Ishihara

Ce test, inventé en 1917 par Shinobu Ishihara, est un recueil de 38 planches utilisé pour dépister les anomalies de la vision des couleurs dont quelques exemples sont illustrés figure 1.

Ces tests composés de planches « pseudoisochromatiques » sont les plus fréquemment utilisés pour la détection des déficiences congénitales des teintes rouge et verte. Quelques-uns testent aussi les anomalies concernant la perception du bleu.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import image
import seaborn as sns

from mpl_toolkits.mplot3d import Axes3D
```

Exercice 1 - Corrélations et contrastes des planches RVB

Les lignes suivantes lit l'image $ishihara-0.png$ codée en RVB (Rouge, Vert, Bleu) et la stocke dans une matrice tridimensionnelle I de taille $hauteur \times largeur \times 3$.

On peut séparer cette matrice en trois sous-matrices bidimensionnelles appelées canaux : $R = I(i,j,1)$ pour le canal rouge, $V = I(i,j,2)$ pour le canal vert, et $B = I(i,j,3)$ pour le canal bleu.

Chacun d'entre eux est composé d'entiers compris entre 0 et 255, qui représentent l'intensité lumineuse du pixel situé sur la ligne i et la colonne j. De part leur dénomination, chaque canal apporte donc une part de couleur à l'image, que ce soit du rouge, du vert ou du bleu.

Représentations R, V et B

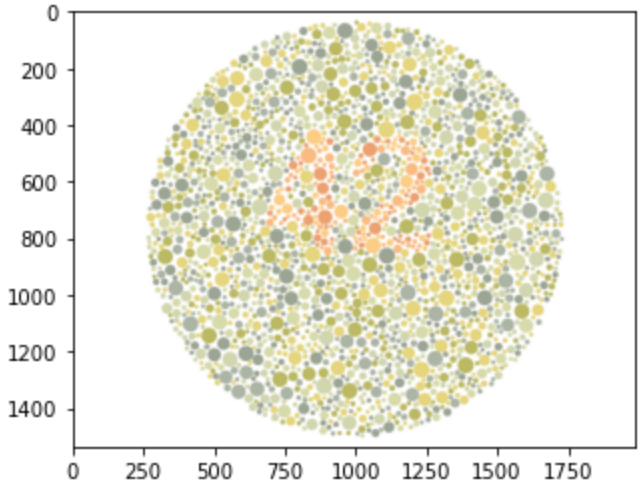
Chargement de l'image (à décommenter)

In [2]:
```
# Importer des fichiers sur Google Colab à partir de votre ordinateur :

#from google.colab import files

#uploaded = files.upload()
```

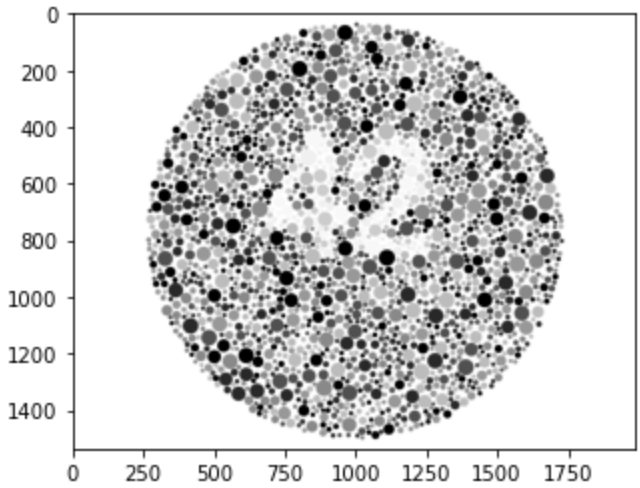
In [3]:
```
# charge le fichier dans une matrice de pixels couleur.
Data=image.imread("assets/ishihara-0.png")

# affiche les dimensions de la matrice.
print(Data.dtype)
print(Data.shape[0:2])

# accède à la valeur du premier pixel.
print(Data[0,0,0])
```

```
# Visualisation image
plt.imshow(Data)
plt.show()
```

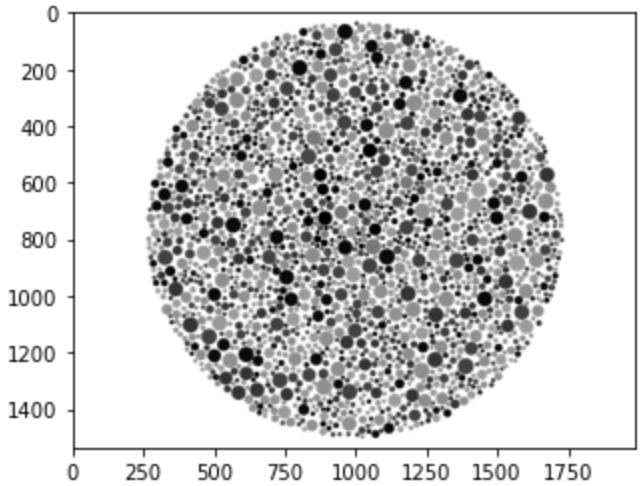
```
float32
(1536, 1985)
1.0
```



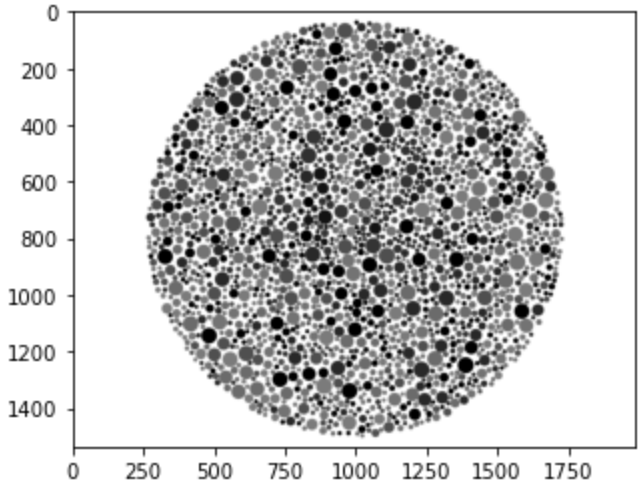
In [4]:
```
# Decoupage de l'image en trois canaux et conversion en doubles :
R=Data[:,:,0]
V=Data[:,:,1]
B=Data[:,:,2]

# Affichage du canal R :
plt.imshow(R,cmap='gray')
plt.show()
```



In [5]:
```
# Affichage du canal V :
plt.imshow(V,cmap='gray')
plt.show()
```



In [6]:
```
# Affichage du canal B :
plt.imshow(B,cmap='gray')
plt.show()
```

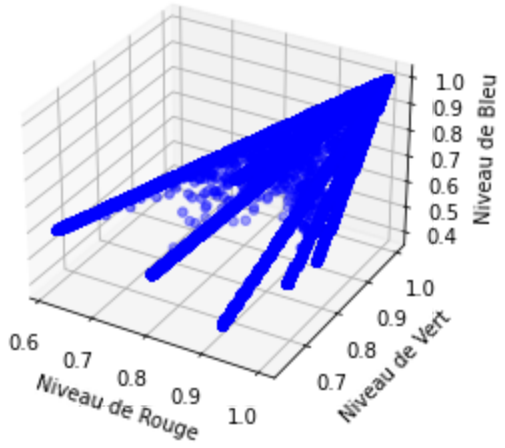


En affichant les matrices I, R, V et B sous forme d'images, on observe que les images sont similaires et on distingue un motif dans les nuances de rouge.

Dans la suite, les pixels sont considérés comme des points de R^3 que l'on affiche dans un repère dont les axes correspondent aux trois niveaux de couleur.

In [7]:
```
# Transformation image en nuage de pixels 3D : les trois canaux sont vectorises et conca
R=np.ravel(R)
V=np.ravel(V)
B=np.ravel(B)

print(R.shape)

# Affichage du nuage de pixels dans le repere RVB :
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(R, V, B, c='b', marker='o')

ax.set_xlabel('Niveau de Rouge')
ax.set_ylabel('Niveau de Vert')
ax.set_zlabel('Niveau de Bleu')

plt.show()
```

```
(3048960,)
```



Ils forment un faisceau allongé suivant plusieurs directions, ce qui confirme l'observation précédente, à savoir que les trois canaux sont fortement corrélés.

Etude de la corrélation entre les couleurs R, V et B

In [8]:
```
# Matrice des donnees :
dim=R.shape[0]

# Les trois canaux sont vectorises et concatenes
X=np.zeros([dim,3])
X[:,0]=R
X[:,1]=V
X[:,2]=B

print(X.shape)

print(X)
```

```
(3048960, 3)
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 ...
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

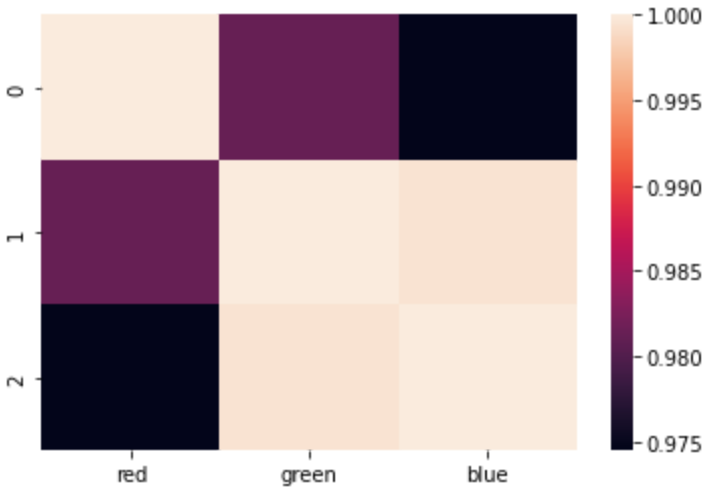
In [9]:
```
# Calculer la matrice de variance/covariance

#Méthode de reduction centrée.

M=np.mean(X, axis = 0)
A=X-M


print("Moyenne : ")
print(M)
print("------------------------")
CO=(A.T).dot(A)/A.shape[0]
print("Matrice covariance : ")
print(CO)
print(CO.shape)
```

```
Moyenne :
[0.92628777 0.92142349 0.85439634]
------------------------
Matrice covariance :
```

```
[[0.01436858 0.01332363 0.02051961]
 [0.01332363 0.01396285 0.02297153]
 [0.02051961 0.02297153 0.04489739]]
(3, 3)
```

In [10]:
```
# Calculer les coefficients de correlation lineaire (np.corrcoef)
print("Matrice de correlation : ")
cof=np.corrcoef(CO)


sns.heatmap(cof,xticklabels=['red','green','blue'])
print(cof)
```

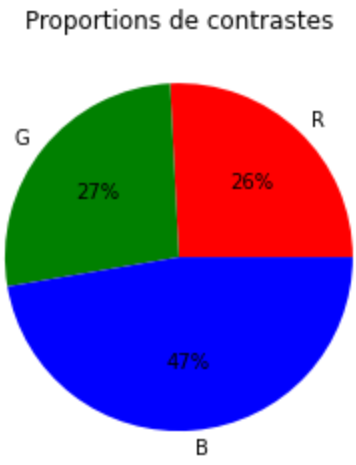
```
Matrice de correlation :
[[1.         0.98123001 0.97451592]
 [0.98123001 1.         0.99948205]
 [0.97451592 0.99948205 1.        ]]
```



In [11]:
```
# Calculer les proportions de contraste :

n=np.sum(CO,axis=0) #
m=np.sum(n)

#Liste des contrastes
L=[x/m for x in n]
print(L)
```

```
[0.25801265581253774, 0.26896311336108136, 0.4730242308263809]
```

In [12]:
```
plt.title("Proportions de contrastes")
plt.pie(L, labels=["R","G","B"],colors = ["red","green","blue"], autopct = '%0.0f%%')
plt.show()
print()
```



Exercice 2 - Analyse en Composantes Principales

Implémenter l'ACP : 1) extraire les 3 vecteurs propres, notés X_1, X_2, X_3, associés aux 3 plus grandes valeurs propres de la matrice de variance-covariance Σ (par les fonctions *np.cov* et *np.linalg.eig*). Ces vecteurs propres constitueront le nouveau & repère P c'est-à-dire les axes principaux.
2) Projetez ensuite les données dans cette nouvelle base en les multipliant par la base $P = [X_1 X_2 X_3]$.

In [13]:
```
# Matrice des donnees :
dim=R.shape[0]

# Les trois canaux sont vectorises et concatenes
X=np.zeros([dim,3])
X[:,0]=R
X[:,1]=V
X[:,2]=B

print(X.shape)
```

```
(3048960, 3)
```

Calculez la matrice de variance-covariance en utilisant par exemple la fonction **np.cov**

In [14]:
```
# Matrice de variance/covariance :
C=np.cov(X.T)
print(C)
```

```
[[0.01436858 0.01332363 0.02051962]
 [0.01332363 0.01396285 0.02297154]
 [0.02051962 0.02297154 0.0448974 ]]
```

La matrice Σ de variance/covariance est symétrique et réelle. Elle admet donc une base orthonormée de vecteurs propres.

Calculez ses valeurs propres et vecteurs propres à l'aide de l'appel à la fonction **np.linalg.eig**

In [15]:
```
# Calcul des valeurs/vecteurs propres de Sigma :
eigen_values,P=np.linalg.eig(C)
```

In [16]:
```
#  Tri des valeurs propres :
print(eigen_values) #Le tri est automatique
```

```
[0.0681779  0.00461629 0.00043465]
```

Calculez la matrice des **composantes principales** des pixels Projetez ensuite les données dans cette nouvelle base en multipliant chaque vecteur par la base $P = [X_1 X_2 X_3]$.

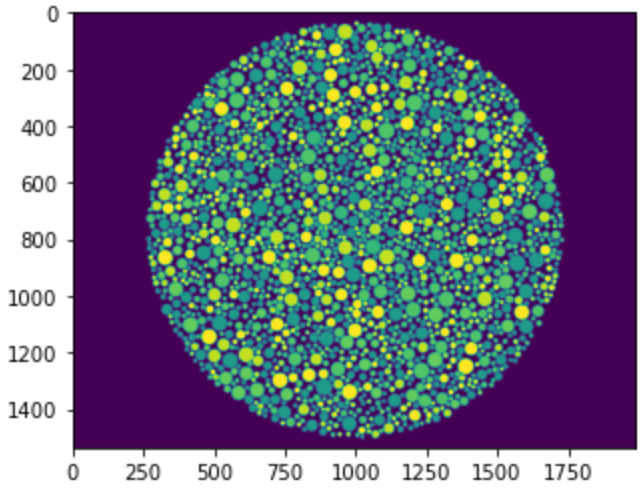
In [17]:
```
# Calcul des composantes principales
#c'est à dire des coefficients de projection sur les axes principaux:
print(P)
print(P.shape,X.shape)
```

```
[[-0.4130039  -0.75423633  0.51044621]
 [-0.4394815  -0.32585122 -0.83706451]
 [-0.79767399  0.57004257  0.19689509]]
(3, 3) (3048960, 3)
```

In [18]:
```
# Projection sur la premiere composante principale :

P1=P.T[0,:] #premiere colonne de P
new_X=np.dot(P1,X.T)
```

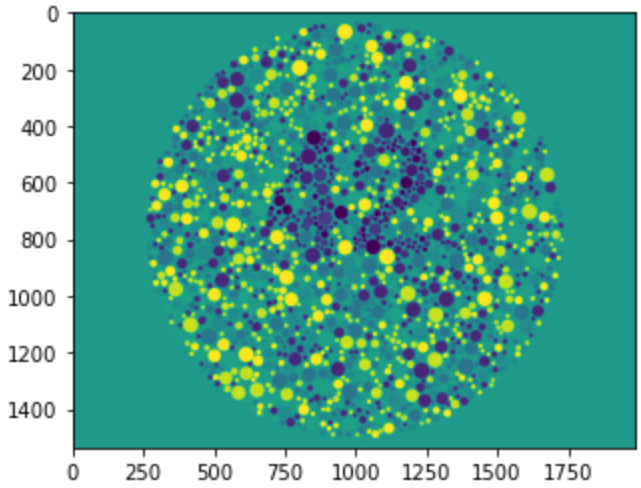
```
print(new_X.shape)


# Affichage de cette projection

new_X_im=new_X.reshape(1536, 1985) #taille origine


plt.imshow(new_X_im)
plt.show()
```

```
(3048960,)
```



In [19]:
```
# Projection sur la deuxieme composante principale :

P2=P.T[1,:] #deuxieme colonne de P
new_X2=np.dot(P2,X.T)

print(new_X2.shape)

# Affichage de cette projection

new_X2_im=new_X2.reshape(1536, 1985) #taille origine

plt.imshow(new_X2_im)
plt.show()
```

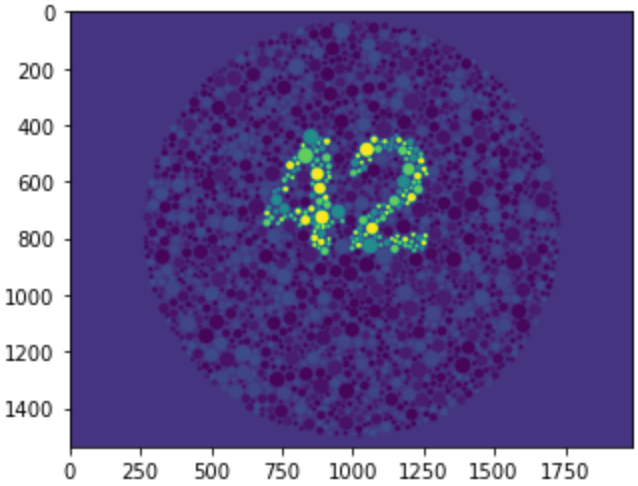
```
(3048960,)
```



In [86]:
```
# Projection sur la troisieme composante principale :

P3=P.T[2,:] #troisieme colonne de P
new_X3=np.dot(P3,X.T)
print(P3)
```

```
# Affichage de cette projection

new_X3_im=new_X3.reshape(1536, 1985) #taille origine

plt.imshow(new_X3_im)
plt.show()
```

```
[ 0.51044621 -0.83706451  0.19689509]
```



Etude la correlation dans le nouveau repère

In [21]:
```
# Matrice de variance/covariance dans le nouveau repere :
M1=np.concatenate((new_X.reshape(len(new_X),1),new_X2.reshape(len(new_X),1),new_X3.resha
#on reshape (3048960,) -> (3048960,1) pour pouvoir concatener

C2=np.cov(M1.T)
print(C2)
```

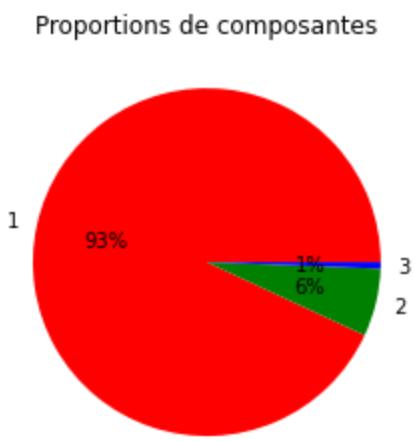
```
[[6.81779017e-02 1.51221969e-15 4.10658660e-16]
 [1.51221969e-15 4.61628851e-03 8.31781538e-16]
 [4.10658660e-16 8.31781538e-16 4.34645801e-04]]
```

In [22]:
```
# Coefficients de correlation lineaire :
cof=np.corrcoef(C2)
print(cof)
```

```
[[ 1.  -0.5 -0.5]
 [-0.5  1.  -0.5]
 [-0.5 -0.5  1. ]]
```

In [23]:
```
# Proportions de contraste :

n=np.sum(C2,axis=0)
m=np.sum(n)

#Liste des contrastes
L=[x/m for x in n]
print(L)

plt.title("Proportions de composantes")
plt.pie(L, labels=["1","2","3"],colors = ["red","green","blue"], autopct = '%0.0f%%')
plt.show()
print()
```

```
[0.931025336697908, 0.06303921731010609, 0.0059354459919859085]
```



Interprétation

Le but du test de Ishihara est d'identifier les personnes qui ne distinguent pas, dans une image de luminance à peu près uniforme, un motif n'apparaissant que dans les chrominances. Il est donc voulu que l'on ne distingue le motif que dans la deuxième et/ou la troisième composante(s) principale(s).

Comment est-ce possible puisque le contraste est censé être maximal dans la première composante principale ?

S. Ishihara a justement créé des images de sorte que le contraste soit faible entre les couleurs du motif et celles du reste de l'image, alors qu'il y a un fort contraste dans la luminance, grâce au fait que ses images contiennent beaucoup de pixels blancs (entre les taches colorées).

Exercice 3 - Quizz

Des symboles de la culture Geek se cachent dans des mosaïques d'Ishihara (archive *Quizz_GroupeXX.zip*).

Utilisez l'ACP pour les faire apparaître et à vous de les identifier !

In [129…

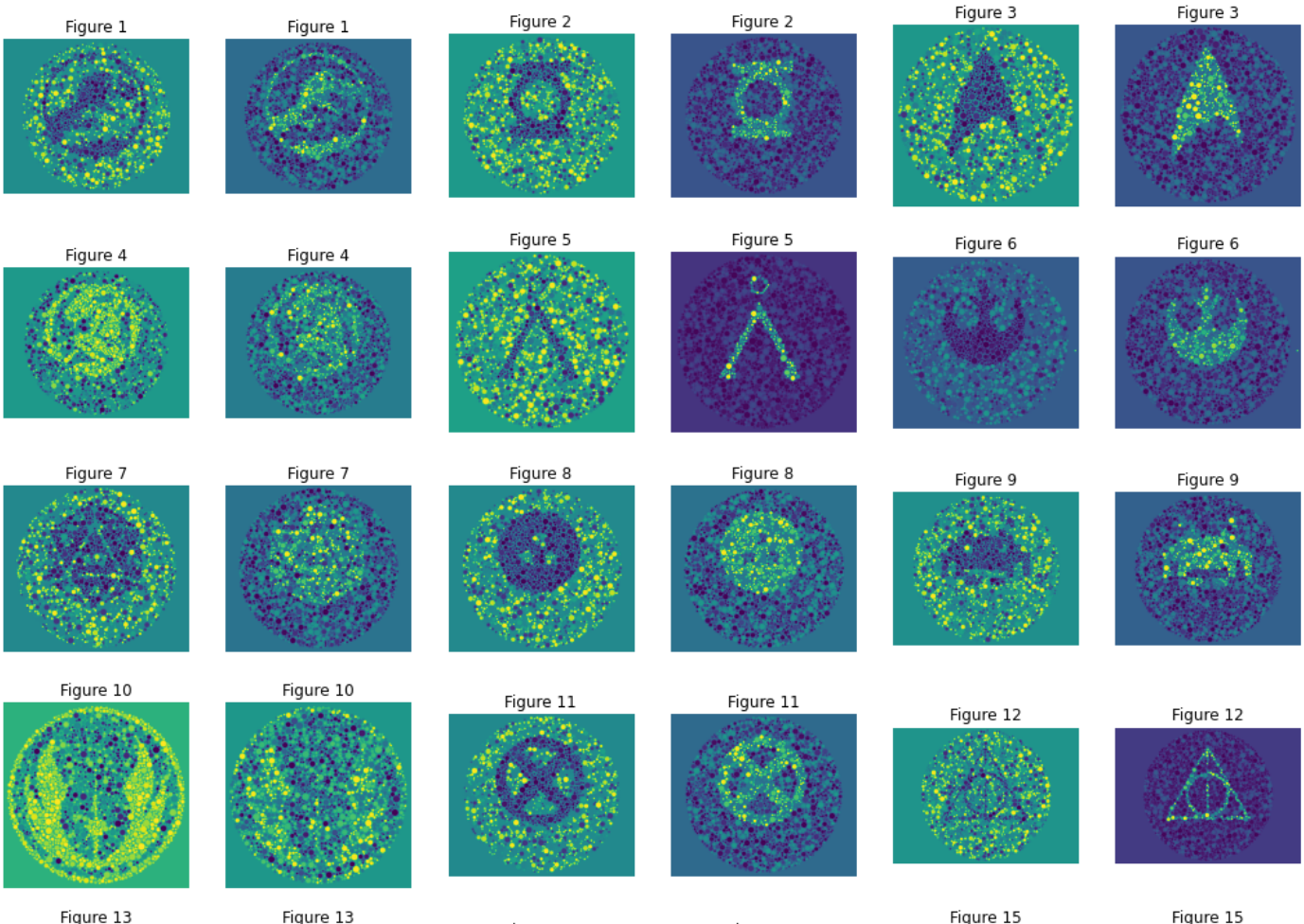
```
def afficherImage(x):
    '''Affiche '''
    Data=image.imread("assets/ishihara-"+str(x)+".png")
    plt.imshow(Data)
    plt.show()

def loadImageAndReshape(x):
    '''Load imageX et reshape'''
    Data=image.imread("assets/ishihara-"+str(x)+".png")
    shapeData=Data.shape[0:2]
    R=np.ravel(Data[:,:,0])
    V=np.ravel(Data[:,:,1])
    B=np.ravel(Data[:,:,2])
    dim=R.shape[0]
    # Les trois canaux sont vectorises et concatenes
    X=np.zeros([dim,3])
    X[:,0]=R
    X[:,1]=V
    X[:,2]=B
    return X,shapeData

def ACP(X,old_shape):
    '''Retournes les images des 3 composantes de X et
```

```
    la shape de la photo avant transformation'''
    M=np.cov(X.T)
    cof=np.corrcoef(M)
    eigen_values,P=np.linalg.eig(M)
    Images=[]
    for comp in range(3):
        new_P=P.T[comp,:]
        new_X=np.ravel(np.dot(new_P,X.T))
        new_X_im=new_X.reshape(old_shape)
        Images.append(new_X_im)
    return Images

def afficherLesComposantes(debut,fin):
    '''Affiche les 2 composantes des images du quizz'''
    fig, axs = plt.subplots(20,6)
    fig.set_figheight(64)
    fig.set_figwidth(18)
    for x in range(debut,fin):
        image,shape=loadImageAndReshape(x)
        lesCompo=ACP(image,shape)
        axs[(x-debut)*2//6,(x-debut)*2%6].imshow(lesCompo[1])
        axs[(x-debut)*2//6,(x-debut)*2%6].set_axis_off()
        axs[(x-debut)*2//6,(x-debut)*2%6].set_title('Figure '+str(x))
        axs[((x-debut)*2+1)//6,((x-debut)*2+1)%6].imshow(lesCompo[2])
        axs[((x-debut)*2+1)//6,((x-debut)*2+1)%6].set_axis_off()
        axs[((x-debut)*2+1)//6,((x-debut)*2+1)%6].set_title(('Figure '+str(x)))
    axs[19,5].set_axis_off()      #derniere
    plt.show()
```

Quizz

In [130…

```
afficherLesComposantes(1,61)
```



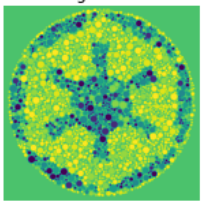
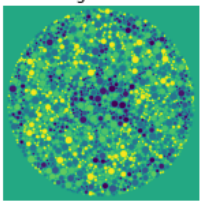



Figure 14

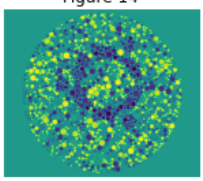



Figure 14

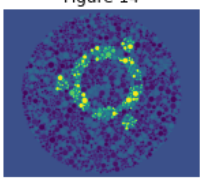
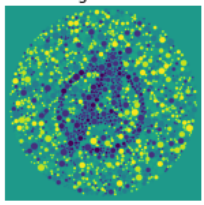
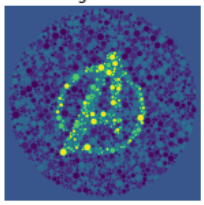





Figure 16

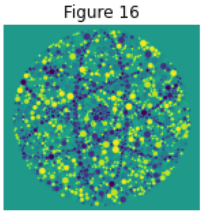



Figure 16

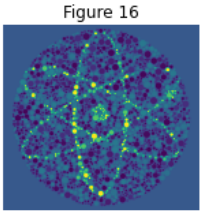



Figure 17

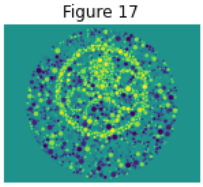



Figure 17

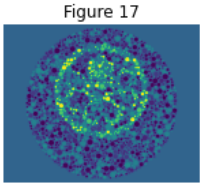



Figure 18

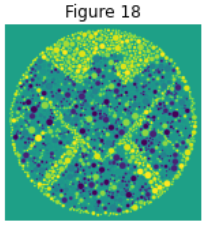



Figure 18

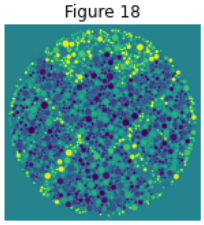



Figure 19

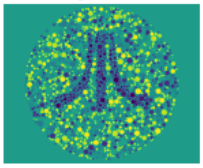



Figure 19

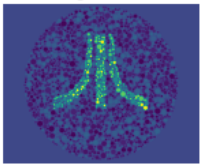



Figure 20

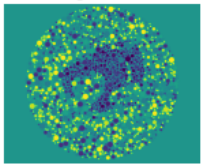



Figure 20

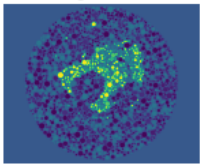



Figure 21

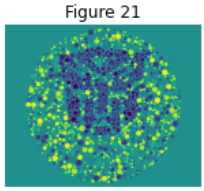



Figure 21

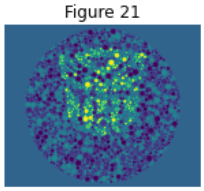



Figure 22

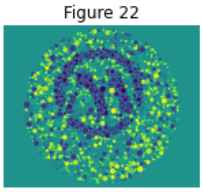



Figure 22

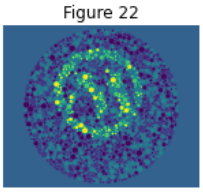



Figure 23

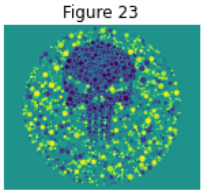



Figure 23

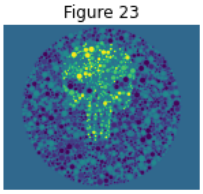



Figure 24

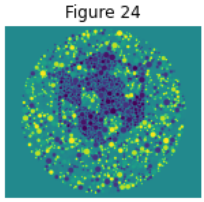



Figure 24

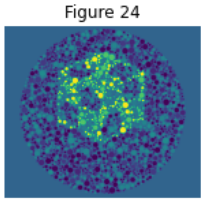



Figure 25

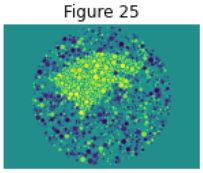



Figure 25

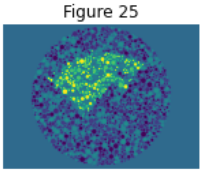



Figure 26

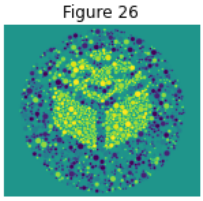



Figure 26

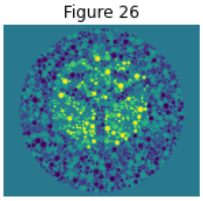



Figure 27

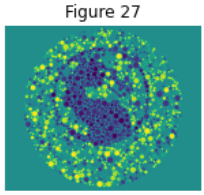



Figure 27

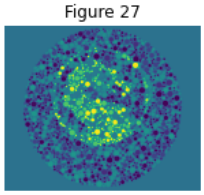



Figure 28

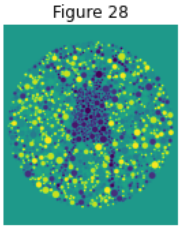



Figure 28

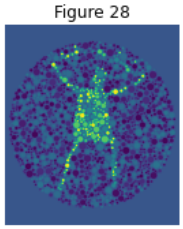



Figure 29

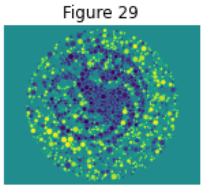



Figure 29

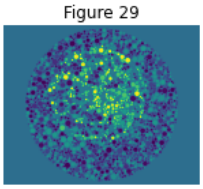



Figure 30

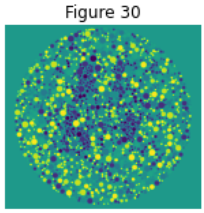



Figure 30

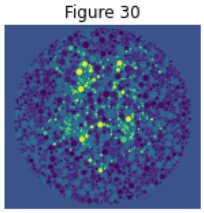



Figure 31

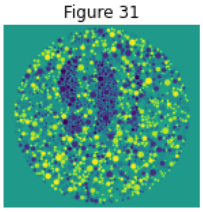



Figure 31

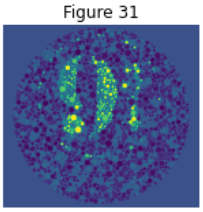



Figure 32

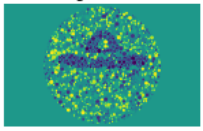



Figure 32

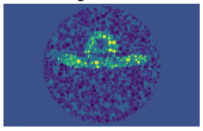



Figure 33

Figure 33

Figure 34

Figure 34

Figure 35

Figure 35

Figure 36

Figure 36

Figure 37

Figure 37

Figure 38

Figure 38

Figure 39

Figure 39

Figure 40

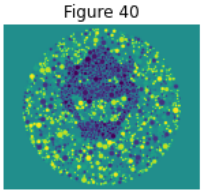


Figure 40

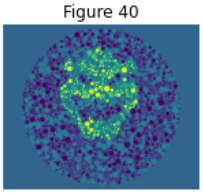


Figure 41

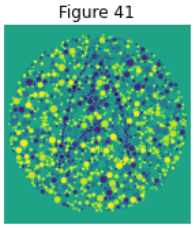


Figure 41

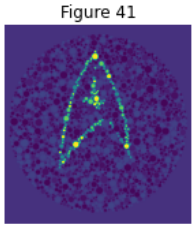


Figure 42

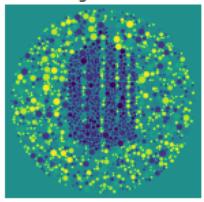


Figure 42

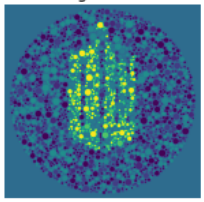


Figure 43

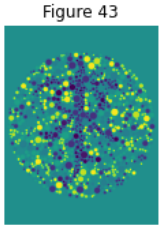


Figure 43

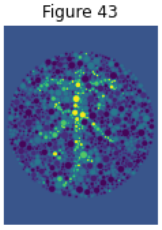


Figure 44

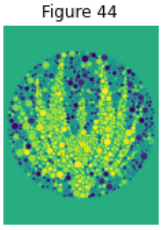


Figure 44

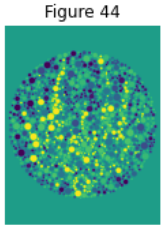


Figure 45

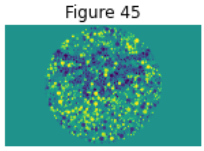


Figure 45

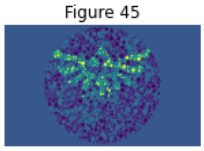


Figure 46

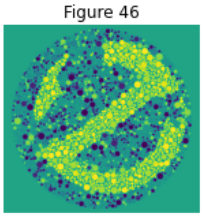


Figure 46

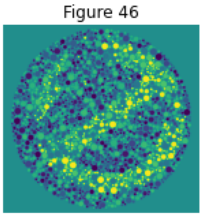


Figure 47

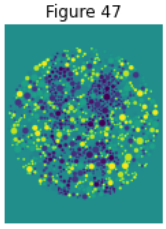


Figure 47

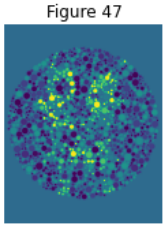


Figure 48

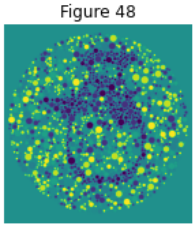


Figure 48

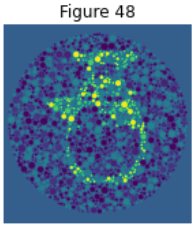


Figure 49

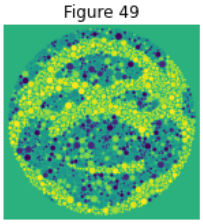


Figure 49

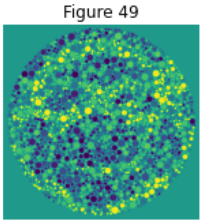


Figure 50

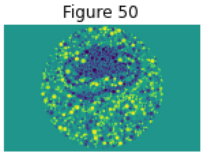


Figure 50

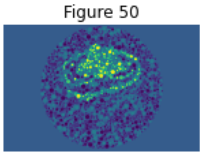


Figure 51

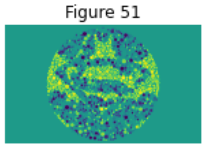


Figure 51

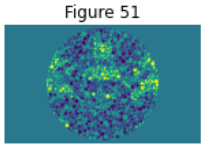


Figure 52

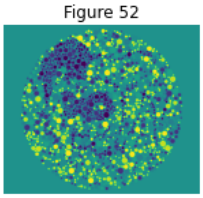


Figure 52

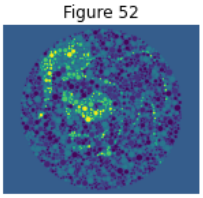


Figure 53

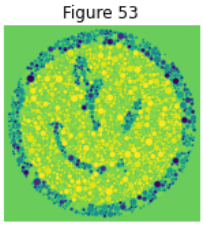


Figure 53

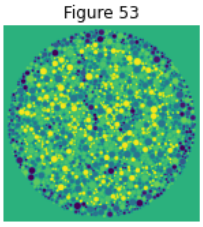


Figure 54

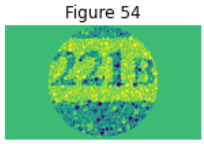


Figure 54

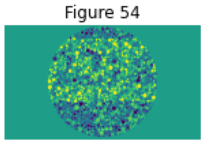


Figure 55

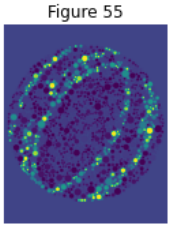


Figure 55

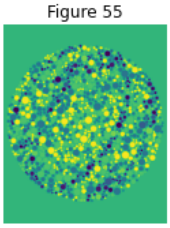


Figure 56

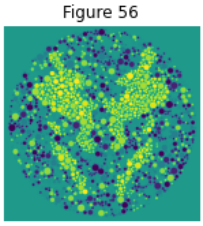


Figure 56

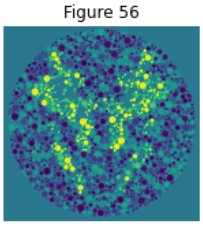


Figure 57

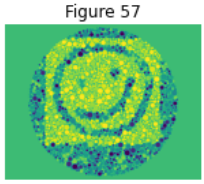


Figure 57

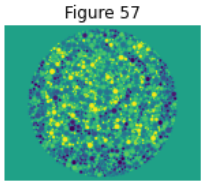


Figure 58

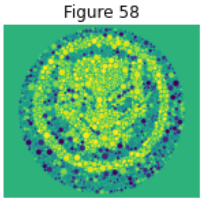


Figure 58

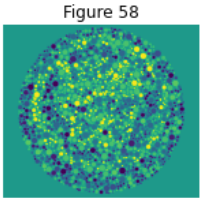


Figure 59

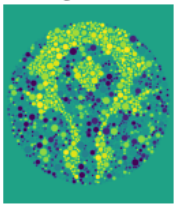


Figure 59

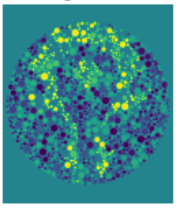


Figure 60

Figure 60