



BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP HỒ CHÍ MINH

Khoa: Công Nghệ Thông Tin



LAB REPORT 04

Student's ID :
Student's name : Hồ Phúc Lâm
Subject : PTHTDPT
Instructor : Nguyễn Thành Thái
Faculty : Công Nghệ Thông Tin
Completed Date : 11/09/2024

This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the entire width of the page, providing a guide for handwriting practice. There are no margins, text, or other markings on the paper.

LAB 3 IMAGE

1) Mục đích yêu cầu :

- +Củng cố kiến thức cơ bản về IMAGE, các khái niệm về hệ màu, độ phân giải,...
- +Tiếp cận thư viện xử lý image như PIL, opencv,... các API hỗ trợ,...

2) Tài liệu tham khảo (mở link sau xem hướng dẫn)

<https://www.pythonforengineers.com/image-and-video-processing-in-python/>

3) Thực hiện:

- Tìm hiểu thuộc tính Blurring và grayscale : trình bày chi tiết và chạy demo các hàm `imshow()`; `imread()`; `cvtColor()`; `GaussianBlur()`
- Tìm hiểu chức năng phát hiện cạnh đối tượng (edge detection): trình bày chi tiết và chạy demo hàm `Canny()`
- Tìm hiểu chức năng đếm đối tượng (count objects): trình bày chi tiết hàm `findContours()`; `drawContours()`
- Viết một chương trình hoàn chỉnh bao gồm các chức năng ở trên, các chức năng trong chương trình viết dạng thủ tục, được gọi khi cần.

4) Công cụ hỗ trợ : Python programming language

a) Spyder IDE hoặc PYTHON commandline

b) Cài đặt các thư viện hỗ trợ :

Câu lệnh cài đặt : `pip install <gói cài đặt>`

Vd : `pip install opencv-python`

`pip install`

`pip install matplotlib`

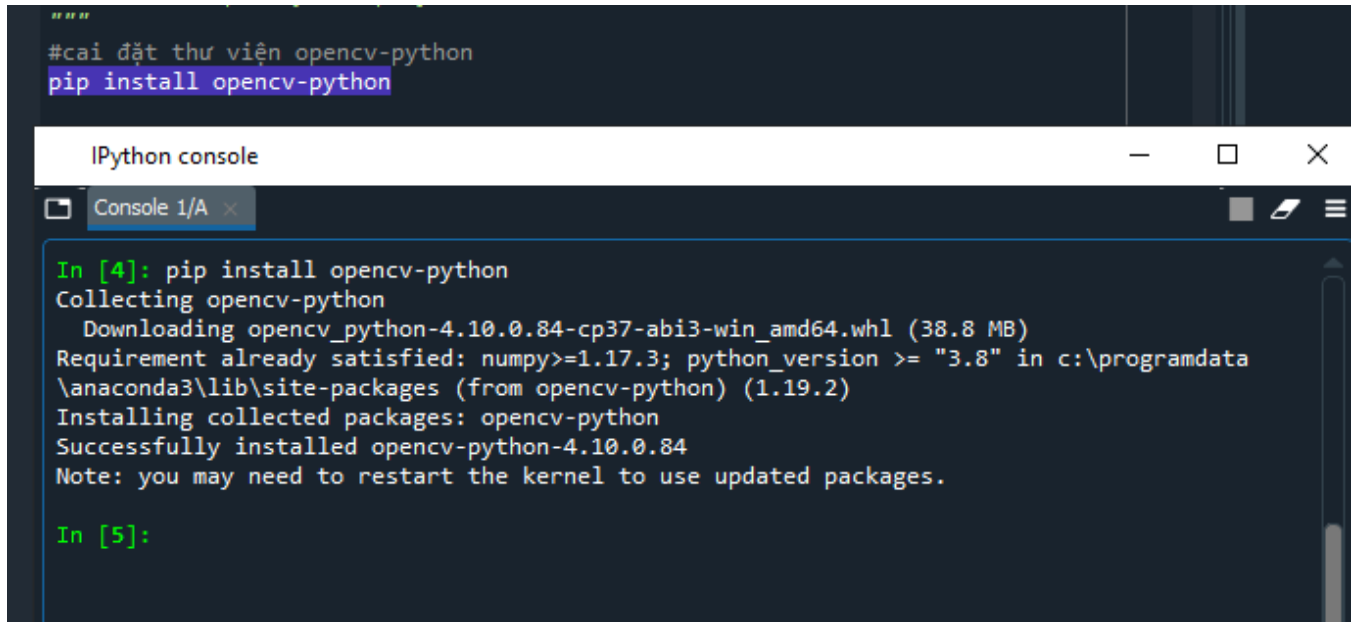
- Cài đặt thư viện (nếu chưa có): mở CMD trên Windows, gõ `pip install simpleaudio`, cài xong chạy python -> gõ lệnh `import simpleaudio as sa` kiểm tra lỗi...
- Vào trang <https://file-examples.com/index.php/sample-images-download/sample-jpg-download/> download file mẫu *.jpg và lưu trong thư mục
- Mở notepad viết chương trình *.py lưu trong một thư mục (thường là chung với thư mục của file Wav)
- Chạy thử code : `>python baitap1.py`

- Lưu các bài tập trong thư mục, nén và nộp (cuối giờ thực hành)

THỰC HIỆN

Tìm hiểu thuộc tính Blurring và grayscale : trình bày chi tiết và chạy demo các hàm `imshow()`; `imread()`; `cvtColor()`; `GaussianBlur()`

- Cài đặt thư viện **opencv-python**



```
#cài đặt thư viện opencv-python
pip install opencv-python

IPython console
Console 1/A x

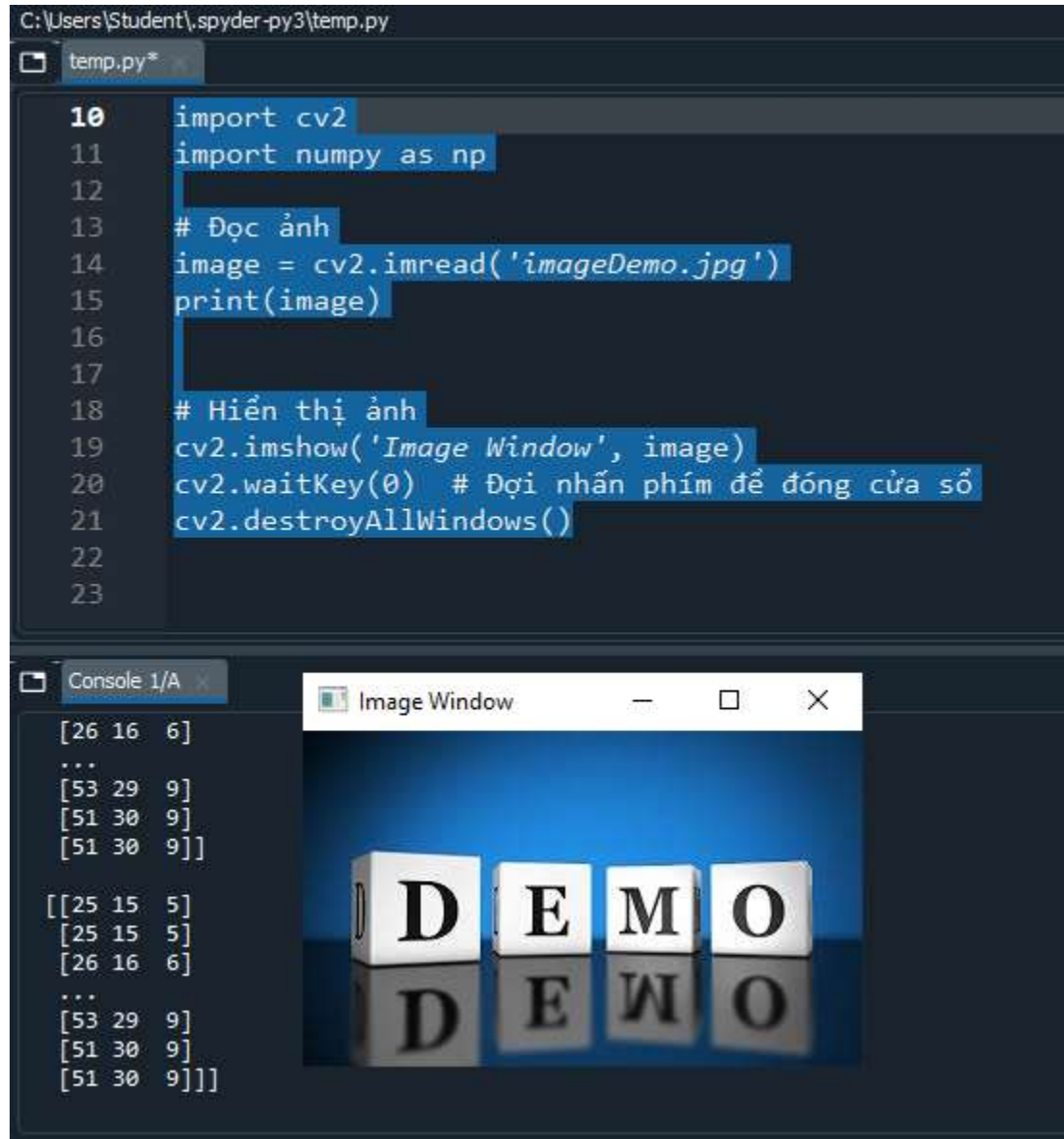
In [4]: pip install opencv-python
Collecting opencv-python
  Downloading opencv_python-4.10.0.84-cp37-abi3-win_amd64.whl (38.8 MB)
Requirement already satisfied: numpy>=1.17.3; python_version >= "3.8" in c:\programdata\anaconda3\lib\site-packages (from opencv-python) (1.19.2)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.10.0.84
Note: you may need to restart the kernel to use updated packages.

In [5]:
```

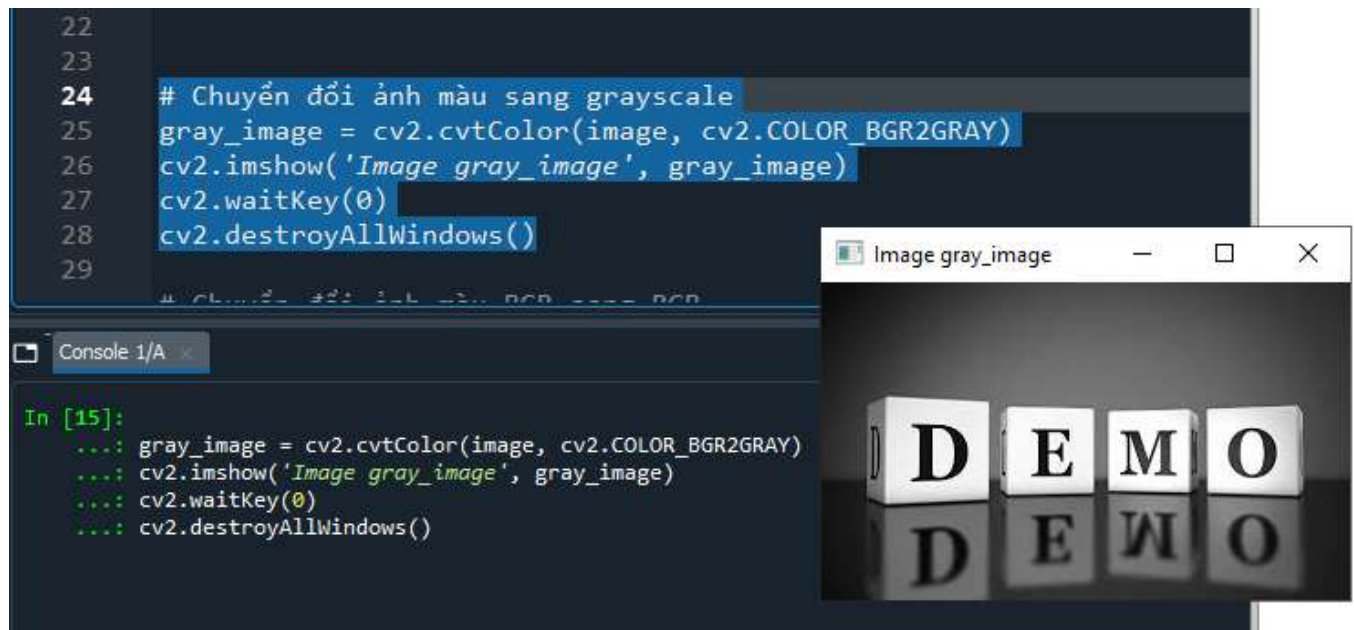
- Hàm **imread()**: Đọc một ảnh từ một file và trả về một đối tượng ảnh dưới dạng ma trận NumPy.
 - Cú pháp: **cv2.imread(filename, flags)**
 - **filename**: Đường dẫn đến file ảnh.
 - **flags**: Cờ để chỉ định cách đọc ảnh, ví dụ:
 - `cv2.IMREAD_COLOR` (hoặc 1): Đọc ảnh màu (mặc định).
 - `cv2.IMREAD_GRAYSCALE` (hoặc 0): Đọc ảnh ở chế độ grayscale.
 - `cv2.IMREAD_UNCHANGED` (hoặc -1): Đọc ảnh với tất cả các kênh (bao gồm alpha).


```
C:\Users\Student\spyder-py3\temp.py
temp.py* x
4
5     This is a temporary script file.
6     """
7     #cai đặt thư viện opencv-python
8     pip install opencv-python
9
10    import cv2
11    import numpy as np
12
13    # Đọc ảnh
14    image = cv2.imread('imageDemo.jpg')
15    print(image)
16
17
Console 1/A x
In [13]: import cv2
...: import numpy as np
...:
...: # Đọc ảnh
...: image = cv2.imread('imageDemo.jpg')
...: print(image)
[[[16 10  3]
  [16 10  3]
  [16 10  3]
  ...
  [79 46 13]
  [76 46 11]
  [76 46 11]]
  ...
  ...]]
```

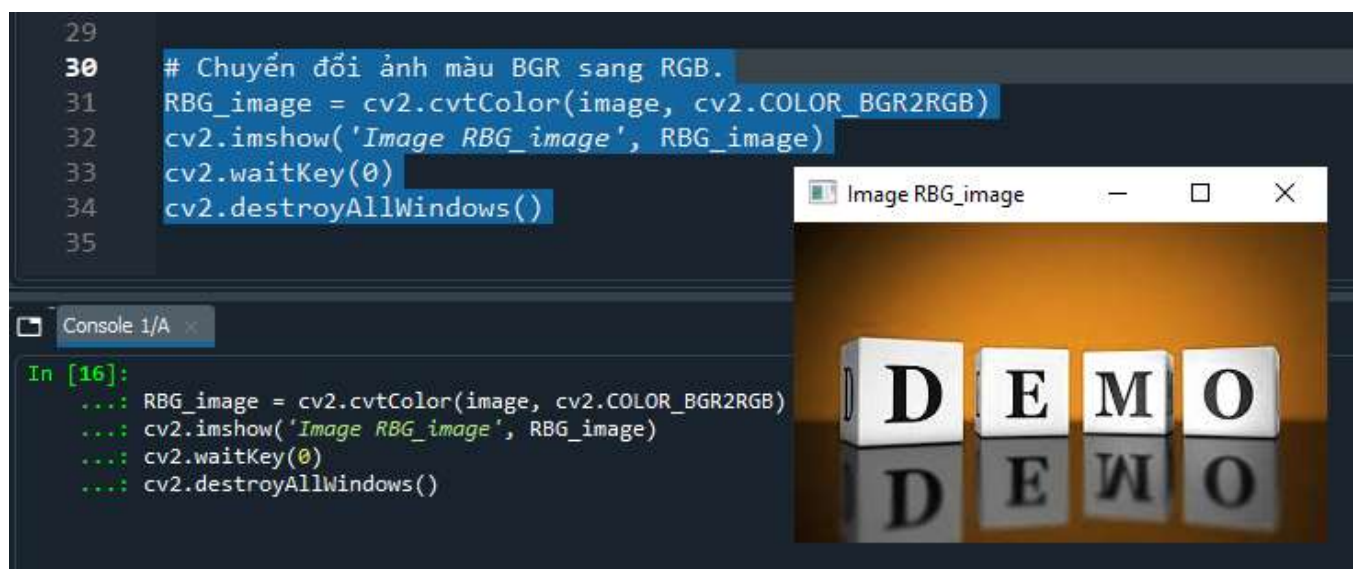

- Hàm **imshow()**: Hiển thị ảnh trong một cửa sổ GUI
 - Cú pháp: **cv2.imshow(window_name, image)**
 - **window_name**: Tên của cửa sổ để hiển thị ảnh.
 - **image**: Đối tượng ảnh (ma trận NumPy) cần hiển thị.



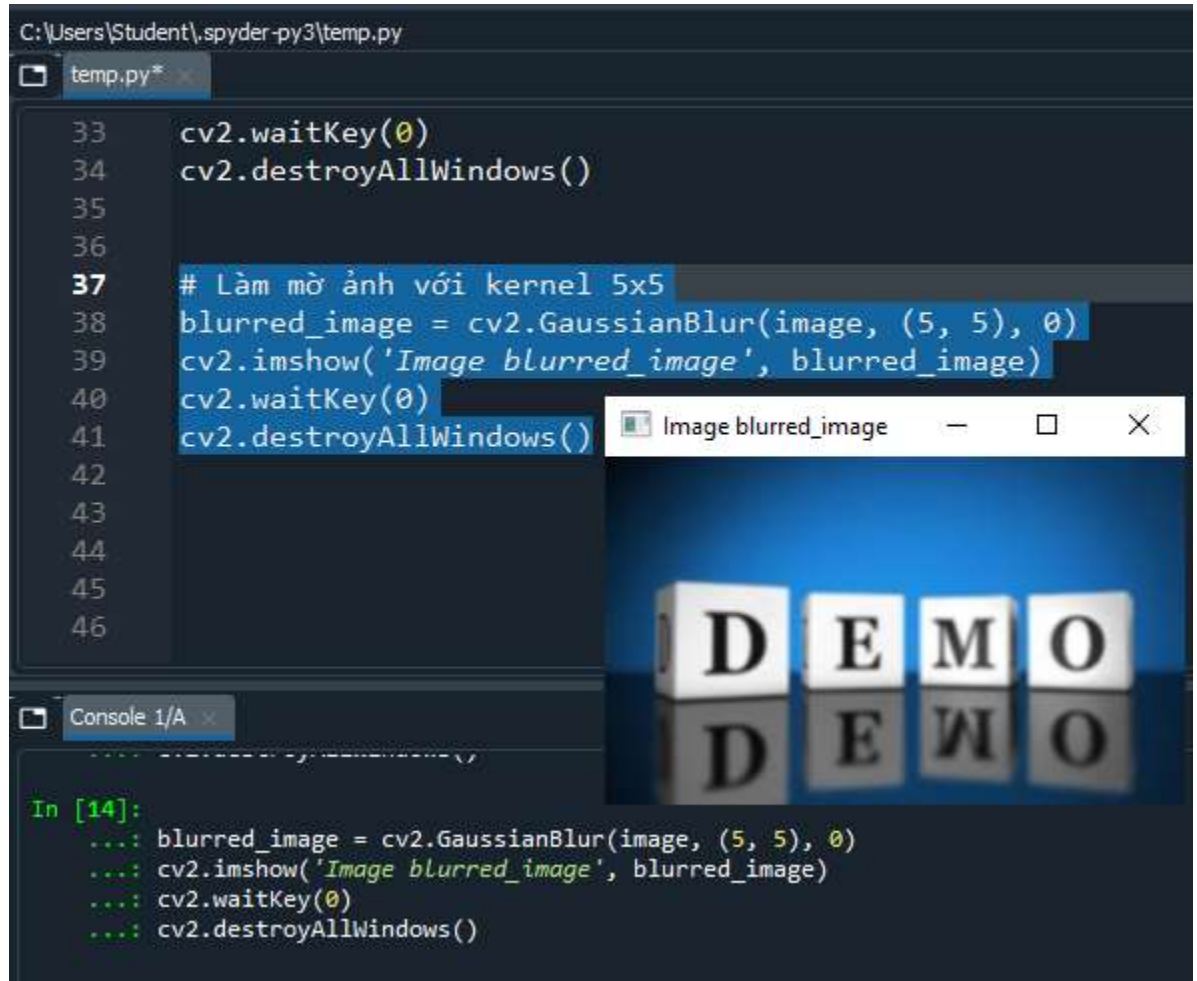
- Hàm **cvtColor()**: Chuyển đổi màu sắc của ảnh từ một không gian màu này sang không gian màu khác
 - o Cú pháp: **cv2.cvtColor(src, code)**
 - o **src**: Ảnh đầu vào.
 - o **code**: Mã chuyển đổi màu sắc, ví dụ:
 - o **cv2.COLOR_BGR2GRAY**: Chuyển đổi từ BGR (màu mặc định của OpenCV) sang grayscale



- o **cv2.COLOR_BGR2RGB**: Chuyển đổi từ BGR sang RGB.



- Hàm **GaussianBlur()**: Làm mờ ảnh bằng cách sử dụng bộ lọc Gaussian.
 - o Cú pháp: **cv2.GaussianBlur(src, ksize, sigmaX)**
 - o **src**: Ảnh đầu vào.
 - o **ksize**: Kích thước của kernel (bộ lọc), phải là số lẻ như (5, 5).
 - o **sigmaX**: Độ lệch chuẩn của Gaussian theo trục X.



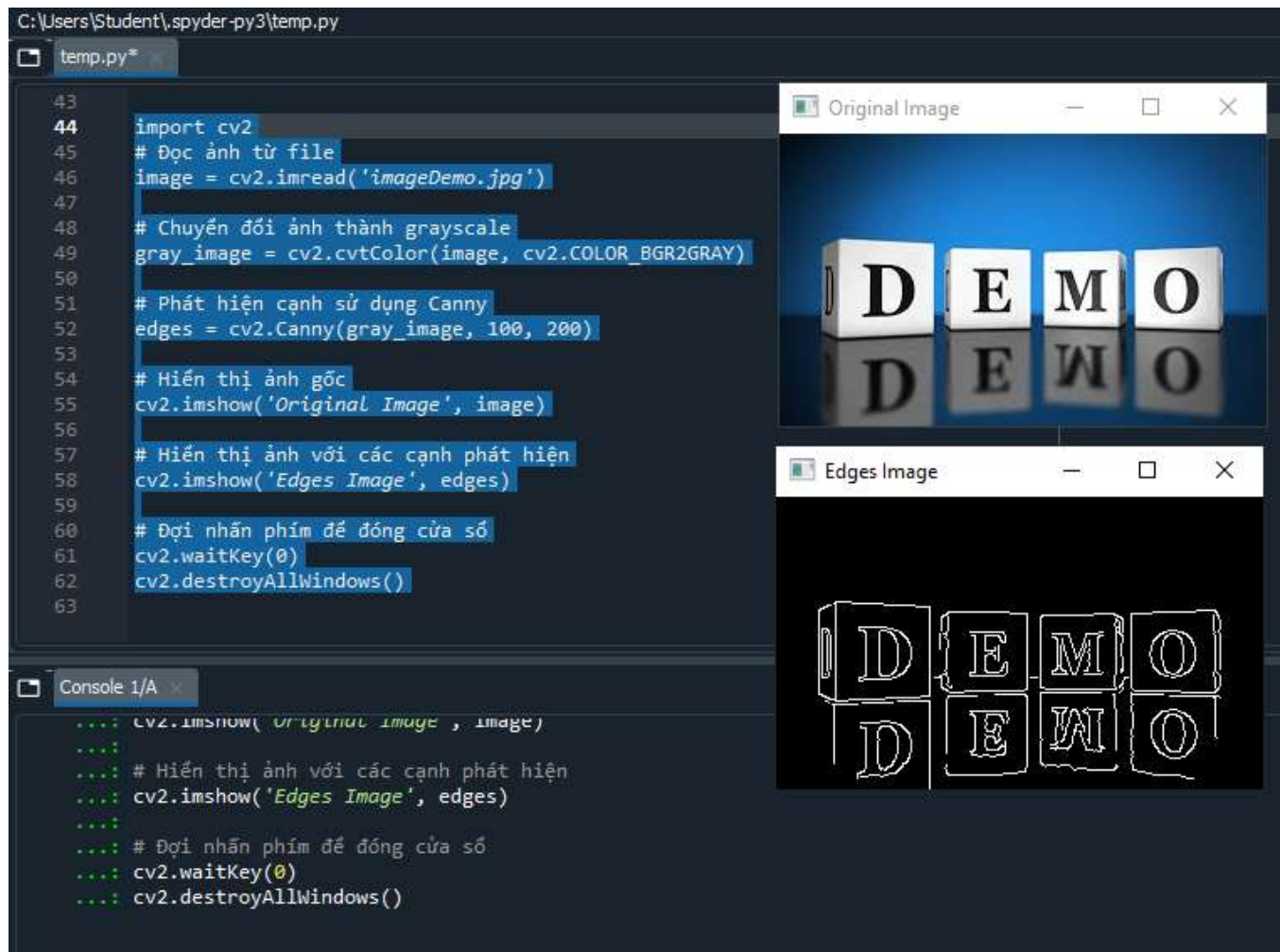
Tìm hiểu chức năng phát hiện cạnh đối tượng(edge detection): trình bày chi tiết và chạy demo hàm Canny()

Thuật toán Canny phát hiện cạnh trong **bốn bước chính**:

1. **Làm mịn ảnh**: Sử dụng bộ lọc Gaussian để làm mịn ảnh và loại bỏ nhiễu.
2. **Tính gradient**: Tính toán gradient của ảnh để xác định các vùng có thay đổi cường độ mạnh. Thường sử dụng bộ lọc Sobel để tính toán gradient theo hướng x và y.
3. **Non-maximum suppression**: Loại bỏ các điểm không phải là điểm cực trị cục bộ trong hướng gradient để giữ lại chỉ các điểm có gradient mạnh nhất.
4. **Thresholding và edge tracking**: Sử dụng hai ngưỡng để phân loại các cạnh. Một ngưỡng cao xác định các cạnh chính, và một ngưỡng thấp để kết nối các cạnh, đảm bảo rằng các đoạn ngắn không bị bỏ sót.

Hàm Canny(): Phát hiện các cạnh trong ảnh bằng cách sử dụng thuật toán Canny.

- **Cú pháp**: `cv2.Canny(image, threshold1, threshold2, apertureSize=3, L2gradient=False)`
- **image**: Ảnh đầu vào, nên là ảnh grayscale.
- **threshold1**: Ngưỡng thấp cho phương pháp double thresholding.
- **threshold2**: Ngưỡng cao cho phương pháp double thresholding.
- **apertureSize**: Kích thước của bộ lọc Sobel (mặc định là 3).
- **L2gradient**: Nếu là True, sử dụng L2 norm để tính toán độ lớn gradient; nếu không, sử dụng L1 norm.



Tìm hiểu chức năng đếm đối tượng (count objects): trình bày chi tiết hàm `findContours()`; `drawContours()`

1. **Hàm `findContours()`:** Phát hiện các đường viền (contours) trong ảnh nhị phân (binary image). Các đường viền thường được sử dụng để xác định hình dạng và kích thước của các đối tượng trong ảnh.

- **Cú pháp:**

`contours, hierarchy = cv2.findContours(image, mode, method, offset=None)`

- **image**: Ảnh đầu vào, phải là ảnh nhị phân (black and white). Bạn có thể sử dụng kỹ thuật thresholding hoặc các phương pháp khác để chuyển đổi ảnh màu hoặc grayscale thành ảnh nhị phân.
- **mode**: Cách phát hiện các đường viền, ví dụ:
 - `cv2.RETR_EXTERNAL`: Chỉ phát hiện các đường viền ngoài cùng.
 - `cv2.RETR_TREE`: Phát hiện tất cả các đường viền và xây dựng một cây phân cấp.
- **method**: Phương pháp để tìm các điểm đường viền, ví dụ:
 - `cv2.CHAIN_APPROX_SIMPLE`: Sử dụng một phương pháp nén để giảm số lượng điểm.
 - `cv2.CHAIN_APPROX_NONE`: Lưu tất cả các điểm đường viền.
- **offset**: Có thể bỏ qua, thường là (0, 0).
- **Trả về**:
 - **contours**: Danh sách các đường viền phát hiện được. Mỗi đường viền là một mảng các điểm (x, y).
 - **hierarchy**: Thông tin phân cấp về các đường viền (có thể bỏ qua nếu không cần).

```
59
60
61 import cv2
62 # Đọc ảnh và chuyển đổi thành grayscale
63 image = cv2.imread('imageDemo.jpg')
64 gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
65
66 # Chuyển đổi thành ảnh nhị phân
67 _, binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
68
69 # Phát hiện các đường viền
70 contours, _ = cv2.findContours(binary_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
71
72 print(contours)
73
```

```
Console 1/A x
...: print(contours)
(array([[233, 135]], dtype=int32), array([[194, 134]],
[[195, 134]],
[[196, 135]],
[[196, 134]], dtype=int32), array([[237, 131]], dtype=int32), array([[234, 131]], dtype=int32), array([[194, 129]],
[[193, 130]],
[[194, 131]],
[[195, 130]],
[[196, 131]]])
```


2. Hàm drawContours(): được sử dụng để vẽ các đường viền lên ảnh.

Cú pháp:

cv2.drawContours(image, contours, contourIdx, color, thickness)

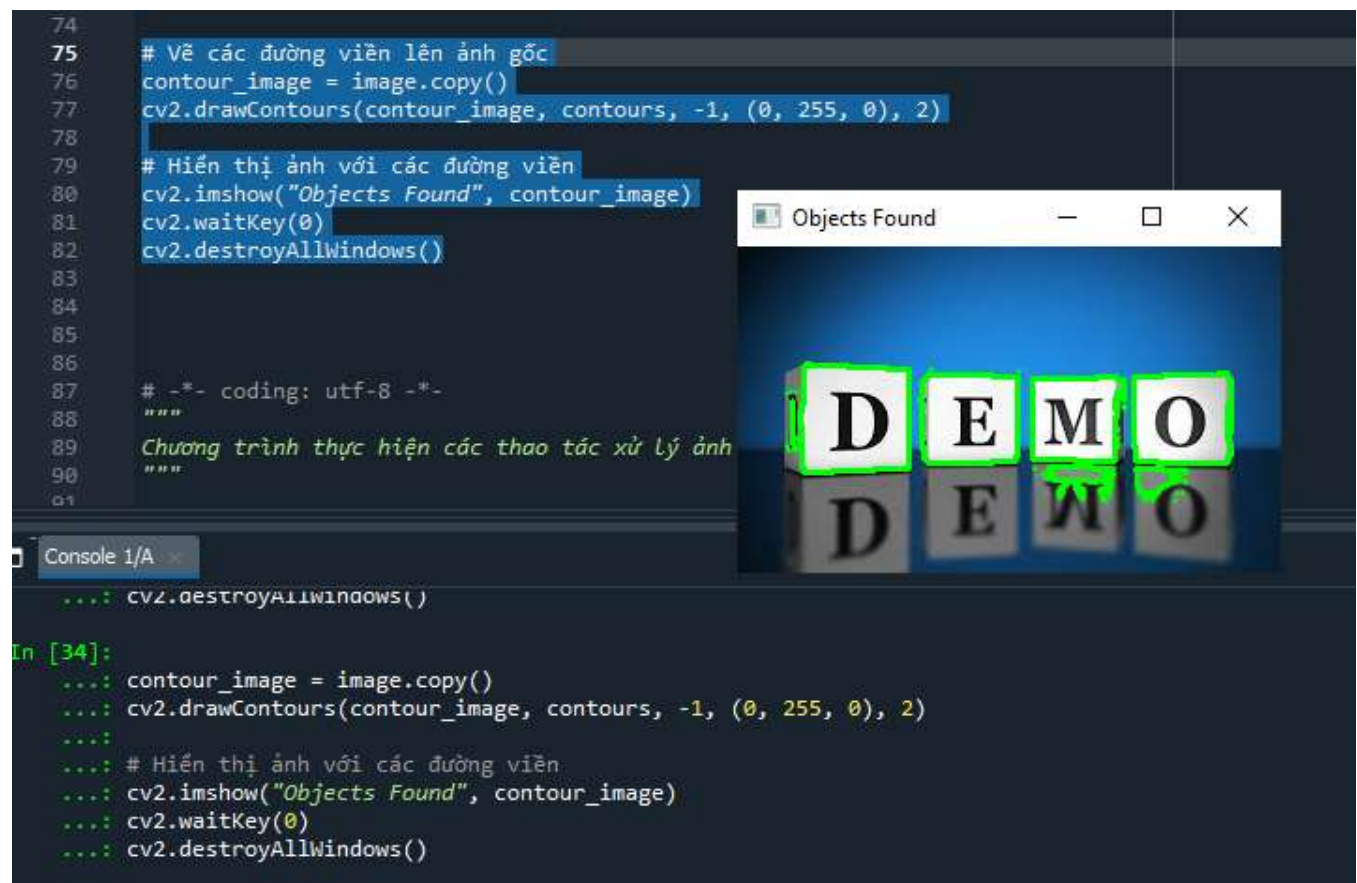
image: Ảnh lên trên đó các đường viền sẽ được vẽ.

contours: Danh sách các đường viền (từ hàm cv2.findContours()).

contourIdx: Chỉ số của đường viền cần vẽ. Sử dụng -1 để vẽ tất cả các đường viền.

color: Màu sắc của đường viền. Đối với ảnh màu, màu sắc được chỉ định dưới dạng (B, G, R). Đối với ảnh grayscale, chỉ cần chỉ định giá trị màu.

thickness: Độ dày của đường viền. Nếu là -1, đường viền sẽ được tô đầy.



Viết một chương trình hoàn chỉnh bao gồm các chức năng ở trên, các chức năng trong chương trình viết dạng thủ tục, được gọi khi cần.

```
import cv2
import numpy as np

def show_image(window_name, image):
    """Hiển thị ảnh trong cửa sổ."""
    cv2.imshow(window_name, image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

def read_image(file_path):
    """Đọc ảnh từ file và trả về ảnh."""
    image = cv2.imread(file_path)
    if image is None:
        raise ValueError(f"Không thể đọc ảnh từ {file_path}.")
    return image

def convert_to_grayscale(image):
    """Chuyển đổi ảnh màu sang grayscale."""
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

def convert_to_rgb(image):
    """Chuyển đổi ảnh màu BGR sang RGB."""
    return cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

def blur_image(image, kernel_size=(25, 25)):
    """Làm mờ ảnh với kernel Gaussian."""
    return cv2.GaussianBlur(image, kernel_size, 0)

def detect_edges(image):
    """Phát hiện cạnh sử dụng thuật toán Canny."""
    gray_image = convert_to_grayscale(image)
    return cv2.Canny(gray_image, 100, 200)

def find_contours(image):
    """Tìm và trả về các đường viền trong ảnh."""
    gray_image = convert_to_grayscale(image)
    _, binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(binary_image, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    return contours
```



```
def draw_contours(image, contours):
    """Vẽ các đường viền lên ảnh."""
    contour_image = image.copy()
    cv2.drawContours(contour_image, contours, -1, (0, 255, 0), 2)
    return contour_image

def main():
    """Chương trình chính để thực hiện các thao tác với ảnh."""
    file_path = 'file_example_JPG_100kB.jpg'

    # Đọc ảnh từ file
    image = read_image(file_path)

    # Hiển thị ảnh gốc
    show_image('Original Image', image)

    # Hiển thị ảnh grayscale
    gray_image = convert_to_grayscale(image)
    show_image('Grayscale Image', gray_image)

    # Hiển thị ảnh RGB
    rgb_image = convert_to_rgb(image)
    show_image('RGB Image', rgb_image)

    # Hiển thị ảnh làm mờ
    blurred_image = blur_image(image)
    show_image('Blurred Image', blurred_image)

    # Hiển thị ảnh với các cạnh phát hiện
    edges_image = detect_edges(image)
    show_image('Edges Image', edges_image)

    # Tìm và in các đường viền
    contours = find_contours(image)
    print(f"Đã tìm thấy {len(contours)} đường viền.")
    print(contours)

    # Hiển thị ảnh với các đường viền vẽ lên
    contour_image = draw_contours(image, contours)
    show_image('Contours Image', contour_image)

if __name__ == '__main__':
    main()
```


Kết quả chạy thử chương trình với `file_path = 'file_example_JPG_100kB.jpg'`

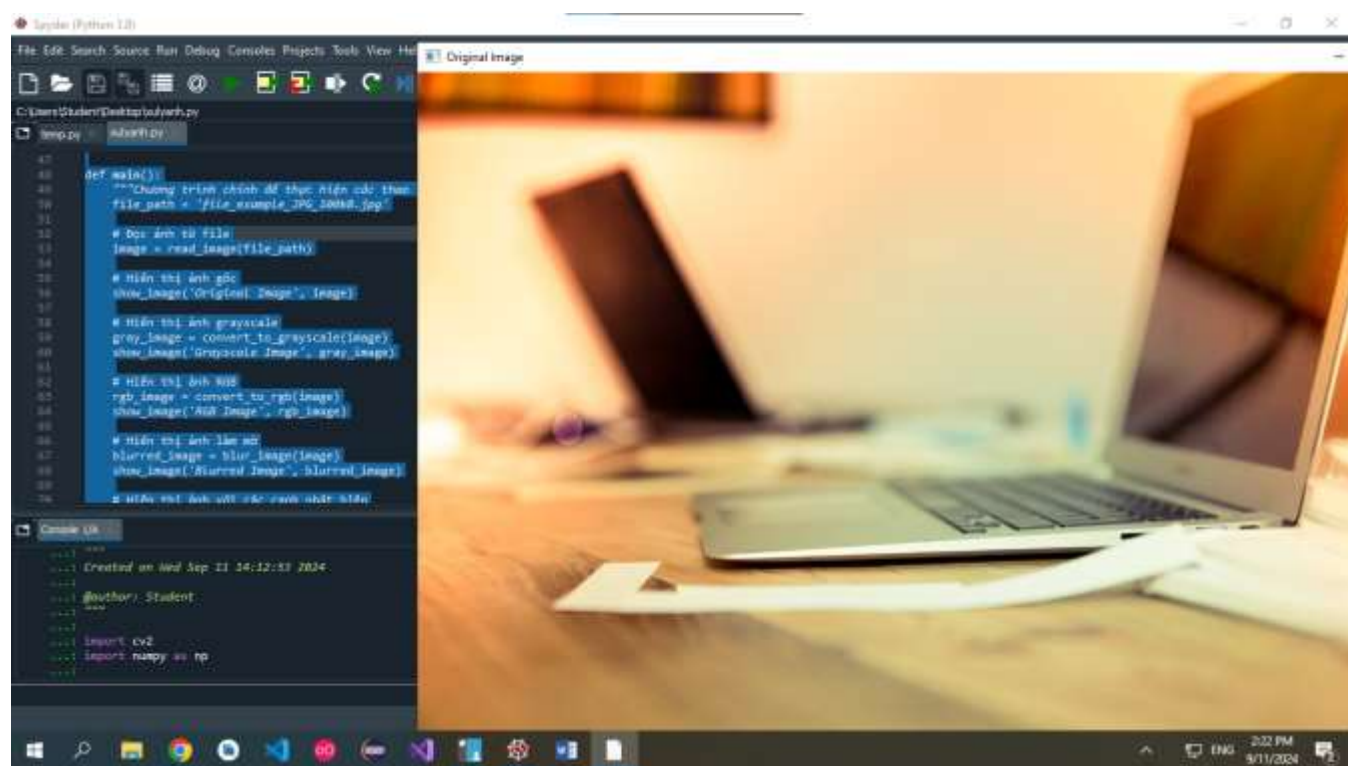
[https://file-](https://file-examples.com/storage/fef44df12666d835ba71c24/2017/10/file_example_JPG_100kB.jpg)

[examples.com/storage/fef44df12666d835ba71c24/2017/10/file_example_JPG_100kB.jpg](https://file-examples.com/storage/fef44df12666d835ba71c24/2017/10/file_example_JPG_100kB.jpg)

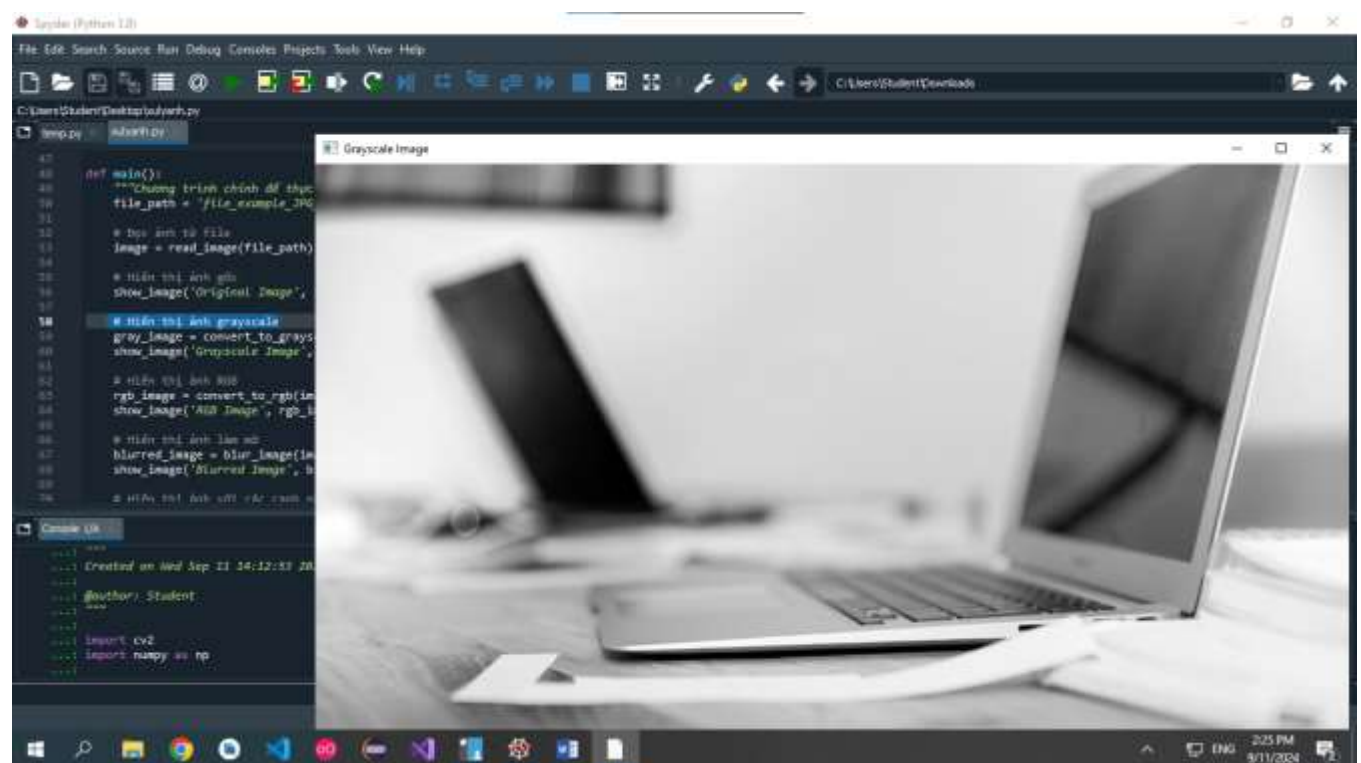


Ảnh gốc:

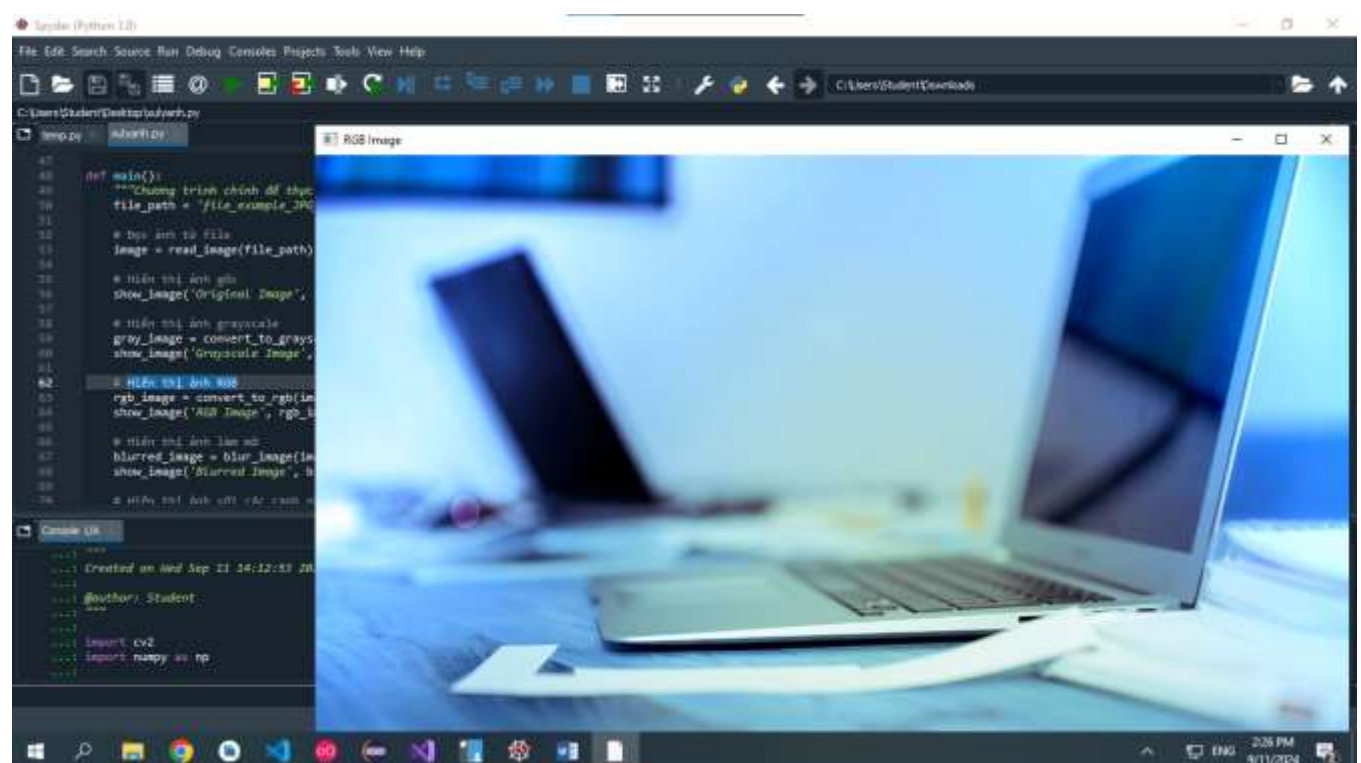
Hiển thị ảnh gốc



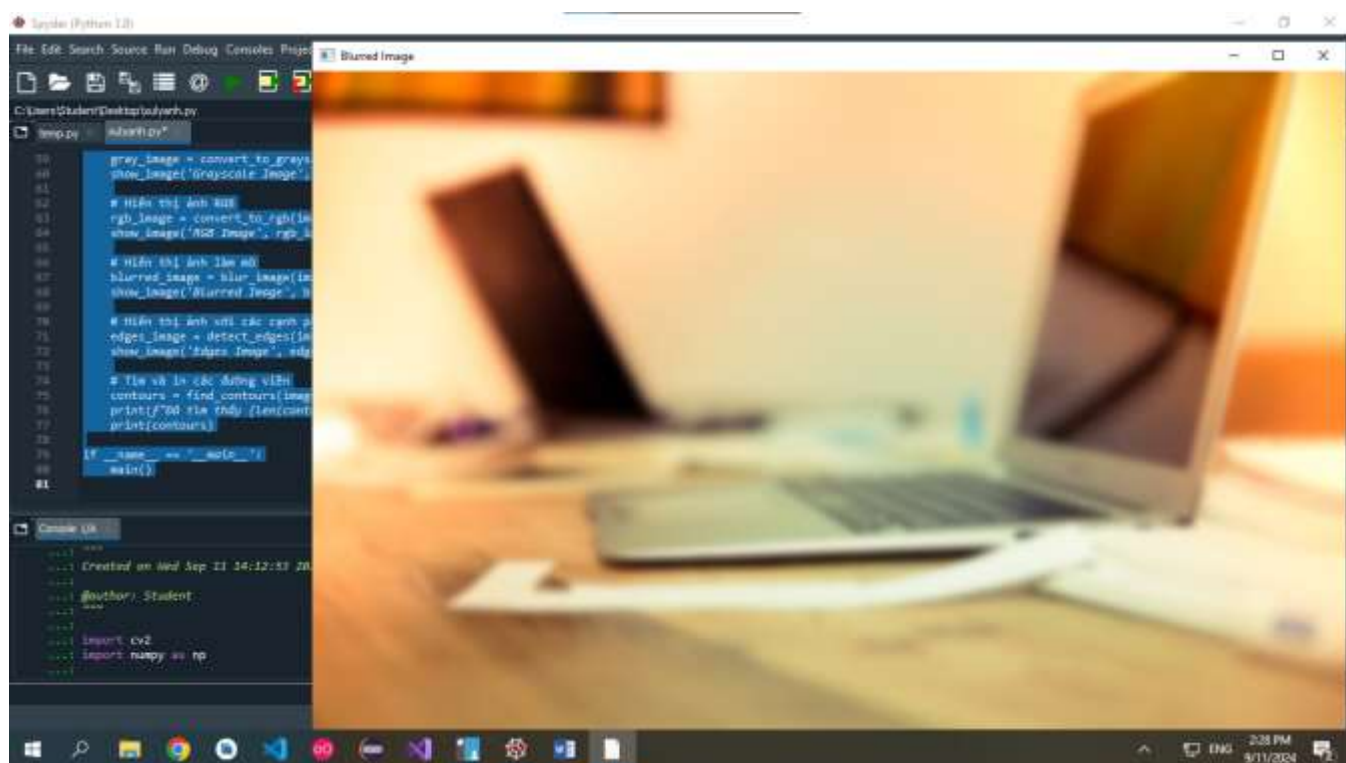
Hiển thị ảnh grayscale



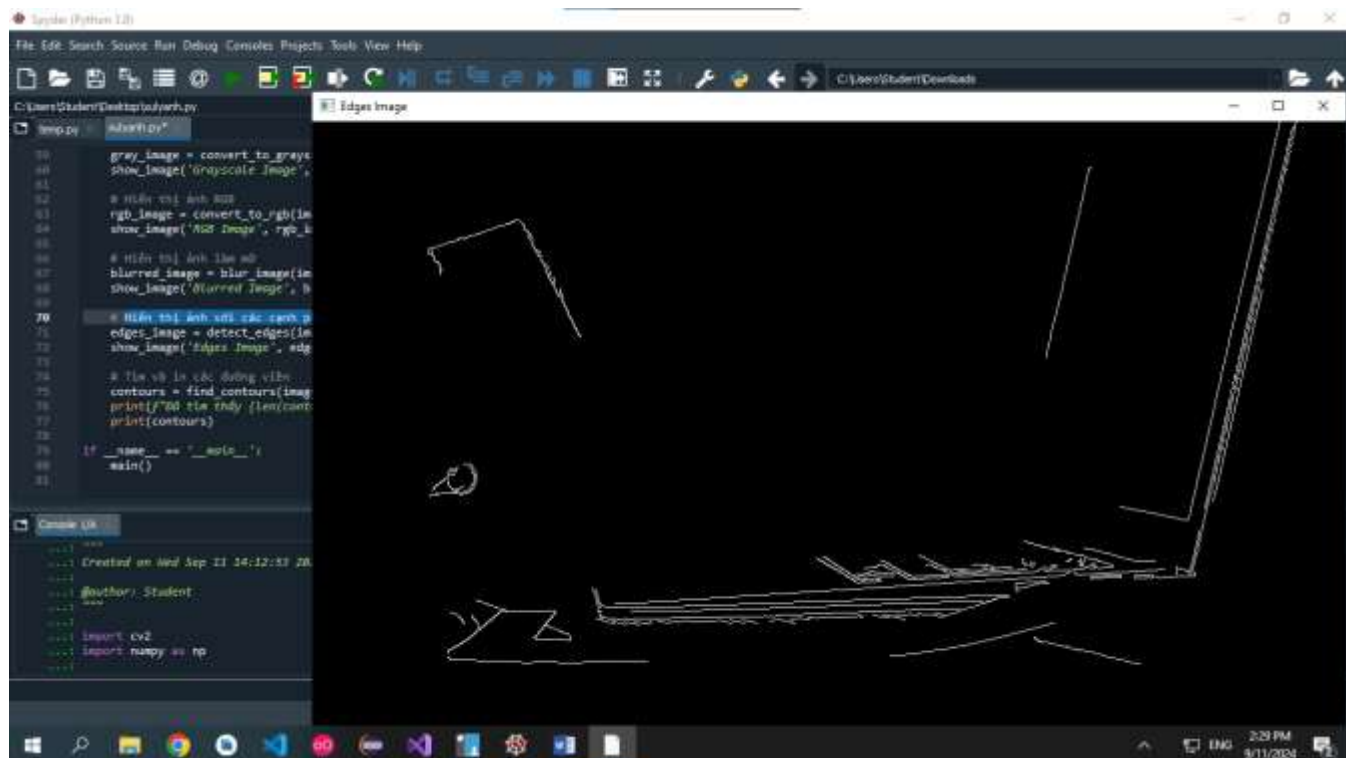
Hiển thị ảnh RGB



Hiển thị ảnh làm mờ



Hiển thị ảnh với các cạnh phát hiện



Tìm và in các đường viền

```
C:\Users\Student\Desktop\ulyanh.py
temp.py ulyanh.py*
59 gray_image = convert_to_grayscale(image)

Console I/A
[[175, 7]],
[[176, 7]],
[[176, 3]],
[[177, 2]],
[[177, 0]],
[[ 49, 0]],
[[ 48, 1]],
[[ 47, 0]],
[[ 46, 0]],
[[ 47, 1]],
[[ 46, 2]],
[[ 43, 2]],
[[ 42, 1]],
[[ 42, 0]], dtype=int32))

In [31]:
```

Hiển thị ảnh với các đường viền vẽ lên

