

Basic Python Programming

[Session 3] OpenCV

Contents

- **Backgrounds**
- **Image Processing**
- **Dealing with Videos**
- **Facial Recognition**

Intro & Preparation

OpenCV

- **An open-source Computer Vision library**
- **It supports..**
 - 2D / 3D image processing
 - Facial / gesture / object recognition/detection
 - ML
 - AR
 - So on...
- **We can use it with many languages**
 - C/C++/Python/Java/Objective-C/...
 - Of course, we will use Python

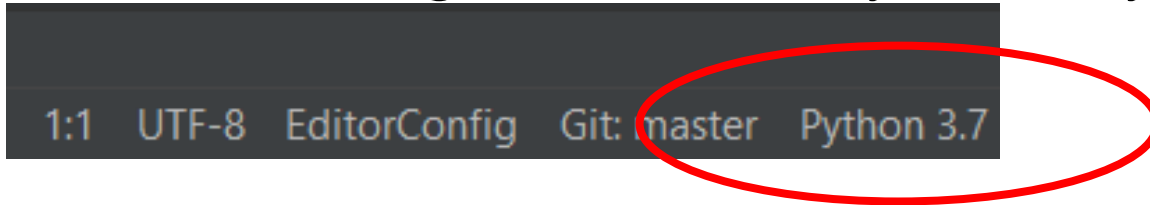


About Today's Class...

- **We will learn by writing code.. but some backgrounds are needed**
- **OpenCV must be available in your environment!!**
 - Before the lab session, please make sure this

Required Environment (IMPORTANT)

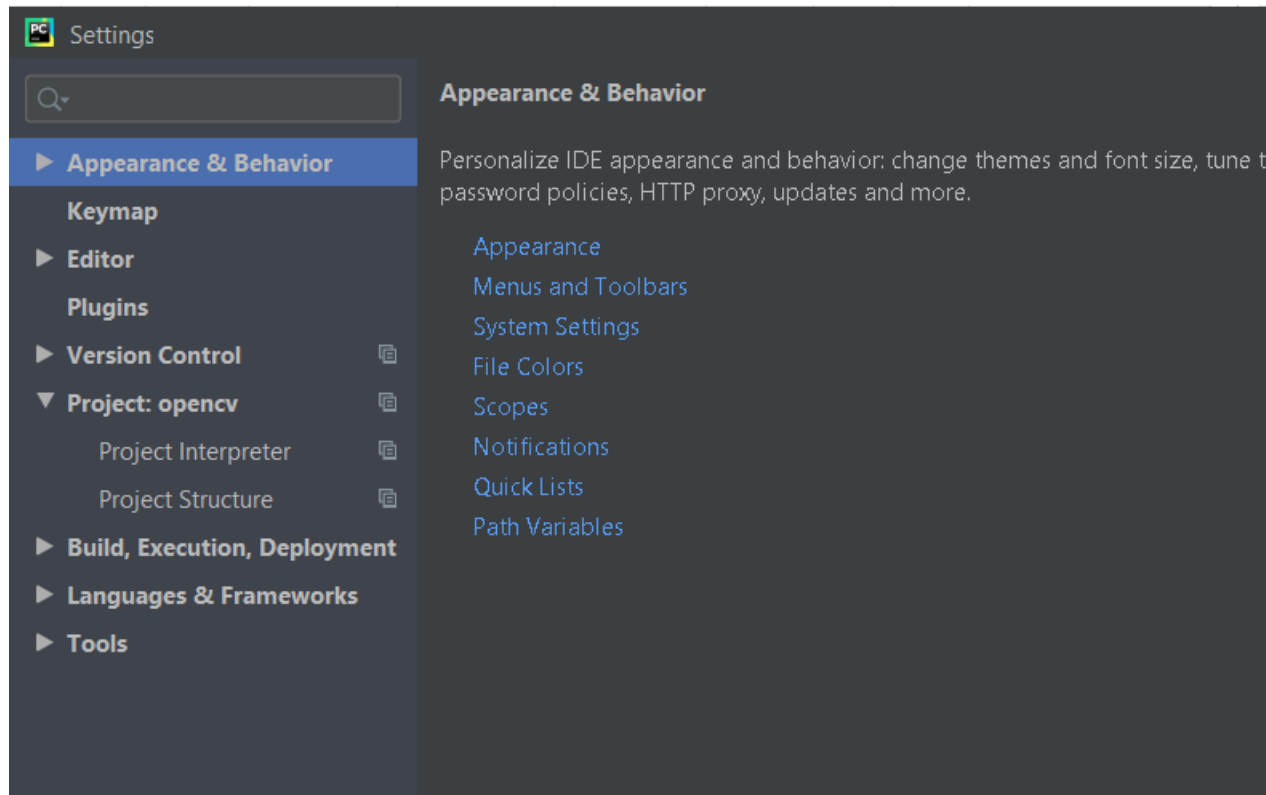
- As mentioned in session 1,2:
 - Python 3.7.8: over 3.8 may not supports OpenCV
 - In the lower-right corner of PyCharm, you can see that



- OpenCV is cross-platform library, so OS doesn't matter
- Perhaps, you don't need to worry about system requirements
 - I think your RAM may be greater than 1GB...
- If you cannot sure about your environment, please ask us
 - With your device / system specifications

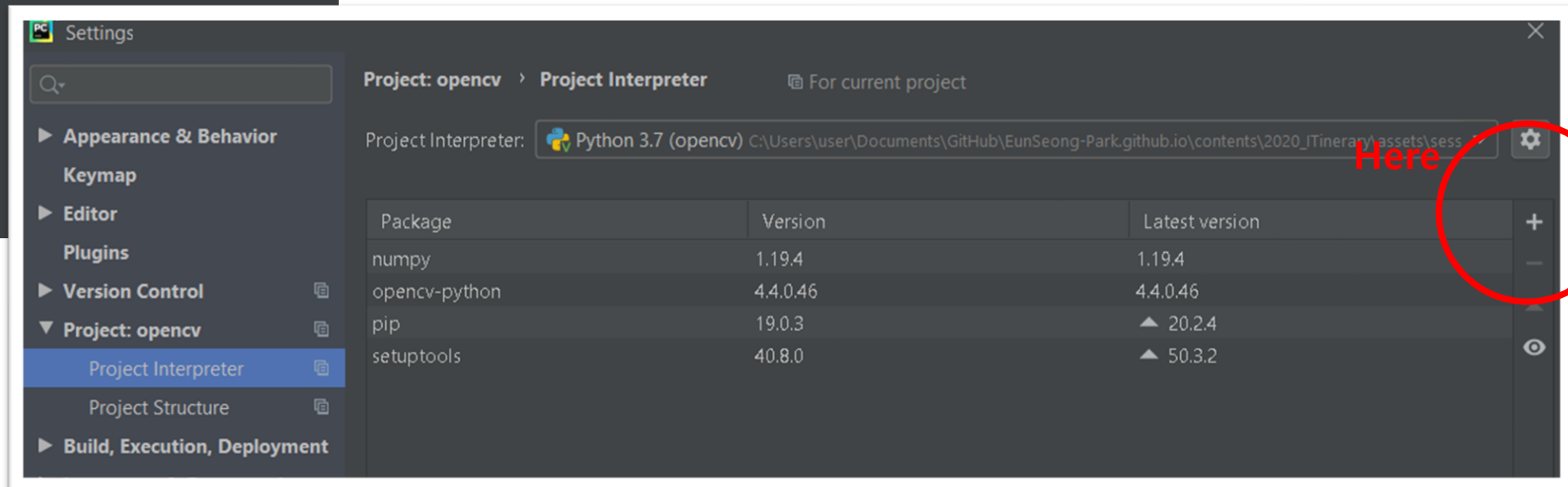
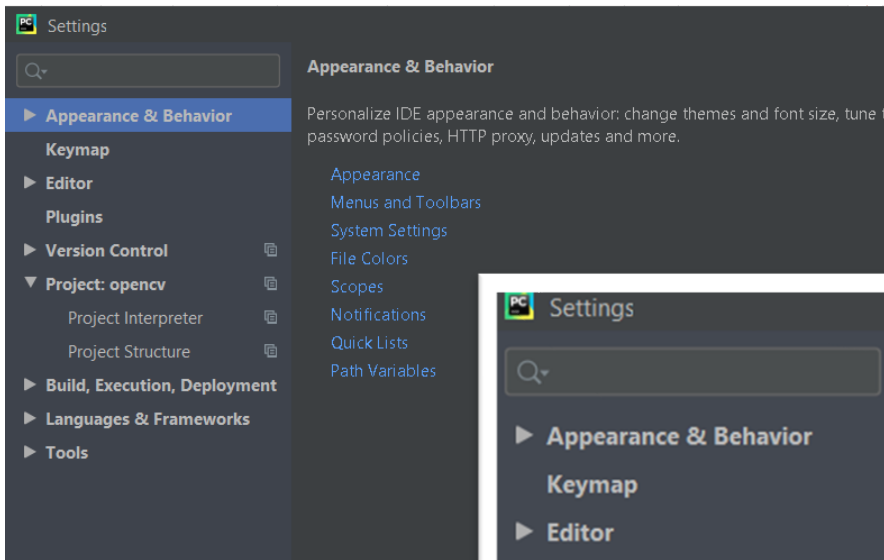
Installing OpenCV [1]

- In PyCharm, go to [file]>[settings]



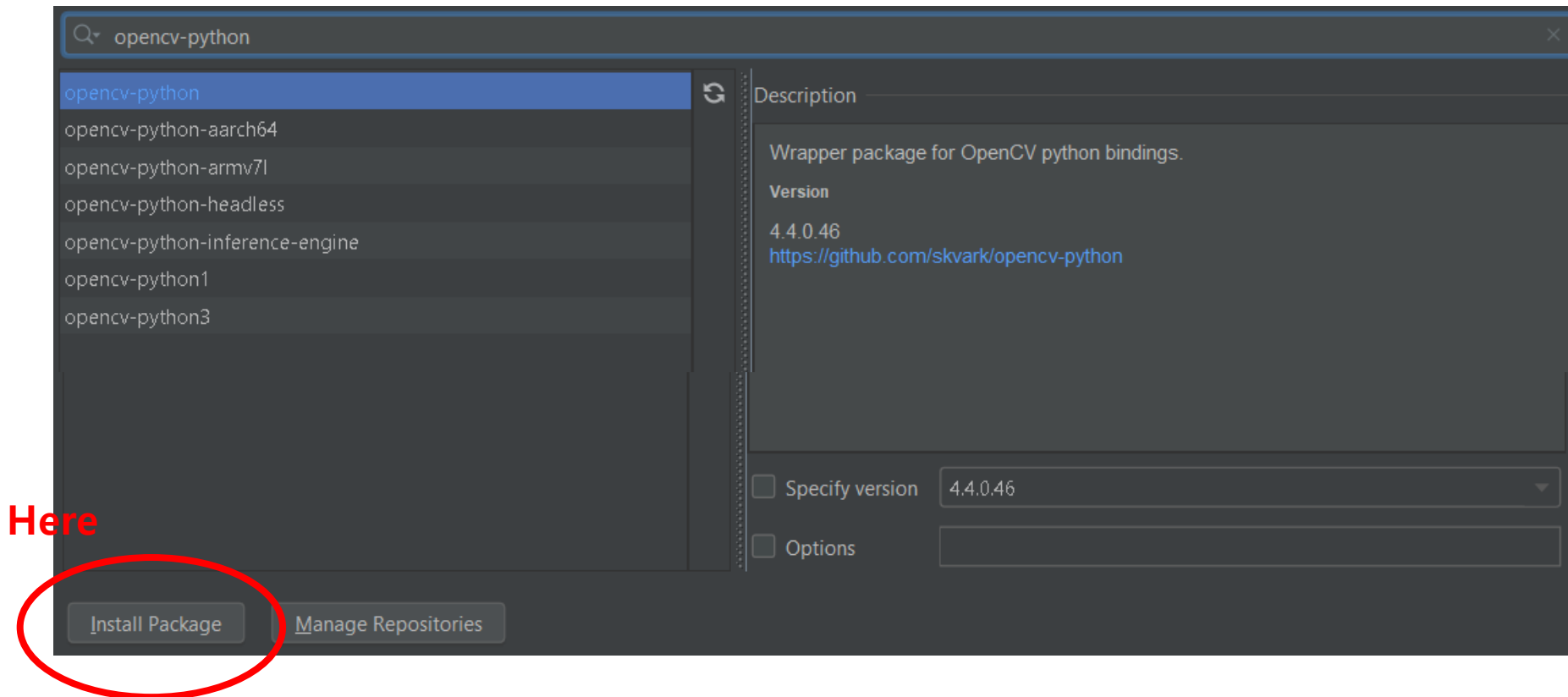
Installing OpenCV [2]

- Go to [Project Interpreter] and click [Install]



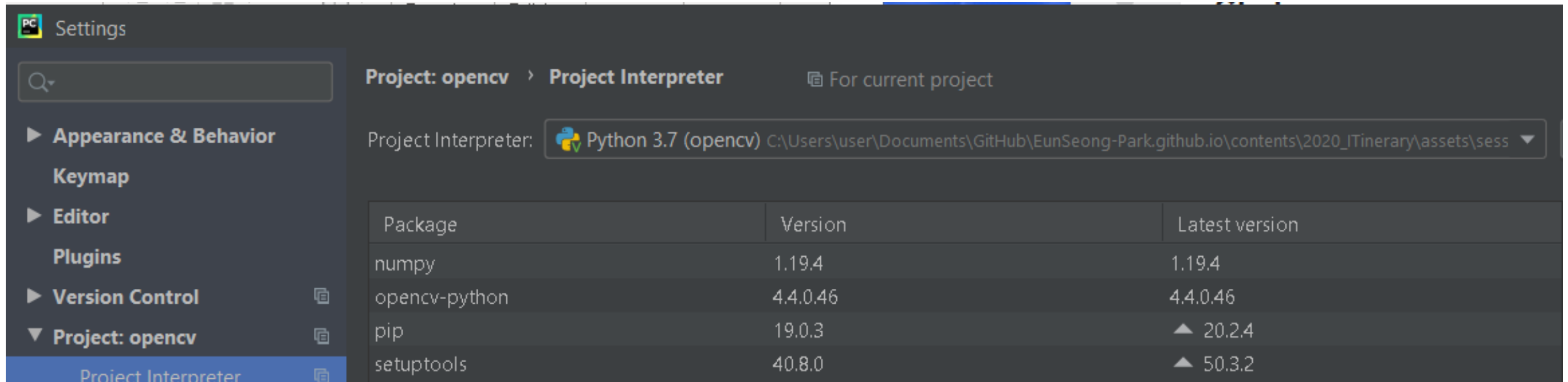
Installing OpenCV [3]

- **Find [opencv-python] and install it**
 - You don't need to change the version, just click [Install]



Installing OpenCV [4]

- Installation takes time (so don't worry)
- After installation, you can find **opencv-python** and **numpy**



Settings

Project: opencv > Project Interpreter For current project

Project Interpreter: Python 3.7 (opencv) C:\Users\user\Documents\GitHub\EunSeong-Park.github.io\contents\2020_ITinerary\assets\sess

Package	Version	Latest version
numpy	1.19.4	1.19.4
opencv-python	4.4.0.46	4.4.0.46
pip	19.0.3	▲ 20.2.4
setuptools	40.8.0	▲ 50.3.2

Installing OpenCV [5]

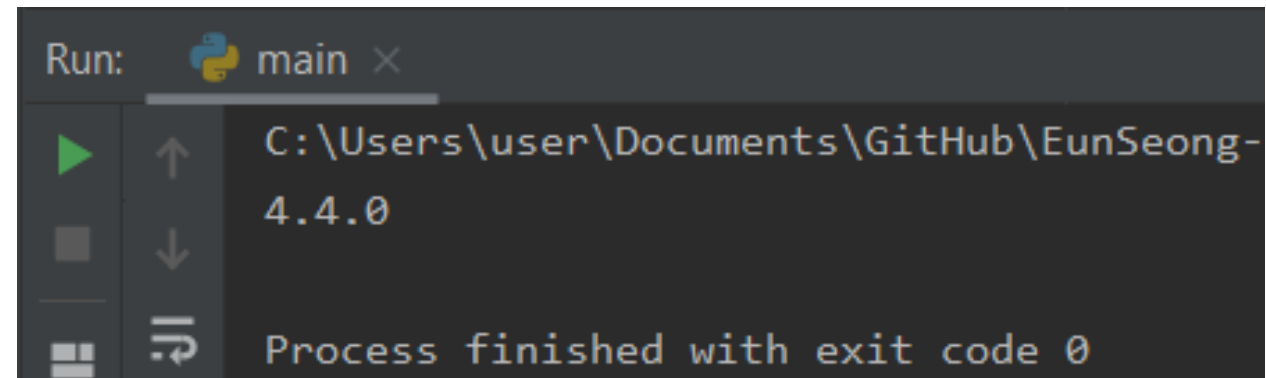
- Then let's check if the installation was successful.

- Write this and run:

```
1 import cv2
2
3 print(cv2.getVersionString())
```

- You got an error?

- Check if you've done correctly and retry
- Or please ask us



The screenshot shows a terminal window titled 'Run: main'. It displays the output of the Python script: 'C:\Users\user\Documents\GitHub\EunSeong-4.4.0'. Below the output, it states 'Process finished with exit code 0', indicating a successful execution.

Backgrounds

Boring time is coming...

Image is a matrix

- Why?



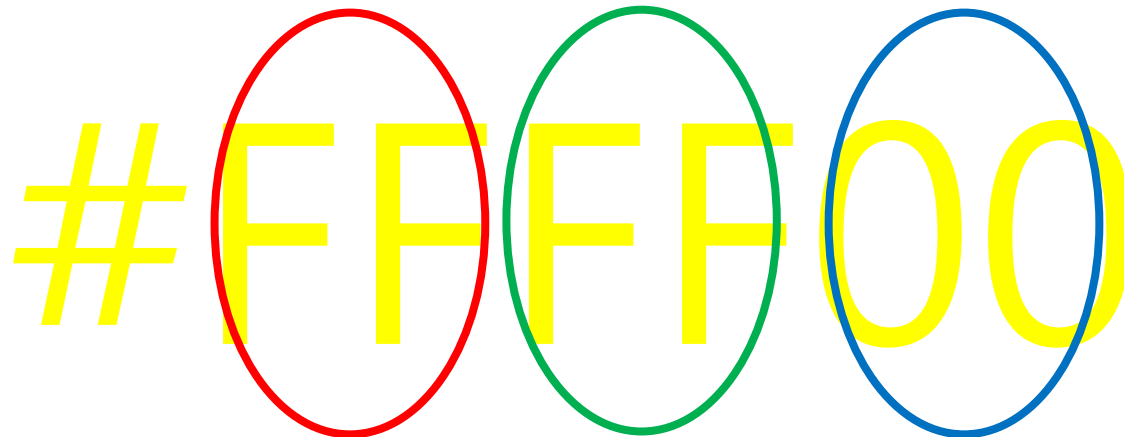
$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

- **An Image contains (Width) * (Height) pixels**
 - So it is a (H)x(W) matrix

Color

- In **RGB**, each component can be 0~255
 - So we can represent $256 * 256 * 256 = 16\text{M}$ colors!
- So each pixel can be dealt with as a 3-tuple(R,G,B) but...
- We can also represent as “an” integer!
 - By hexadecimal representation

#FFF000

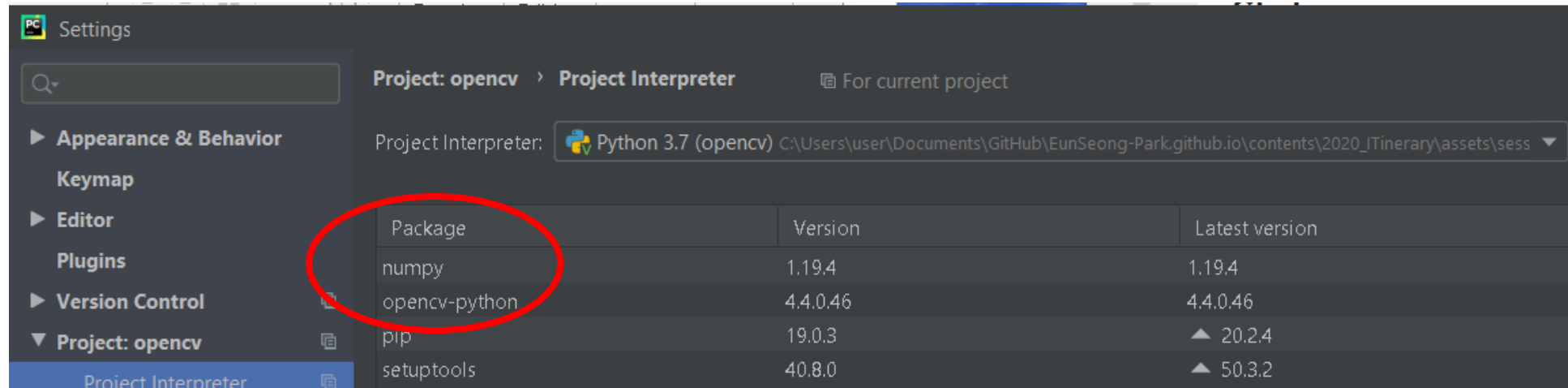
The image shows the hexadecimal color code #FFF000 in a large yellow font. The first two 'F's are enclosed in a red oval, the next two 'F's are enclosed in a green oval, and the final '00' is enclosed in a blue oval. This visualizes the mapping of the hex digits to the RGB components: FF for Red, FF for Green, and 00 for Blue.

Notes

- **Anyway, in OpenCV, image is regarded as 3-dimensional matrix(array)**
 - Height X Width X 3 (RGB)

NumPy [1]

- You might see numpy when we install OpenCV



- What is this?

NumPy [2]



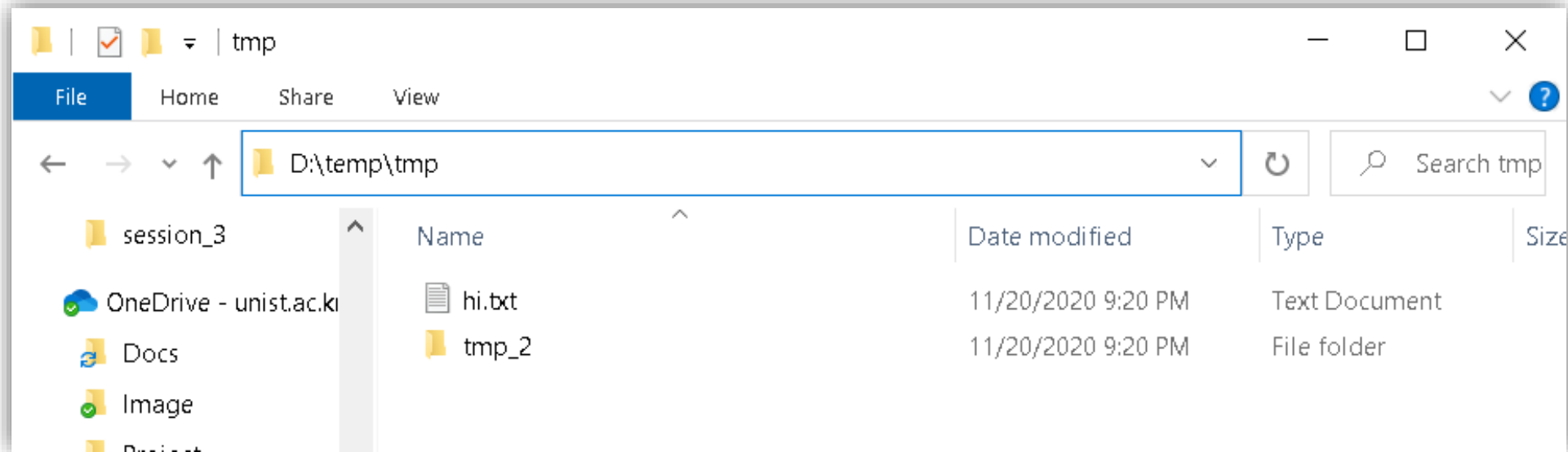
- **NumPy is an open-source library for arrays and matrices**
 - It shows good performance even on very large / multi-dimensional case
 - Using Python list for large-scale calculation is too slow...
- **It is used in many other Python libraries**
 - SciPy
 - Matplotlib
 - Pandas
 - So on...

NumPy [3]

- **But we do not practice it in this class**
 - We don't need to learn about "miscellaneous" things
- **We only use/learn what we need**
 - But some supplement will be given
 - And feel free to ask us!
- A good reference is here:
<https://numpy.org/devdocs/user/whatisnumpy.html>

Path [1]

- **Sometimes, we read/write some file**
 - In OpenCV, we may read/write images/videos
- **Every file has a path**
 - Path is a way to find the file (So, the answer of "where is it?")
 - For example, the following hi.txt has a path: D:\temp\tmp\hi.txt



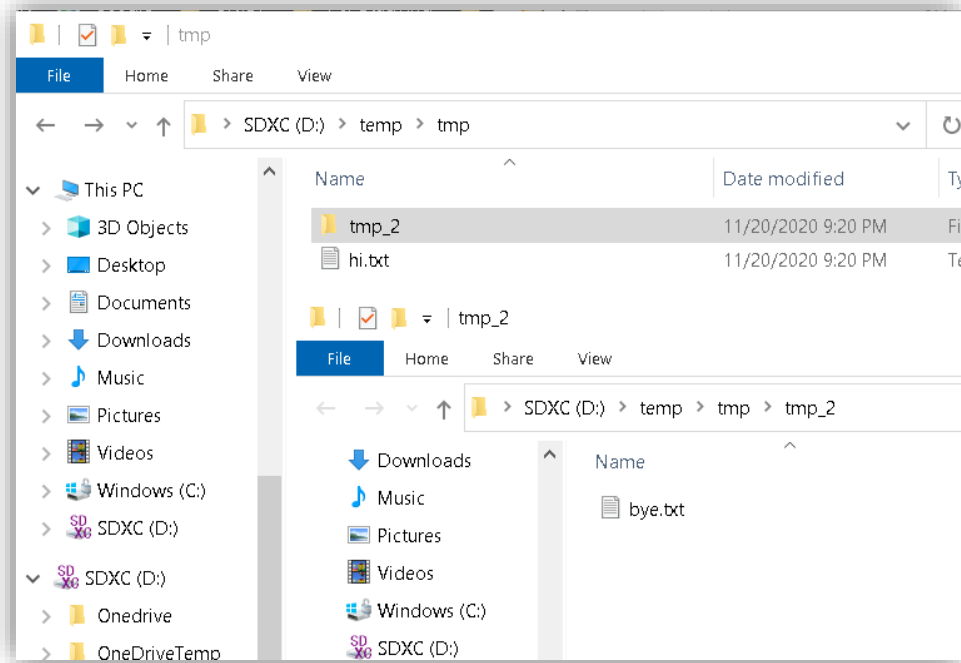
Path [2]

- **Two types of path:**

- Absolute path: A path from some fixed (reference) point (usually root)
 - Previous example used absolute path
 - Because the reference is fixed, absolute path of a file is unique
- Relative path: A path from the current point
 - Let's take an example!

Path [3]

- In a view of hi.txt, what is the relative path of bye.txt?



- It's denoted by
 - "tmp_2\bye.txt", or
 - ".\tmp_2\bye.txt"

Path [4]

- **Some practices are in exercise!**

OpenCV: Image Processing

Boring time is over!

Before We Start

- OpenCV provides various functions
- Note that we **don't need to memorize everything**. Why?
 - There are **MANY** functions/operations/features
 - ...and they are explained in documentation
- Just look around and see how to use it

Preparation

- Just import this

```
1 import cv2
```

Image Read [1]

- Put any image in your project directory

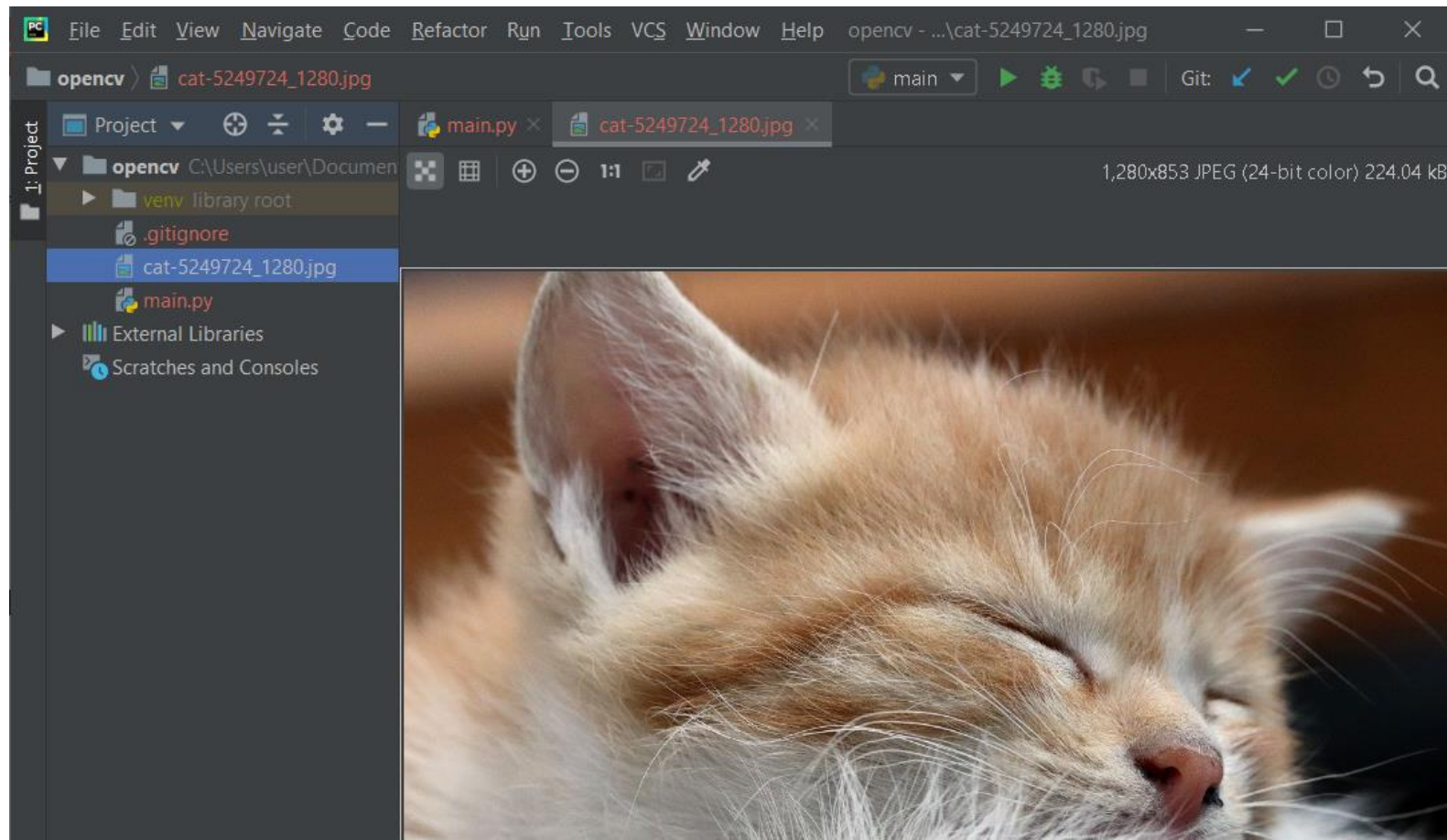
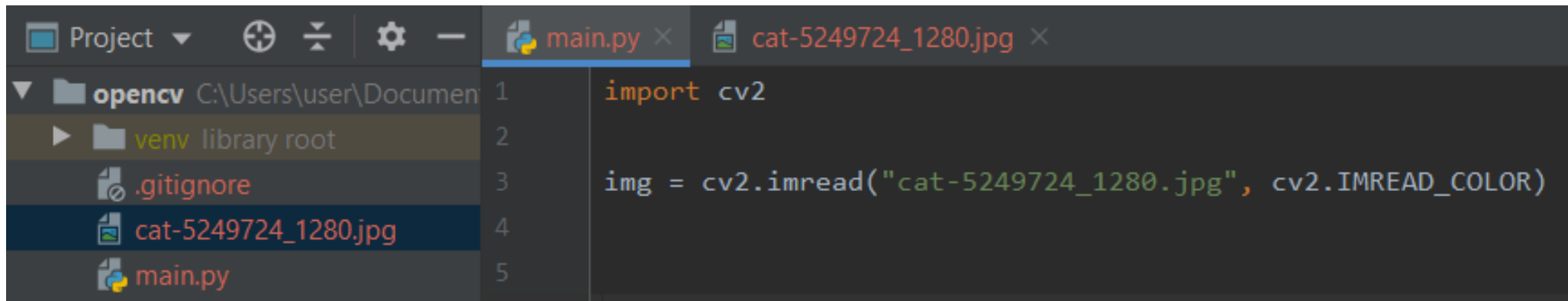


Image Read [2]

- Use `cv2.imread("filename", "flag")`
 - Filename: the path
 - Flag:
 - `cv2.IMREAD_COLOR`: load image with color
 - `cv2.IMREAD_GRAYSCALE`: load image with grayscale
 - `cv2.IMREAD_UNCHANGED`: load image with color (also with alpha-channel)



The screenshot shows a code editor with a dark theme. On the left, a file explorer sidebar shows a project named 'opencv' with a subdirectory 'venv' (library root). Files listed include '.gitignore', 'cat-5249724_1280.jpg', and 'main.py'. The 'main.py' file is open in the editor, showing the following code:

```
1 import cv2
2
3 img = cv2.imread("cat-5249724_1280.jpg", cv2.IMREAD_COLOR)
4
5
```

Image Show [1]

- **Usually, we use three functions to show image**

- `cv2.imshow("window", "image")`: Show image(we got by `imread()`) in window
 - if there is no window named "window", then create it
- `cv2.waitKey("time")`: Wait for any keyboard input or time(in ms, infinite when time=0)
 - returns the ASCII of key
- `cv2.destroyAllWindows()`: Destroy all windows

Image Show [2]

- Oh... slightly big but OK

```
1 import cv2
2
3 img = cv2.imread("cat-5249724_1280.jpg", cv2.IMREAD_COLOR)
4
5 cv2.imshow("My cute cat", img)
6 cv2.waitKey(0)
7 cv2.destroyAllWindows()
```

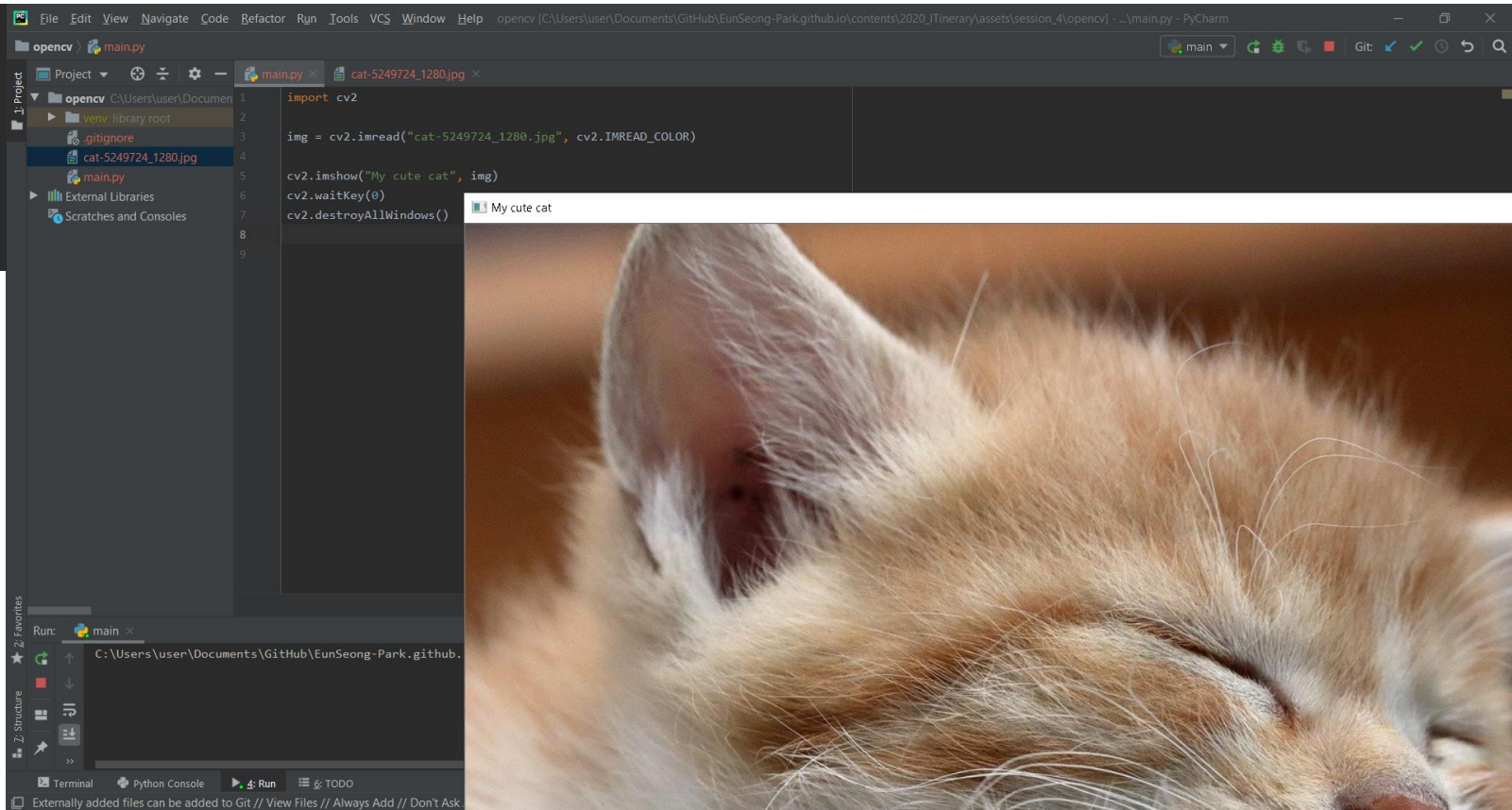


Image Show [3]

- **Try to use grayscale**
 - `cv2.imread("filename", cv2.IMREAD_GRAYSCALE)`

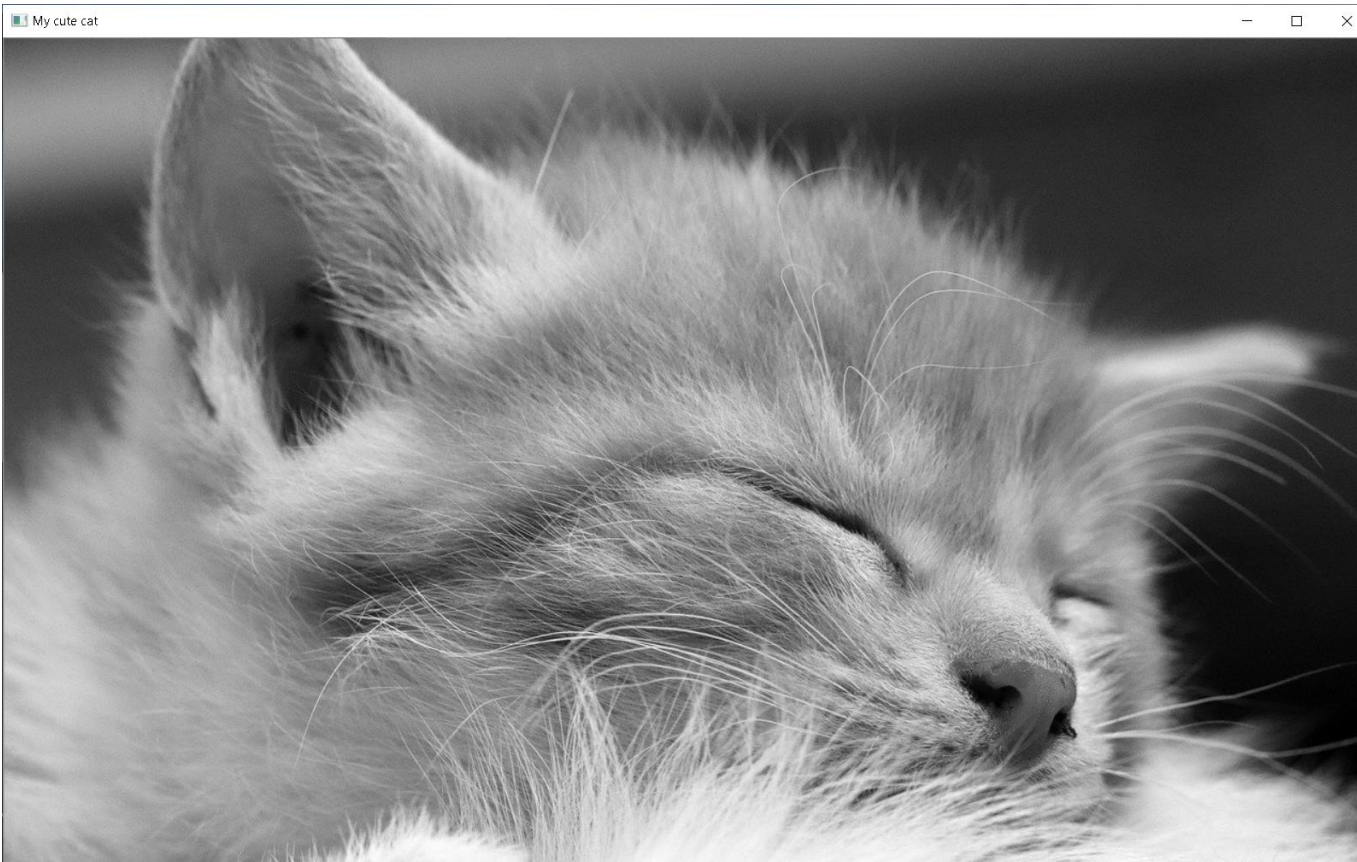
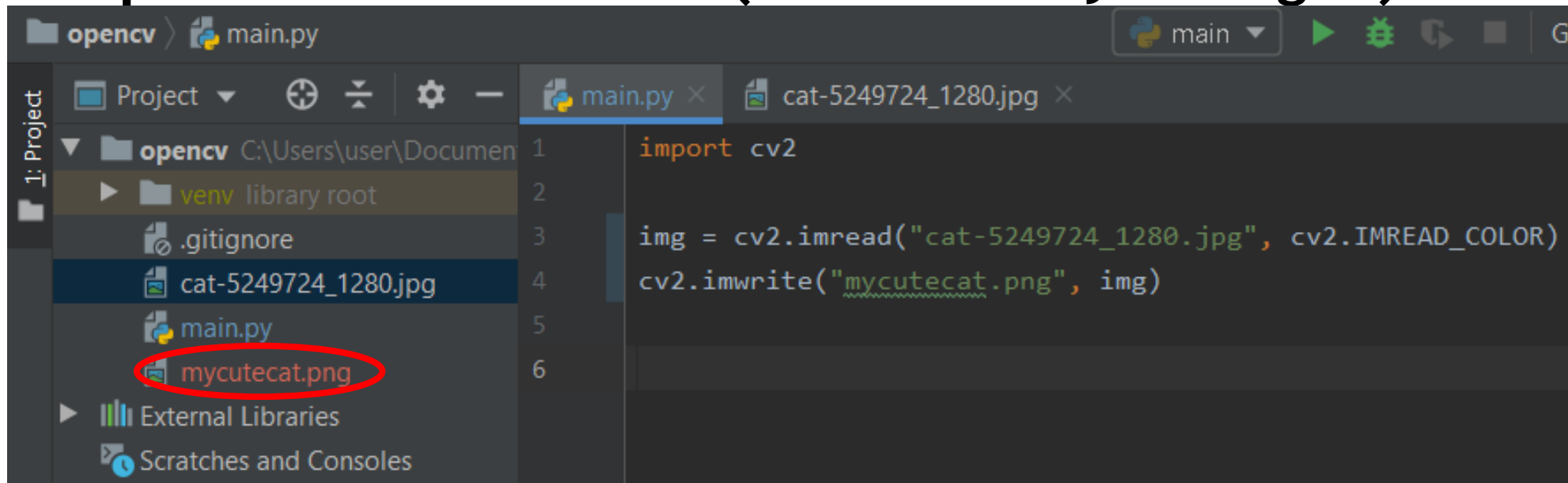


Image Write

- We can make or modify some image, but how to write(save as a file)?
- Simple! Use `cv2.imwrite("filename", "image")`



Basic Operations [1]

- **As mentioned, we can treat image as a matrix**
 - Actually, image object in OpenCV is a NumPy array

Basic Operations [1]

- We can access to an individual pixel: a dot!
 - What does it mean?

```
10 img = cv2.imread("mycutecat.png", cv2.IMREAD_COLOR)
11 print(img[25, 31])
```

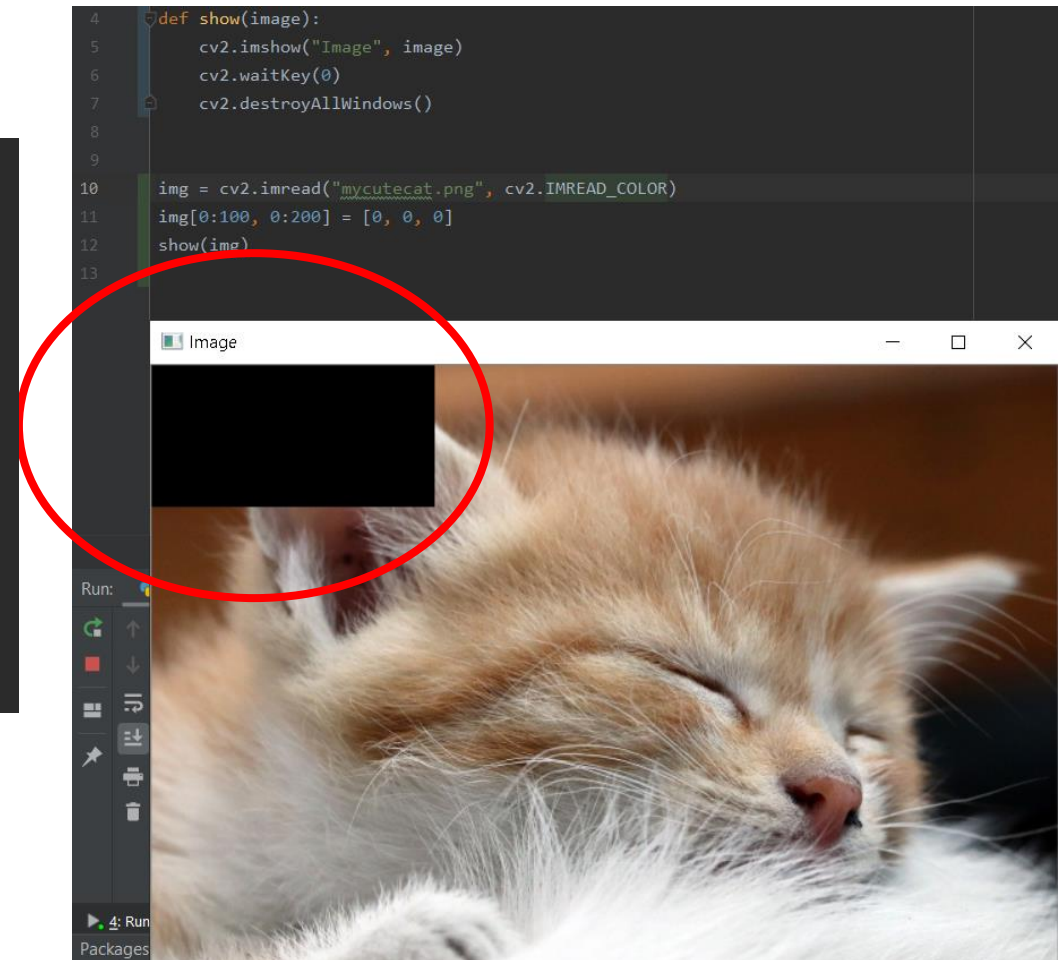
```
▶ ↑ C:\Users\user\Documents\GitHub\EunSeong-Park.github.io\cor
[117 141 184]
```

- Of course, we can change it
 - But can we notice?

Basic Operations [2]

- Using slicing may be more practical

```
4 def show(image):
5     cv2.imshow("Image", image)
6     cv2.waitKey(0)
7     cv2.destroyAllWindows()
8
9
10 img = cv2.imread("mycutecat.png", cv2.IMREAD_COLOR)
11 img[0:100, 0:200] = [0, 0, 0]
12 show(img)
```

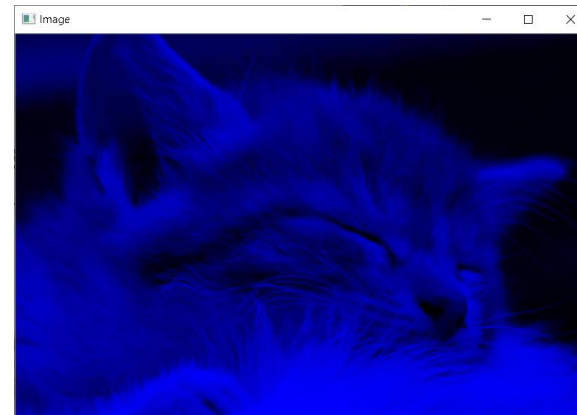
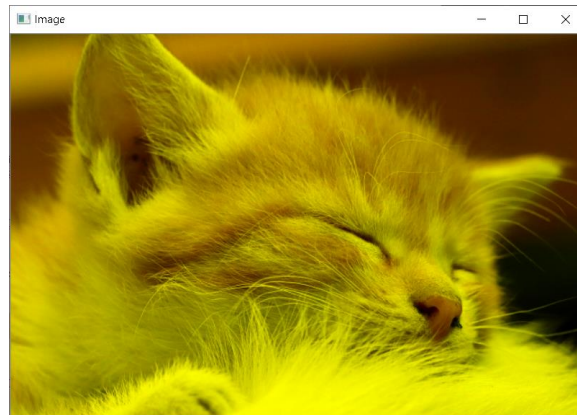
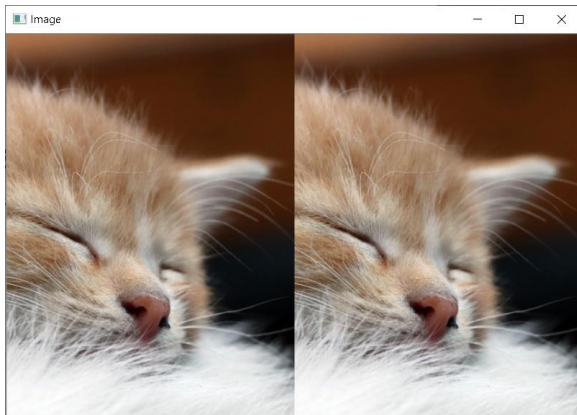


Basic Operations [3]

- **We can do many things only with matrix operation**
 - Try them! How are they made?



Original one



Basic Operations [4]

- Note that the color channel is in order, BGR, not RGB !

```
10 img = cv2.imread("mycutecat.png", cv2.IMREAD_COLOR)
11 img[:, :] = [255, 0, 0]
12 show(img)
```

Image



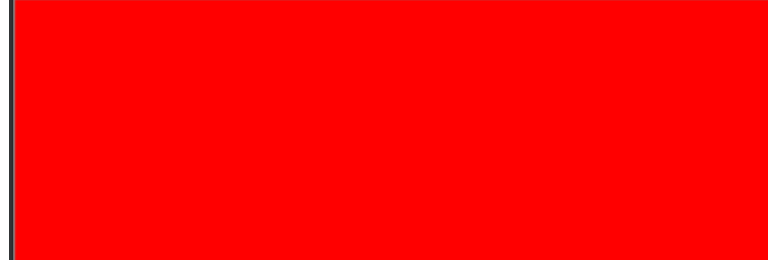
```
10 img = cv2.imread("mycutecat.png", cv2.IMREAD_COLOR)
11 img[:, :] = [0, 255, 0]
12 show(img)
```

Image



```
10 img = cv2.imread("mycutecat.png", cv2.IMREAD_COLOR)
11 img[:, :] = [0, 0, 255]
12 show(img)
```

Image



- Anyway, practice it in many way

Drawing [1]

- **We can draw something in image by**
 - Calling functions
 - Mouse event

Drawing [2]

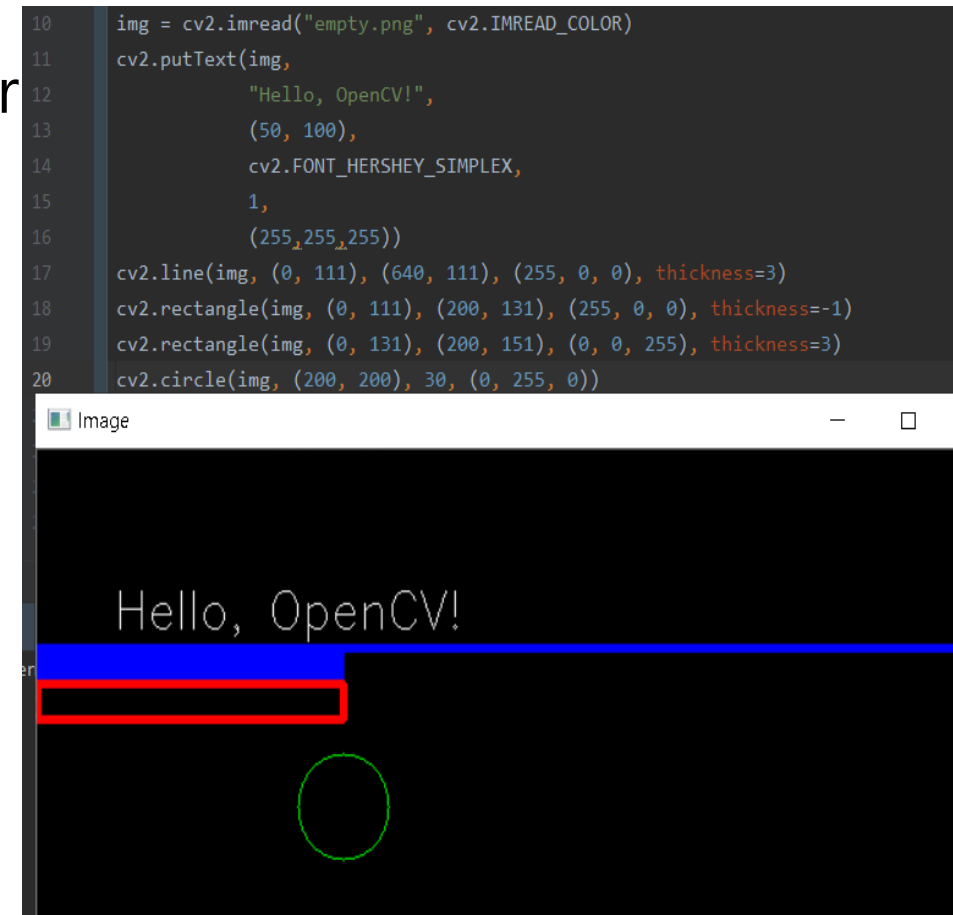
- **We can draw lines, rects, circles, etc.**
 - Also, we can put some text in the image
 - Ellipse? Arbitrary polygon?
- **These are very intuitive:**
 - Points are 2-tuple
 - Origin stands for string's bottom left corner
 - Some fonts are defined in OpenCV (cv2.FONT_XXX)

<code>cv2.line(image, start_point, end_point, color)</code>
<code>cv2.rectangle(image, start_point, end_point, color)</code>
<code>cv2.circle(image, center_point, radius, color)</code>
<code>cv2.putText(image, text, origin, font, fontsize, color)</code>

Drawing [3]

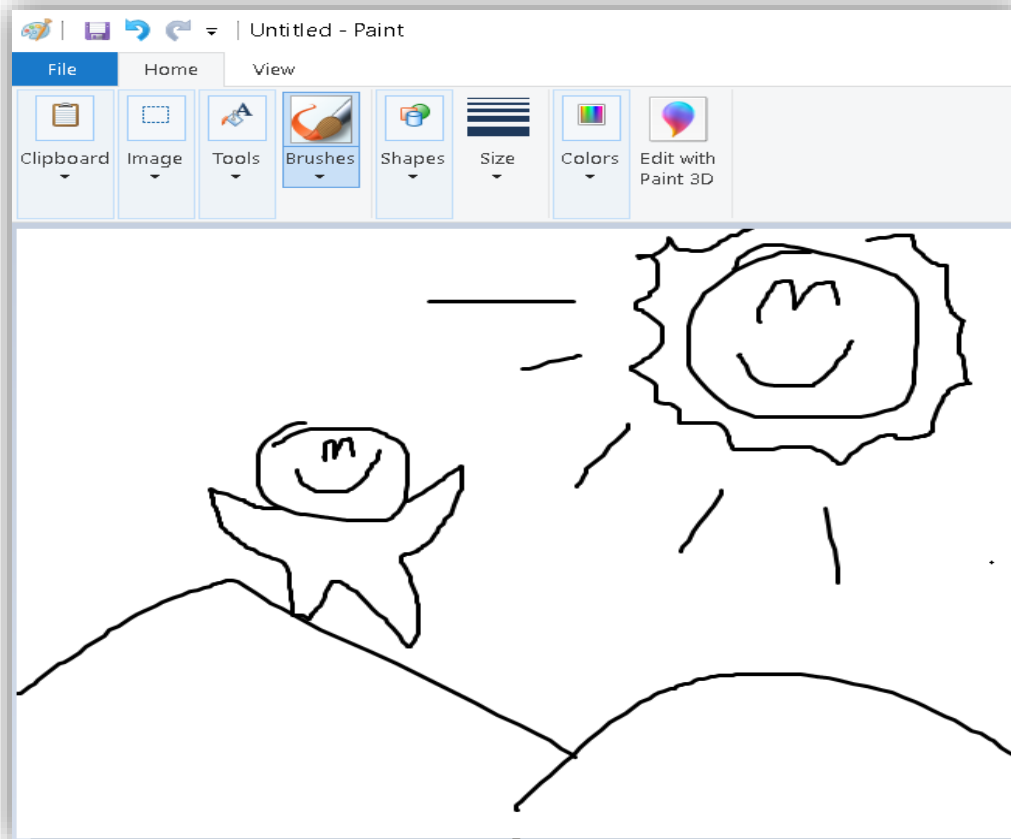
- **Notes:**

- Points are 2-tuple
- Origin stands for string's bottom left corner
- Some fonts are defined in OpenCV
 - cv2.FONT_XXX
- Thickness is optional
 - If -1, inside of shapes is filled
 - Look at the blue one



Drawing with Mouse [1]

- We want to draw with our mouse, like...



Drawing with Mouse [2]

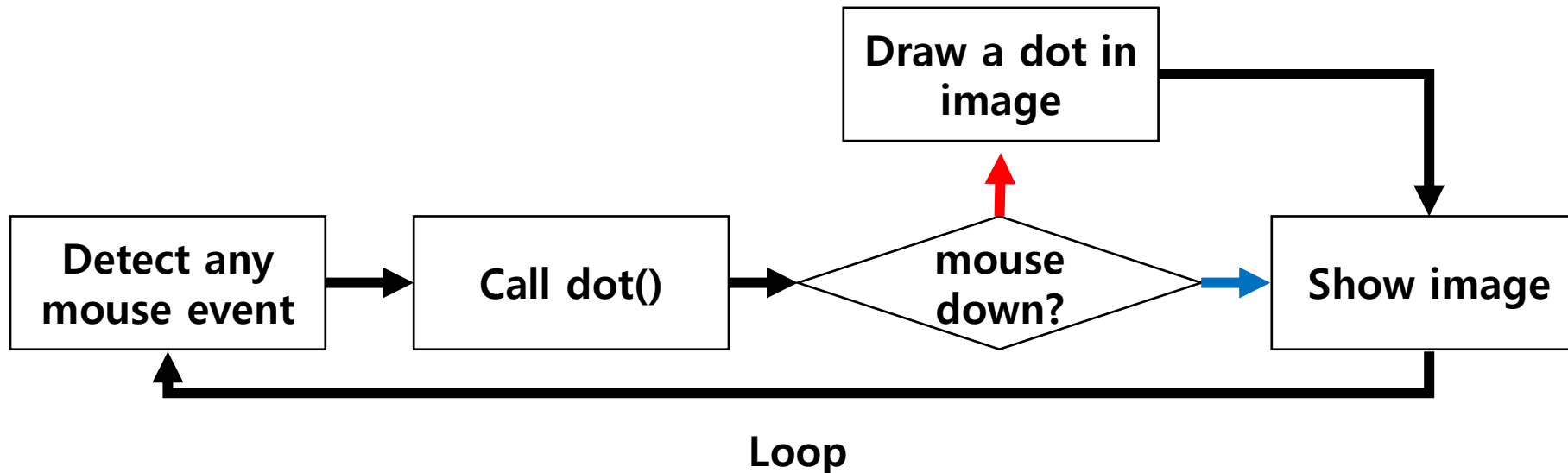
- **There are many kinds of mouse event in OpenCV**
 - up / down / doubleclick / wheel / move...
- **We will use `cv2.setMouseCallback()` function**
 - Detect events and call some function
 - More detail in the next slide

Drawing with Mouse [3]

- **`cv2.setMouseCallback("window", "function", "param")`**
 - "window": Window
 - "function": When event is detected, call it
 - "param": Parameter that will be passed to "function"
- **The function is called with parameter:**
 - event, x, y, flags, param
 - You should define the function carefully

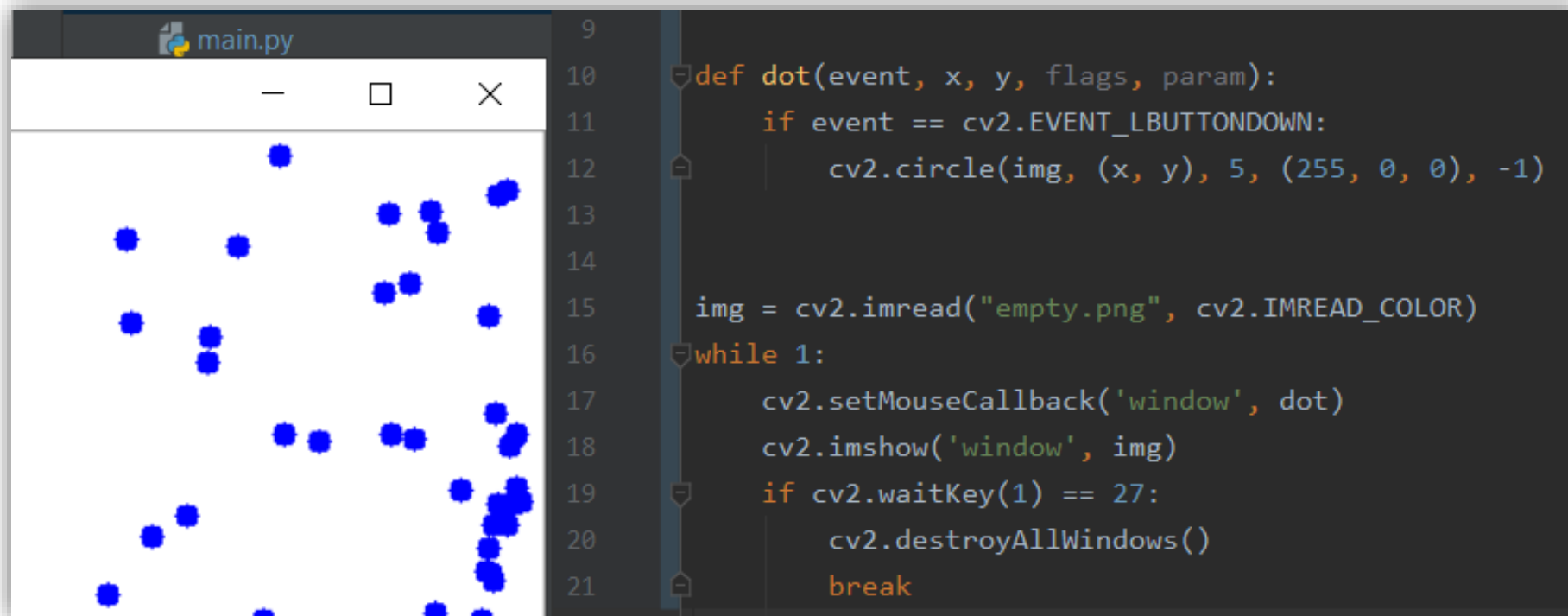
Drawing with Mouse [4]

- **Let's make a program that:**
 - draws a small circle(dot) when we click (left-mouse-down)
 - Detect an event, `cv2.EVENT_LBUTTONDOWN`
- **The flow is like this:**



Drawing with Mouse [5]

- It works well, but something awkward...
 - Unfortunately, there's no event like mouse-holding



Drawing with Mouse [6]

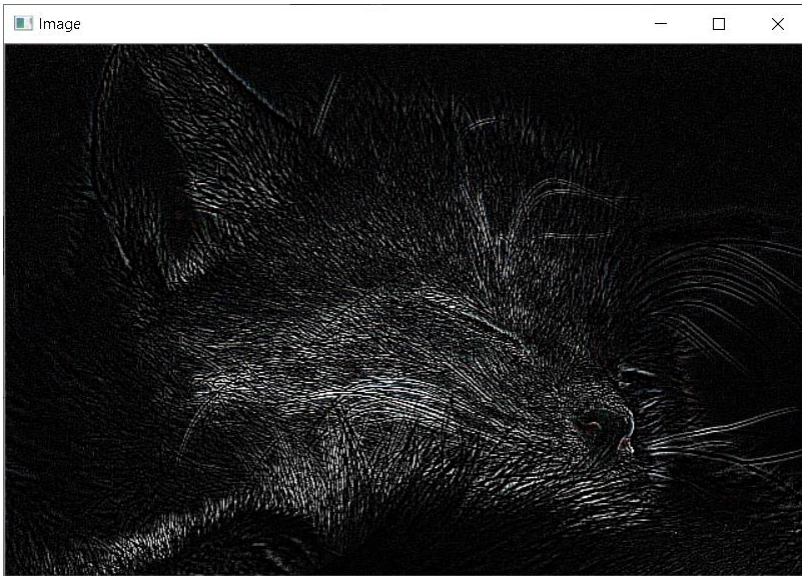
- We will make more realistic painter in lab session!

Other Basic Operations [1]

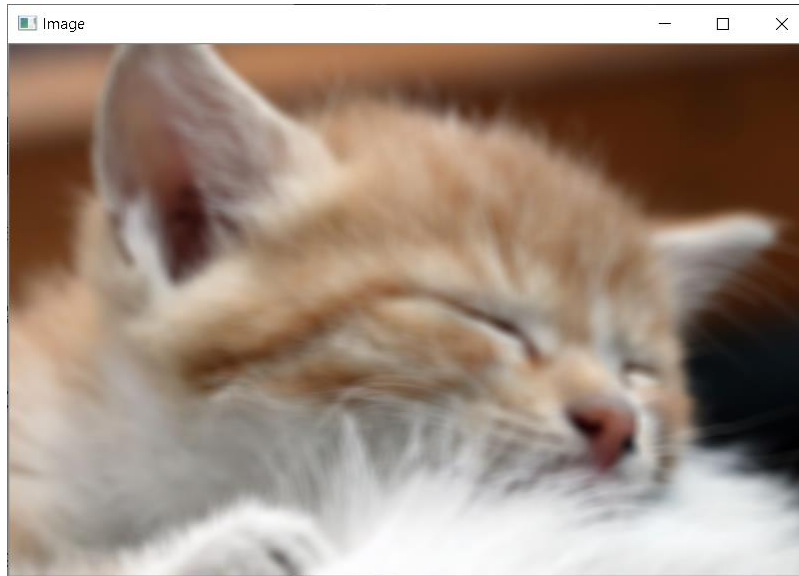
- **cv2.flip("image", "flipcode")**
 - "flipcode"=0: top-down
 - "flipcode"=1: left-right
- **cv2.pyrUp("image") / cv2.pyrDown("image")**
 - only double/half
 - How can we do that for arbitrary scale?
- **cv2.resize("image", "dsize")**
 - "dsize": (w, h) 2-tuple

Other Basic Operations [2]

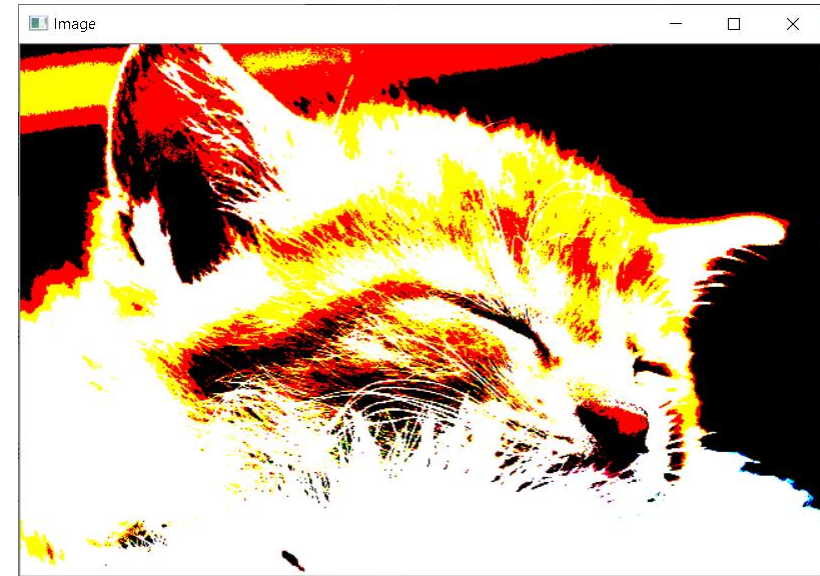
- **There are many interesting functions in OpenCV**
 - Perhaps, something need some mathematical background



Edge detection



Blur



Binarization

Video

- **Usually, the term “image processing” involves video**
- **From now, we will deal with video**
 - Read, modify, and write!
 - Also, we will extract some useful information from video

Video: Preparation

- **Is your webcam available?**
- **Otherwise, you can prepare any something other video**
 - You should have it as a file! (.mp4, .wmv, etc.)
 - You can take a video with your phone
 - Anything with 10~30 seconds is ok.

Video Capture Using Camera [1]

- **`cv2.VideoCapture("something")` captures video**
 - If argument is number: (usually 0) uses camera in your computer
 - If argument is path: uses the video located in path
- **Example:**
 - `cv2.VideoCapture(0)`
 - `cv2.VideoCapture("/some_video.avi")`

Video Capture Using Camera [2]

- **The function returns a special type**

- It shows nothing, just capture and save
- How can we use it?

```
16 cap = cv2.VideoCapture(0)
17 print(type(cap))
```



```
C:\Users\user\Documents\GitHu
<class 'cv2.VideoCapture'>
```

- **We can use with read() method that returns 2-tuple, (ret, frame)**

- ret: if reading was successful (Boolean)
- frame: the captured frame (image)

- **Note that a video (w/o sound) is a sequence of image**

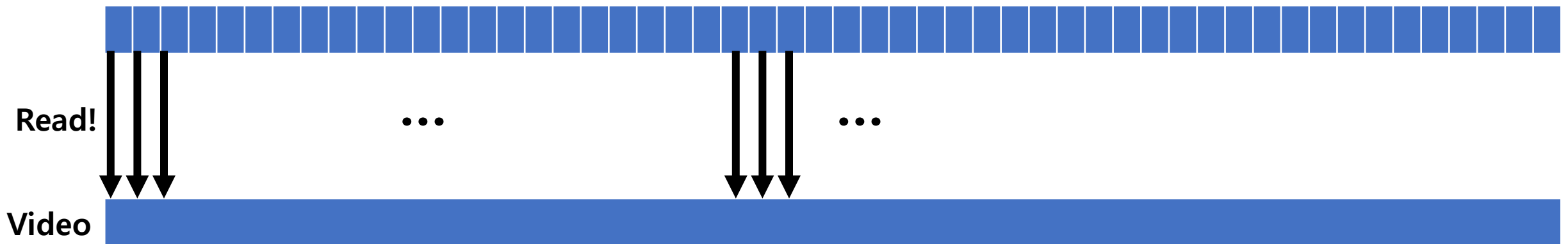
- Unfortunately, OpenCV does not support sound
- We need other library for this

Video Capture Using Camera [3]

- **VideoCapture class supports several methods**
 - `set("property", "value")`: set the property to the value
 - `get("property")`: get value of the property
 - Frame width: `cv2. CAP_PROP_FRAME_WIDTH` (or just use number, 3)
 - Frame height: `cv2. CAP_PROP_FRAME_HEIGHT` (or just use number, 4)
 - `isOpened()`: if successfully opened, then returns True / otherwise False
 - `release()`: release the camera
 - call this when we finished to use

Video Capture Using Camera [4]

- **Idea: Read from captured video and show it, at some intervals**
 - The interval is called frame rate(FPS?)
 - usually 30, 60, or higher... **Check your cam's FPS**
 - So, video is sequence of image, with sufficiently high frame rate
 - If it is too slow, it may be called slide show...



- **We should “wait” a moment, before show next frame**

Video Capture Using Camera [5]

```
16 cap = cv2.VideoCapture(0)
17 cap.set(3, 640) # width
18 cap.set(4, 480) # height
19
20 while cap.isOpened():
21     isSuccess, frame = cap.read()
22     if isSuccess:
23         cv2.imshow('Video', frame)
24         key = cv2.waitKey(1000 // 60)
25         if key == 27:
26             break
27
28 cap.release()
29 cv2.destroyAllWindows()
```

Setting frame width / height

wait for 1/60 sec => 60 FPS

It should be ended eventually (by ESC)

Video Capture Using Camera [6]

- Now, we can load and show videos
- Because each frame is given, we can use same methods for image processing we covered

Writing Video File [1]

- Then let's write a video file
- We use `cv2.VideoWriter` object to write
 - `cv2.VideoWriter("Filename", "Codec", "FPS", "Size")`
- And append each frame with `write("frame")` method
- When you finished to write, release with `release()` method

Codec [1]

- **We encode / decode the video, why?**
 - Encode: to make computer understand it
 - Decode: to make human understand it
- **The rule for ENcode and DECode is codec**
 - ...and there are many kinds!
- **Detail explanation in the supplemental material!**
 - It's an interesting topic

Codec [2]

- **Anyway, we should specify the codec appropriately**
 - We will only deal with .avi ("DIVX")
 - It may depend on your system
 - **So please try it and check!**
- **Your codec can be declared with...**
 - `codec = cv2.VideoWriter_fourcc(*"DIVX")`
 - put it into the parameter, codec

Example: Video Recorder [1]

- Record video from your webcam, and save as a file
- More in detail...
 - You should show the real-time video (we already learned)
 - SpaceBar to start/pause the recording
 - When the recording is done, stop to write and save it
 - ESC to terminate the recording and program

Example: Video Recorder [2]

- Some preparation

```
16 cap = cv2.VideoCapture(0)
17 cap.set(3, 640) # width
18 cap.set(4, 480) # height
19
20 codec = cv2.VideoWriter_fourcc(*"DIVX") Codec
21 output = cv2.VideoWriter('MyRecording.avi', codec, 30, (640, 480))
22 recording = False
```

Example: Video Recorder [3]

- Some preparation

```
16 cap = cv2.VideoCapture(0)
17 cap.set(3, 640) # width
18 cap.set(4, 480) # height
19
20 codec = cv2.VideoWriter_fourcc(*"DIVX")
21 output = cv2.VideoWriter('MyRecording.avi', codec, 30, (640, 480))
22 recording = False
```

File name	Codec	FPS	Size(w, h)
-----------	-------	-----	------------

- We recommend to use 30 FPS

Example: Video Recorder [4]

- Some preparation

```
16 cap = cv2.VideoCapture(0)
17 cap.set(3, 640) # width
18 cap.set(4, 480) # height
19
20 codec = cv2.VideoWriter_fourcc(*"DIVX")
21 output = cv2.VideoWriter('MyRecording.avi', codec, 30, (640, 480))
22 recording = False
```

Records only when recording is set to True

Example: Video Recorder [5]

```
24 while cap.isOpened():
25     isSuccess, frame = cap.read()
26     if isSuccess:
27         cv2.imshow('Video', frame)
28
29         key = cv2.waitKey(1000 // 30)
30         if recording:
31             output.write(frame)
32
33     if key == 27: # ESC
34         output.release()
35         break
36
37     if key == 32: # SpaceBar
38         if not recording:
39             recording = True
40             print("Start recording")
41         else:
42             recording = False
43             print("Pause!")
44
45 cap.release()
46 cv2.destroyAllWindows()
```

Records only when recording is set to True

Start / pause the recording

In the Lab Session

- **We will implement something more advanced**
 - Choice 1: Application of facial recognition
 - Choice 2: Video call (with network programming)
 - And so on....
- **Practice yourself and ask any question!**

Thank you