

# OpenCV: Getting Started

---

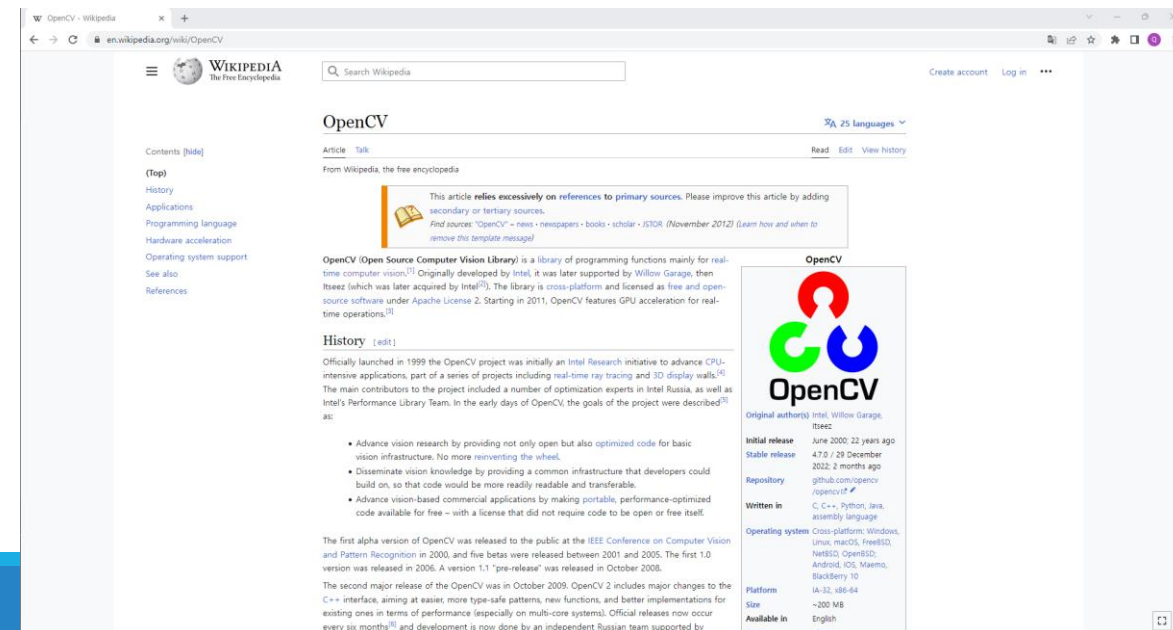
COMPUTER VISION (COURSE-HY23992)

Q YOUN HONG



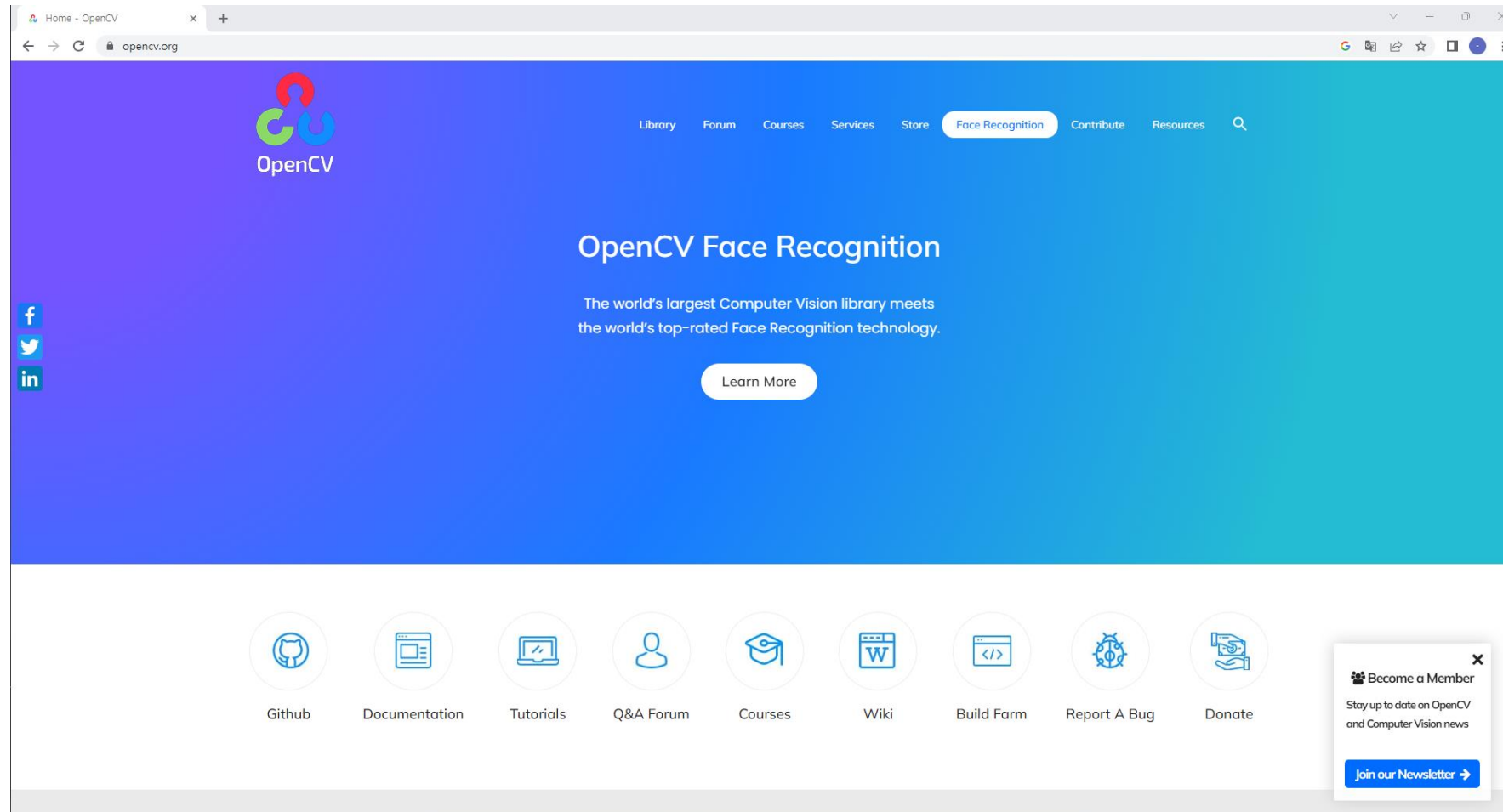
# OpenCV Overview

- OpenCV (Open Source Computer Vision Library)
  - A library including several hundreds of computer vision algorithms
  - Originally developed by Intel (June, 2000)
  - Current stable version: 4.9.0 (Dec. 2023)
  - Cross-platform, free and open-source software (Apache License 2)
    - Developed in C/C++
    - C, C++, Java, JavaScript, **Python** interfaces
    - Support Windows, macOS, Linux



# OpenCV Online Resources

- Official website: <https://opencv.org>



# OpenCV Online Resources

- OpenCV manuals: <https://docs.opencv.org>

The screenshot shows the OpenCV documentation website at <https://docs.opencv.org/4.9.0/>. The browser's address bar and tabs are visible at the top. The website header includes the OpenCV logo, the text "Open Source Computer vision", and a version selector dropdown menu currently set to "4.9.0". A red arrow points to this dropdown with the label "OpenCV version". Below the header is a navigation bar with tabs: "Main Page", "Related Pages", "Modules", "Namespaces", "Classes", "Files", "Examples", and "Java documentation". A search bar is located on the right side of this navigation bar. The main content area is titled "OpenCV modules" and contains a list of links. A red bracket on the right side of the list groups the links under the label "Modules". Another red bracket on the left side of the list groups the links under the label "OpenCV Python tutorials".

OpenCV version

OpenCV Python tutorials

Modules

- Introduction
- OpenCV Tutorials
- OpenCV-Python Tutorials
- OpenCV.js Tutorials
- Tutorials for contrib modules
- Frequently Asked Questions
- Bibliography
- Main modules:
  - core. Core functionality
  - imgproc. Image Processing
  - imgcodecs. Image file reading and writing
  - video. Video I/O
  - highgui. High-level GUI
  - video. Video Analysis
  - calib3d. Camera Calibration and 3D Reconstruction
  - features2d. 2D Features Framework
  - objdetect. Object Detection
  - dnn. Deep Neural Network module
  - ml. Machine Learning
  - flann. Clustering and Search in Multi-Dimensional Spaces
  - photo. Computational Photography
  - stitching. Images stitching
  - gapi. Graph API
- Extra modules:
  - alphasat. Alpha Matting
  - aruco. Aruco markers, module functionality was moved to objdetect module
  - bgsegm. Improved Background-Foreground Segmentation Methods
  - bioinspired. Biologically inspired vision models and derived tools
  - cannops. Operations for Ascend Backend.
  - ccalib. Custom Calibration Pattern for 3D reconstruction
  - cudaarithm. Operations on Matrices
  - cudahoseom. Background Segmentation

# OpenCV Online Resources

---

- GitHub: <https://github.com/opencv/opencv>
- Other resources
  - <https://opencv.org/resources/>
  - <https://cafe.naver.com/opencv> (in Korean)

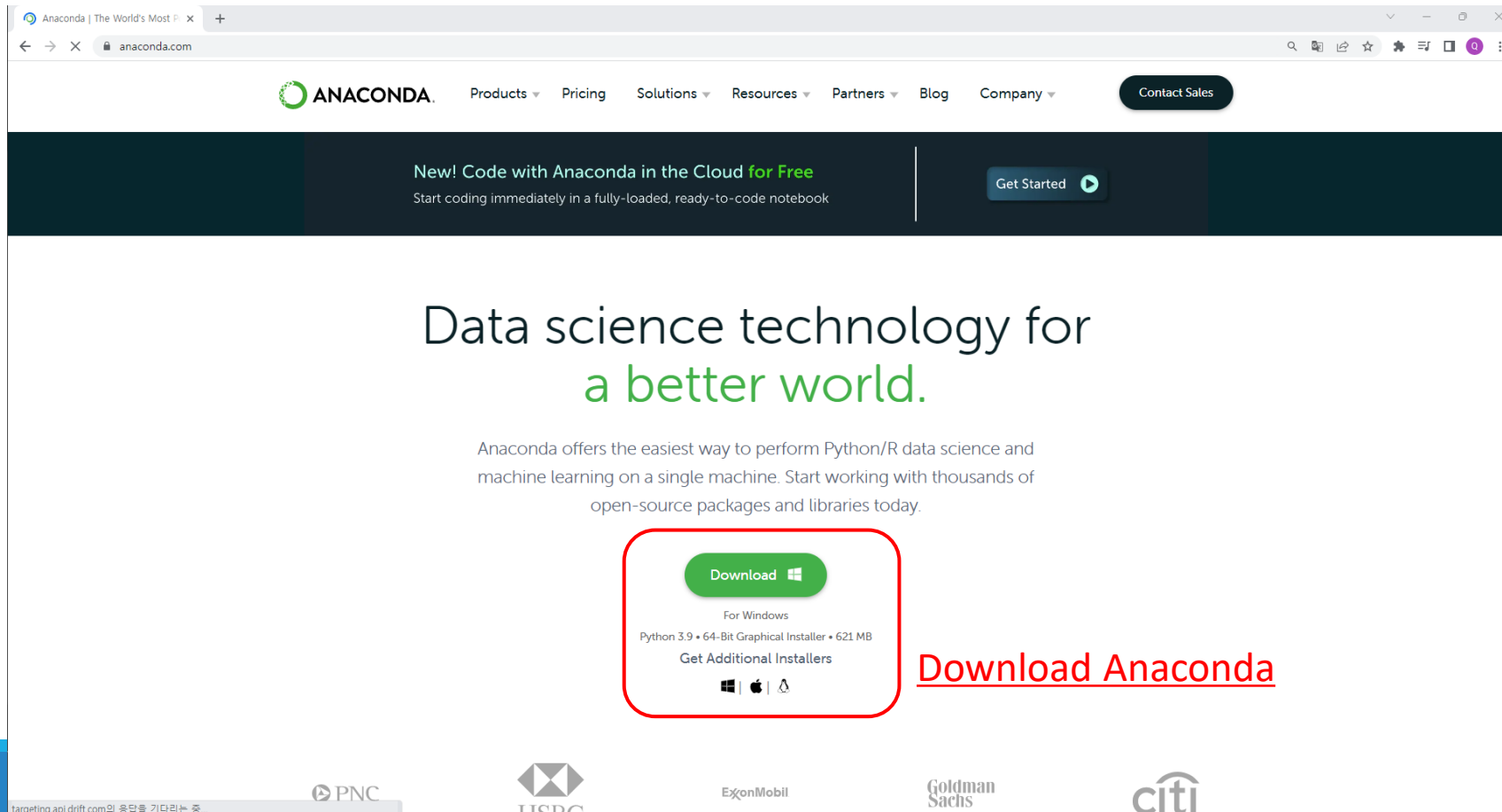
# Setting Up Development Environment

---

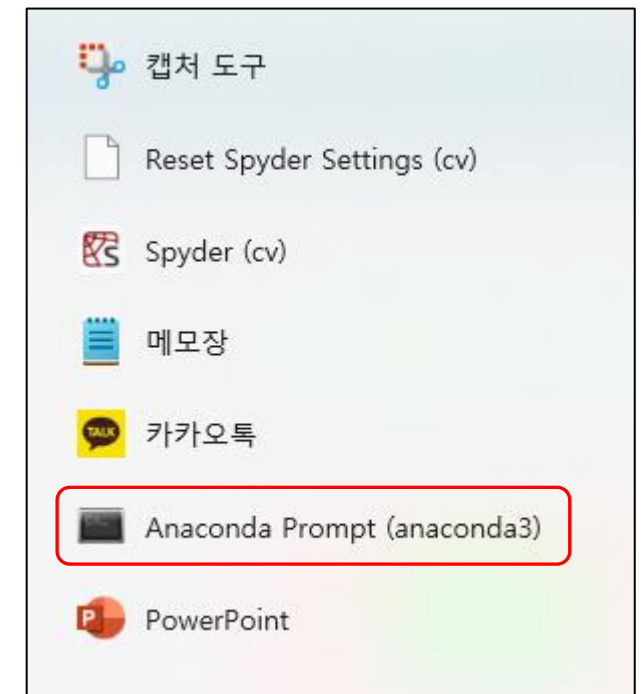
- Install Python Compiler
- Install IDE (Integrated Development Environment)
  - PyCharm, Jupyter, Spyder
- Libraries
  - OpenCV (+ NumPy, Matplotlib, PyQt)
  - (Optional) TensorFlow....

# Installing Anaconda

- Anaconda (<https://anaconda.com>)
  - Create an integrated environment to manage Python and packages
  - Provide a virtual environment to use packages of different version



[Download Anaconda](#)



# Setting Up Development Environment

- Step 1: install anaconda
- Step 2: create a virtual environment
- Step 3: install IDE (Spyder) and OpenCV

```
(base) C:\Users\HYU> conda create -c conda-forge  
-n cv spyder  
(base) C:\Users\HYU> conda activate cv  
(cv) C:\Users\HYU> pip install opencv-python  
(cv) C:\Users\HYU> conda list  
(cv) C:\Users\HYU> spyder
```

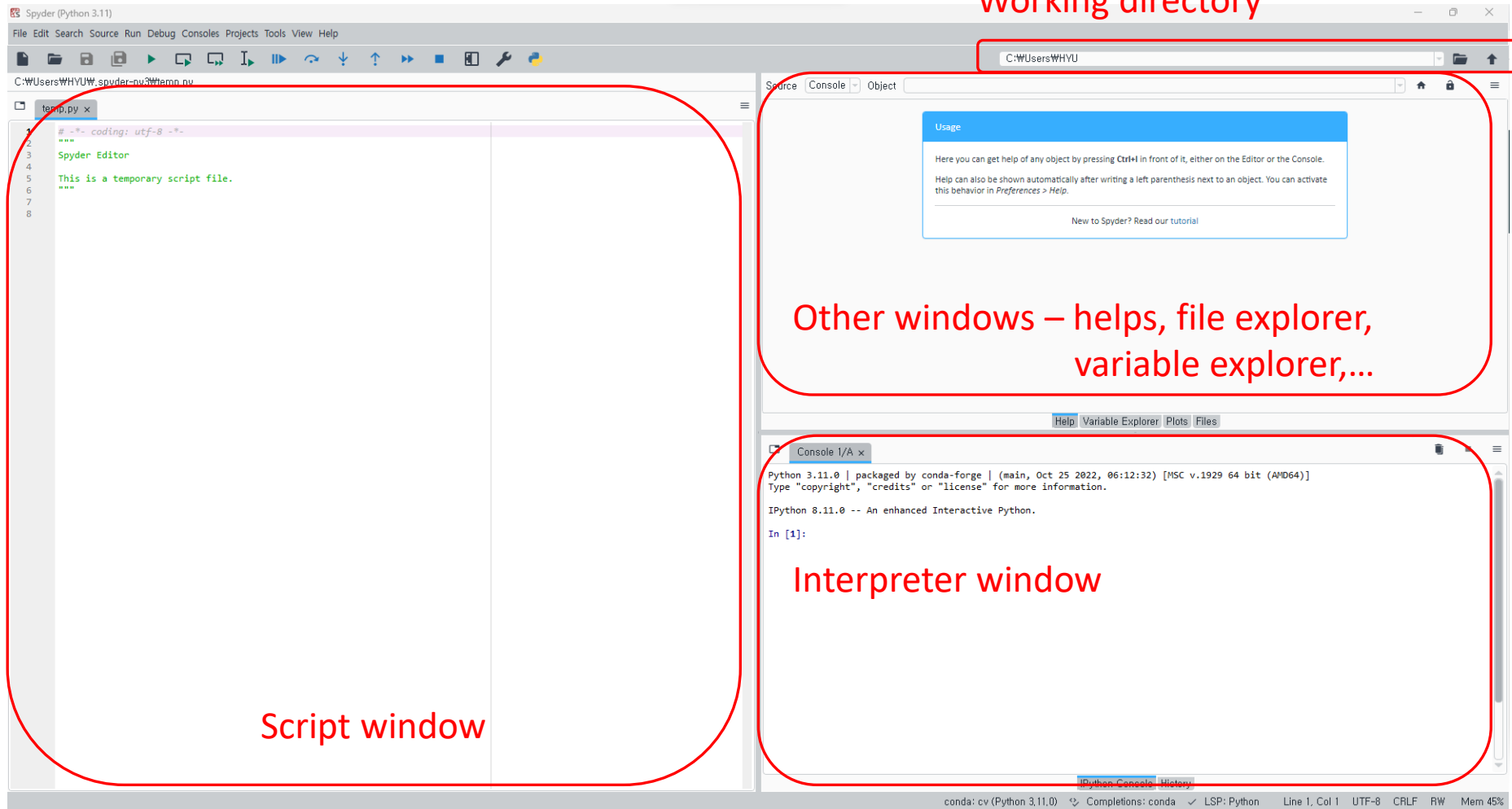
1. Create a new virtual environment 'cv' and install Spyder (IDE) in 'cv'
2. Switch the base environment to 'cv'
3. Install OpenCV
4. List the packages in 'cv'
5. Launch Spyder in 'cv'

Anaconda Prompt (anaconda3)			
nbconvert	7.2.9	pyhd8ed1ab_0	conda-forge
nbconvert-core	7.2.9	pyhd8ed1ab_0	conda-forge
nbconvert-pandoc	7.2.9	pyhd8ed1ab_0	conda-forge
nbformat	5.7.3	pyhd8ed1ab_0	conda-forge
nest-asyncio	1.5.6	pyhd8ed1ab_0	conda-forge
numpy	1.24.2	pypi_0	pypi
numpydoc	1.5.0	pyhd8ed1ab_0	conda-forge
opencv-python	4.7.0.72	pypi_0	pypi
openssl	3.0.8	hcfcfb64_0	conda-forge
packaging	23.0	pyhd8ed1ab_0	conda-forge

sphinxcontrib-qthelp	1.0.3	py_0	conda-forge
sphinxcontrib-serializinghtml	1.1.5	pyhd8ed1ab_2	conda-forge
<u>spyder</u>	5.4.2	py311h1ea47a8_0	conda-forge
<u>spyder-kernels</u>	2.4.2	win_pyhd8ed1ab_0	conda-forge
<u>stack_data</u>	0.6.2	pyhd8ed1ab_0	conda-forge
text-unidecode	1.3	py_0	conda-forge
textdistance	4.5.0	pyhd8ed1ab_0	conda-forge
three-merge	0.1.1	pyh9f0ad1d_0	conda-forge
tinycss2	1.2.1	pyhd8ed1ab_0	conda-forge
tk	8.6.12	h8ffe710_0	conda-forge
toml	0.10.2	pyhd8ed1ab_0	conda-forge
tomli	2.0.1	pyhd8ed1ab_0	conda-forge



# Let's Launch Spyder!



# OpenCV: Getting Started!

- Example 1: read an image file and display the image

```
example1.py x
1  import cv2 as cv
2  import sys
3
4  img = cv.imread('Erica.jpg')
5
6  if img is None:
7      sys.exit('No file found')
8
9  cv.imshow('HY24011', img)
10
11  cv.waitKey()
12  cv.destroyAllWindows()
```



```
Console 1/A x
Python 3.11.0 | packaged by conda-forge | (main, Oct 25 2022, 06:12:32) [MSC v.1929 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.11.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/CV2024-Spring/practice1/src/example1.py', wdir='D:/CV2024-Spring/practice1/src')
```

# OpenCV: Getting Started!

- Example 1: read an image file and display the image



```
example1.py x
1  import cv2 as cv
2  import sys
3
4  img = cv.imread('Erica.jpg')
5
6  if img is None:
7      sys.exit('No file found')
8
9  cv.imshow('HY24011', img)
10
11 cv.waitKey()
12 cv.destroyAllWindows()
```

(L1) import opencv module and bind it to 'cv'

(L2) to use system calls (i.e. exit())

(L4) read an image from an image file and store to 'img'

(L9) display 'img' on the screen

# Image in OpenCV

---

- An image in OpenCV: a `numpy.ndarray` object
  - Numpy: a package including mathematical functions, random number generators, linear algebra routines, etc. (<https://numpy.org>)
  - `numpy.ndarray`: N-dimensional array object
  - Can use member functions of `numpy.ndarray`
    - `'dir(img)'` to list all member functions

```
In [2]: type(img)
Out[2]: numpy.ndarray

In [3]: img.shape
Out[3]: (600, 800, 3)
```

```
In [7]: help(img.trace)
Help on built-in function trace:

trace(...) method of numpy.ndarray instance
    a.trace(offset=0, axis1=0, axis2=1, dtype=None, out=None)

    Return the sum along diagonals of the array.

    Refer to `numpy.trace` for full documentation.

    See Also
    -----
    numpy.trace : equivalent function
```

# Image in OpenCV

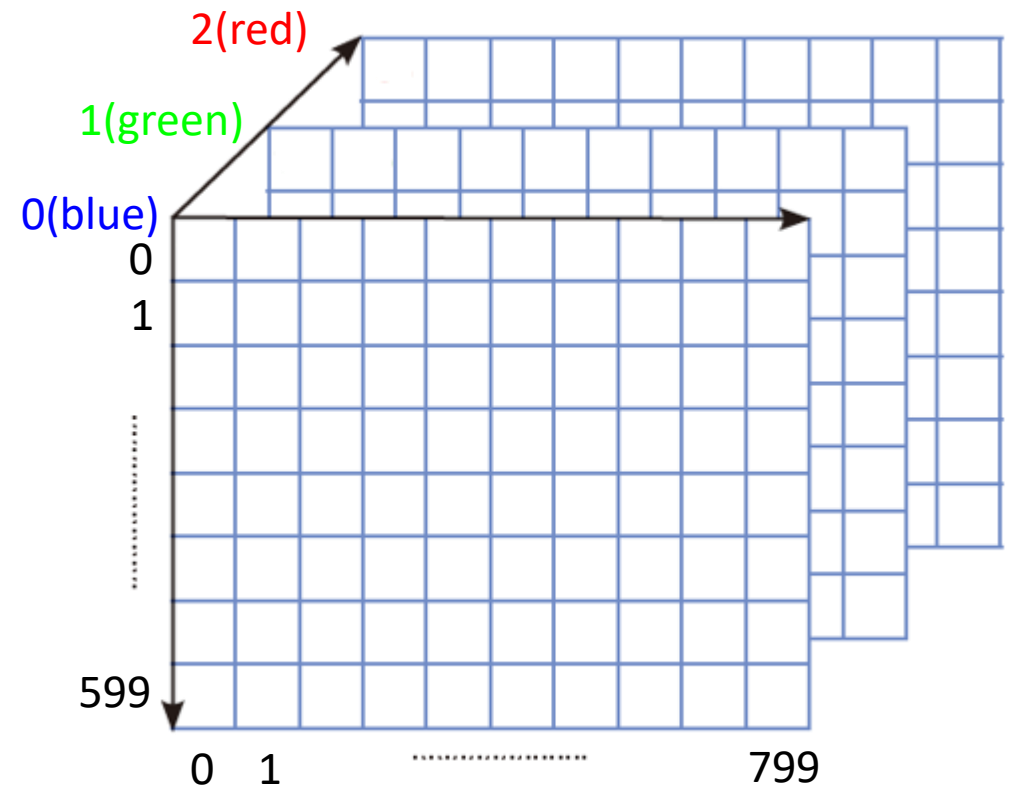
- An image in OpenCV: a 3-dimensional array object
  - (Number of rows) x (Number of columns) x (Number of color channels)
  - An image has #(rows) x #(columns) pixels
  - A pixel is represented as (r, c) = (y, x)
  - A pixel has blue, green, red values

```
In [10]: print(img[0,0])  
[248 251 255]
```

```
In [11]: print(img[599,0])  
[ 84  95 127]
```

```
In [12]: print(img[599,799])  
[ 87  99 133]
```

```
In [13]: print(img[0,799])  
[226 226 226]
```





# Simple Image Processing

Example 2: convert a color image to a grayscale image and resize the grayscale image

```
example2.py X
1 import cv2 as cv
2 import sys
3
4 img = cv.imread('Erica.jpg')
5
6 if img is None:
7     print('No file found')
8
9 gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
10 gray_small = cv.resize(gray, dsize=(0,0), fx=0.5, fy=0.5)
11
12 cv.imwrite('Erica_gray.jpg', gray)
13 cv.imwrite('Erica_grayscale.jpg', gray_small)
14
15 cv.imshow('Color Image', img)
16 cv.imshow('Gray Image', gray)
17 cv.imshow('Gray Small Image', gray_small)
18
19 cv.waitKey()
20 cv.destroyAllWindows()
```



# Simple Image Processing

---

```
example2.py X
1  import cv2 as cv
2  import sys
3
4  img = cv.imread('Erica.jpg')
5
6  if img is None:
7      print('No file found')
8
9  gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
10 gray_small = cv.resize(gray, dsize=(0,0), fx=0.5, fy=0.5)
11
12 cv.imwrite('Erica_gray.jpg', gray)
13 cv.imwrite('Erica_grayscale.jpg', gray_small)
14
15 cv.imshow('Color Image', img)
16 cv.imshow('Gray Image', gray)
17 cv.imshow('Gray Small Image', gray_small)
18
19 cv.waitKey()
20 cv.destroyAllWindows()
```

(L9) change color-space of the image:

- `cvtColor()`: a function to convert color space of an image
- `COLOR_BGR2GRAY`: convert the color space of img from BGR model to Grayscale model

❖ Color space: will be covered later in the class

❖ Color spaces covered in OpenCV

[https://docs.opencv.org/4.x/d8/d01/group\\_imgproc\\_color\\_conversions.html](https://docs.opencv.org/4.x/d8/d01/group_imgproc_color_conversions.html)

(L12~L13) save the processed images to the image files

- `Imwrite()`

# Simple Image Processing

---

```
example2.py X
1  import cv2 as cv
2  import sys
3
4  img = cv.imread('Erica.jpg')
5
6  if img is None:
7      print('No file found')
8
9  gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
10 gray_small = cv.resize(gray, dsize=(0,0), fx=0.5, fy=0.5)
11
12 cv.imwrite('Erica_gray.jpg', gray)
13 cv.imwrite('Erica_graysmall.jpg', gray_small)
14
15 cv.imshow('Color Image', img)
16 cv.imshow('Gray Image', gray)
17 cv.imshow('Gray Small Image', gray_small)
18
19 cv.waitKey()
20 cv.destroyAllWindows()
```

(L10) resize a source image (down/up) to the specified size

- dsize: output image size. (0,0) if fx, fy are used instead
- fx, fy: scale factor along horizontal/vertical axes

(Question)

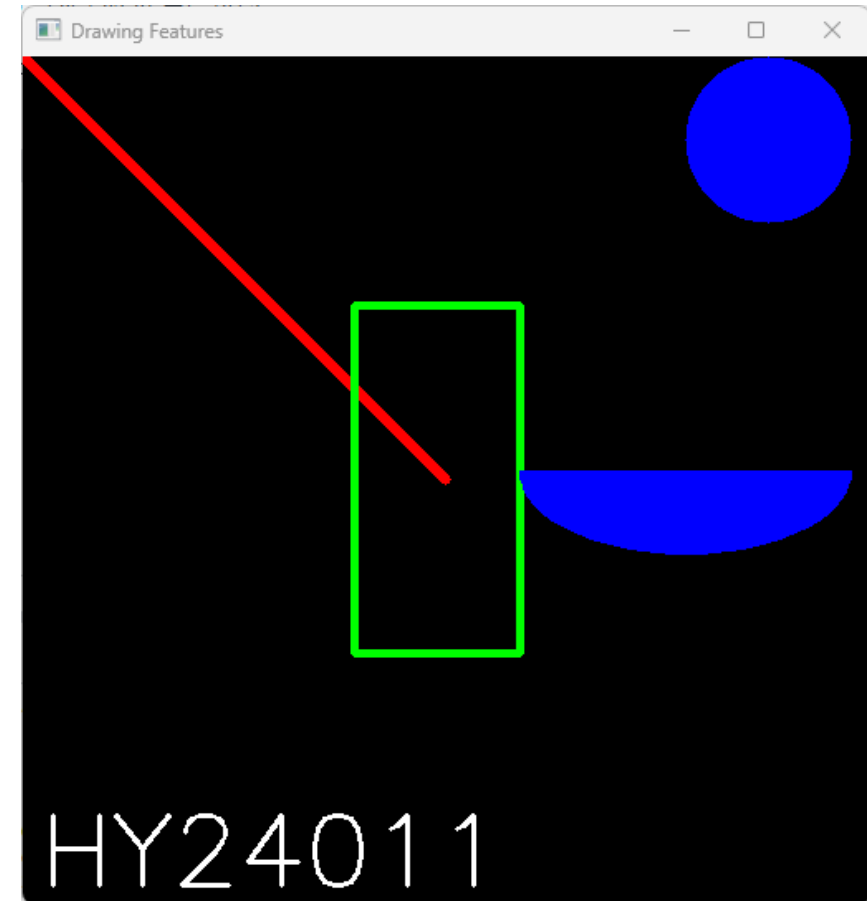
How does the image look like if fx and fy are not the same?



# Example3: Drawing Functions

- Draw geometric primitives on an image

```
example3.py x
1  import numpy as np
2  import cv2 as cv
3
4  img = np.zeros((512, 512, 3), np.uint8)
5
6  cv.line(img, (0, 0), (255, 255), (0, 0, 255), 5)
7  cv.rectangle(img, (200, 150), (300, 360), (0, 255, 0), 3)
8  cv.circle(img, (450, 50), 50, (255, 0, 0), -1)
9  cv.ellipse(img, (400, 250), (100, 50), 0, 0, 180, 255, -1)
10
11 cv.putText(img, 'HY24011', (10, 500), cv.FONT_HERSHEY_SIMPLEX,
12           2, (255, 255, 255), 2)
13
14 cv.imshow('Drawing Features', img)
15
16 cv.waitKey()
17 cv.destroyAllWindows()
```



# Drawing Functions

example3.py x

```
1 import numpy as np
2 import cv2 as cv
3
4 img = np.zeros((512, 512, 3), np.uint8)
5
6 cv.line(img, (0, 0), (255, 255), (0, 0, 255), 5)
7 cv.rectangle(img, (200, 150), (300, 360), (0, 255, 0), 3)
8 cv.circle(img, (450, 50), 50, (255, 0, 0), -1)
9 cv.ellipse(img, (400, 250), (100, 50), 0, 0, 180, 255, -1)
10
11 cv.putText(img, 'HY24011', (10, 500), cv.FONT_HERSHEY_SIMPLEX,
12           2, (255, 255, 255), 2)
13
14 cv.imshow('Drawing Features', img)
15
16 cv.waitKey()
17 cv.destroyAllWindows()
```

(L4) Create an new image in black (0, 0, 0)

(L6) cv.line(): Draw a line on the image

- From (x,y) = (0, 0) to (255, 255) in red (0, 0, 255) with thickness = 5

(L7) cv.rectangle(): Draw a box on the image

- (200, 150): upper left corner vertex of a box
- (300, 360): lower right corner vertex of a box
- Draw a box in green (0,255,0) with thickness = 3

(L8) cv.circle(): Draw a circle on the image

- (450, 50): the center point of a circle
- Radius = 50
- Fill the interior of a circle (-1)

(L9) cv.ellipse(): Draw an ellipse on the image

- (400, 250): the center point of an ellipse
- (100, 50): half of the sizes of the main axes of an ellipse
- 0: rotate an ellipse in 0 degrees
- 0, 180: draw an ellipse from 0 to 180 degrees
- Fill the interior of an ellipse (-1)

# Drawing Functions

---

example3.py x

```
1  import numpy as np
2  import cv2 as cv
3
4  img = np.zeros((512, 512, 3), np.uint8)
5
6  cv.line(img, (0, 0), (255, 255), (0, 0, 255), 5)
7  cv.rectangle(img, (200, 150), (300, 360), (0, 255, 0), 3)
8  cv.circle(img, (450, 50), 50, (255, 0, 0), -1)
9  cv.ellipse(img, (400, 250), (100, 50), 0, 0, 180, 255, -1)
10
11 cv.putText(img, 'HY24011', (10, 500), cv.FONT_HERSHEY_SIMPLEX,
12           2, (255, 255, 255), 2)
13
14 cv.imshow('Drawing Features', img)
15
16 cv.waitKey()
17 cv.destroyAllWindows()
```

(L11) cv.putText(): write 'HY24011' on the image

- (10, 500): the bottom left corner point of the text string
- cv.FONT\_HERSHEY\_SIMPLEX: font type
- 2: Scale(size) of the text
- Write the text in white (255, 255, 255)
- 2: Thickness of the text

❖ All points in drawing functions are given in 'Rect' type: (x,y)

❖ For more details:

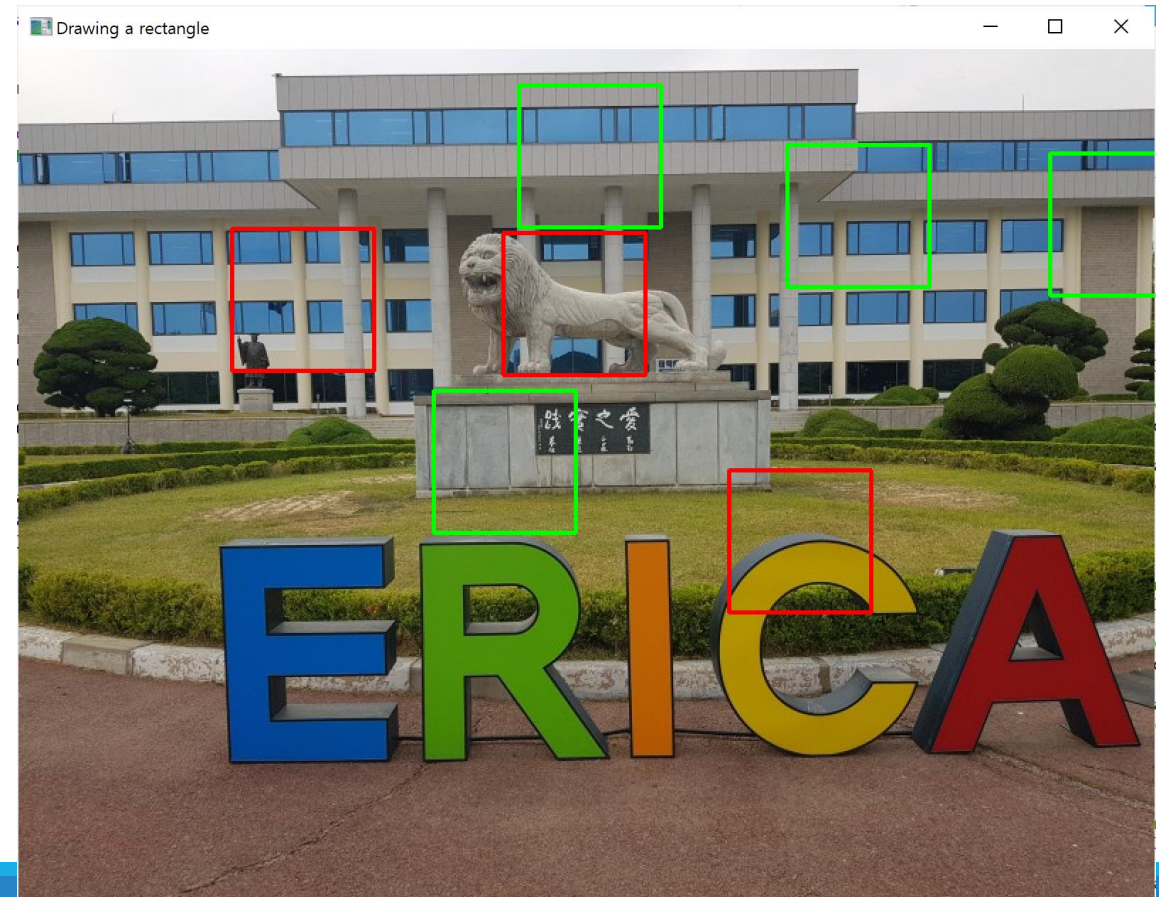
[https://docs.opencv.org/4.x/d6/d6e/group\\_\\_imgproc\\_\\_draw.html](https://docs.opencv.org/4.x/d6/d6e/group__imgproc__draw.html)

# Example 4: GUI Features

- GUI (Graphical User Interface) enables users to interact with opencv applications using input/output devices
- Ex4-1: Draw boxes on an image when clicking mouse buttons

example4.py X

```
1 import cv2 as cv
2 import sys
3
4 img = cv.imread('Erica.jpg')
5
6 if img is None:
7     print('No file found')
8
9
10 # mouse callback function
11 def draw(event,x,y,flags,param):
12     if event == cv.EVENT_LBUTTONDOWN:
13         cv.rectangle(img,(x,y),(x+100,y+100),(0,0,255),2)
14     elif event == cv.EVENT_RBUTTONDOWN:
15         cv.rectangle(img,(x,y),(x+100,y+100),(0,255,0),2)
16     cv.imshow('Drawing a rectangle', img)
17
18 cv.namedWindow('Drawing a rectangle')
19 cv.setMouseCallback('Drawing a rectangle',draw)
20
21 while(True):
22     if cv.waitKey(1)==ord('q'):
23         break
24 cv.destroyAllWindows()
```



# Handling Mouse Events

```
example4.py X
1 import cv2 as cv
2 import sys
3
4 img = cv.imread('Erica.jpg')
5
6 if img is None:
7     print('No file found')
8
9
10 # mouse callback function
11 def draw(event,x,y,flags,param):
12     if event == cv.EVENT_LBUTTONDOWN:
13         cv.rectangle(img,(x,y),(x+100,y+100),(0,0,255),2)
14     elif event == cv.EVENT_RBUTTONDOWN:
15         cv.rectangle(img,(x,y),(x+100,y+100),(0,255,0),2)
16     cv.imshow('Drawing a rectangle', img)
17
18 cv.namedWindow('Drawing a rectangle')
19 cv.setMouseCallback('Drawing a rectangle',draw)
20
21 while(True):
22     if cv.waitKey(1)==ord('q'):
23         break
24 cv.destroyAllWindows()
```

- ❖ Handle events in OpenCV: via callback functions
- ❖ Identify a window with the title of the window (cv.namedWindow())

(L10-L16) Define a callback function :

- The function handles left and right click events
- Draw a red square when clicking left mouse button
- Draw a blue square when clicking right mouse button

(L19) Register a callback function to handle mouse events

(L21-23) Execute an infinite while loop:

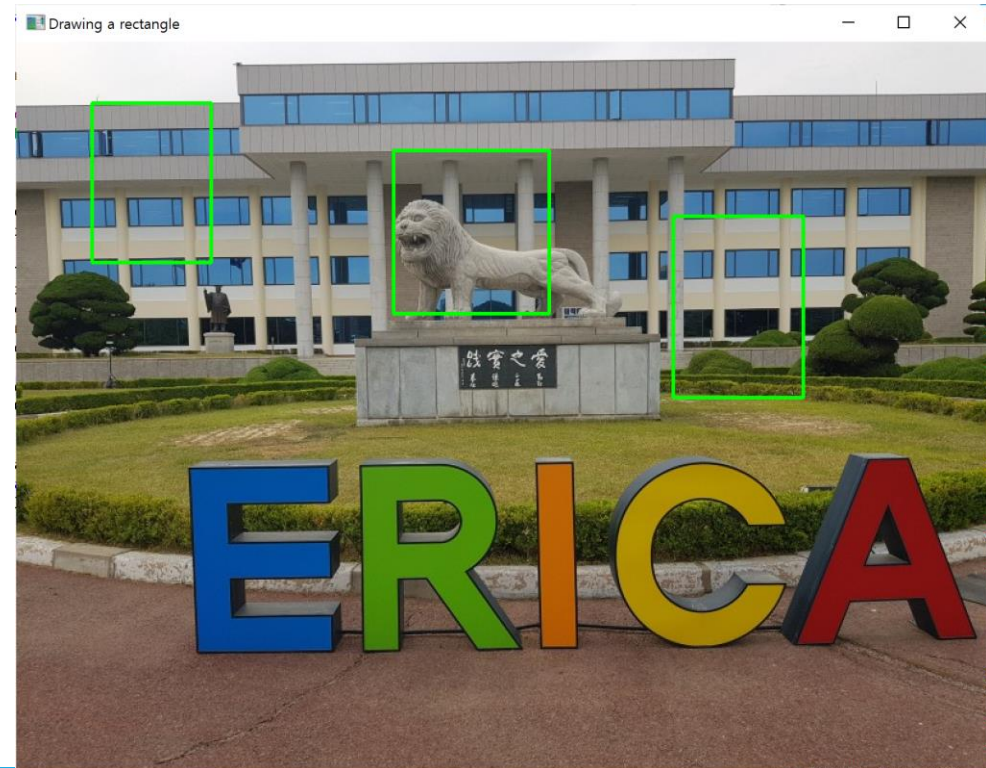
Whenever mouse events happen, call draw()



# Ex4-2. Handling Mouse Dragging Events

- ❖ Change the size of boxes while pressing a left mouse button
  - Save the initial position of a box when clicking down a mouse, but do not draw anything
  - Draw a box with the saved position and the current position when clicking up a mouse

```
1 import cv2 as cv
2 import sys
3
4 img = cv.imread('Erica.jpg')
5
6 if img is None:
7     print('No file found')
8
9
10 # mouse callback function
11 def draw(event,x,y,flags,param):
12     global ix, iy;
13
14     if event == cv.EVENT_LBUTTONDOWN:
15         ix,iy=x,y
16     elif event == cv.EVENT_LBUTTONUP:
17         cv.rectangle(img,(ix,iy),(x,y),(0,255,0),2)
18         cv.imshow('Drawing a rectangle', img)
19
20 cv.namedWindow('Drawing a rectangle')
21 cv.setMouseCallback('Drawing a rectangle',draw)
22
23 while(True):
24     if cv.waitKey(1)==ord('q'):
25         break
26 cv.destroyAllWindows()
```



# Ex4-3: Paint Brush

- ❖ Draw circles not only in pressing mouse buttons, but in moving mouse (L16-L20) When a mousemove event is detected, the event also saves which button is pressed in the 'flags' parameter

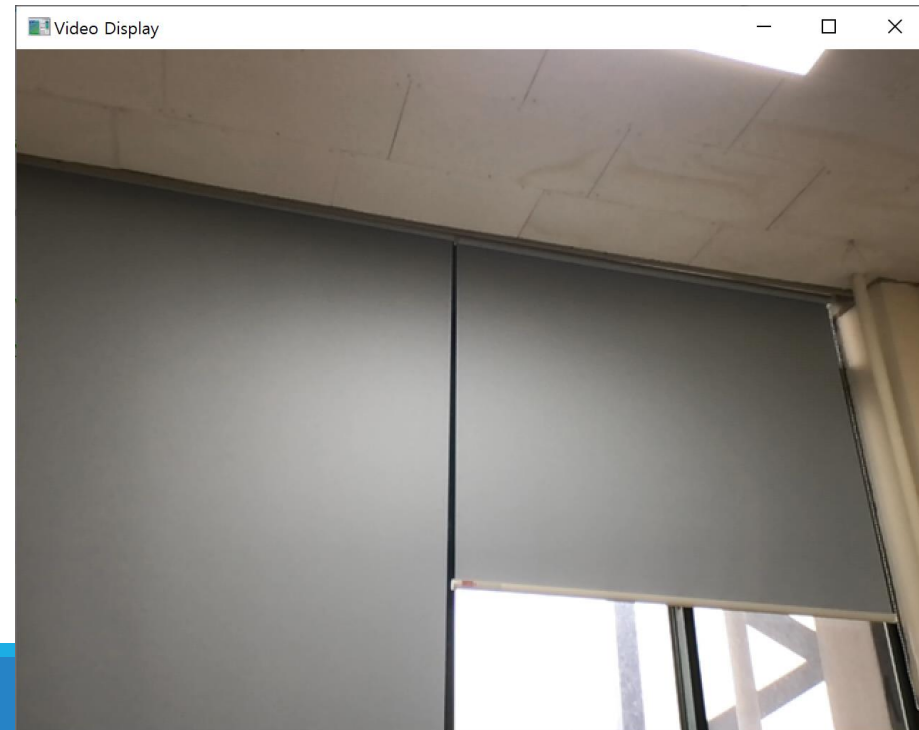
```
1 import cv2 as cv
2
3 img = cv.imread('Erica.jpg')
4
5 if img is None:
6     print('No file found')
7
8 BrushSize=5
9 LColor,RColor=(255,0,0),(0,0,255)
10
11 def painting(event,x,y,flags,param):
12     if event==cv.EVENT_LBUTTONDOWN:
13         cv.circle(img,(x,y),BrushSize,LColor,-1)
14     elif event==cv.EVENT_RBUTTONDOWN:
15         cv.circle(img,(x,y),BrushSize,RColor,-1)
16     elif event==cv.EVENT_MOUSEMOVE:
17         if flags==cv.EVENT_FLAG_LBUTTON:
18             cv.circle(img,(x,y),BrushSize,LColor,-1)
19         elif flags==cv.EVENT_FLAG_RBUTTON:
20             cv.circle(img,(x,y),BrushSize,RColor,-1)
21     cv.imshow('Painting', img)
22
23 cv.namedWindow('Painting')
24 cv.imshow('Painting', img)
25 cv.setMouseCallback('Painting', painting)
26
27 while(True):
28     if cv.waitKey(1)==ord('q'):
29         break
30 cv.destroyAllWindows()
```



# (Example\_optional) Video Capture

```
1 import cv2 as cv
2 import sys
3
4 cap = cv.VideoCapture(0, cv.CAP_DSHOW)
5 if not cap.isOpened():
6     sys.exit('Cannot open camera')
7
8 while True:
9     # Capture frame-by-frame
10    ret, frame = cap.read()
11
12    if not ret:
13        print('Cannot receive frame')
14        break
15    cv.imshow('Video Display', frame)
16
17    key=cv.waitKey(1)
18    if key==ord('q'):
19        break
20
21 # release the capture
22 cap.release()
23 cv.destroyAllWindows()
```

- ❖ Capture live stream with a camera and display on the screen
- (L4) `cv.VideoCapture()`: create a 'VideoCapture' object
- (L22) `cap.release()`: release(disconnect) the camera after use
- (L8-L20) Capture frame-by-frame images
  - (L10) Capture the current frame to 'frame'
  - (L12~L14) If failed to capture, halt
  - (L15) Display the captured frame





# Review Task!

- Create an image as follows:



Write your student ID here

## [Instruction]

- Grayscale image
- Draw a rectangle around 'ERICA' in magenta
- Write your own student ID below in green
- Save the new image to 'jpg' file

## [Submission]

Submit your python file and jpg file to LMS

## [Hint]

[https://docs.opencv.org/4.x/de/d25/imgproc\\_color\\_conversions.html](https://docs.opencv.org/4.x/de/d25/imgproc_color_conversions.html)

You might need 2 color conversions