Identify and explain five color models used in digital image processing.

1. **RGB (Red, Green, Blue) Model**

- **Description**: The RGB model is an additive color model, meaning it combines different intensities of red, green, and blue light to produce a wide range of colors. It is commonly used in digital displays such as monitors, televisions, and mobile devices.
- **Application**: Suitable for digital media like web design, app development, and online advertisements.

```
# Load an image in BGR format
img = cv2.imread("/content/Lenna.png")
print("Original Color Image")
cv2_imshow(img)

# Splitting into Red, Green, Blue channels
B, G, R = cv2.split(img)

# Creating black mask for visualization
zeros = np.zeros(img.shape[:2], dtype="uint8")

print("Red Channel")
cv2_imshow(cv2.merge([zeros, zeros, R]))  # Red in BGR

print("Green Channel")
cv2_imshow(cv2.merge([zeros, G, zeros]))  # Green in BGR

print("Blue Channel")
cv2_imshow(cv2.merge([B, zeros, zeros]))  # Blue in BGR
```

2. **CMY (Cyan, Magenta, Yellow) Model**

- **Description**: This is a subtractive color model used primarily in printing. It works by absorbing certain wavelengths of light and reflecting others. However, it often lacks the ability to produce true blacks, leading to the use of CMYK.
- **Application**: Primarily used in color printing, though not as common as CMYK due to its inability to produce deep blacks.
- **Code Example**: Converting RGB to CMY is straightforward, but OpenCV doesn't directly support CMY. However, you can convert RGB to CMY manually:

```
def rgb_to_cmy(r, g, b):
    c = 1 - r / 255
    m = 1 - g / 255
    y = 1 - b / 255
    return c, m, y

r, g, b = 255, 0, 0  # Red color in RGB
c, m, y = rgb_to_cmy(r, g, b)
print(f"Cyan: {c}, Magenta: {m}, Yellow: {y}")
```

3. **CMYK (Cyan, Magenta, Yellow, Black) Model**

- **Description:** An extension of the CMY model, CMYK adds black ink (Key) to improve the production of darker colors and true blacks. It is a subtractive model, meaning colors are created by absorbing certain wavelengths of light.
- **Application:** Widely used in printing industries for materials like brochures, magazines, and packaging.

```
def rgb_to_cmyk(r, g, b):
    r_prime = r / 255
    g_prime = g / 255
    b_prime = b / 255

    k = 1 - max(r_prime, g_prime, b_prime)
    if k == 1:
        c, m, y = 0, 0, 0
    else:
        c = (1 - r_prime - k) / (1 - k)
        m = (1 - g_prime - k) / (1 - k)
        y = (1 - b_prime - k) / (1 - k)
    return c, m, y, k

r, g, b = 255, 0, 0  # Red color in RGB
c, m, y, k = rgb_to_cmyk(r, g, b)
print(f"Cyan: {c}, Magenta: {m}, Yellow: {y}, Black: {k}")
```

4. **HSI (Hue, Saturation, Intensity) Model**

- **Description**: This model represents colors based on human perception, with hue referring to the color itself, saturation to its purity, and intensity to its brightness. It is useful for image processing tasks that require color manipulation based on human visual perception.
- **Application**: Used in various image processing applications where color manipulation is needed, such as enhancing or converting images.

```
import cv2
import numpy as np

# Load an image
img = cv2.imread('image.jpg')

# Convert RGB to HSV
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

cv2.imshow('HSV Image', hsv_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

5. **LAB (CIELab) Model**

- **Description**: The LAB model is designed to approximate human color vision. It consists of a lightness channel (L) and two color channels (A for red-green axis and B for blue-yellow axis). This model is device-independent, making it useful for color matching across different devices.
- **Application**: Commonly used in photography, branded products, and color reference applications where precise color reproduction is crucial.

```
import cv2
import numpy as np

# Load an image
img = cv2.imread('image.jpg')

# Convert RGB to LAB
lab_img = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)

cv2.imshow('LAB Image', lab_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6. HSV (Hue, Saturation, Value) Model

- **Description**: The **HSV model** is designed to **align with human color perception**. Instead of using RGB components, it represents colors based on:
    - **Hue (H):** The type of color (0°-360° on a color wheel).
    - **Saturation (S):** The intensity or purity of the color (0%-100%).
    - **Value (V):** The brightness of the color (0%-100%).

    - More intuitive for **color-based operations** than RGB.
- **Application**:
    - Used in **computer vision (e.g., object tracking, color segmentation)**.
    - Found in **image editing software** for color adjustments.

```
# Convert BGR to HSV
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
H, S, V = cv2.split(hsv_img)

print("Hue Channel")
cv2_imshow(H)

print("Saturation Channel")
cv2_imshow(S)

print("Value Channel")
cv2_imshow(V)
```

---

5. HSL (Hue, Saturation, Lightness) Model

- **Description**:

  Similar to HSV, but it replaces **Value (V) with Lightness (L)**.
    - **Lightness (L)**: Measures the **average brightness** of the color.
    - More evenly distributes colors in the spectrum.
    - Used in **web design** because of its balance in light intensity.
- **Application**:
    - Used in **CSS, digital graphics, and color manipulation**.
    - Found in **photo editing tools**.

```
# Convert BGR to HLS
hls_img = cv2.cvtColor(img, cv2.COLOR_BGR2HLS)
H, L, S = cv2.split(hls_img)

print("Hue Channel")
cv2_imshow(H)

print("Lightness Channel")
cv2_imshow(L)

print("Saturation Channel")
cv2_imshow(S)
```

6. YCrCb (Luminance-Chrominance) Model

- **Description**:

  The **YCrCb model** separates **luminance (Y)** from **chrominance (Cr, Cb)**, which is useful for image compression.

  - **Y:** Represents brightness (grayscale information).
  - **Cr & Cb:** Represent color differences.
  - Helps **reduce storage size while maintaining quality**.

- **Application**:
  - Used in **JPEG compression**, **MPEG video encoding**, and **broadcast television**.
  - Found in **face detection algorithms (OpenCV Haar cascades)**.

```
# Convert BGR to YCrCb
ycrcb_img = cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)
Y, Cr, Cb = cv2.split(ycrcb_img)

print("Luminance (Y) Channel")
cv2_imshow(Y)

print("Chrominance (Cr) Channel")
cv2_imshow(Cr)

print("Chrominance (Cb) Channel")
cv2_imshow(Cb)
```