파이썬으로 배우는 딥러닝

이미지 전처리 (OpenCV)

Ch.01 OpenCV 개요와 환경구축

강의에 관한 전반적인 안내

- **본 강의는 OpenCV를 활용하여 다양한 영상처리 기법을 학습하는 것을 목표로 합니다.**

- **본 강의에서 복잡한 영상처리 원리에 대한 설명은 가급적 배제하고 향후 있을 프로젝트 수행에 필요한 다양한 기법을 연습하는데 중점을 두었습니다.**

- **본 강의는 파이썬 실습 코드를 다수 포함하고 있으며 직접 수행해 봄으로서 해당 API 활용법을 연습해 봅니다.**

OpenCV 개요

◆ OpenCV (Open Source Computer Vision)

- 컴퓨터 비전 및 머신러닝을 위한 이미지 처리 라이브러리
- 인텔에서 개발한 이후, **컴퓨터 비전 관련 작업을 위한 표준 라이브러리**로 자리 잡음
- Python, C++, Java 등을 지원하며 실시간 이미지 및 동영상 분석을 위한 기능 제공

◆ 주요 기능

- **이미지 처리** : 이미지 필터링, 블러링 등 기본적인 이미지 변형 및 개선
- **개체 검출** : 얼굴, 사물 등의 특정 개체 검출 및 인식 기능
- **동영상 분석** : 영상에서 개체를 추적하거나 변화 감지
- **증강 현실** : 가상 개체를 실제 이미지나 영상에 결합하는 기능

OpenCV 개요

◆ 활용 사례

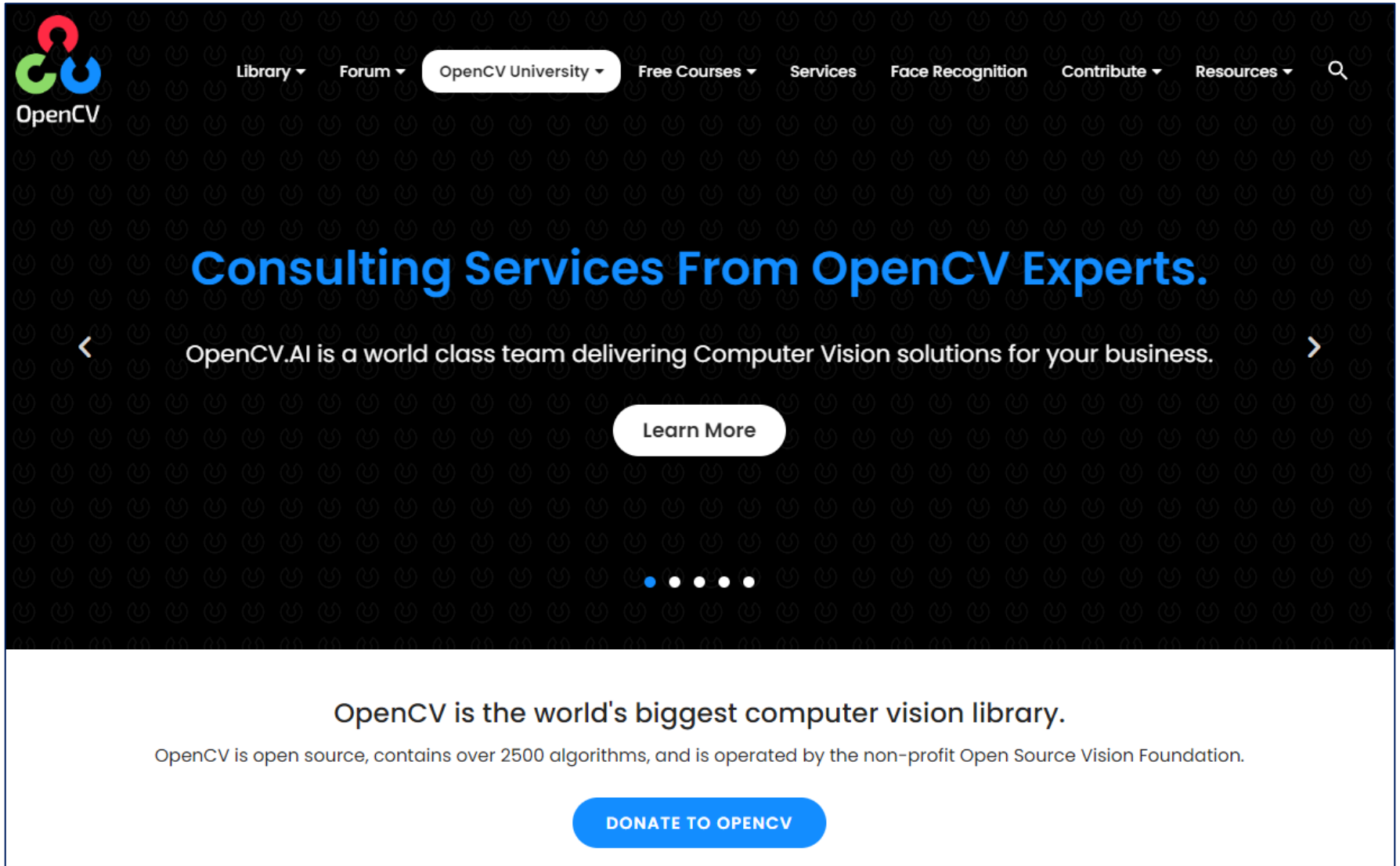
- 얼굴 인식 : 접근 제어, 보안 시스템 등에서 활용
- 산업 자동화 : 제품 검사, 결함 검출 등으로 생산성 향상
- 자율 주행 : 차량 내/외부의 다양한 개체 인식 및 경로 파악

◆ 소스코드 구성과 Cross platform

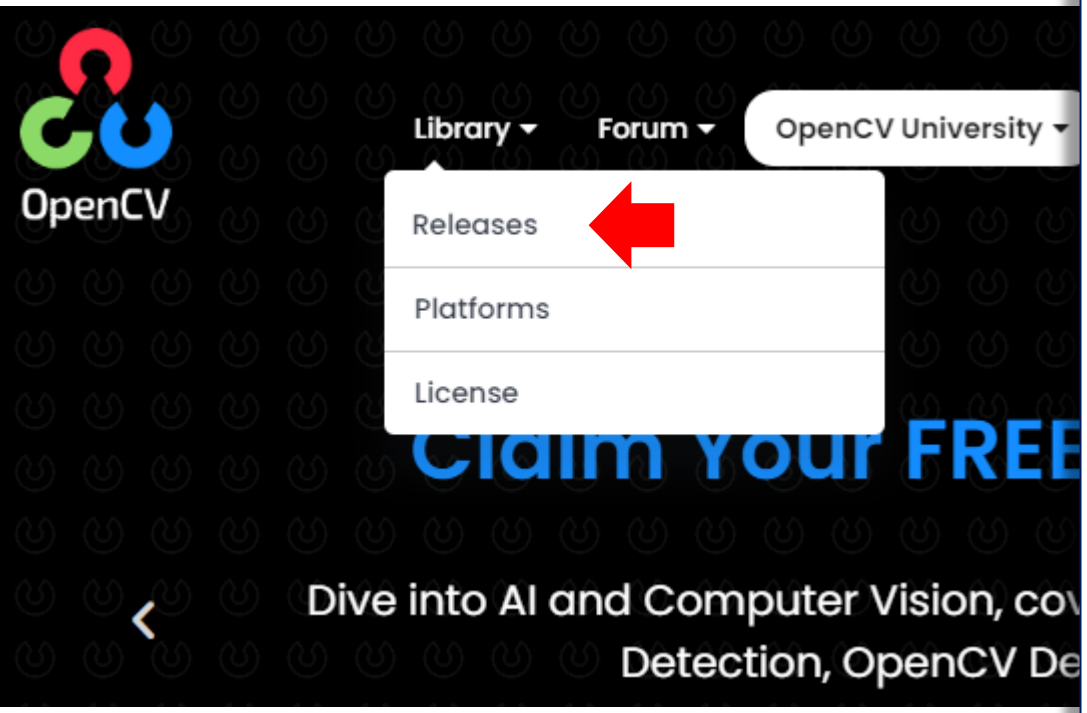
- 클래스와 함수는 C와 C++로 개발. 전체 코드는 180만 라인 이상
- 인터페이스 언어는 C, C++, 자바, 자바스크립트, 파이썬
- OS 플랫폼은 윈도우, 리눅스, macOS, 안드로이드, iOS
- Cross-platform 지원
- 교육과 상업 목적 모두 무료

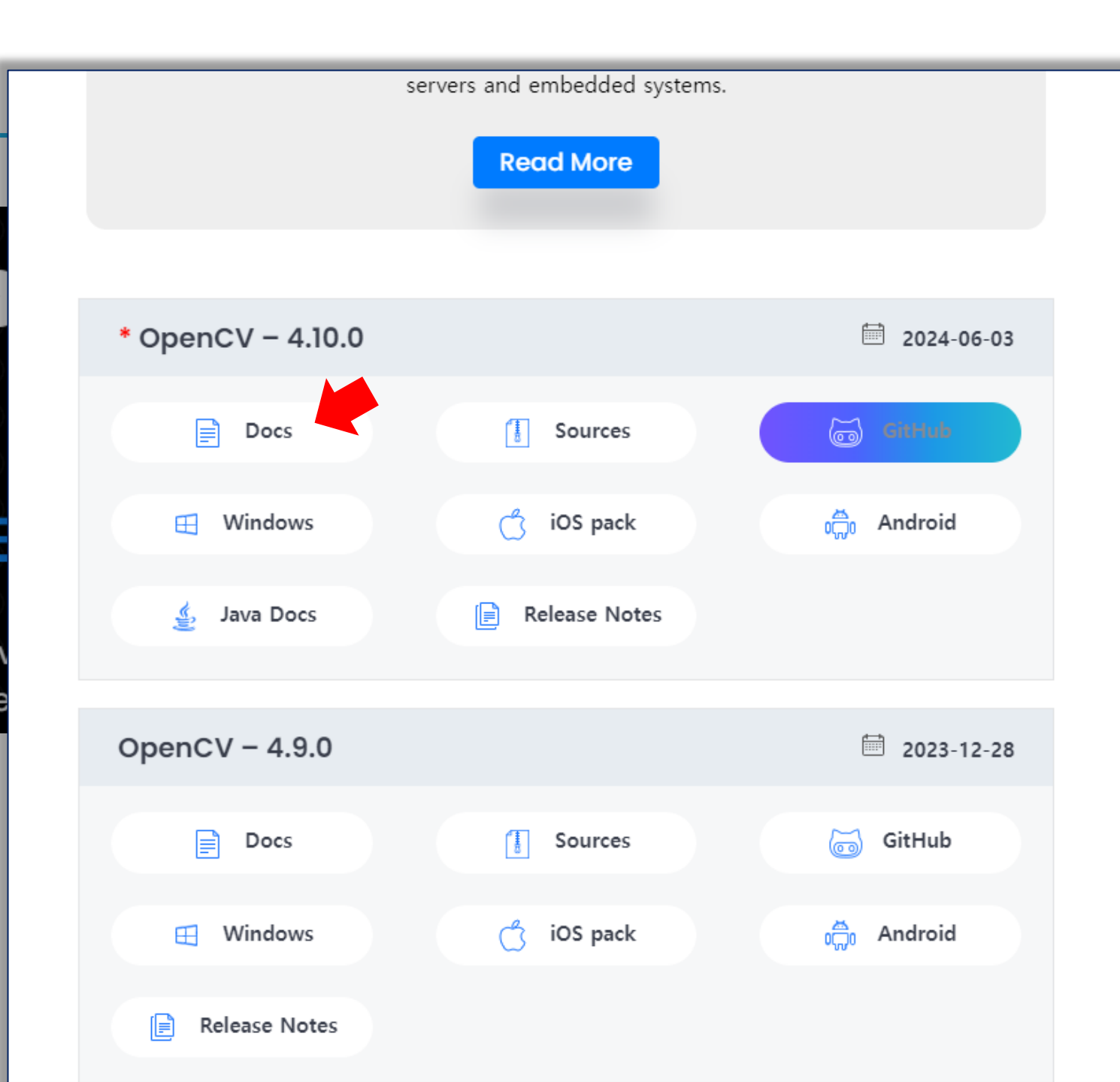
OpenCV 개요

https://opencv.org/



OpenCV 개요


OpenCV
Library
Forum
OpenCV University
Releases
Platforms
License
Claim Your FREE
Dive into AI and Computer Vision, co
Detection, OpenCV De


servers and embedded systems.
Read More
* OpenCV – 4.10.0
2024-06-03
Docs
Sources
GitHub
Windows
iOS pack
Android
Java Docs
Release Notes
OpenCV – 4.9.0
2023-12-28
Docs
Sources
GitHub
Windows
iOS pack
Android
Release Notes

OpenCV 개요

OpenCV 4.10.0
Open Source Computer Vision

Main Page | Related Pages | Namespaces | Classes | Files | Examples | Java documentation | Search

OpenCV 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
  - videoio. 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

OpenCV 개요

Image Processing

Modules

- Image Filtering
- Geometric Image Transformations
- Miscellaneous Image Transformations
- Drawing Functions
- Color Space Conversions
- ColorMaps in OpenCV
- Planar Subdivision
- Histograms
- Structural Analysis and Shape Descriptors
- Motion Analysis and Object Tracking
- Feature Detection
- Object Detection
- Image Segmentation
- Hardware Acceleration Layer

OpenCV 개요

Image Filtering

Image Processing

Detailed Description

Functions and classes described in this section are used to perform various linear or non-linear filtering operations on 2D images (represented
means that for each pixel location (x, y) in the source image (normally, rectangular), its neighborhood is considered and used to compute the r
case of a linear filter, it is a weighted sum of pixel values. In case of morphological operations, it is the minimum or maximum values, and so on
computed response is stored in the destination image at the same location (x, y). It means that the output image will be of the same size as th
Normally, the functions support multi-channel arrays, in which case every channel is processed independently. Therefore, the output image will
same number of channels as the input one.

Another common feature of the functions and classes described in this section is that, unlike simple arithmetic functions, they need to extrapola
some non-existing pixels. For example, if you want to smooth an image using a Gaussian 3×3 filter, then, when processing the left-most pixel
you need pixels to the left of them, that is, outside of the image. You can let these pixels be the same as the left-most image pixels ("replicated
extrapolation method), or assume that all the non-existing pixels are zeros ("constant border" extrapolation method), and so on. OpenCV enabl
specify the extrapolation method. For details, see **BorderTypes**

Depth combinations

Input depth (src.depth())	Output depth (ddepth)
CV_8U	-1/CV_16S/CV_32F/CV_64F
CV_16U/CV_16S	-1/CV_32F/CV_64F

OpenCV 개요

함수 정의 예시

◆ medianBlur()

```
void cv::medianBlur ( InputArray  src,
                      OutputArray dst,
                      int         ksize
                    )
```

Python:

```
cv.medianBlur( src, ksize[, dst] ) -> dst
```

```
#include <opencv2/imgproc.hpp>
```

Blurs an image using the median filter.

The function smoothes an image using the median filter with the **ksize** × **ksize** aperture. Each channel of a multi-channel image is processed independently. In-place operation is supported.

Note

The median filter uses **BORDER_REPLICATE** internally to cope with border pixels, see **BorderTypes**

Parameters

src input 1-, 3-, or 4-channel image; when ksize is 3 or 5, the image depth should be CV_8U, CV_16U, or CV_32F, for larger aperture sizes, it can only be CV_8U.

dst destination array of the same size and type as src.

ksize aperture linear size; it must be odd and greater than 1, for example: 3, 5, 7 ...

See also

bilateralFilter, **blur**, **boxFilter**, **GaussianBlur**

OpenCV 개요

함수 정의 예시

◆ filter2D()

```
void cv::filter2D ( InputArray    src,
                    OutputArray   dst,
                    int           ddepth,
                    InputArray    kernel,
                    Point         anchor = Point(-1,-1) ,
                    double        delta = 0 ,
                    int           borderType = BORDER_DEFAULT
                  )
```

Python:

cv.filter2D(src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]]) -> dst

```
#include <opencv2/imgproc.hpp>
```

Convolves an image with the kernel.

The function applies an arbitrary linear filter to an image. In-place operation is supported. When the aperture is partially outside the image, the function interpolates outlier pixel values according to the specified border mode.

The function does actually compute correlation, not the convolution:

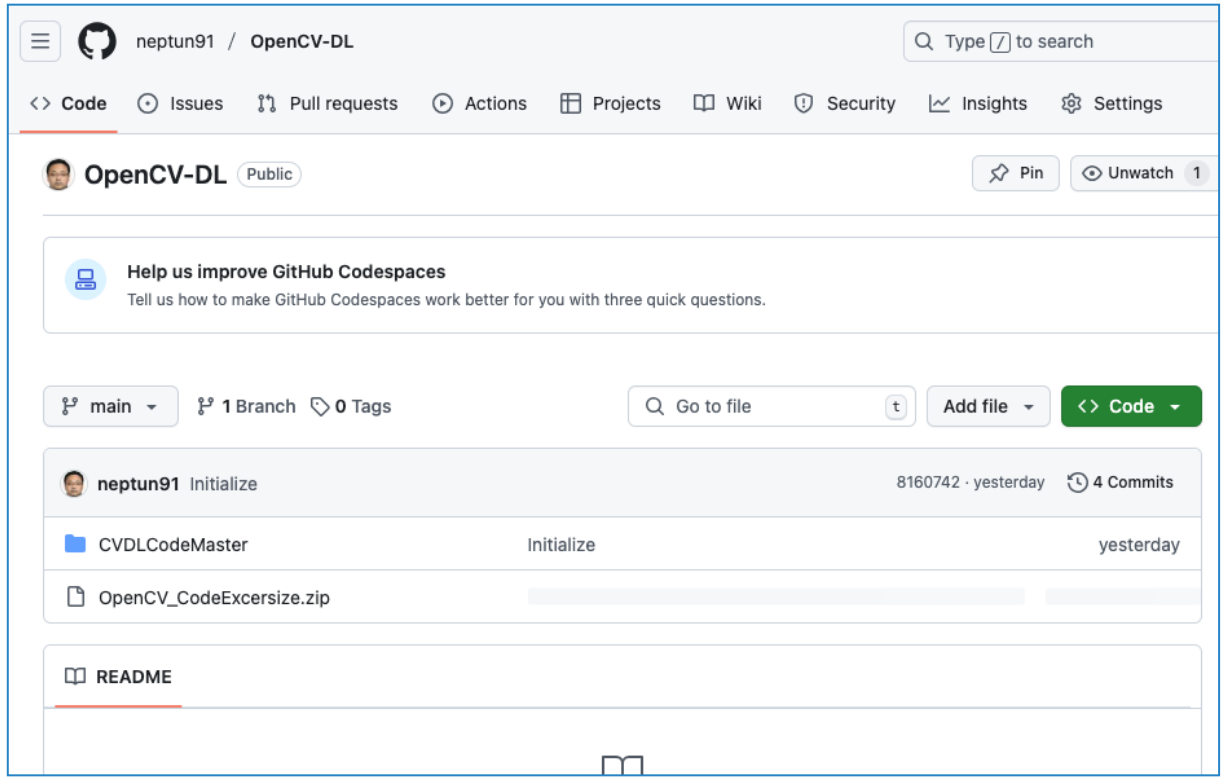
$$\texttt{dst}(x,y) = \sum_{\substack{0 \le x' < \texttt{kernel.cols} \\ 0 \le y' < \texttt{kernel.rows}}} \texttt{kernel}(x',y') * \texttt{src}(x + x' - \texttt{anchor.x}, y + y' - \texttt{anchor.y})$$

That is, the kernel is not mirrored around the anchor point. If you need a real convolution, flip the kernel using **flip** and set the new anchor to

실습 환경 구축

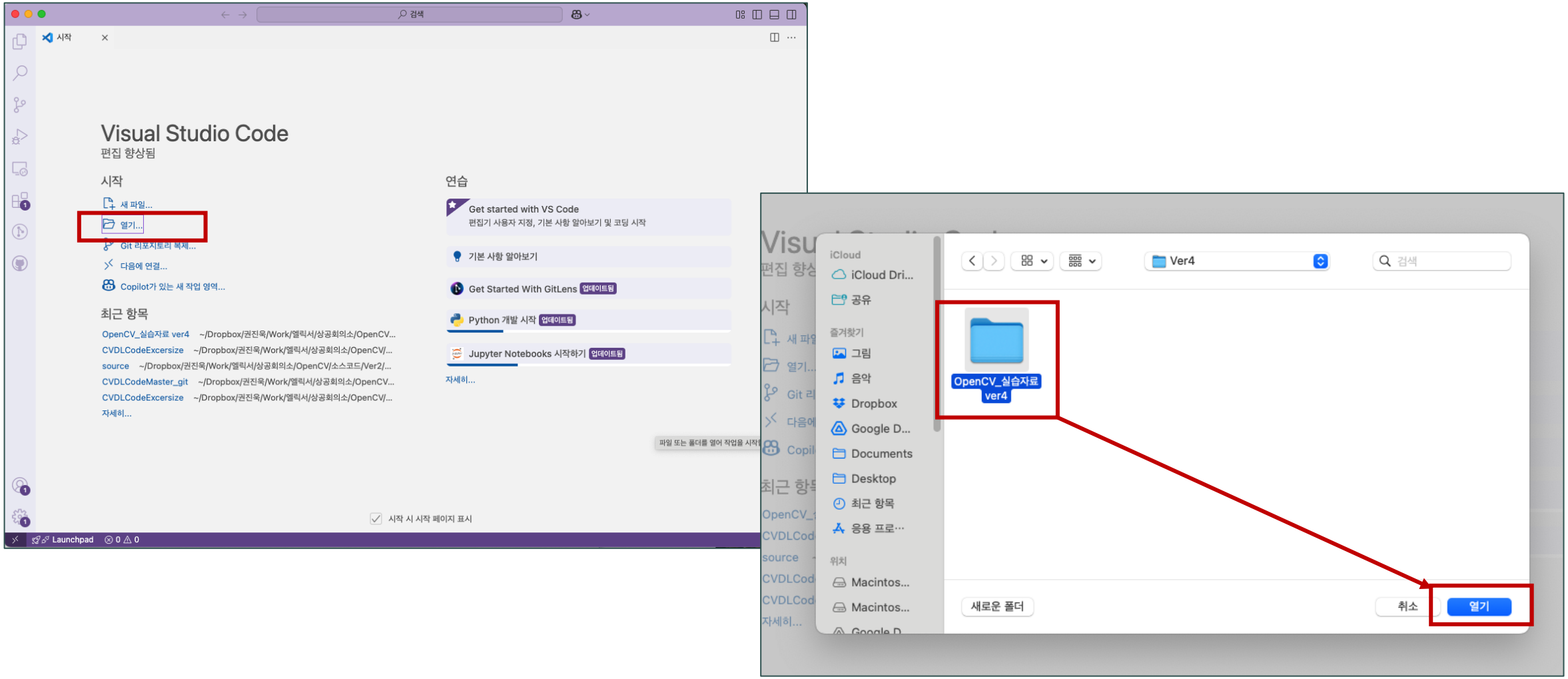
◆ 실습 파일 다운로드

- https://github.com/neptun91/OpenCV-DL 에 접속하여
- OpenCV_CodeExcersize.Ver2.zip 다운로드 후 압축 해제



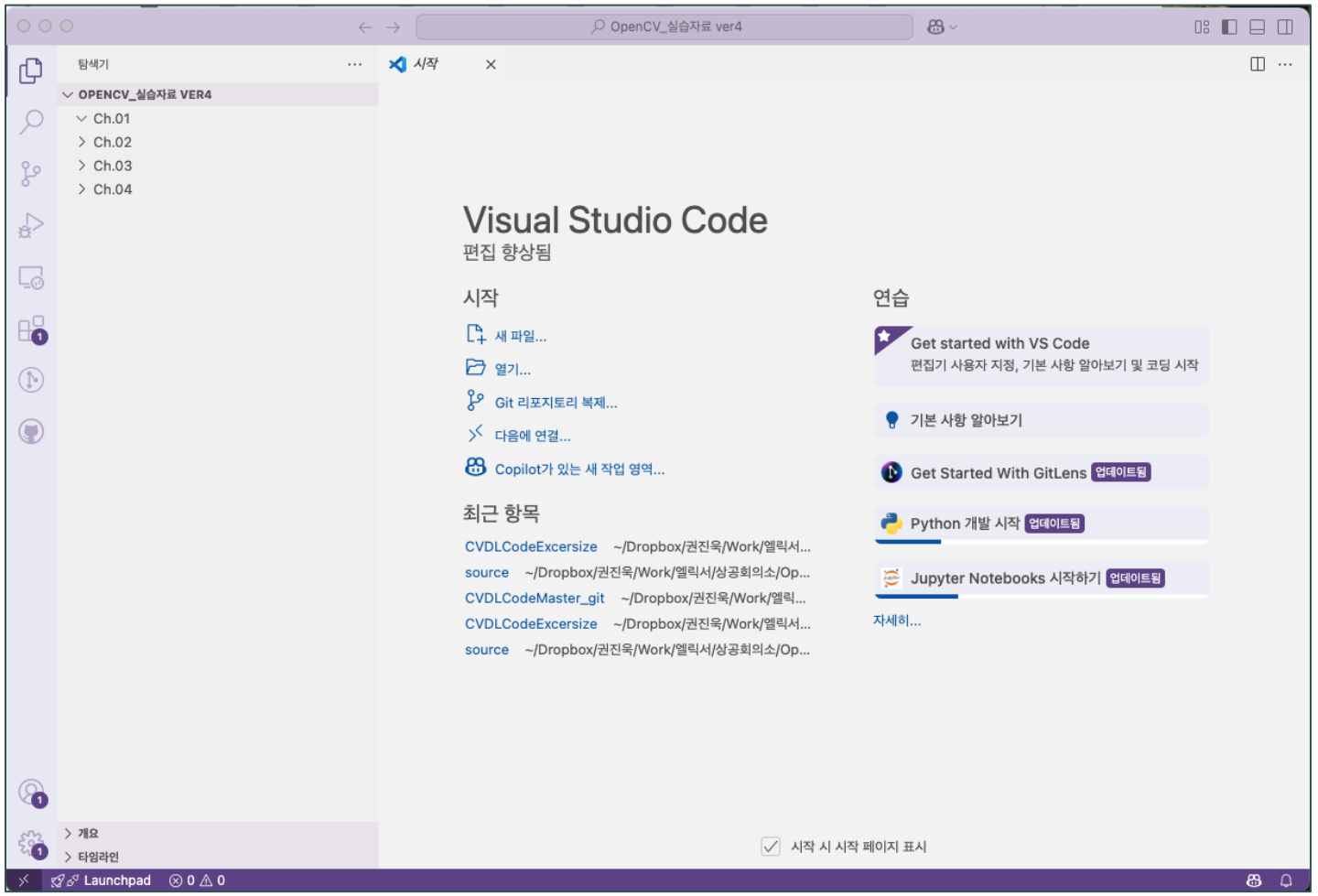
실습 환경 구축

◆ Visual Source Code에 실습파일 등록



실습 환경 구축

◆ Visual Source Code에 실습파일 등록 완료



실습 환경 구축

◆ **Ch.01₩1-1.py에 아래 코드 작성해 넣기**

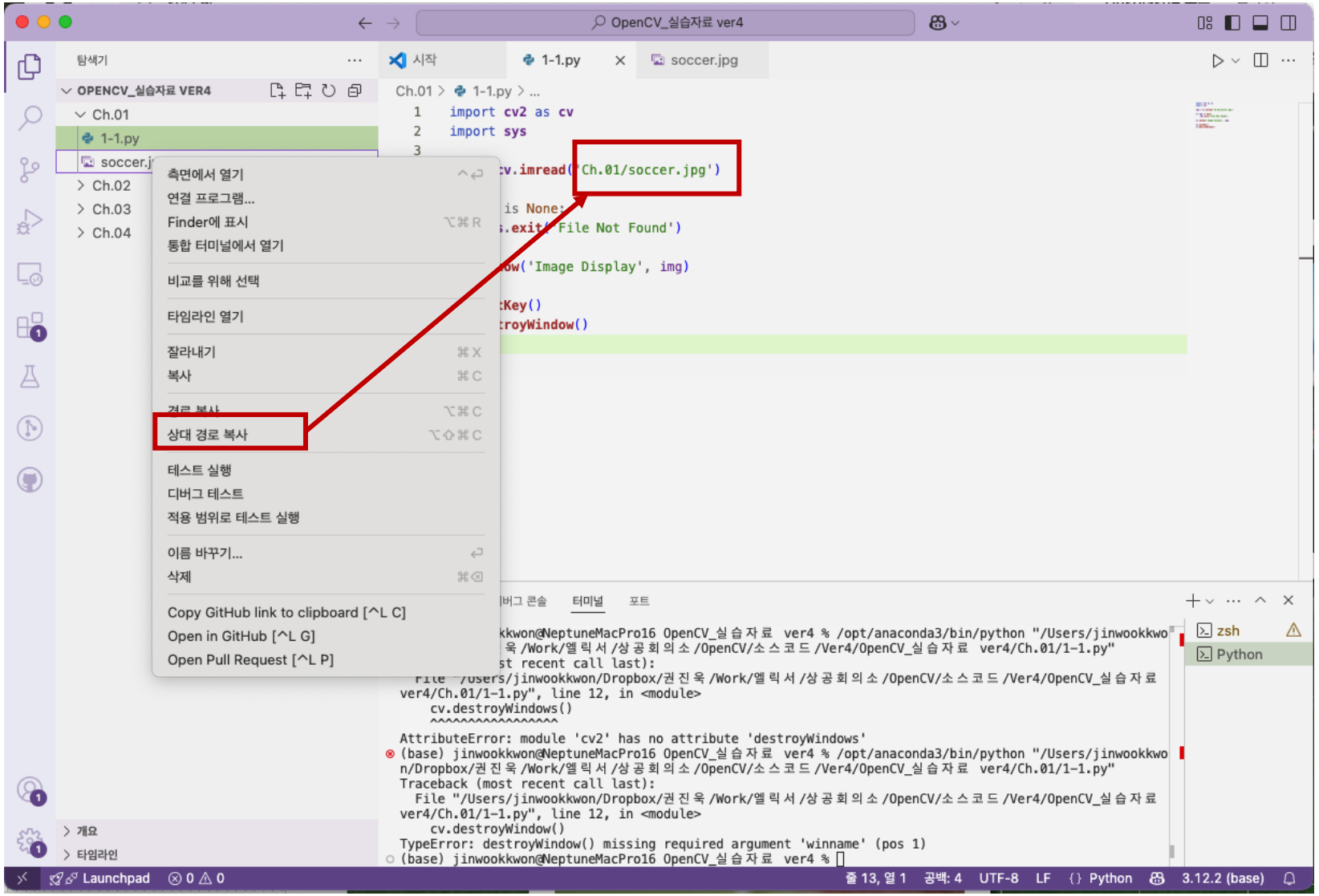
시작 | 1-1.py | soccer.jpg

Ch.01 > 1-1.py > ...

```
1   import cv2 as cv
2   import sys
3
4   img = cv.imread('Ch.01/soccer.jpg')
5
6   if img is None:
7       sys.exit('File Not Found')
8
9   cv.imshow('Image Display', img)
10
11  cv.waitKey()
12  cv.destroyWindow()
13
```

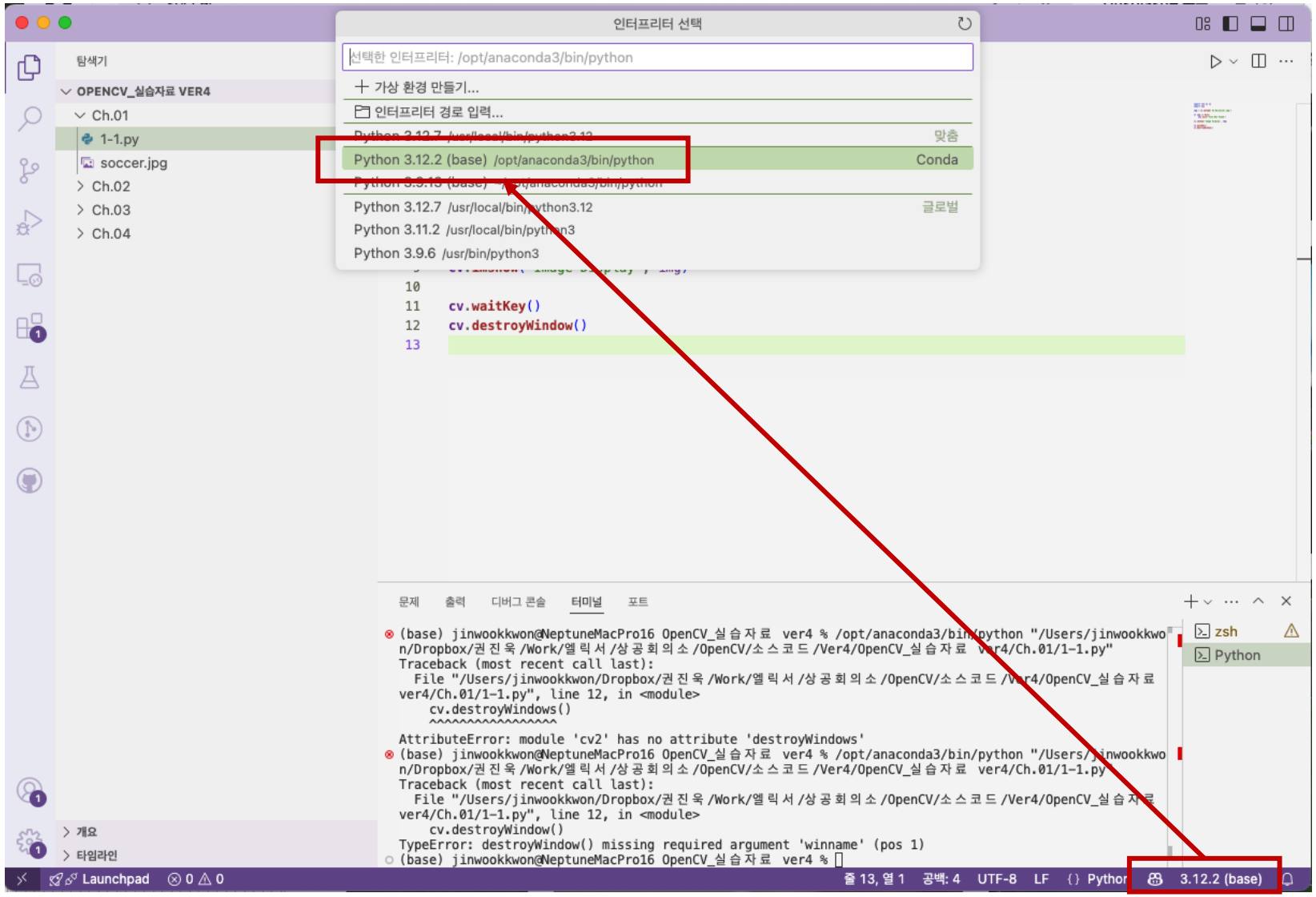
실습 환경 구축

◆ soccer.jpg를 찾지 못할 경우 상대 경로 복사해 넣기



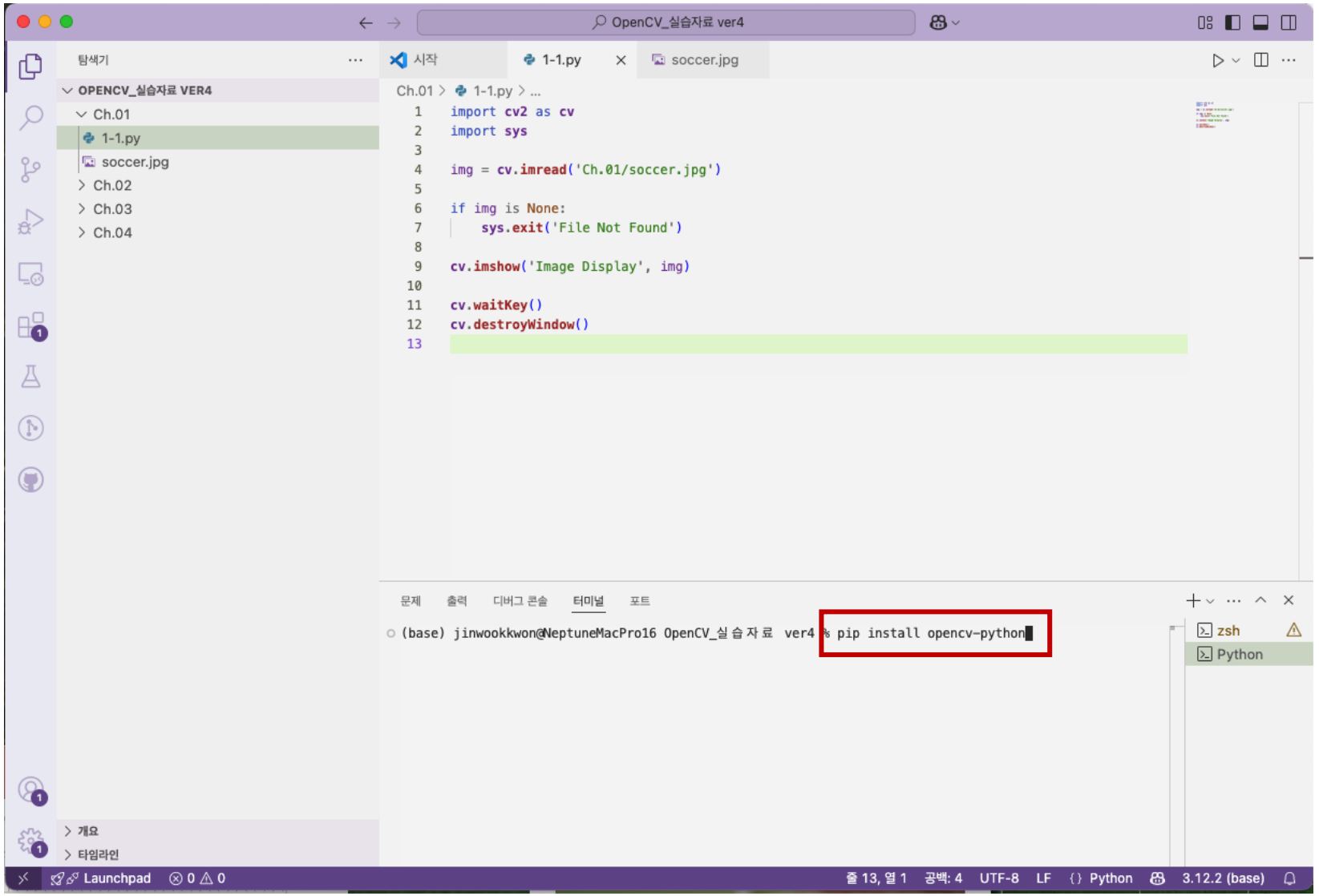
실습 환경 구축

◆ Python 실행 환경을 anaconda 기반으로 변경



실습 환경 구축

◆ OpenCV package 설치하기 – ‘pip install opencv-python’



실습 환경 구축

◆ Test program 수행