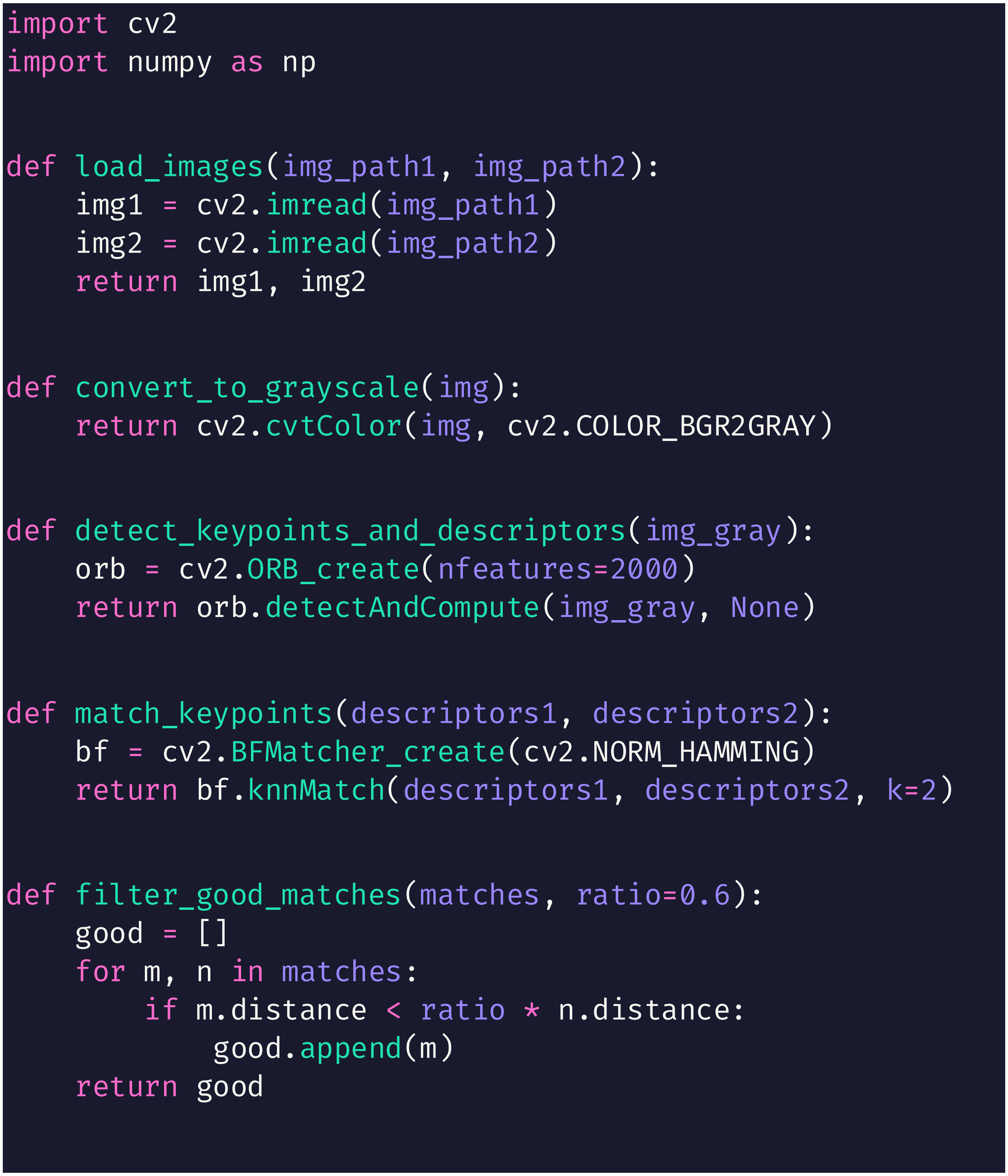
Nguyễn Hoàng Thanh – 2151013087

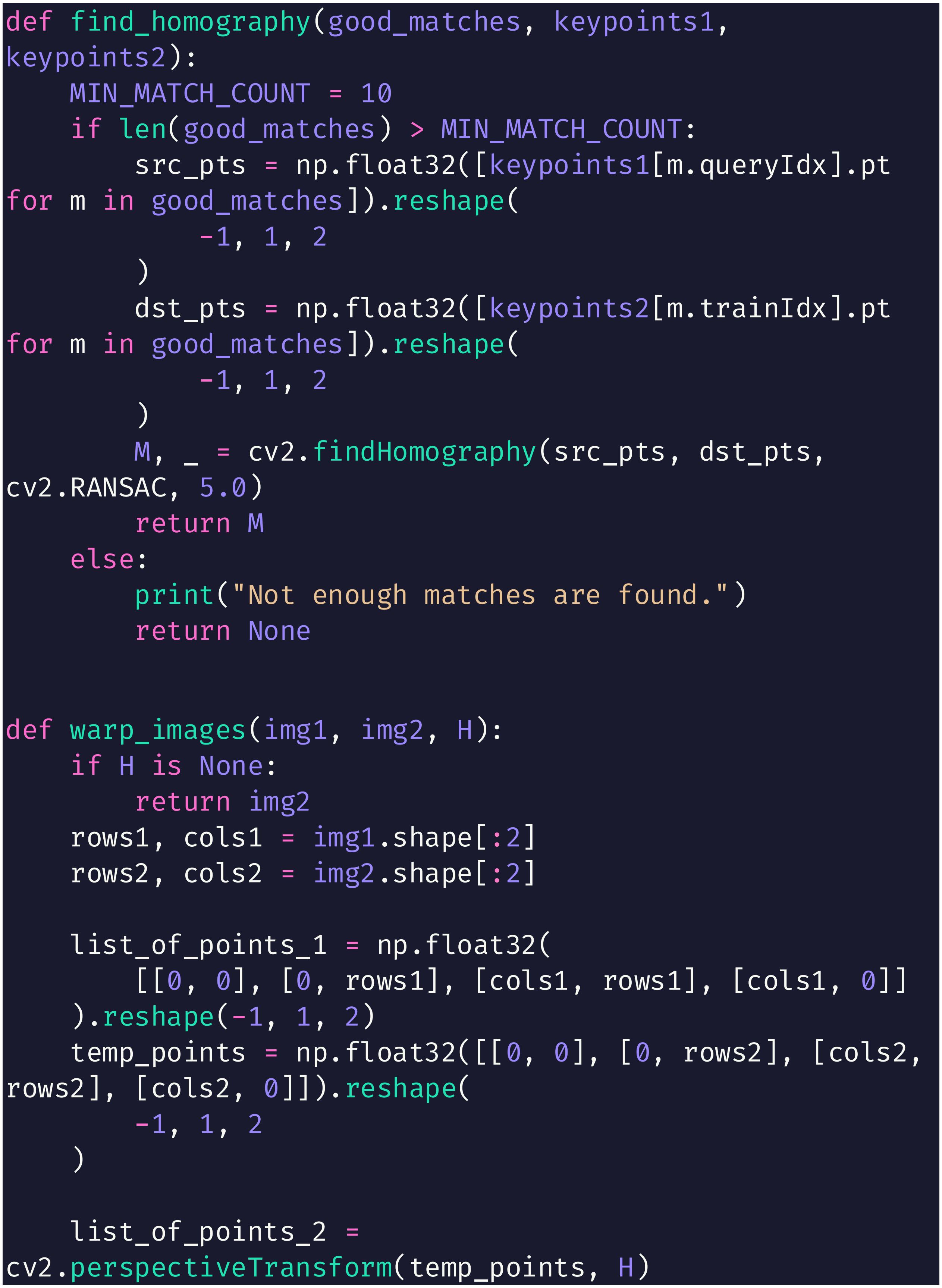
LAB 4: PANORAMA

Assigment 1:

Code:

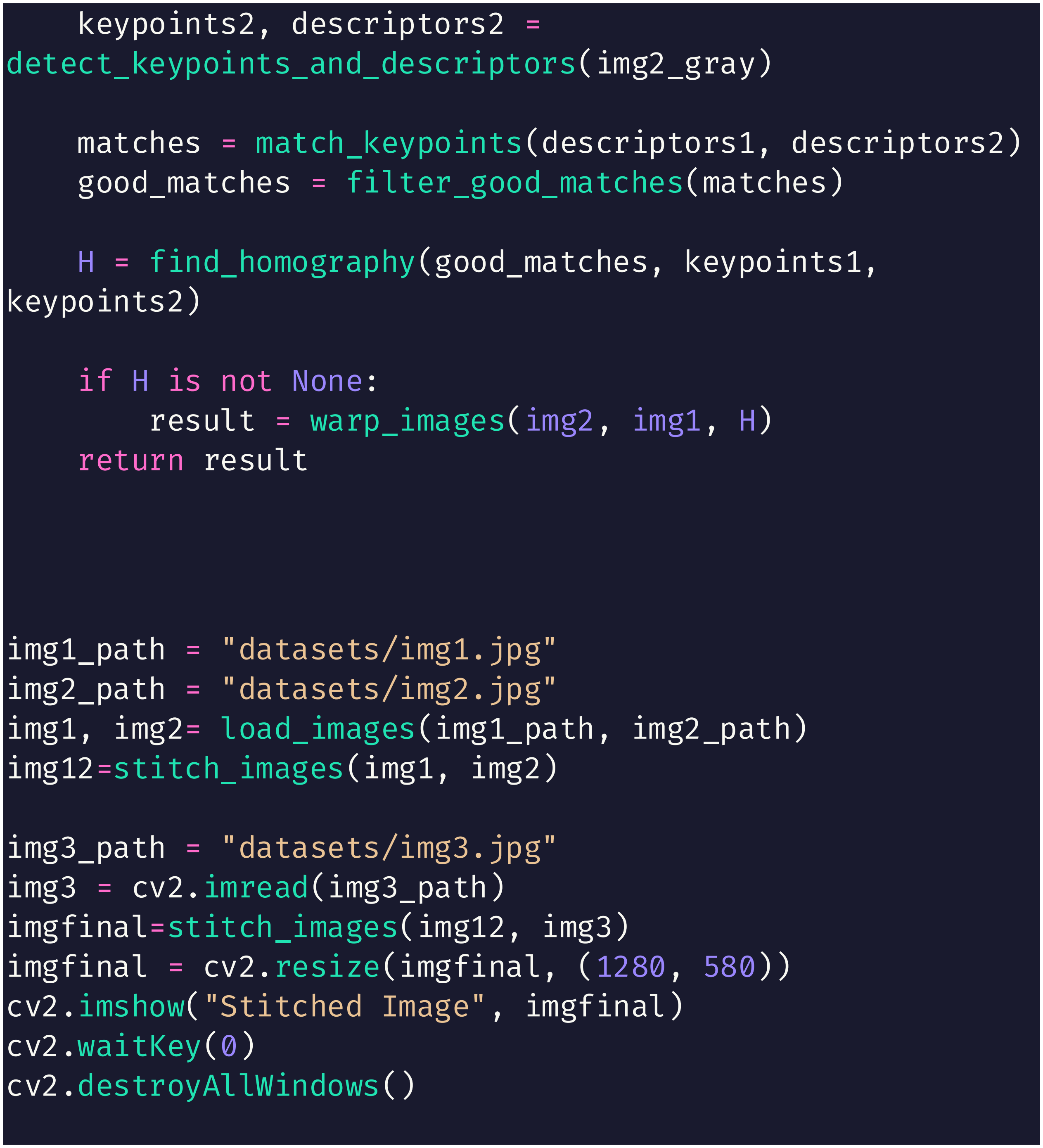


```python
import cv2
import numpy as np


def load_images(img_path1, img_path2):
    img1 = cv2.imread(img_path1)
    img2 = cv2.imread(img_path2)
    return img1, img2


def convert_to_grayscale(img):
    return cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


def detect_keypoints_and_descriptors(img_gray):
    orb = cv2.ORB_create(nfeatures=2000)
    return orb.detectAndCompute(img_gray, None)


def match_keypoints(descriptors1, descriptors2):
    bf = cv2.BFMatcher_create(cv2.NORM_HAMMING)
    return bf.knnMatch(descriptors1, descriptors2, k=2)


def filter_good_matches(matches, ratio=0.6):
    good = []
    for m, n in matches:
        if m.distance < ratio * n.distance:
            good.append(m)
    return good
```



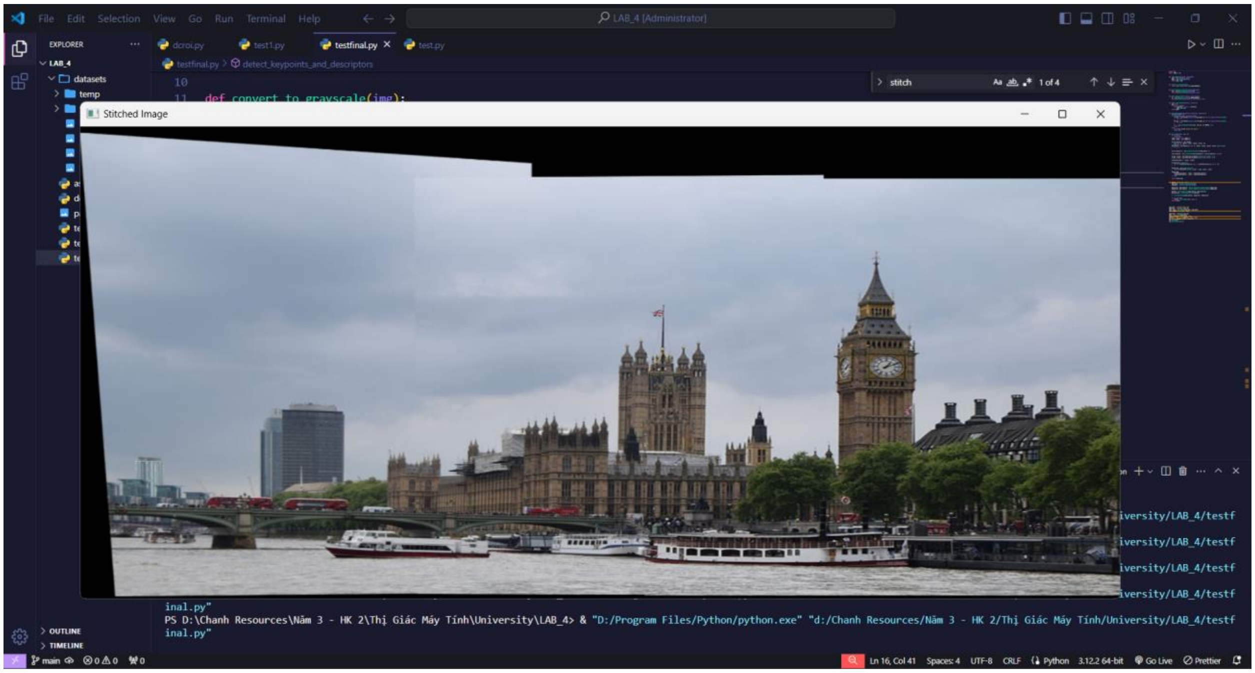
```python
def find_homography(good_matches, keypoints1,
keypoints2):
    MIN_MATCH_COUNT = 10
    if len(good_matches) > MIN_MATCH_COUNT:
        src_pts = np.float32([keypoints1[m.queryIdx].pt
for m in good_matches]).reshape(
            -1, 1, 2
        )
        dst_pts = np.float32([keypoints2[m.trainIdx].pt
for m in good_matches]).reshape(
            -1, 1, 2
        )
        M, _ = cv2.findHomography(src_pts, dst_pts,
cv2.RANSAC, 5.0)
        return M
    else:
        print("Not enough matches are found.")
        return None


def warp_images(img1, img2, H):
    if H is None:
        return img2
    rows1, cols1 = img1.shape[:2]
    rows2, cols2 = img2.shape[:2]

    list_of_points_1 = np.float32(
        [[0, 0], [0, rows1], [cols1, rows1], [cols1, 0]]
    ).reshape(-1, 1, 2)
    temp_points = np.float32([[0, 0], [0, rows2], [cols2,
rows2], [cols2, 0]]).reshape(
        -1, 1, 2
    )

    list_of_points_2 =
cv2.perspectiveTransform(temp_points, H)
```

```
    list_of_points = np.concatenate((list_of_points_1,
list_of_points_2), axis=0)

    [x_min, y_min] =
np.int32(list_of_points.min(axis=0).ravel() - 0.5)
    [x_max, y_max] =
np.int32(list_of_points.max(axis=0).ravel() + 0.5)

    translation_dist = [-x_min, -y_min]

    H_translation = np.array(
        [[1, 0, translation_dist[0]], [0, 1,
translation_dist[1]], [0, 0, 1]]
    )

    output_img = cv2.warpPerspective(
        img2, H_translation.dot(H), (x_max - x_min, y_max
- y_min)
    )
    output_img[
        translation_dist[1] : rows1 +
translation_dist[1],
        translation_dist[0] : cols1 +
translation_dist[0],
    ] = img1

    return output_img


def stitch_images(img1, img2):
    img1_gray = convert_to_grayscale(img1)
    img2_gray = convert_to_grayscale(img2)

    keypoints1, descriptors1 =
detect_keypoints_and_descriptors(img1_gray)
```



```
    keypoints2, descriptors2 =
detect_keypoints_and_descriptors(img2_gray)

    matches = match_keypoints(descriptors1, descriptors2)
    good_matches = filter_good_matches(matches)

    H = find_homography(good_matches, keypoints1,
keypoints2)

    if H is not None:
        result = warp_images(img2, img1, H)
    return result




img1_path = "datasets/img1.jpg"
img2_path = "datasets/img2.jpg"
img1, img2= load_images(img1_path, img2_path)
img12=stitch_images(img1, img2)

img3_path = "datasets/img3.jpg"
img3 = cv2.imread(img3_path)
imgfinal=stitch_images(img12, img3)
imgfinal = cv2.resize(imgfinal, (1280, 580))
cv2.imshow("Stitched Image", imgfinal)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Result:

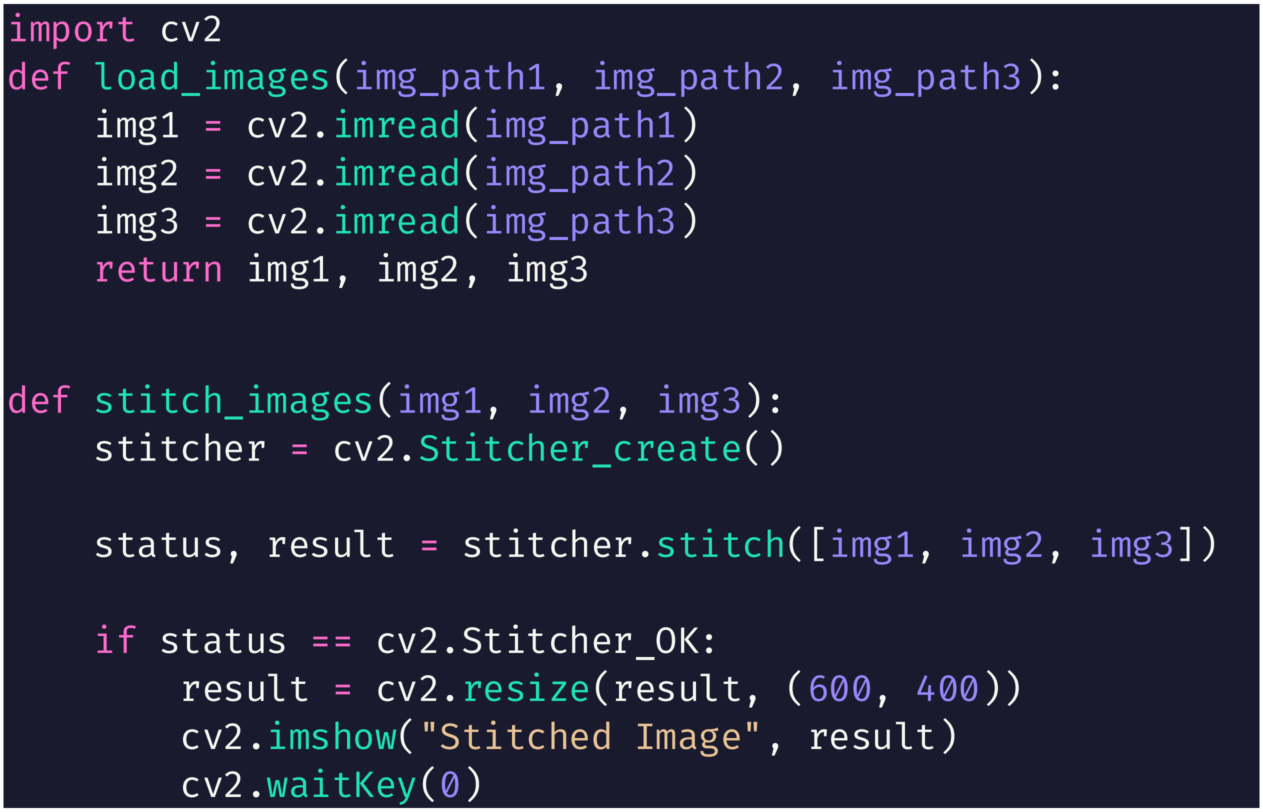


```
def convert_to_grayscale(img):
```

```
iversity/LAB_4/testf
iversity/LAB_4/testf
iversity/LAB_4/testf
iversity/LAB_4/testf
inal.py"
PS D:\Chanh Resources\Năm 3 - HK 2\Thị Giác Máy Tính\University\LAB_4> & "D:/Program Files/Python/python.exe" "d:/Chanh Resources/Năm 3 - HK 2/Thị Giác Máy Tính/University/LAB_4/testf
inal.py"
```

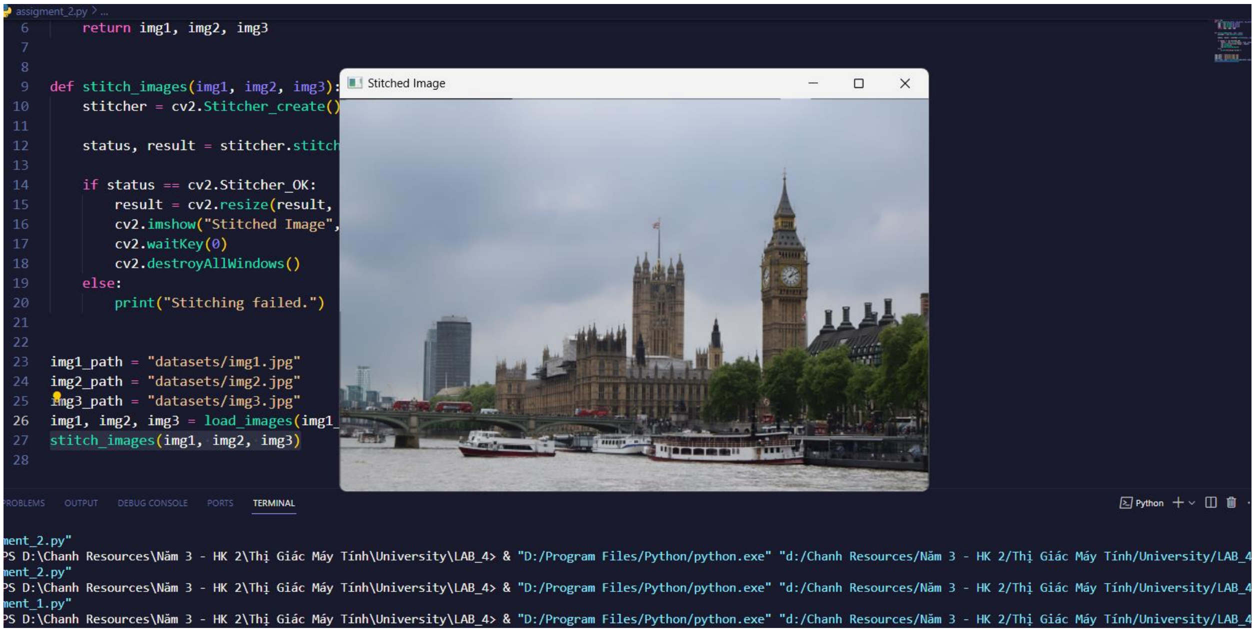
Assigment 2:

Code:



```python
import cv2
def load_images(img_path1, img_path2, img_path3):
    img1 = cv2.imread(img_path1)
    img2 = cv2.imread(img_path2)
    img3 = cv2.imread(img_path3)
    return img1, img2, img3


def stitch_images(img1, img2, img3):
    stitcher = cv2.Stitcher_create()

    status, result = stitcher.stitch([img1, img2, img3])

    if status == cv2.Stitcher_OK:
        result = cv2.resize(result, (600, 400))
        cv2.imshow("Stitched Image", result)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
    else:
        print("Stitching failed.")


img1_path = "datasets/img1.jpg"
img2_path = "datasets/img2.jpg"
img3_path = "datasets/img3.jpg"
img1, img2, img3 = load_images(img1_path, img2_path,
img3_path)
stitch_images(img1, img2, img3)
```

Result:



```python
    return img1, img2, img3


def stitch_images(img1, img2, img3):
    stitcher = cv2.Stitcher_create()

    status, result = stitcher.stitch

    if status == cv2.Stitcher_OK:
        result = cv2.resize(result,
        cv2.imshow("Stitched Image",
        cv2.waitKey(0)
        cv2.destroyAllWindows()
    else:
        print("Stitching failed.")


img1_path = "datasets/img1.jpg"
img2_path = "datasets/img2.jpg"
img3_path = "datasets/img3.jpg"
img1, img2, img3 = load_images(img1_
stitch_images(img1, img2, img3)
```

```
ment_2.py"
PS D:\Chanh Resources\Năm 3 - HK 2\Thị Giác Máy Tính\University\LAB_4> & "D:/Program Files/Python/python.exe" "d:/Chanh Resources/Năm 3 - HK 2/Thị Giác Máy Tính/University/LAB_4
ment_2.py"
PS D:\Chanh Resources\Năm 3 - HK 2\Thị Giác Máy Tính\University\LAB_4> & "D:/Program Files/Python/python.exe" "d:/Chanh Resources/Năm 3 - HK 2/Thị Giác Máy Tính/University/LAB_4
ment_1.py"
PS D:\Chanh Resources\Năm 3 - HK 2\Thị Giác Máy Tính\University\LAB_4> & "D:/Program Files/Python/python.exe" "d:/Chanh Resources/Năm 3 - HK 2/Thị Giác Máy Tính/University/LAB_4
```