

[2018년 여름]  
**Tensorflow를 이용한  
Deep Learning의 기초**

**제1강: 신경망 개념과 환경 구축**

부경대학교 IT융합응용공학과  
권오흠

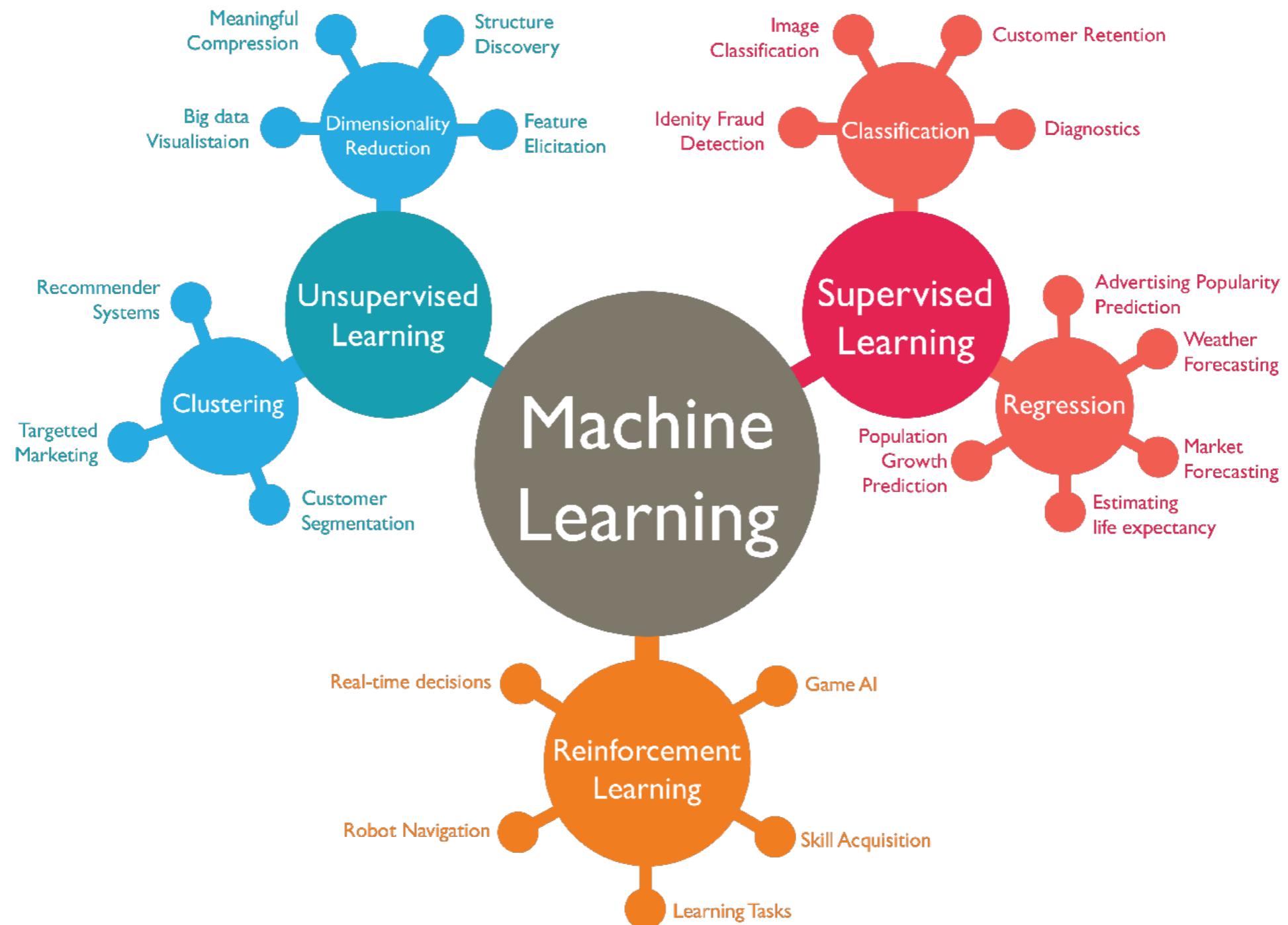


## 개요

- 신경망(Neural Network)을 중심으로
- Tensorflow **Low-Level API**를 중심으로
- 최소한의 **수학**...
- 예제**를 통해서...

## 강좌 일정

- 신경망(neural network) 개념과 실습 환경 구축
- Tensorflow를 이용한 기본적인 Network 구현
- Tensorflow Low-Level API 살펴보기
- 합성곱(Convolutional) 신경망
- DataSet API와 High Level API



# **Gradient Descent**

## 경사 하강법

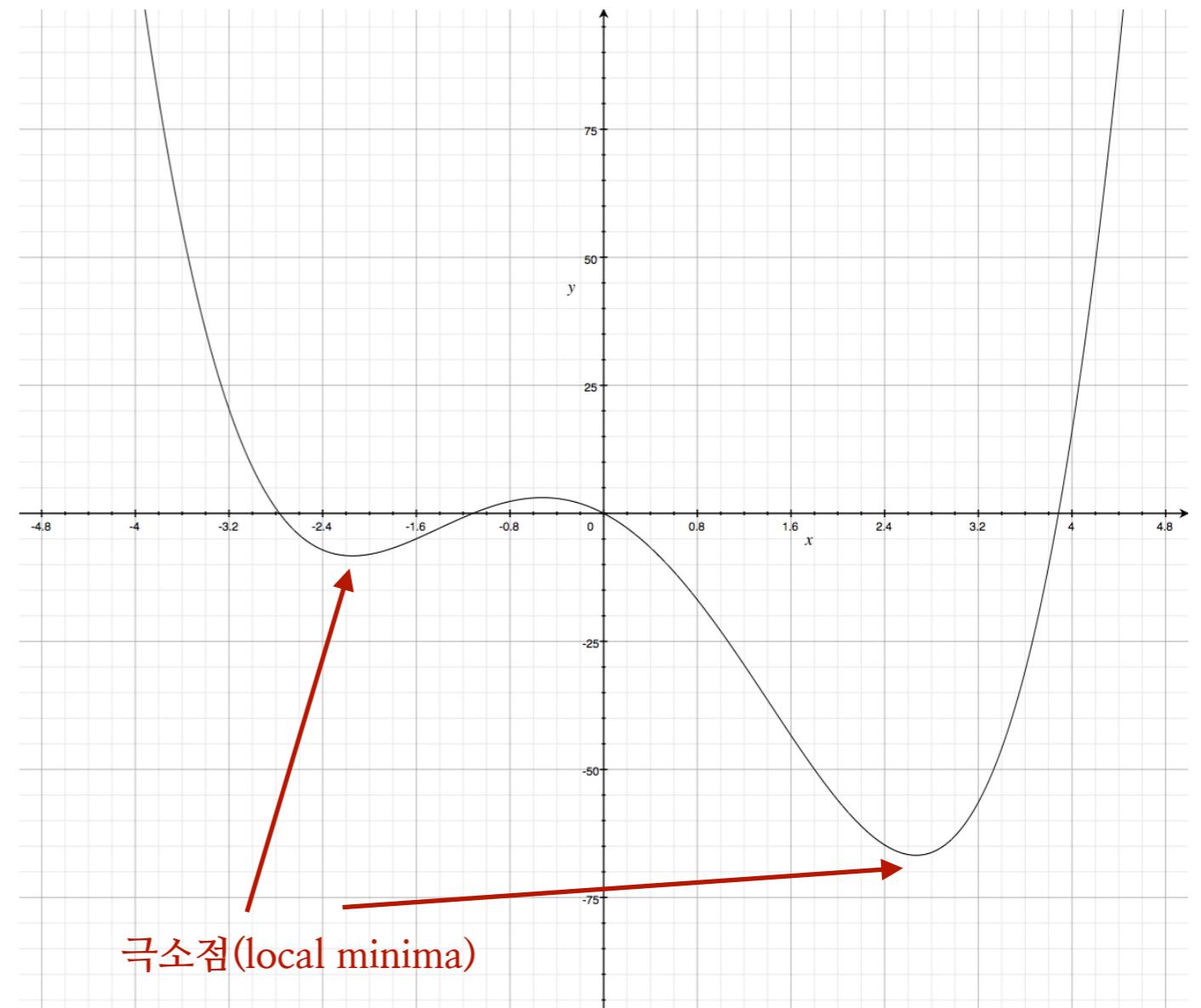
다음의 함수  $f(x)$ 의 최소값은?

$$f(x) = x^4 - 12x^2 - 12x$$

$$f'(x) = 4x^3 - 24x - 12 = 0$$

$$x = ?$$

미분계수가 0이 되는  $x$ 값을  
항상 대수적으로 구할수 있는 것은 아니다.



# Gradient Descent Method

Let  $x_0 = 4$  (random choice)

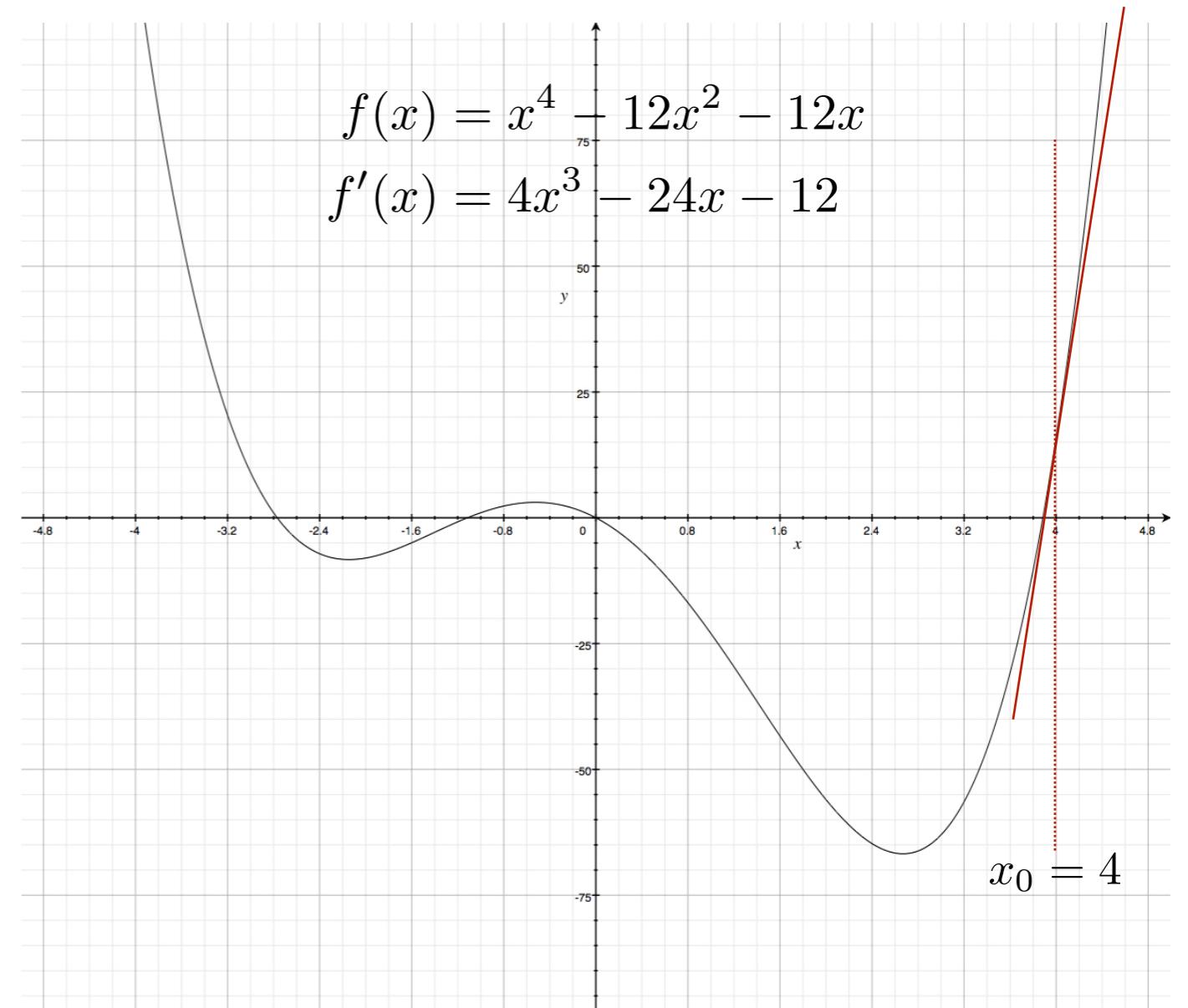
$$f'(4) = 4 \cdot 4^3 - 24 \cdot 4 - 12 = 148.0$$

함수  $f(x)$ 는  $x=4$ 에서 증가상태이다.  
따라서  $x=4$ 에서 왼쪽으로 이동하면  
함수값은 감소한다.  
따라서  $x=4$ 보다 왼쪽의 적절한 값을  
 $x_1$ 으로 선택한다.

$$x_1 = x_0 - \rho f'(x_0)$$



$\rho$ 는 적당한 상수(예를 들어 0.005), 기울기가 가파르면 많이 이동하고  
기울기가 완만하면(0에 가까우면) 극소점 근처일 가능성성이 있으므로 조금만 이동한다.



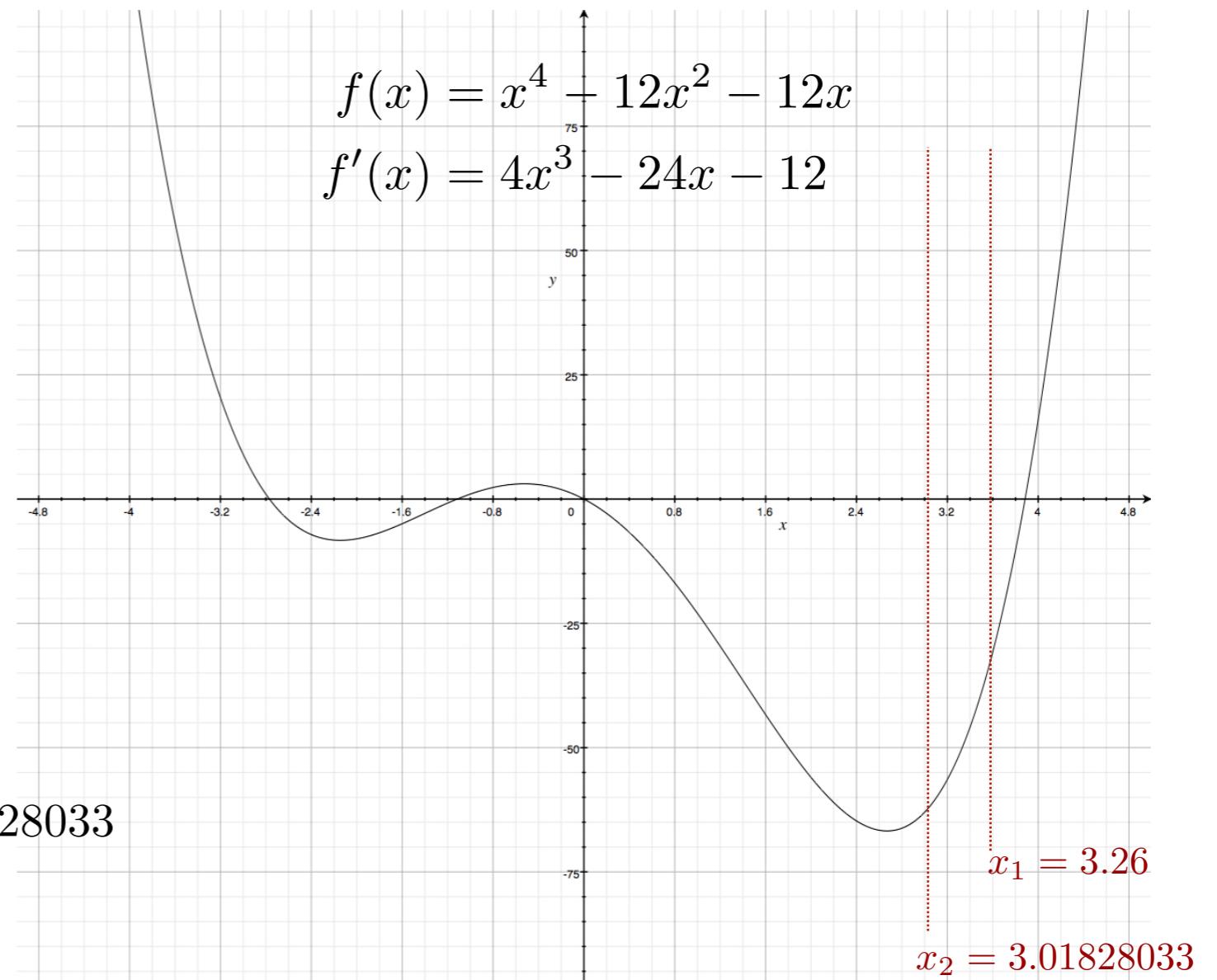
# Gradient Descent Method

$\rho$ 는 적절한 작은 상수

$$\begin{aligned}x_1 &= x_0 - \rho f'(x_0) \\&= 4.0 - 0.005 \cdot 148.0 = 3.26.\end{aligned}$$

$$f'(3.26) = 48.343904$$

$$\begin{aligned}x_2 &= x_1 - \rho f'(x_1) \\&= 3.26 - 0.005 \cdot 48.343904 = 3.01828033\end{aligned}$$



# Gradient Descent Method

```
from random import random
```

```
def f(x):
    return x**4 - 12.0*x**2 - 12.0*x

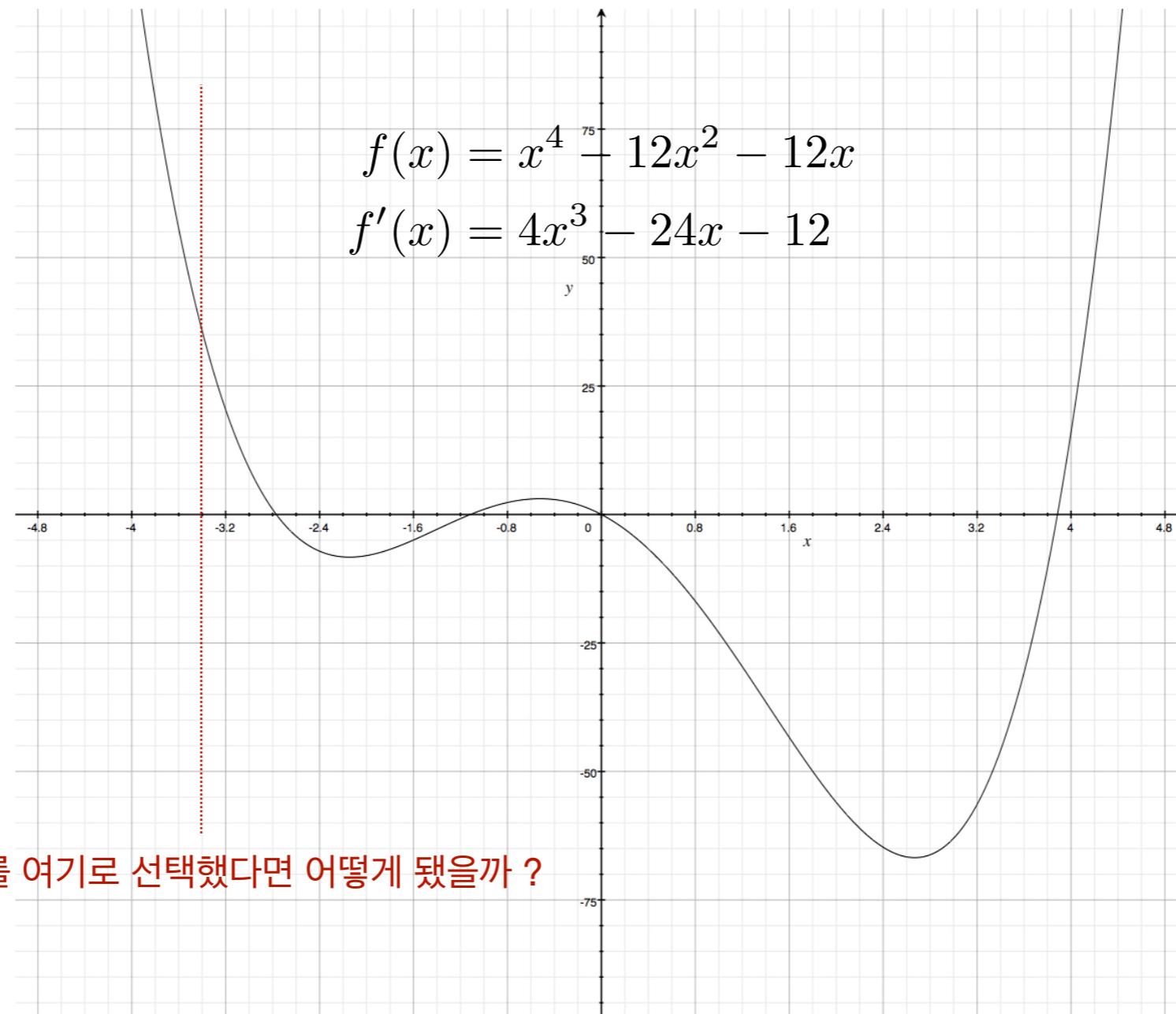
def df(x):
    return 4 * x**3 - 24 * x - 12

rho = 0.005
precision = 0.000000001
difference = 100
x = random()
# x = 4.0

while difference > precision:
    dr = df(x)
    prev_x = x
    x = x - rho * dr
    difference = abs(prev_x - x)
    print("x = {:.f}, df = {:.10.6f}, f(x) = {:.f}".format(x, dr, f(x)))
```

```
x = 0.507647, df = -21.362430, f(x) = -9.117818
x = 0.625948, df = -23.660235, f(x) = -12.059596
x = 0.756157, df = -26.041743, f(x) = -15.608237
x = 0.898249, df = -28.418364, f(x) = -19.810186
x = 1.051544, df = -30.658959, f(x) = -24.664778
x = 1.214474, df = -32.586094, f(x) = -30.097584
x = 1.384385, df = -33.982236, f(x) = -35.937832
x = 1.557447, df = -34.612424, f(x) = -41.913333
x = 1.728785, df = -34.267496, f(x) = -47.677470
x = 1.892903, df = -32.823581, f(x) = -52.873332
x = 2.044403, df = -30.299973, f(x) = -57.218908
x = 2.178836, df = -26.886671, f(x) = -60.576838
x = 2.293423, df = -22.917486, f(x) = -62.973170
x = 2.387376, df = -18.790455, f(x) = -64.558244
x = 2.461721, df = -14.869031, f(x) = -65.537041
...
...
```

# Gradient Descent Method



만약 처음에  $x_0$ 를 여기로 선택했다면 어떻게 됐을까 ?

**Gradient descent 알고리즘은 local minimum에 빠질 수 있다.**  
즉 항상 최소값을 찾는 것은 아니다.

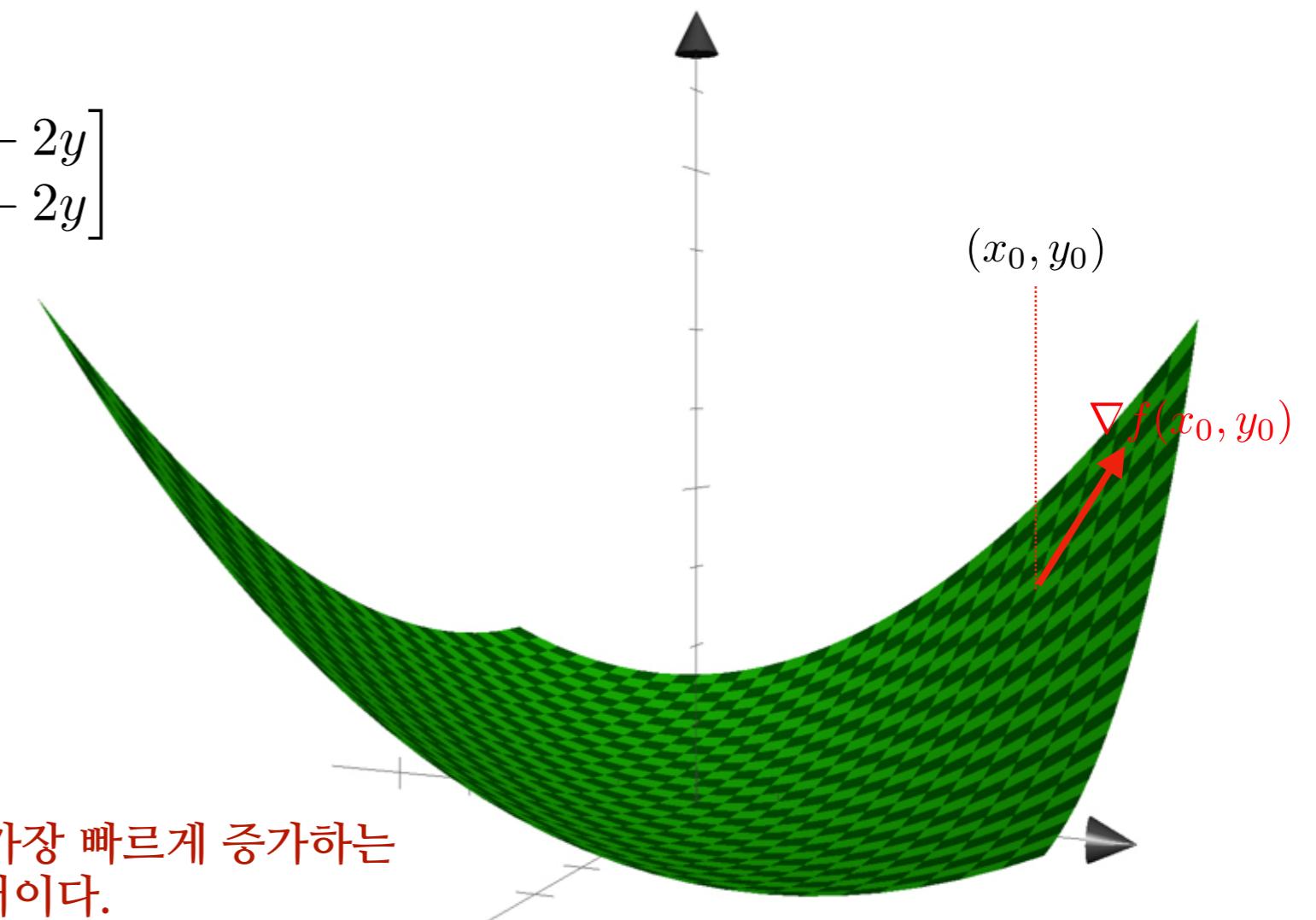
- 다음과 같이 2개의 변수를 가지는 함수의 최소값은?

$$f(x, y) = 2x^2 + 2xy + y^2$$

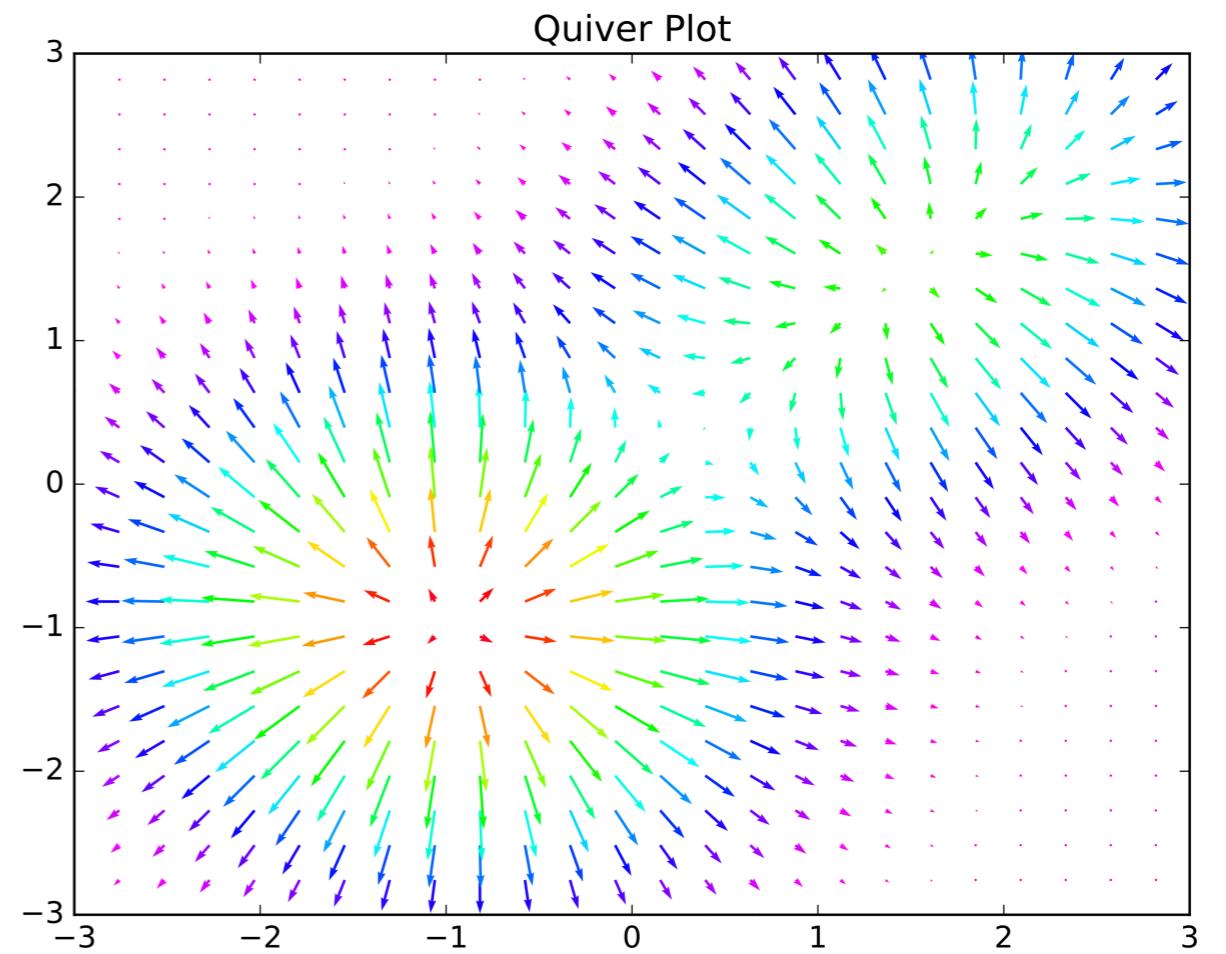
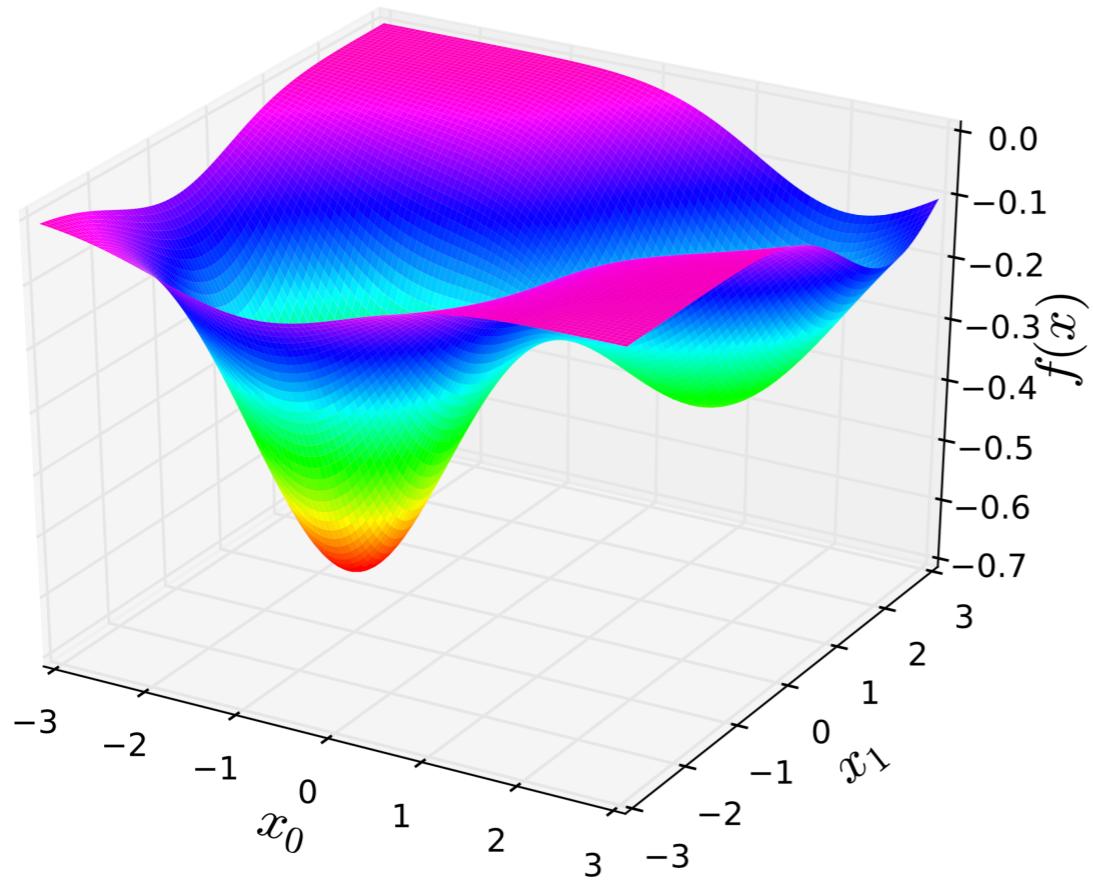
$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial}{\partial x} f(x, y) \\ \frac{\partial}{\partial y} f(x, y) \end{bmatrix} = \begin{bmatrix} 4x + 2y \\ 2x + 2y \end{bmatrix}$$

↑  
gradient  
(편미분들의 집합)

$\nabla f(x_0, y_0)$ 는  $(x_0, y_0)$ 에서 함수값이 가장 빠르게 증가하는 방향과 증가하는 정도를 나타내는 벡터이다.

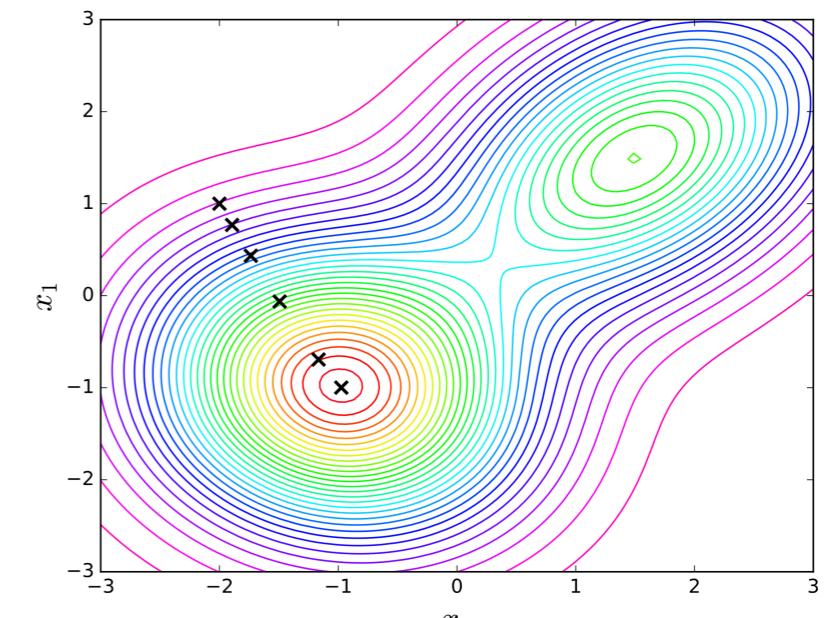
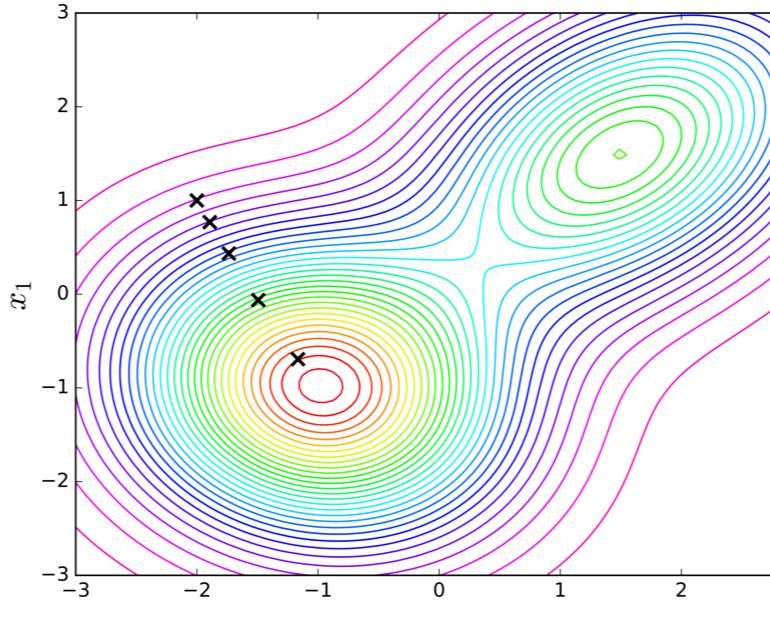
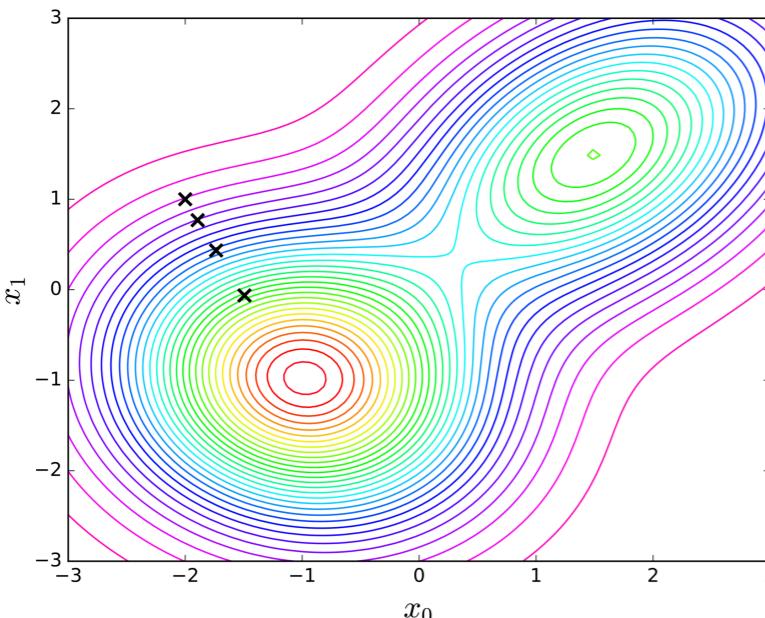
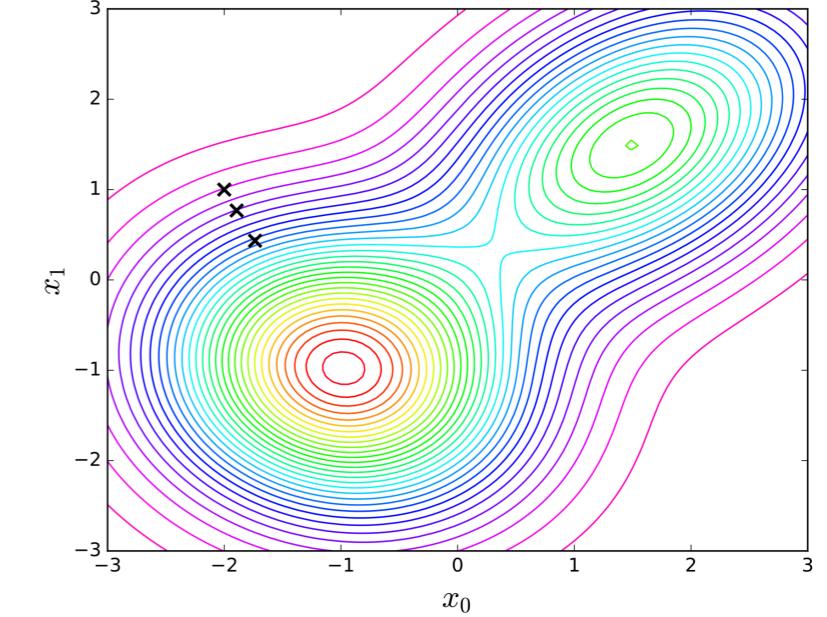
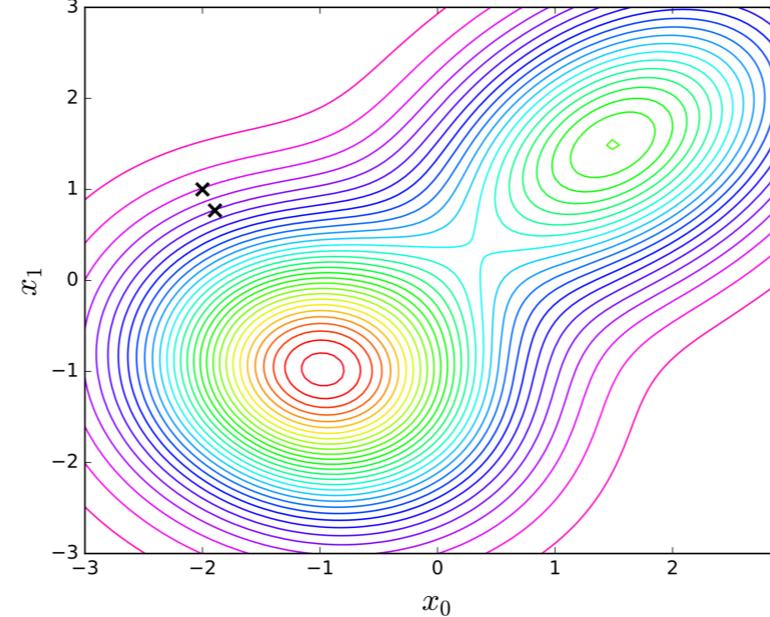
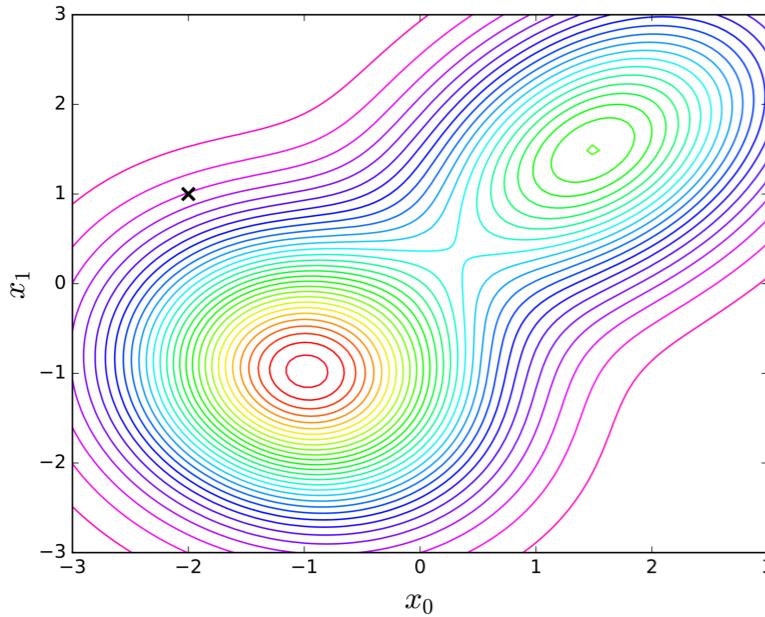


# Gradient Descent Method



Gradient는 함수값이 가장 빠르게 증가하는 방향과  
증가하는 정도를 나타낸다.

# Gradient Descent Method



## Gradient Descent Method

$$\mathbf{x}^{\text{new}} = \mathbf{x}^{\text{old}} - \rho \cdot \nabla_{\mathbf{x}} f(\mathbf{x}^{\text{old}})$$

Current value of  $\mathbf{x}$

Step size

Gradient

The diagram illustrates the gradient descent update rule. The equation  $\mathbf{x}^{\text{new}} = \mathbf{x}^{\text{old}} - \rho \cdot \nabla_{\mathbf{x}} f(\mathbf{x}^{\text{old}})$  is displayed. Three arrows point to the terms: one from "Current value of  $\mathbf{x}$ " to  $\mathbf{x}^{\text{old}}$ , one from "Step size" to  $\rho$ , and one from "Gradient" to  $\nabla_{\mathbf{x}} f(\mathbf{x}^{\text{old}})$ .

# Gradient Descent Method

```
import numpy as np

def f(x):
    return 2*x[0]**2 + 2*x[0]*x[1] + x[1]**2

def df(x):
    dx = 4*x[0] + 2*x[1]
    dy = 2*x[0] + 2*x[1]
    return np.array([dx, dy])

rho = 0.005
precision = 0.0000001
difference = 100
x = np.random.rand(2)

while difference > precision:
    dr = df(x)
    prev_x = x
    x = x - rho * dr
    difference = np.dot(x-prev_x, x-prev_x)
    print("x = {}, df = {}, f(x) = {:.f}"
          .format(np.array2string(x), np.array2string(dr), f(x)))
```

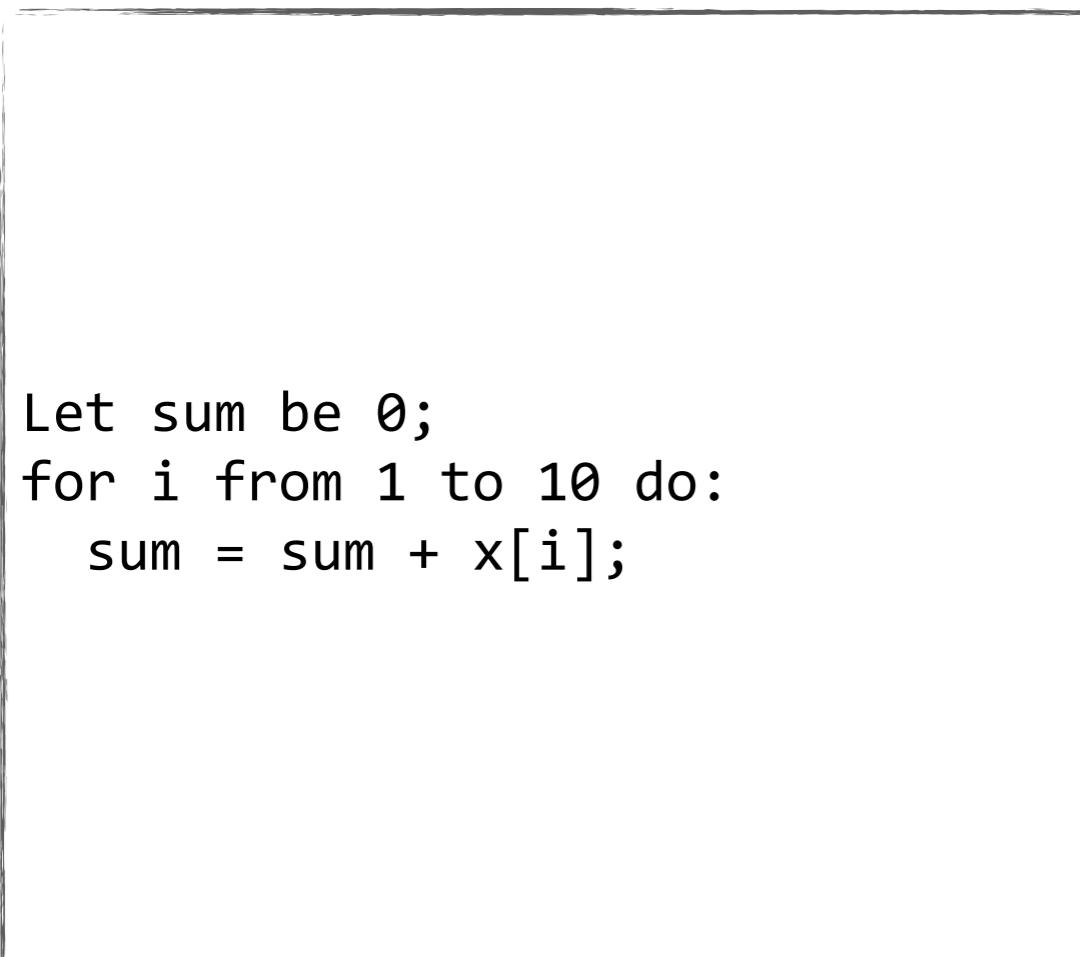
```
...
x = [-0.02640796  0.06073794], df = [ 0.01687681  0.06952397], f(x) = 0.001876
x = [-0.02648718  0.06039464], df = [ 0.01584404  0.06865996], f(x) = 0.001851
x = [-0.02656139  0.06005557], df = [ 0.01484056  0.06781492], f(x) = 0.001827
x = [-0.02663071  0.05972063], df = [ 0.0138656   0.06698837], f(x) = 0.001804
x = [-0.02669531  0.05938973], df = [ 0.0129184   0.06617983], f(x) = 0.001782
x = [-0.0267553   0.05906278], df = [ 0.01199824  0.06538885], f(x) = 0.001760
x = [-0.02681082  0.05873971], df = [ 0.01110438  0.06461498], f(x) = 0.001738
x = [-0.026862    0.05842042], df = [ 0.01023614  0.06385778], f(x) = 0.001718
x = [-0.02690896  0.05810484], df = [ 0.00939284  0.06311684], f(x) = 0.001697
x = [-0.02695183  0.05779288], df = [ 0.00857382  0.06239175], f(x) = 0.001678
```

# **Neural Network**

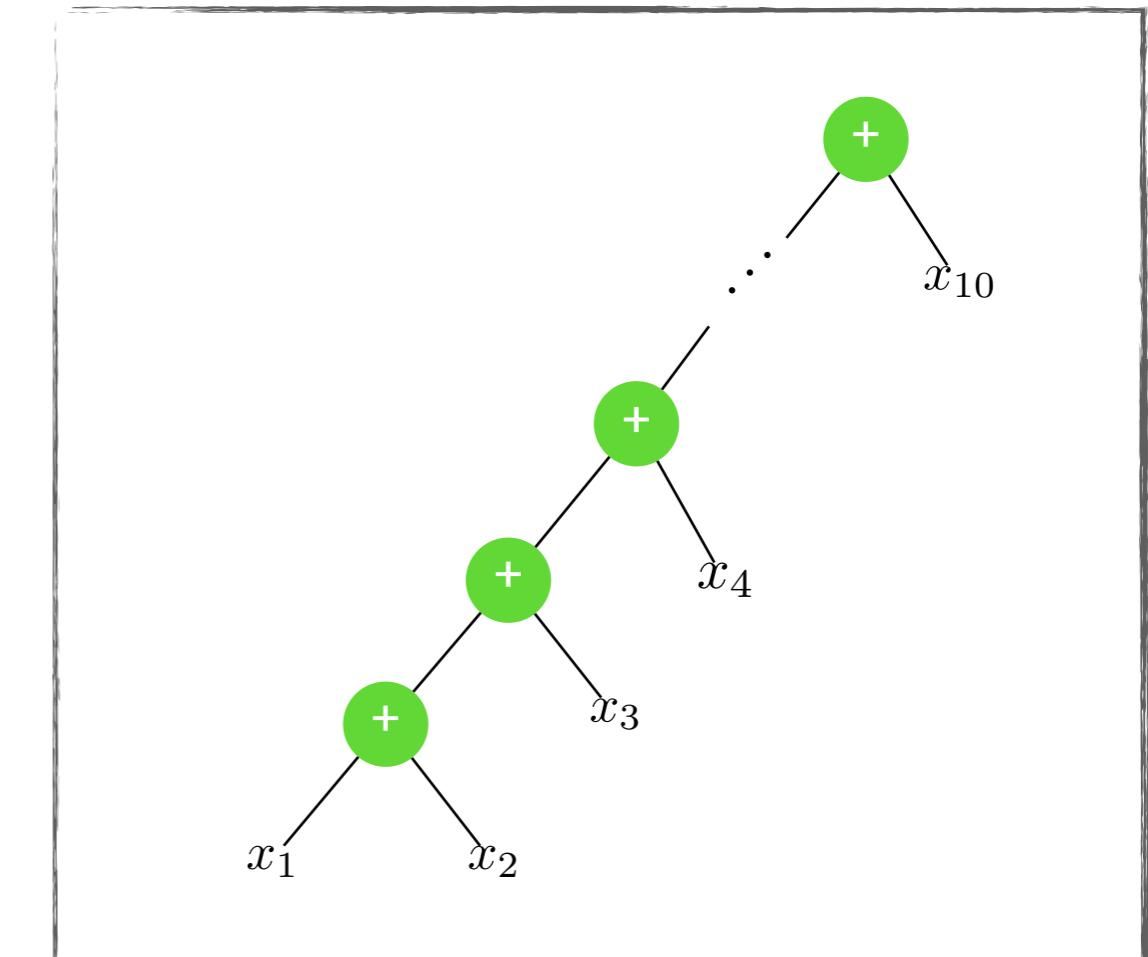
## 신경망

## 계산의 과정을 기술하는 대표적인 두 가지 방법

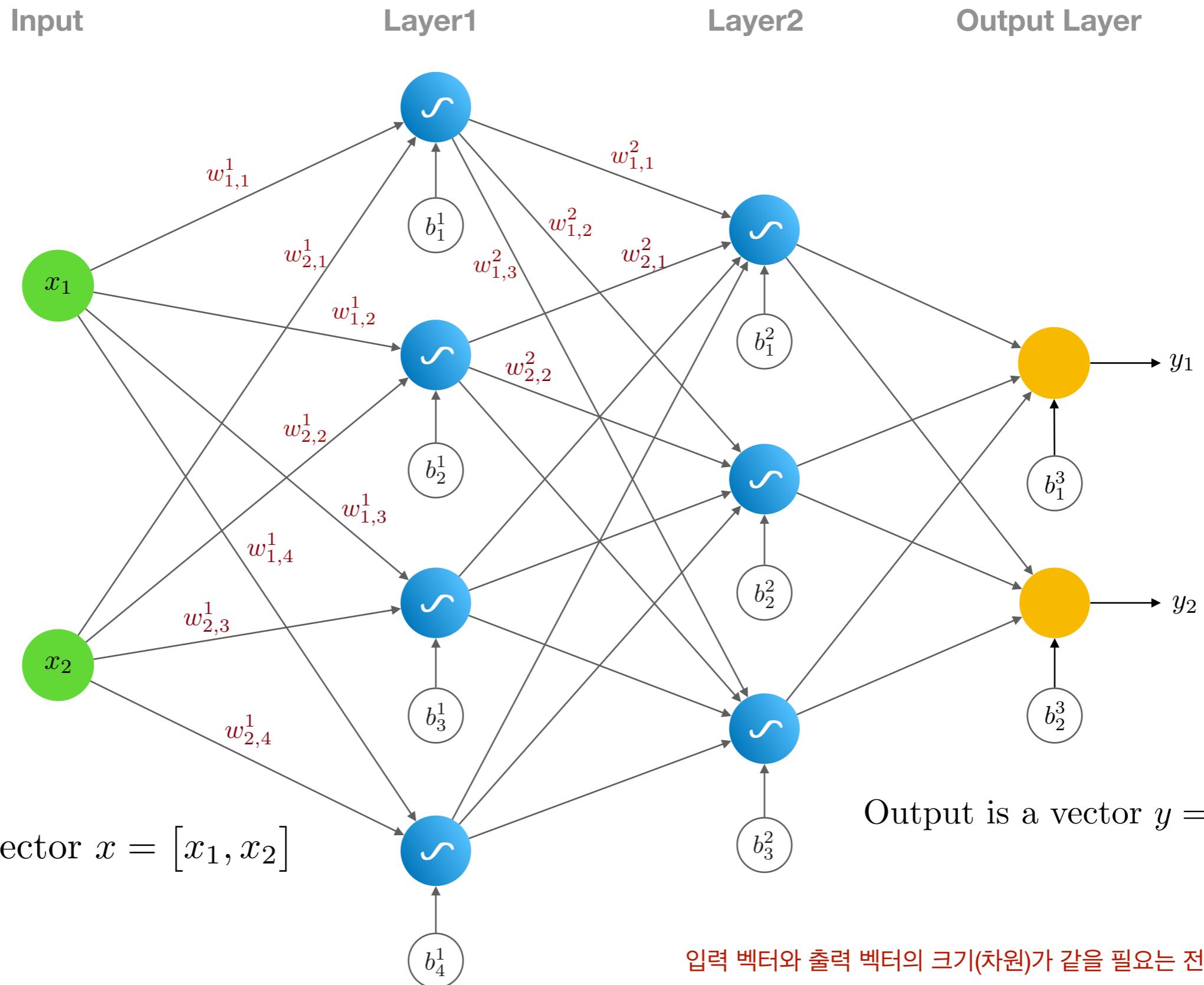
$$S = \sum_{i=1}^{10} x_i$$



Procedure

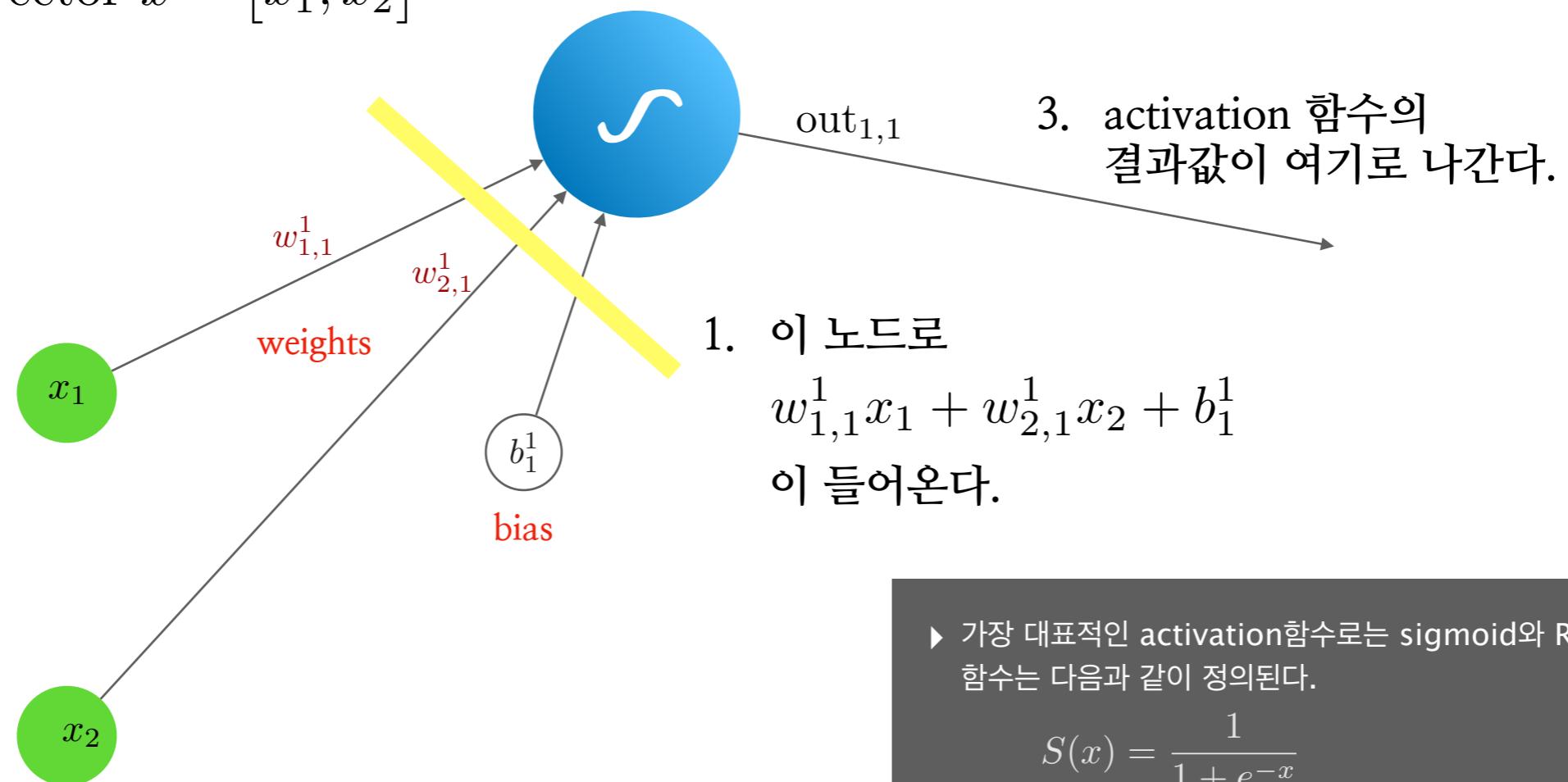


Network, Graph, Data Flow Machine 등



# How does it work?

Input is a vector  $x = [x_1, x_2]$



▶ 가장 대표적인 activation 함수는 sigmoid와 Relu가 있다. Sigmoid 함수는 다음과 같이 정의된다.

$$S(x) = \frac{1}{1 + e^{-x}}$$

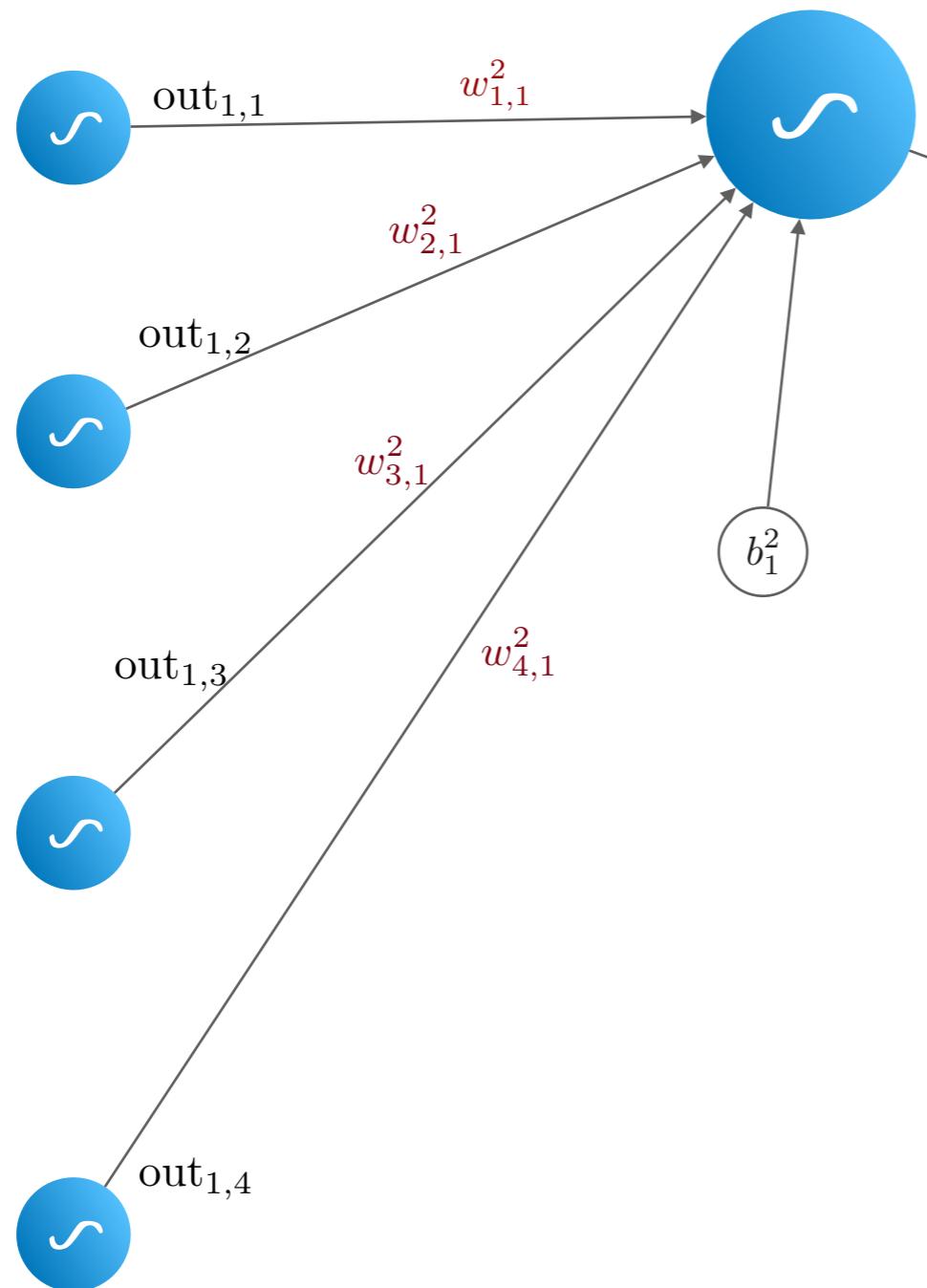
▶ 따라서, 이 노드의 출력값은 다음과 같다.

$$\text{out}_{1,1} = \frac{1}{1 + e^{-(w_{1,1}^1 x_1 + w_{1,2}^1 x_2 + b_1^1)}}$$

▶ Activation 함수를 적용하는 이유는?

# What is Neural Network ?

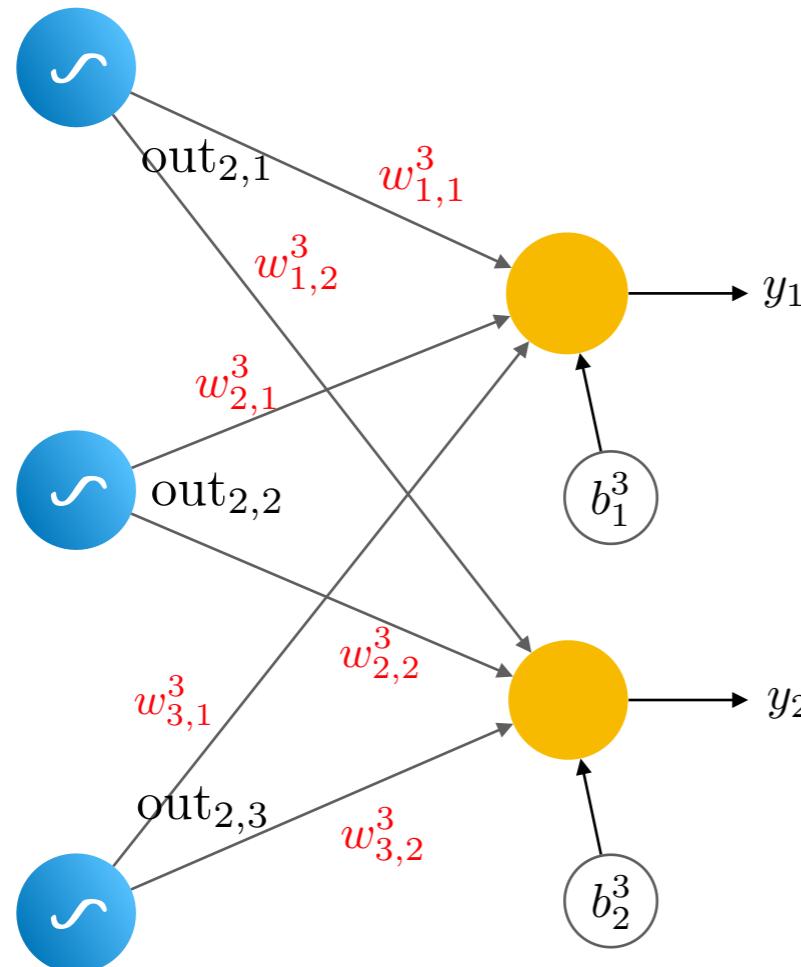
Layer1



Layer2

$$out_{2,1} = \frac{1}{1 + e^{-(w_{1,1}^2 out_{1,1} + w_{2,1}^2 out_{1,2} + w_{3,1}^2 out_{1,3} + w_{4,1}^2 out_{1,4} + b_1^2)}}$$

Layer2

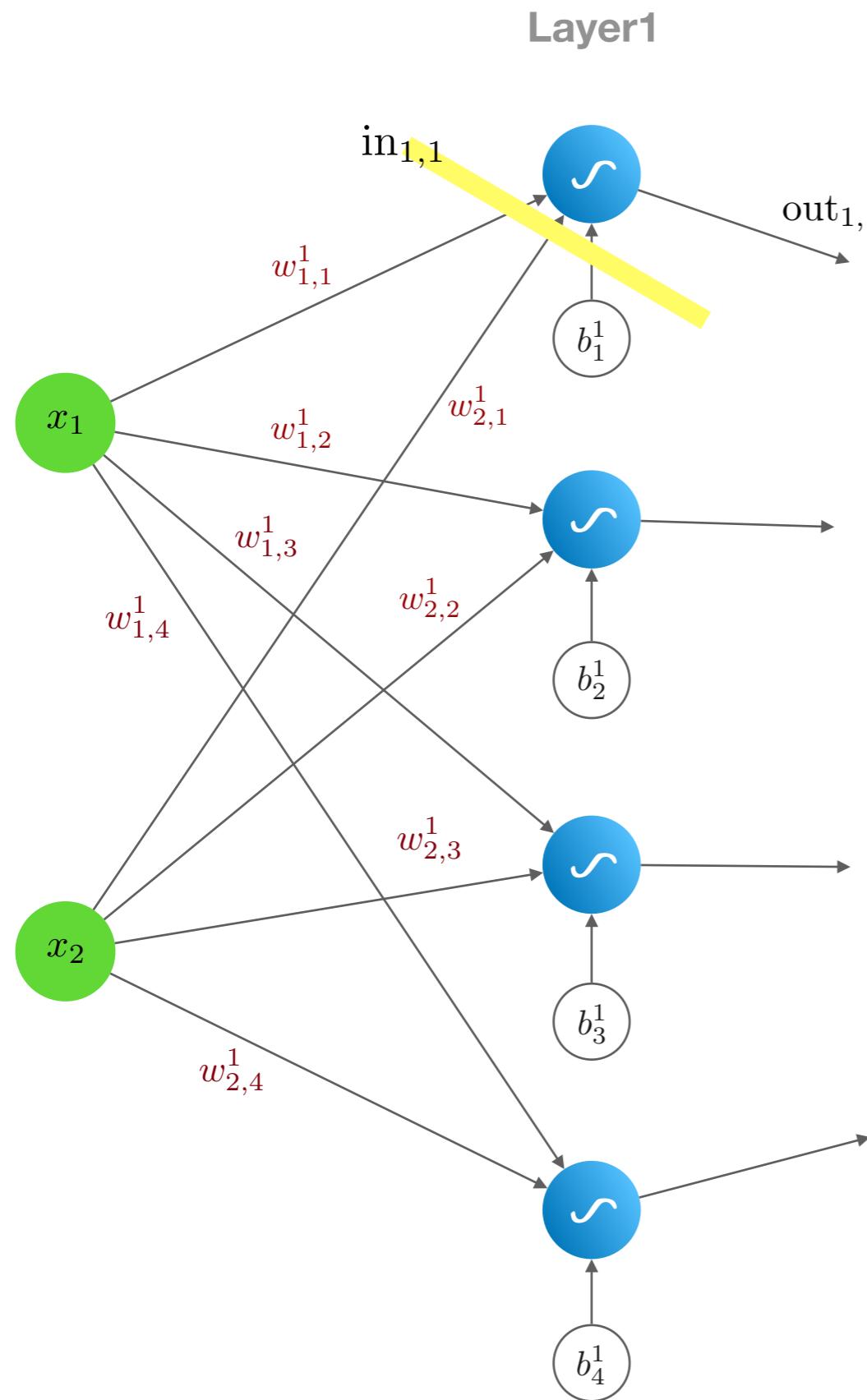


$$y_1 = w_{1,1}^3 \text{out}_{2,1} + w_{2,1}^3 \text{out}_{2,2} + w_{3,1}^3 \text{out}_{2,3} + b_1^3$$

$$y_2 = w_{1,2}^3 \text{out}_{2,1} + w_{2,2}^3 \text{out}_{2,2} + w_{3,2}^3 \text{out}_{2,3} + b_2^3$$

Output layer에 어떤 activation 함수를 사용할지는 문제의 성격에 따라 다르다.  
 (이 예에서는 identity function  $g(x) = x$ 를 사용한 것으로 하였다.)

# How does it work?



▶ Layer1의 각 노드에서 activation 함수를 적용하기 직전의 값을  $in_{1,1}$ ,  $in_{1,2}$ ,  $in_{1,3}$ ,  $in_{1,4}$ 라고 부른다면 각각은 다음과 같다.

$$in_{1,1} = w_{1,1}^1 x_1 + w_{2,1}^1 x_2 + b_1^1$$

$$in_{1,2} = w_{1,2}^1 x_1 + w_{2,2}^1 x_2 + b_2^1$$

$$in_{1,3} = w_{1,3}^1 x_1 + w_{2,3}^1 x_2 + b_3^1$$

$$in_{1,4} = w_{1,4}^1 x_1 + w_{2,4}^1 x_2 + b_4^1$$

▶ 행렬  $W_1$ 과  $b_1$ 을 다음과 같이 정의한다.

$$W_1 = \begin{bmatrix} w_{1,1}^1 & w_{1,2}^1 & w_{1,3}^1 & w_{1,4}^1 \\ w_{2,1}^1 & w_{2,2}^1 & w_{2,3}^1 & w_{2,4}^1 \end{bmatrix}$$

$$b_1 = [b_1^1, b_2^1, b_3^1, b_4^1]$$

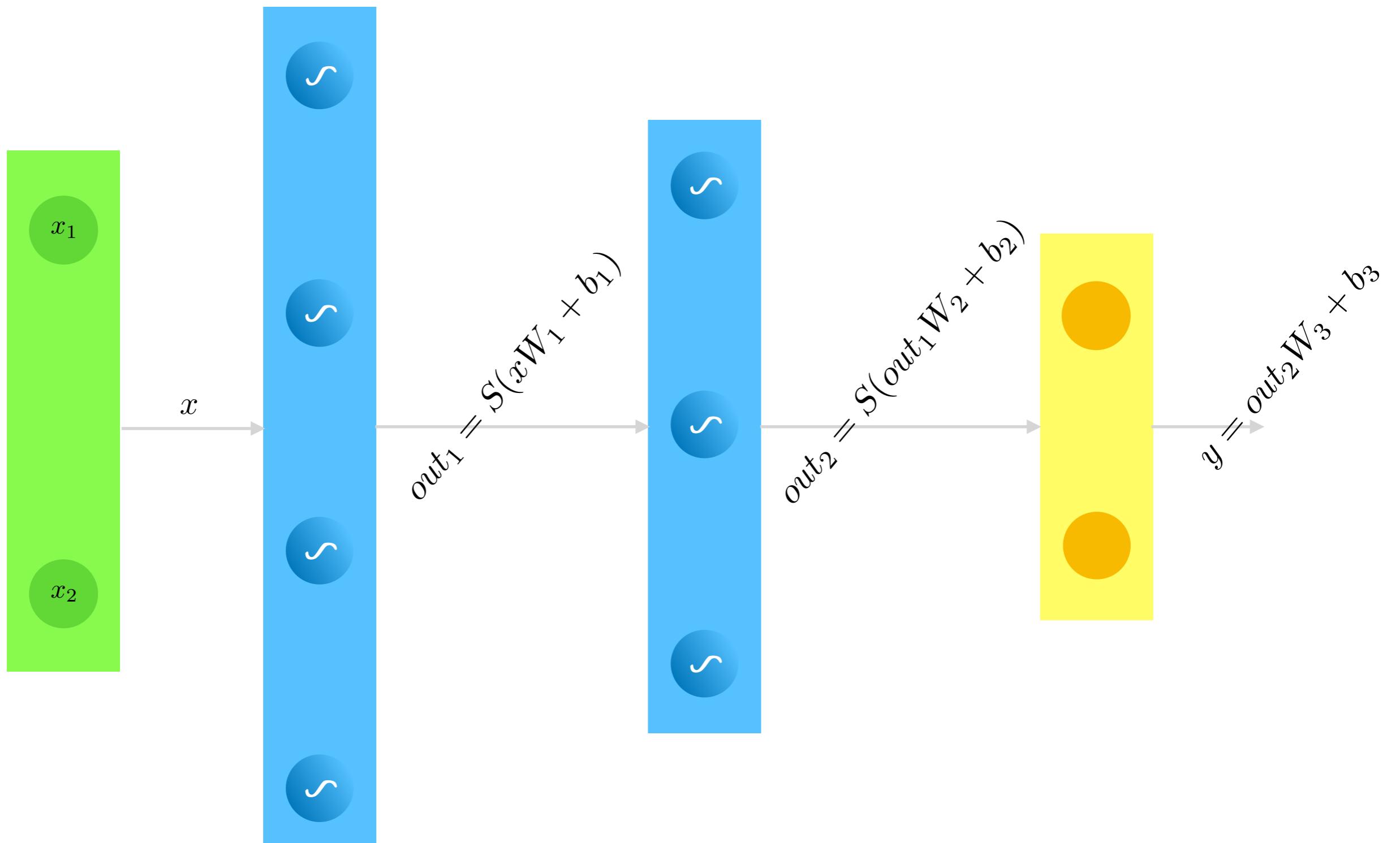
“이전 층의 노드 개수 × 다음 층의 노드 개수”

▶ 그러면 다음과 같은 행렬식으로 표현된다.

$$in_1 = xW_1 + b_1, \quad x = [x_1, x_2]$$

$$out_1 = S(in_1) \quad \leftarrow \text{S는 activation 함수}$$

# How does it work?



Fully connected neural network은 잘 정리된 행렬식으로 표현 가능하다.

$$y = \text{out}_2 W_3 + b_3$$

$$\text{out}_2 = S(\text{out}_1 W_2 + b_2)$$

$$\text{out}_1 = S(x W_1 + b_1)$$

$$S(x) = \frac{1}{1 + e^{-x}}$$

x가 벡터일 때

$$S([x_0, x_1]) = \left[ \frac{1}{1 + e^{-x_0}}, \frac{1}{1 + e^{-x_1}} \right]$$

$$y = \frac{1}{1 + e^{-(\frac{1}{1+e^{-(xW_1+b_1)}} W_2 + b_2)}} W_3 + b_3$$

“n개의 층을 가진 신경망은 n개의 행렬  $W_1, W_2, \dots, W_n$ 과 벡터  $b_1, b_2, \dots, b_n$ 으로 정의된 미분가능한 하나의 함수일 뿐이다.”

# **Training Neural Network**

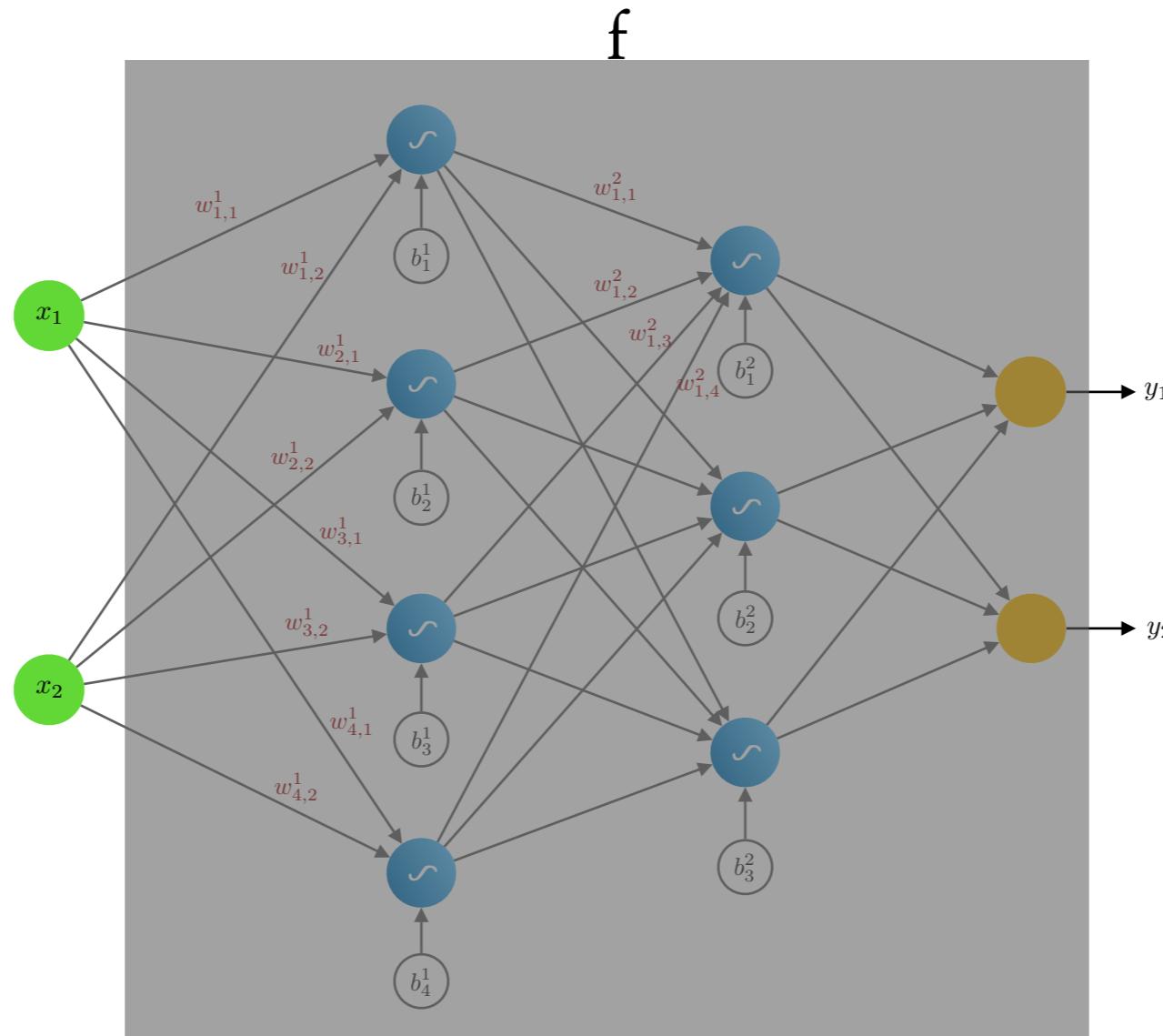
## 신경망의 학습

- ⦿ 신경망은 기본적으로는 **지도학습(supervised learning)**을 위한 도구이다. 즉 정답을 알고 있는 학습 데이터를 사용하여 신경망을 학습시킨다.
- ⦿ 학습(training) 데이터는 데이터(**feature**, 동물 이미지들)와 정답(**label**, 무슨 동물인지)으로 구성

Training Data Set		
	feature	label
X <sub>1</sub>	x <sub>1</sub> =2, x <sub>2</sub> =3	y <sub>1</sub> =1.0, y <sub>2</sub> =0.0
X <sub>2</sub>	x <sub>1</sub> =5, x <sub>2</sub> =8	y <sub>1</sub> =0.5, y <sub>2</sub> =0.0
X <sub>3</sub>	x <sub>1</sub> =1, x <sub>2</sub> =9	y <sub>1</sub> =0.0, y <sub>2</sub> =0.5
...		
X <sub>N</sub>	x <sub>1</sub> =3, x <sub>2</sub> =0	y <sub>1</sub> =0.0, y <sub>2</sub> =1.0

Test Data Set		
	feature	label
X <sub>1</sub>	x <sub>1</sub> =5, x <sub>2</sub> =1	?
X <sub>2</sub>	x <sub>1</sub> =2, x <sub>2</sub> =8	?
X <sub>3</sub>	x <sub>1</sub> =2, x <sub>2</sub> =3	?



feature	label (desired output)	(actual) output	error
$X_1 = (2, 3)$	$\bar{Y}_1 = (1.0, 0.0)$	$Y_1 = f(X_1, W, b)$	$\bar{Y}_1 - Y_1$
$X_2 = (5, 8)$	$\bar{Y}_2 = (0.5, 0.0)$	$Y_2 = f(X_2, W, b)$	$\bar{Y}_2 - Y_2$
$X_3 = (1, 9)$	$\bar{Y}_3 = (0.0, 0.5)$	$Y_3 = f(X_3, W, b)$	$\bar{Y}_3 - Y_3$
...		...	
$X_N = (3, 0)$	$\bar{Y}_N = (0.0, 1.0)$	$Y_N = f(X_N, W, b)$	$\bar{Y}_N - Y_N$

여기서  $W$ 와  $b$ 는 각각 모든 weight들과 bias들의 합집합을 나타낸다.

feature	label (desired output)	(actual) output	error
$X_1 = (2, 3)$	$\bar{Y}_1 = (1.0, 0.0)$	$Y_1 = f(X_1, W, b)$	$\bar{Y}_1 - Y_1$
$X_2 = (5, 8)$	$\bar{Y}_2 = (0.5, 0.0)$	$Y_2 = f(X_2, W, b)$	$\bar{Y}_2 - Y_2$
$X_3 = (1, 9)$	$\bar{Y}_3 = (0.0, 0.5)$	$Y_3 = f(X_3, W, b)$	$\bar{Y}_3 - Y_3$
...		...	
$X_N = (3, 0)$	$\bar{Y}_N = (0.0, 1.0)$	$Y_N = f(X_N, W, b)$	$\bar{Y}_N - Y_N$

## >Total error (혹은 loss)

$$E_{X, \bar{Y}}(W, b) = \frac{1}{N} \sum_{i=1}^N ||\bar{Y}_i - Y_i||^2$$

모든 학습 데이터에 대한 에러의 mean squared sum,  
 여기서 중요한 점은 이것이 “W와 b의 함수”라는 점  
 (왜냐하면 X들과  $\bar{Y}$ 들은 입력으로 주어진 상수값들이기 때문)

“Gradient descent 알고리즘을 이용하여  $E_{X, \bar{Y}}(W, b)$ 를 최소로하는  
 W와 b를 찾아라 !!!”

## >Error(loss) 함수와 그것의 gradient:

$$E_{X, \bar{Y}}(W, b) = \frac{1}{N} \sum_{i=1}^N \|\bar{Y}_i - Y_i\|^2$$

$$\nabla E_{X, \bar{Y}}(W, b) = ?$$

Gradient descent 알고리즘을 적용하기 위해서는  
error함수의 gradient를 구해야 함

하지만 이걸 어떻게 구할지 걱정할 필요는 없음.  
똑똑한 사람들이 이미 구해 놓았음.  
그리고 별로 어렵지도 않음

W와 b의 값을 random하게 지정한다.  
각각을  $W_0$ 와  $b_0$ 라고 하자.

$Y_i, i = 1, \dots, N$ ,를 계산한다.

$\nabla E_{X, \bar{Y}}(W_0, b_0)$ 를 계산한다.

W와 b를 다음과 같이 갱신한다.

$$W_1, b_1 = W_0, b_0 - \rho \nabla E_{X, \bar{Y}}(W_0, b_0)$$

$Y_i, i = 1, \dots, N$ ,를 다시 계산한다.

$\nabla E_{X, \bar{Y}}(W_1, b_1)$ 를 계산한다.

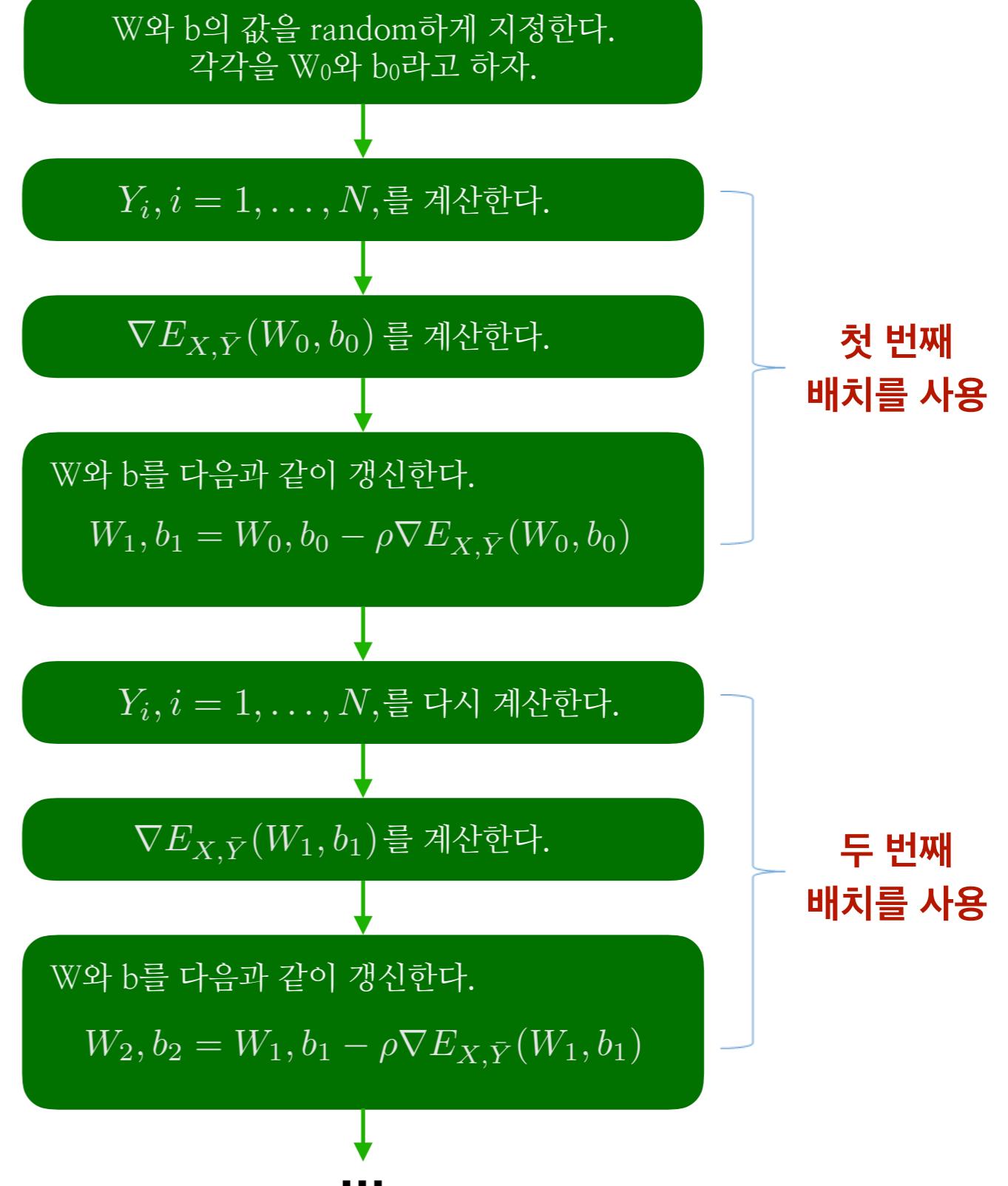
W와 b를 다음과 같이 갱신한다.

$$W_2, b_2 = W_1, b_1 - \rho \nabla E_{X, \bar{Y}}(W_1, b_1)$$

...

# SGD: Stochastic Gradient Descent

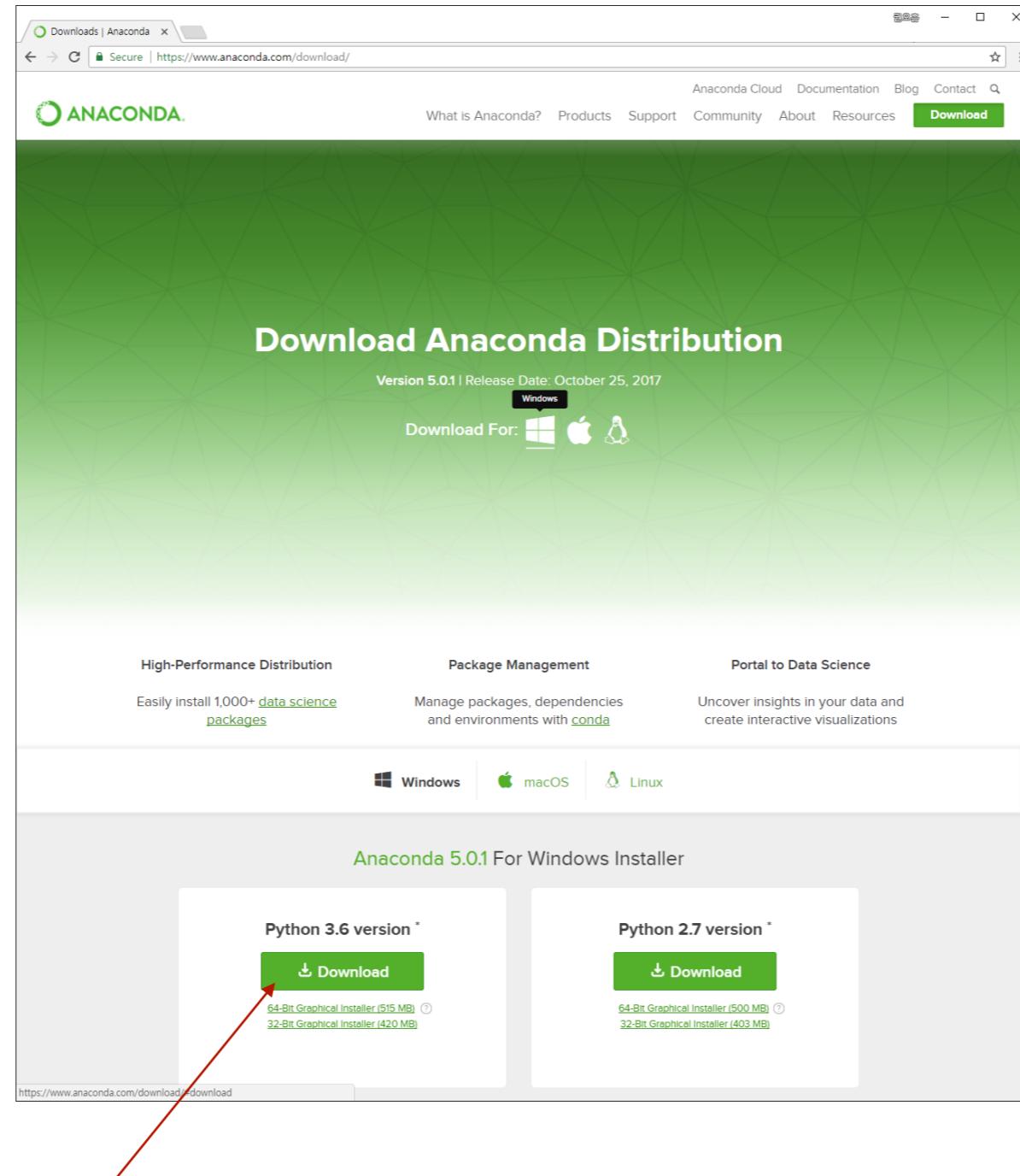
- Loss는 모든 학습 데이터에 대한 오류의 평균이다.
$$E_{X,\bar{Y}}(W, b) = \frac{1}{N} \sum_{i=1}^N \|\bar{Y}_i - Y_i\|^2$$
- 학습 데이터가 많을 경우 계산량이 많다.
- 학습 데이터를 일정한 크기로 랜덤하게 분할하여 각각을 **mini-batch**라고 부른다.
- 각 라운드마다 다른 mini-batch를 사용한다.
- 모든 학습 데이터가 한 번씩 사용되면 한 시대(epoch)가 지났다고 말하고, 보통 학습은 여러 시대를 거쳐서 이루어진다.



# 실습 환경 구축

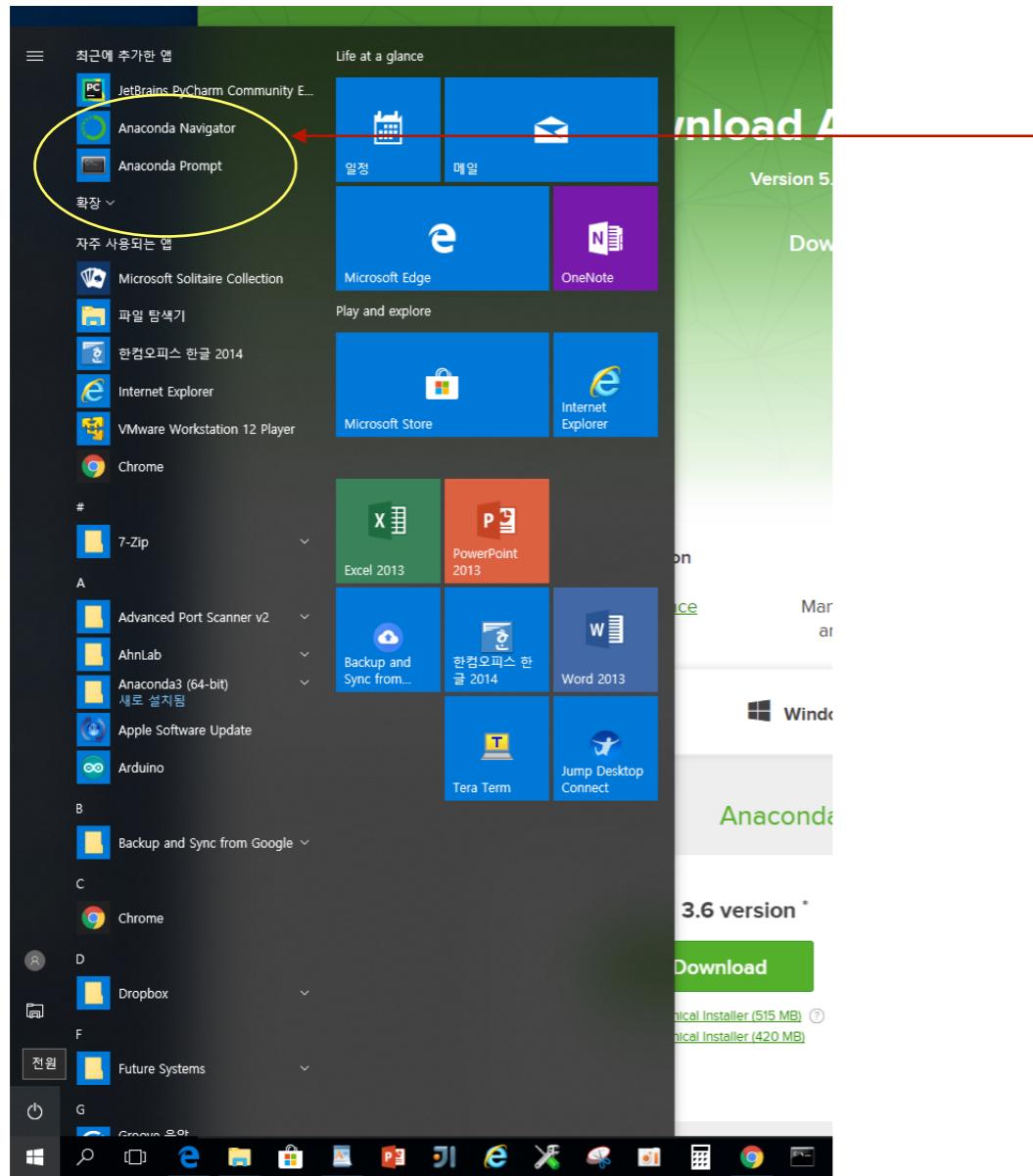
(Anaconda, Python3.5, OpenCV, Tensorflow, PyCharm 설치하기)

<https://www.anaconda.com/download/>

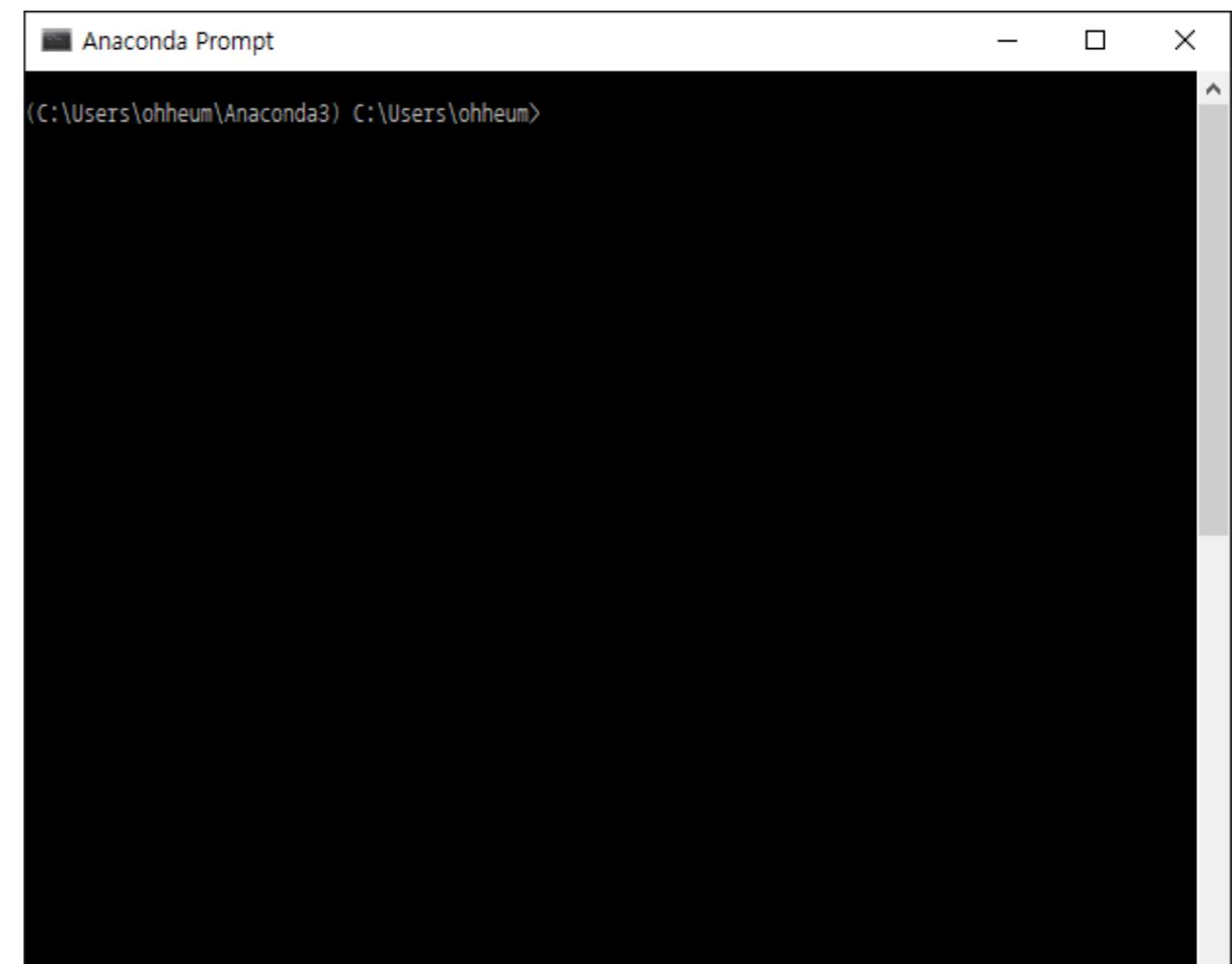


Python3.6 버전을 다운로드하여 설치한다. 설치과정에서는 전부 default 설정을 따른다.

# Tensorflow 설치



Anaconda Prompt를 실행하면  
아래 그림과 같은 콘솔이 실행된다.  
(Mac이나 Linux에서는 그냥 Terminal을 실행한다.)



먼저 python3.5를 사용하는 virtual environment를 이렇게 생성한다. 이름(tfenv3.5)은 각자 다르게 정해도 된다.



```
(C:\Users\ohheum\Anaconda3) C:\Users\ohheum> conda create -n tfenv3.5 python=3.5  
Fetching package metadata .....  
Solving package specifications: .
```

```
Package plan for installation in environment C:\Users\ohheum\Anaconda3\envs\tfenv3.5:  
The following NEW packages will be INSTALLED:
```

certifi:	2017.11.5-py35h456c6ae_0
pip:	9.0.1-py35h691316f_4
python:	3.5.4-h1357f44_23
setuptools:	36.5.0-py35h21a22e4_0
vc:	14-h2379b0c_2
vs2015_runtime:	14.0.25123-hd4c4e62_2
wheel:	0.30.0-py35h38a90bc_1
wincertstore:	0.2-py35hfebbdb8_0

Proceed ([y]/n)? **y** ← y를 입력한다.

```
#  
# To activate this environment, use:  
# > activate tfenv3.5  
#  
# To deactivate an active environment, use:  
# > deactivate  
#  
# * for power-users using bash, you must source  
#
```

▶ 이 수업은 Python3.5 환경에서 실습한다. OpenCV가 Python3.6 환경에서는 OS에 따라서 지원이 안되는 경우가 있기 때문이다.

프롬프트가 이런 식으로 바뀐다.

생성한 virtual environment (tfenv3.5)를 활성화한다.

(Mac이나 Linux에서는 \$ source activate tfenv3.5라고 실행한다.)

```
(C:\Users\ohheum\Anaconda3) C:\Users\ohheum> activate tfenv3.5
```

```
(tfenv3.5) C:\Users\ohheum> pip install --ignore-installed --upgrade tensorflow
Collecting tensorflow
  Downloading tensorflow-1.4.0-cp35-cp35m-win_amd64.whl (28.3MB)
    100% |#####| 28.3MB 289kB/s
```

... (생략)

Tensorflow를 설치한다.

(Mac이나 Linux는 tensorflow 홈페이지 참조)

```
Successfully built html5lib markdown
```

```
Installing collected packages: six, setuptools, protobuf, enum34, wheel, numpy, html5lib, bleach, werkzeug, markdown, tensorflow-tensorboard, tensorflow
```

```
Successfully installed bleach-1.5.0 enum34-1.1.6 html5lib-0.9999999 markdown-2.6.10
```

```
numpy-1.13.3 protobuf-3.5.0.post1 setuptools-38.2.4 six-1.11.0 tensorflow-1.4.0 tensorflow-tensorboard-0.4.0rc3 werkzeug-0.13 wheel-0.30.0
```

```
(tfenv3.5) C:\Users\ohheum> conda install -c anaconda opencv
```

opencv3 패키지를 설치한다.

```
(tfenv3.5) C:\Users\ohheum> python
Python 3.5.4 |Anaconda, Inc.| (default, Nov  8 2017, 14:34:30) [MSC v.1900 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.1.0'
>>> quit()
```

opencv가 제대로 import되는지 간단하게 테스트해본다.

```
(tfenv3.5) C:\Users\ohheum> deactivate ← virtual environment를 비활성화한다.
```

## 설치과정에서 문제가 생겼을 때

설치된 virtual environment의 목록을 본다.



```
(C:\Users\ohheum\Anaconda3) C:\Users\ohheum> conda info --envs
```

```
# conda environments:
```

```
#
```

```
tfenv3.5          C:\Users\ohheum\Anaconda3\envs\tfenv3.5
```

```
root             * C:\Users\ohheum\Anaconda3
```

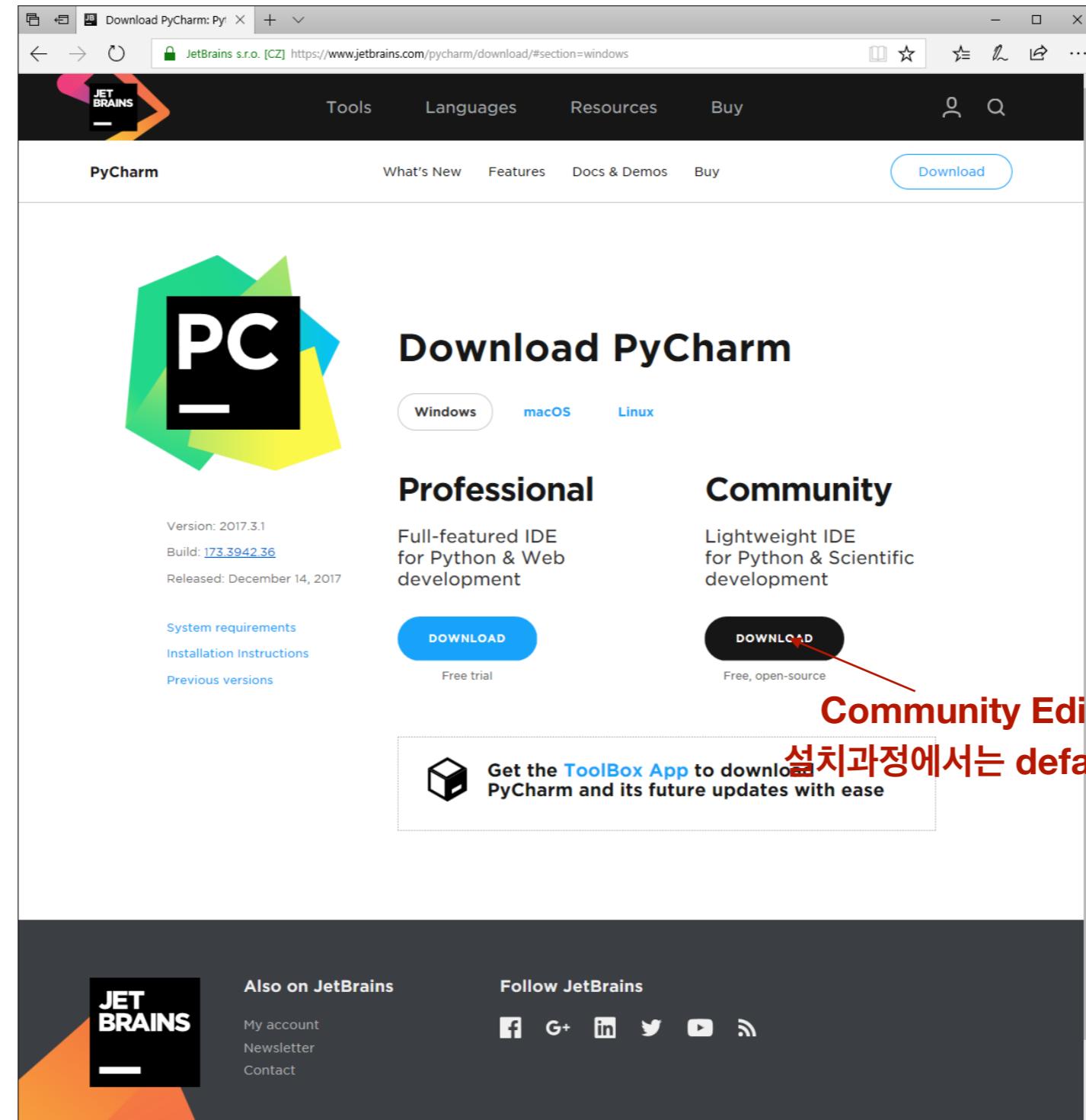
```
(C:\Users\ohheum\Anaconda3) C:\Users\ohheum> conda remove --name tfenv3.5 --all
```



뭔가 잘못 되었을 때는 Anaconda Prompt를  
새로 실행한 후 이렇게 생성한 virtual env를 삭제하고  
새로 해본다.

# Pycharm Community Edition 설치

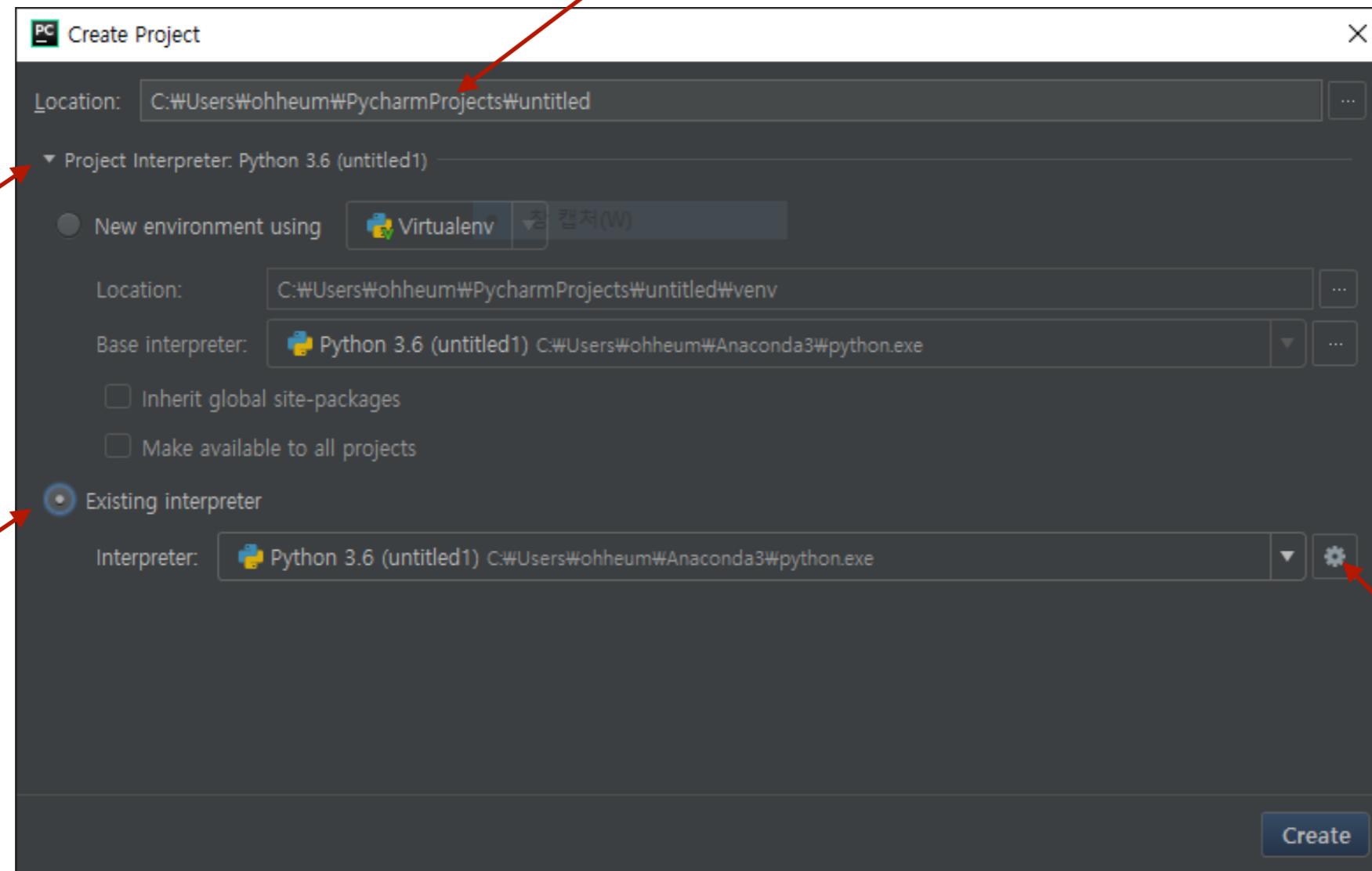
<https://www.jetbrains.com/pycharm/download/#section=windows>



Community Edition을 다운로드하여 설치한다.  
설치과정에서는 default 설정을 그대로 따라가면 된다.

## 1. PyCharm을 실행한 후 새로운 프로젝트를 생성한다 (File ->New Project).

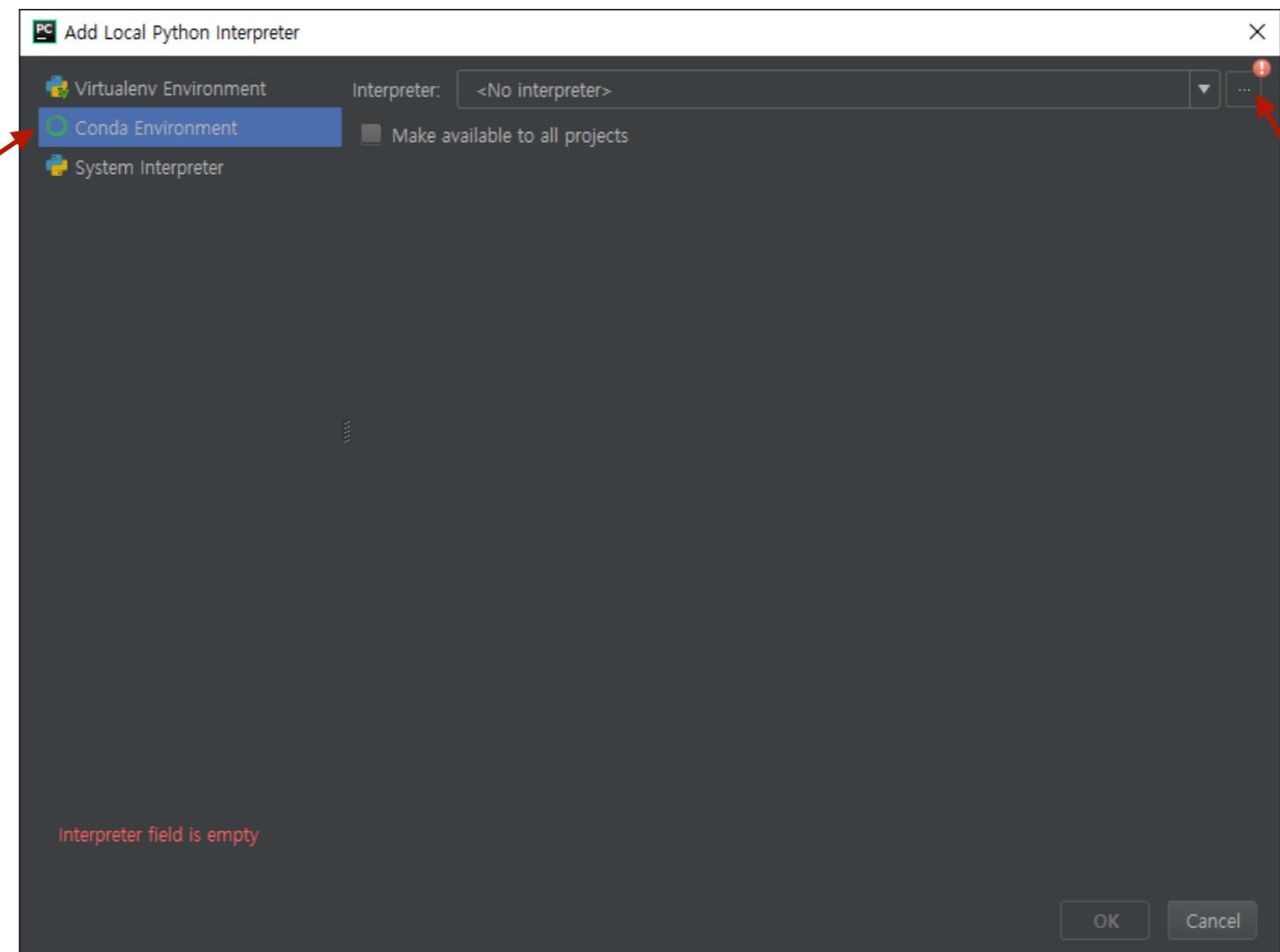
프로젝트 저장 위치와 이름을 따로 지정하지 않았더니 이 위치에 untitled라는 이름으로 생성되었다.  
물론 원하는 위치에 원하는 이름으로 지정할 수 있다.

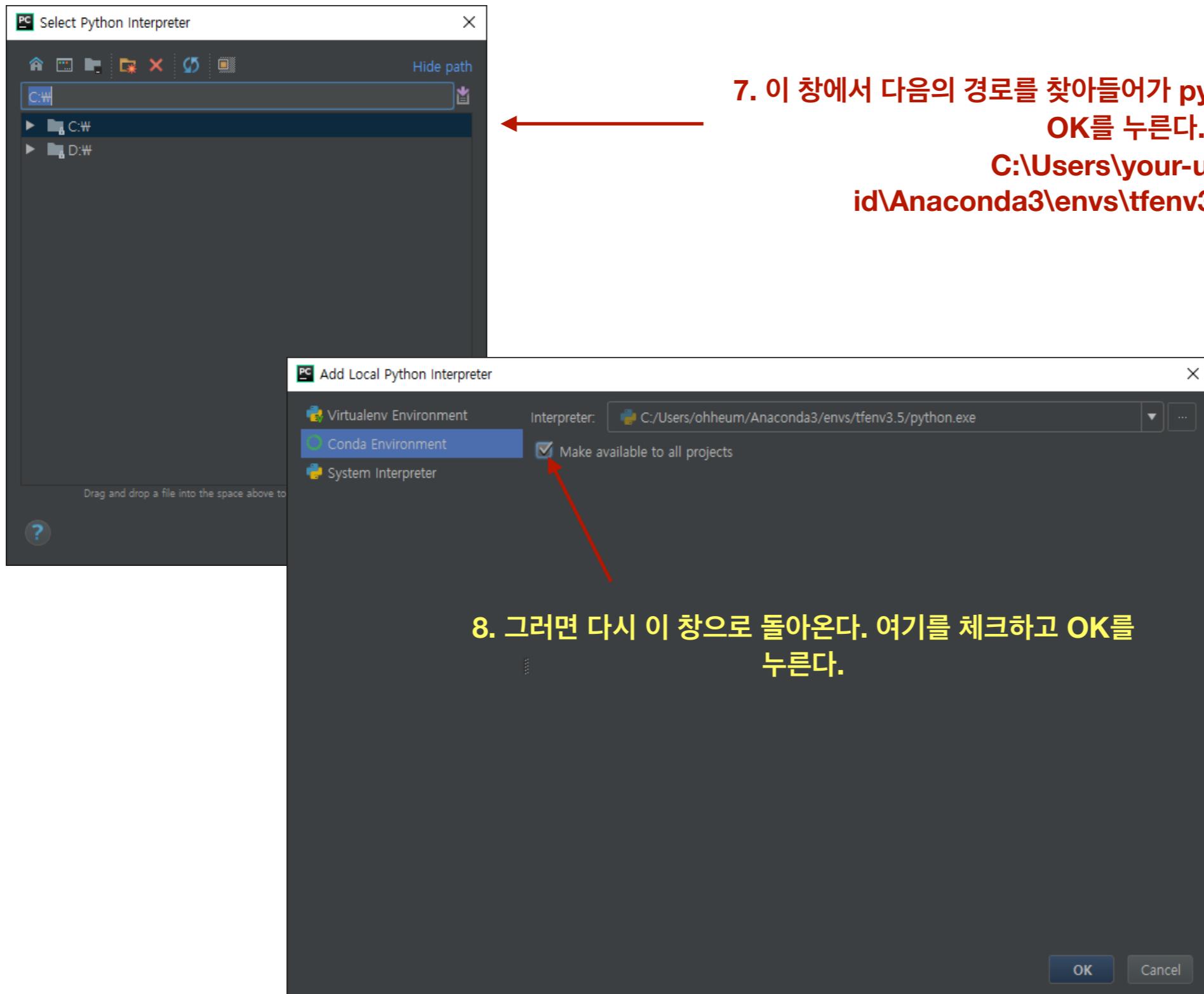


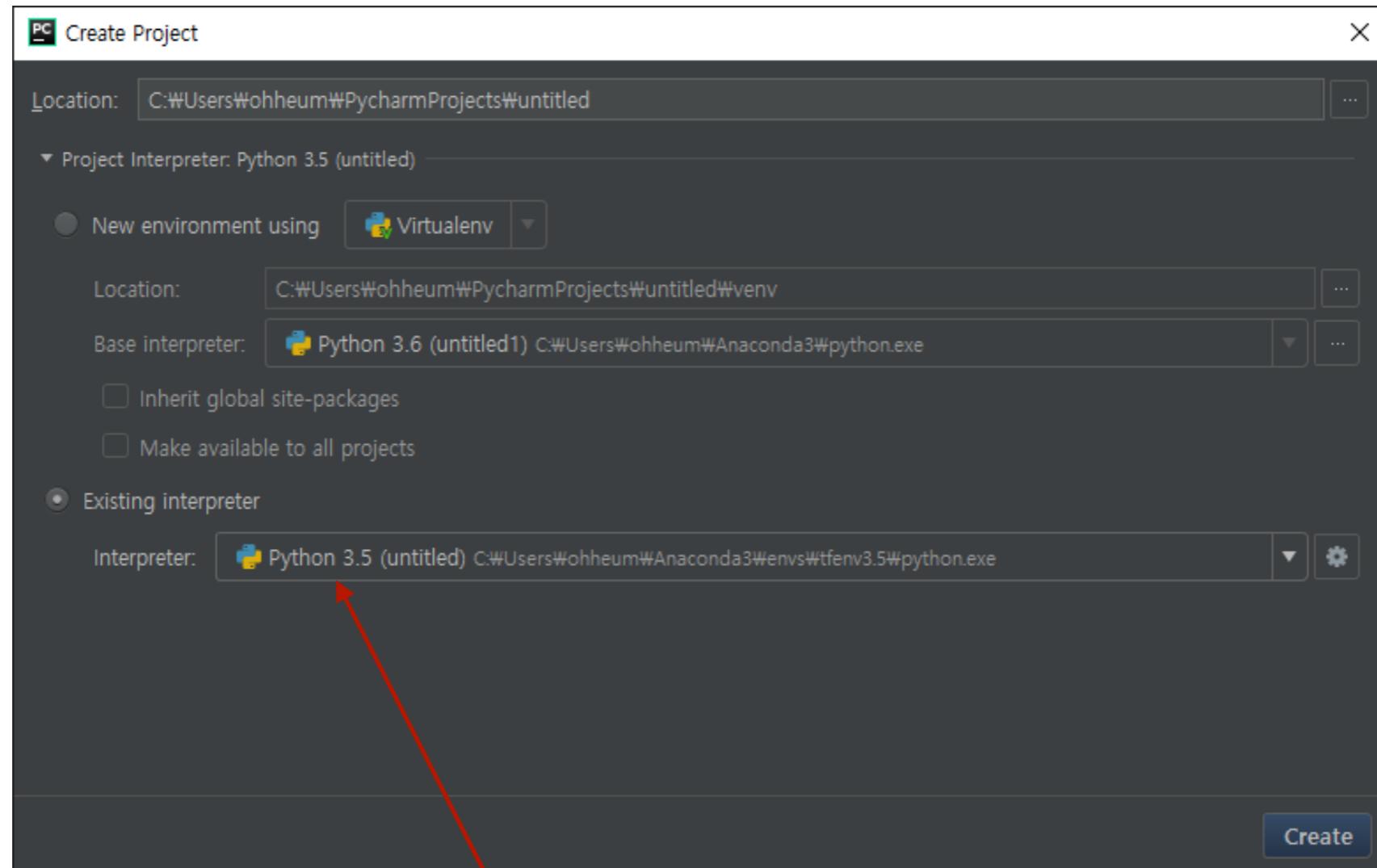
2. 여기의 세모를 클릭하면 그럼 처럼 메뉴가 펼쳐진다.

3. Existing Interpreter를 선택 한다.

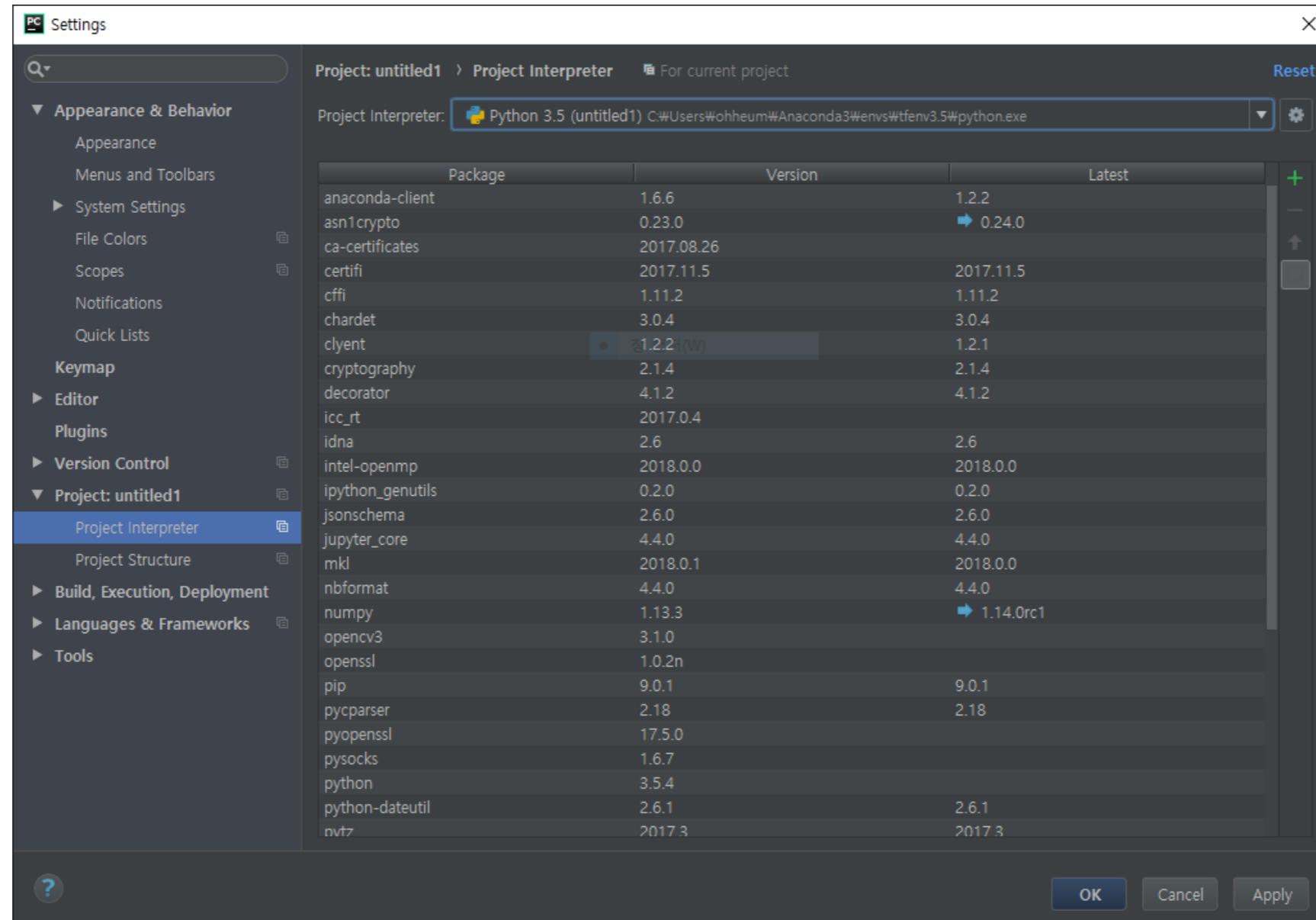
4. 이걸 누르고 Add Local을 선택한다.  
그러면 다음 페이지의 대화상자가 나타난다.







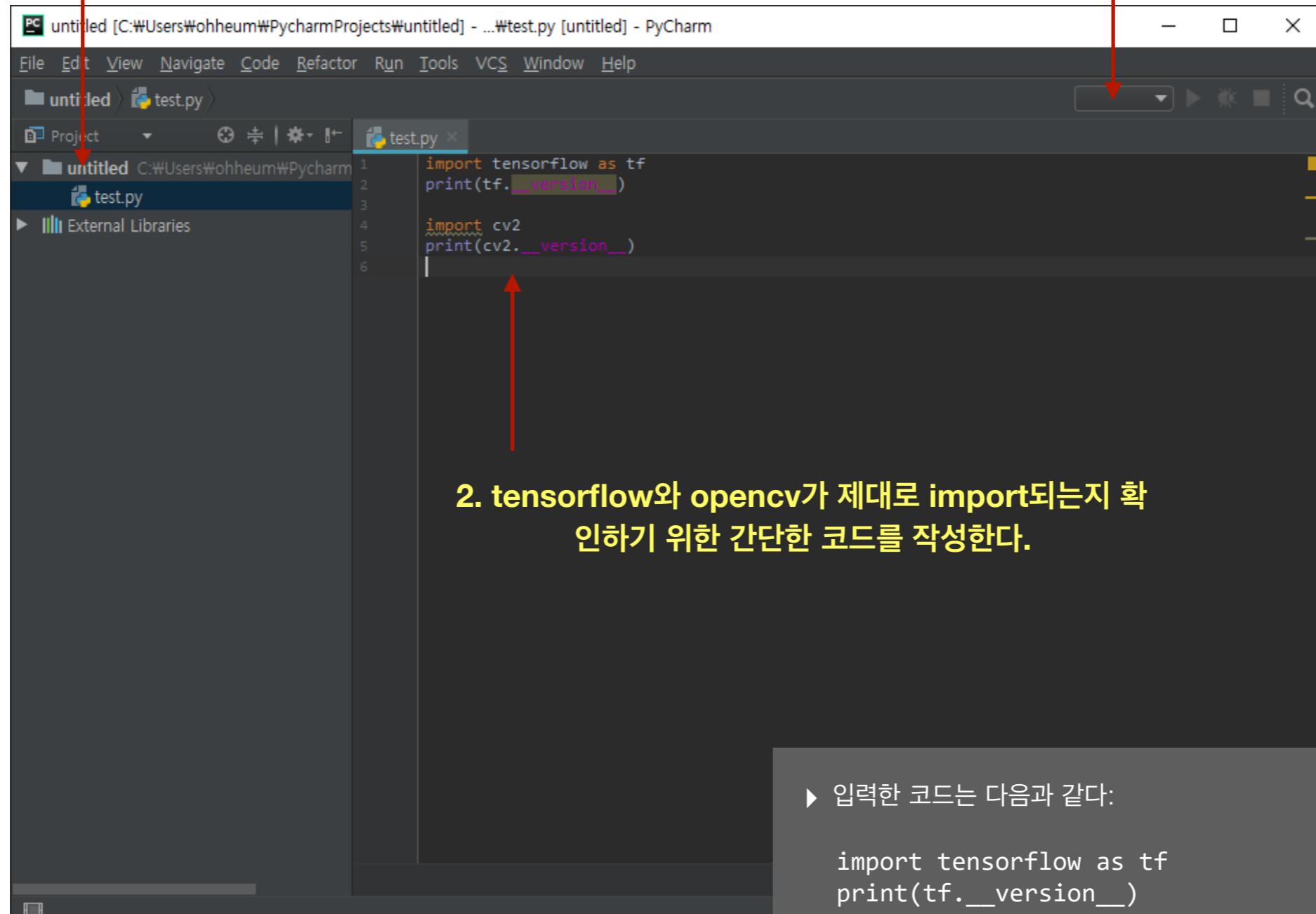
9. Python3.5가 인터프리터로 설정되었다.  
Create 버튼을 클릭하여 설정을 마친다.



프로젝트를 먼저 생성한 후 File ->Setting에서도 유사한 방법으로 Interpreter를 지정 혹은 변경 할 수 있다.

# 첫 번째 프로젝트

1. 프로젝트 이름을 마우스 오른쪽 버튼으로 클릭하고  
New->Python File을 선택한 후 test.py라는 이름의  
소스 파일을 생성하였다.

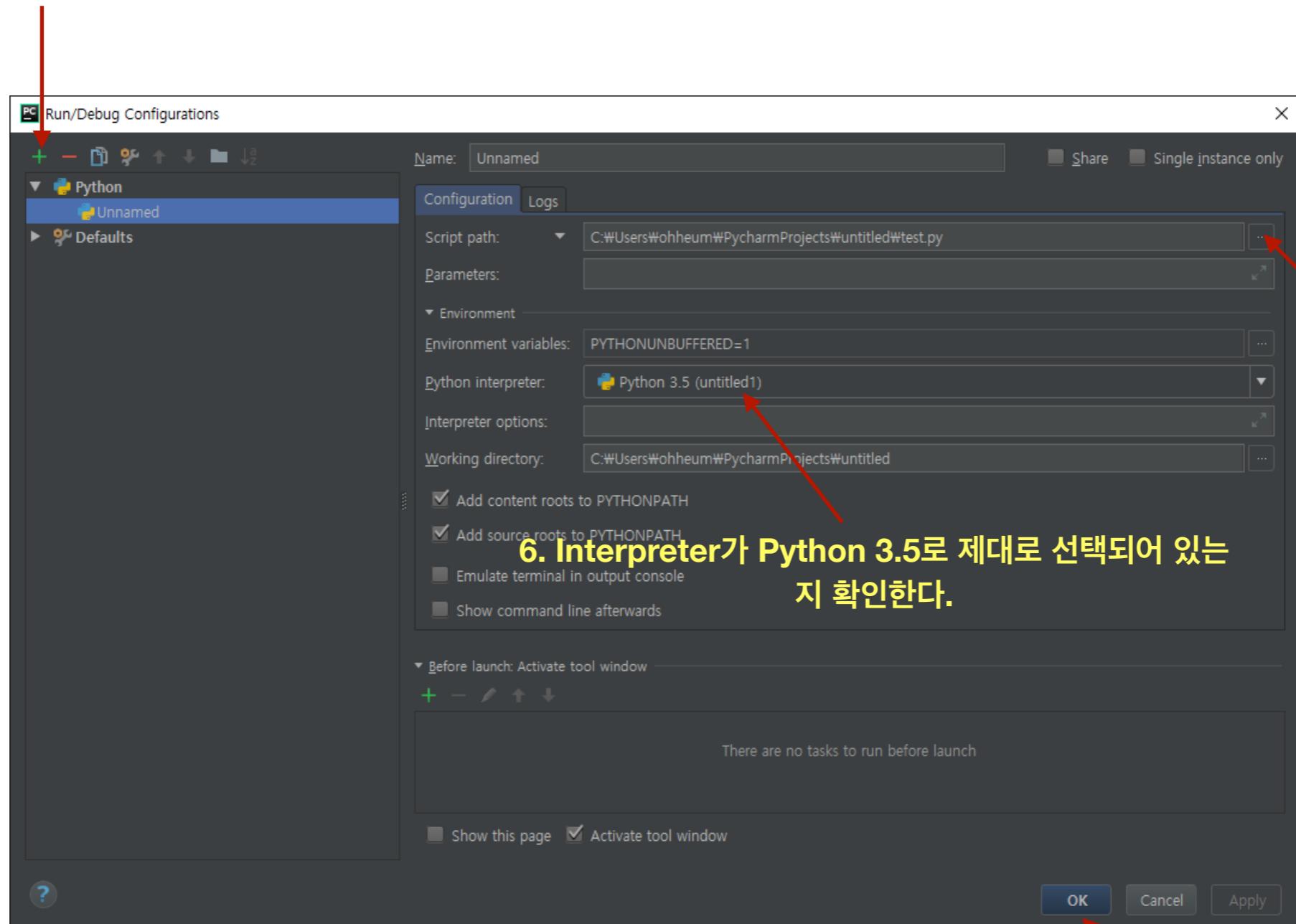


3. 여기를 누르고 Edit Configurations를 선택한다.

▶ 입력한 코드는 다음과 같다:

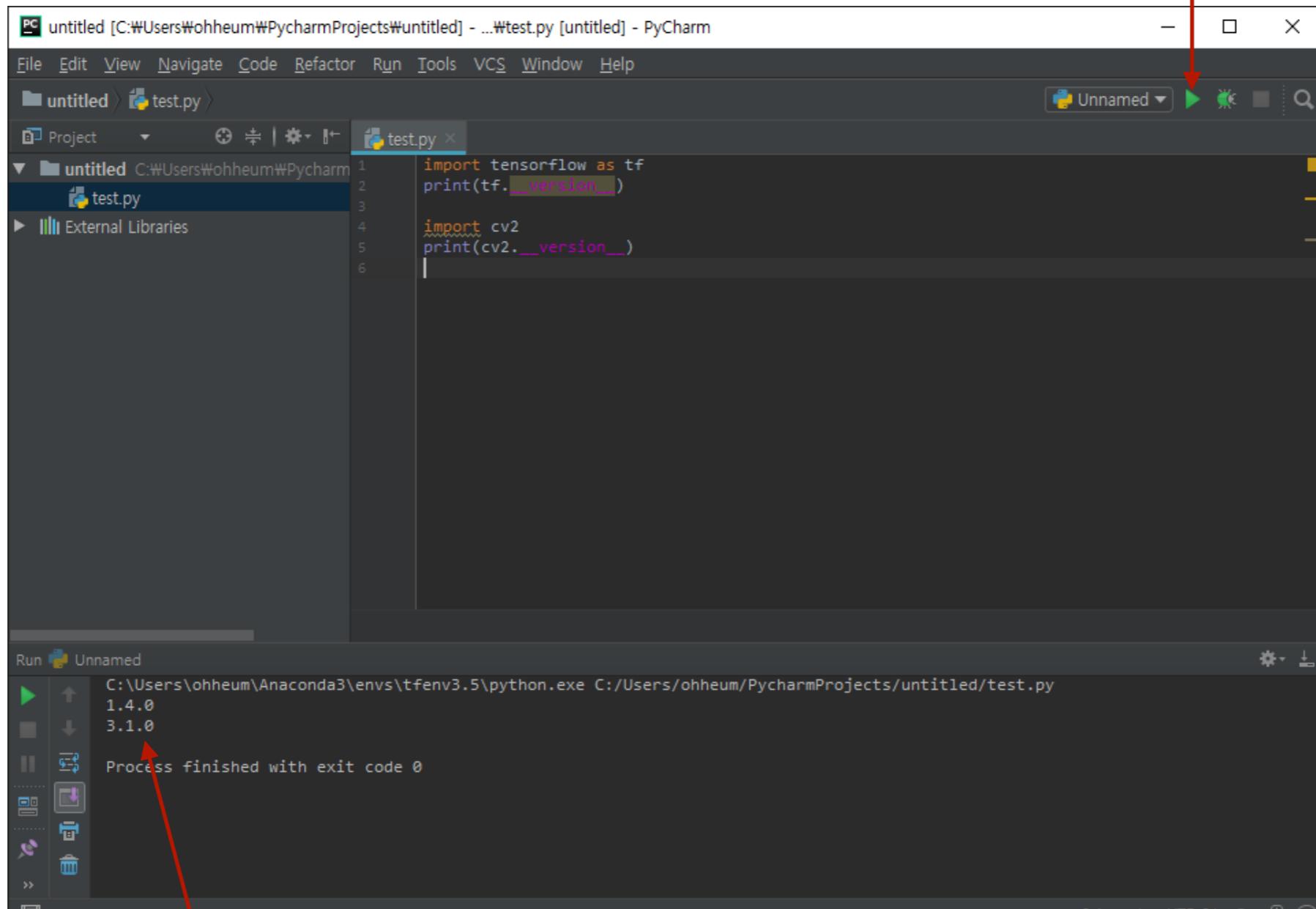
```
import tensorflow as tf
print(tf.__version__)
import cv2
print(cv2.__version__)
```

4. '+'를 누르고 Python을 선택한다.



7. OK를 누른다.

8. 이 버튼을 눌러 프로그램을 실행한다.



9. 이런 식으로 출력되면 모두 정상이다.

## 과제: Python List와 Dictionary 익히기

- ☞ <https://www.youtube.com/watch?v=W8KRzm-HUcc>
- ☞ <https://www.youtube.com/watch?v=ajrtAuDg3yw>
- ☞ <https://www.youtube.com/watch?v=daefaLgNkw0&t=2s>