Aim: To get familiar with OpenCV operations

Note: Use matplotlib or OpenCV to display images and write the code after the #Answer comment

In [7]:
```
import warnings
warnings.filterwarnings('ignore')

import cv2
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import matplotlib.patches as patches
```

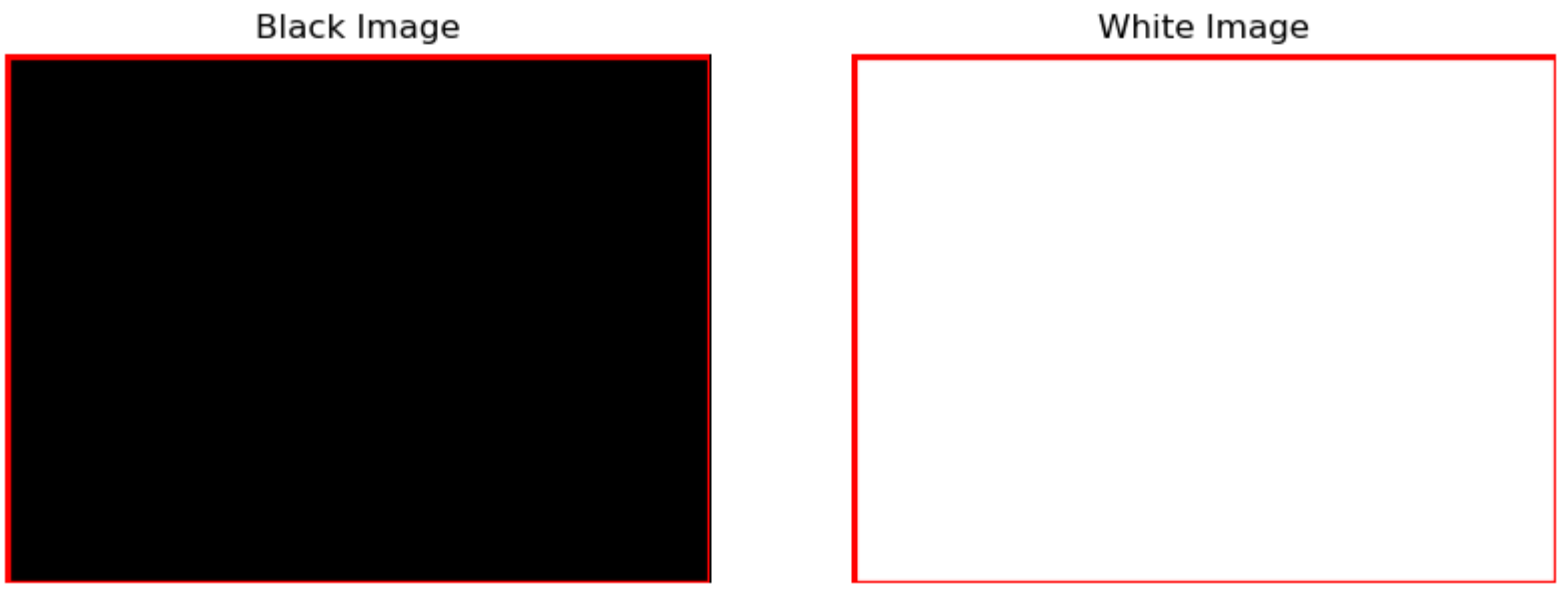
Question 1

In [2]:
```
#read image opencv-logo
#display shape and size of the image

#Answer

img = cv2.imread('./opencv-logo.png')

img_shape = img.shape
img_size = img.size

print(f'Shape of the Image : {img_shape}')
print(f'Size of the Image : {img_size}')
```

```
Shape of the Image : (739, 600, 3)
Size of the Image : 1330200
```

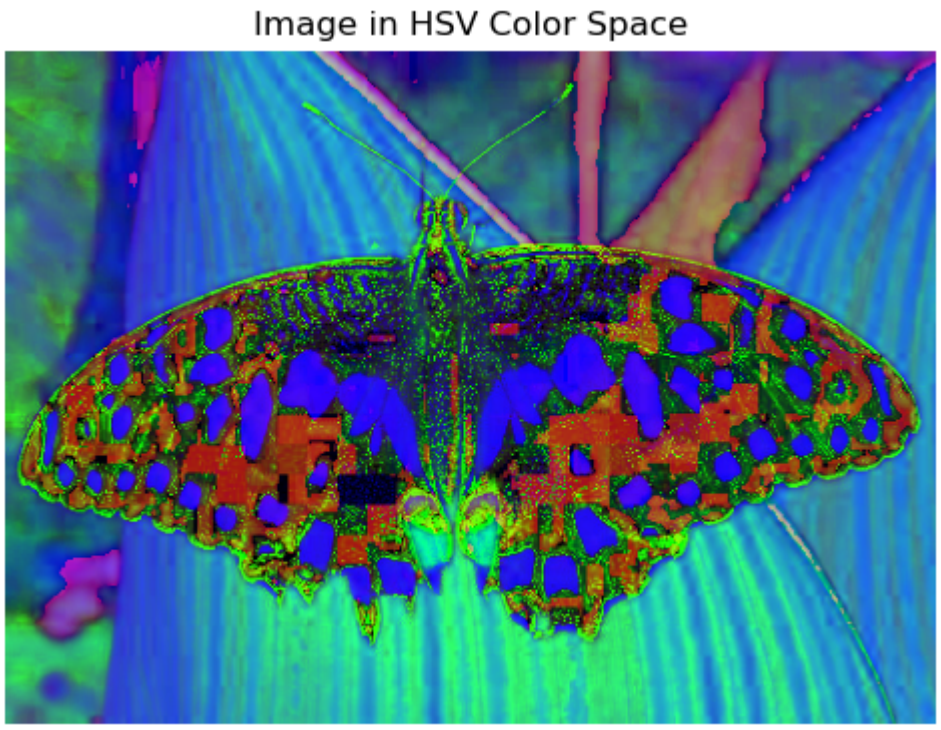
Question 2

In [9]:
```
#create a numpy array of zeroes that is 150 pixels tall, 200 pixels wide
#display this black image
#Similarly create a numpy array on ones with above dimensions and display the white image

#Answer

black_img = np.zeros((150, 200))
white_img = np.ones((150, 200))

fig = plt.figure(figsize=(10, 5))

ax1 = plt.subplot(1, 2, 1)
ax1.imshow(black_img, cmap='gray', vmin=0, vmax=1)
ax1.set_title('Black Image')
ax1.axis('off')

border1 = patches.Rectangle((0, 0), black_img.shape[1], black_img.shape[0], linewidth=3, edgecolor='red', facecolor
ax1.add_patch(border1)

ax2 = plt.subplot(1, 2, 2)
ax2.imshow(white_img, cmap='gray', vmin=0, vmax=1)
ax2.set_title('White Image')
ax2.axis('off')

border2 = patches.Rectangle((0, 0), white_img.shape[1], white_img.shape[0], linewidth=3, edgecolor='red', facecolor
ax2.add_patch(border2)

plt.show()
```



Question 3

In [4]:
```
#read the butterfly image and display the r,g,b channels of the image
#Then display the butterfly image in the HSV color space

#Answer

image = cv2.imread('butterfly.jpg')
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

r_channel = image_rgb[:, :, 0]
g_channel = image_rgb[:, :, 1]
b_channel = image_rgb[:, :, 2]

image_hsv = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2HSV)

plt.figure(figsize=(20, 12))
plt.subplot(2, 2, 1)
plt.imshow(image_rgb)
plt.title('Original Image')
plt.axis('off')

plt.subplot(2, 2, 2)
plt.imshow(r_channel, cmap='Reds')
plt.title('Red Channel')
plt.axis('off')

plt.subplot(2, 2, 3)
plt.imshow(g_channel, cmap='Greens')
plt.title('Green Channel')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.imshow(b_channel, cmap='Blues')
plt.title('Blue Channel')
plt.axis('off')
plt.show()

plt.figure(figsize=(6, 6))
plt.imshow(image_hsv)
plt.title('Image in HSV Color Space')
plt.axis('off')
plt.show()
```



Question 4

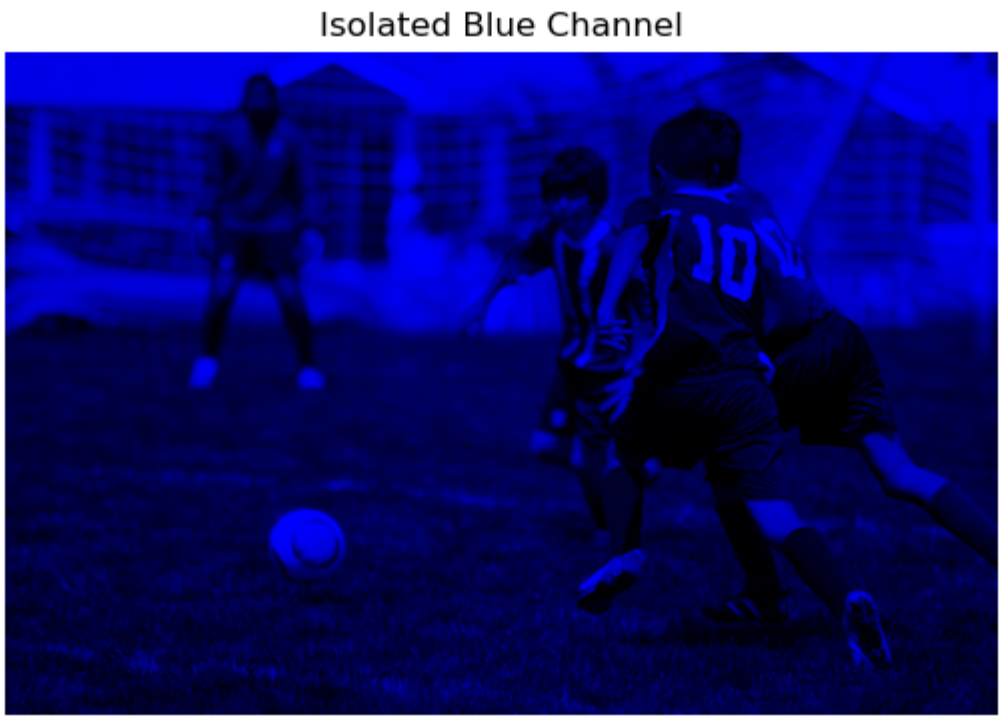
In [5]:
```
#create a 5 by 5 array where every number is a 10
#run the cell below to create an array of random numbers and see if you can figure out how it works.
#what are the largest and smalled values in this array?
#use PIL and matplotlib to read and display the any image of your choice from the given images.
#convert the image to a NumPy Array
#use slicing to set the RED and GREEN channels of the picture to 0, then use imshow() to show the isolated blue cha

#Answer
array_5x5 = np.full((5, 5), 10)
print("5x5 Array where every number is 10:\n", array_5x5)

np.random.seed(0)
random_array = np.random.randint(0, 100, size=(5, 5))
print("Random Array:\n", random_array)

largest_value = np.max(random_array)
smallest_value = np.min(random_array)

print(f"Largest value in the random array: {largest_value}")
print(f"Smallest value in the random array: {smallest_value}")
```

```
5x5 Array where every number is 10:
 [[10 10 10 10 10]
 [10 10 10 10 10]
 [10 10 10 10 10]
 [10 10 10 10 10]
 [10 10 10 10 10]]
Random Array:
 [[44 47 64 67 67]
 [ 9 83 21 36 87]
 [70 88 88 12 58]
 [65 39 87 46 88]
 [81 37 25 77 72]]
Largest value in the random array: 88
Smallest value in the random array: 9
```

In [6]:
```
image_path = 'players.jpg'
image = Image.open(image_path)
plt.imshow(image)
plt.axis('off')
plt.title('Original Image')
plt.show()

image_array = np.array(image)
print("Shape of the arrange image :", image_array.shape)
```


```
Shape of the arrange image : (363, 544, 3)
```

In [7]:
```
image_array[:, :, 0] = 0
image_array[:, :, 1] = 0

plt.imshow(image_array)
plt.axis('off')
plt.title('Isolated Blue Channel')
plt.show()
```



Question 5

In [8]:
```
#read in the players image
#Perform scaling using resize method
#a) half the image using dim=(0,0), fx=0.5, fy=0.5
#b) stretch the image to dim = (600,600)
#c) stretch the image to dim = (600,600 using interpolation=cv2.INTER_NEAREST

#Answer

# Read the player's image using OpenCV
image_path = 'players.jpg'
image = cv2.imread(image_path)

if image is None:
    print(f"Error: Unable to read the image at '{image_path}'")
else:
    # Display the original image
    plt.figure(figsize=(8, 6))
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title('Original Image')
    plt.axis('off')
    plt.show()

    # Perform scaling operations
    # a) Half the image using fx=0.5, fy=0.5
    resized_half = cv2.resize(image, None, fx=0.5, fy=0.5)

    # b) Stretch the image to (600, 600)
    resized_stretch = cv2.resize(image, (600, 600))

    # c) Stretch the image to (600, 600) using interpolation=cv2.INTER_NEAREST
```

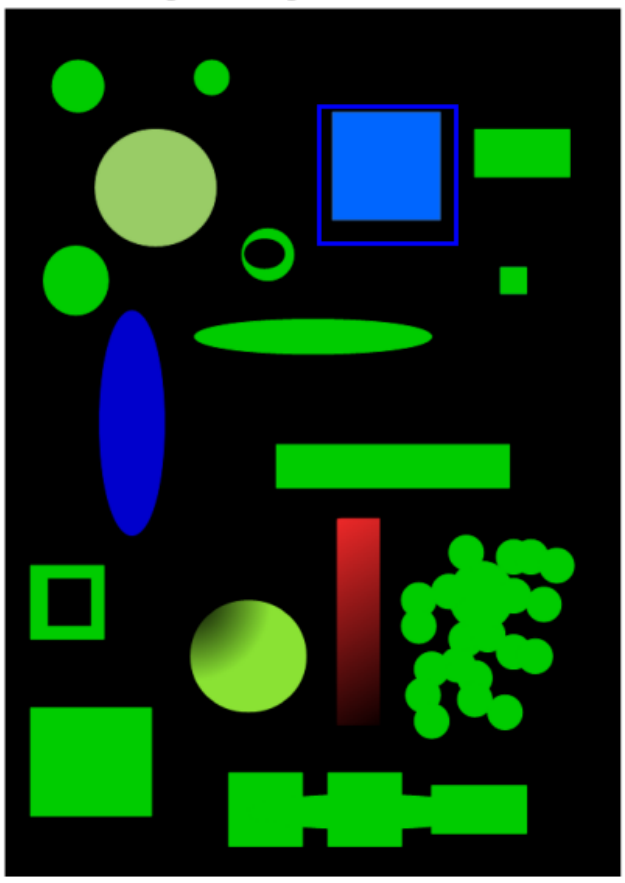
```
resized_nearest = cv2.resize(image, (600, 600), interpolation=cv2.INTER_NEAREST)

plt.figure(figsize=(18, 6))
plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(resized_half, cv2.COLOR_BGR2RGB))
plt.title('Resized Half')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(cv2.cvtColor(resized_stretch, cv2.COLOR_BGR2RGB))
plt.title('Resized Stretch (600, 600)')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(cv2.cvtColor(resized_nearest, cv2.COLOR_BGR2RGB))
plt.title('Resized Stretch (600, 600) Nearest')
plt.axis('off')

plt.show()
```

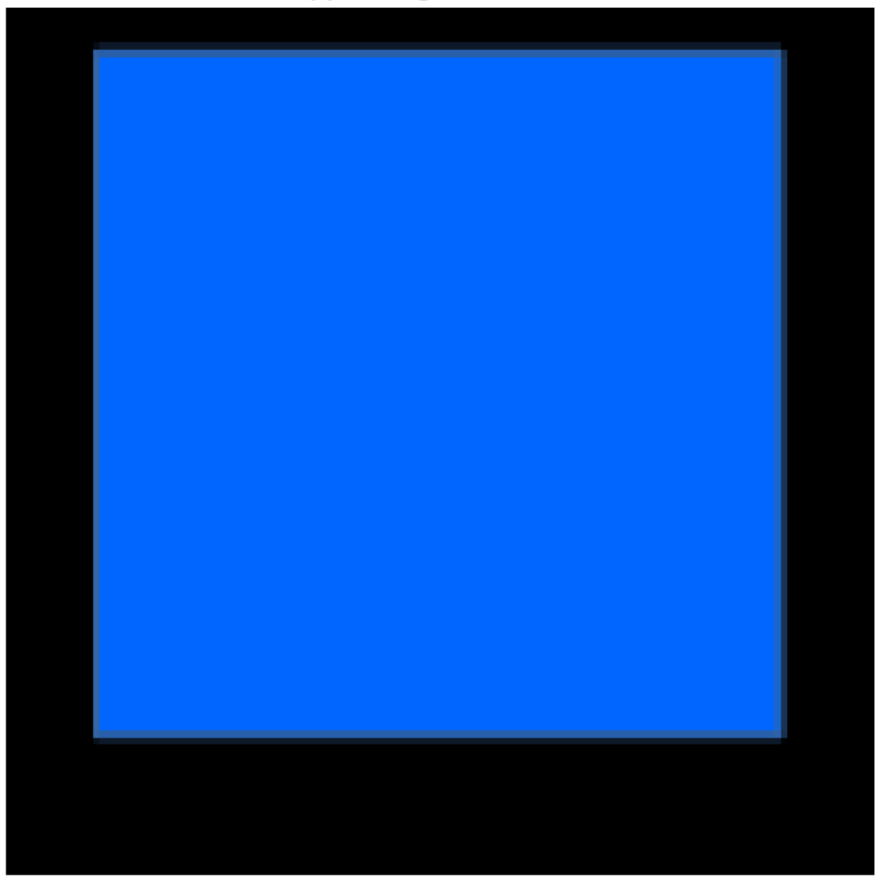
In [ ]:

Question 6

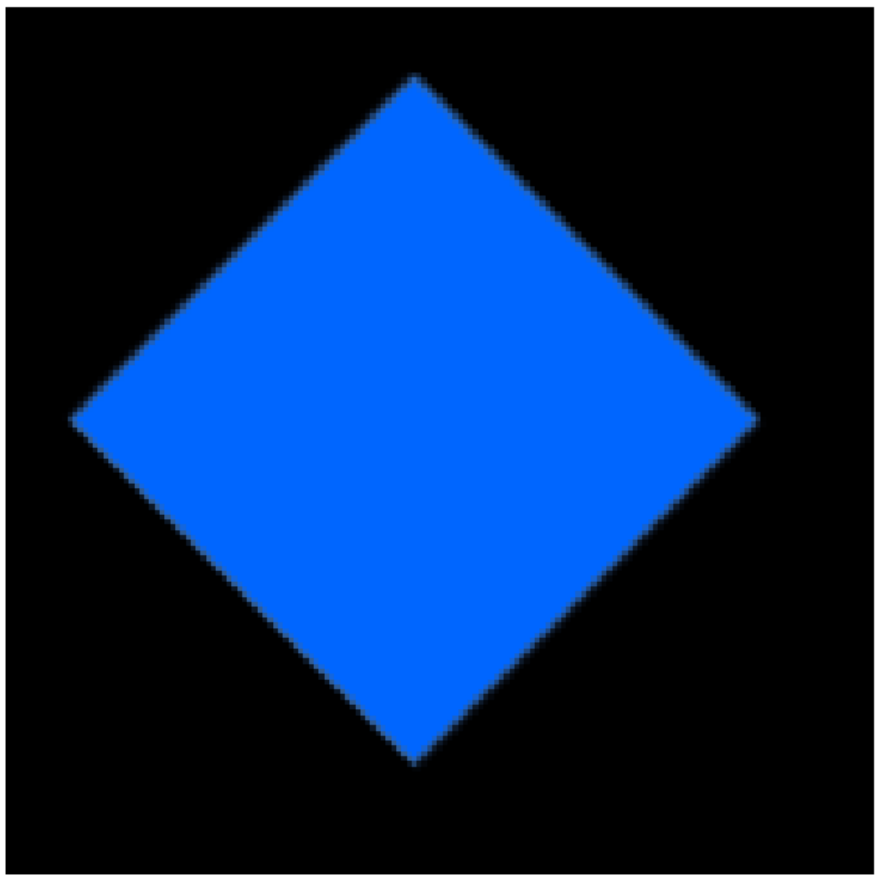
In [9]:
```
#read in the detect_blob image
#then draw a box shape around the blue solid box in the image that is your region of interest
#crop your region of interest
#and then rotate it 45 degrees so that it is not clipped

#Answer
import matplotlib.patches as patches

blob = cv2.imread('./detect_blob.png')

# Convert the image from BGR to RGB
img_rgb = cv2.cvtColor(blob, cv2.COLOR_BGR2RGB)

x, y, w, h = 275, 85, 120, 120

plt.figure(figsize=(12, 12))
```

```
plt.subplot(2, 2, 1)
plt.imshow(img_rgb)
plt.title('Original Image with Blue Box')
plt.axis('off')

rect = patches.Rectangle((x, y), w, h, linewidth=2, edgecolor='blue', facecolor='none')

ax = plt.gca()

ax.add_patch(rect)

roi = img_rgb[y:y+h, x:x+w]

plt.subplot(2, 2, 2)
plt.imshow(roi)
plt.title('Cropped Region of Interest')
plt.axis('off')


center = (w // 2, h // 2)

rotation_matrix = cv2.getRotationMatrix2D(center, 45, 1.0)

cos = np.abs(rotation_matrix[0, 0])
sin = np.abs(rotation_matrix[0, 1])
new_w = int((h * sin) + (w * cos))
new_h = int((h * cos) + (w * sin))

rotation_matrix[0, 2] += (new_w / 2) - center[0]
rotation_matrix[1, 2] += (new_h / 2) - center[1]

rotated_roi = cv2.warpAffine(roi, rotation_matrix, (new_w, new_h))

plt.subplot(2, 2, 3)
plt.imshow(rotated_roi)
plt.title('Rotated ROI (45 degrees)')
plt.axis('off')

plt.subplot(2, 2, 4)
plt.axis('off')

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





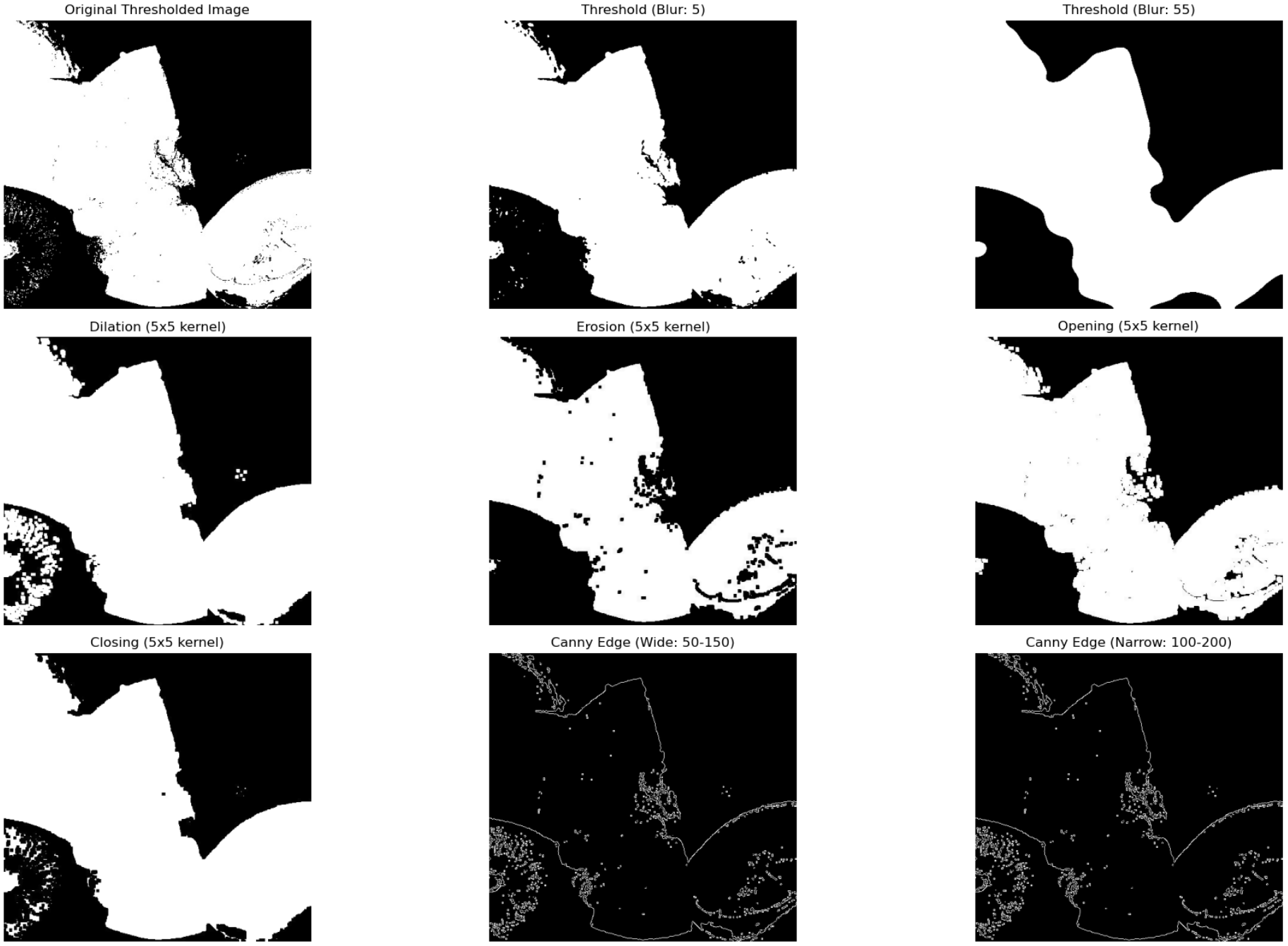
Rotated ROI (45 degrees)



Question 7

In [10]:
```
#read in the thresh image, perform thresholding by using gaussian blur with thresholds 5,55
#then perform dilation, erosion, opening & closing using a 5x5 kernel with 1 iteration on the original thresh image
#Finally apply canny edge detection - experiment using wide and narrow thresholds
#(so just 2 images in total one wide and one narrow)

#Answer

img_thresh = cv2.imread('thresh.jpg', cv2.IMREAD_GRAYSCALE)

# Apply Gaussian blur and thresholding with different thresholds
blur_thresh_5 = cv2.GaussianBlur(img_thresh, (5, 5), 0)
_, thresh_5 = cv2.threshold(blur_thresh_5, 127, 255, cv2.THRESH_BINARY)

blur_thresh_55 = cv2.GaussianBlur(img_thresh, (55, 55), 0)
_, thresh_55 = cv2.threshold(blur_thresh_55, 127, 255, cv2.THRESH_BINARY)

# Morphological operations: dilation, erosion, opening, closing
kernel = np.ones((5, 5), np.uint8)
dilation = cv2.dilate(img_thresh, kernel, iterations=1)
erosion = cv2.erode(img_thresh, kernel, iterations=1)
opening = cv2.morphologyEx(img_thresh, cv2.MORPH_OPEN, kernel)
closing = cv2.morphologyEx(img_thresh, cv2.MORPH_CLOSE, kernel)

# Canny edge detection with wide and narrow thresholds
edges_wide = cv2.Canny(img_thresh, 50, 150)
edges_narrow = cv2.Canny(img_thresh, 100, 200)


images = [img_thresh, thresh_5, thresh_55, dilation, erosion, opening, closing, edges_wide, edges_narrow]
```

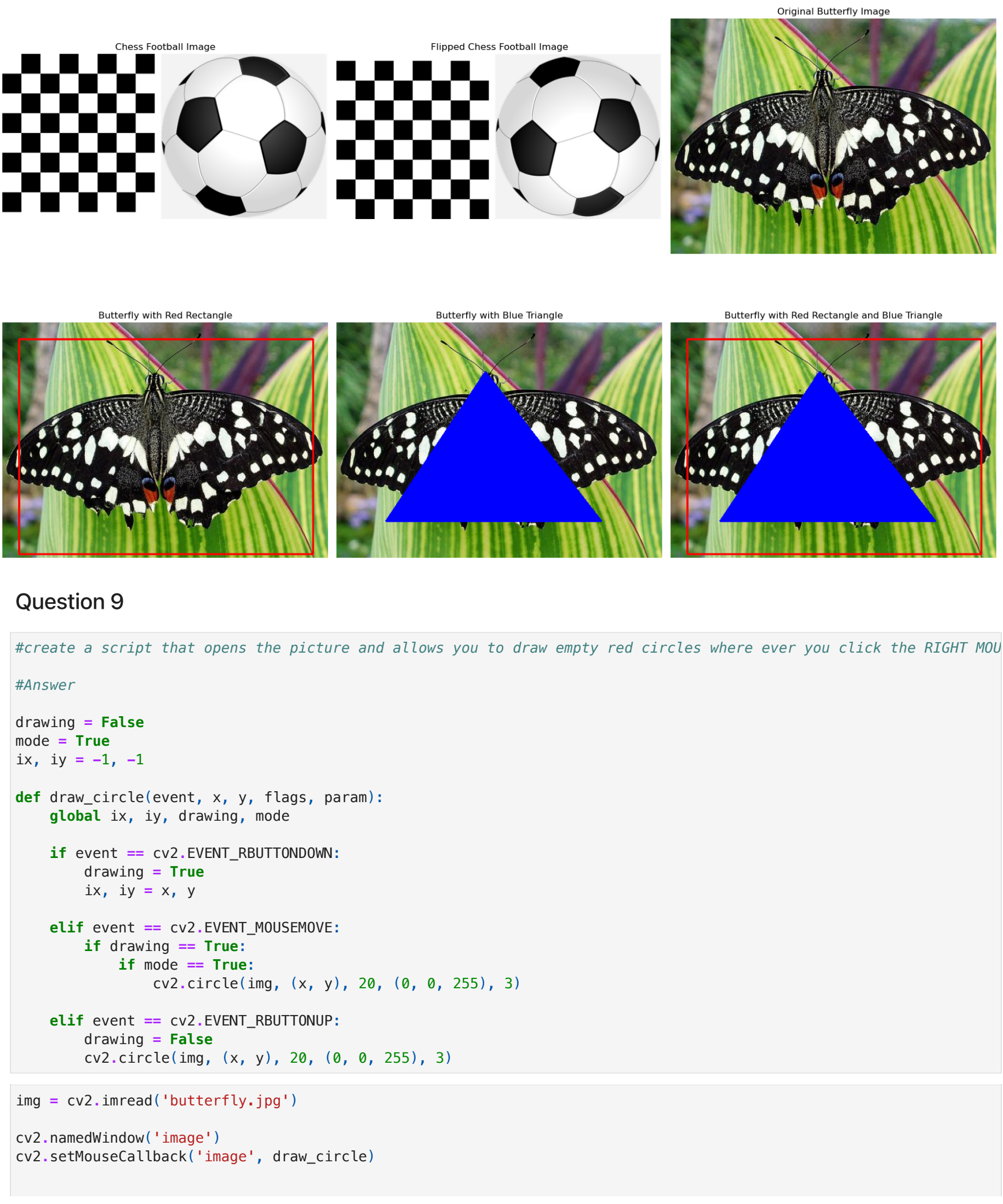
```
titles = ['Original Thresholded Image', 'Threshold (Blur: 5)', 'Threshold (Blur: 55)',
          'Dilation (5x5 kernel)', 'Erosion (5x5 kernel)', 'Opening (5x5 kernel)',
          'Closing (5x5 kernel)', 'Canny Edge (Wide: 50-150)', 'Canny Edge (Narrow: 100-200)']

plt.figure(figsize=(20, 12))

for i in range(len(images)):
    plt.subplot(3, 3, i + 1)
    plt.imshow(images[i], cmap='gray')
    plt.title(titles[i])
    plt.axis('off')

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



Question 8

In [11]:
```
#read in the butterfly image
#open the chess_football image and display it in the notebook. Make sure to correct for the RGB order
#flip the image upside down and display it in the notebook.
#draw an empty RED rectangle around the butterfly and display the image in the notebook.
#draw a BLUE TRIANGLE in the middle of the image. The size and angle is up to you, but it should be a triangle (thr
#now fill in this triangle
#display the original image as well as the resultant image

#Answer


butterfly_img = cv2.imread('butterfly.jpg')
butterfly_img_rgb = cv2.cvtColor(butterfly_img, cv2.COLOR_BGR2RGB)

chess_football_img = cv2.imread('chess_football.png')
chess_football_img_rgb = cv2.cvtColor(chess_football_img, cv2.COLOR_BGR2RGB)

flipped_chess_football_img = cv2.flip(chess_football_img_rgb, 0)

butterfly_img_with_rectangle = butterfly_img_rgb.copy()
rect_start_point = (25, 25)
rect_end_point = (470, 350)
cv2.rectangle(butterfly_img_with_rectangle, rect_start_point, rect_end_point, (255, 0, 0), 2)

butterfly_img_with_triangle = butterfly_img_rgb.copy()
triangle_points = np.array([[225, 75], [75, 300], [400, 300]])
cv2.polylines(butterfly_img_with_triangle, [triangle_points], isClosed=True, color=(0, 0, 255), thickness=2)
cv2.fillPoly(butterfly_img_with_triangle, [triangle_points], color=(0, 0, 255))
```

```
resultant_img = butterfly_img_with_rectangle.copy()
cv2.polylines(resultant_img, [triangle_points], isClosed=True, color=(0, 0, 255), thickness=2)
cv2.fillPoly(resultant_img, [triangle_points], color=(0, 0, 255))

fig, axs = plt.subplots(2, 3, figsize=(18, 12))

images = [
    (chess_football_img_rgb, 'Chess Football Image'),
    (flipped_chess_football_img, 'Flipped Chess Football Image'),
    (butterfly_img_rgb, 'Original Butterfly Image'),
    (butterfly_img_with_rectangle, 'Butterfly with Red Rectangle'),
    (butterfly_img_with_triangle, 'Butterfly with Blue Triangle'),
    (resultant_img, 'Butterfly with Red Rectangle and Blue Triangle')
]

for i, (img, title) in enumerate(images):
    ax = axs[i // 3, i % 3]
    ax.imshow(img)
    ax.set_title(title)
    ax.axis('off')

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



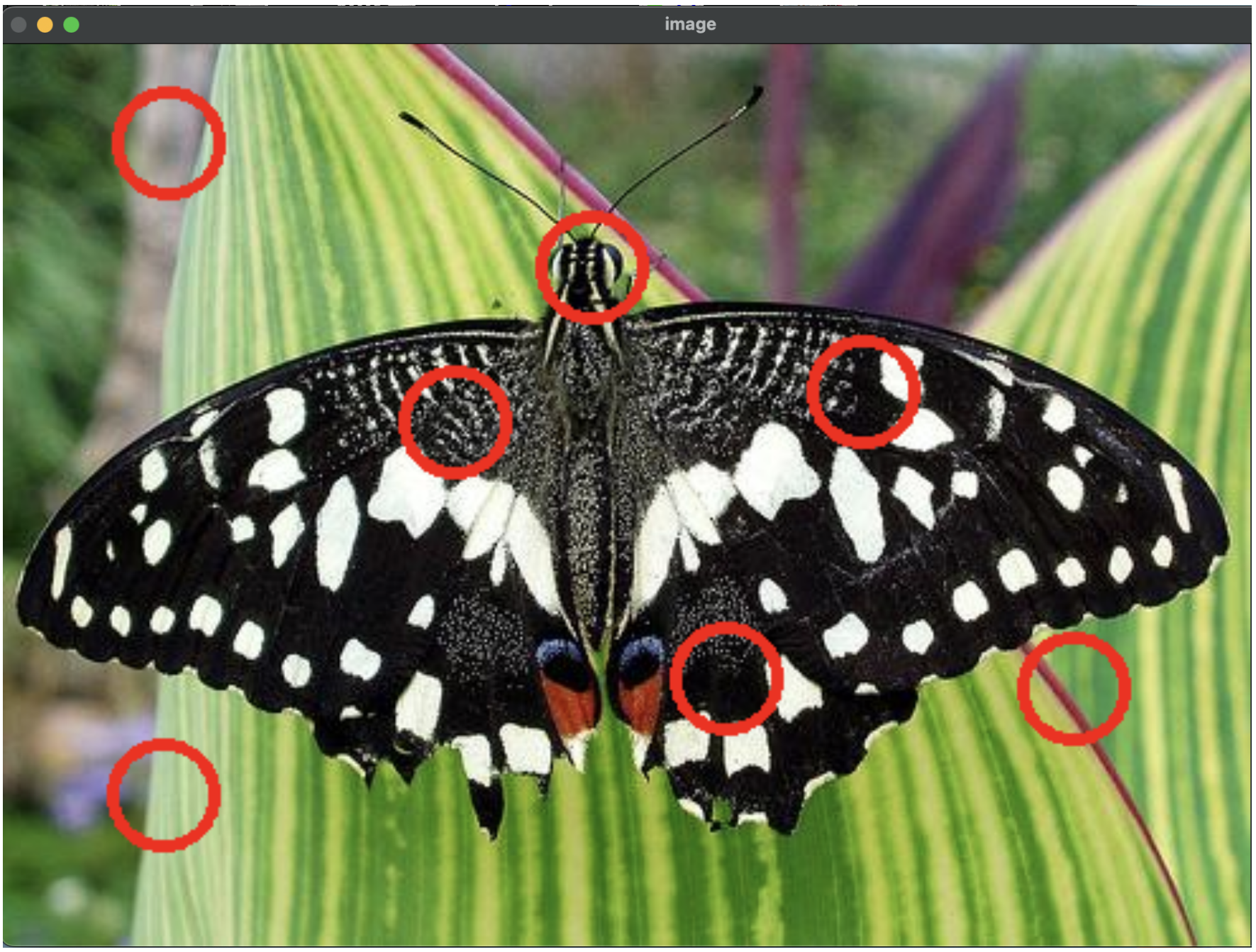
Question 9

In [17]:
```
#create a script that opens the picture and allows you to draw empty red circles where ever you click the RIGHT MOU
#Answer

drawing = False
mode = True
ix, iy = -1, -1

def draw_circle(event, x, y, flags, param):
    global ix, iy, drawing, mode

    if event == cv2.EVENT_RBUTTONDOWN:
        drawing = True
        ix, iy = x, y

    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing == True:
            if mode == True:
                cv2.circle(img, (x, y), 20, (0, 0, 255), 3)

    elif event == cv2.EVENT_RBUTTONUP:
        drawing = False
        cv2.circle(img, (x, y), 20, (0, 0, 255), 3)
```

In [18]:
```
img = cv2.imread('butterfly.jpg')

cv2.namedWindow('image')
cv2.setMouseCallback('image', draw_circle)
```

```
while True:
    cv2.imshow('image', img)
    k = cv2.waitKey(1) & 0xFF
    if k == ord('m'):
        mode = not mode
    elif k == 27:
        break

cv2.destroyAllWindows()

#-------------------------------------- ESC key to exit ---------------------------------------
```

For this code output i have added the screenshort, ypu can see that the script opens the image and i can draw the circles on that image where ever i want to :

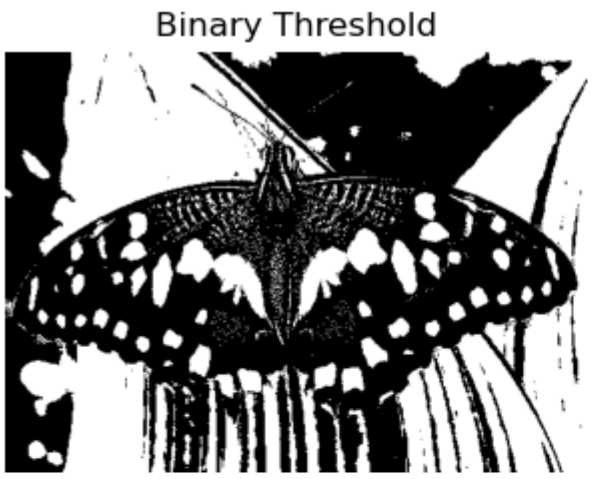


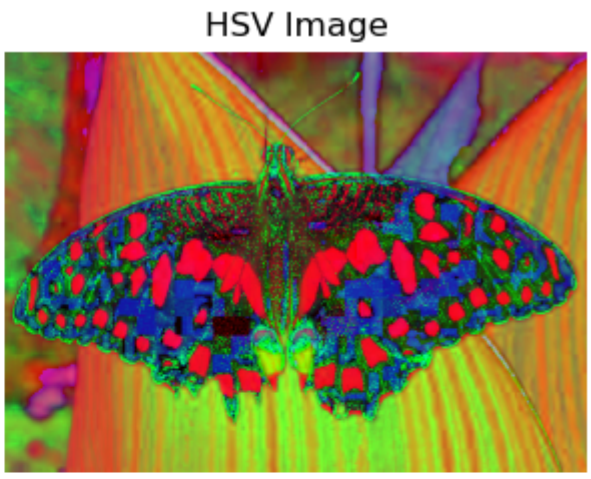
Question 10

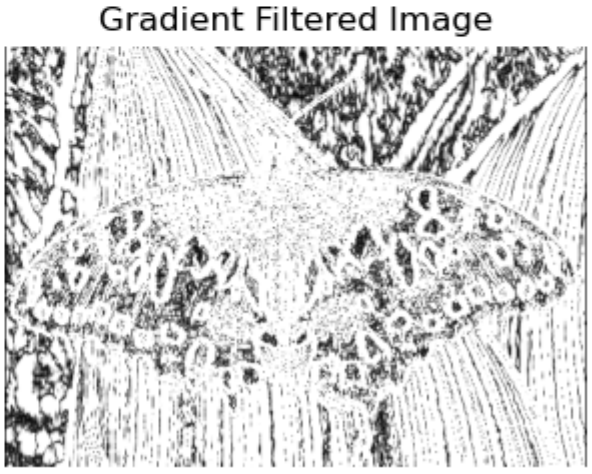
In [27]:
```
#open and display the any image of your choice
#apply a binary threshold onto the image
#convert image colorspace to HSV and display the image
#create a low pass filter with a 4 by 4 Kernel filled with values of 1/10 (0.01) and then use 2-D Convolution to bl
#create a Horizontal Sobel Filter with a kernel size of 5 to the grayscale version of the image and then display th
#plot the color histograms for the RED, BLUE, and GREEN channel of the image. Pay careful attention to the ordering

#Answer
image_path = 'butterfly.jpg'
image = cv2.imread(image_path)

gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
_, binary_threshold = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)

hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

kernel = np.ones((4, 4), np.float32) / 10
blurred_image = cv2.filter2D(image, -1, kernel)

sobel_x = cv2.Sobel(gray_image, cv2.CV_64F, 1, 0, ksize=5)
gradient_filtered_image = cv2.convertScaleAbs(sobel_x)
```

In [28]:
```
colors = ('b', 'g', 'r')
plt.figure(figsize=(12, 6))

# Display images and histograms using a loop
for i, (img, title) in enumerate([(binary_threshold, 'Binary Threshold'),
                                  (hsv_image, 'HSV Image'),
```

```
                                    (blurred_image, 'Blurred Image'),
                                    (gradient_filtered_image, 'Gradient Filtered Image')]):
    plt.subplot(2, 2, i + 1)
    if img.ndim == 2:
        plt.imshow(img, cmap='gray')
    else:
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title(title)
    plt.axis('off')
```







In [29]:
```
plt.figure(figsize=(12, 6))
for i, color in enumerate(colors):
    histogram = cv2.calcHist([image], [i], None, [256], [0, 256])
    plt.plot(histogram, color=color)
plt.xlabel('Pixel Intensity')
plt.ylabel('Frequency')
plt.title('Color Histogram')

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