

CS 5330: Pattern Recognition and Computer Vision

Northeastern University

OpenCV Workshop

Lab 3: Basics of Images

** Contributed by Fall 2024 TAs: Byunghyun Ko, Yihan Wang and Taiwei Cui*

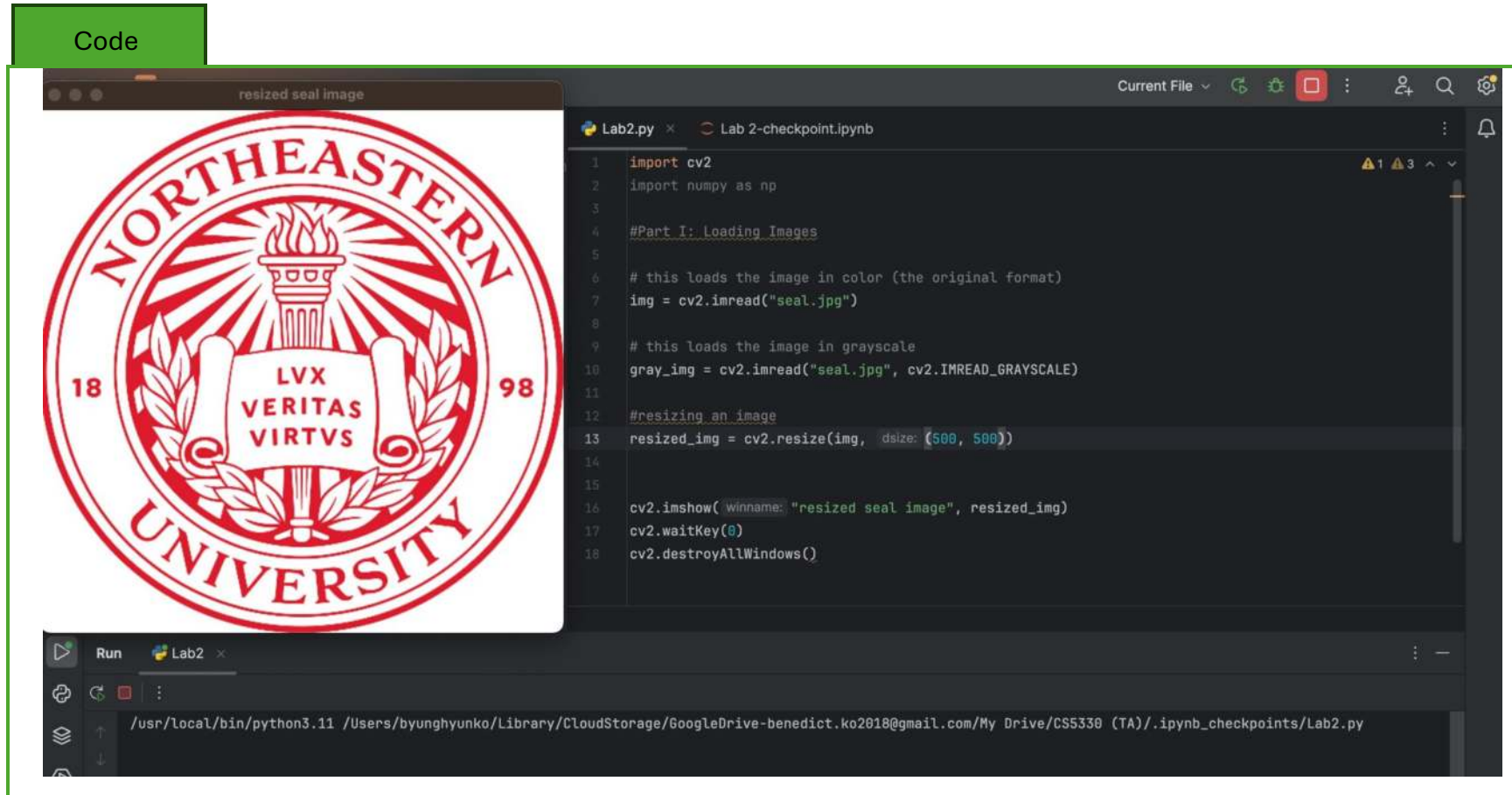
Basics of Images

1. Basic Image Operations
 1. Resizing Images
 2. Cropping Images
 3. Rotating Images
2. Image Colors Spaces and Conversions
3. Drawing Shapes and Text

Basic Image Operations - Resizing

- `cv2.resize(image, (width, height), fx, fy, interpolation)`
 - source: Input Image array (Single-channel, 8-bit or floating-point)
 - dsize: Size of the output array
 - dest: Output array (Similar to the dimensions and type of Input image array) [optional]
 - fx: Scale factor along the horizontal axis [optional]
 - fy: Scale factor along the vertical axis [optional]
 - interpolation: One of the above interpolation methods [optional]

Basic Image Operations - Resizing



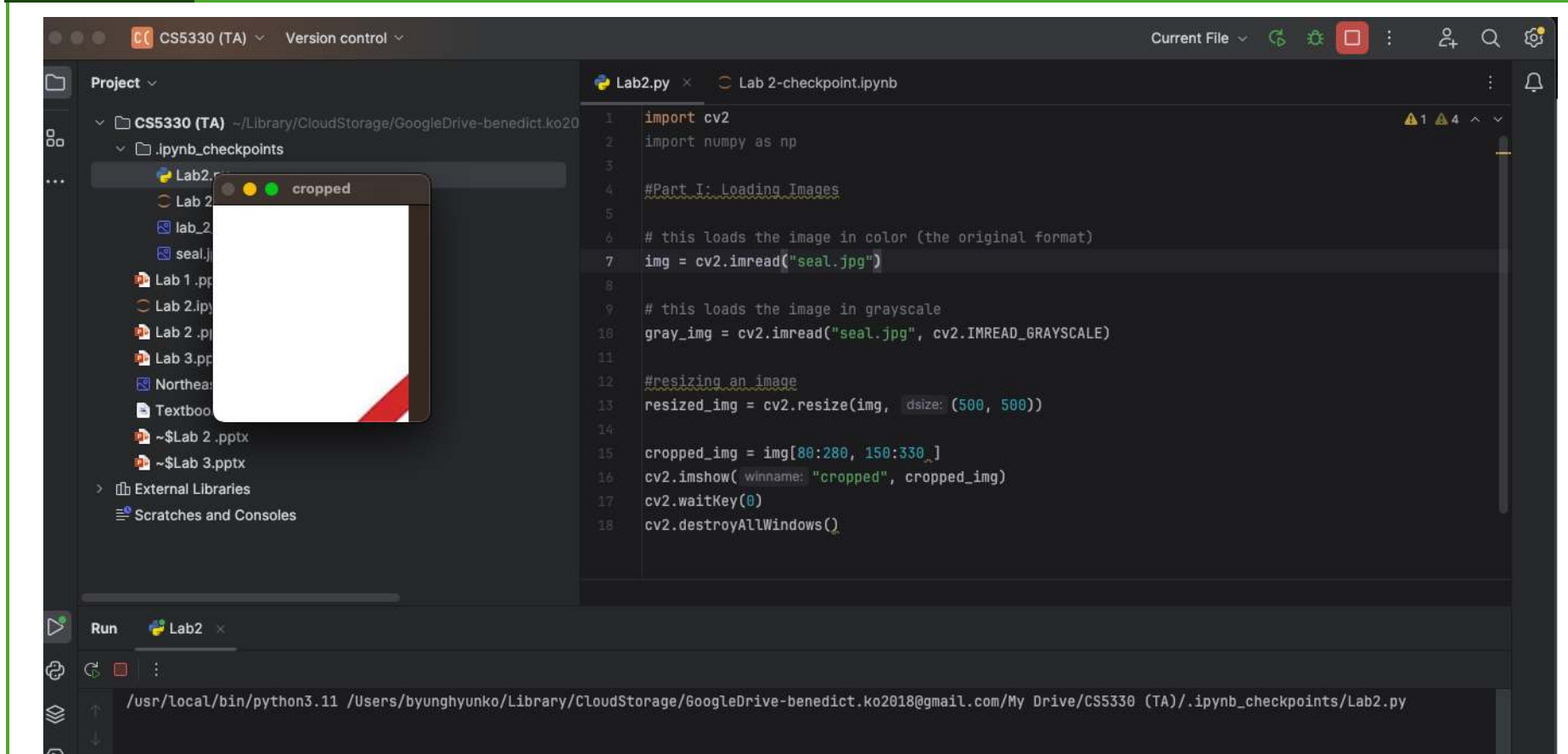
- Resizing an Image:
 - `resized_img = cv2.resize(image, (width, height), fx, fy, interpolation)`

Basic Image Operations - Cropping

- When cropping, simply specify the height and width (in pixels) of the area to be cropped
- `cropped_img = img[y + yh, x + xw]`
- This is the same method as NumPy array slicing
- First dimension is **always** the number of rows and heights of the image
- Second dimension is the number of columns or the width of the image

Basic Image Operations - Cropping

Code



The screenshot shows a Jupyter Notebook environment with a file explorer on the left, a code editor in the center, and a terminal at the bottom. A small window titled 'cropped' displays a white image with a red corner, representing the result of the cropping operation. The code in the notebook is as follows:

```
1 import cv2
2 import numpy as np
3
4 #Part I: Loading Images
5
6 # this loads the image in color (the original format)
7 img = cv2.imread("seal.jpg")
8
9 # this loads the image in grayscale
10 gray_img = cv2.imread("seal.jpg", cv2.IMREAD_GRAYSCALE)
11
12 #resizing an image
13 resized_img = cv2.resize(img, dsize=(500, 500))
14
15 cropped_img = img[80:280, 150:330]
16 cv2.imshow( winname: "cropped", cropped_img)
17 cv2.waitKey(0)
18 cv2.destroyAllWindows()
```

The terminal at the bottom shows the command used to run the notebook:

```
/usr/local/bin/python3.11 /Users/byunghyunko/Library/CloudStorage/GoogleDrive-benedict.ko2018@gmail.com/My Drive/CS5330 (TA)/.ipynb_checkpoints/Lab2.py
```

- Cropping an Image:
 - `cropped_img = img[y + yh, x + xw]`

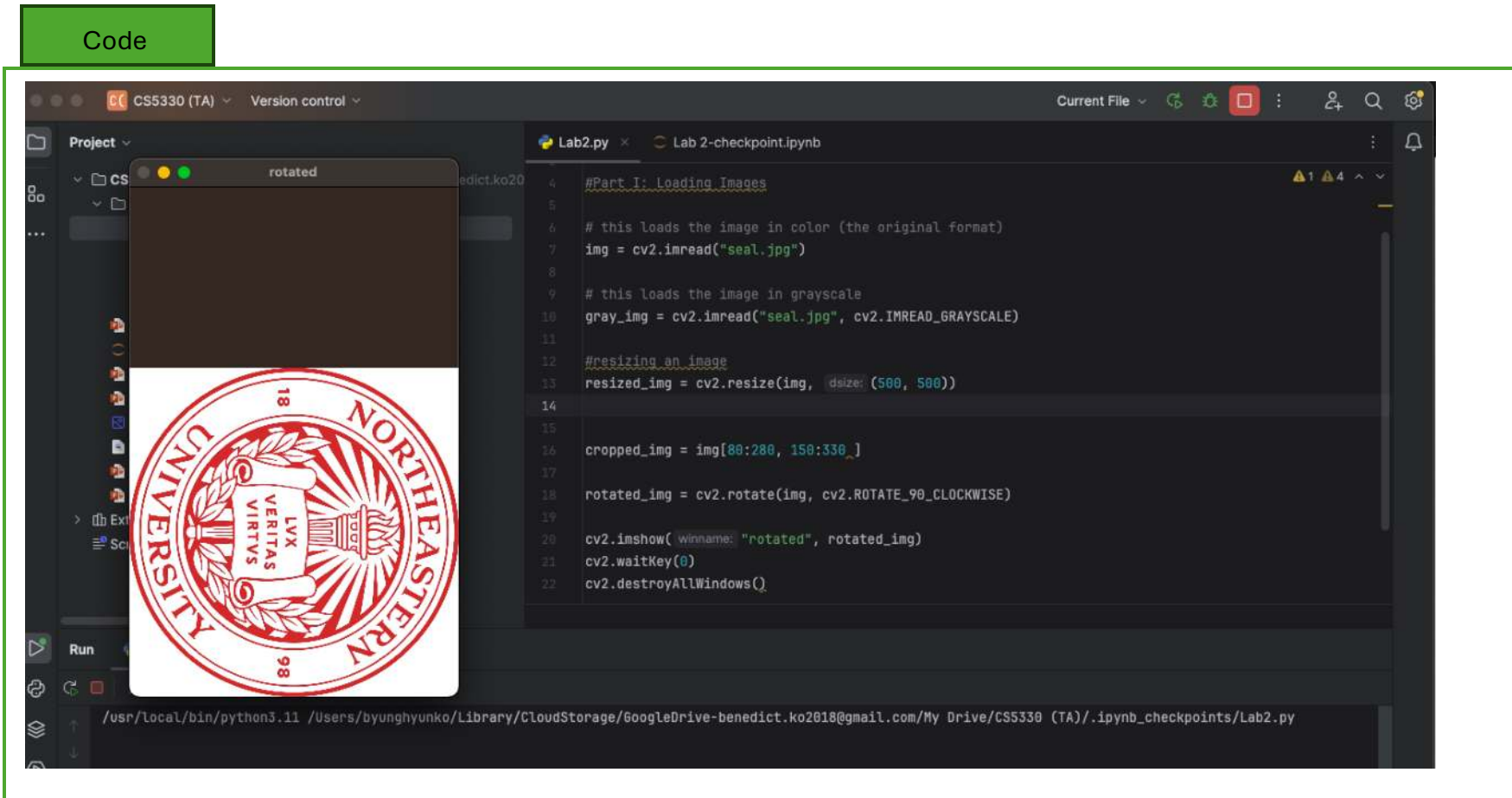
Basic Image Operations - Rotating

- `cv2.rotate(src, rotateCode[,dst])`
 - `rotateCode`: enum to specify how to rotate the array
 - To rotate clockwise by 90 degrees:
 - `cv2.ROTATE_90_CLOCKWISE`
 - To rotate counter-clockwise by 90 degrees:
 - `cv2.ROTATE_90_COUNTERCLOCKWISE`
 - To rotate clockwise by 180 degrees:
 - `Cv2.ROTATE_180`

Basic Image Operations - Rotating

- How about if you want to rotate in any other degree?
 - `(h, w) = image.shape[:2]`
 - `center = (w // 2, h // 2)`
 - `matrix = cv2.getRotationMatrix2D(center, angle, scale)`
 - `rotated_image = cv2.warpAffine(image, matrix, (w, h))`
- `cv2.getRotationMatrix2D()` is used to make the transformation matrix M which will be used for rotating an image
 - Center = center of rotation
 - Angle: Angle is positive for anti-clockwise and negative for clockwise
 - Scale: scaling factor which scales the image
- `cv2.warpAffine()` is used to apply affine transformation to an image, which is a linear mapping method that preserves points, straight lines, and planes

Basic Image Operations - Rotating



- `cv2.rotate(src, rotateCode[, dst])`

Image Colors Spaces and Conversions

- When loading images, OpenCV loads images in RGB by default
- Can use **cv2.cvtColor** to convert RGB to **Grayscale** or **HSV**
 - **Grayscale** is a color space where each pixel value represents the intensity of light. Instead of having multiple color channels (like RGB), a grayscale image has only one channel, which contains shades of gray ranging from black to white.
 - Grayscale is used in image thresholding, edge/object detection, and facial recognition
 - **HSV** is a cylindrical color space that describes colors in terms of three components:
 - Hue (H): The color type, represented as a degree on the color wheel (0-360°). In OpenCV, it's scaled to 0-179.
 - Saturation (S): The intensity or purity of the color (0-255).
 - Value (V): The brightness or lightness of the color (0-255).
 - HSV is used in color filtering, object tracking, and image segmentation

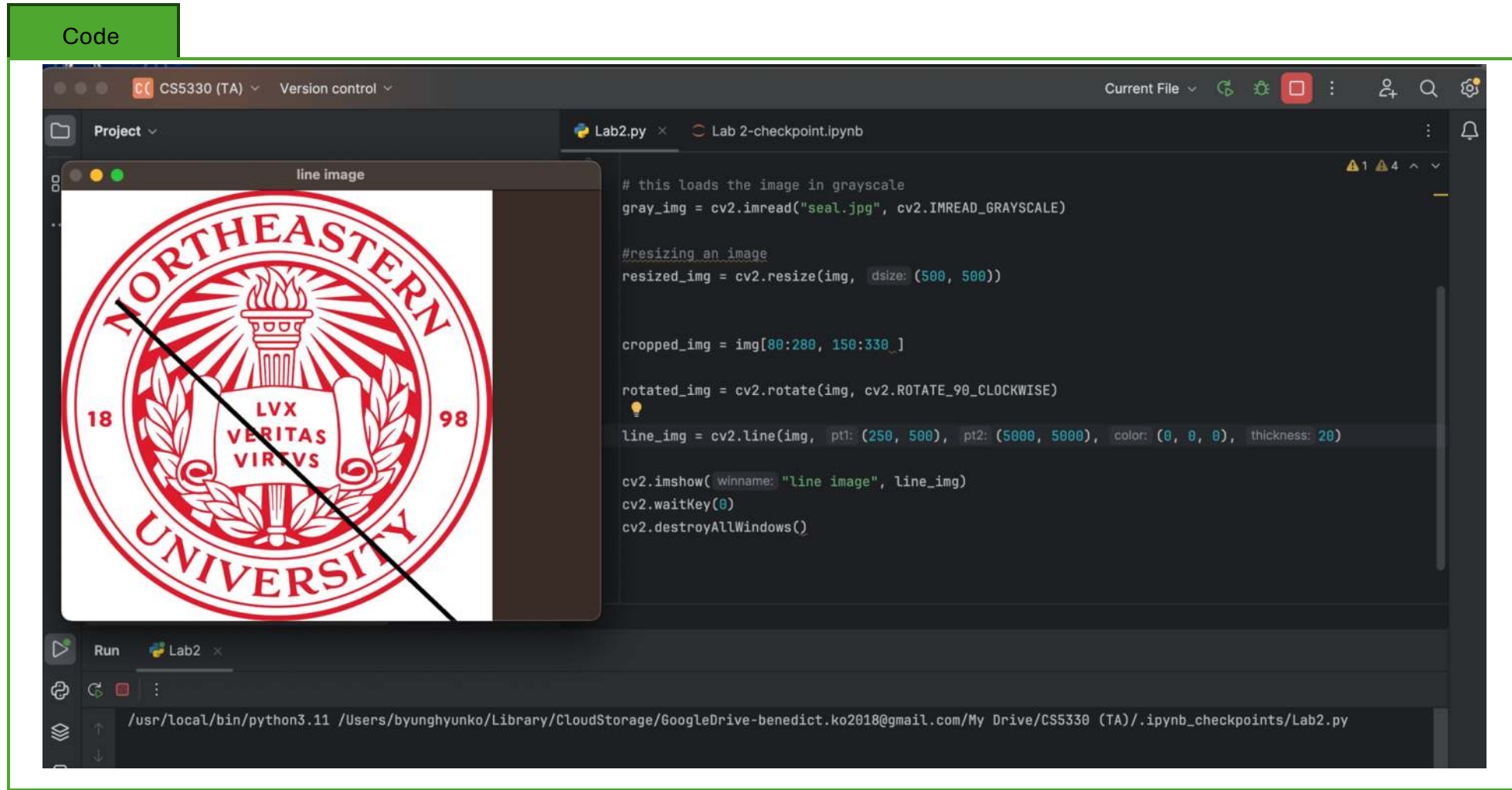
Image Colors Spaces and Conversions

- To convert an RGB image to Grayscale:
 - `gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)`
- To convert an RGB image to HSV:
 - `hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)`

Drawing Shapes and Text

- Drawing a line on an image
 - `cv2.line(image, (x1, y1), (x2, y2), (color), thickness)`
 - (x1, y1) are the starting coordinates of the line
 - (x2, y2) are the ending coordinates of the line
 - Color is the color of the line to be drawn in RGB
 - Thickness is the thickness of the line in px

Drawing Shapes and Text



- Drawing a line on an image (example)

Drawing Shapes and Text

- Drawing a rectangle:
 - `cv2.rectangle(image, (x1, y1), (x2, y2), (color), thickness)`
 - (x1, y1) is the starting coordinate
 - (x2, y2) is the ending coordinate
 - (color) is color in RGB
 - Thickness in thickness in px
- Drawing a circle:
 - `cv2.circle(image, (center x, center y), radius, (color), thickness)`
 - (center x, center y) is the center coordinate of the circle

Drawing Shapes and Text

- Adding a text to an image
 - `cv2.putText(image, 'text', (x, y), font, fontScale, (color), thickness)`
 - (x, y) is the coordinate of the **bottom-left** corner of the text string in the image
 - Font denotes the font type. (e.g. FONT_HERSHEY_SIMPLEX, FONT_HERSHEY_PLAIN)
 - fontScale is a font scale factor that is multiplied by the font specific base size
 - Color is the color of text string to be drawn in RGB
 - Thickness if the thickness of the line in px