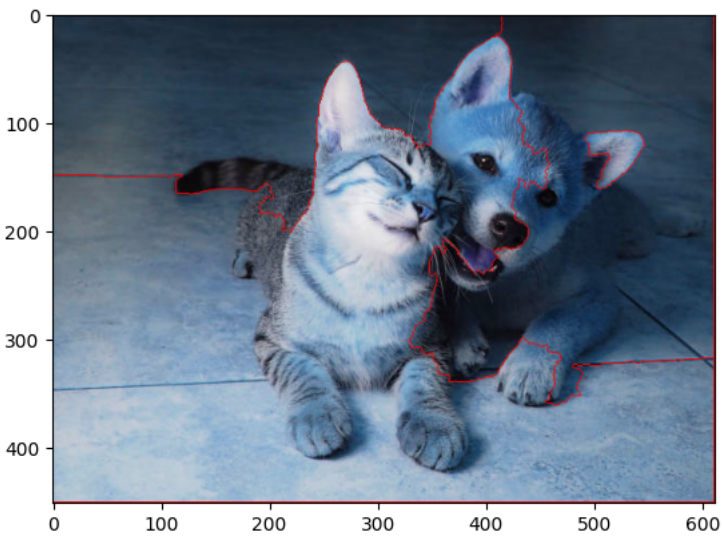
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image with the correct file extension
image = cv2.imread("/content/sample_data/catdog.jpg")

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Threshold the image to create markers (you can adjust the threshold value)
_, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

# Perform morphological operations to remove noise
kernel = np.ones((7, 7), np.uint8)
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=2)

# Create sure background and sure foreground markers
sure_bg = cv2.dilate(opening, kernel, iterations=3)
dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
_, sure_fg = cv2.threshold(dist_transform, 0.6 * dist_transform.max(), 255, cv2.THRESH_BINARY)

# Subtract sure foreground from sure background to get the unknown region
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg, sure_fg)

# Label markers for the watershed algorithm
_, markers = cv2.connectedComponents(sure_fg)
markers = markers + 1
markers[unknown == 255] = 0

# Apply the watershed algorithm
cv2.watershed(image, markers)

# Mark boundaries with a red color
image[markers == -1] = [255, 0, 0]

# Display the result
plt.imshow(image)
plt.axis("on")
plt.show()
```




```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow  # This is for displaying OpenCV images in Colab.

# Load the image
image = cv2.imread("/content/sample_data/dog.jpg")

# Reshape the image to a 2D array of pixels
pixel_values = image.reshape((-1, 3))
```

```
pixel_values = image.reshape((-1, 3))

# Convert to Float32 for K-Means clustering
pixel_values = np.float32(pixel_values)

# Define criteria and apply K-Means clustering
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 180, 0.4)
K = 6  # Number of clusters (you can adjust this)
_, labels, centers = cv2.kmeans(pixel_values, K, None, criteria, 50, cv2.KMEANS_RANDOM_CENTERS)

# Convert back to 8-bit values
centers = np.uint8(centers)

# Map the labels to the centers to segment the image
segmented_image = centers[labels.flatten()]

# Reshape the segmented image back to its original shape
segmented_image = segmented_image.reshape(image.shape)

# Display the segmented image using cv2_imshow
cv2_imshow(segmented_image)
```