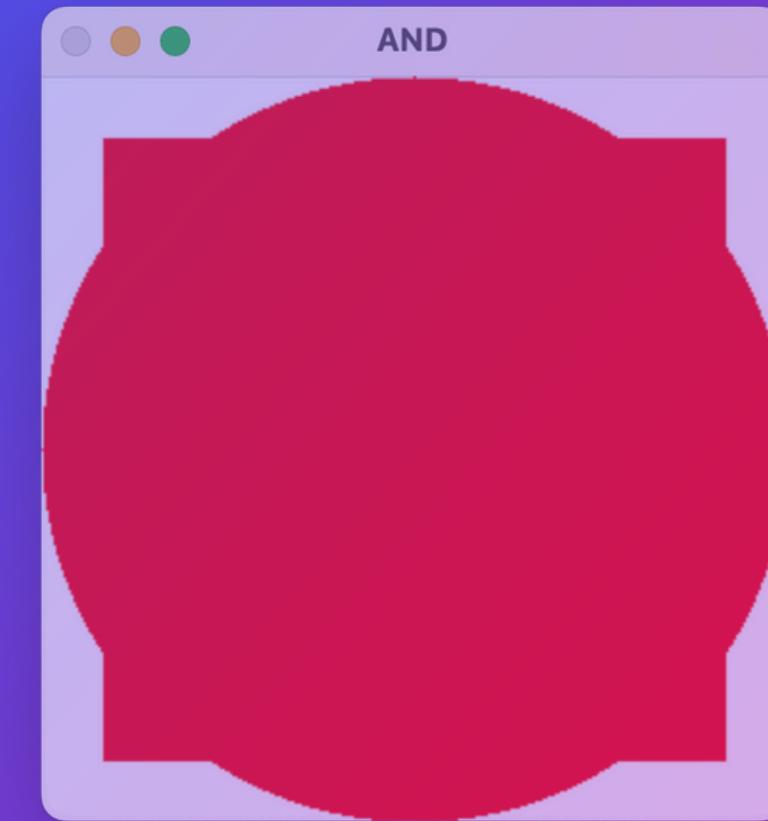


# Bitwise Mask inrange



# BITWISE OPENCV

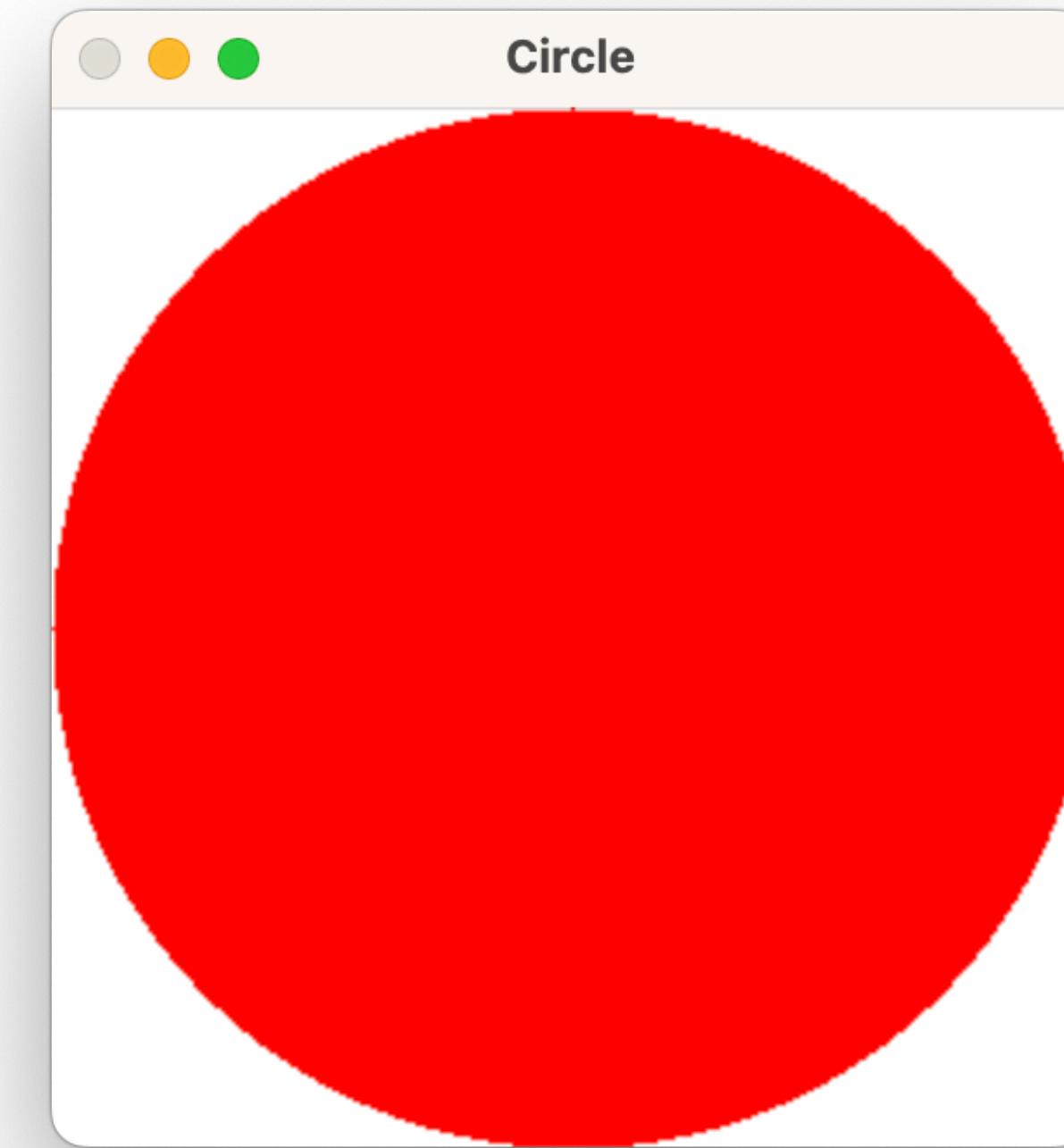
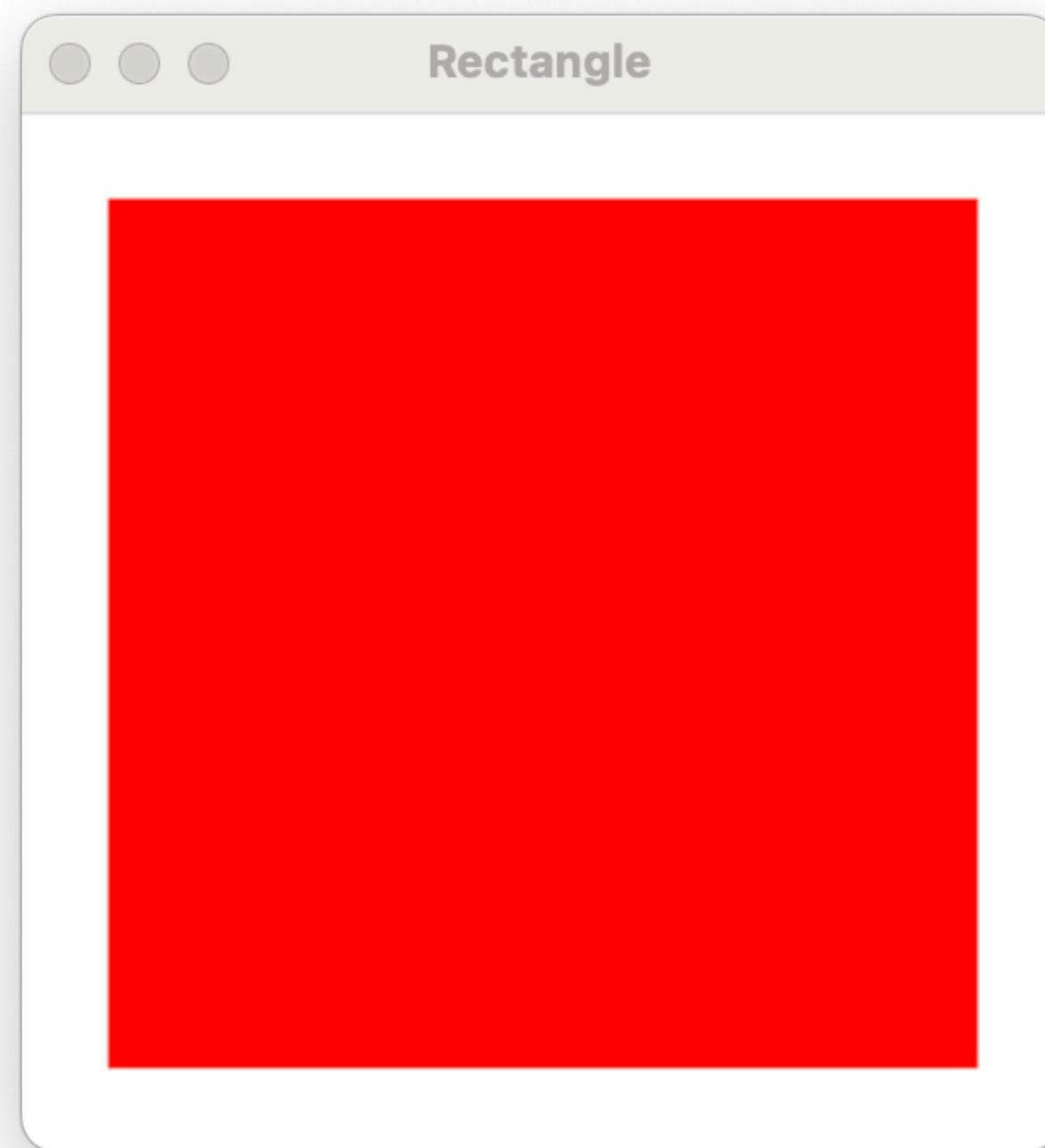
เป็นตัวดำเนินการในระดับบิตกับรูปภาพโดยใช้ OpenCV โดยคำสั่ง bitwise\_ โดยรายละเอียดคำสั่งทั้ง 4 คือ

1. **bitwise\_and**
2. **bitwise\_or**
3. **bitwise\_not**
4. **bitwise\_xor**

# CODE

```
● ● ●  
1 import numpy as np  
2 import cv2 as cv  
3  
4 rectangle = np.zeros((300, 300,3), dtype="uint8")  
5 rectangle[:] = [255,255,255]  
6 cv.rectangle(rectangle, (25, 25), (275, 275), (0, 0, 255), -1)  
7 cv.imshow("Rectangle", rectangle)  
8  
9 circle = np.zeros((300, 300,3), dtype = "uint8")  
10 circle[:] = [255,255,255]  
11 cv.circle(circle, (150, 150), 150, (0, 0, 255), -1)  
12 cv.imshow("Circle", circle)
```

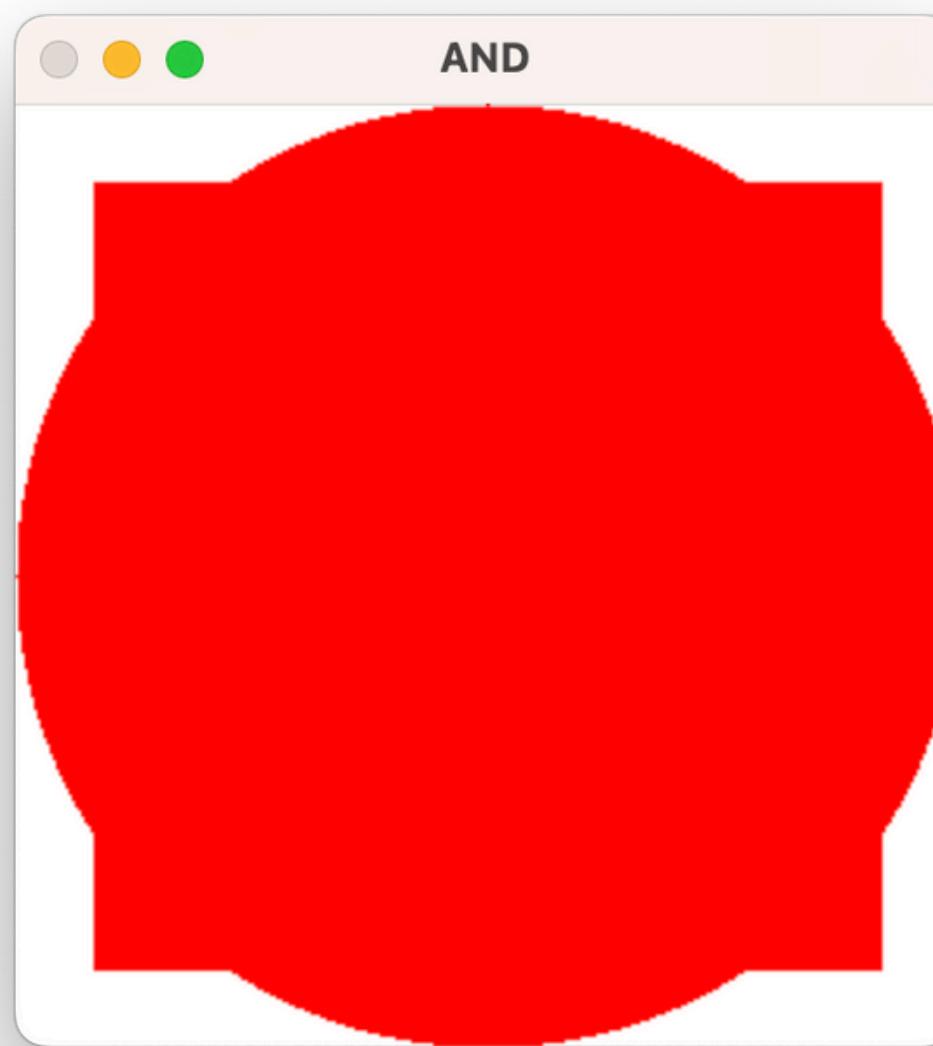
# OUTPUT



# BITWISE\_AND

- **bitwise\_and(img1, img2 mask=mask )** หมายถึง การ img1 และ img2 ส่องภาพเหมือนกันรวมกัน
- Mask คือหน้ากาก หรือ ส่วนที่ปิด บัง ไว้ซึ่งในที่นี้ คือการเอาเฉพาะส่วนที่ ไม่ได้ถูกบังไว้ของภาพที่ 1 และ 2 มารวมกับ mask

# OUTPUT



# CODE



```
1 bitwise_and = cv.bitwise_and(rectangle,circle)
2 cv.imshow("AND", bitwise_and)
```

# BITWISE\_OR

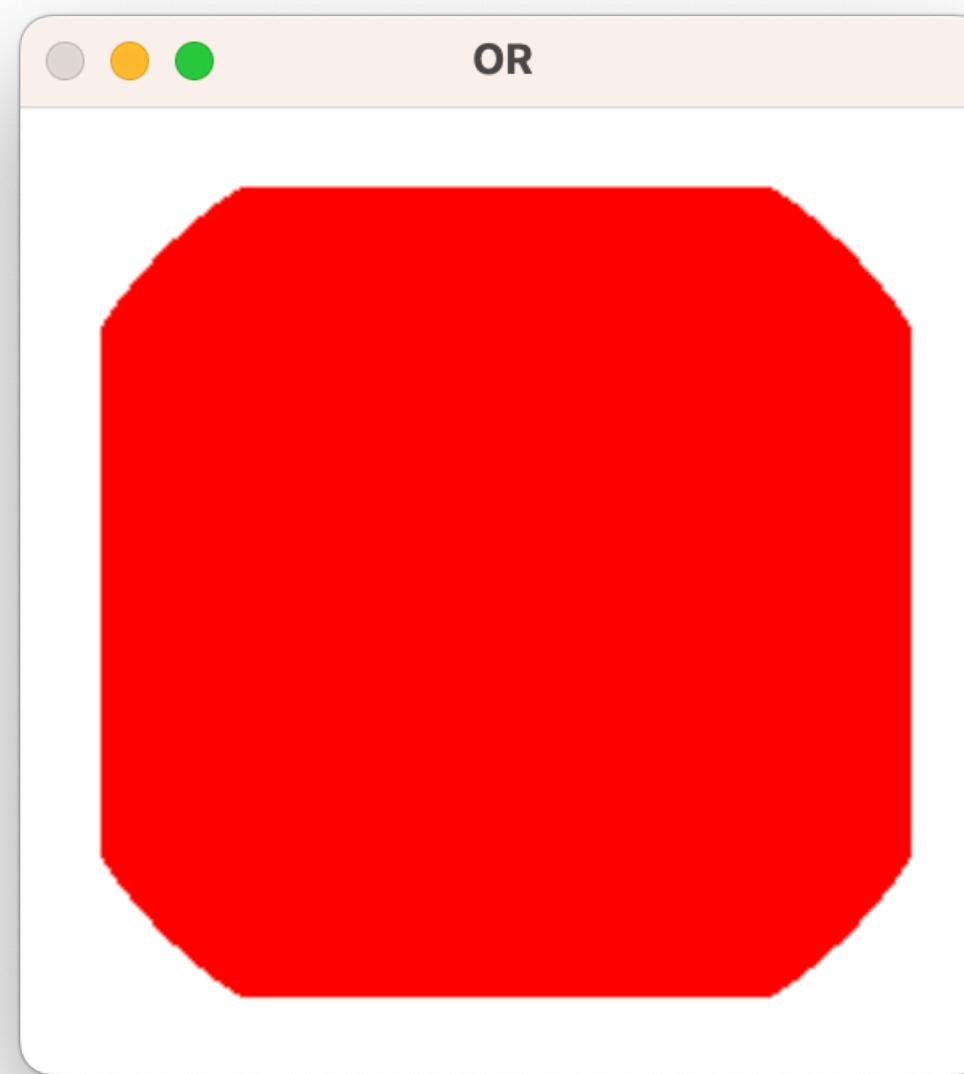
- **bitwise\_or(img1, img2)** หมายถึง การแสดงภาพเฉพาะส่วนที่เหมือนกันของ img1 และ img2

# CODE



```
1 bitwise_or = cv.bitwise_or(rectangle,circle)
2 cv.imshow("OR", bitwise_or)
```

# OUTPUT



**BITWISE\_NOT**

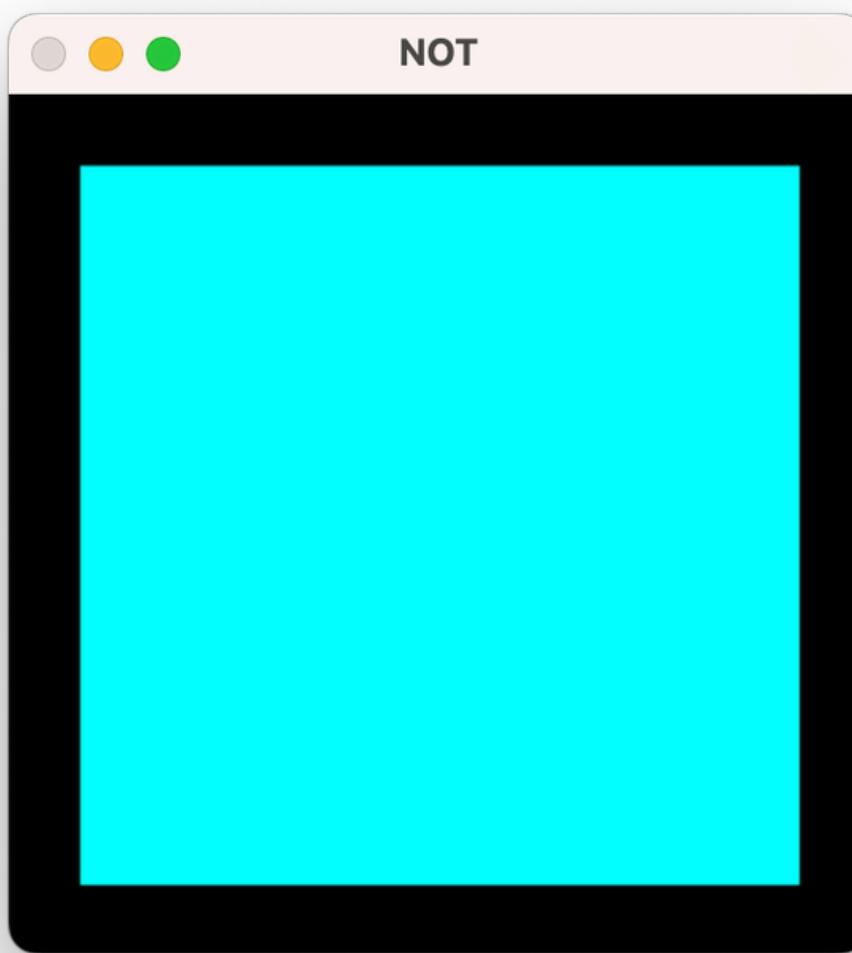
**bitwise\_not(img1)** หมาย  
ถึง การแสดงภาพส่วนกลับ  
ของรูปที่ 1

# CODE



```
1 bitwise_not = cv.bitwise_not(rectangle)
2 cv.imshow("NOT", bitwise_not)
```

# OUTPUT



# BITWISE\_XOR

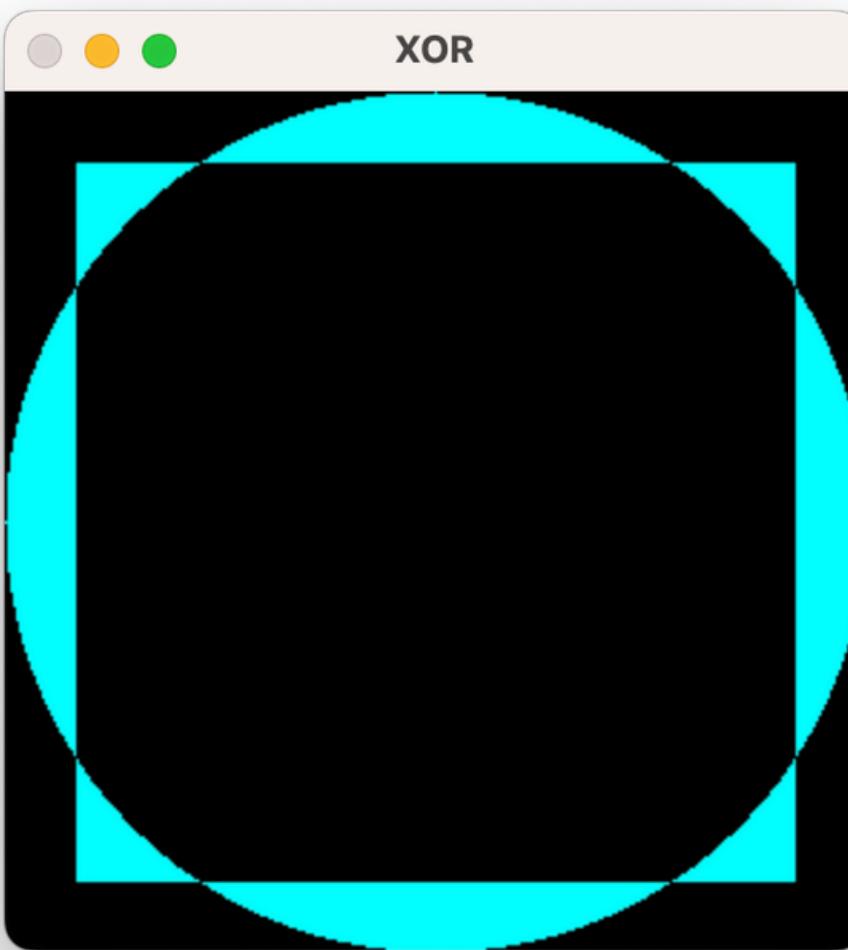
**bitwise\_xor(img1, img2)** หมายถึง การแสดงภาพส่วนที่ไม่เหมือนกันของ img1 และ img2

# CODE



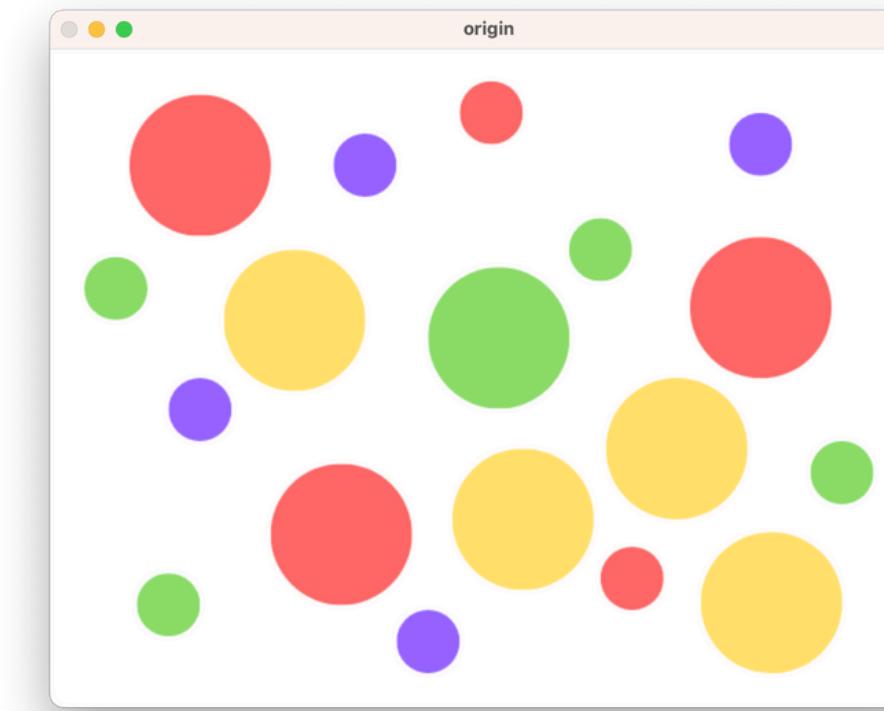
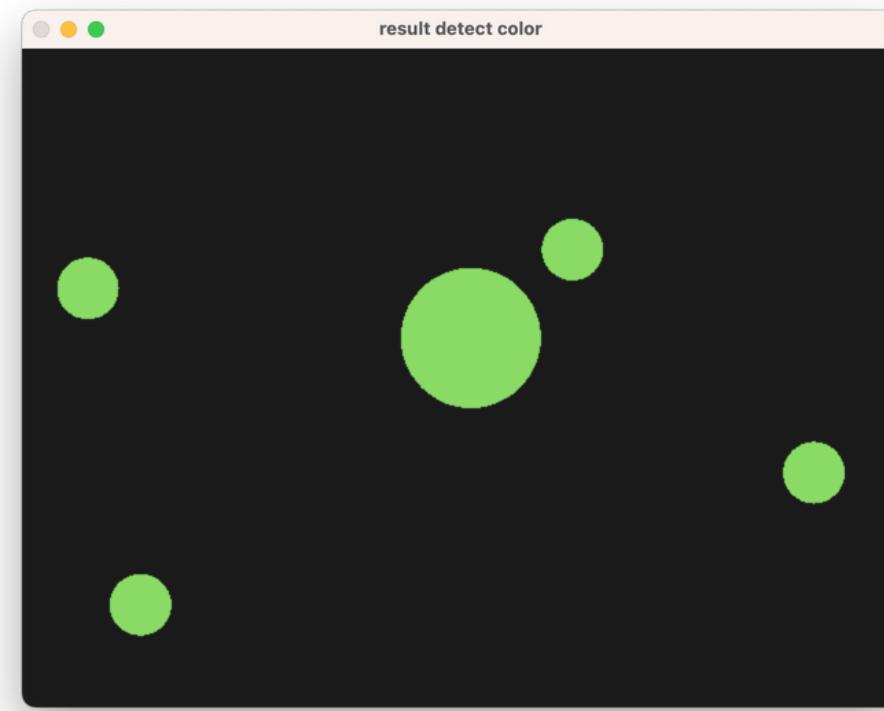
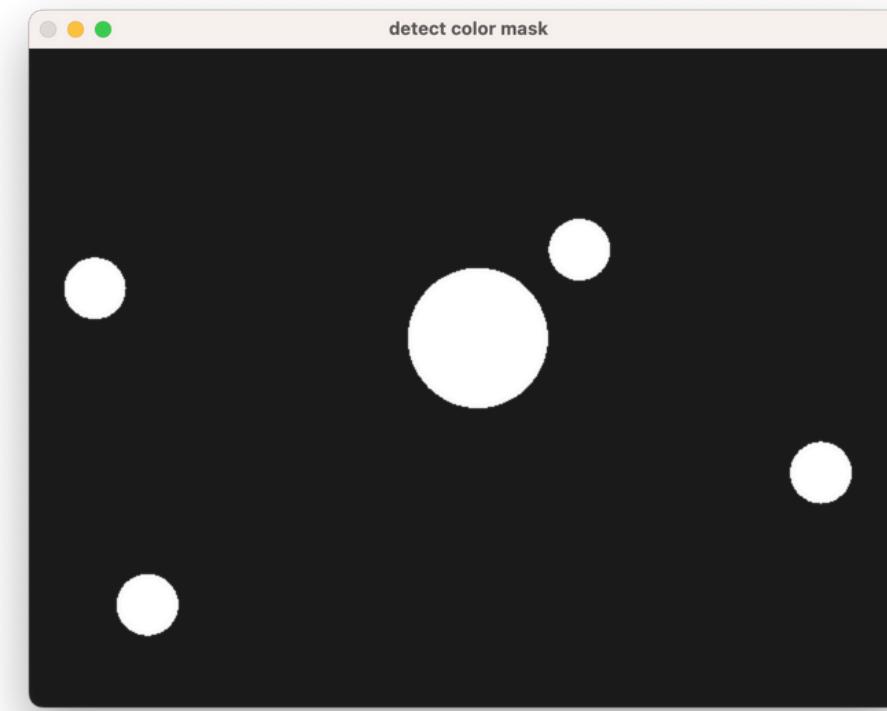
```
1 bitwise_xor = cv.bitwise_xor(rectangle,circle)
2 cv.imshow("XOR", bitwise_xor)
```

# OUTPUT



# Detect Object

การตรวจจับกลุ่มวัตถุ ด้วยสี

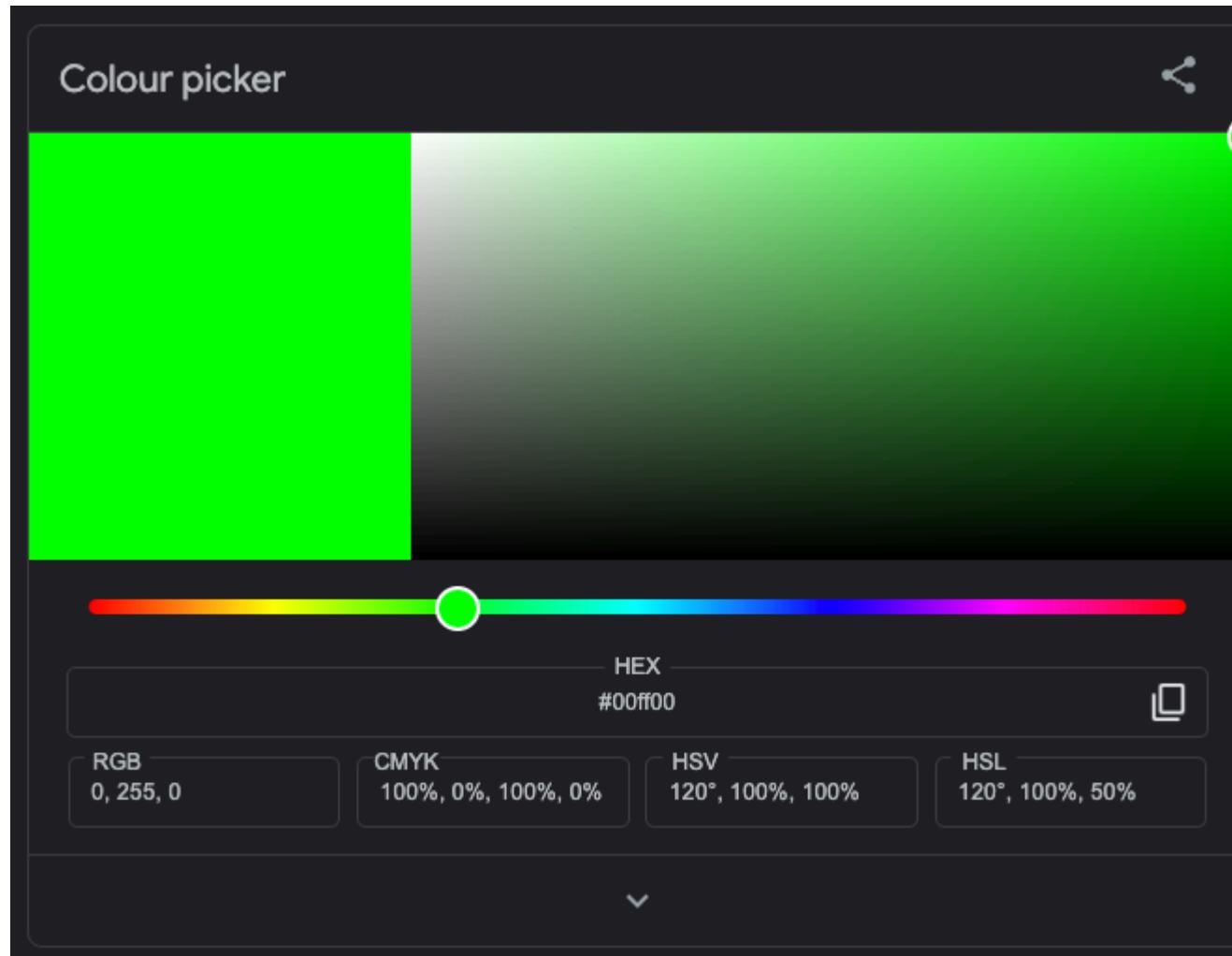


# DETECT OBJECT

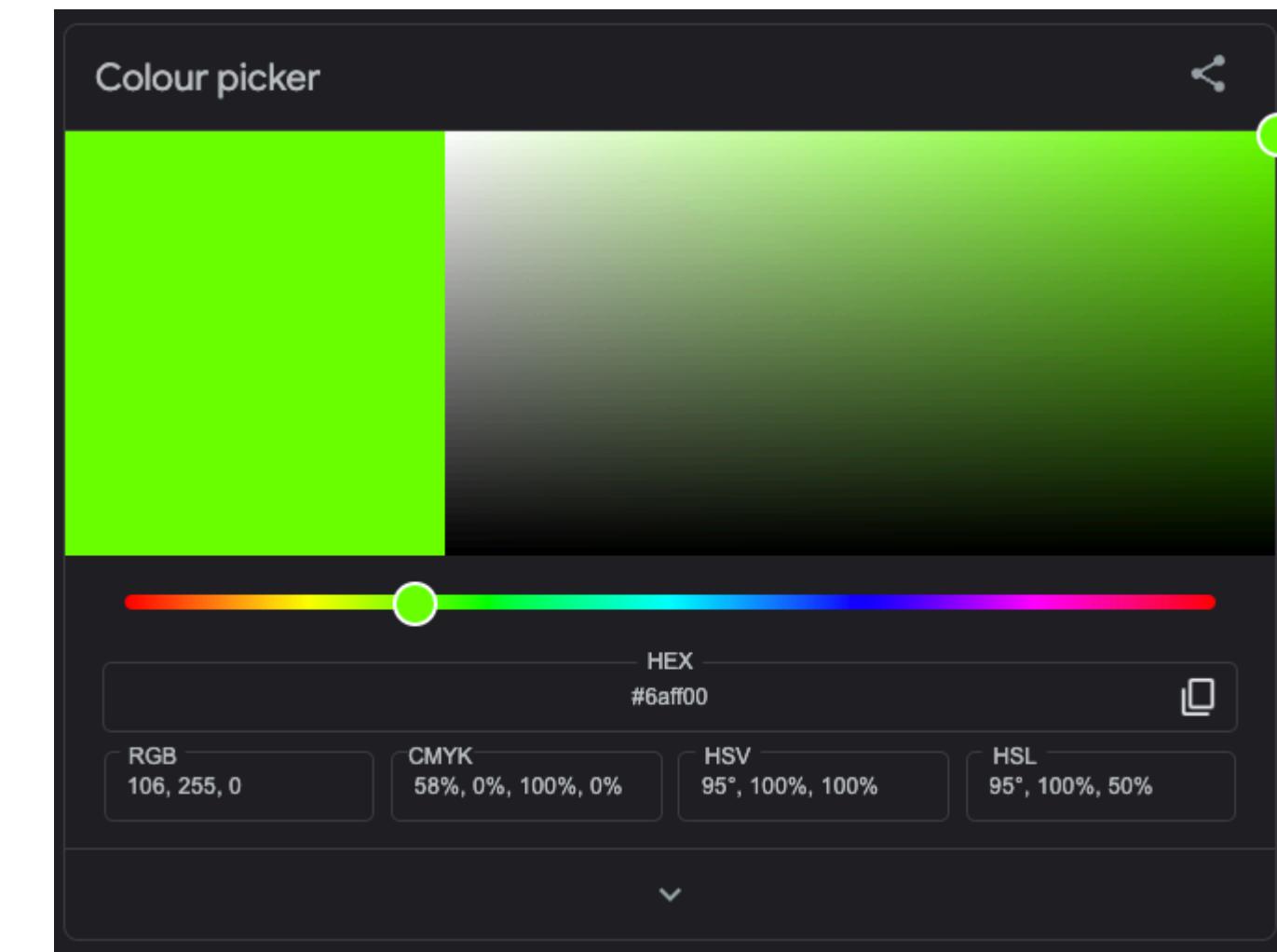
# DETECT OBJECT

หลักการทำงานคือ ระบุช่วงของสีต่างๆ เช่น สีเขียว ซึ่งสีเขียวมาตรฐาน จะมีค่าสี BGR คือ (0,255,0) แต่สีเขียวนั้นมีสีเขียวหลายแบบ เช่น เขียวอมม่วง สีเขียวอมแดง สีเขียวอมดำ

# GREEN

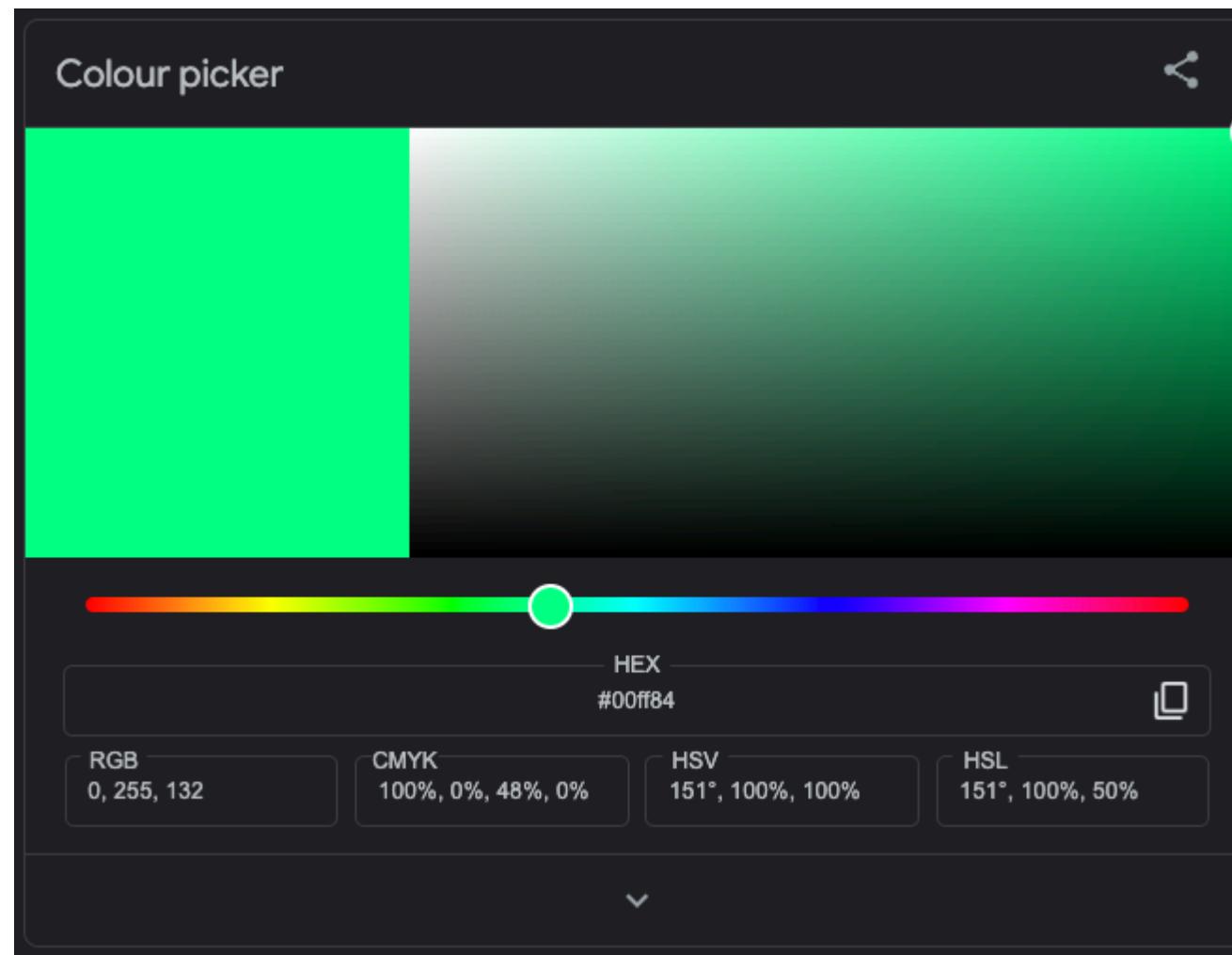


สีเขียวมาตรฐาน BGR(0,255,0)

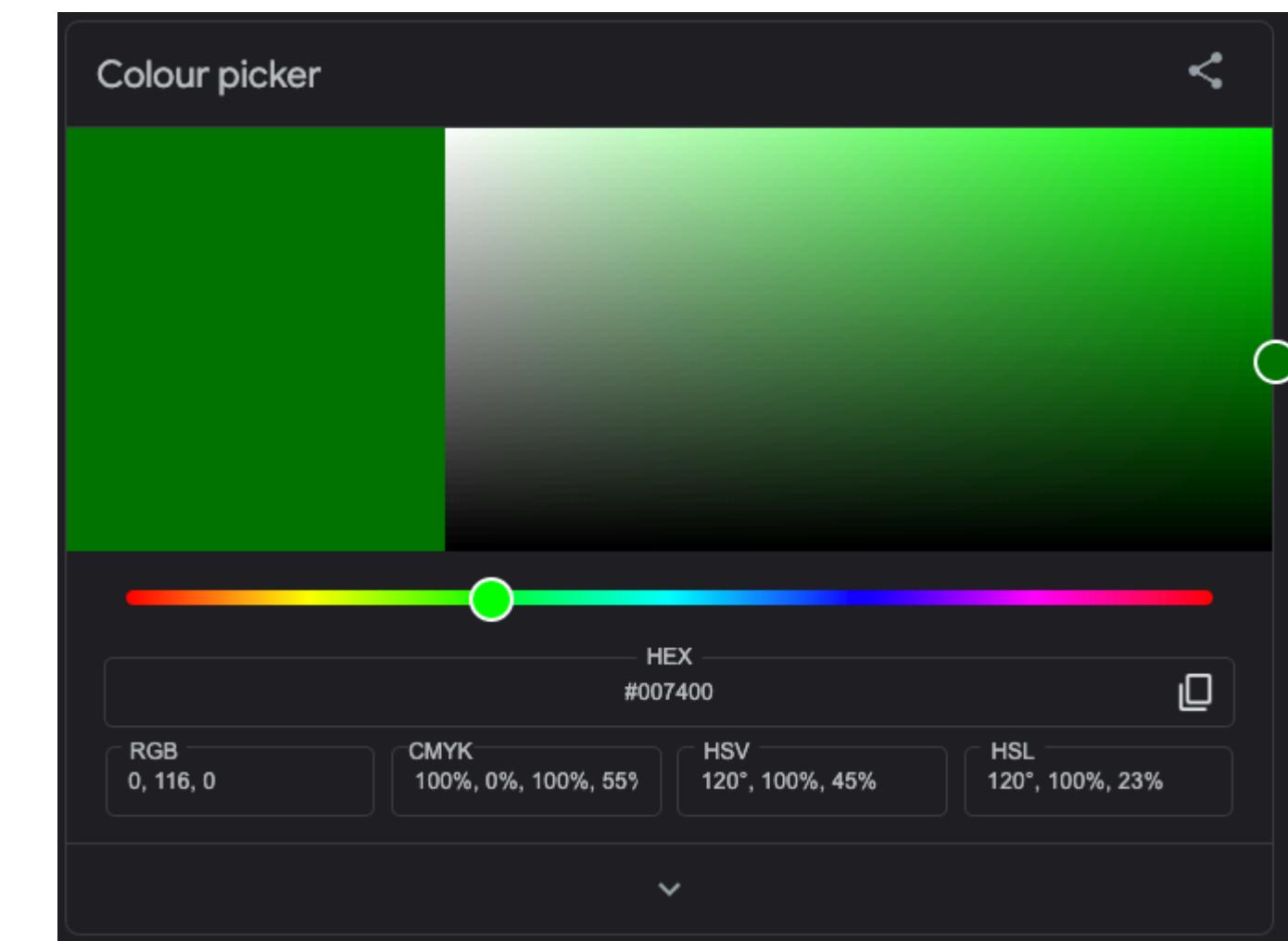


สีเขียวอ่อนแดง BGR(0,214,106)

# GREEN



สีเขียวอมน้ำเงิน BGR(132,255,0)

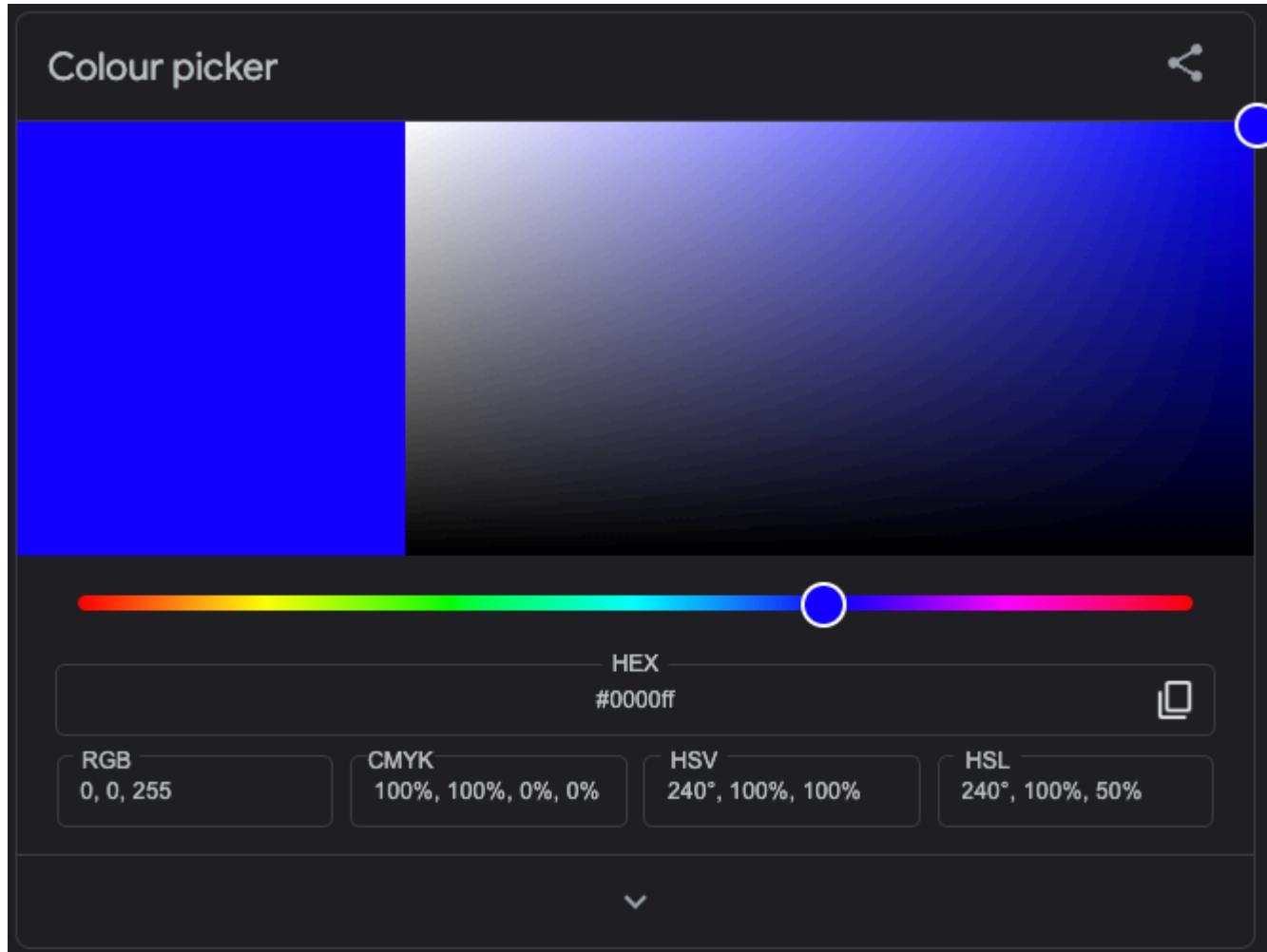


สีเขียวอมดำ BGR(0,116,0)

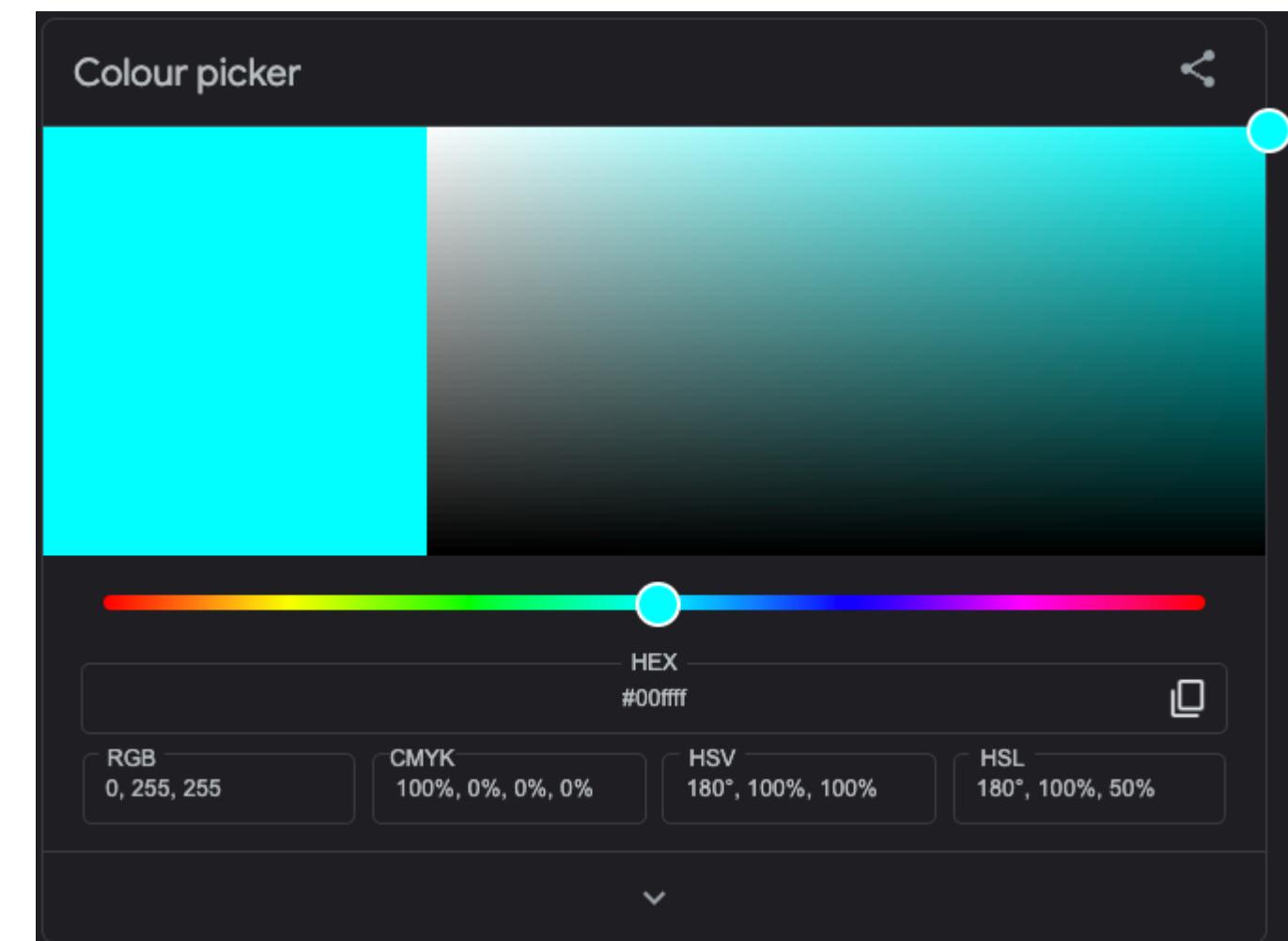
# GREEN

- ดังนั้นการกำหนดหาช่วงของสีเขียวนั้น อาจจะกำหนดตามช่วงดังกล่าวได้โดย การกำหนด
  - ค่าช่วง สีเขียวสูงสุดที่อาจเป็นไปได้ ซึ่งจะได้ค่าเป็น `upper_green = (132,255,106)`
  - ค่าช่วงสีเขียวต่ำสุดที่อาจเป็นไปได้ ซึ่งจะได้ค่าเป็น `lower_green = (0,116,0)`
- ซึ่งการกำหนดช่วงของสีนั้น เพื่อจะนำมาทำการ mask หรือการครอบสีที่อยู่ใน ช่วงนั้นไว้และให้สีที่ไม่ได้ในช่วงสีที่ระบุถูกยกเป็นสีดำ

# BLUE

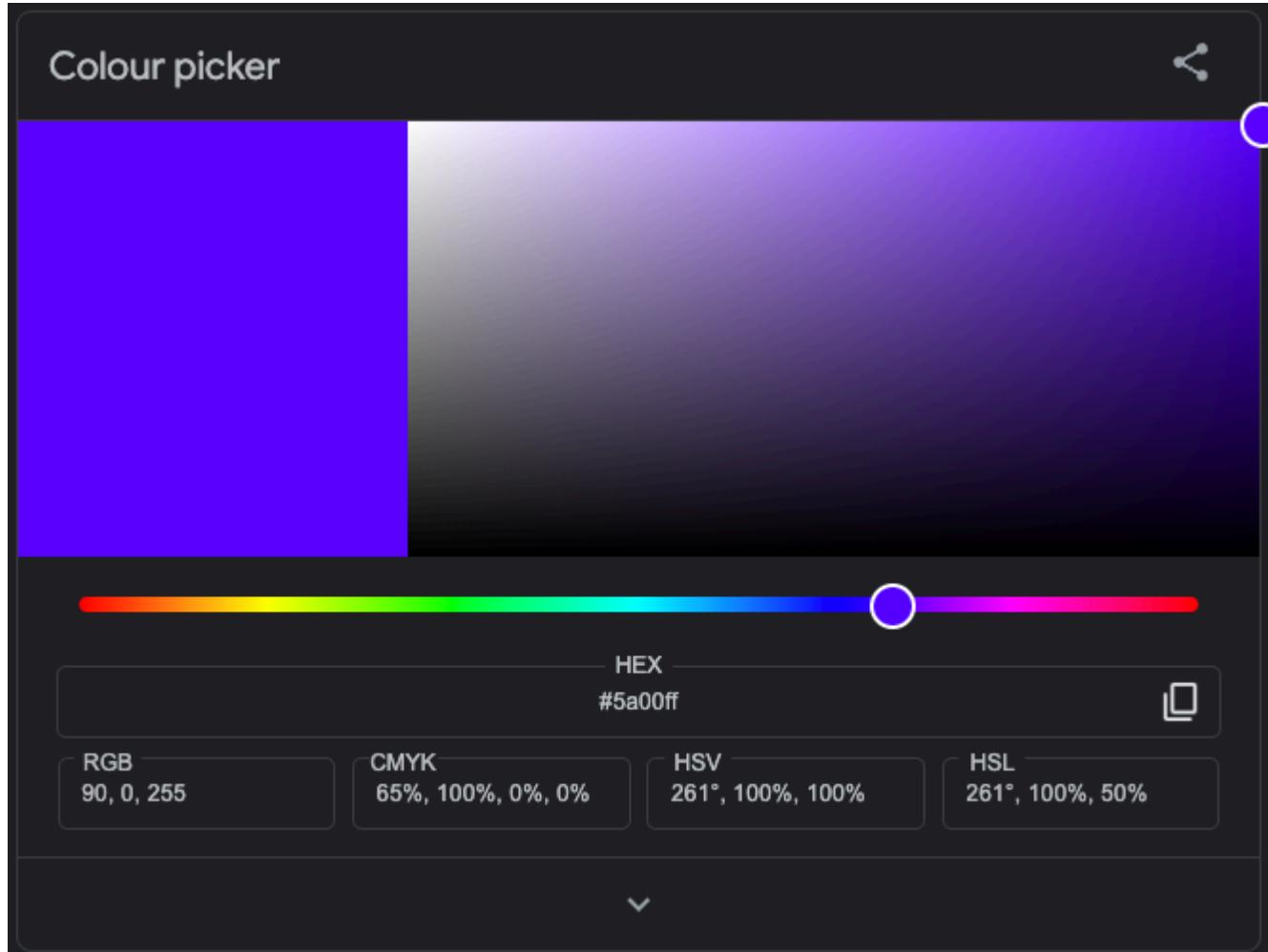


สีบานเงินมาตรฐาน BGR (255,0,0)

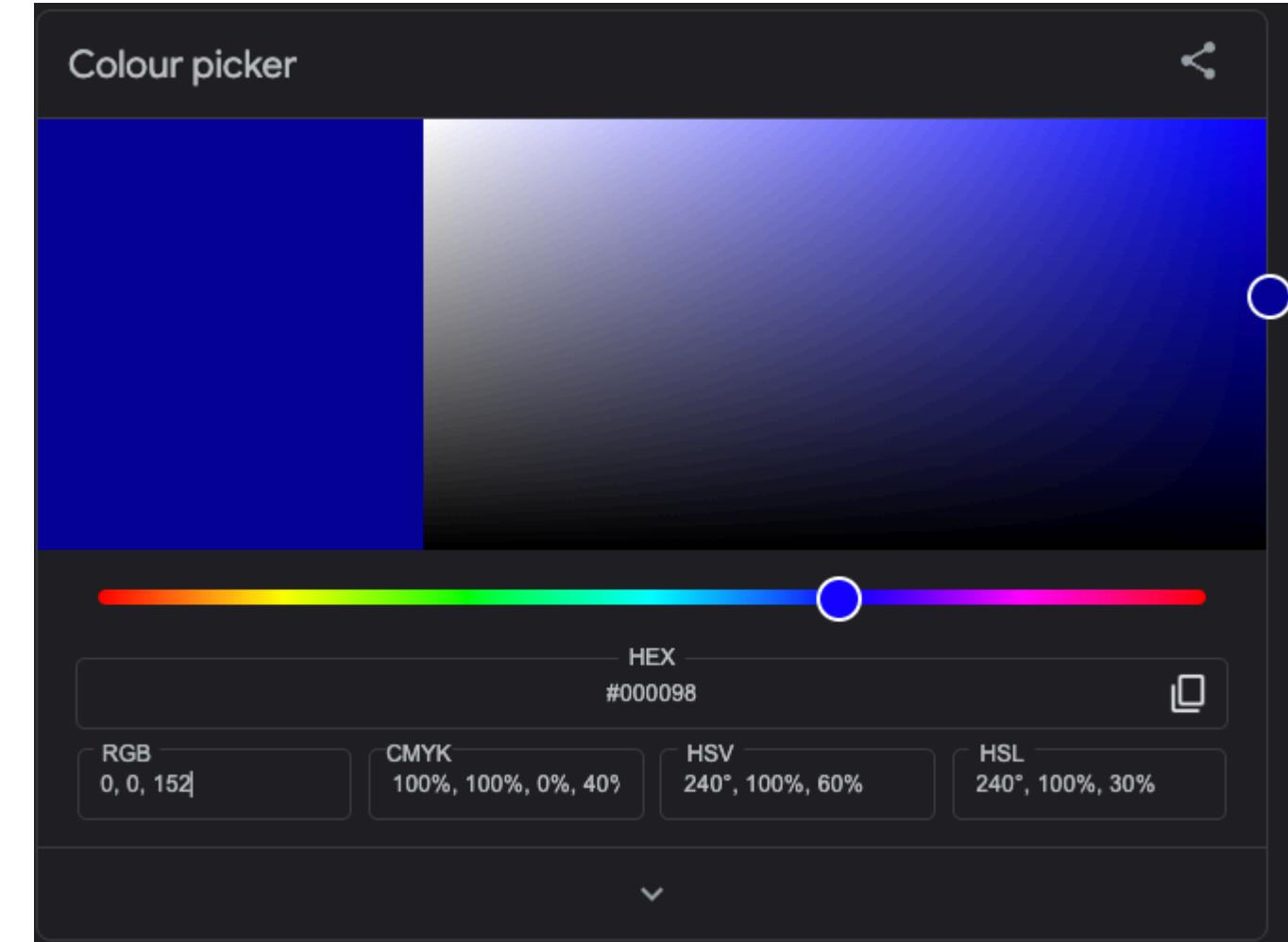


สีบานเงินอมเขียว BGR (255,255,0)

# BLUE



สีบานเงินอมแดง BGR (0,255,90)

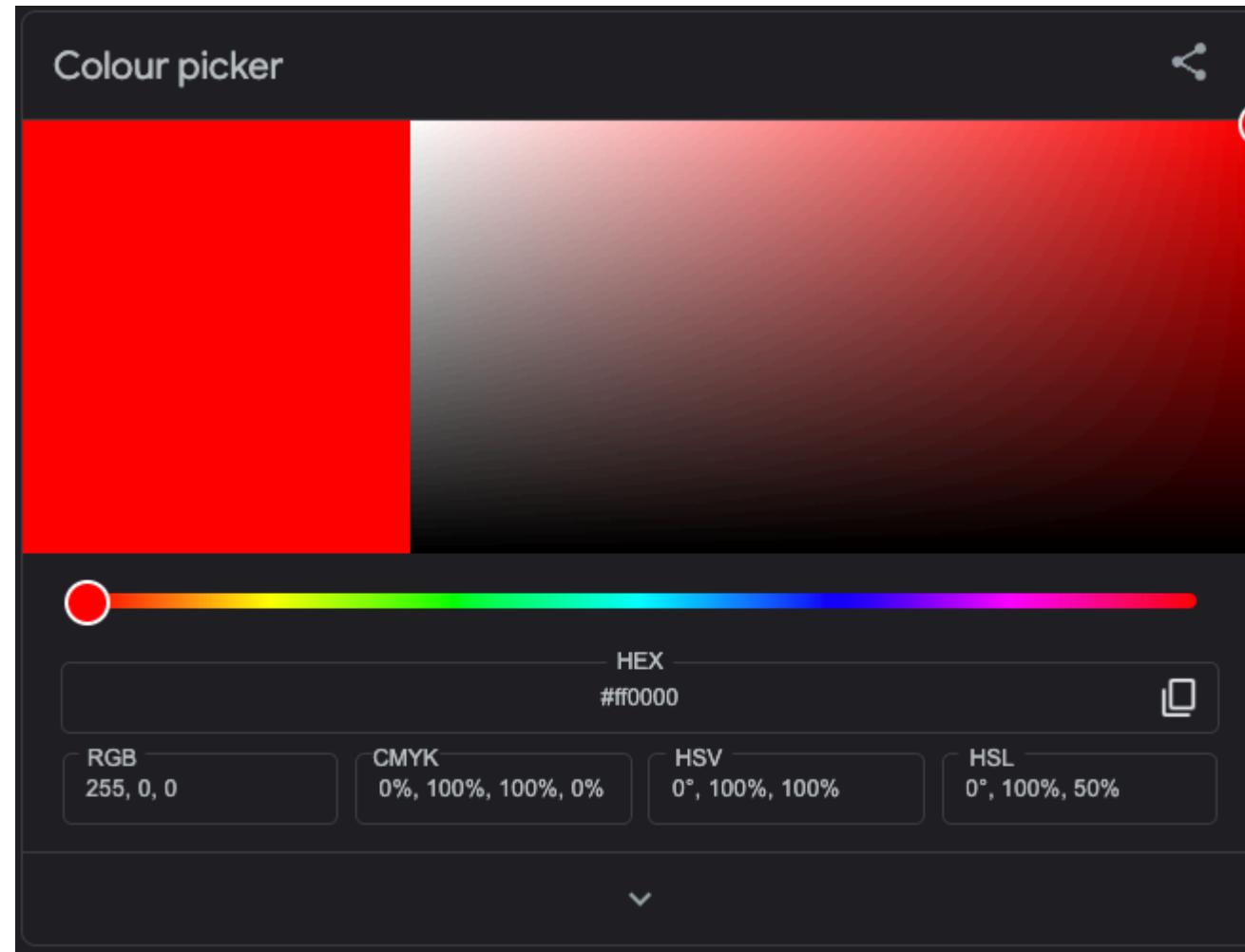


สีบานเงินอมดำ BGR (152,0,0)

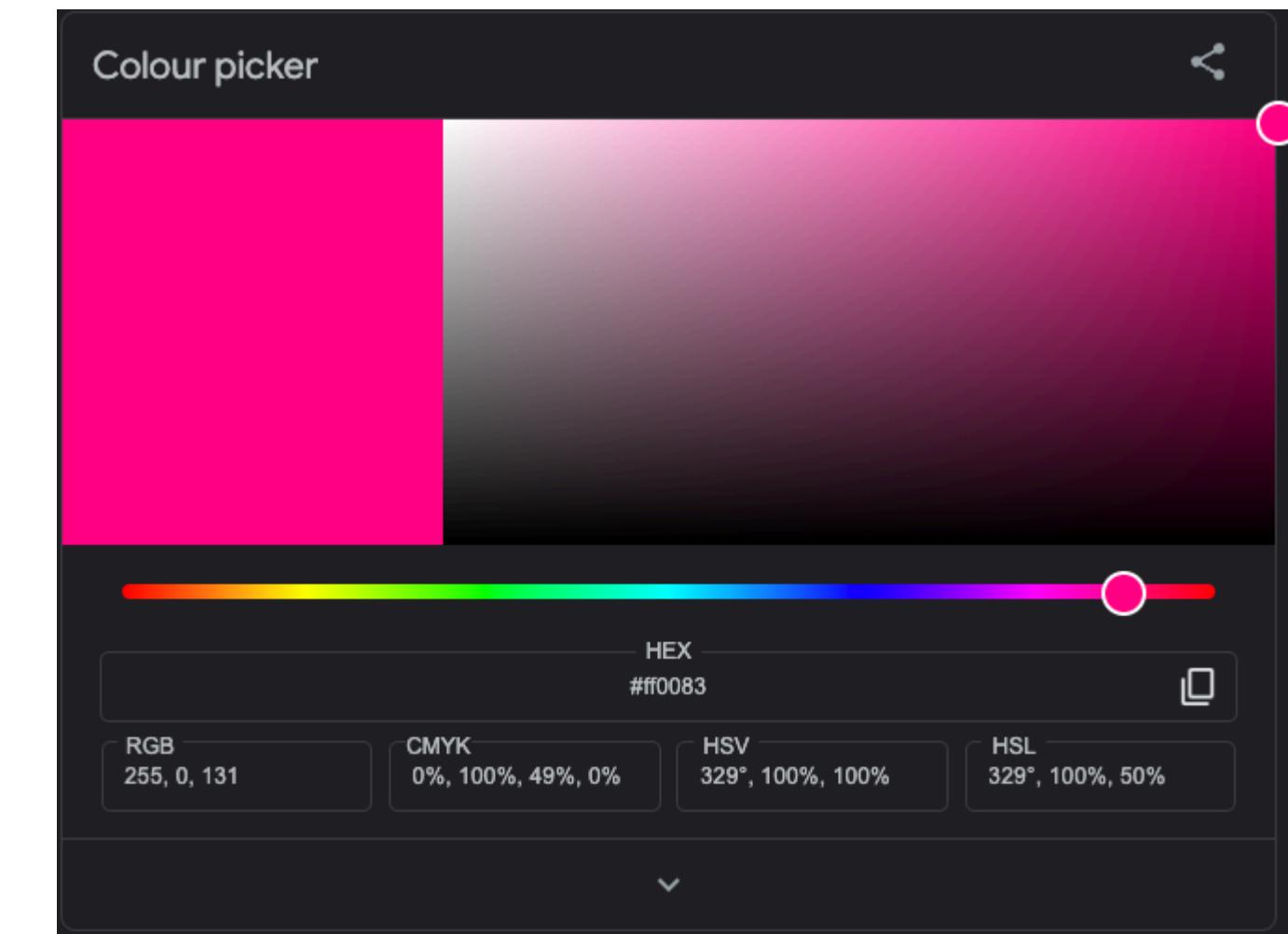
# BLUE

- ดังนั้นการกำหนดขาช่วงของสีน้ำเงินนั้น อาจจะกำหนดตามช่วงดังกล่าวได้โดย การกำหนด
- ค่าช่วง สีน้ำเงินสูงสุดที่อาจเป็นไปได้ ซึ่งจะได้ค่าเป็น upper\_blue = (255,255,90)
- ค่าช่วงสีน้ำเงินต่ำสุดที่อาจเป็นไปได้ ซึ่งจะได้ค่าเป็น lower\_blue = (152,0,0)

# RED

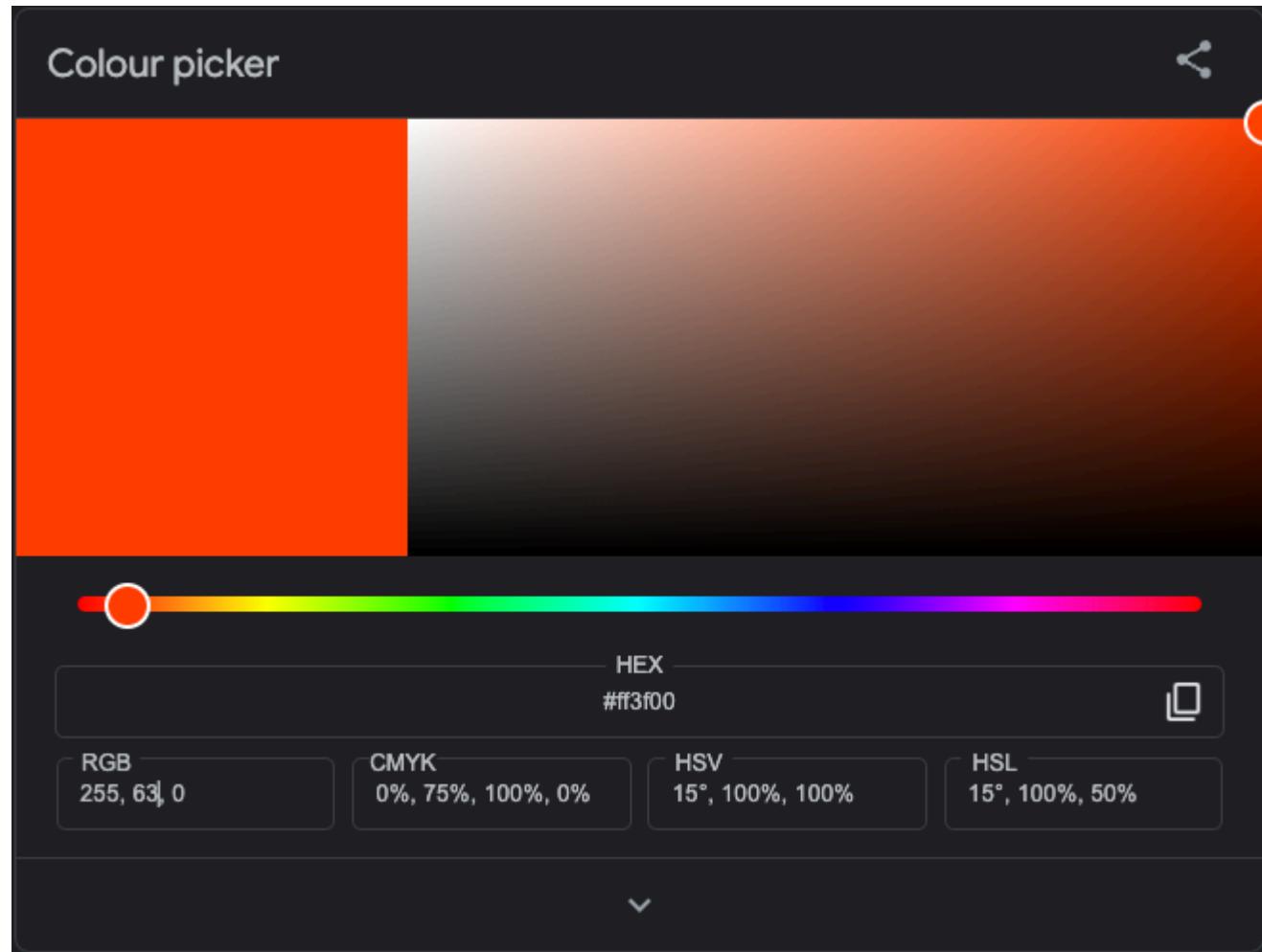


สีแดงมาตรฐาน BGR (0,0,255)

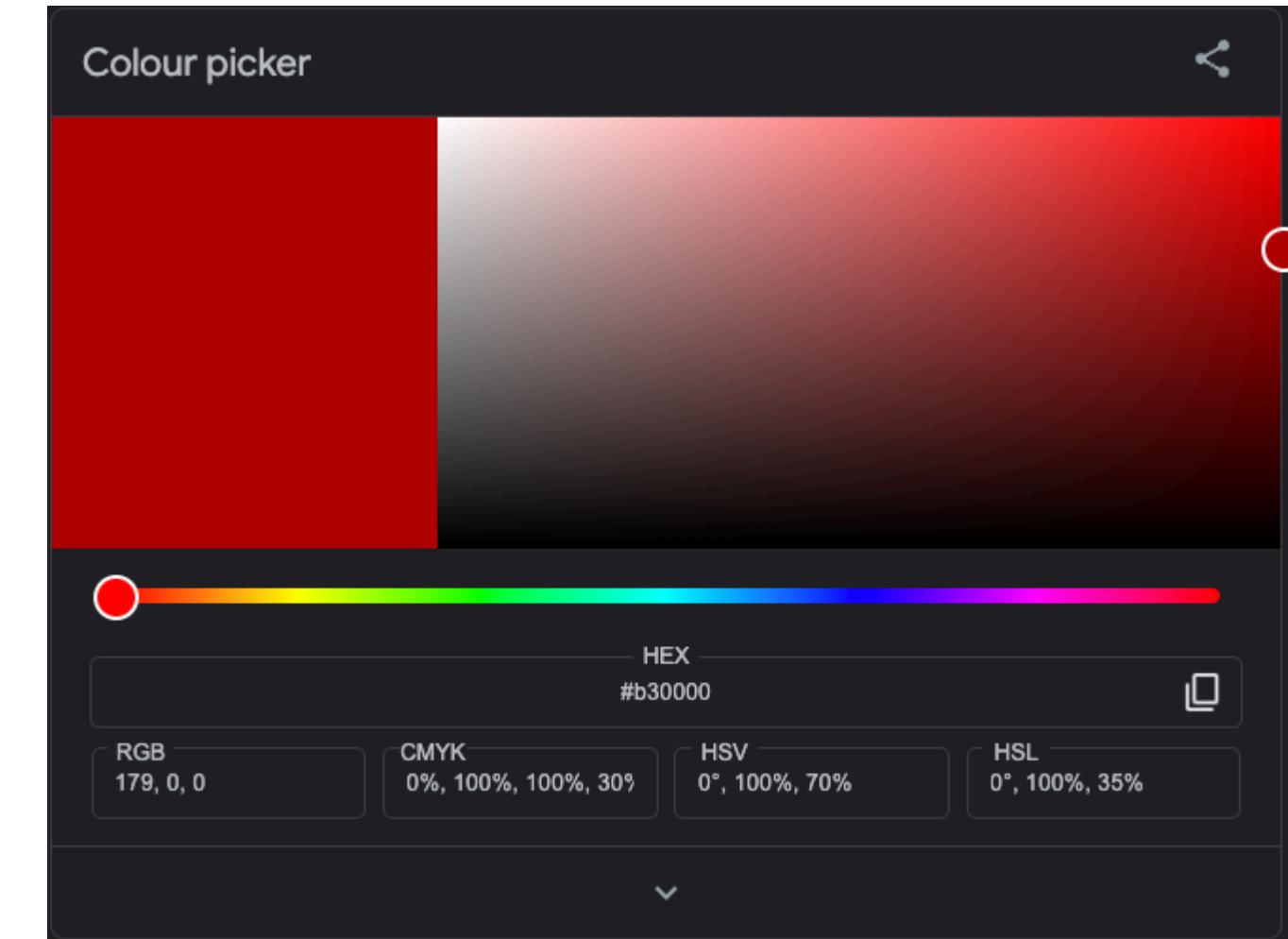


สีแดงอมน้ำเงิน BGR (131,0,255)

# RED



สีแดงอมเขียว BGR (0,63,255)



สีแดงอมดำ BGR (0,0,179)

# RED

- ดังนั้นการกำหนดหาช่วงของสีแดงนั้น อาจจะกำหนดตามช่วงดังกล่าวได้โดย การกำหนด
- ค่าช่วง สีน้ำเงินสูงสุดที่อาจเป็นไปได้ ซึ่งจะได้ค่าเป็น  $\text{upper\_red} = (131,63,255)$
- ค่าช่วงสีน้ำเงินต่ำสุดที่อาจเป็นไปได้ ซึ่งจะได้ค่าเป็น  $\text{lower\_red} = (0,0,179)$

# DETECT OBJECT

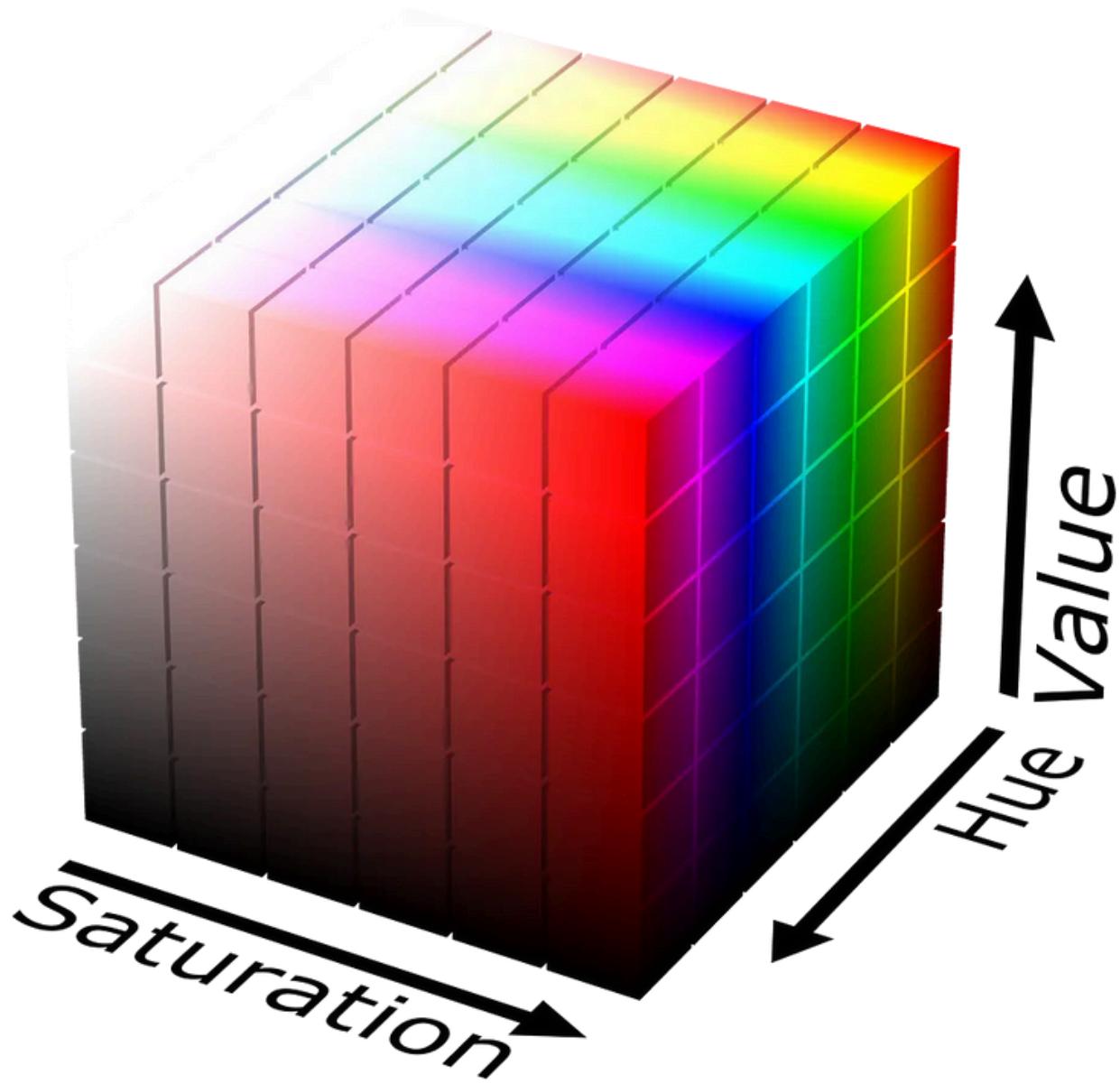
นำค่าที่กำหนดช่วงของค่ามาทำการ mask ภาพโดยใช้คำสั่ง cv.inRange() โดยนำค่าต่ำสุด และสูงสุดมาใส่เพื่อจะทำการ mask หรือ ตัดภาพสีนั้นออกไป

# CODE

```
● ● ●  
1 import cv2 as cv  
2 import numpy as np  
3  
4 img = cv.imread("image/lunares.png")  
5 upper_green = np.array([100, 255, 130])  
6 lower_green = np.array([0, 50, 0])  
7 mask = cv.inRange(img,lower_green,upper_green)  
8 result = cv.bitwise_and(img,img,mask=mask)  
9  
10 cv.imshow("origin",img)  
11 cv.imshow("detect color mask",mask)  
12 cv.imshow("result detect color",result)  
13 cv.waitKey(0)  
14 cv.destroyAllWindows()
```

# HSV filter

เนื่องจากการกรองสีออกจากวัตถุมีหลากหลายลักษณะ ไม่ว่าจะเป็นการแยก RGB, หรือการกรองผ่าน HSV แต่ HSV เป็นหลักเนื่องจากสามารถควบคุมความสว่าง และความอิ้มตัวของสีที่เราต้องการได้



# ແບບຝັກຫົດ

