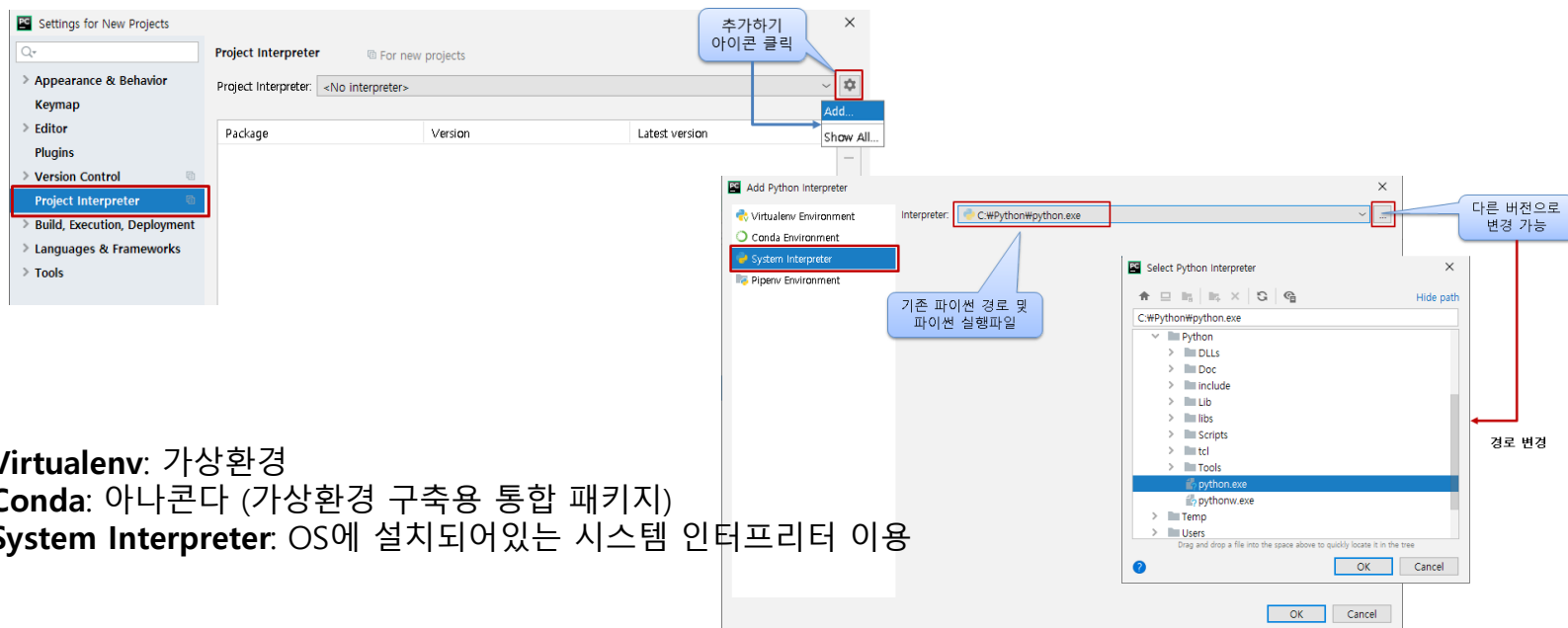


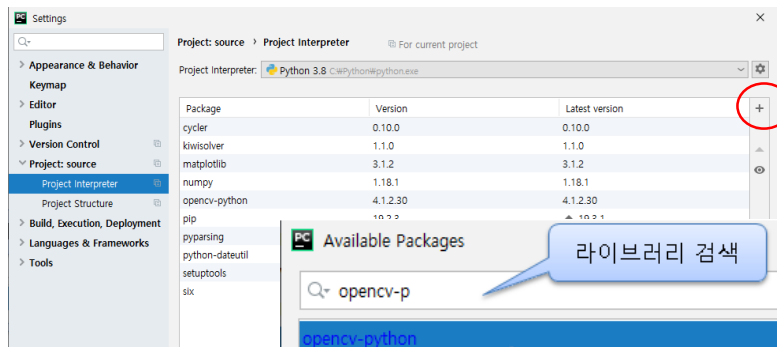
실습 07. 프로젝트 디버깅

오류 1. Python interpreter 설정

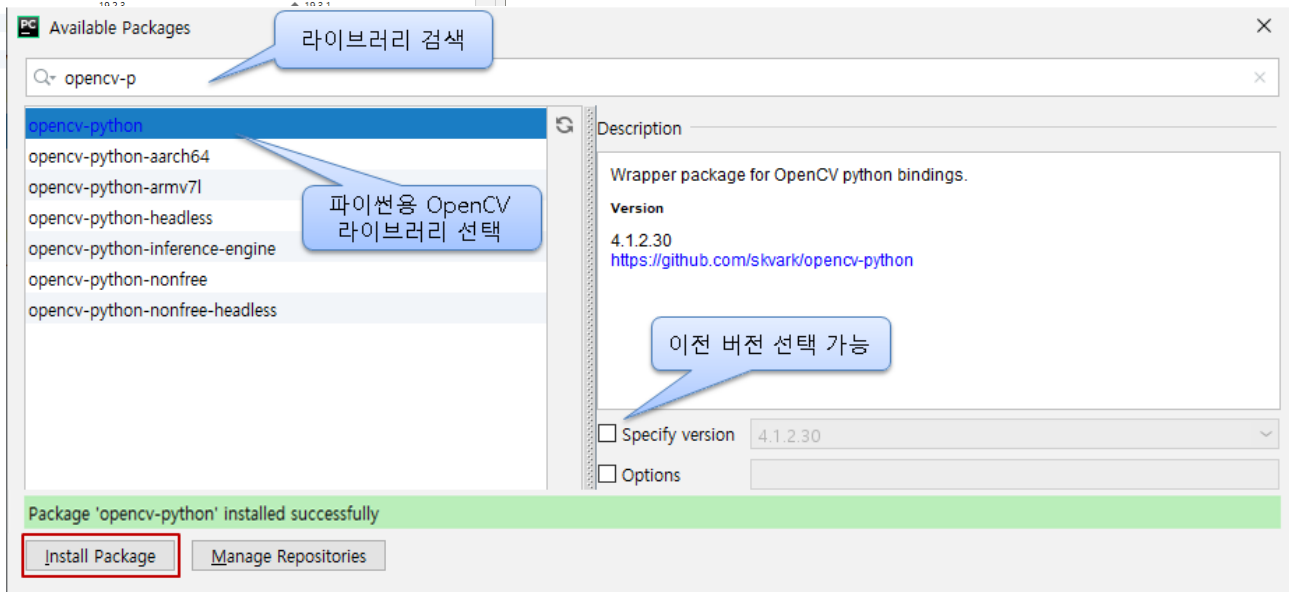
- 파이썬은 오픈소스 프로젝트로 많이 진행되며 라이브러리들이 매우 빈번하게 업데이트 됨.
- 어떤 업데이트에서는 구조가 크게 바뀌거나 인터페이스가 변경되는 경우도 있음.
- 잘 동작하던 코드가 특정 버전에서 갑자기 오류가 발생할 수 있다.
- 특정 버전을 사용하는 프로젝트를 위해, **가상환경**을 구축하여 진행할 수 있음.



오류 2. Python 패키지 설치 확인



예: opencv가 설치 되어
있지 않다면?



오류 3. 추가 파일 설명을 따르지 않음

- 지정한 파일을 다운로드 안한 경우
- 지정한 파일 대신 다른 이름의 파일을 설치한 경우

```
    return image, gray  
  
    face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_alt2.xml") # 정면 검출기  
    eye_cascade = cv2.CascadeClassifier("haarcascade_eye.xml") # 눈 검출기  
    image, gray = preprocessing(34) # 전처리  
    if image is None: raise Exception("영상 파일 읽기 에러")  
  
    faces = face_cascade.detectMultiScale(gray, 1.1, 2, 0, (100, 100)) # 얼굴 검출  
    if faces.any():  
        x, y, w, h = faces[0]
```

```
06.detect_face x  
C:\ProgramData\Anaconda3\envs\body-morp-segmentation\python.exe "D:/Dropbox/2. 한림대 강  
Traceback (most recent call last):  
  File "D:/Dropbox/2. 한림대 강의/2021-1 영상처리 프로그래밍/OpenCV-Python으로 배우는 영상처리 5  
    faces = face_cascade.detectMultiScale(gray, 1.1, 2, 0, (100, 100)); # 얼굴 검출  
cv2.error: OpenCV(4.4.0) C:\Users\appveyor\AppData\Local\Temp\1\pip-req-build-6lylwdcz  
'cv::CascadeClassifier::detectMultiScale'
```

오류 4. 소스코드 잘못 기입, 오타.

- imshow() 함수 입력 빠뜨림 → 결과 출력X
- waitKey() 함수 이름을 waitkey()로 입력 → 대소문자 오타
- face_cascade 대신 face_casacde로 입력
- 들여쓰기 잘못함 → if/else 구문의 로직이 다르게 동작: 논리적 오류

```
cv2.circle(image, center, 10, (0, 255, 0), 2)
else:
    print("눈 미검출")

cv2.rectangle(image, faces[0], (255, 0, 0), (255, 0, 0))
cv2.imshow("image", image)

else: print("얼굴 미검출")
cv2.waitKey(0)
```

따라치기 바른 예

```
cv2.circle(image, center, 10, (0, 255, 0), 2)
else:
    print("눈 미검출")

cv2.rectangle(image, faces[0], (255, 0, 0), (255, 0, 0))

else: print("얼굴 미검출")
cv2.waitKey(0)
```

따라치기 나쁜 예1

```
else:
    print("눈 미검출")

    cv2.rectangle(image, faces[0], (255, 0, 0), (255, 0, 0))
    cv2.imshow("image", image)

else: print("얼굴 미검출")
cv2.waitKey(0)]
```

따라치기 나쁜 예2

에러 메시지를 읽어보자

- 모든 사람은 오류를 낸다. (코딩의 신 조차)
- 대부분의 오류는 오타에서 발생 → **꼼꼼하게!!**
- 많은 경우 에러메시지를 읽음으로써 파악 가능
 - 에러메시지 해석
 - 구글 검색

```
import no_package
ModuleNotFoundError: No module named 'no_package'

Process finished with exit code 1
```

디버깅을 하자 (Debugging: 버그찾기)

1. 더블클릭

```
10 face_cascade = cv2.CascadeClassifier(  
11 eye_cascade = cv2.CascadeClassifier(  
12 image, gray = preprocessing(gray, cv2.COLOR_BGR2GRAY))  
13 if image is None: raise Exception(  
14  
15 faces = face_cascade.detectMultiScale(  
16 if faces.any():  
17     x, y, w, h = faces[0]  
18     face_image = image[y:y+h, x:x+w]  
19     eyes = eye_cascade.detectMultiScale(  
20     if len(eyes) > 0:
```

2. 디버그 버튼 클릭

3. 디버그 창: 코드 라인별 실행

4. 변수 값 확인

- 코드를 검증하고, 버그를 찾는데 아주 유용한 기능.
- 코드가 생각대로 동작하는지 라인별로 확인해 봅시다.

디버깅을 하자 (Debugging: 버그찾기)

코드 옆 더블클릭
(브레이크 포인트)

```
10 face_cascade = cv2.CascadeCla
11 eye_cascade = cv2.CascadeClas
12 image, gray = preprocessing(3
13 if image is None: raise Excep
14
15 faces = face_cascade.detectMu
16 if faces.any():
17     x, y, w, h = faces[0]
18     face_image = image[y:y +
19     eyes = eye_cascade.detect
20     if len(eyes) == 2: # 눈
```

계속 진행 버튼
(브레이크 포인트까지)

다음 줄 실행

함수 내부 실행

함수 호출 스택

The screenshot shows the 'Debug Console' window in an IDE. The 'Call Stack' (함수 호출 스택) is visible, listing the following frames from top to bottom: 'MainThread', 'forward, fullyconnected.py:12' (highlighted in blue), 'train_step, network.py:18', and '<module>, mnist_test.py:84'. To the right of the call stack is the 'Variables' panel, which shows 'inputs =' and 'self = {F'.

디버깅을 하자 (Debugging: 버그찾기)

마우스 커서를 가져다 대면, 변수의 상세 내용을 확인할 수 있다.

```
17
18
19 eyes = eye_cascade.detectMultiScale(face_image, 1.1,
20 if len(eyes) == 2: # 눈 2개가 검출되면
21     for ex, ey, ew, eh in eyes: ex: 82 ey: 30 ew: 45 eh: 45
22     center = (x + ex + ew // 2, y + ey + eh // 2)
23     + [ndarray: (2, 4)] [[82 30 45 45], [25 47 41 41]] (0),
24 else:
25     print("눈 미검출")
26
27     cv2.rectangle(image, faces[0], (255, 0, 0), 2)
28     cv2.imshow("image", image)
29
30 else:
```

현재 코드상의 변수 목록

Variables

>	ew	=	{int32} 45
>	ex	=	{int32} 82
>	ey	=	{int32} 30
>	eye_cascade	=	{CascadeClassifier} <CascadeClassifier 0000015B2A206F5

현재 코드상의 변수를 사용한 임시 코드 실행

```
20 if len(eyes) == 2: # 눈 2개가 검출되면
21     for ex, ey, ew, eh in eyes: ex: 82 ey: 30 ew: 45 eh: 45
22     center = (x + ex + ew // 2, y + ey + eh // 2)
23     + [ndarray: (2, 4)] [[82 30 45 45], [25 47 41 41]] (0),
24 else:
25     print("눈 미검출")
26
27     cv2.rectangle(image, faces[0], (255, 0, 0), 2)
28     cv2.imshow("image", image)
29
30 else:
```

CV Evaluate

Expression: `ex+ey`

Result:

>	result	=	{int32} 112
>	T	=	{int32} 112
>	base	=	{NoneType} None
>	data	=	{memoryview: 1} <memoryview of bytearray object>
>	denominator	=	{int} 1
>	dtype	=	{dtype: 0} int32
>	flags	=	{flags: 0} C_CONTIGUOUS

코드 분석 및 디버깅 연습 실습

- 프로젝트 설치 후 코드를 실행하시오. (학습에 2~3분 시간 소요)
 - mnist_test.py 실행
 - Line 31의 layers 리스트의 역할은 무엇인가?
- Fully connected layer를 한줄 추가해 보시오.
- 첫번째 layer의 입력과 출력 (z)의 shape를 살펴보고 그 의미를 서술하시오.
 1. mnist_test.py의 31번째 줄의 layer 리스트 살펴보기: 첫번째와 두번째 layer의 역할은?
 2. Nn/network.py의 18번째줄에 break point (빨간점)을 찍고, 디버깅을 실행
 3. Step over(단축키 F8)를 눌러보고 입력과 출력 z에 대해 살펴보기
 4. Step into(단축키 F7)를 눌러보고 함수 호출 스택 살펴보기