



OpenCV의 사용

최석재 lingua@naver.com

OpenCV

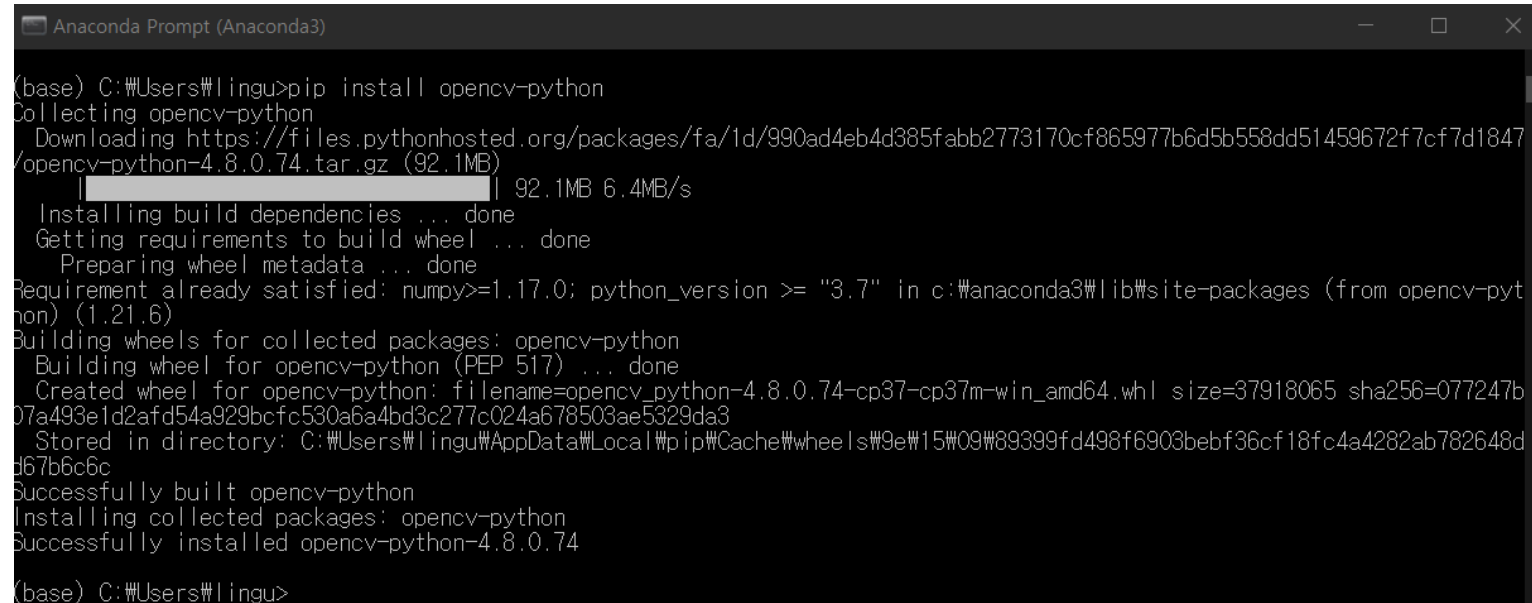
Open Source Computer Vision Library

OpenCV *Open Source Computer Vision Library*

- 컴퓨터 비전을 빠른 속도로 처리하기 위한 라이브러리
- 1999년, Intel 에서 개발된 이후, 많은 업데이트가 이루어졌음
- 이미지 및 동영상 파일에 대한 전처리에 OpenCV가 자주 사용된다
- 우선 기본 사용법을 알아본다

Local PC에 설치

- 설치가 매우 어려웠으나, 최근 pip install 로 쉽게 설치할 수 있게 되었음
- Colab에는 이미 설치되어 있음
- Anaconda Prompt에서 다음의 명령어로 설치
- `pip install opencv-python`



```
Anaconda Prompt (Anaconda3)

(base) C:\Users\#lingu>pip install opencv-python
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/fa/1d/990ad4eb4d385fabb2773170cf865977b6d5b558dd51459672f7cf7d1847/opencv-python-4.8.0.74.tar.gz (92.1MB)
    |#####| 92.1MB 6.4MB/s
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing wheel metadata ... done
Requirement already satisfied: numpy>=1.17.0; python_version >= "3.7" in c:\#anaconda3\lib\site-packages (from opencv-python) (1.21.6)
Building wheels for collected packages: opencv-python
  Building wheel for opencv-python (PEP 517) ... done
  Created wheel for opencv-python: filename=opencv_python-4.8.0.74-cp37-cp37m-win_amd64.whl size=37918065 sha256=077247b07a493e1d2afd54a929bcfc530a6a4bd3c277c024a678503ae5329da3
  Stored in directory: C:\Users\#lingu\AppData\Local\pip\Cache\wheels\9e\15\09\89399fd498f6903bebf36cf18fc4a4282ab782648dd67b6c6c
Successfully built opencv-python
Installing collected packages: opencv-python
Successfully installed opencv-python-4.8.0.74

(base) C:\Users\#lingu>
```

Local PC에서 이미지 불러오기

```
import cv2
```

```
print('OpenCV Version:', cv2.__version__)
```

```
# 기본 모드로 이미지 읽기
```

```
img = cv2.imread('D:/Downloads/mountain.jpg')
```

```
# 윈도우에 표시될 텍스트 설정
```

```
window_name = 'mountain'
```

```
# 이미지 그리기
```

```
cv2.imshow(window_name, img)
```

```
# 사용자의 반응을 기다리는 함수
```

```
# 파이썬 커널과의 충돌을 막기 위해 필요
```

```
cv2.waitKey(0)
```

```
# 모든 윈도우를 닫기
```

```
cv2.destroyAllWindows()
```

```
pytest_img > opencv > mountain.jpg
```

cv2.imread() 는 이미지를 읽어서 NumPy 배열로 반환

cv2.imshow()는 NumPy 배열을 읽어서 이미지를 화면에 출력



사이즈 조절하기

이미지 축소하여 그리기

```
img_resize = cv2.resize(img, dsize=(600, 400))  
cv2.imshow(window_name, img_resize)
```

사용자의 반응을 기다리는 함수

파이썬 커널과의 충돌을 막기 위해 필요

```
cv2.waitKey(0)
```

모든 윈도우를 닫기

```
cv2.destroyAllWindows()
```



Colab에서 이미지 불러오기

```
from google.colab import drive  
drive.mount('/content/gdrive')
```

경로 변경

```
%cd /content/gdrive/MyDrive/pytest_img/opencv/
```

Colab에서 이미지 불러오기

```
import cv2
from google.colab.patches import cv2_imshow
```

```
print('OpenCV Version:', cv2.__version__)
```

기본 모드로 이미지 읽기

```
img = cv2.imread('mountain.jpg')
```

이미지 그리기

```
cv2_imshow(img)
```

Colab에서는 텍스트 설정 부분이 없으며,
cv2.imshow(**name**, img) → cv2_imshow(img) 이다

사용자의 반응을 기다리는 함수

파이썬 커널과의 충돌을 막기 위해 필요

```
cv2.waitKey(0)
```

모든 윈도우를 닫기

```
cv2.destroyAllWindows()
```



사이즈 조절하기

이미지 축소하여 그리기

```
img_resize = cv2.resize(img, dsize=(600, 400))  
cv2.imshow(img_resize)
```

사용자의 반응을 기다리는 함수

파이썬 커널과의 충돌을 막기 위해 필요

```
cv2.waitKey(0)
```

모든 윈도우를 닫기

```
cv2.destroyAllWindows()
```



스타일 변화주기

이미지 축소

```
img_resize = cv2.resize(img, dsize=(600, 400))
```

이미지 스타일을 변경하여 그리기

sigma_s: 이미지를 얼마나 부드럽게 할 것인지

sigma_r: 외곽선을 얼마나 부드럽게 할 것인지

```
img_style = cv2.stylization(img_resize, sigma_s=10, sigma_r=1)
```

```
cv2.imshow(img_style)
```

사용자의 반응을 기다리는 함수

파이썬 커널과의 충돌을 막기 위해 필요

```
cv2.waitKey(0)
```

모든 윈도우를 닫기

```
cv2.destroyAllWindows()
```



채널 다루기

GRAY 컬러로 표현

흑백으로 이미지 읽기

```
img_gray = cv2.imread('/content/gdrive/MyDrive/pytest_img/opencv/mountain.jpg', cv2.IMREAD_GRAYSCALE)
```

이미지 축소하여 그리기

```
img_resize = cv2.resize(img_gray, dsize=(600, 400))
```

```
cv2.imshow(img_resize)
```

사용자의 반응을 기다리는 함수

파이썬 커널과의 충돌을 막기 위해 필요

```
cv2.waitKey(0)
```

모든 윈도우를 닫기

```
cv2.destroyAllWindows()
```



GRAY 컬러로 표현

cvtColor()를 이용하여 이미지의 컬러를 이후에 변경할 수도 있다

이미지 축소하여 그리기

```
img_resize = cv2.resize(img, dsize=(600, 400))
```

GRAY 컬러로 변경

```
img_resize_gray = cv2.cvtColor(img_resize, cv2.COLOR_BGR2GRAY)
```

```
cv2.imshow(img_resize_gray)
```

사용자의 반응을 기다리는 함수

파이썬 커널과의 충돌을 막기 위해 필요

```
cv2.waitKey(0)
```

모든 윈도우를 닫기

```
cv2.destroyAllWindows()
```



채널 확인

- 흑백으로 읽은 이미지는 단일 채널이어서 2D 데이터이다
- 앞에서 컬러로 읽은 이미지는 3D, 3채널을 갖는다
- `print(img_resize_gray.shape)` `# (400, 600)`

HSV 표현

- cvt.color() 주요 함수를 알아본다
- BGR에서 HSV로 변환
- 색상(Hue), 색상의 강도인 채도(Saturation), 색상의 밝기인 명도(Value)로 표현

이미지 축소하여 그리기

```
img_resize = cv2.resize(img, dsize=(600, 400))
```

BGR에서 HSV로 변환

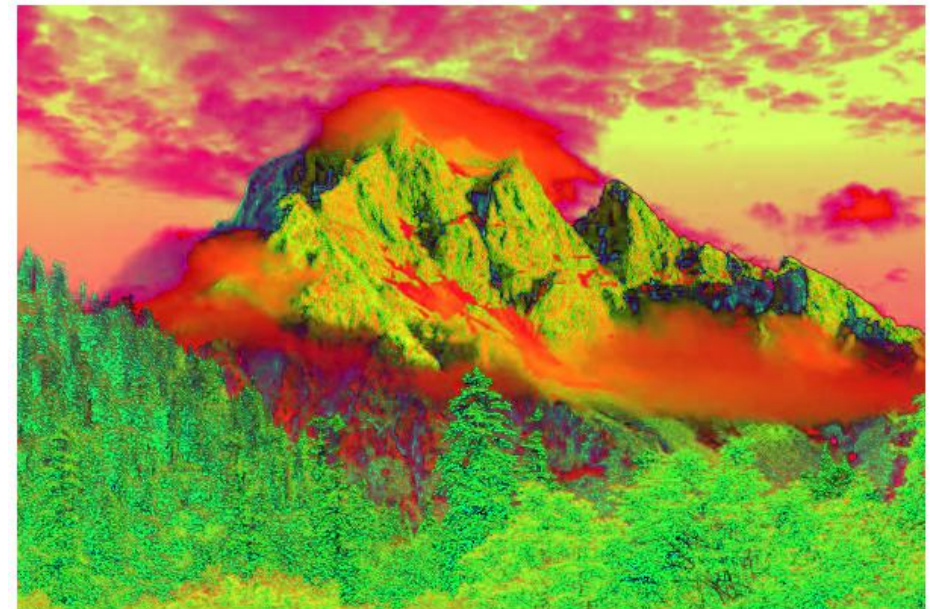
```
img_hsv = cv2.cvtColor(img_resize, cv2.COLOR_BGR2HSV)
```

```
cv2.imshow(img_hsv)
```

```
print(img_hsv.shape)
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```



(400, 600, 3)

RGB 표현

- OpenCV는 기본적으로 BGR 색상 순서를 사용한다
- RGB 순서를 사용하는 라이브러리와 함께 사용하려면 그에 맞춰 순서를 바꿔야 한다

이미지 축소하여 그리기

```
img_resize = cv2.resize(img, dsize=(600, 400))
```

BGR에서 RGB로 변환

```
img_rgb = cv2.cvtColor(img_resize, cv2.COLOR_BGR2RGB)
```

```
cv2.imshow('img_rgb', img_rgb)
```

```
print(img_rgb.shape)
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```



(400, 600, 3)

이미지 정보 확인

- 먼저 이미지의 높이, 너비, 채널을 확인한다

```
img_resize = cv2.resize(img, dsize=(600, 400))  
cv2_imshow(img_resize)
```

```
print("높이:", img_resize.shape[0])           # 400  
print("너비:", img_resize.shape[1])           # 600  
print("채널:", img_resize.shape[2])           # 3
```

```
cv2.waitKey(0)  
cv2.destroyAllWindows()
```



높이: 400
너비: 600
채널: 3

채널 분리

- 이미지에서 Blue, Green, Red 채널을 분리하여 표현해본다

```
img_resize = cv2.resize(img, dsize=(600, 400))
```

```
# BGR 채널을 분리
```

```
B, G, R = cv2.split(img_resize)
```

```
# Green과 Red 채널을 0으로 설정하여 Blue 채널만 표시
```

```
G[:] = 0
```

```
R[:] = 0
```

```
# Blue 채널만 있는 이미지 생성
```

```
blue_image = cv2.merge([B, G, R])
```

```
cv2.imshow(blue_image)
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```



채널 분리

- 같은 방식으로 Green과 Red도 분리해본다



특정 위치 표시

특정 위치 컬러 정보 표시

- 특정 위치의 컬러 정보를 표현하려면 다음과 같이 한다

원하는 픽셀 위치 지정

x = 50

y = 100

해당 위치의 BGR 색상 정보 얻기

(b, g, r) = img_resize[y, x]

print(f"Pixel at (x={x}, y={y}) is B: {b}, G: {g}, R: {r}")

Pixel at (x=50, y=100) is B: 235, G: 193, R: 130

특정 위치 그리기

- 관심 있는 영역만 그려본다

X축과 Y축 모두 50~100 사이를 찾는다

start_point = 50

end_point = 100

roi = img[start_point:end_point, start_point:end_point]

cv2.imshow(roi)

cv2.waitKey(0)

cv2.destroyAllWindows()



특정 위치 그리기

- 관심 있는 영역을 사각형으로 표시해본다
- 원본 객체에 직접 그리기 때문에 사본을 만들어 그리는 것이 좋다

사각형 좌표 지정

start_point = (50, 50) *# (x1, y1)*

end_point = (100, 100) *# (x2, y2)*

color = (255, 0, 0) *# 파란색*

thickness = 2 *# 선의 두께*

img_copy = img.copy() *# 원본 이미지 복사*

사각형 그리기

cv2.rectangle(img_copy, start_point, end_point, color, thickness)

cv2.imshow(img) *# 이미지 표시 및 창닫기*

cv2.waitKey(0)

cv2.destroyAllWindows()



특정 위치 그리기

- 관심 있는 영역을 원으로 표시해본다
- 원본 객체에 직접 그리기 때문에 사본을 만들어 그리는 것이 좋다

원을 중심 좌표 지정

center_coordinates = (50, 50)

radius = 40 *# 원의 반지름*

color = (255, 0, 0) *# 파란색*

thickness = 2 *# 선의 두께*

img_copy = img.copy() *# 원본 이미지 복사*

원 그리기

cv2.circle(img_copy, center_coordinates, radius, color, thickness)

cv2.imshow(img_copy) *# 이미지 표시 및 창닫기*

cv2.waitKey(0)

cv2.destroyAllWindows()



텍스트 그리기

- 선과 텍스트를 함께 그려본다

선을 그릴 시작점과 끝점 지정

start_point = (50, 50) *# 시작점 (x1, y1)*

end_point = (150, 150) *# 끝점 (x2, y2)*

color = (255, 0, 0)

thickness = 2

img_copy = img.copy()

파란색

선의 두께 지정

원본 이미지 복사

선 그리기

cv2.line(img_copy, start_point, end_point, color, thickness)

선의 오른쪽 끝에 텍스트를 추가

text_position = (end_point[0] + 10, end_point[1] + 20)

font = cv2.FONT_HERSHEY_SIMPLEX *# 폰트 지정*

text_color = (0, 0, 255)

font_scale = 1.5

텍스트 색상: 빨간색

텍스트 크기



텍스트 그리기

- 이미지의 좌상단이 (0, 0)이기 때문에 text_position의 (x, y) 값에서
- y값을 키울수록 텍스트가 아래에 위치하게 된다

텍스트 그리기

```
cv2.putText(img_copy, 'mountain', text_position, font, font_scale, text_color, thickness)
```

```
cv2.imshow(img_copy)
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

이미지 표시 및 창 닫기

폰트 종류>

FONT_ITALIC

FONT_HERSHEY

FONT_HERSHEY_COMPLEX

FONT_HERSHEY_COMPLEX_SMALL

FONT_HERSHEY_DUPLEX

FONT_HERSHEY_SCRIPT_COMPLEX

FONT_HERSHEY_SCRIPT_SIMPLEX

FONT_HERSHEY_SIMPLEX

FONT_HERSHEY_TRIPLEX



한글 텍스트 그리기

한글 텍스트 그리기

- cv2.putText()는 한글 지원을 하지 않기 때문에 OpenCV로 한글 출력을 하려면
- 다음의 과정을 거친다

- ① 한글 폰트 설치
- ② 원본 넘파이 배열 이미지를 PIL 객체로 변환
- ③ 텍스트 출력
- ④ PIL 객체를 다시 넘파이 배열로 변환
- ⑤ 이미지 그리기

한글 폰트 설치

- 먼저 그래프에서 한글 폰트를 설치한다
- `!sudo apt-get install -y fonts-nanum`
- `!sudo fc-cache -fv`
- `!rm ~/.cache/matplotlib -rf`
- 한글의 올바른 인식을 위해 런타임 > 세션 다시 시작

※ NanumGothic과 NanumMyeongjo 중 하나를 선택한다

※ 기본 글꼴을 변경하였으므로 "런타임 > 세션 다시 시작" 수행 후 여기부터 다시 진행한다

이미지 로딩 및 변환

- 세션이 다시 시작되었으므로 이미지를 다시 불러들인다
- 이어서 넘파이 배열 이미지를 PIL 이미지 객체로 변환한다

```
import numpy as np
```

```
import cv2
```

```
from google.colab.patches import cv2_imshow
```

```
img = cv2.imread('/content/gdrive/MyDrive/pytest_img/opencv/mountain.jpg')
```

```
from PIL import Image, ImageDraw, ImageFont
```

```
img_copy = img.copy()
```

```
img_pil = Image.fromarray(img_copy)
```

```
draw = ImageDraw.Draw(img_pil)
```

원본 이미지 복사

넘파이 배열에서 PIL 이미지 객체로 변환

주어진 이미지에 그리기 작업을 할 수 있는 PIL 타입의 드로잉 객체 생성

폰트 설정

- 한글 폰트를 설정한다

NanumMyeongjo 또는 NanumGothic 사용

```
font_path = '/usr/share/fonts/truetype/nanum/NanumGothic.ttf'
```

```
font_size = 30
```

```
font = ImageFont.truetype(font_path, font_size)
```

한글 텍스트와 위치 지정

```
text = '안녕하세요!'
```

```
text_position = (50, 80)
```

```
text_color = (255, 0, 0)
```

파란색



텍스트와 이미지 출력

- PIL 이미지를 다시 넘파이 배열로 변환한 뒤, 이미지를 출력한다

```
# 드로잉 객체를 이용하여 이미지에 텍스트 그리기  
draw.text(text_position, text, font=font, fill=text_color)
```

```
# PIL 이미지를 다시 넘파이 배열로 변환  
img_np_final = np.array(img_pil)
```

```
# 이미지 표시 및 창 닫기  
cv2.imshow(img_np_final)  
cv2.waitKey(0)  
cv2.destroyAllWindows()
```



마스킹

마스킹

- 마스킹이란 이미지의 특정 영역을 선택적으로 처리하여
- 특정 영역을 강조하거나, 또는 무시하는 작업을 말한다
- 마스킹을 위해 먼저 이미지의 높이와 너비 정보를 확인한다



```
img_resize = cv2.resize(img, dsize=(600, 400))
```

```
print("높이:", img_resize.shape[0])  
print("너비:", img_resize.shape[1])  
print("채널:", img_resize.shape[2])
```

```
cv2.imshow(img_resize)
```

```
# 이미지의 높이와 너비를 가져온다  
(height, width) = img_resize.shape[:2]
```

마스킹

- 검은색 마스크를 만들고, 안쪽에 흰색을 채운다

이미지의 중심을 구한다

```
center = (width // 2, height // 2)
```

이미지의 높이 및 너비와 동일한 검정색 mask를 만든다

```
mask = np.zeros(img_resize.shape[:2], dtype="uint8")
```

mask에 filter시킬 모양으로 흰색 원을 그린다

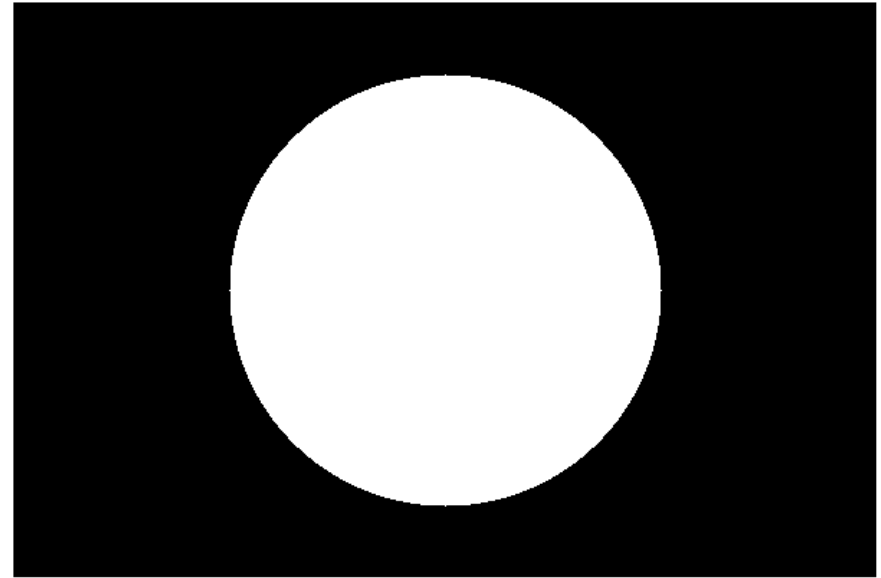
thickness 값을 음수를 주면 안쪽이 채워지게 된다

```
print('----- Mask -----')
```

```
cv2.circle(img=mask, center=center, radius=150, color=(255, 255, 255), thickness= -1)
```

```
cv2.imshow(mask)
```

----- Mask -----



마스킹

- 이미지에 마스크를 적용한다

이미지에 mask를 적용한다

```
print('----- Masked Image -----')
```

```
masked = cv2.bitwise_and(src1=img_resize, src2=img_resize, mask=mask)
```

```
cv2_imshow(masked)
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```



cv2.bitwise_and() 함수는 src1과 src2에 동일한 사이즈의 이미지를 넣으면 이미지를 겹쳐서 그릴 수 있게 한다
여기서는 mask를 적용시키기 위하여 이 함수를 사용하면서 필수 파라미터인 src1와 src2에 동일한 값을 주었다

OpenCV와 PIL

- OpenCV로 이미지를 읽은 결과는 Numpy 배열로서,
 - 파이썬 이미지 분석에서 기본적으로 사용되는 PIL 타입과는 다르다
 - 또한, OpenCV의 결과는 RGB가 아니라, BGR의 순서로 읽는 점도 다르다
-
- PIL 타입 이미지는 메타데이터를 가질 수 있어 이미지의 정보를 파악할 수 있다
 - 이미지를 Numpy 배열로 읽으면 숫자 형태이므로 직접적인 픽셀 조작이 가능하다
-
- 그러나 어느 쪽으로 읽든 상호 변환이 가능하다

PIL 타입 메타데이터 확인

- `from keras.preprocessing.image import array_to_img, img_to_array, load_img`
- `img_pil = load_img('/content/gdrive/MyDrive/pytest_img/opencv/mountain.jpg')`
- `print(type(img_pil))`

기본 정보 출력

- `print("Format:", img_pil.format)`
- `print("Mode:", img_pil.mode)`
- `print("Size:", img_pil.size)`

```
<class 'PIL.JpegImagePlugin.JpegImageFile'>  
Format: JPEG  
Mode: RGB  
Size: (1024, 700)
```


OpenCV 이미지 타입 확인

- `import cv2`
- `img_cv =
cv2.imread('/content/gdrive/MyDrive/pytest_img/opencv/mountain.jpg')`
- `print(type(img_cv))` `# <class 'numpy.ndarray'>`

Numpy 배열을 PIL 이미지로 변환

- OpenCV의 numpy 배열을 PIL 이미지로 변환
- 먼저 OpenCV의 BGR 순서로 되어 있는 것을 RGB 순서로 바꾼 다음, PIL 타입으로 변환한다

- `from PIL import Image`

- `img_rgb = cv2.cvtColor(img_cv, cv2.COLOR_BGR2RGB)` # BGR -> RGB
- `img_pil = Image.fromarray(img_rgb)`

- `print(type(img_pil))`

기본 정보 출력

- `print("Format:", img_pil.format)`
- `print("Mode:", img_pil.mode)`
- `print("Size:", img_pil.size)`

```
<class 'PIL.Image.Image'>
```

```
Format: None
```

```
Mode: RGB
```

```
Size: (1024, 700)
```

numpy 배열에서 변환된 이미지로 Format 정보는 나오지 않는다

PIL 타입을 Numpy 배열로 변환하기

- 넘파이 배열로의 변환은 간단히 `np.array()`를 사용하여 가능하다
- 단, OpenCV의 BGR 순서로 변환하는 과정을 추가한다
- `import numpy as np`
- `img_cv = np.array(img_pil)`
- `img_cv_bgr = cv2.cvtColor(img_cv, cv2.COLOR_RGB2BGR)`
- `print(type(img_cv_bgr))` # <class 'numpy.ndarray'>