Sinh viên: Lưu Đức Đạt

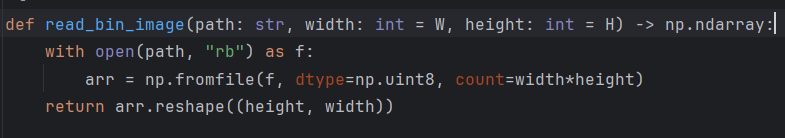
MSSV: N22DCCN019

HOMEWORK\_2

Bài 1.

**Giải thích bước làm:**

1) Đọc ảnh từ RAW nhị phân:



reshape((height, width)): đổi vector 1D thành ma trận 2D 256×256.

2) Chọn ngưỡng tách nền/mô: Trong code chọn *Threshold* = 95

3) Biến ảnh xám thành ảnh nhị phân:

Với từng pixel p:

* Nếu p > threshold → gán **255** (logic 1).
* Ngược lại → gán **0** (logic 0).

Ép kiểu về uint8 để đảm bảo đúng định dạng ảnh 8-bit.

Kết quả: ảnh nhị phân binary\_img với 2 mức **0/255**

4) Định nghĩa tiêu chí đường biên xấp xỉ:

Một pixel được coi là **đường biên** nếu:

1. Pixel đó thuộc giá trị **255.**
2. Trong **lân cận 3×3** quanh nó tồn tại ít nhất 1 pixel thuộc **nền** (giá trị **0**).

đường biên là nơi vùng mô “tiếp giáp” với nền.

5) Tạo ảnh đường biên xấp xỉ

Duyệt các vị trí (y, x) bên trong ảnh nhị phân (từ 1 → H−1, 1 → W−1).

Với mỗi (y, x):

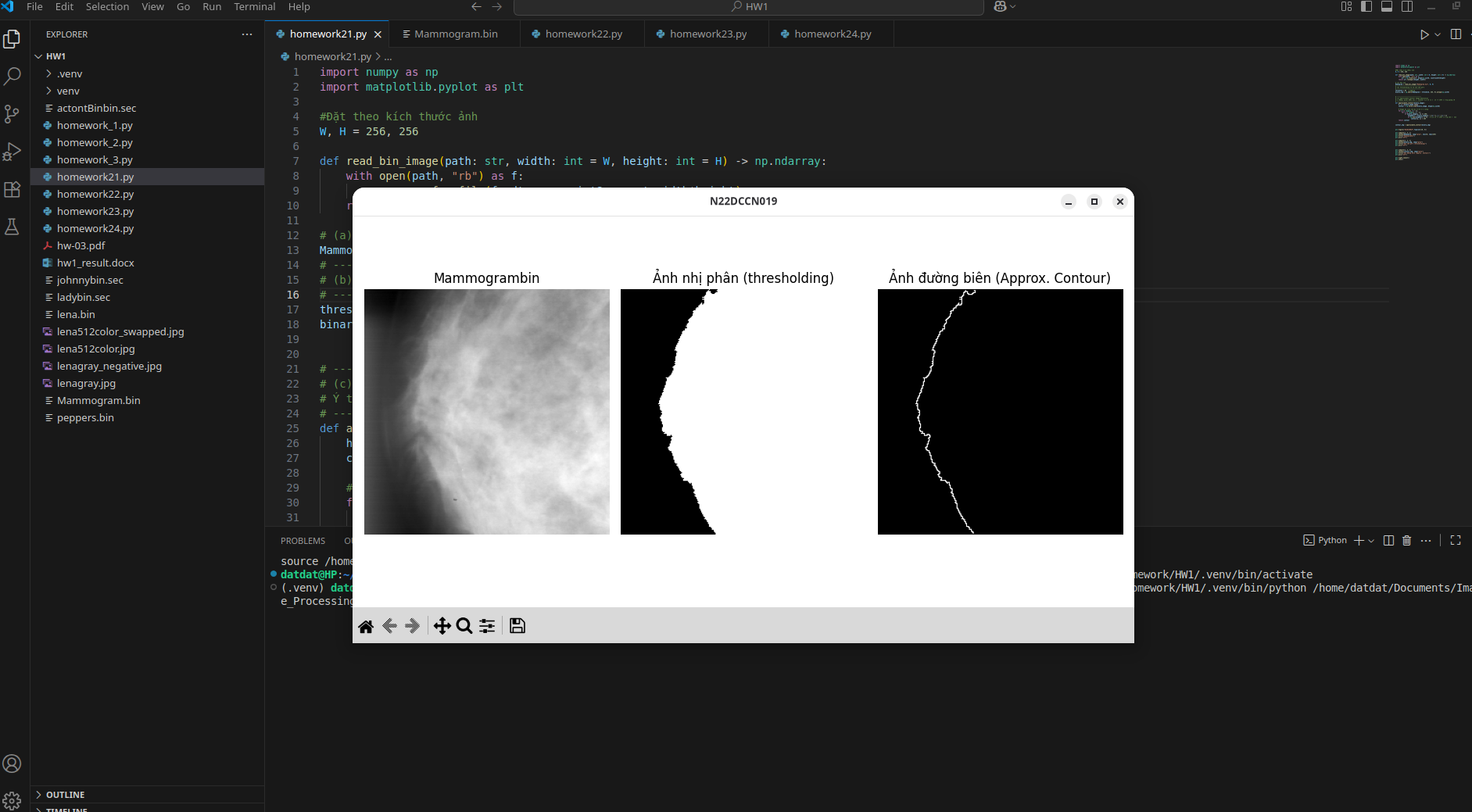
* Nếu binary\_img[y, x] == 255, lấy cửa sổ **3×3** xung quanh.
* Nếu trong cửa sổ tồn tại pixel **0**, gán contour\_img[y, x] = 255.

Kết quả: contour\_img là ảnh nhị phân biểu diễn đường biên của vùng mô.

1. Hiển thị tất cả các ảnh.

Trả lời ý c): Đúng vậy, mã chuỗi có thể biểu diễn đường viền chính vì về cơ bản nó là một ranh giới khép kín của vùng mô. Tuy nhiên: Nếu có nhiều đường viền, sẽ cần nhiều mã chuỗi. Mã chuỗi nhạy cảm với nhiễu và cần được xử lý trước (xác định ngưỡng + loại bỏ nhiễu). Đối với các ranh giới hình ảnh y tế mượt mà, mã chuỗi hợp lệ nhưng không phải là biểu diễn hiệu quả nhất (xấp xỉ spline hoặc đa giác có thể tốt hơn).

**Kết quả chạy demo:**

****

Bài 2.

**Giải thích bước làm:**

1) Mở file ở chế độ nhị phân ("rb").

2) Gọi read\_bin\_image("ladybin.sec") để lấy ma trận ảnh lady.

3) Histogram ảnh gốc

plt.subplot(2,2,2): chọn ô (hàng 1, cột 2).

plt.hist(lady.ravel(), bins=256, range=(0,255)):

**Trục X**: mức xám 0..255.

**Trục Y**: tần suất pixel ở từng mức xám.

ravel() “trải phẳng” ảnh 2D thành 1D để vẽ histogram.

4) Kéo giãn tương phản:

Tính f\_min = lady.min(), f\_max = lady.max() để biết khoảng giá trị đang dùng trong ảnh gốc.

Dùng công thức:

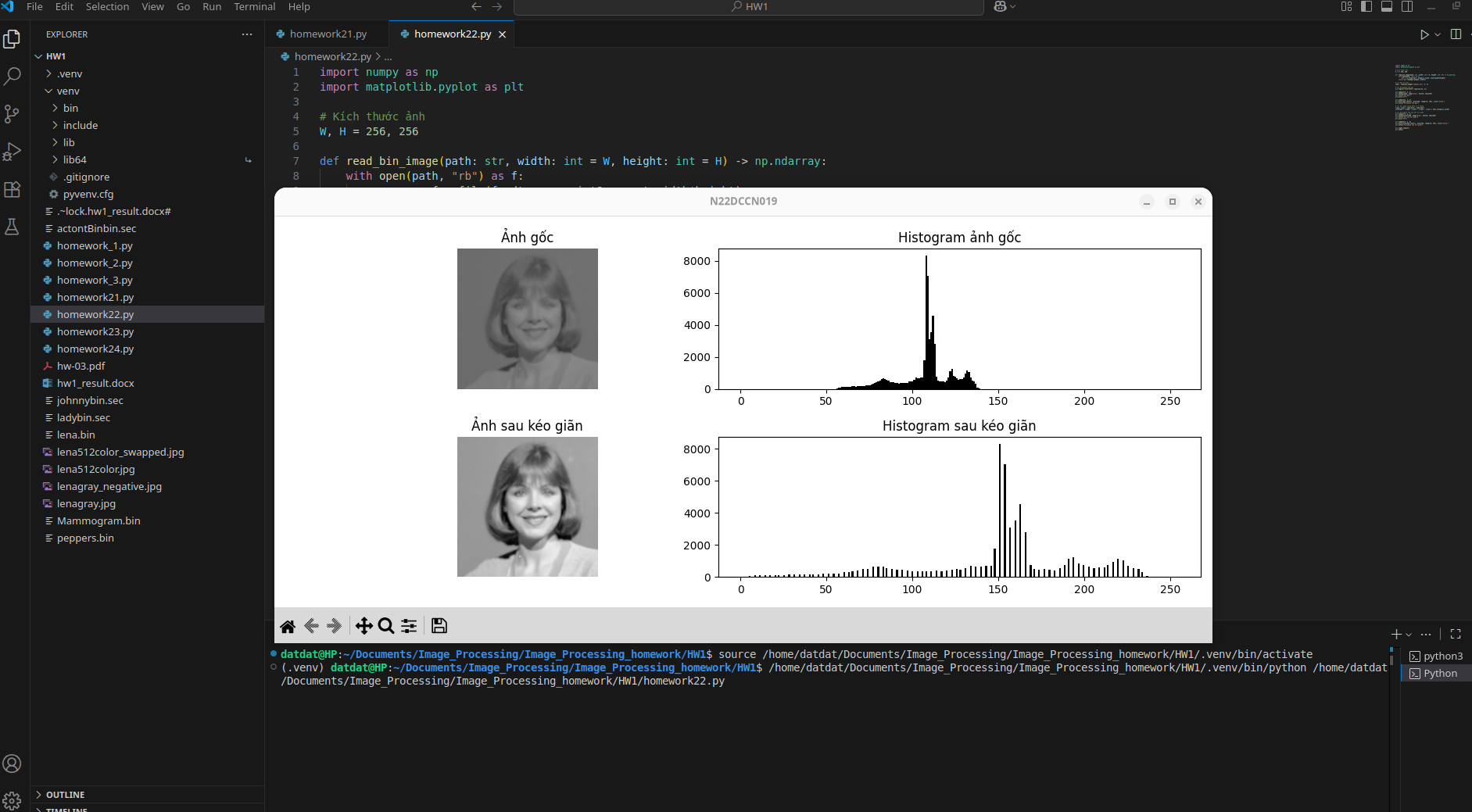
g(x,y)= ((f(x,y)−fmin) / (fmax​−fmin) ) × 255.

Ánh xạ **f\_min → 0**, **f\_max → 255**, các giá trị giữa được scale **tuyến tính**.

Kết quả ép kiểu uint8 để trở lại ảnh 8-bit.

5) Hiển thị hình ảnh gốc, histogram gốc, ảnh sau khi kéo giãn tương phản, histogram sau khi kéo giãn.

**Kết quả chạy demo:**

****

Bài 3:

**Giải thích bước làm:**

1)Đọc ảnh nhị phân RAW 256x256

2) Tạo template chữ T:

Template là khuôn (ảnh nhỏ) mô tả hình chữ **T** để đi quét tìm trong ảnh.

Chọn kích thước **47×15** cho kích cỡ chữ T.

Template khởi tạo toàn 0 (nền). Ta tô các vùng 255 tạo thành:

**Thanh ngang**: hàng 11–16 (index 10..15), phủ toàn bộ chiều ngang.

**Thanh dọc**: cột 7–10 (index 6..9), từ hàng 17–37 (index 16..36).

Vì ảnh là nhị phân (0/255), so khớp sẽ yêu cầu vừa foreground (255) khớp và vừa background (0) khớp → giảm false positives.

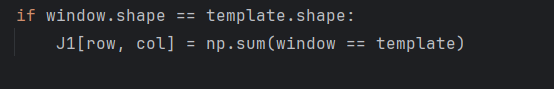
3) Trượt cửa sổ và tính match measure M2

Quét template trên ảnh:

**Cửa sổ**: tại mỗi vị trí tâm (i,j), cắt **window** cùng kích thước template (47×15).

**Miền quét hợp lệ:** i∈[⌊r/2⌋,H−1−⌊r/2⌋],𝑗∈[⌊𝑐/2⌋,𝑊−1−⌊𝑐/2⌋]để không vượt biên (với r=47,c=15).

Độ khớp M2 tại ( 𝑖 , 𝑗 ):



4) Xây dựng bản đồ độ khớp **J1:**

J1(i,j) nhận giá trị M2(i,j) ở mọi tâm có cửa sổ hợp lệ.

Ở các vị trí không đủ lân cận (rìa ngoài), J1 = 0 theo yêu cầu đề.

5) Chuẩn hóa **J1** để hiển thị:

Tìm cách đưa J1 (giá trị nguyên lớn) về thang hiển thị 8-bit.

J1\_vis=((J1−min(J1) /(max(J1)−min(J1)))​×255

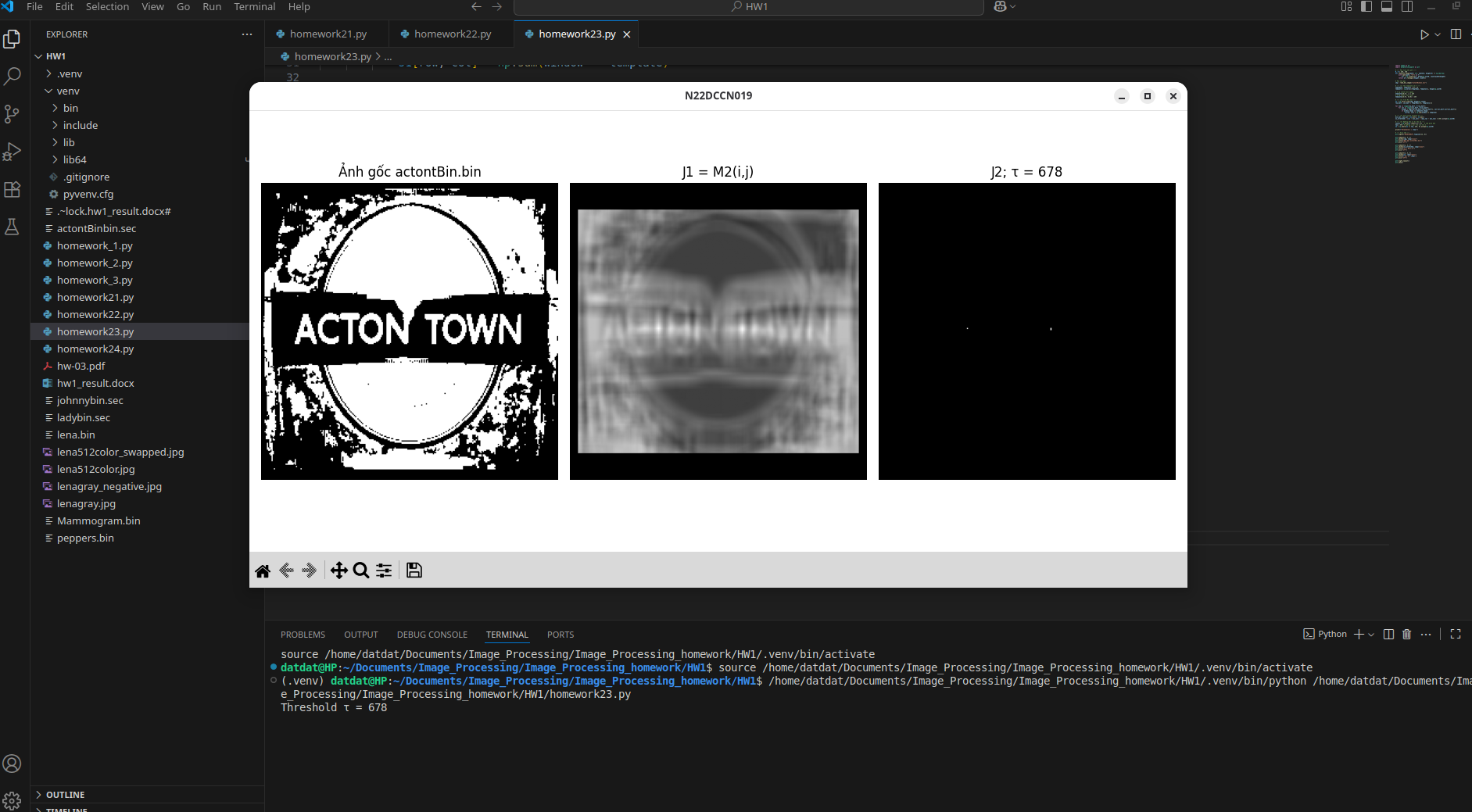
Hiển thị **J1** dưới dạng ảnh xám để quan sát vùng có độ khớp cao

6) Chọn ngưỡng **τ** và tạo ảnh phát hiện **J2.**

Ngưỡng τ: đặt lớn để chỉ giữ các đáp ứng mạnh nhất (tâm chữ T), ở đây chọn τ=678.

7) Hiển thị kết quả 3 ảnh ra màn hình.

**Kết quả chạy demo:**

****

Bài 4:

**Giải thích bước làm:**

1) Đọc ảnh RAW 256×256:

Định nghĩa hàm read\_bin\_image để mở file nhị phân, đọc đúng 256\*256 byte (8- bit/pixel), rồi reshape thành ma trận 2D 256×256.

Gọi hàm để lấy ảnh gốc img.

2)Tạo hàm Histogram Equalization:

**Đếm histogram**: np.histogram(image.flatten(), bins=256, range=[0,256]) → hist (tần suất 0..255), bins

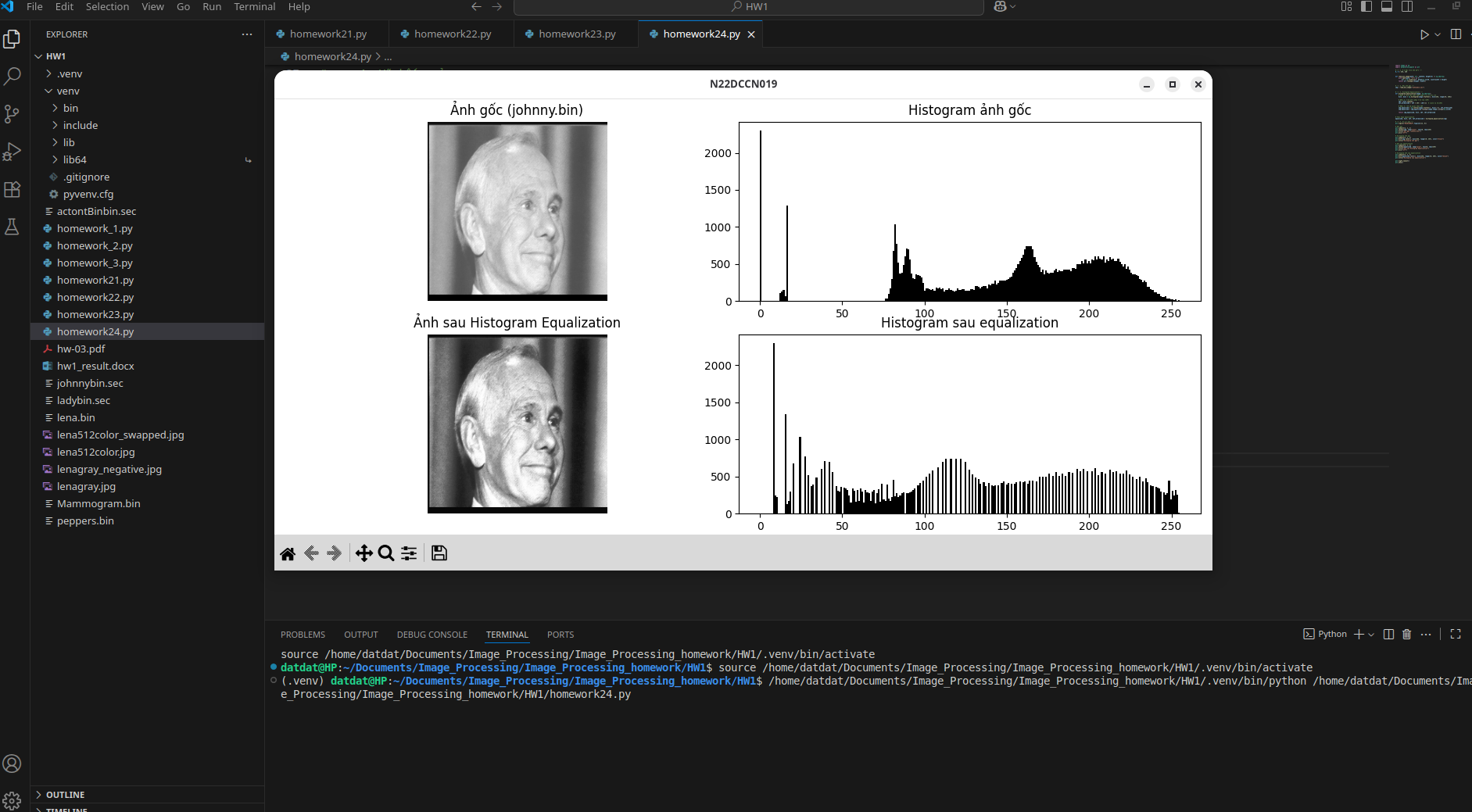
**Tính CDF**: cdf = hist.cumsum(); **chuẩn hóa** về 0..255: cdf\_normalized = cdf\*255/cdf[-1].

**Ánh xạ mức xám**: dùng np.interp để thay mỗi pixel bằng giá trị CDF đã chuẩn hóa tương ứng → thu được equalized (ảnh đã cân bằng).

Trả về equalized, hist, cdf, cdf\_normalized.

3) Gọi hàm và vẽ 4 ô kết quả.

**Kết quả chạy demo:**

****