02. Configure and Basic

오늘 실습 내용

- Colab 소개
- Pytorch 소개
- Tensor

Pytorch

- Python으로 실습 진행
  - 개인 노트북에서 GPU를 사용할 수 있으면 로컬로 학습해도 됨
  - 실습은 colab으로 진행함

Colab 소개

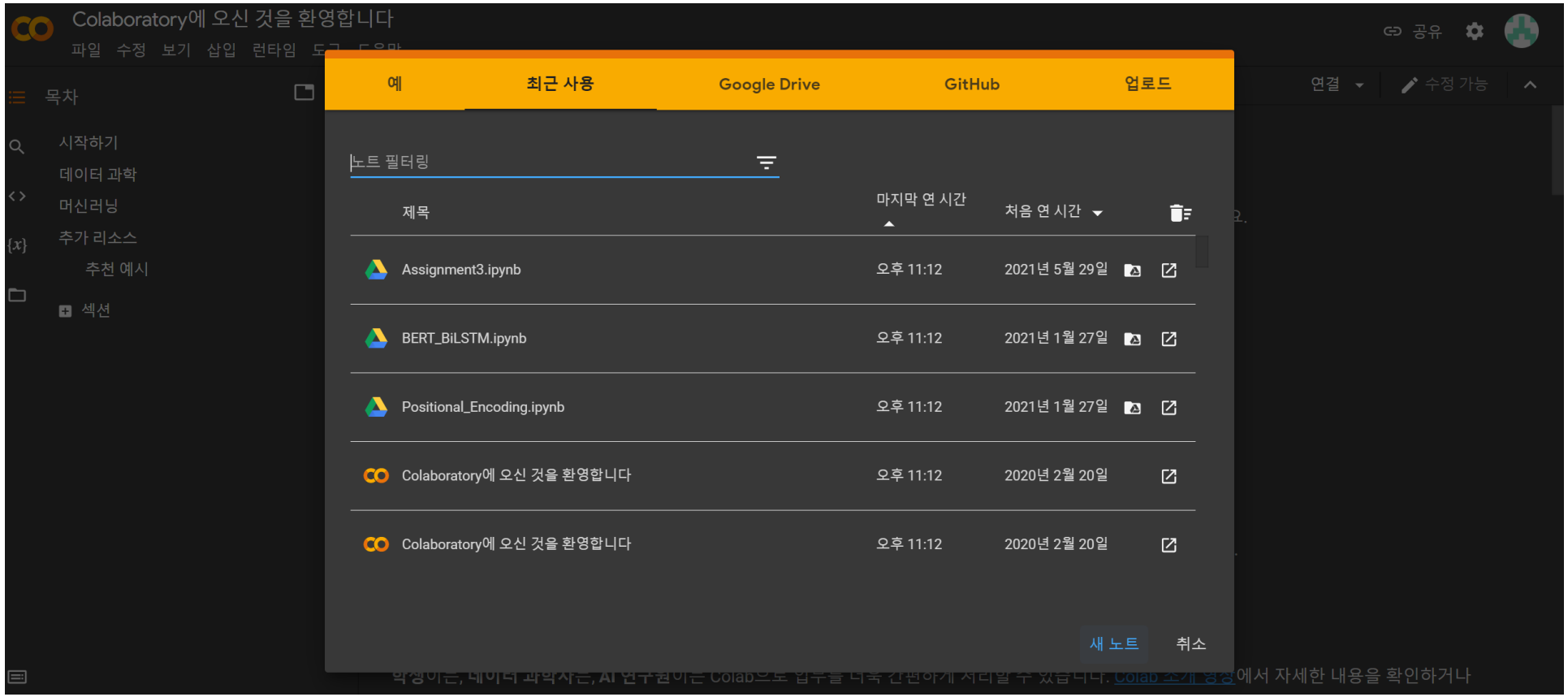
Colab 사용

Colab

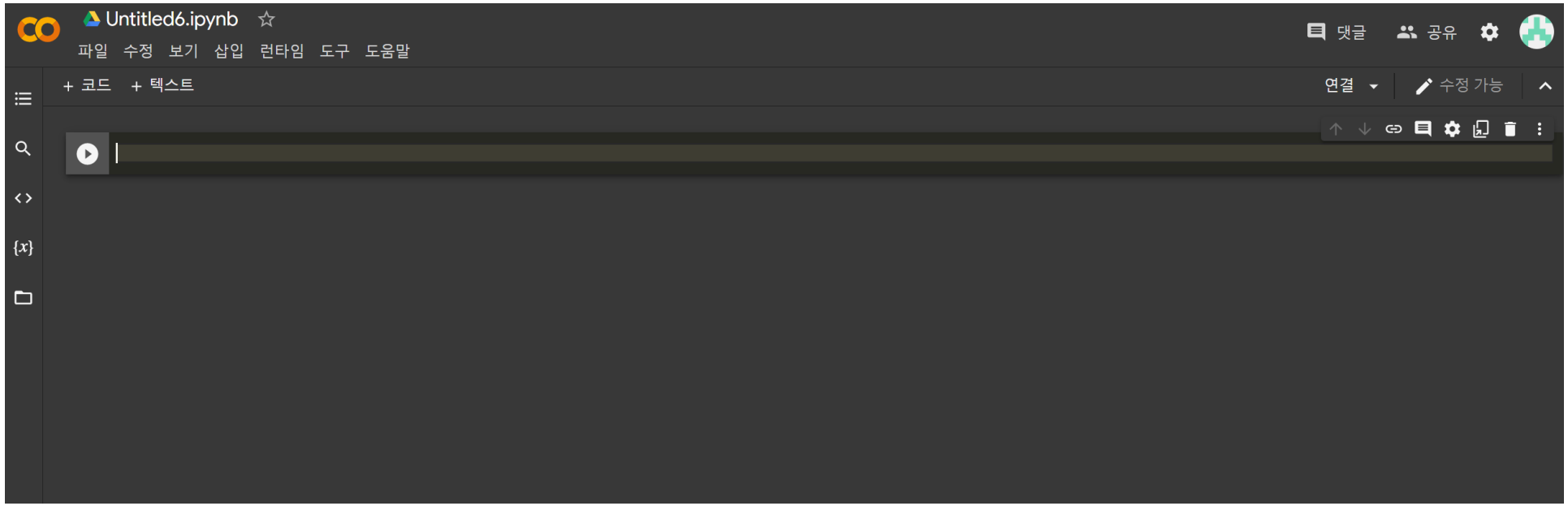
- Google의 Colaboratory약자

- Python 스크립트를 작성&실행
- GPU 무료 액세스
  - 최대 12시간 연결가능

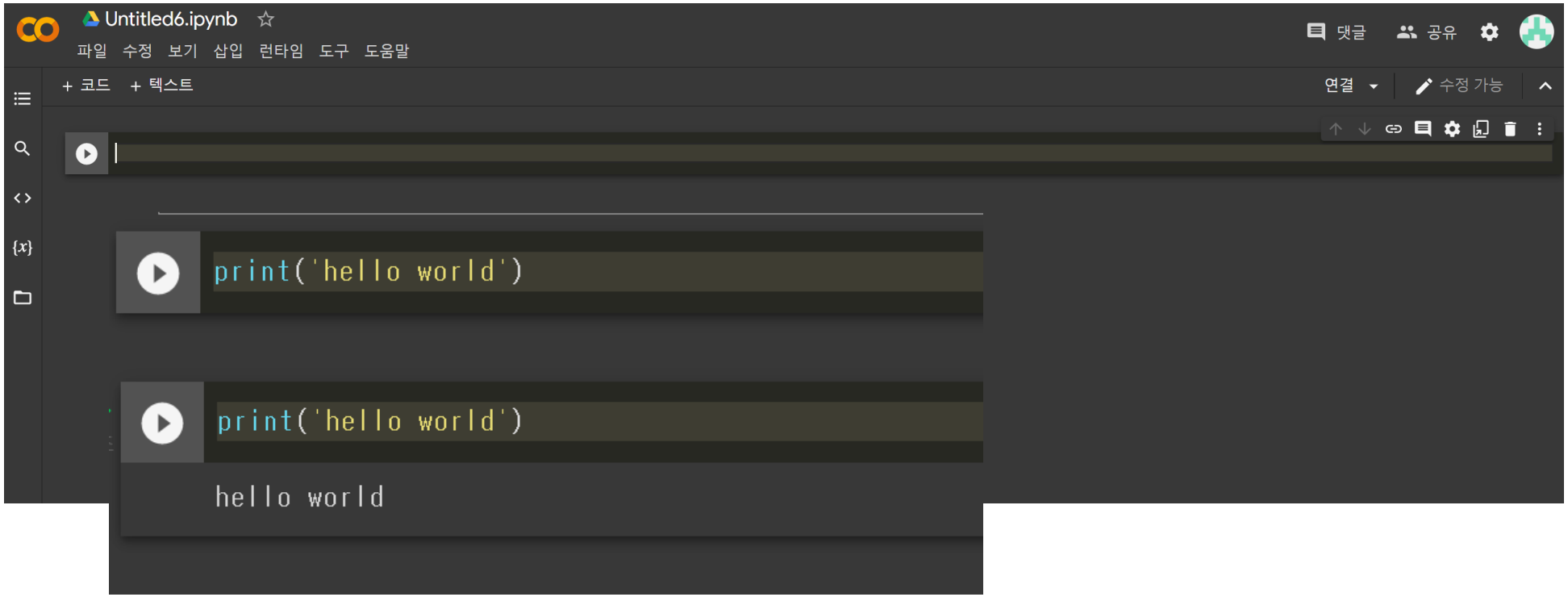
Colab 사용


Colaboratory에 오신 것을 환영합니다
파일 수정 보기 삽입 런타임
공유
연결
수정 가능
목차
시작하기
데이터 과학
머신러닝
추가 리소스
추천 예시
섹션
예
최근 사용
Google Drive
GitHub
업로드
노트 필터링
제목
마지막 연 시간
처음 연 시간
Assignment3.ipynb 오후 11:12 2021년 5월 29일
BERT_BiLSTM.ipynb 오후 11:12 2021년 1월 27일
Positional_Encoding.ipynb 오후 11:12 2021년 1월 27일
Colaboratory에 오신 것을 환영합니다 오후 11:12 2020년 2월 20일
Colaboratory에 오신 것을 환영합니다 오후 11:12 2020년 2월 20일
새 노트
취소

Colab 사용



Colab 사용

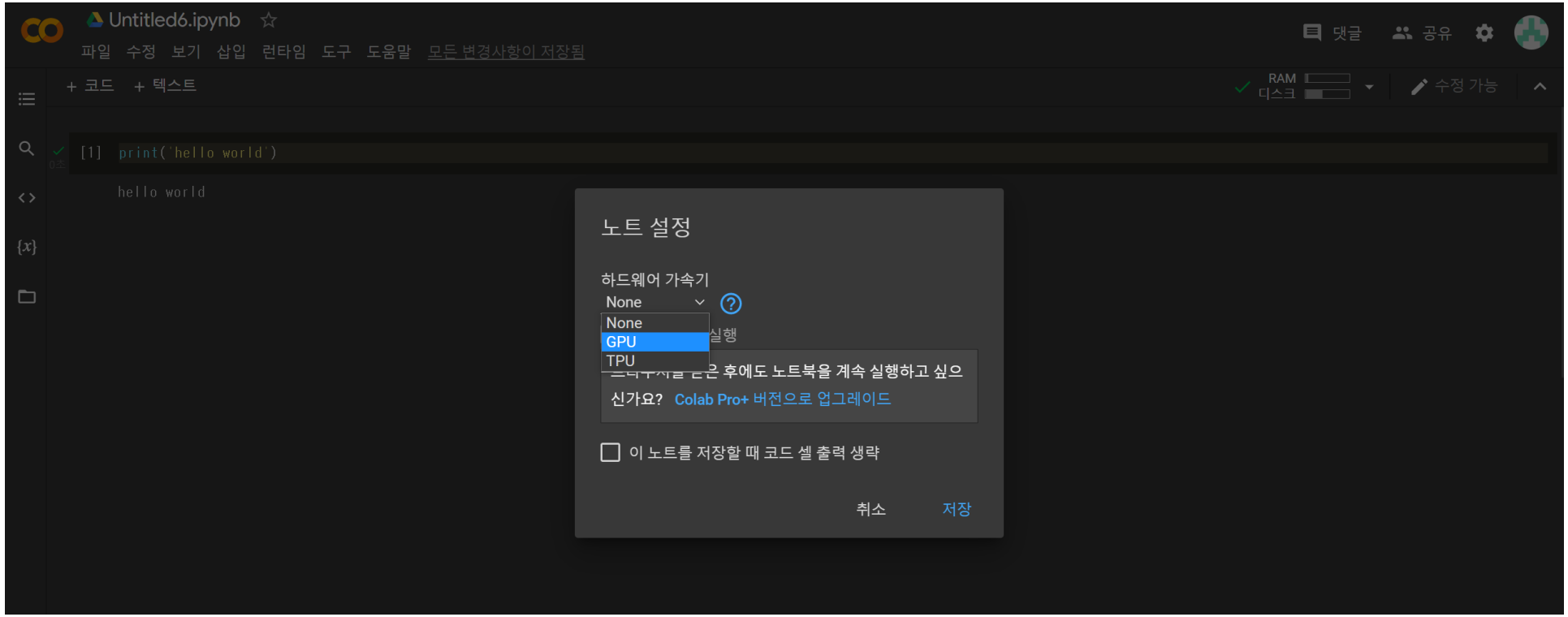

Untitled6.ipynb
파일 수정 보기 삽입 런타임 도구 도움말
댓글 공유
+ 코드 + 텍스트
연결 수정 가능
print('hello world')
print('hello world')
hello world

Colab 사용

- GPU 사용

Colab 사용

- GPU 사용
  - 지금 당장은 gpu 필요 없음



Pytorch 소개

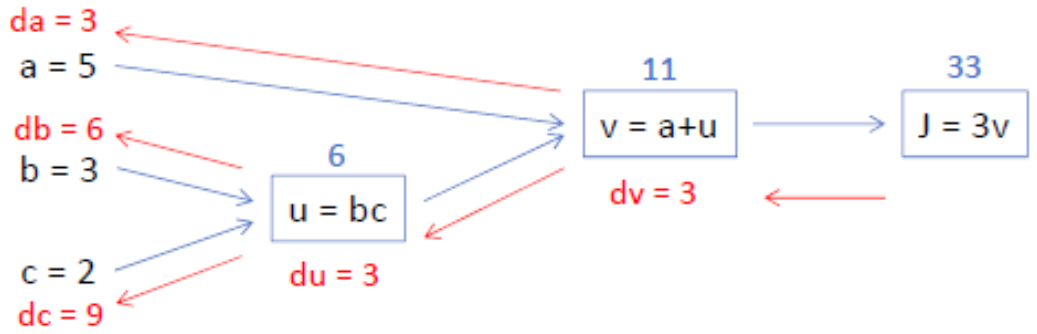
Pytorch

- 딥러닝 프레임워크
  - GPU를 활용하여 딥러닝 모델을 만들고 학습
  - ex) tensorflow, pytorch

- Meta AI

Pytorch

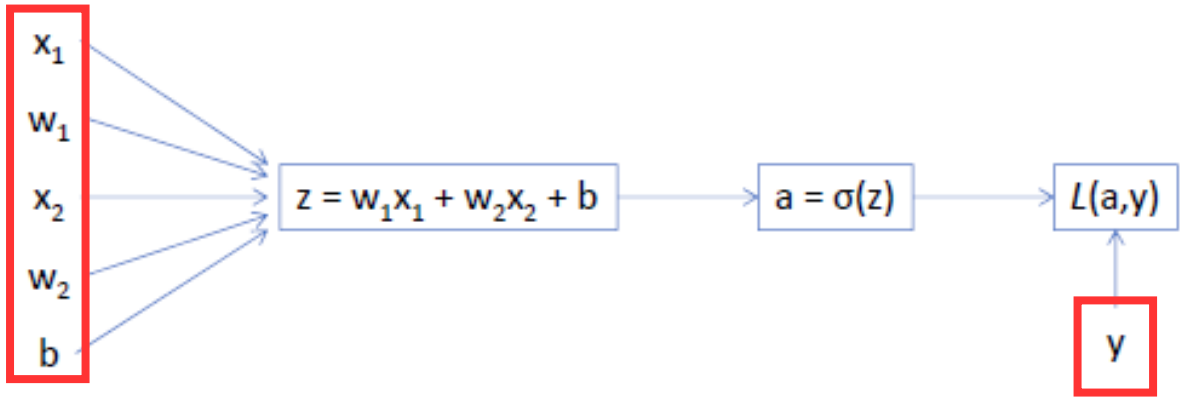
- Forward Propagation과 Backward Propagation을 자동으로 해줌
  - Tensor, Module class 필요



Tensor

Tensor

- Numpy의 ndarray와 비슷
  - 행렬의 개념
- GPU에서 사용가능
- Tensor 사용하여 모델의 입력과 출력, 모델의 매개변수사용



Module

nn.Module

- PyTorch의 모든 모듈은 torch.nn.Module 의 하위 클래스(subclass)
- 신경망은 하나의 모듈. 다른 모듈(layer)로 구성됨.

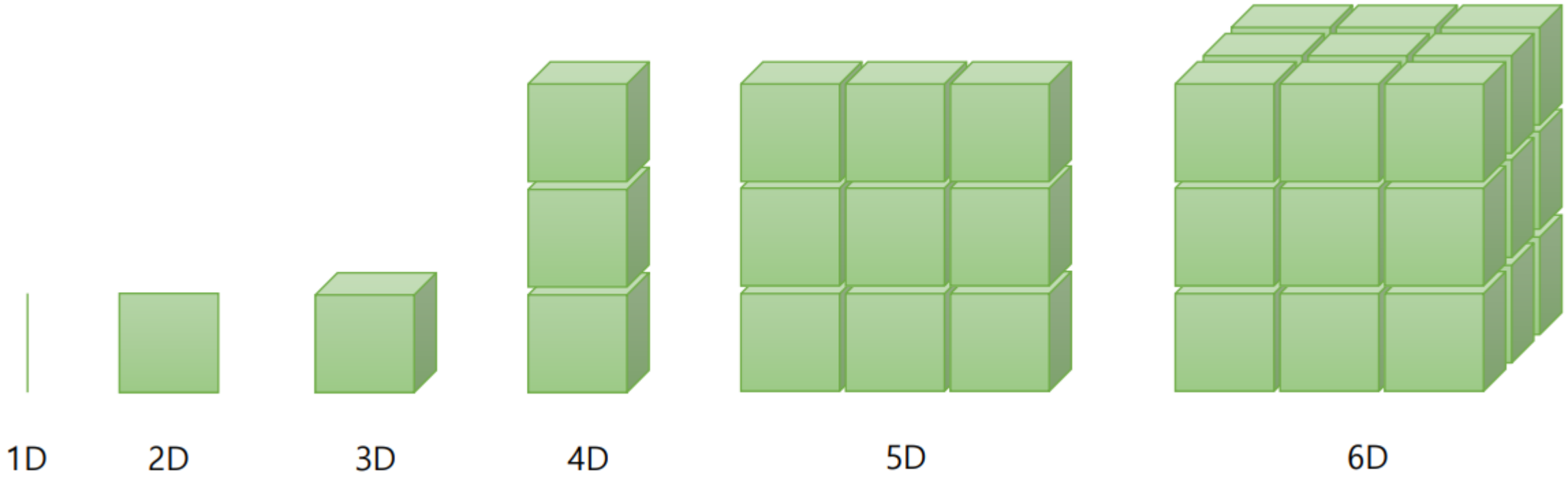
```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10),
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits
```

```
[docs]class Linear(Module):
    r"""Applies a linear transfo
```

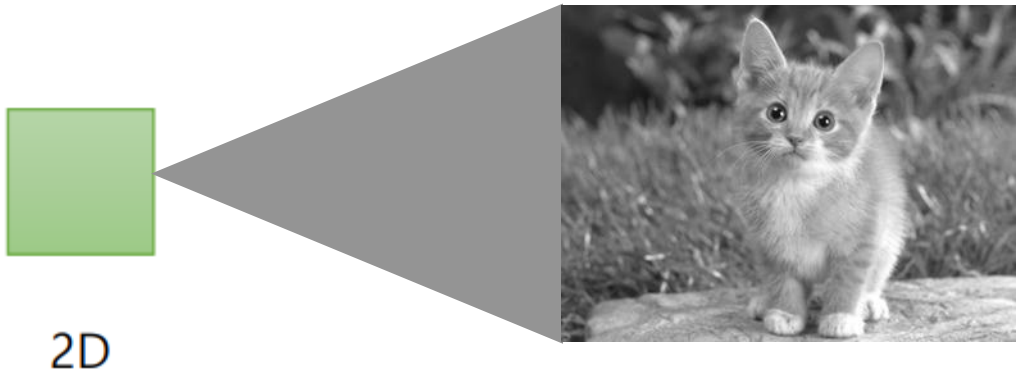
Tensor

Tensor



Tensor

- 2D Tensor
  - (image width, image height) ; 이미지 데이터
  - dimension = 0 , 1




```
[ 59,  60,  61,  ...,  76,  64,  57]
[ 60,  61,  62,  ...,  76,  64,  58]
[ 61,  62,  62,  ...,  74,  64,  59]
...,
[ 83,  95, 108,  ..., 195, 192, 191]
[ 93,  99, 120,  ..., 176, 174, 175]
[114,  99, 106,  ..., 172, 167, 162]
```

dim =0

dim =1

Tensor

- 3D tensor

(batch size, image width, image height) ; 이미지 데이터
dimension = 0 , 1, 2

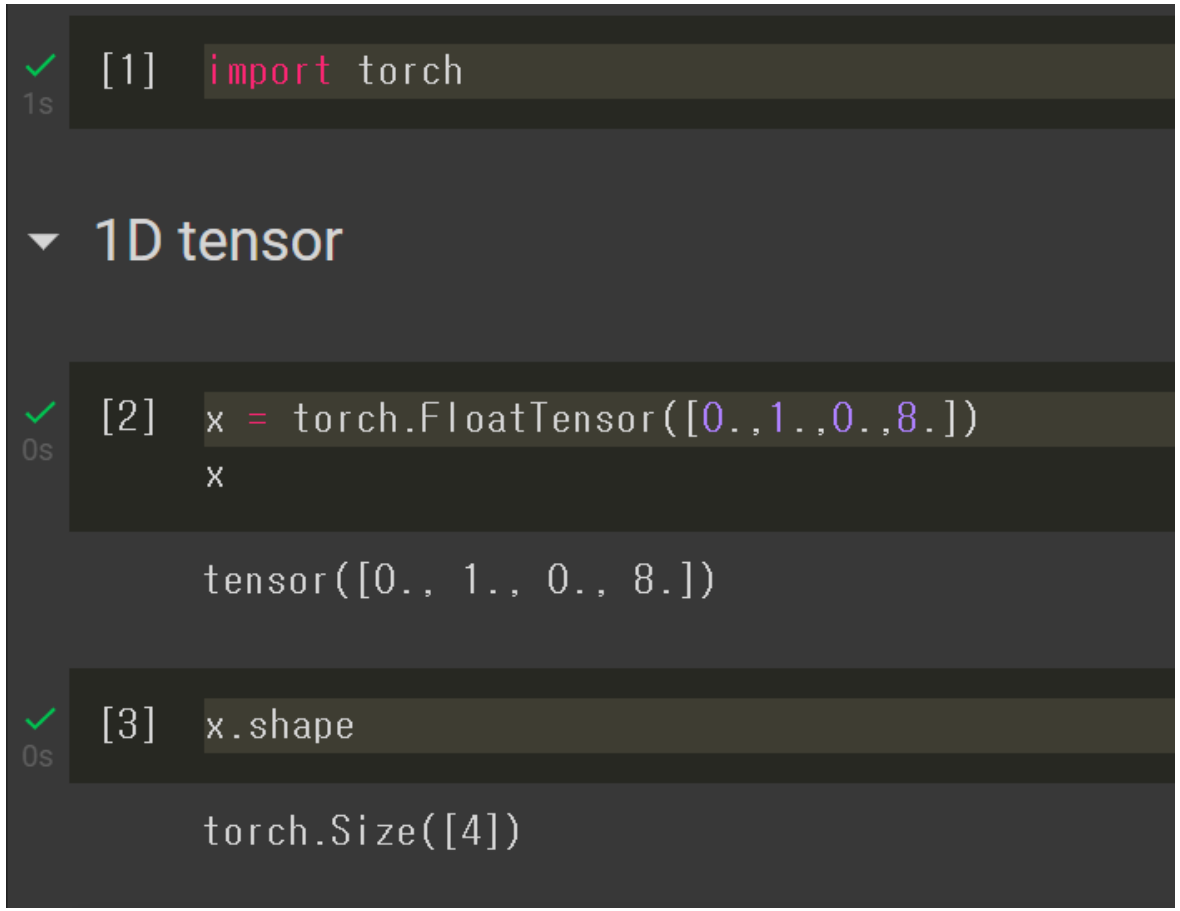
3D

* batch size

batch size : input으로 들어가는 데이터 수

Tensor

- 1D Tensor



pytorch import

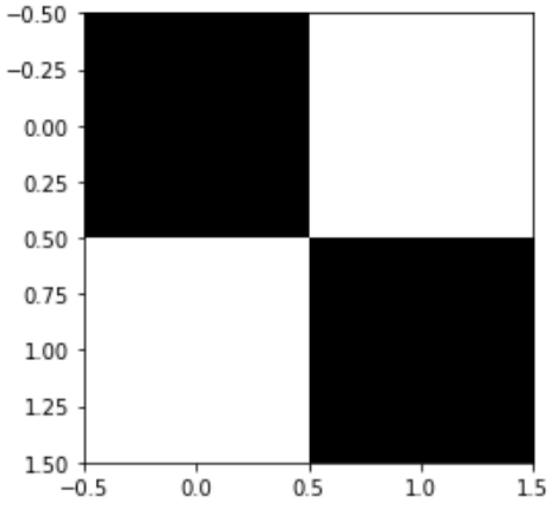
FloatTensor로 float type tensor 선언

Tensor

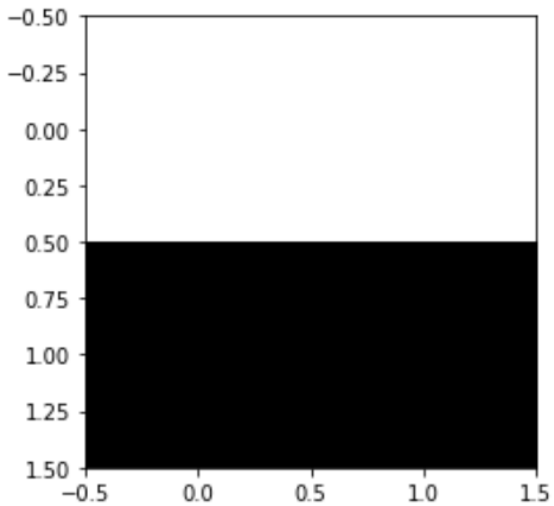
- 3D Tensor

0이 검은색, 1이 흰색일때

[[0,1], [1,0]]



[[1,1], [0,0]]



(batch size, image width, image height) = (2, 2, 2)

3D tensor

```
[15] x = torch.FloatTensor([ [[0.,1.],[1.,0.,]],
                             [[1.,1,],[0.,0.]]])
     x
```

```
tensor([[[0., 1.],
         [1., 0.]],

        [[1., 1.],
         [0., 0.]]])
```

```
x.shape
```

```
torch.Size([2, 2, 2])
```

Tensor

- 덧셈

```
[17]  x = torch.FloatTensor([[0.,1.],[1.,0.,]])
      y = torch.FloatTensor([[1.,1,],[0.,0.]])
      x+y

      tensor([[1., 2.],
              [1., 0.]])

[23]  x = torch.FloatTensor([[0.,1.],[1.,0.,]])
      y = torch.FloatTensor([3])
      x+y

      tensor([[3., 4.],
              [4., 3.]])
```

Broadcasting
차원이 일치하지 않아도 사칙연산이 가능하게 한다.

Tensor

- 곱셈

```
[19]  x.matmul(y)

      tensor([[0., 0.],
              [1., 1.]])

[20]  x*y

      tensor([[0., 1.],
              [0., 0.]])

[21]  x.mul(y)

      tensor([[0., 1.],
              [0., 0.]])
```

$$x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, y = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

1. 매트릭스 곱

$$x * y = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

2. element-wise 곱

$$x * y = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Tensor

- Mean

```
[29]  x = torch.FloatTensor([[2.,2.],[4.,4.,]])

      print(x.mean())
      print(x.mean(dim=0))
      print(x.mean(dim=1))

      tensor(3.)
      tensor([3., 3.])
      tensor([2., 4.])
```

$$x = \begin{bmatrix} 2 & 2 \\ 4 & 4 \end{bmatrix}$$

Tensor

- Concatenate

```
x = torch.FloatTensor([[2.,2.],[4.,4.,]])
y = torch.FloatTensor([[5.,5.],[10.,10.,]])
```

```
[33] print(torch.cat([x,y],dim=0))
     print(torch.cat([x,y],dim=1))
```

```
tensor([[ 2.,  2.],
        [ 4.,  4.],
        [ 5.,  5.],
        [10., 10.]])
tensor([[ 2.,  2.,  5.,  5.],
        [ 4.,  4., 10., 10.]])
```

$$x = \begin{bmatrix} 2 & 2 \\ 4 & 4 \end{bmatrix}, y = \begin{bmatrix} 5 & 5 \\ 10 & 10 \end{bmatrix}$$

Tensor

- View
  - Reshape tensor size
  - x.view(-1,2)
    - -1 torch가 알아서 계산

```
[2]  x=torch.FloatTensor([[[0.,1.],[1.,0.,]],
                          [[1.,1,],[0.,0.]]])
     x.shape

     torch.Size([2, 2, 2])

[3]  x.view(-1,2).shape

     torch.Size([4, 2])

[4]  x.view(-1,1,2).shape

     torch.Size([4, 1, 2])

     x.view(-1,-1,2).shape
```

Tensor

- Squeeze
  - 1인 차원 줄이기
- Unsqueeze
  - 차원 1 늘리기

```
x=torch.FloatTensor([[[0.],[1.], [2.]]])
x.shape
```

```
torch.Size([1, 3, 1])
```

```
[7] torch.squeeze(x,dim=0).shape
```

```
torch.Size([3, 1])
```

```
[8] torch.squeeze(x,dim=2).shape
```

```
torch.Size([1, 3])
```

```
[9] torch.unsqueeze(x,dim=0).shape
```

```
torch.Size([1, 1, 3, 1])
```

```
[11] torch.unsqueeze(x,dim=2).shape
```

```
torch.Size([1, 3, 1, 1])
```

Appendix

Pytorch 설치

Pytorch 설치

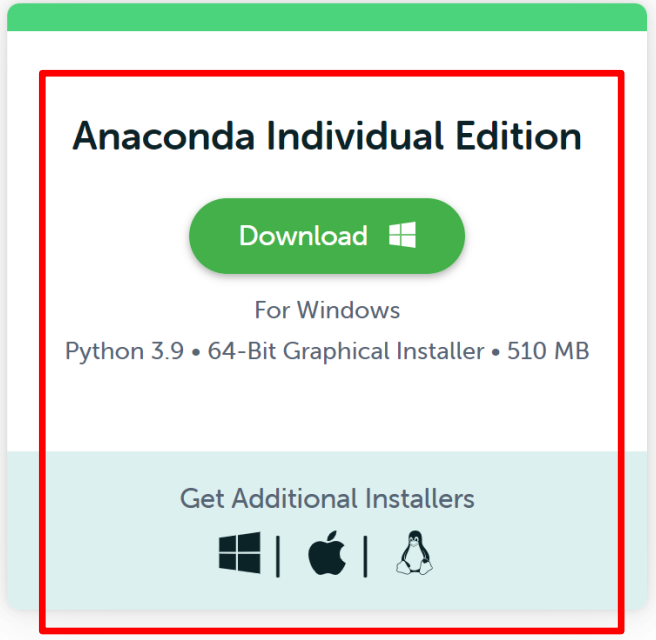
1. Anaconda 설치

https://www.anaconda.com/products/individual

Individual Edition

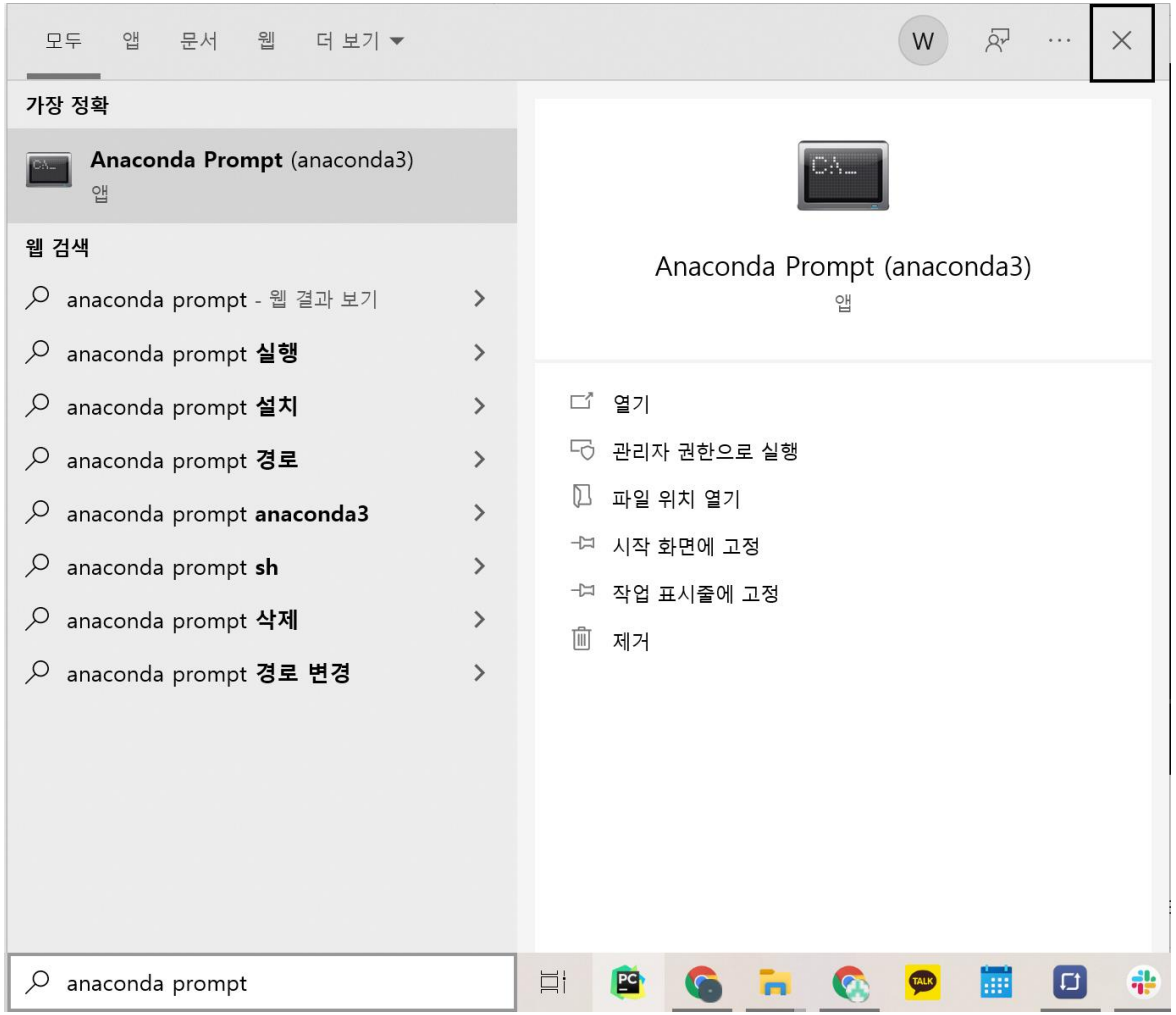
Your data science toolkit

With over 25 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.



Pytorch 설치

2. Anaconda Prompt 열기



Pytorch 설치

3. 가상환경 생성

"pytorch"라는 가상환경을 만든다.

- conda create -n pytorch python=3.7

```
(base) C:₩Users₩whanh>conda create -n pytorch python=3.7
```

- conda activate pytorch
  - 가상환경에 접근

```
(base) C:₩Users₩whanh>conda activate pytorch

(pytorch) C:₩Users₩whanh>
```

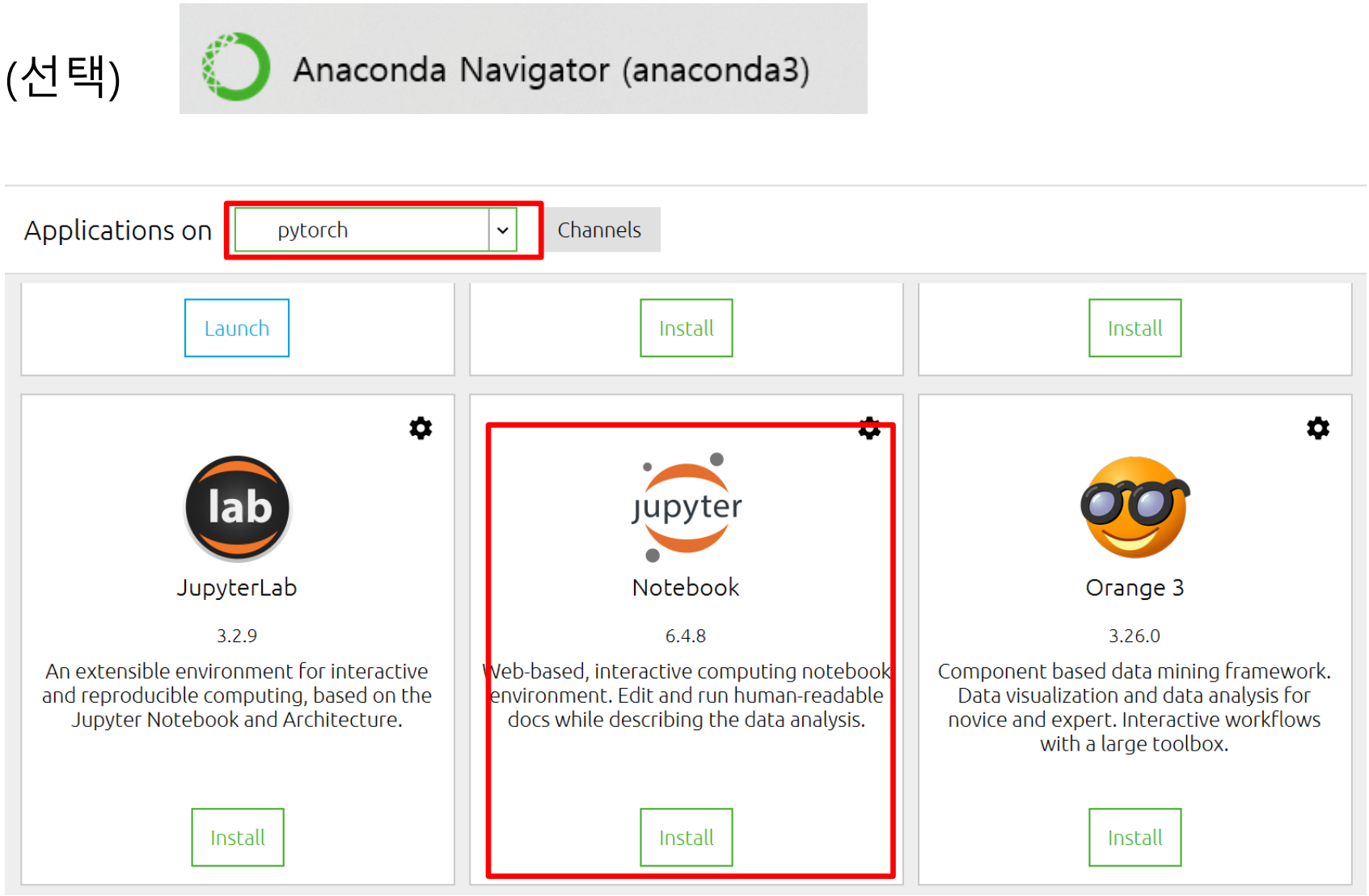
4. Pytorch 설치

- conda install pytorch

```
(pytorch) C:₩Users₩whanh>conda install pytorch
```

Pytorch 설치

5. Anaconda Navigatior (선택)
Jupyter notebook 설치



Pytorch 설치

5. Jupyter notebook 실행