ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN SỐ 2

IMAGE COMPRESSION

Môn học: **Toán ứng dụng và thống kê cho công nghệ thông tin.**

Mã môn học: **MTH00057.**

Giảng viên hướng dẫn: **Phan Thị Phương Uyên.**

Sinh viên thực hiện: **Nguyễn Đức Vĩnh Hòa.**

Mã số sinh viên: **21127609.**

Lớp: **21CLC05.**

MỤC LỤC ĐỒ ÁN IMAGE COMPRESSION

ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH:

STT	Tên chức năng	Mức độ hoàn thành
1	Làm sáng ảnh	100%
2	Tăng độ tương phản	100%
3	Lật ảnh	100%
4	Chuyển thành ảnh xám	100%
5	Chuyển thành ảnh sepia	100%
6	Làm sắc nét ảnh	100%
7	Làm mờ ảnh	100%
8	Cắt ảnh khung tròn ở trung tâm	100%
9	Cắt ảnh 2 khung elip chéo nhau	0%

GIỚI THIỆU VỀ ĐỒ ÁN:

Ý tưởng xây dựng chương trình:

Ảnh được lưu trữ dưới dạng ma trận các điểm ảnh. Mỗi điểm ảnh có thể là một giá trị (ảnh xám) hoặc một vector (ảnh màu). Từ đó ta có thể áp dụng các phép toán trên ma trận để chỉnh sửa và thay đổi chi tiết ảnh như tăng/giảm độ sáng ảnh, tăng/giảm độ tương phản,chuyển thành ảnh xám, làm mờ/sắc nét ảnh,…

Một số hàm chức năng sẽ ép mảng hình ảnh dạng uint8 về dạng số nguyên hoặc số thực trong quá trình cài đặt. Lý do là do kiểu unit8 chỉ có khả năng hoạt động trong khoảng 0-255, do đó sẽ có trường hợp khi tính toán vượt quá miền giá trị. Do đó, trước khi tiến hành tính toán sẽ ép về dạng int hoặc float, sau đó xử lý các giá trị vượt quá miền giá trị này và trả lại kiểu uint8.

CHI TIẾT CHƯƠNG TRÌNH:

Các thư viện được sử dụng:

- numpy
- PIL.Image
- matplotlib.pyplot
- math

Các hàm trong chương trình:

1/ brightenImage (image, scale).

- Chức năng: Hàm thay đổi độ sáng hình ảnh.
- Ý tưởng: Cộng các giá trị màu R,B,G với một giá trị được xác định bởi tỉ lệ scale nhập vào. [1]
- Tham số đầu vào: mảng vector pixels của ảnh có kiểu np.array, giá trị scale (mặc định là 50%).
- Kết quả trả về: mảng vector pixels của ảnh có kiểu np.array đã được thay đổi độ sáng.
- Mô tả:
  → Cộng mảng vector pixels của ảnh với một số nguyên theo tỉ lệ scale.
  → Đưa tất cả các giá trị màu bị vượt quá 255 về lại 255 (nếu có).

2/ contrastImage (image, scale).

- Chức năng: Hàm thay đổi độ tương phản hình ảnh.
- Ý tưởng: Đổi miền giá trị màu từ 0->255 về -0.5->0.5, sau đó nhân với một giá trị xác định. [1]
- Tham số đầu vào: mảng vector pixels của ảnh có kiểu np.array, giá trị scale (mặc định là 50%),

- Kết quả trả về: mảng vector pixels của ảnh có kiểu np.array đã làm thay đổi độ tương phản.
- Mô tả:
  → Miền giá trị màu được thay đổi từ 0->255 về -0.5->0.5.
  → Cộng với một số nguyên dựa trên tỷ lệ scale.
  → Đổi lại miền giá trị màu ban đầu.
  → Đưa tất cả các giá trị màu vượt quá 0 và 255 (thấp hơn 0 hoặc lớn hơn 255) về lại 0 và 255.

3/ flipImage (image, dim, mode).

- Chức năng: Hàm lật hình ảnh.
- Ý tưởng: Lật ma trận bằng cách đưa không gian mảng hình về lại ban đầu và sử dụng hàm np.flip của numpy. [1] [2]
- Tham số đầu vào mảng vector pixels của ảnh có kiểu np.array, giá trị không gian gốc dim và chế độ lật (ngang hoặc dọc) với chế độ 0 là lật dọc và chế độ 1 là lật ngang.
- Kết quả trả về: Mảng vector pixels của hình ảnh có kiểu np.array đã được lật ngang/dọc.
- Mô tả:
  → Sử dụng reshape để trả lại không gian mảng hình ban đầu.
  → Ma trận có trục tương ứng với giá trị mode (0 là ngang, 1 là dọc) sẽ được lật bằng hàm np.flip của numpy.
  → Đưa không gian mảng hình về mảng các vector pixels.

4/ greyImage (image, alpha, beta, gamma).

- Chức năng: Hàm chuyển ảnh màu thành ảnh xám.
- Ý tưởng: Chuyển ba giá trị màu R, B, G về 1 giá trị. [1] [3] .
- Tham số đầu vào: mảng vector pixels của ảnh có kiểu np.array, tham số alpha, beta, gamma tương ứng với độ lớn phần giá trị của R, B, G với alpha

+ beta + gamma = 1. Với giá trị mặc định [alpha, beta, gamma] = [0.299, 0.587, 0.114].

- Kết quả trả về: mảng vector pixels của ảnh có kiểu np.array đã được chuyển thành màu xám.
- Mô tả:
  → Chuyển ba giá trị R, B và G về một giá trị bằng cách sử dụng công thức nhân tuyến tính sau: *Grey = (alpha*R + beta*B + gamma*G)* [1]
  → Thực hiện bằng cách nhân ma trận chứa các vector pixel với vector có giá trị [alpha, beta, gamma].
  → Các vector pixel sẽ có đúng một giá trị sau khi nhân với vector. Vì vậy để hàm plt.imshow của pyplot có thể đọc được hình cần nhân bản nó thành ba giá trị để tạo thành các vector pixels.
  → Thực hiện bằng cách nhân với vector dòng có ba giá trị, sau đó chuyển vị ma trận kết quả.

5/ sepiaImage(image)

- Chức năng: Hàm chuyển ảnh màu thành ảnh sepia.
- Ý tưởng: Chuyển ba giá trị màu R, B, G về ba giá trị R, B, G mang được hiệu ứng sepia.
- Tham số đầu vào: mảng vector pixels của ảnh có kiểu np.array.
- Kết quả trả về: mảng vector pixels của ảnh có kiểu np.array đã được chuyển thành ảnh sepia.
- Mô tả:
  → Sử dụng ma trận như sau:
  → R' = 0.393 * R + 0.769 * G + 0.189 * B
  → G' = 0.349 * R + 0.686 * G + 0.168 * B
  → B' = 0.272 * R + 0.534 * G + 0.131 * B
  → Từ đó có ma trận sepia_matrix:

[0.393, 0.769, 0.189]

[0.349, 0.686, 0.168]

[0.272, 0.534, 0.131]

→ Áp dụng phép biến đổi sepia cho từng pixel trong ảnh ban đầu bằng cách nhân ma trận sepia_matrix với mỗi vector pixel trong ảnh. Chúng ta sử dụng phép nhân ma trận của NumPy bằng cách sử dụng hàm np.dot.

6, 7/ blurImage(image, dim, kernel) và sharpenImage(image, dim, kernel)

- Chức năng: blurImage là hàm làm mờ hình ảnh, sharpenImage là hàm làm sắc nét hình ảnh.
- Ý tưởng Các hàm làm mờ và làm sắc nét sử dụng phép tích chập (convolution) với một ma trận kernel để làm mờ hoặc làm sắc nét hình ảnh. Làm sắc nét và làm mờ khác nhau về giá trị của các phần tử trong ma trận kernel. [5] [6] [7]
- Numpy có cung cấp hàm thực hiện tích chập: np.convolve . Tuy nhiên chỉ thao tác được trên mảng và kernel 1 chiều. Vì vậy, thay vì duyệt từng giá trị và thực hiện phép chập ở từng vị trí trong mảng hình, ta dịch ma trận theo dòng và cột sau đó nhân với từng giá trị của kernel, sau đó lấy tổng các ma trận vừa tính, ta sẽ được ma trận kết quả sau khi thực hiện phép tích chập.
- Tham số đầu vào: mảng vector pixels của ảnh có kiểu np.array, giá trị không gian gốc dim, ma trận vuông kernel.
- Kết quả trả về: mảng vector pixels của ảnh có kiểu np.array đã được làm mờ hoặc sắc nét.
- Mô tả: Mã nguồn tham khảo được tìm thấy trên Stackoverflow [5]
  → Sử dụng reshape để đưa không gian mảng hình về lại ban đầu.
  → Khởi tạo mảng hình ảnh kết quả image_result.

  image_result = np.zeros(image.shape)

  → Phép tích chập của ma trận hình với kernel được thực hiện như sau:
    - Duyệt từng phần tử trong kernel theo dòng và cột.
    - Sử dụng hàm np.roll của numpy để dịch chuyển dòng hoặc cột tương ứng theo phần tử hiện tại của kernel.
    - Nhân giá trị hiện tại của kernel với mảng đã được dịch chuy.
    - Cộng dồn của mảng tích kết quả vào image_result.

    - Sau khi duyệt qua tất cả các phần tử của kernel, image_result chứa các kết quả của phép tích chập tại vị trí tương ứng.

→ Đưa không gian mảng hình kết quả về dạng mảng các vector pixels.

8/ circleFrame(image, dim, mode).

- Chức năng: Cắt ảnh khung tròn ở trung tâm
- Ý tưởng: Tạo một ma trận hình tròn có tâm ở giữa khung hình, sau đó phủ nó lên ảnh gốc. [8] [9]
- Tham số đầu vào: mảng vector pixels của ảnh có kiểu np.array, giá trị không gian gốc dim, chế độ khung mode với mode = 0 đường kính đường tròn bằng chiều dài không gian ngắn nhất của khung hình, mode = 1 đường kính đường tròn bằng chiều dài không gian dài nhất của khung hình.
- Kết quả trả về: mảng vector pixels của ảnh có kiểu np.array đã được cắt khung tròn.
- Mô tả:
  → Khởi tạo tọa độ tâm của đường tròn trùng với vị trí trung tâm của hình và bán kính khung tròn phụ thuộc vào giá trị mode.
  → Sử dụng hàm np.ogrid của numpy để tạo giá trị tọa độ x, y tương ứng với chiều dài và rộng của khung hình.
  → Sau đó tạo mảng mask_circle với kiểu boolean chứa kết quả chân trị của từng cặp x, y trong biểu thức bất phương trình nằm ngoài hình tròn.
  → Xóa giá trị màu tại các vị trí mà kết quả chân trị của mask_circle là True bằng cách gán với vector [0, 0, 0].

9/ saveImage(opt, imName, image)

- Chức năng: lưu lại hình ảnh đã xử lí có tên tương ứng với loại chức năng mà ảnh đã được xử lí.
- Tham số đầu vào: số thứ tự chức năng, tên của hình ảnh, mảng vector pixels của ảnh có kiểu np.array.
- Kết quả trả về: Hàm không trả về giá trị (None). Thay vào đó, nó lưu ảnh đã chỉnh sửa vào một tệp mới với tên tùy chỉnh tùy thuộc vào tùy chọn chỉnh sửa đã chọn.
- Mô tả:

→ Dùng if elif để xác định tùy chọn chỉnh sửa đã chọn.
→ Sau đó, sử dụng phép cắt chuỗi ([:-4] và [-4:]) để loại bỏ đuôi tệp và thêm phần tên tùy chỉnh.
→ Kết hợp các phần đó lại để tạo tên tệp mới và lưu ảnh đã chỉnh sửa vào tệp mới đó.

10/ showImage(optIndex, converted_image, Options)

- Chức năng: hiển thị hình ảnh đã được chỉnh sửa ra màn hình
- Tham số đầu vào: số thứ tự chức năng, mảng vector pixels của ảnh có kiểu np.array đã được chỉnh sửa theo thứ tự trên, mảng chứa tên các chức năng.

11/ generateImage(optIndex, image)

- Chức năng: thực hiện các loại chỉnh sửa ảnh khác nhau dựa trên giá trị của tham số 'optIndex'
- Tham số đầu vào: số thứ tự chức năng, hình ảnh cần chỉnh sửa.
- Kết quả trả về: mảng vector pixels của ảnh có kiểu np.array đã được chỉnh sửa theo thứ tự trên .
- Mô tả:

→ Ứng với mỗi 'optIndex', chạy các hàm chỉnh sửa ảnh tương ứng.
→ Với chức năng lật ảnh, thêm vào chọn lật ngang hoặc lật dọc.
→ Với chức năng làm sắc nét ảnh, khởi tạo ma trận kernel:

```
kernel = np.array([[0, -1, 0],
                   [-1, 5, -1],
                   [0, -1, 0]])
kernel = kernel / np.sum(kernel)
```

→ Với chức năng làm mờ ảnh, khởi tạo ma trận kernel:

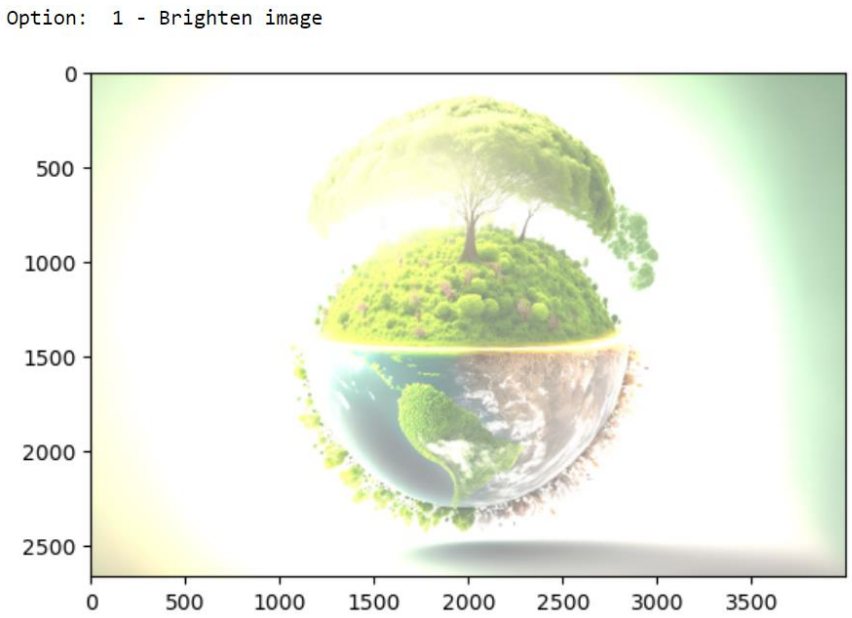
```
kernel = np.ones((3, 3))
kernel = kernel / np.sum(kernel)
```

→ Sau khi hoàn thành chỉnh sửa, hàm tiến hành chuyển đổi mảng ảnh kết quả từ dạng mảng một chiều thành dạng mảng hai chiều với kích thước tương tự như ảnh gốc

CHẠY CHƯƠNG TRÌNH, THỬ NGHIỆM VÀ KẾT QUẢ

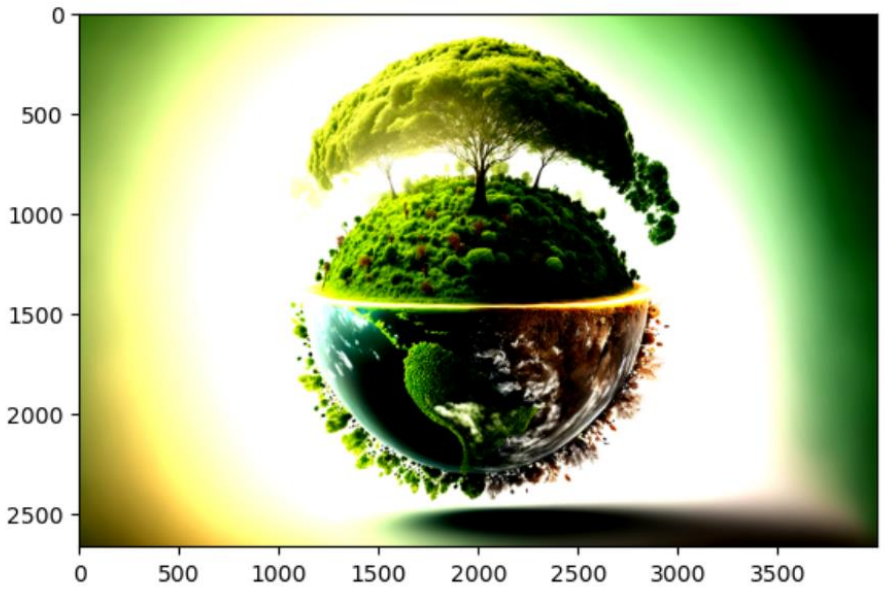
Ảnh gốc:

Chức năng tăng độ sáng hình ảnh (Brighten):



Chức năng tăng độ tương phản hình ảnh (Contrast):

```
Option:  2 - Increase contrast image
```

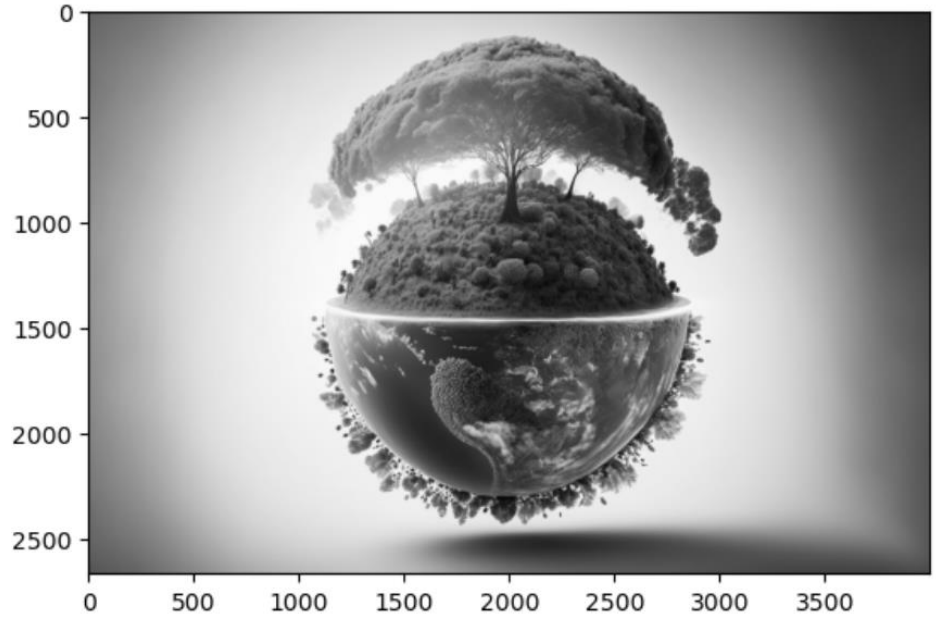


Chức năng lật ảnh (Flip):

```
Option:  3 - Flip image
```

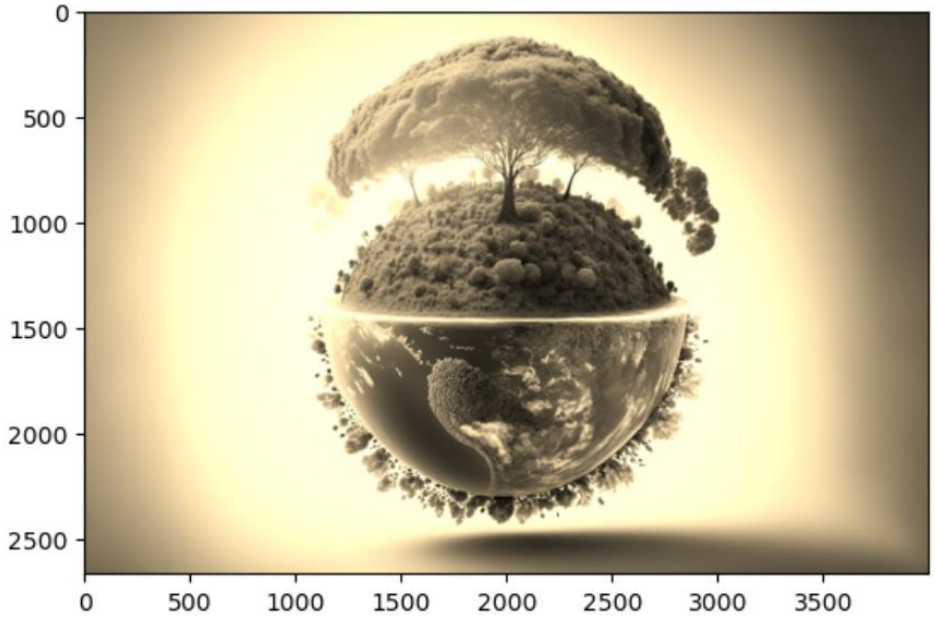
Chức năng chuyển ảnh thành ảnh xám (Grey):

```
Option:  4 - Grey image
```



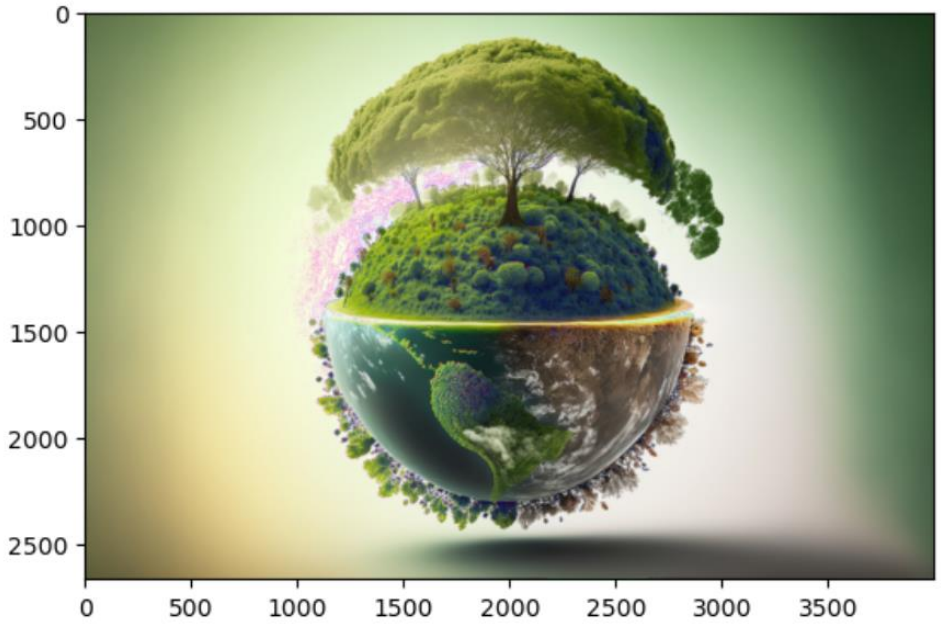
Chức năng chuyển ảnh thành ảnh sepia:

```
Option:  5 - Sepia image
```



Chức năng làm sắc nét ảnh (Sharpen):

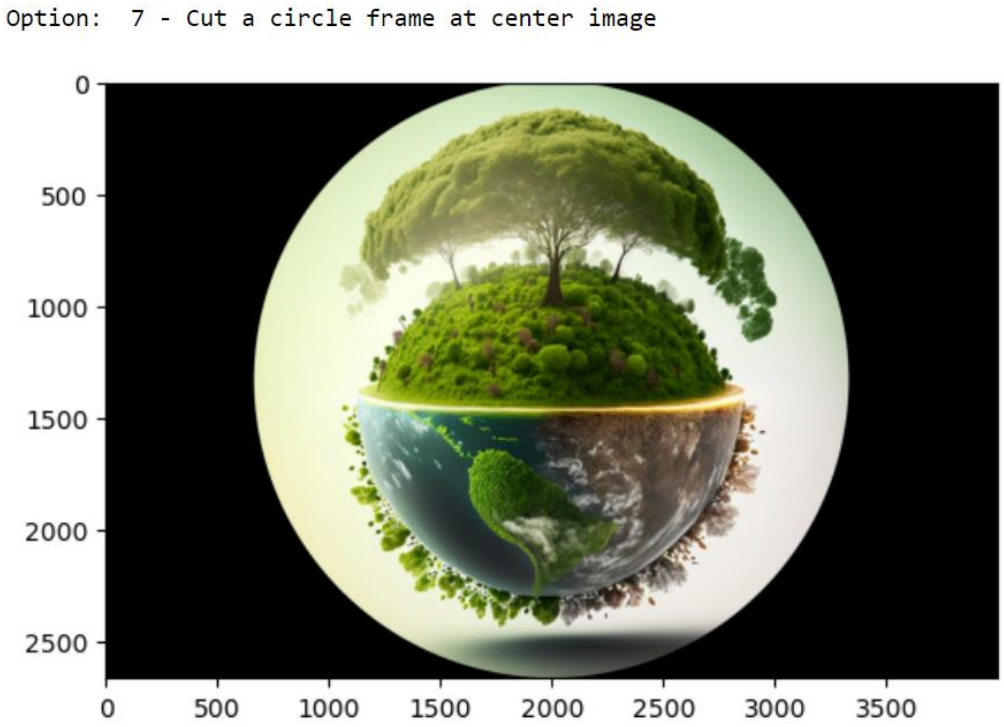
```
Option:  5 - Sharpen image
```



Chức năng làm mờ ảnh (Blur):

```
Option:  6 - Blur image
```

Chức năng khung tròn ở trung tâm ảnh (Circle frame):



TÀI LIỆU THAM KHẢO:

[1] https://www.codeproject.com/Articles/33838/Image-Processing-using-C

[2] https://numpy.org/doc/stable/reference/generated/numpy.flip.html

[3] https://stackoverflow.com/questions/1550130/cloning-row-or-column-vectors

[4] https://stackoverflow.com/questions/36434905/processing-an-image-to-sepia-tone-in-python

[5] https://stackoverflow.com/questions/29920114/how-to-gauss-filter-blur-a-floating-point-numpy-array

[6] https://numpy.org/doc/stable/reference/generated/numpy.convolve.html

[7] https://numpy.org/doc/stable/reference/generated/numpy.roll.html

[8] https://stackoverflow.com/questions/44865023/how-can-i-create-a-circular-mask-for-a-numpy-array

[9] https://towardsdatascience.com/the-little-known-ogrid-function-in-numpy-19ead3bdae40#:~:text=What%20is%20ogrid%3F,their%20row%20and%20column%20index

Ngoài ra cũng có sự tham khảo từ ChatGPT.