# 2주차 - 디지털 영상

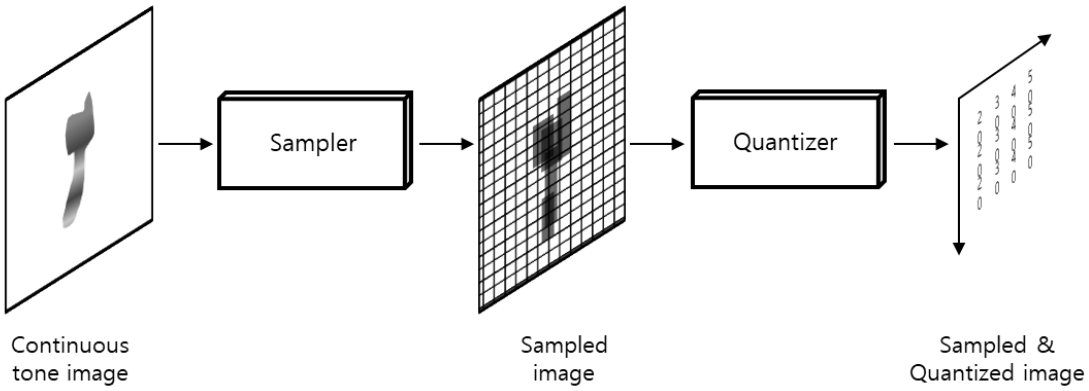
---

## ■ 아날로그 영상과 디지털 영상

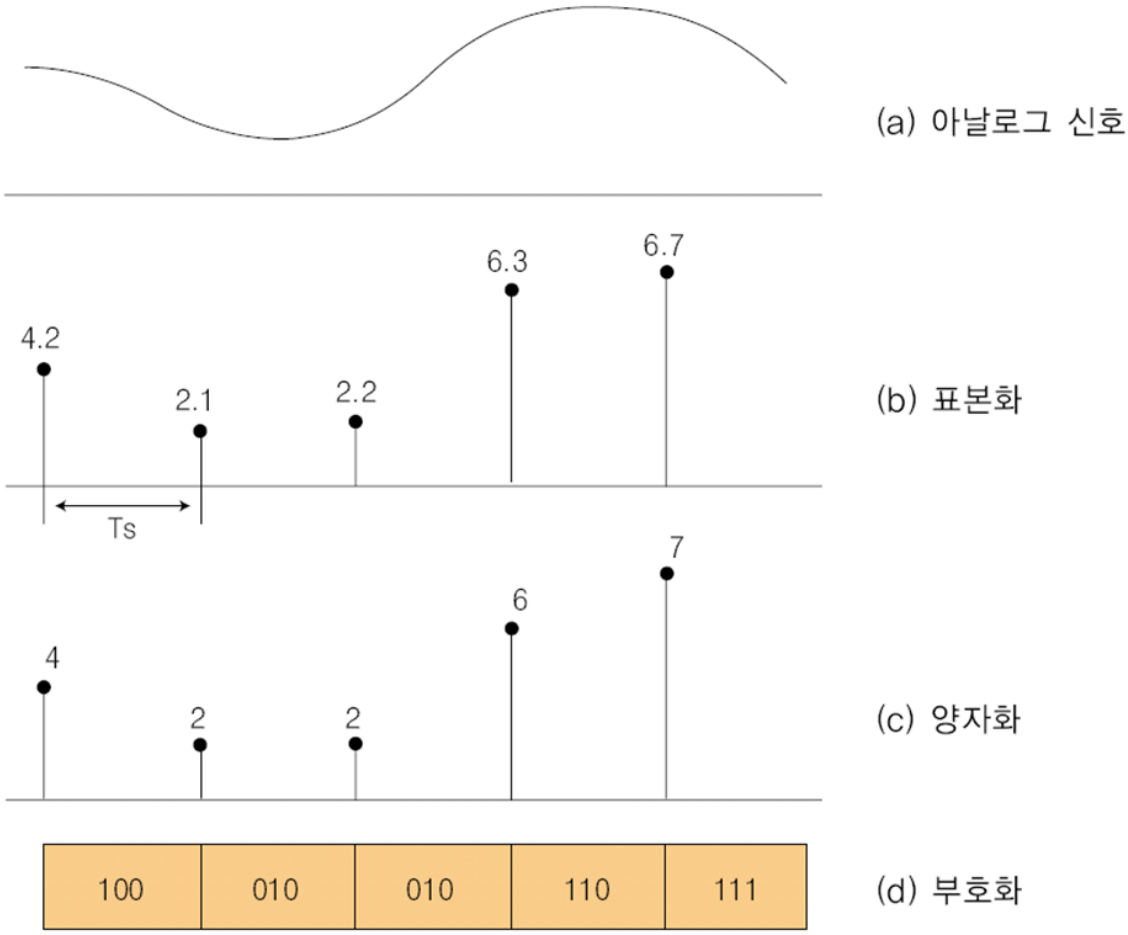
- 아날로그 영상
  - 연속 색조 영상(Continuous-Tone Image)
  - 다양한 명암과 색이 혼합되어 원래의 영상을 정확히 재현
- 디지털 영상
  - 아날로그 영상을 디지털화하는 과정으로 생성
  - 디지털화는 표본화, 양자화, 부호화로 구성
  - 밝기의 불연속점으로 구성

## ■ 영상의 디지털화

- 표본화(샘플링): 연속적인 아날로그 영상을 일정 간격으로 수치화
- 양자화: 각 샘플의 값을 정해진 수치로 대응

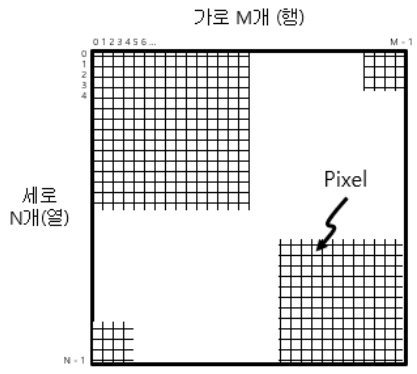



- 아날로그 신호를 디지털 신호로 디지털화하는 과정




## ■ 디지털 영상의 표현

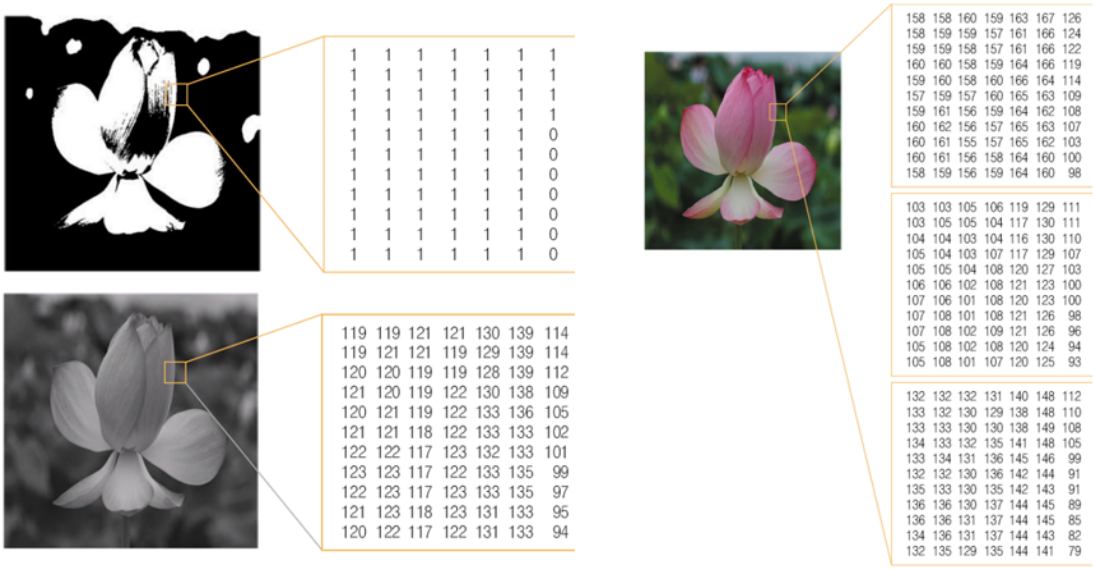
- 픽셀
  - 디지털 영상을 구성하는 최소 단위 (화소)
- 해상도
  - 영상의 가로 길이와 세로 길이의 곱




## ■ 디지털 영상의 종류

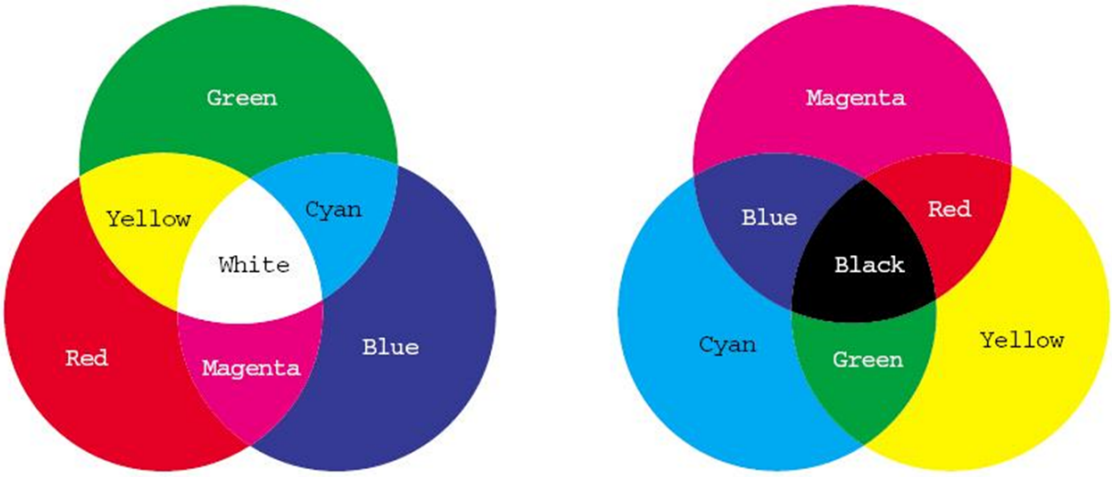
- 이진 영상 (Binary Image)
  - 화소 값이 두 가지(검정색, 흰색)만 있는 영상
- 그레이 레벨 영상(Gray-Level Image)
  - 각 화소의 밝기가 여러 단계로 보통 흑백 사진이 이에 해당

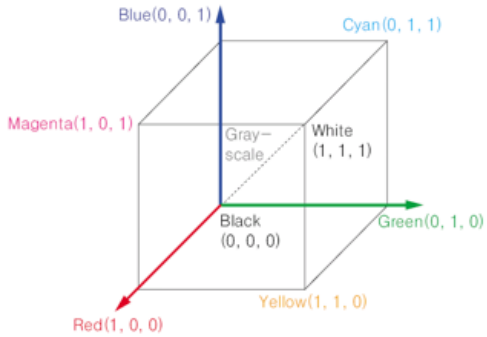
- 컬러 영상(Color Image)
  - 실제로 눈에 보이는 모습과 비슷하게 밝기와 색상을 표현하는 영상
  - 빛의 삼원색인 빨강색(R), 초록색(G), 파란색(B)을 이용하여 모든 색을 표현할 수 있음
  - 각 색을 그레이 레벨 영상처럼 독립적 형태로 처리
  - 각 색의 상호작용이 너무 커서 영상을 처리하는 데 어려움이 있음

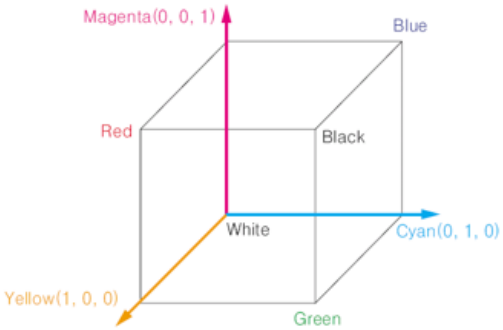



## ■ 컬러 모델

- RGB
  - 컬러 모니터, TV, 디스플레이 장치 등에서 사용
- CMYK
  - 컬러 인쇄, 출판 분야
- HSI
  - 인간의 색체 지각 능력 표현하는 응용
- YCbCr
  - 영상 압축






## ■ 디지털 영상의 다양한 저장 포맷

| 파일 포맷 | 특징 | 응용 |
|---|---|---|
| JPG | 정지 영상에 대한 압축 표준으로, 영상 정보가 헤더 정보에 포함된다. | 이미지 압축 |
| BMP | 각 픽셀을 나타내는 비트를 나열하여 이미지를 저장 및 출력하는 영상 포맷으로, 비트맵 방식이라고도 부른다. | 윈도 기반의 이미지 편집 |
| GIF | 비트맵 그래픽의 데이터 포맷으로, 웹에서 널리 쓰이는 파일 포맷이다. 다양한 환경에서 쉽게 쓸 수 있으며 간단한 애니메이션을 제작할 때도 용이하다. | 인터넷 이미지 |
| PDF | 일반 문서, 문자, 도형, 그림 등을 저장하는 전자 문서 형식의 데이터 포맷이다. | 문서 및 이미지 변환 |
| TIFF | 무손실 압축 방법으로 단일한 표준 파일 형식을 갖지 못해 업체간 서로 다른 태그를 사용, 고정 파일 형식에 비해 복잡하고 구현이 어려운 것이 단점이다. | 스캐너, 디지털 카메라 |
| OpenEXR | HDR 이미지를 저장하는 포맷으로, 채널당 16비트 이상의 컬러를 사용할 수 있다. | HDR 영상 생성 |
| PCX | 도스 기반의 프로그램에서 영상처리를 위해 사용한다. | 도스 기반의 이미지 편집 |
| PNG | 무손실 압축 방법을 이용한 영상 데이터 포맷 중 일부로, GIF의 컬러 표현 한계를 극복하기 위해 고안되었다. | 인터넷 이미지 |

## ■ 디지털 영상처리의 단계

- 저수준 영상처리
  - 디지털 영상 획득, 포맷에 맞춰 저장
- 중간 수준 영상처리
  - 영상 분할, 심볼 매핑 등 특별 목적에 따라 영상을 가공하는 과정
- 고수준 영상처리
  - 영상 해석, 영상 인식

---

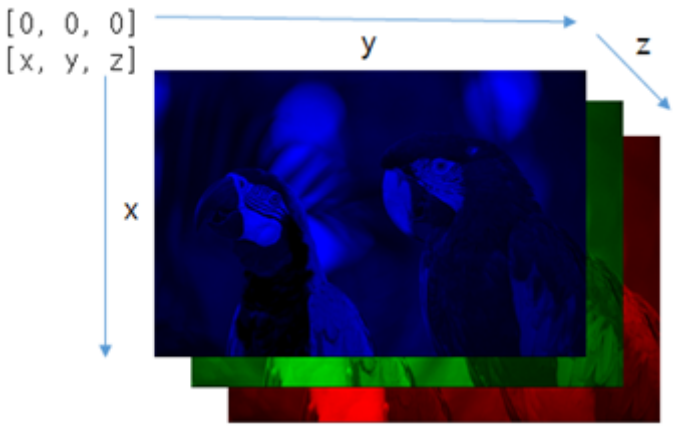
In [1]:
```
### Packages
import cv2
import numpy as np
from matplotlib import pyplot as plt
import os
```

In [2]:
```
### 출력 영상 크기
plt.rcParams["figure.figsize"] = (16,9)
### 한글 표시
plt.rcParams['font.family'] = "Gulim"   # 'AppleGothic' in mac
```

## ■ RGB 컬러

-컬러 영상의 구성

- 컬러 영상의 인덱스(index)




In [3]:
```
### 영상 읽기
img_raw = cv2.imread(r'D:WimageWlena.png')

if img_raw is None:
    print("Could not read the image.")
```

In [4]:
```
### 자료 구조
type(img_raw), img_raw.dtype, img_raw.shape
```

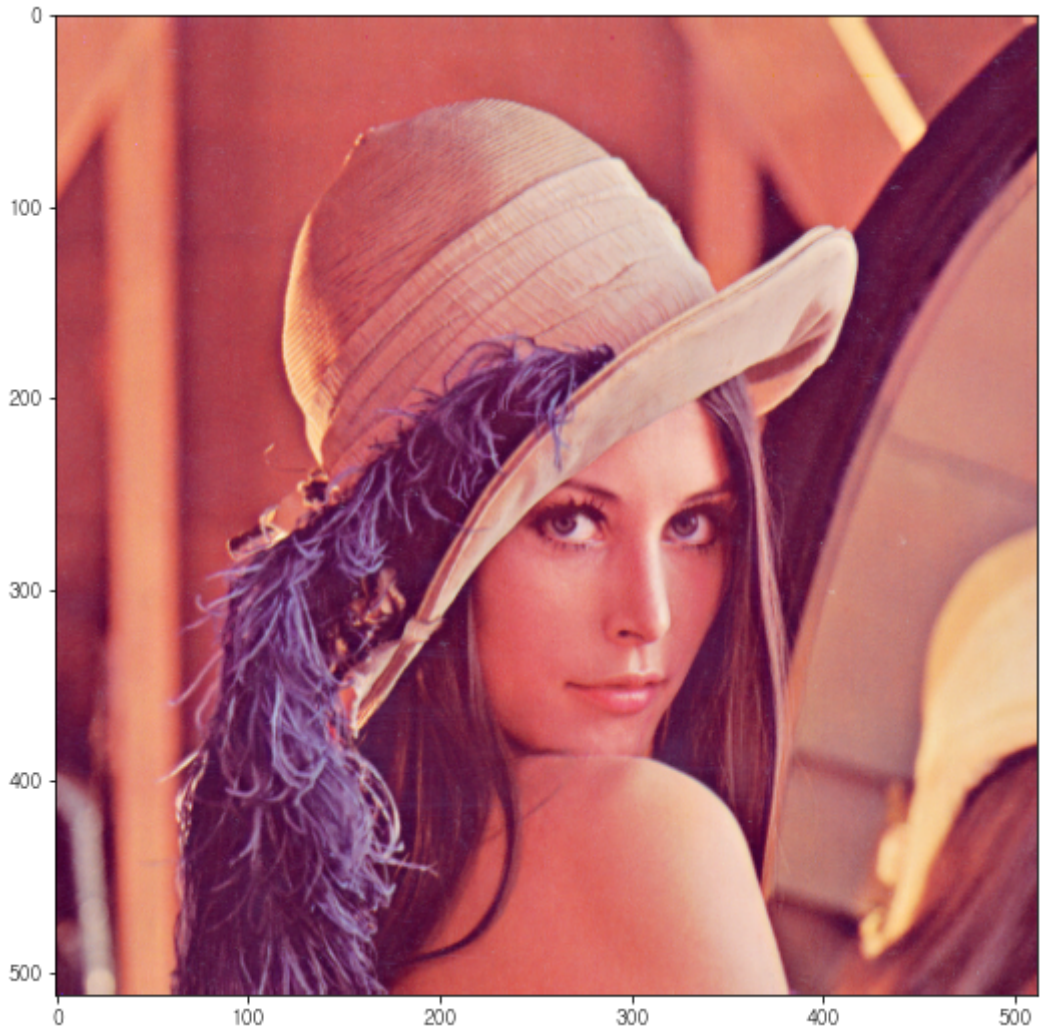
Out[4]:
```
(numpy.ndarray, dtype('uint8'), (512, 512, 3))
```

In [5]:
```
### Blue
img_raw[:,:,0]
```

Out[5]:
```
array([[125, 125, 133, ..., 122, 110,  90],
       [125, 125, 133, ..., 122, 110,  90],
       [125, 125, 133, ..., 122, 110,  90],
       ...,
       [ 60,  60,  58, ...,  84,  76,  79],
       [ 57,  57,  62, ...,  79,  81,  81],
       [ 57,  57,  62, ...,  79,  81,  81]], dtype=uint8)
```

In [6]:
```
### BGR → RGB 변환
img_rgb = cv2.cvtColor(img_raw, cv2.COLOR_BGR2RGB)
```

```
plt.imshow(img_rgb)
plt.show()
```



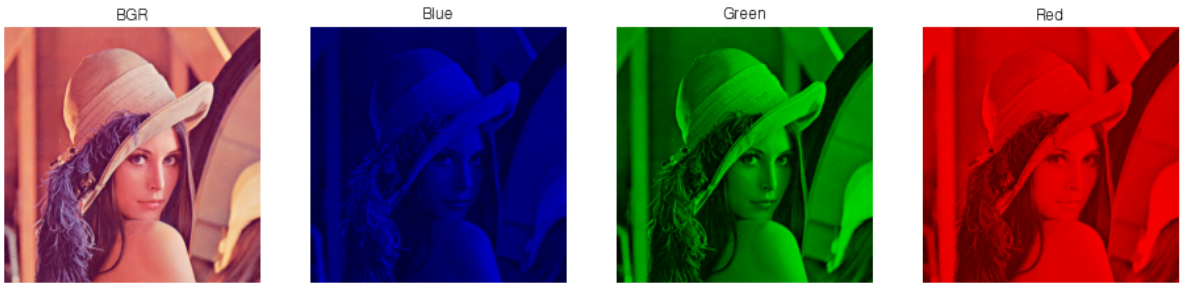

In [7]:
```
### 영상 분리
B, G, R = cv2.split(img_raw)
```

In [8]:
```
### RGB 영상으로 출력
titles = ["BGR", "Blue", "Green", "Red"]
images = [img_raw, B, G, R]
for i in range(len(images)):
    plt.subplot(1,4,i+1)
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```


In [9]:
```
### 영상 분리
B = np.zeros(img_raw.shape, dtype="uint8")
G = B.copy()
```

```
R = B.copy()
B[:,:,0] = img_raw[:,:,0]
G[:,:,1] = img_raw[:,:,1]
R[:,:,2] = img_raw[:,:,2]
```

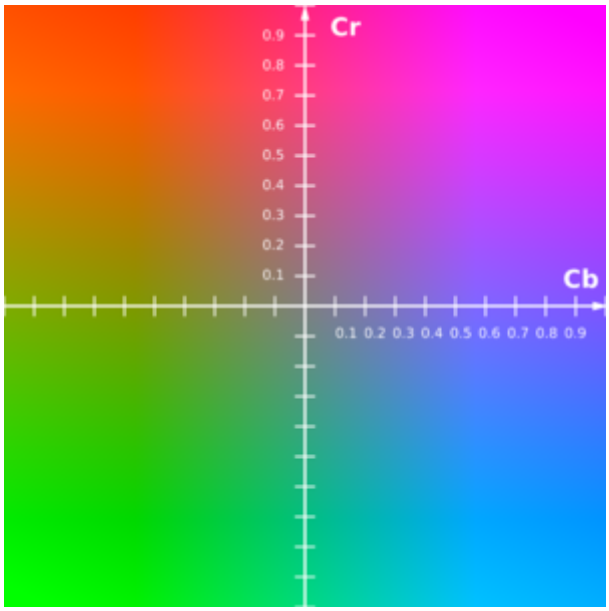
In [10]:
```
### RGB 영상으로 출력
titles = ["BGR", "Blue", "Green", "Red"]
images = [img_raw, B, G, R]
for i in range(len(images)):
    plt.subplot(1,4,i+1)
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```




---

## ■ YCrCb(YCC) 컬러

- 인간의 눈이 명도에 민감하고 색도 성문에 덜 민감한 시각적인 특성을 고려한 것
- 흑백(grayscale) 영상과 색차로 구성
- 디지털 표준 텔레비전에 사용: ITU-R BT.601 형식
- 비디오 및 스틸 이미지 압축과 전송(예: MPEG, JPEG)에 적합한 색 정보의 디지털 인코딩에 사용
  - $Y$(Luminance): 밝기
  - $Cr$(Chroma Red): 밝기와 빨간색의 색차
  - $Cb$(Chroma Blue): 밝기와 파란색의 색차




- RGB ↔ YCrCb JPEG (or YCC)
  - $Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$

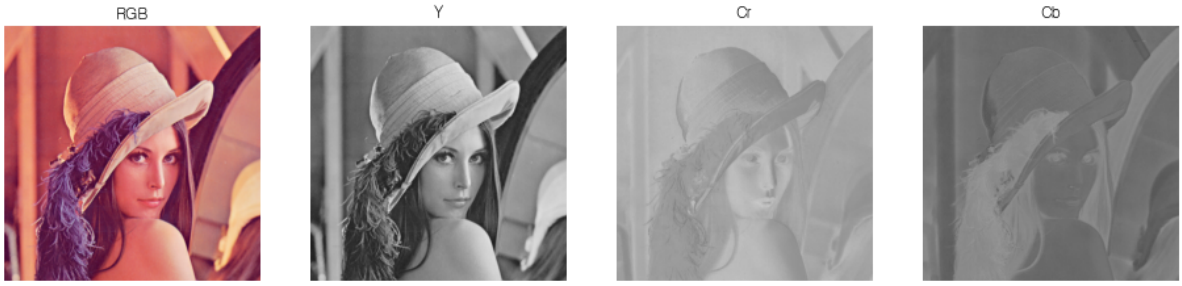
- $Cr \leftarrow (R - Y) \cdot 0.713 + delta$
- $Cb \leftarrow (B - Y) \cdot 0.564 + delta$
- $R \leftarrow Y + 1.403 \cdot (Cr - delta)$
- $G \leftarrow Y - 0.714 \cdot (Cr - delta) - 0.344 \cdot (Cb - delta)$
- $B \leftarrow Y + 1.773 \cdot (Cb - delta)$

$$delta = \begin{cases} 128 & \text{for 8-bit images} \\ 32768 & \text{for 16-bit images} \\ 0.5 & \text{for for floating-point images} \end{cases}$$

In [11]:
```
### BGR → YCrCb 변환
img_ycc = cv2.cvtColor(img_raw, cv2.COLOR_BGR2YCrCb)
```

In [12]:
```
### 영상 분리
Y, Cr, Cb = cv2.split(img_ycc)
```

In [13]:
```
### 영상 출력
titles = ["RGB", "Y", "Cr", "Cb"]
images = [img_raw, Y, Cr, Cb]
for i in range(len(images)):
    plt.subplot(1,4,i+1)
    img_rgb = cv2.cvtColor(images[i], cv2.COLOR_BGR2RGB)
    plt.imshow(img_rgb)
    plt.title(titles[i])
    plt.axis('off')
plt.show()
```




---

## ■ 조각 영상

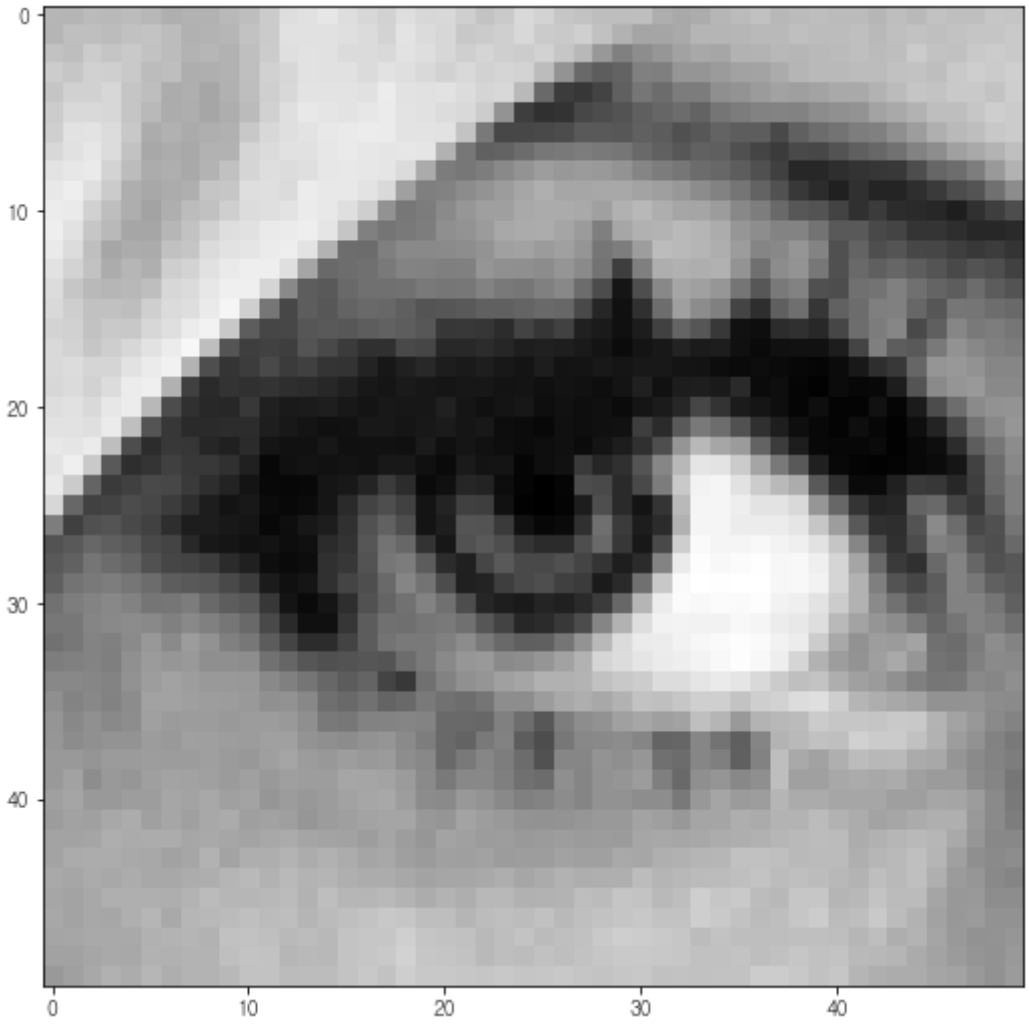
- 영상의 일부 영역을 추출한 영상

In [14]:
```
### Grayscale
img_gray = cv2.cvtColor(img_raw, cv2.COLOR_BGR2GRAY)
img_gray.shape
```

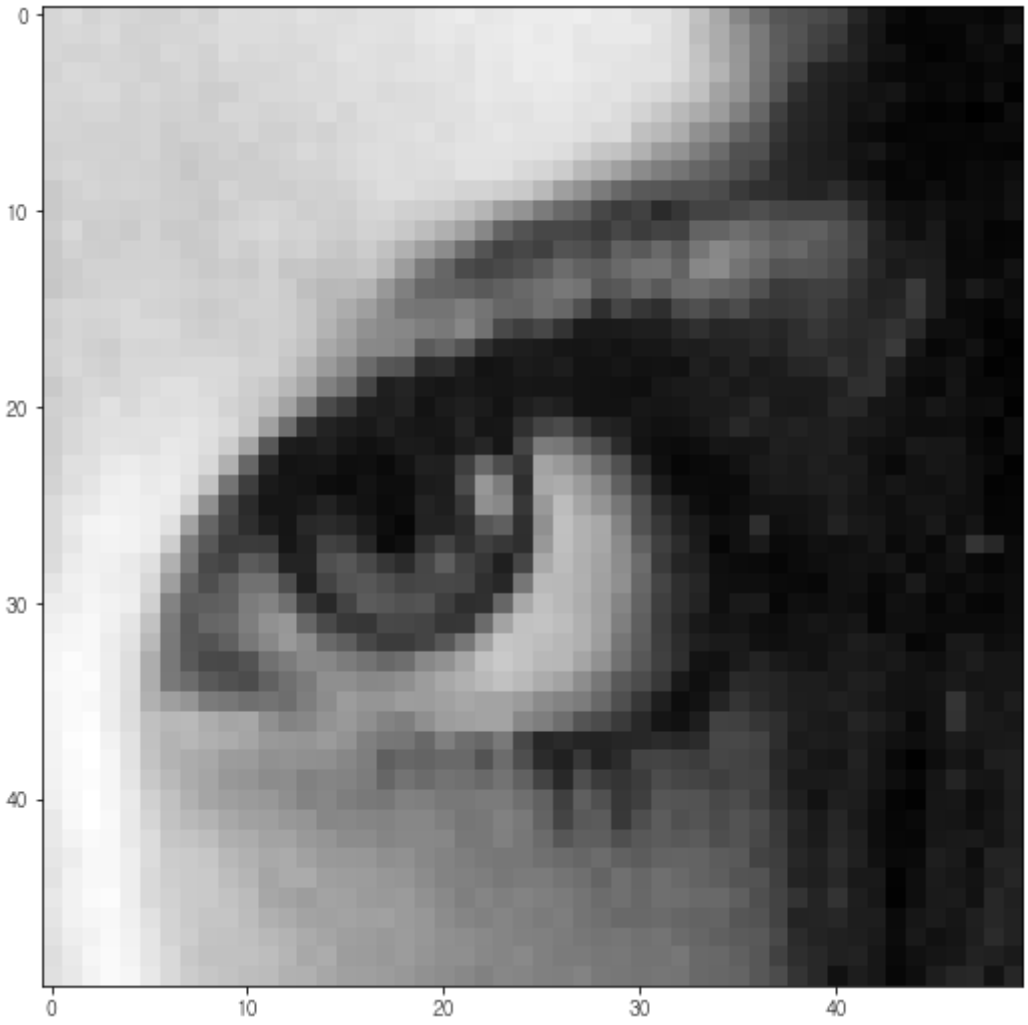
Out[14]:
```
(512, 512)
```

In [15]:
```
### 영상 출력 - grayscale
plt.imshow(img_gray, 'gray')
plt.show()
```

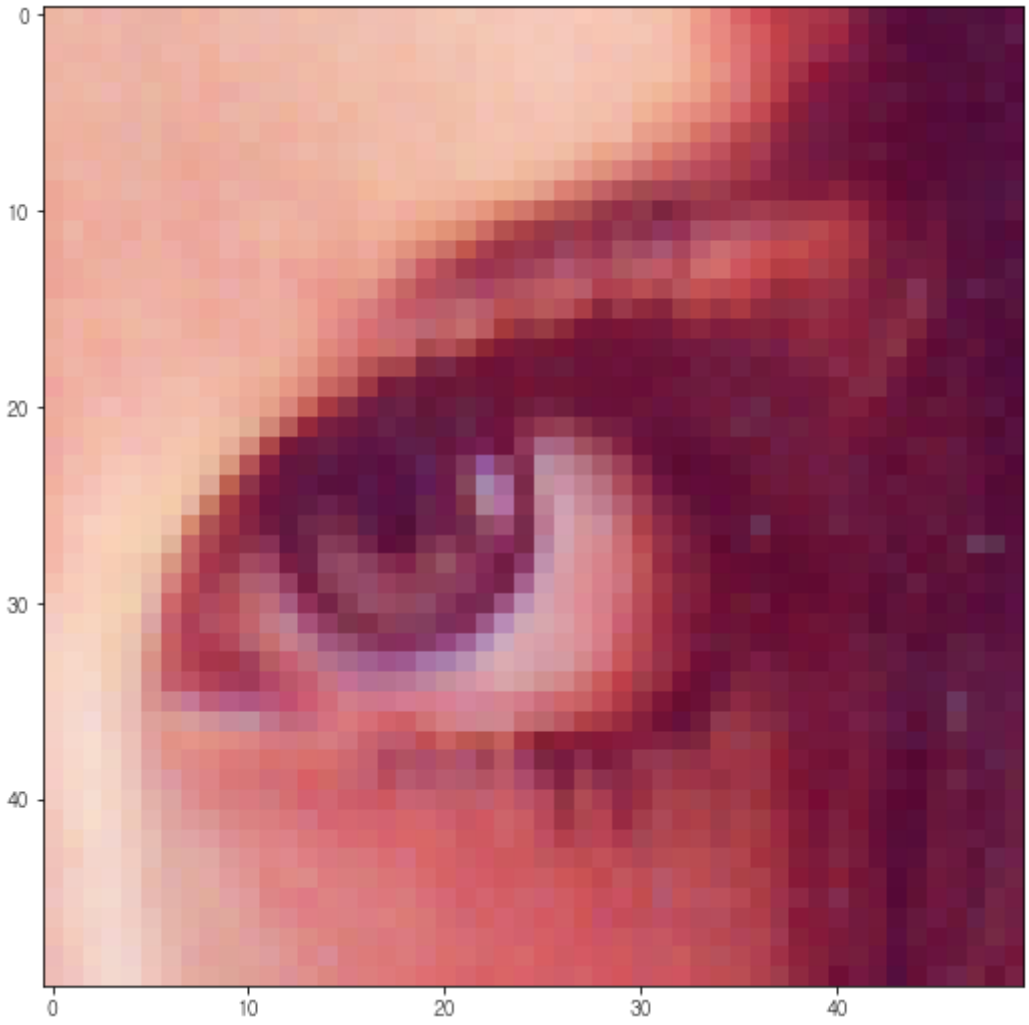

In [16]:
```
### 조각 영상(1) - grayscale
img_gray_sub = img_gray[240:290, 240:290]
plt.imshow(img_gray_sub, "gray")
plt.show()
```




In [17]:
```
### 조각 영상(2) - grayscale
img_gray_sub = img_gray[240:290, 310:360]
plt.imshow(img_gray_sub, "gray")
plt.show()
```




In [18]:
```
### 조각 영상(3) - color
img_raw_sub = img_raw[240:290, 310:360, :]
plt.imshow(cv2.cvtColor(img_raw_sub, cv2.COLOR_BGR2RGB))
plt.show()
```


0
10
20
30
40
0
10
20
30
40