PROJET PYTHON

Nom du projet : Traitement d'image

Présenté par : IMANI Mourad
ICHMAWIN Anas
FANNOUCH Oussama
BOUAZIZ Amine

Encadré par : Ms. SAADI Mostafa

Plan du Projet

Partie I:
Les opérations d'entrée/sortie sur les images

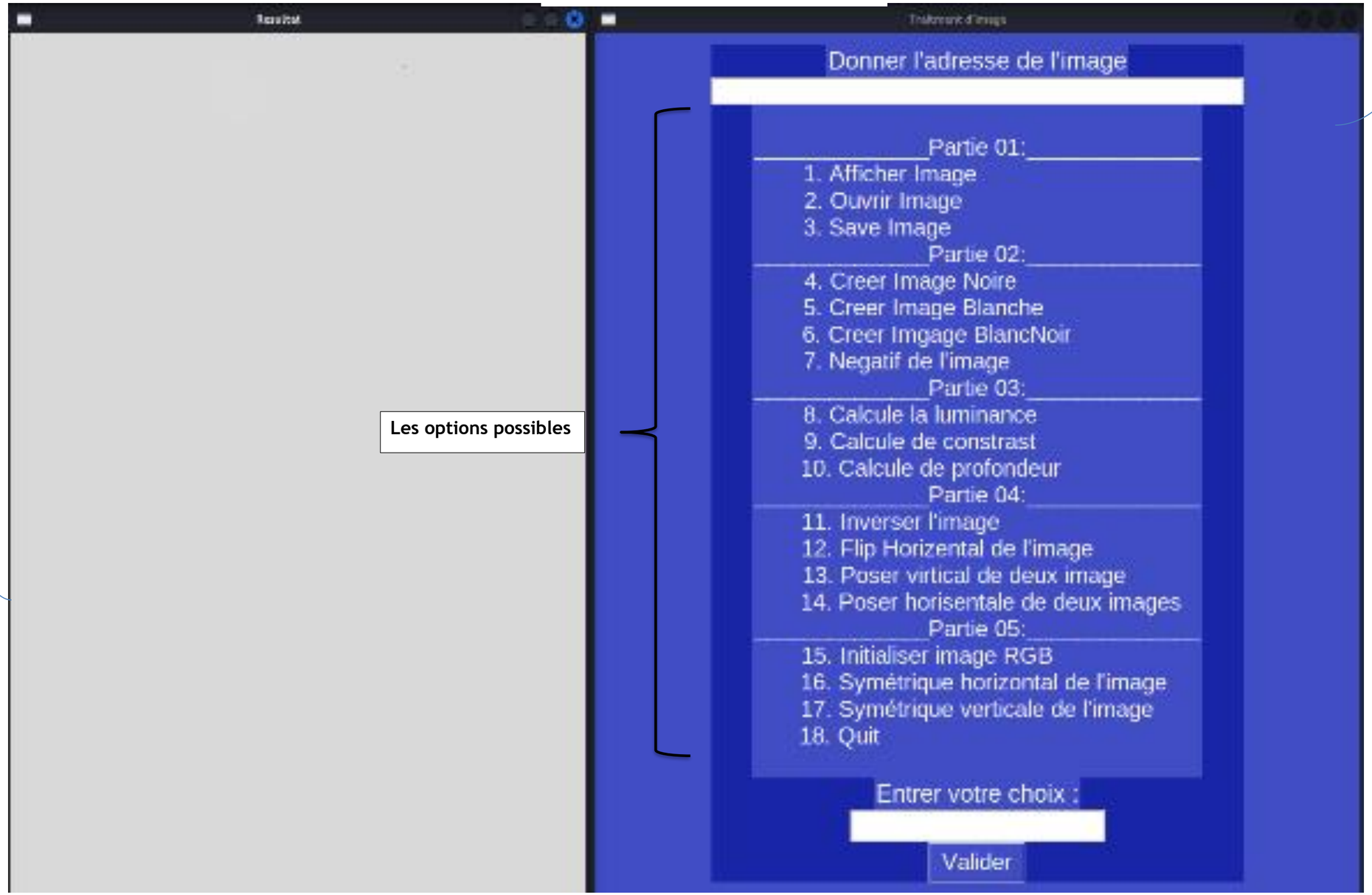
Partie II:
Les images Noir et blanc

Partie III:
Les images en niveau de gris

Partie IV:
Opérations élémentaires sur les images en mode gris

Partie V:
Les images RGB

L’interface



Affichage de résultat

Les options possibles

Interface de commande

Partie I:

Les opérations d'entrée/sortie
sur les images

- AfficherImg
- ouvrirImage
- saveImage

Code Source

```
################## affichage ##################
def AfficherImg(img,m=255,n=0):
    plt.axis("off")
    plt.imshow(img,cmap = 'gray',interpolation="nearest",vmax=m,vmin=n)
    # J'ai chosi m,n comme valeur pour assurer l'accer au choix des valeurs vmax
    # pour assurer l'accer au choix des valeurs vmax et vmin ce qui donne le fait de choisir entre noire blanc et rgb
    plt.show()
```

Explication

La fonction AfficherImg affiche l'image associée à la matrice donnée en argument sur l'écran.

Code Source

```
##################### open ######################
def ouvrirImage(chemin):
    img=plt.imread(chemin)
    return img
```

Explication

La fonction ouvrirImage prend en argument un chemin d'image et retourne une matrice contenant son codage.

Code Source

```
#################### saveimage ####################
def saveImage(img):
    plt.imsave("image1.png",img)
```

Explication

La fonction saveImage prend en argument une matrice représentant une image et enregistre cette image sur le disque dur.

Partie II:

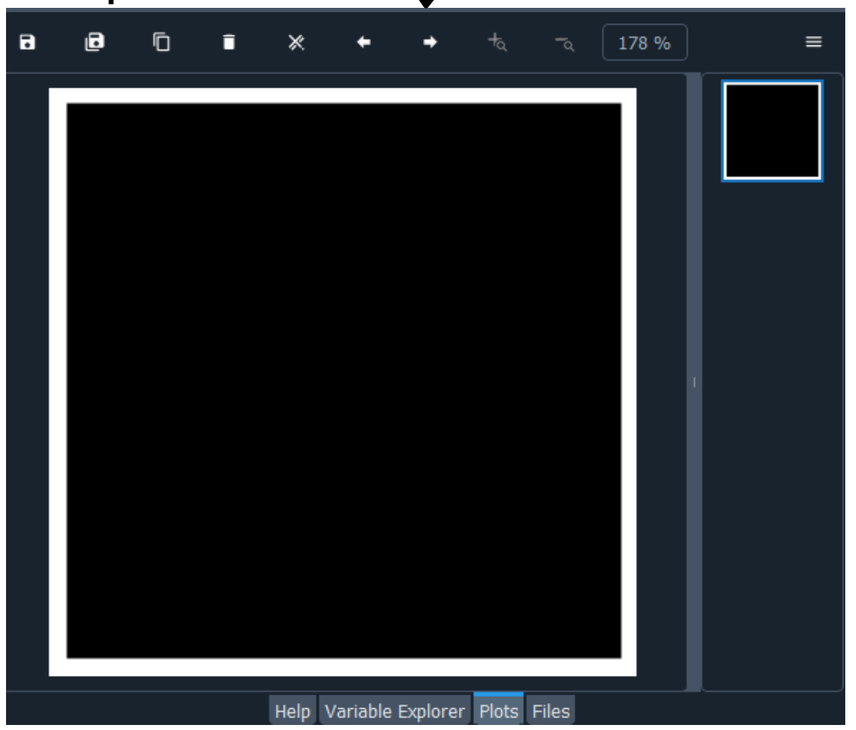
Les images Noir et blanc

- image_noire
- image_blanche
- creerImgBlancNoir
- negatif

Code Source

```
##################### Q1 ######################
def image_noire(h, l):
    img=np.zeros((h,l))
    return img
```

Compilation



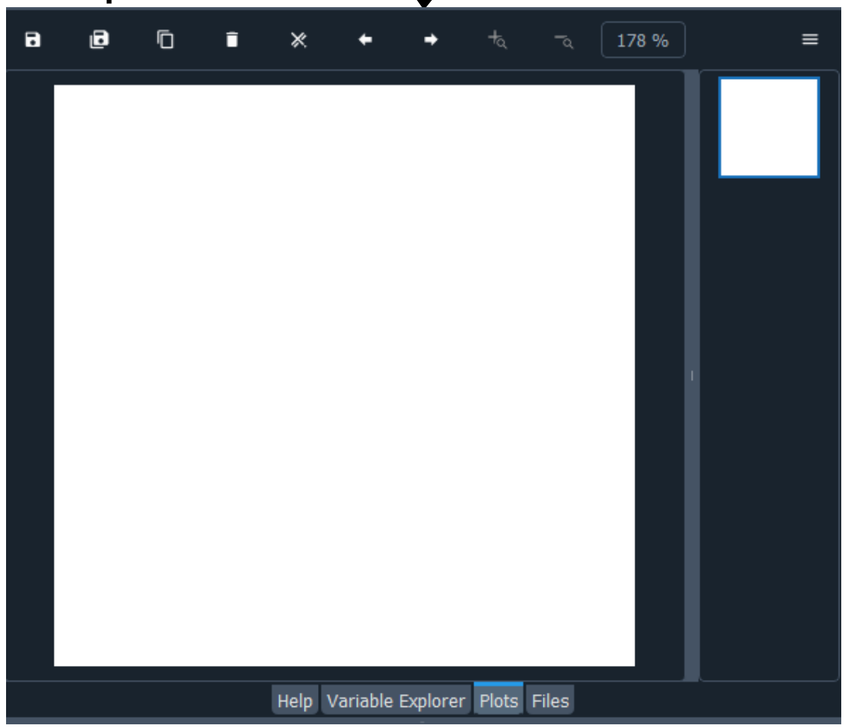
Explication

La fonction image_noire crée une image noire de dimensions spécifiées en entrée (hauteur h et largeur l). Elle retourne une matrice l lignes * h colonnes, dans laquelle chaque pixel est initialisé à la valeur 0.

Code Source

```
##################### Q2 ####################
def image_blanche(h, l):
    img=np.ones((h,l))
    return img
```

Compilation



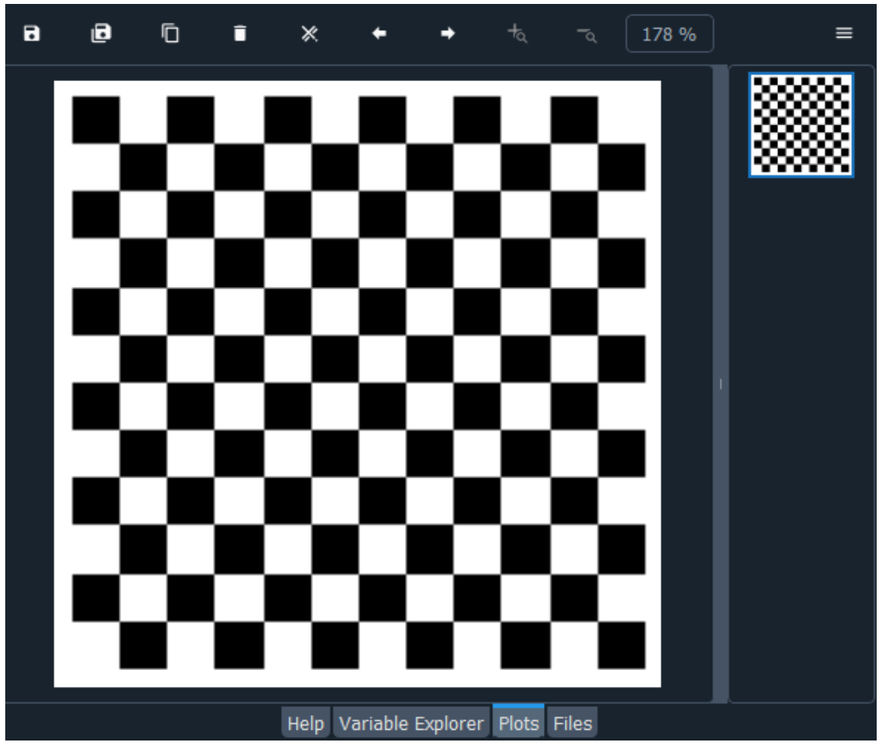
Explication

La fonction image_blanche crée une image blanche de dimensions spécifiées en entrée (hauteur h et largeur l). Elle retourne une matrice l lignes * h colonnes, dans laquelle chaque pixel est initialisé à la valeur 1.

Code Source

```
################## Q3 ##################
def creerImgBlancNoir(h,l):
    img=np.array([[(i+j)%2 for i in range(l)]for j in range(h)])
    return img
```

Compilation



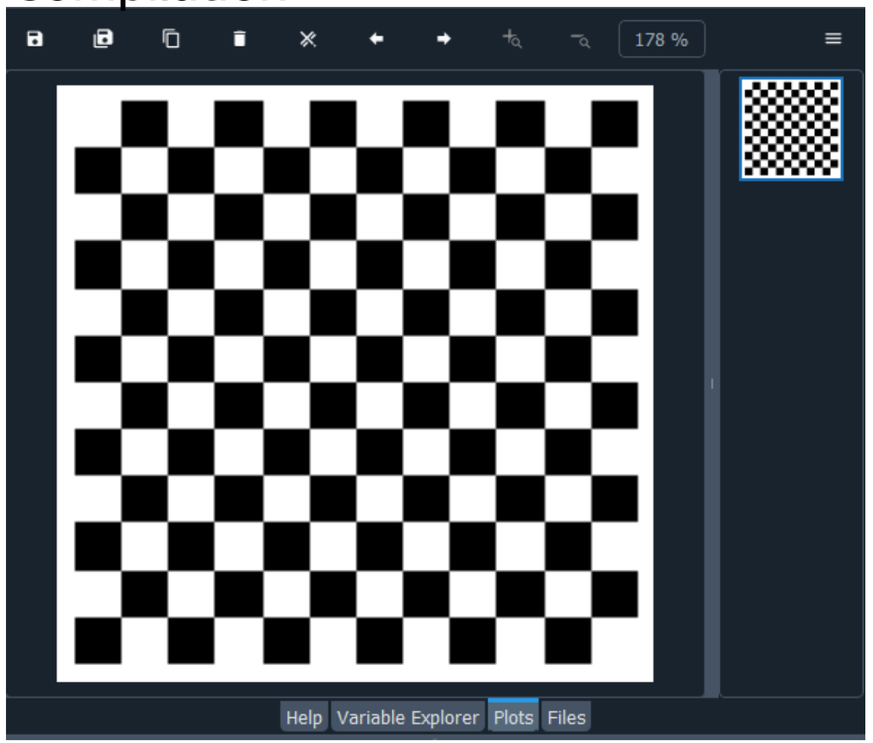
Explication

La fonction creerImgBlancNoir génère une image en noir et blanc. Cette image peut être représentée sous forme d'une matrice de pixels, chaque pixel ayant une valeur de 0 (noir) ou 1 (blanc).

Code Source

```
###################### Q4 ######################
def negatif(Img):
    img=np.array([[1-j for j in i] for i in Img])
    return img
```

Compilation



Explication

La fonction "negatif" prend en entrée une matrice "Img" représentant une image et renvoie une image négative en inversant les valeurs de la matrice "Img" (les valeurs 0 deviennent 1 et les valeurs 1 deviennent 0).

Partie III:

Les images en niveau de gris

- luminance
- constrast
- profondeur
- ouvrir

Code Source

```python
def luminance(Img):
    s=0
    for i in range(len(Img)):
        for j in range(len(Img[0])):
            s+=Img[i][j][0]
    return s/(len(Img)*len(Img[0]))
```

Explication

La fonction luminance calcule la luminance de l'image , qui est une mesure de la luminosité perçue par l'œil humain.Elle renvoie enfin une valeur numérique représentant la luminance de l'image.

Compilation

```
la valeur de luminace est : 119.22241111111
```

Code Source

```python
################## Q2 ##################
def constrast(Img):
    s=0
    for i in Img:
        for j in i:
            s+=(j[0]-luminance(Img))**2
    # au lieu de transformer à deux dimension nous avons pris un j[0]
    # puisque les trois en la meme valeur
    return (1/(len(Img)*len(Img[0])))*s
```

Explication

La fonction contraste calcule le contraste de l'image , qui est une mesure de la variance des niveaux de gris de l'image. Le contraste est donc déterminé en comparant les niveaux de gris de chaque pixel de l'image.

Compilation

```
la valeur de constrast est : 2522.5170332980347
```

Code Source

```
################## Q3 ##################
def profondeur(Img):
    if Img.ndim==2:
        if Img.max()==1:
            return 1
        else: return 8
    else:
        return 24
    #dans une image d’une profondeur de 1 bit par pixel,
    #les pixels peuvent prendre deux valeurs possibles : noir et blanc.
    #Une image en mode Niveaux de gris à une profondeur de 8 bits par pixelS
    #Les images RGB sont constituées de 3 couches de couleur.
    #Les images RVB 8 bits par couche sont parfois appelées images 24 bits
    #(8 bits x 3 couches = 24 bits de données pour chaque pixel.
```

Compilation

```
la valeur de la profondeur est :  24
```

Explication

La fonction profondeur renvoie la valeur maximale des pixels dans l'image "Img". Cette valeur est également appelée "profondeur de couleur" ou "profondeur de bits", et elle détermine le nombre de bits utilisés pour représenter chaque pixel de l'image.

Partie IV:

Opérations élémentaires sur les images en mode gris

- inverser
- flipH
- poserV
- poserH

Code Source

```
################### Q5 ###################
def inverser(img):
    inv=np.array([[255-j for j in i] for i in img])
    return inv
```

Compilation

Explication

La fonction inverser prend en entrée une matrice d'une image et renvoie une image inversée de cette image. Pour créer l'image inversée, la fonction calcule la valeur inverse de chaque pixel de l'image

Code Source :

```
#################### Q6 ####################
def flipH(img):
    t=[]
    for i in range(len(img)):
        t=t+[img[i][::-1]]
    return t
```

Compilation :

Explication

La fonction flipH prend en entrée une matrice image et renvoie une version de cette image qui a été symétrique par rapport à un axe vertical passant par le milieu de l'image.

Code Source

```
##################### Q7 ######################
def poserV(img1,img2):
    img3=[]
    if profondeur(img1)==profondeur(img2):
        if len(img1[0])==len(img2[0]):
            img3+=list(img1)+list(img2)
            np.array(img3)
    return img3
```

Compilation

Explication

La fonction poserV prend en entrée deux images de même largeur et profondeur, et renvoie une nouvelle image qui est obtenue en superposant verticalement les deux images.

Code Source

```
################### Q8 ###################
def poserH(img1, img2):
    img3=[[[0] for i in range(len(img1[0])+len(img2[0]))]for j in range(len(img1))]
    if profondeur(img1)==profondeur(img2):
        if len(img1)==len(img2):
            for i in range(len(img1)):
                img3[i]= list(img1[i])+list(img2[i])
    return np.array(img3)
```

Compilation

Explication

La fonction poserH prend en entrée deux images de même hauteur et profondeur, et renvoie une nouvelle image qui est obtenue en superposant horizontalement 2eme image à droite de 1 ere image.

Partie V:

Les images RGB

- initImageRGB
- symetrie
- grayscale

Code Source

```
M=[[[210, 100, 255],[100, 50, 255],[90, 90, 255],[90, 90, 255],[90, 90, 255],[90, 80, 255]],
[[190, 255,89],[ 201, 255,29],[200, 255,100],[100, 255,90],[20, 255,200], [100, 255,80]],
[[255,0, 0],[ 255,0, 0],[255,0, 0],[255,0, 0],[255,0, 0], [255,0, 0]]]

print(M[0][1][1])
print(M[1][0][1])
print(M[2][1][0])
```

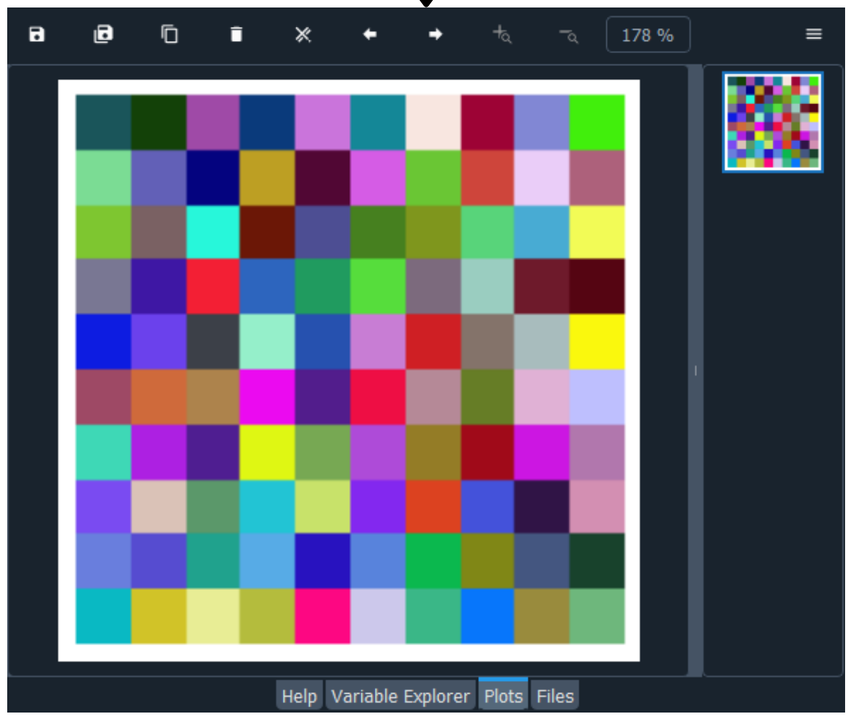
Compilation

```
l'element M[0][1][1]= 50
l'element M[1][0][1]= 255
l'element M[2][1][0]= 255
```

Code Source

```
################### Q3 ###################
def initImageRGB(h,l):
    img=[[[random.randrange(255) for i in range(3)] for j in range(l) ]for k in range(h)]
    return img
# list comprehension
```

Compilation



Explication

La fonction initImageRGB initialise et renvoie une matrice à trois dimensions de manière aléatoire, cette matrice est représente une image couleur au format RGB (rouge, vert, bleu).

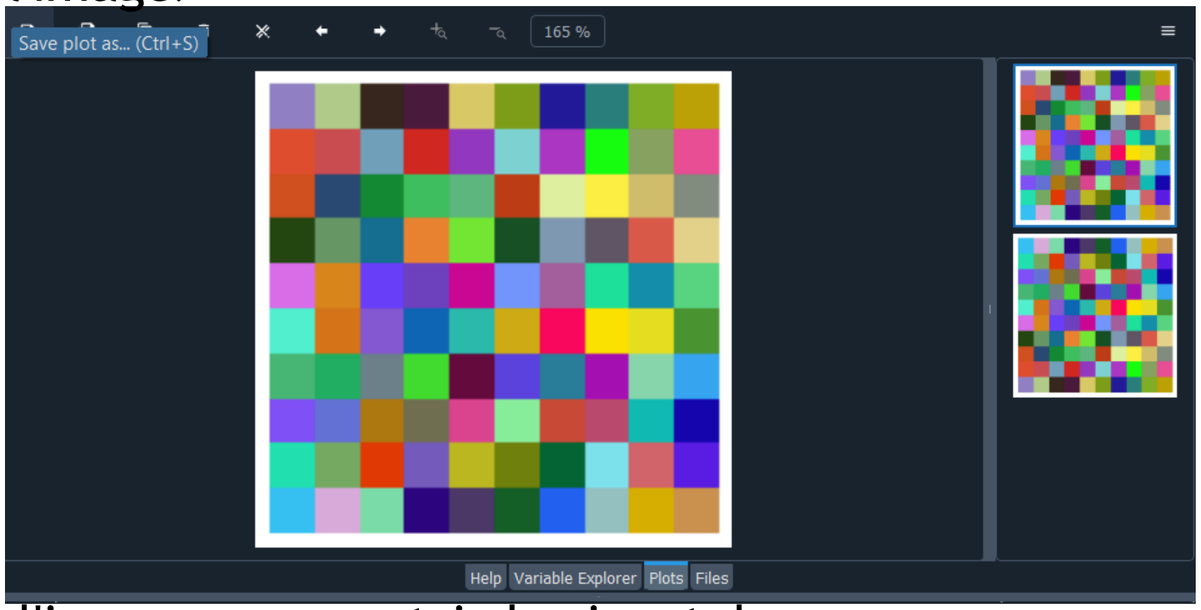
Code Source

```python
################### Q4 ###################
def symetrieH(img):
    return img[::-1]
```

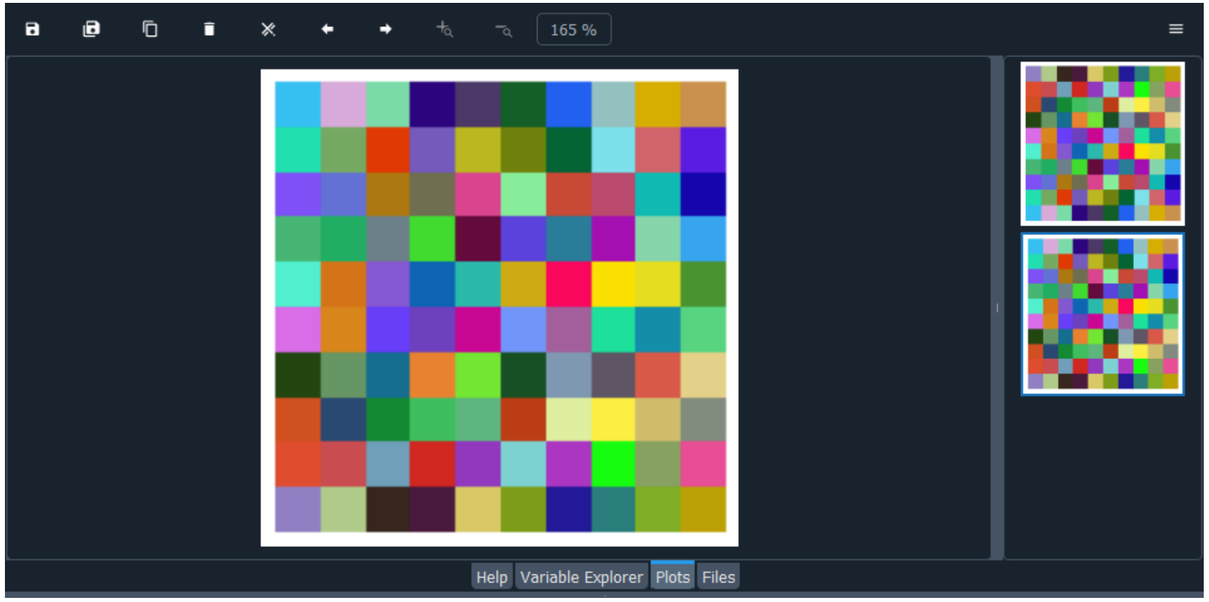
Explication

La fonction symetrie prend en entrée une matrice et renvoie une version symétrique de cette image par rapport à l'axe horizontal.

Compilation l'image:



l'image par symetrie horizontale



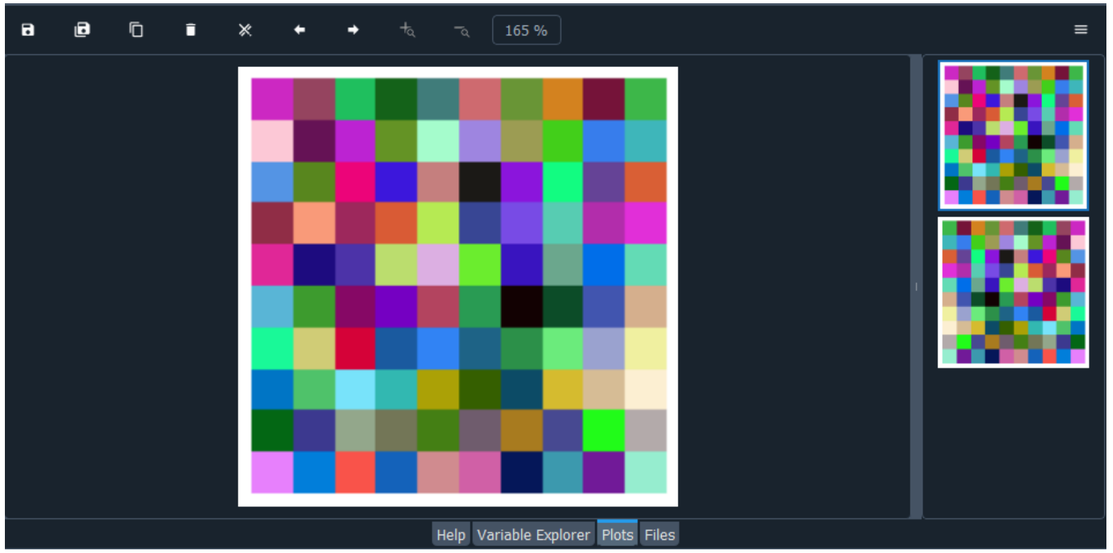
Code Source

```
################### Q5 ###################
def symetrieV(img):
    img3=[[[0] for i in range(len(img[0]))]for j in range(len(img))]
    for i in range(len(img)):
        img3[i]=list(img[i][::-1])
    return np.array(img3)
# on a crée une list dans laquelle
# on a affecté les ligne inversé de image initiale
```

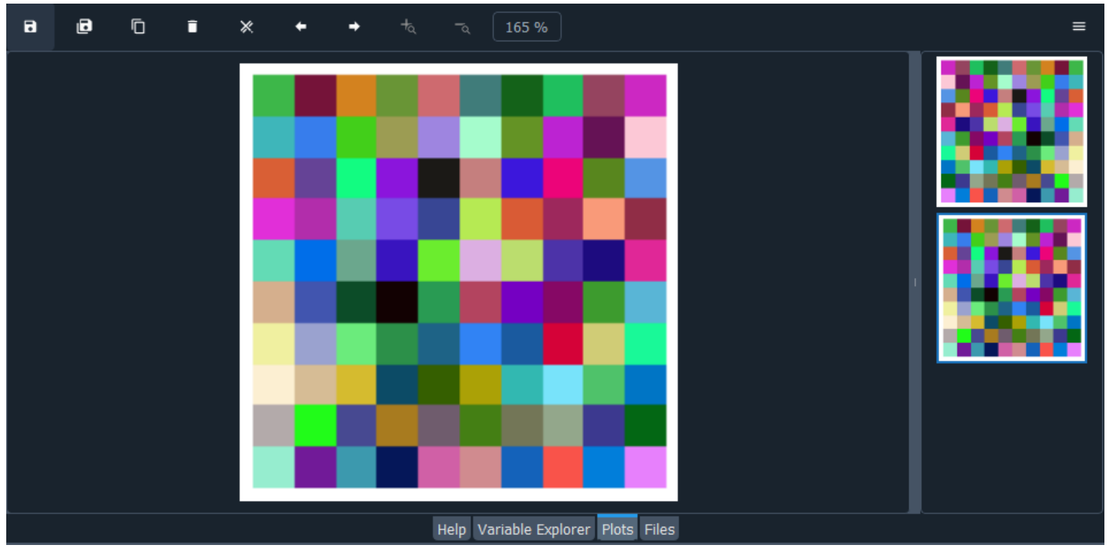
Explication

La fonction symetrie prend en entrée une matrice et renvoie une version symétrique de cette image par rapport à l'axe verticale.

Compilation l'image:



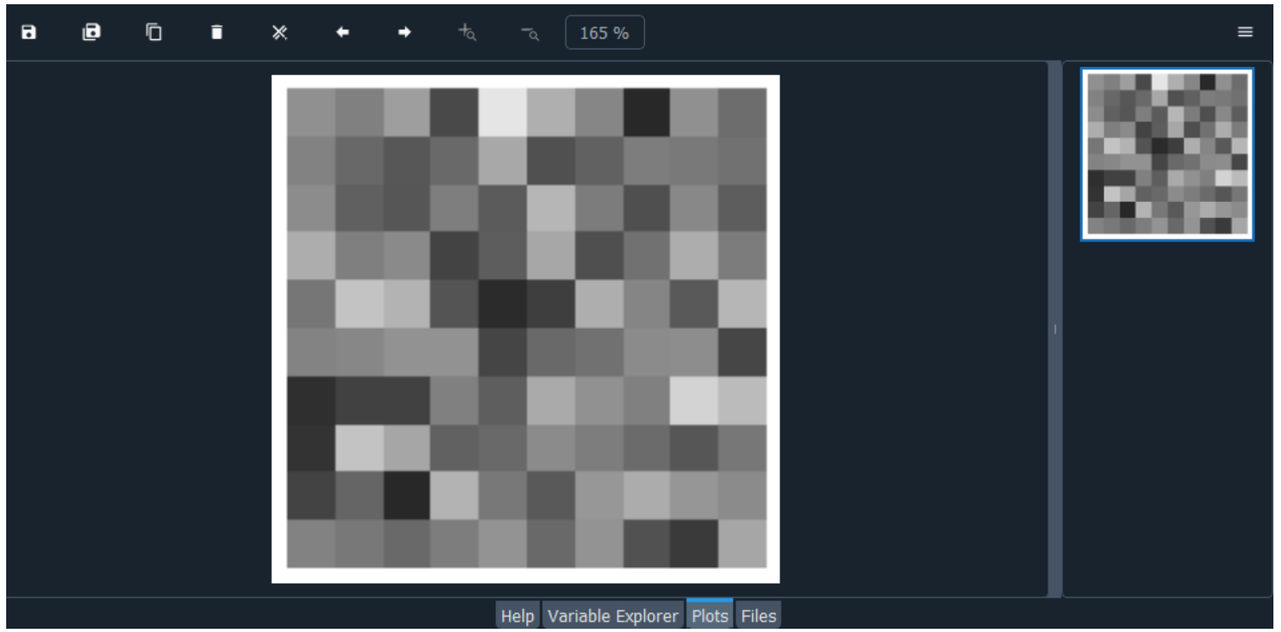
l'image par symetrie verticale



Code Source

```
################## Q6 ##################
def grayscale(imageRGB):
    e=[[floor(((j.max()+j.min())/2))for j in i]for i in np.array(imageRGB)]
    return e
AfficherImg(grayscale(M))
print(grayscale(M))
```

Compilation



Explication

La fonction Grayscale prend en entrée une matrice d'image couleur RGB, et renvoie une version en niveaux de gris de cette image.

Bilan de travail

Apres avoir reçu le travail nous avons décidé que chacun de nous va faire tout le travail on se qui concerne la création des fonctions après on se réunit pour les discuter et corrigé après choisir les codes qu'on a apprécié.

Pour la préparation du reste on a diviser les taches comme suit :

- préparation du fichier PowerPoint.
- Code main et interface graphique.
- préparation du rapport Word.
- Fichier exécutable.