Write a program that reads an image, converts its color space from BGR to RGB, and transforms it between NumPy array and PIL Image object representations, so that the same can be used for filtering, enhancement, or analysis

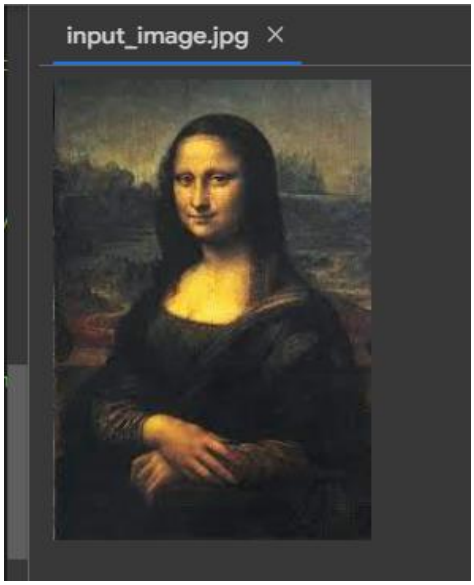
Code

Write a program that reads an image, converts its color space from BGR to RGB, and transforms it between NumPy array and PIL Image object representations, so that the same can be used for filtering, enhancement, or analysis

```python
import cv2
from PIL import Image
import numpy as np

def bgr_to_rgb(image_bgr):
    # Convert BGR image to RGB using numpy
    image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)
    return image_rgb

def rgb_to_bgr(image_rgb):
    # Convert RGB image to BGR using numpy
    image_bgr = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2BGR)
    return image_bgr

def numpy_array_to_pil(image_np):
    # Convert NumPy array to PIL Image object
    pil_image = Image.fromarray(image_np)
    return pil_image

def pil_to_numpy_array(pil_image):
    # Convert PIL Image object to NumPy array
    image_np = np.array(pil_image)
    return image_np

if __name__ == "__main__":
    # Replace 'input_image.jpg' with the path to your input image
    input_image_path = 'input_image.jpg'

    # Read the image using OpenCV
    image_bgr = cv2.imread(input_image_path)

    # Convert BGR image to RGB
    image_rgb = bgr_to_rgb(image_bgr)

    # Do filtering, enhancement, or analysis on the RGB image here if needed

    # Convert RGB image back to BGR
    image_bgr_transformed = rgb_to_bgr(image_rgb)

    # Convert BGR image to PIL Image object
    pil_image = numpy_array_to_pil(image_bgr_transformed)

    # Save the transformed image (optional)
    output_image_path = 'output_image.jpg'
    pil_image.save(output_image_path)

    # Convert PIL Image object back to NumPy array
    image_np_transformed = pil_to_numpy_array(pil_image)
```

Output

Before



After