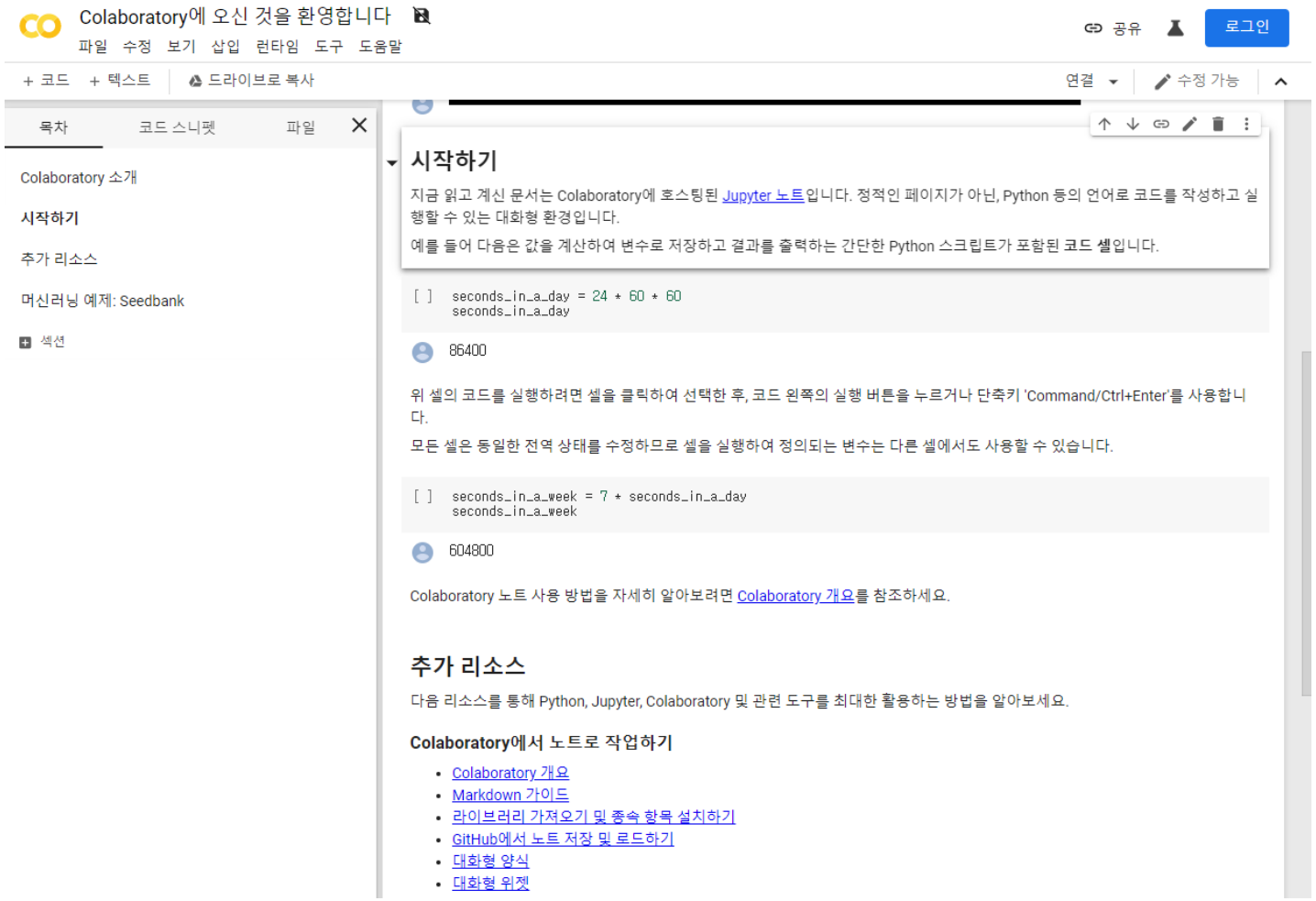
Google Colab 실습

2019-2nd semester

Machine Learning Course

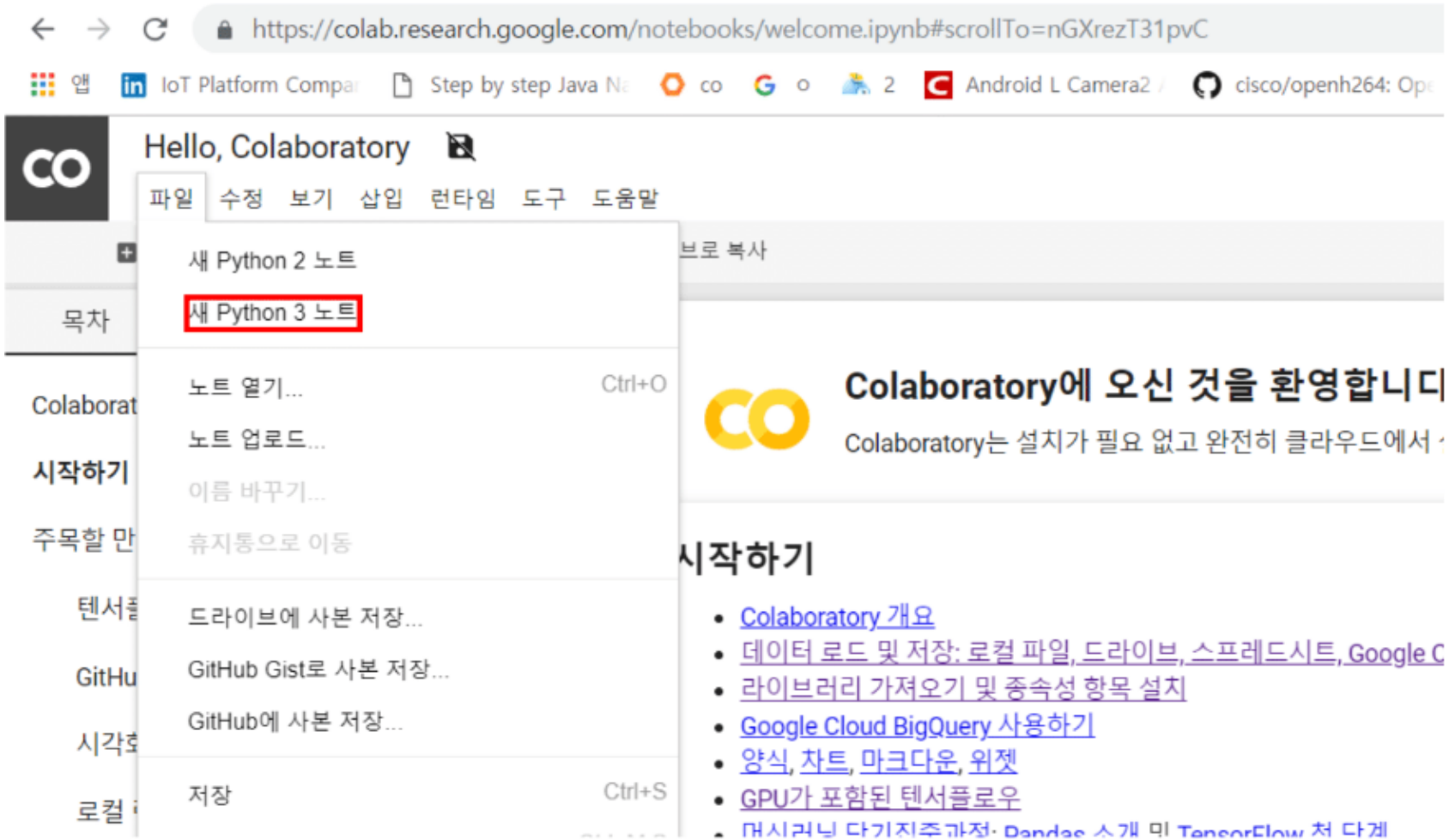
1. Google Colab이란?

- Google Colaboratory의 줄임말
- 클라우드 기반 Jupyter Notebook UI 및 기능 제공
- 머신러닝을 위한 GPU(Tesla K80) 및 TPU를 무료로 제공



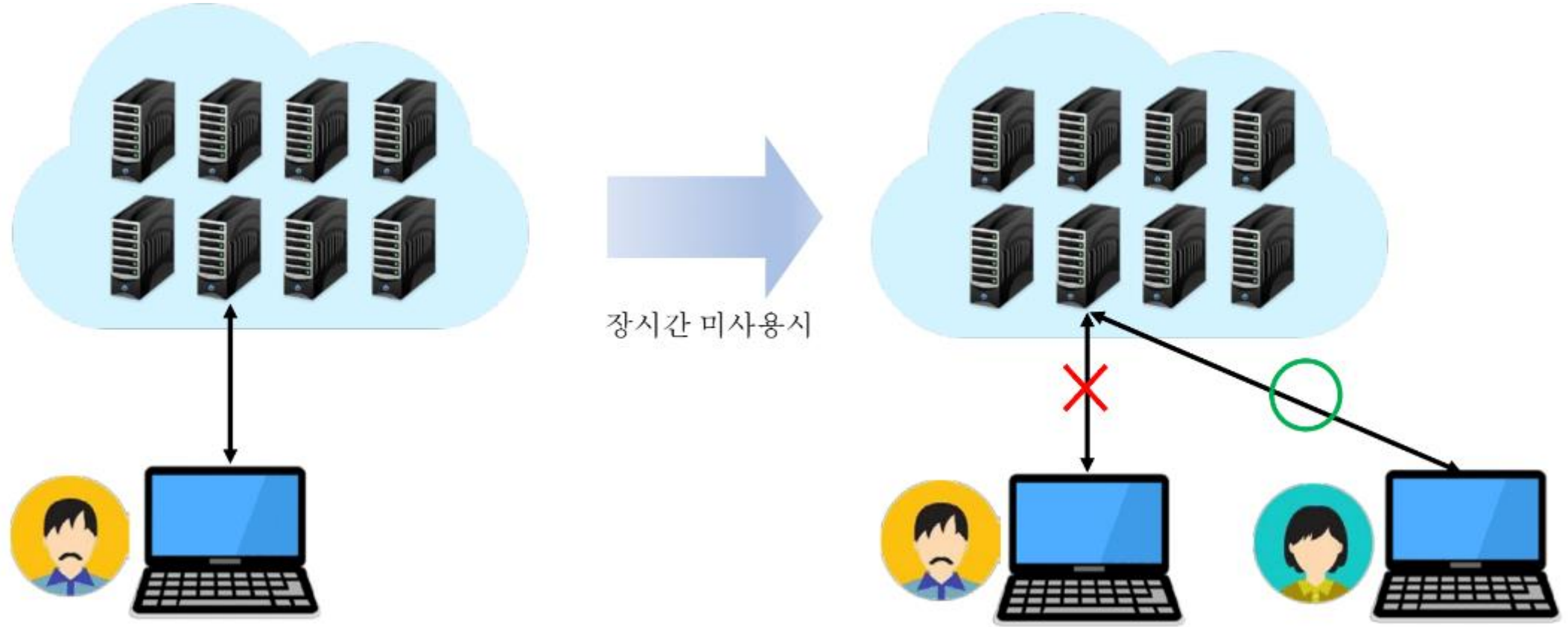
2. Google Colab 접속

1. Google 계정으로 로그인
2. http://colab.research.google.com
3. 새 Python3 노트 실행



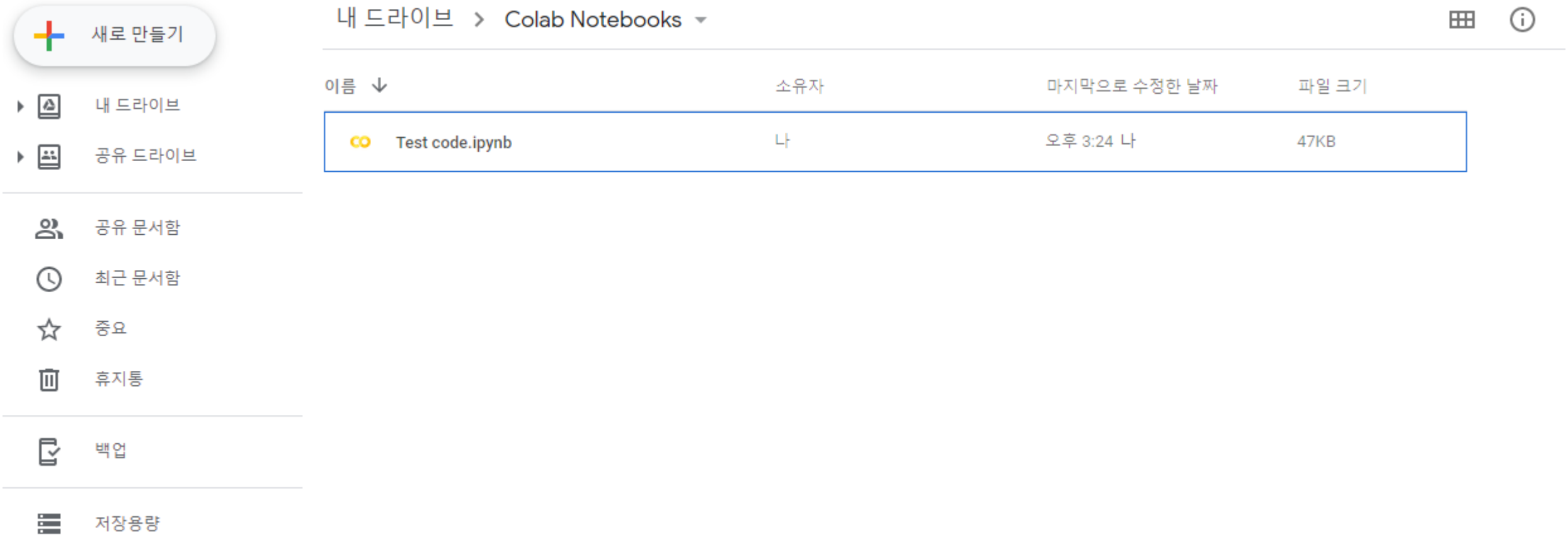
2-1. Google Colab – Google Drive 연동

- 구글 드라이브를 연동하는 이유? -> 임시로 제공받는 서버이기 때문에 삭제될 수 있음(장시간 미사용시 삭제 -> 사람이 많이 몰리면 몰릴수록 삭제될 가능성이 높음)
- 데이터를 제공받은 서버 내에 두지 않고, 외부 스토리지인 구글 드라이브와 연동하여 사용해야 함



2-1. Google Colab – Google Drive 연동

- 기본적으로 .ipynb 및 .py 파일의 경우 내 드라이브 -> Colab Notebooks에 저장
- 데이터를 제공받은 서버 내에 두지 않고, 외부 스토리지인 구글 드라이브와 연동하여 사용해야 함



3. 예제 코드 Test

- 새 Python3 노트에서 GPU 연동하기
- Colab에서는 하루 최대 12시간동안 GPU(Tesla K80)을 제공
- TPU 연동을 하지 않는 이유
  -> Pytorch에서 TPU를 사용할 수 있는 기능을 제공하지 않음
  -> Tensorflow 및 Keras에서는 사용 가능

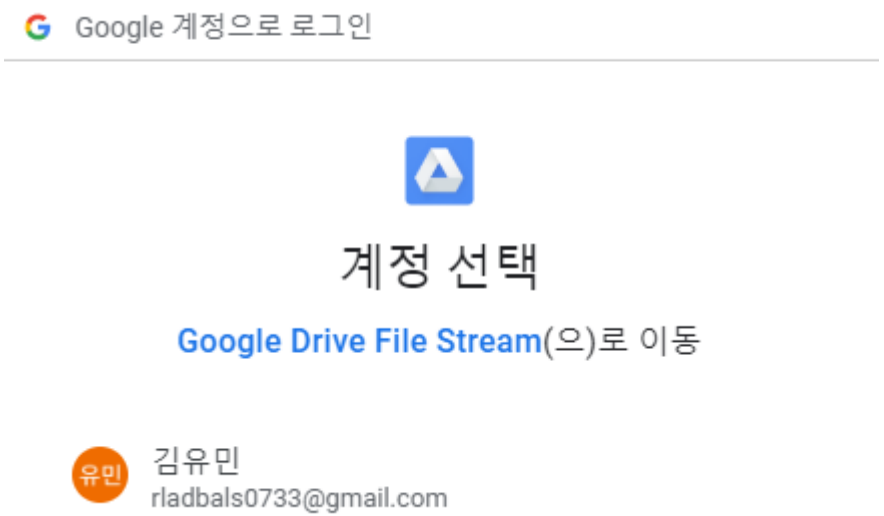
<GPU 연동 화면>

3. 예제 코드 Test

- 새 Python3 노트에서 "+코드" 를 눌러 아래 연동 코드 작성 및 실행
- Authorization 코드 URL을 클릭 후 Authorization 코드를 복사하여 입력
- 앞으로 "+코드"를 눌러 새로운 셀을 생성하여 예제 코드를 실행할 것

```
[1]   from google.colab import drive
      drive.mount('/content/gdrive')
      import sys
      sys.path.append("/content/gdrive/My Drive/Colab Notebooks")
```

<Google Drive 연동 코드>



<URL 클릭 후 화면>

```
Enter your authorization code:
· · · · · · · · · ·
Mounted at /content/gdrive
```

<Authorization 코드 입력 후 실행 화면>

3. 예제 코드 Test

- 새로운 셀 추가 하여 아래 코드 입력
- 본 예제는 MNIST 데이터 셋을 통해 숫자 데이터를 구별하는 Task를 학습할 것임
- 학습을 위해 Pytorch 라이브러리 사용(2019.01.26부터 Colab에 내장됨)

<MNIST 데이터셋>

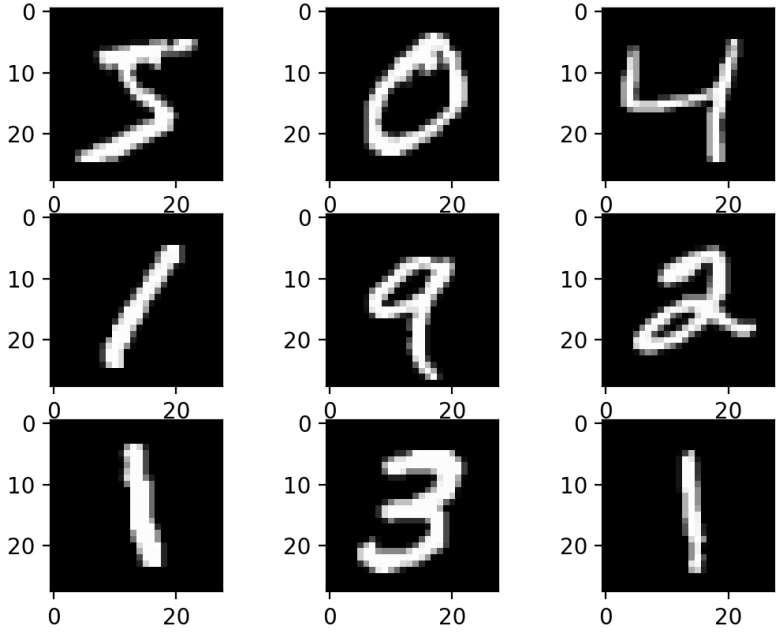
```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torchvision import datasets, transforms

import numpy as np
import matplotlib.pyplot as plt

if torch.cuda.is_available():
    device = torch.device('cuda')
else:
    device = torch.device('cpu')

print('Using PyTorch version:', torch.__version__, ' Device:', device)
```

```
Using PyTorch version: 1.1.0  Device: cuda
```

<Pytorch 및 CUDA 버전 확인>

3. 예제 코드 Test

- 한 학습 iteration에 사용할 이미지 개수(Batch size) 설정
- MNIST Dataset은 28x28 gray-scale 숫자 이미지들로 구성됨(6만장의 학습 데이터, 1만장의 테스트 데이터)
- MNIST Dataset은 Pytorch에 내장되어 있어 즉시 사용가능



<MNIST 이미지>

```
[ ]   batch_size = 32

      train_dataset = datasets.MNIST('./data',
                                     train=True,
                                     download=True,
                                     transform=transforms.ToTensor())

      validation_dataset = datasets.MNIST('./data',
                                          train=False,
                                          transform=transforms.ToTensor())

      train_loader = torch.utils.data.DataLoader(dataset=train_dataset,
                                                 batch_size=batch_size,
                                                 shuffle=True)

      validation_loader = torch.utils.data.DataLoader(dataset=validation_dataset,
                                                      batch_size=batch_size,
                                                      shuffle=False)
```

<MNIST 데이터셋 로드>

3. 예제 코드 Test

- 학습에는 입력 이미지와 각 입력 이미지에 해당하는 라벨 데이터를 사용함
- 입력으로 들어오는 이미지 확인

```
[ ]   for (X_train, y_train) in train_loader:
          print('X_train:', X_train.size(), 'type:', X_train.type())
          print('y_train:', y_train.size(), 'type:', y_train.type())
          break
```

<입력 이미지 및 라벨 데이터 사이즈 확인>

```
[ ]   pltsize=1
      plt.figure(figsize=(10*pltsize, pltsize))

      for i in range(10):
          plt.subplot(1,10,i+1)
          plt.axis('off')
          plt.imshow(X_train[i,:,:,:].numpy().reshape(28,28), cmap="gray")
          plt.title('Class: '+str(y_train[i].item()))
```

<입력 이미지 확인>

3. 예제 코드 Test

- 학습에 사용하는 모델(뉴럴 네트워크) 만들고 확인

```
[ ]  class Net(nn.Module):
         def __init__(self):
             super(Net, self).__init__()
             self.fc1 = nn.Linear(28*28, 50)
             self.fc1_drop = nn.Dropout(0.2)
             self.fc2 = nn.Linear(50, 50)
             self.fc2_drop = nn.Dropout(0.2)
             self.fc3 = nn.Linear(50, 10)

         def forward(self, x):
             x = x.view(-1, 28*28)
             x = F.relu(self.fc1(x))
             x = self.fc1_drop(x)
             x = F.relu(self.fc2(x))
             x = self.fc2_drop(x)
             return F.log_softmax(self.fc3(x), dim=1)

     model = Net().to(device)
     optimizer = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0.5)
     criterion = nn.CrossEntropyLoss()

     print(model)
```

```
Net(
  (fc1): Linear(in_features=784, out_features=50, bias=True)
  (fc1_drop): Dropout(p=0.2)
  (fc2): Linear(in_features=50, out_features=50, bias=True)
  (fc2_drop): Dropout(p=0.2)
  (fc3): Linear(in_features=50, out_features=10, bias=True)
)
```

<모델 구조 확인>

3. 예제 코드 Test

- 모델을 학습시키는 함수(train), 테스트 하는 함수(validate) 작성

```
[ ]  def train(epoch, log_interval=200):
         # Set model to training mode
         model.train()

         # Loop over each batch from the training set
         for batch_idx, (data, target) in enumerate(train_loader):
             # Copy data to GPU if needed
             data = data.to(device)
             target = target.to(device)

             # Zero gradient buffers
             optimizer.zero_grad()

             # Pass data through the network
             output = model(data)

             # Calculate loss
             loss = criterion(output, target)

             # Backpropagate
             loss.backward()

             # Update weights
             optimizer.step()

             if batch_idx % log_interval == 0:
                 print('Train Epoch: {} [{}/{} ({:.0f}%)]₩tLoss: {:.6f}'.format(
                     epoch, batch_idx * len(data), len(train_loader.dataset),
                     100. * batch_idx / len(train_loader), loss.data.item()))
```

<학습 함수>

```
[ ]  def validate(loss_vector, accuracy_vector):
         model.eval()
         val_loss, correct = 0, 0
         for data, target in validation_loader:
             data = data.to(device)
             target = target.to(device)
             output = model(data)
             val_loss += criterion(output, target).data.item()
             pred = output.data.max(1)[1] # get the index of the max log-probability
             correct += pred.eq(target.data).cpu().sum()

         val_loss /= len(validation_loader)
         loss_vector.append(val_loss)

         accuracy = 100. * correct.to(torch.float32) / len(validation_loader.dataset)
         accuracy_vector.append(accuracy)

         print('₩nValidation set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)₩n'.format(
             val_loss, correct, len(validation_loader.dataset), accuracy))
```

<테스트 함수>

3. 예제 코드 Test

- 작성한 학습 함수, 테스트 함수를 호출하여 모델 학습 epochs 만큼 진행

```
[ ]   epochs = 5

      lossv, accv = [], []
      for epoch in range(1, epochs + 1):
          train(epoch)
          validate(lossv, accv)
```

<학습 및 테스트 진행 코드>

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.297845
Train Epoch: 1 [6400/60000 (11%)]       Loss: 1.977456
Train Epoch: 1 [12800/60000 (21%)]      Loss: 1.247260
Train Epoch: 1 [19200/60000 (32%)]      Loss: 0.886104
Train Epoch: 1 [25600/60000 (43%)]      Loss: 0.756474
Train Epoch: 1 [32000/60000 (53%)]      Loss: 0.583003
Train Epoch: 1 [38400/60000 (64%)]      Loss: 0.537233
Train Epoch: 1 [44800/60000 (75%)]      Loss: 0.299226
Train Epoch: 1 [51200/60000 (85%)]      Loss: 0.485791
Train Epoch: 1 [57600/60000 (96%)]      Loss: 0.525107

Validation set: Average loss: 0.3402, Accuracy: 9064/10000 (91%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 0.527739
Train Epoch: 2 [6400/60000 (11%)]       Loss: 0.370524
Train Epoch: 2 [12800/60000 (21%)]      Loss: 0.380130
Train Epoch: 2 [19200/60000 (32%)]      Loss: 0.453238
Train Epoch: 2 [25600/60000 (43%)]      Loss: 0.266777
Train Epoch: 2 [32000/60000 (53%)]      Loss: 0.354032
Train Epoch: 2 [38400/60000 (64%)]      Loss: 0.494110
Train Epoch: 2 [44800/60000 (75%)]      Loss: 0.364410
Train Epoch: 2 [51200/60000 (85%)]      Loss: 0.332283
Train Epoch: 2 [57600/60000 (96%)]      Loss: 0.356527

Validation set: Average loss: 0.2521, Accuracy: 9250/10000 (92%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 0.316200
Train Epoch: 3 [6400/60000 (11%)]       Loss: 0.253810
Train Epoch: 3 [12800/60000 (21%)]      Loss: 0.589426
Train Epoch: 3 [19200/60000 (32%)]      Loss: 0.619542
Train Epoch: 3 [25600/60000 (43%)]      Loss: 0.196636
Train Epoch: 3 [32000/60000 (53%)]      Loss: 0.267882
Train Epoch: 3 [38400/60000 (64%)]      Loss: 0.289375
Train Epoch: 3 [44800/60000 (75%)]      Loss: 0.380577
Train Epoch: 3 [51200/60000 (85%)]      Loss: 0.594070
Train Epoch: 3 [57600/60000 (96%)]      Loss: 0.393653

Validation set: Average loss: 0.2082, Accuracy: 9372/10000 (94%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 0.400030
Train Epoch: 4 [6400/60000 (11%)]       Loss: 0.132549
Train Epoch: 4 [12800/60000 (21%)]      Loss: 0.376023
Train Epoch: 4 [19200/60000 (32%)]      Loss: 0.182976
Train Epoch: 4 [25600/60000 (43%)]      Loss: 0.231020
Train Epoch: 4 [32000/60000 (53%)]      Loss: 0.353738
Train Epoch: 4 [38400/60000 (64%)]      Loss: 0.487107
Train Epoch: 4 [44800/60000 (75%)]      Loss: 0.171016
Train Epoch: 4 [51200/60000 (85%)]      Loss: 0.473919
Train Epoch: 4 [57600/60000 (96%)]      Loss: 0.219377

Validation set: Average loss: 0.1852, Accuracy: 9440/10000 (94%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 0.085754
Train Epoch: 5 [6400/60000 (11%)]       Loss: 0.184135
Train Epoch: 5 [12800/60000 (21%)]      Loss: 0.087964
Train Epoch: 5 [19200/60000 (32%)]      Loss: 0.243350
Train Epoch: 5 [25600/60000 (43%)]      Loss: 0.252898
Train Epoch: 5 [32000/60000 (53%)]      Loss: 0.303688
Train Epoch: 5 [38400/60000 (64%)]      Loss: 0.656003
Train Epoch: 5 [44800/60000 (75%)]      Loss: 0.135485
Train Epoch: 5 [51200/60000 (85%)]      Loss: 0.191508
Train Epoch: 5 [57600/60000 (96%)]      Loss: 0.229463

Validation set: Average loss: 0.1640, Accuracy: 9501/10000 (95%)
```

<학습 및 테스트 진행 화면>

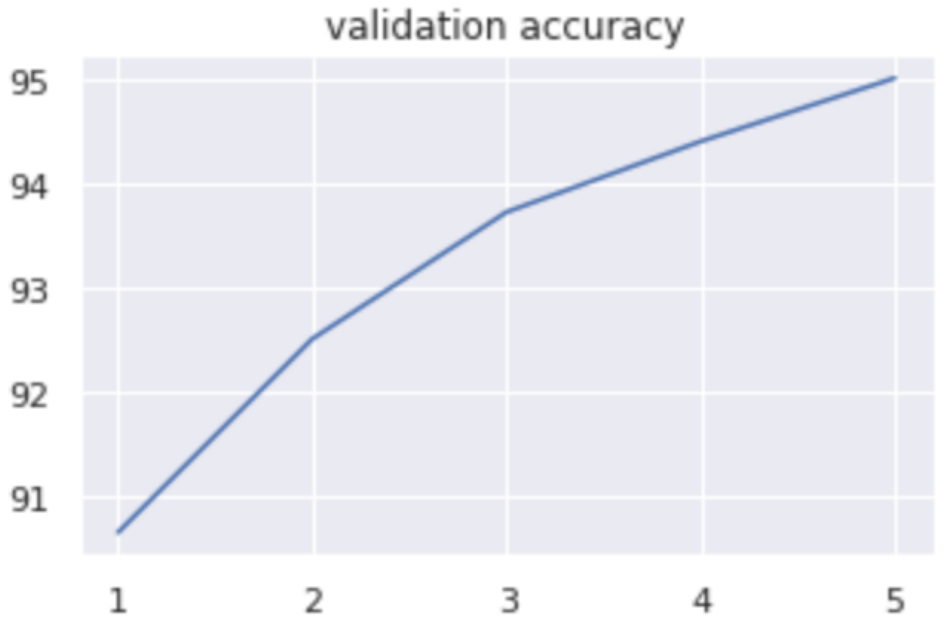
3. 예제 코드 Test

- Matplotlib을 이용하여 학습 완료 후 epoch 마다의 테스트 결과 확인

```
[ ]  plt.figure(figsize=(5,3))
     plt.plot(np.arange(1,epochs+1), lossv)
     plt.title('validation loss')

     plt.figure(figsize=(5,3))
     plt.plot(np.arange(1,epochs+1), accv)
     plt.title('validation accuracy');
```

<결과 확인 코드>

<테스트 Loss 결과>



<테스트 Accuracy 결과>