Solution travaux pratiques N°6

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

#======== Affichage des résultats sur une matrice ==================
image = np.array([
    [0, 0, 0, 0, 0],
    [0, 1, 1, 1, 0],
    [0, 1, 1, 1, 0],
    [0, 1, 1, 1, 0],
    [0, 0, 0, 0, 0],
], dtype=np.uint8)

# Conversion à une image au format binaire (0 et 255)
binary_image = image * 255
#binary_image = image
# Élément structurant (carré de 3x3)
kernel = np.ones((3, 3), np.uint8)
# Opérations morphologiques
erosion = cv2.erode(binary_image, kernel, iterations=1)
dilatation = cv2.dilate(binary_image, kernel, iterations=1)
ouverture = cv2.morphologyEx(binary_image, cv2.MORPH_OPEN, kernel)
fermeture = cv2.morphologyEx(binary_image, cv2.MORPH_CLOSE, kernel)

print('============= Image Origine =================')
print(binary_image)
print('============= Kernel =================')
print(kernel)
print('============= erosion =================')
print(erosion)
print('============= dilatation =================')
print(dilatation)
print('============= ouverture =================')
print(ouverture)
print('============= fermeture =================')
print(fermeture)



#======== Affichage des résultats sur une image ======================

binary_image = cv2.imread('D:\Enseignements\Maamar\TP-06\img\morph.png', cv2.IMREAD_GRAYSCALE)
# Opérations morphologiques
erosion = cv2.erode(binary_image, kernel, iterations=1)
dilatation = cv2.dilate(binary_image, kernel, iterations=1)
ouverture = cv2.morphologyEx(binary_image, cv2.MORPH_OPEN, kernel)
fermeture = cv2.morphologyEx(binary_image, cv2.MORPH_CLOSE, kernel)

titles = ['Image d\'origine', 'Érosion', 'Dilatation', 'Ouverture', 'Fermeture']
images = [binary_image, erosion, dilatation, ouverture, fermeture]

plt.figure(figsize=(12, 6))
for i in range(5):
    plt.subplot(2, 3, i+1)
    plt.imshow(images[i], cmap='gray')
    plt.title(titles[i])
    plt.axis('off')
plt.tight_layout()
plt.show()
```

```python
#========================= Contour ==================================================
# Charger l'image en niveaux de gris
image = cv2.imread('D:\Enseignements\Maamar\TP-06\img\Contour.png', cv2.IMREAD_GRAYSCALE)

# Binarisation de l'image (conversion en noir et blanc)
_, binary_image = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)

# Création d'un élément structurant (taille et forme personnalisables)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))

# Dilatation et Erosion
dilated = cv2.dilate(binary_image, kernel, iterations=1)
eroded = cv2.erode(binary_image, kernel, iterations=1)

# Extraction des contours
contours = cv2.subtract(dilated, eroded)

# Affichage des résultats
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.title("Image originale")
plt.imshow(image, cmap='gray')

plt.subplot(1, 3, 2)
plt.title("Image binaire")
plt.imshow(binary_image, cmap='gray')

plt.subplot(1, 3, 3)
plt.title("Contours extraits")
plt.imshow(contours, cmap='gray')

plt.tight_layout()
plt.show()

#==================================Remplissage de trous ================================
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Charger l'image en niveaux de gris
image = cv2.imread('D:\Enseignements\Maamar\TP-06\img\Remplissage.png', cv2.IMREAD_GRAYSCALE)

# Binarisation de l'image (conversion en noir et blanc)
_, binary_image = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)

# Création d'un élément structurant (taille et forme personnalisables)
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))

# Appliquer la dilatation pour combler les trous
k = 10
dilated = cv2.dilate(binary_image, kernel, iterations=k)

# Appliquer l'érosion pour restaurer les contours
filled_image = cv2.erode(dilated, kernel, iterations=1)

# Affichage des résultats
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.title("Image originale")
plt.imshow(image, cmap='gray')

plt.subplot(1, 3, 2)
plt.title("Image binaire")
plt.imshow(binary_image, cmap='gray')

plt.subplot(1, 3, 3)
plt.title("Image avec trous remplis")
plt.imshow(filled_image, cmap='gray')

plt.tight_layout()
plt.show()
```