**Google Colab 사용하기**

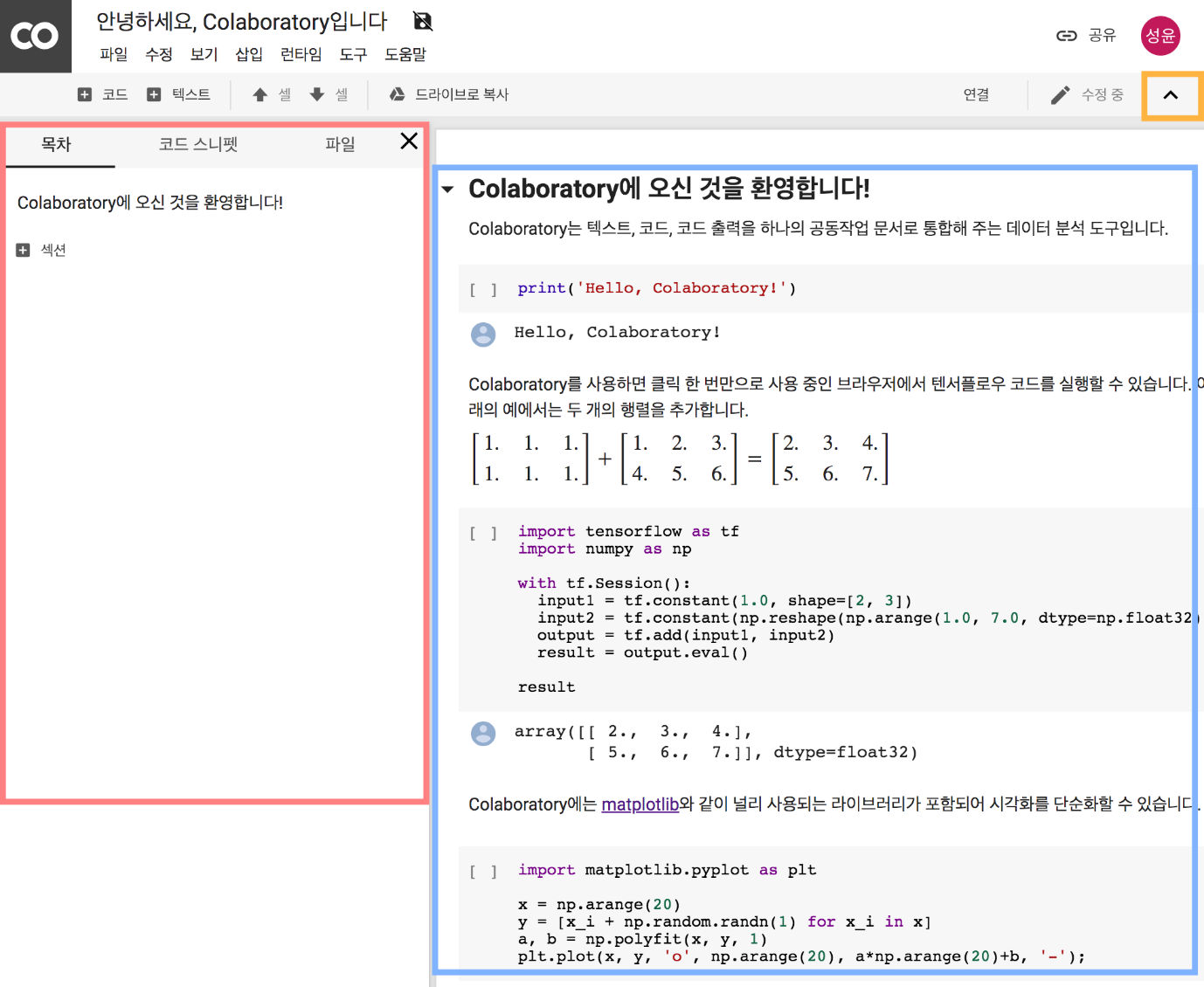
**30 Aug 2018 in**[**Data**](https://zzsza.github.io/category/data/)**on**[**Deep-Learning**](https://zzsza.github.io/tag/data-dl/)

* Google의 Colab 사용법에 대해 정리한 글입니다
* 이 글은 계속 업데이트 될 예정입니다!
* 목차
  + [UI](https://zzsza.github.io/data/2018/08/30/google-colab/#ui)
  + [상단 설정](https://zzsza.github.io/data/2018/08/30/google-colab/#%EC%83%81%EB%8B%A8-%EC%84%A4%EC%A0%95)
  + [구글 드라이브와 Colab 연동](https://zzsza.github.io/data/2018/08/30/google-colab/#%EA%B5%AC%EA%B8%80-%EB%93%9C%EB%9D%BC%EC%9D%B4%EB%B8%8C%EC%99%80-colab-%EC%97%B0%EB%8F%99)
  + [구글 드라이브와 로컬 연동](https://zzsza.github.io/data/2018/08/30/google-colab/#%EA%B5%AC%EA%B8%80-%EB%93%9C%EB%9D%BC%EC%9D%B4%EB%B8%8C%EC%99%80-%EB%A1%9C%EC%BB%AC-%EC%97%B0%EB%8F%99)
  + [Tensorflow 2.0 설치하기](https://zzsza.github.io/data/2018/08/30/google-colab/#tensorflow-%EC%84%A4%EC%B9%98%ED%95%98%EA%B8%B0)
  + [PyTorch 사용하기](https://zzsza.github.io/data/2018/08/30/google-colab/#pytorch-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0)
  + [KoNLPy 설치](https://zzsza.github.io/data/2018/08/30/google-colab/#konlpy-%EC%84%A4%EC%B9%98)
  + [Github 코드를 Colab에서 사용하기](https://zzsza.github.io/data/2018/08/30/google-colab/#github-%EC%BD%94%EB%93%9C%EB%A5%BC-colab%EC%97%90%EC%84%9C-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0)
  + [BigQuery 사용하기](https://zzsza.github.io/data/2018/08/30/google-colab/#bigquery-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0)
  + [Matplotlib에서 한글 사용하기](https://zzsza.github.io/data/2018/08/30/google-colab/#matplotlib%EC%97%90%EC%84%9C-%ED%95%9C%EA%B8%80-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0)
  + [TensorBoard 사용하기](https://zzsza.github.io/data/2018/08/30/google-colab/#tensorboard-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0)
  + [JDK 설치하기](https://zzsza.github.io/data/2018/08/30/google-colab/#jdk-%EC%84%A4%EC%B9%98%ED%95%98%EA%B8%B0)
  + [Google Storage에서 파일 읽기](https://zzsza.github.io/data/2018/08/30/google-colab/#google-storage%EC%97%90%EC%84%9C-%ED%8C%8C%EC%9D%BC-%EC%9D%BD%EA%B8%B0)
  + [MNIST on TPU 소스](https://colab.research.google.com/github/GoogleCloudPlatform/training-data-analyst/blob/master/courses/fast-and-lean-data-science/01_MNIST_TPU_Keras.ipynb)
  + [Kaggle 연동하기](https://zzsza.github.io/data/2018/08/30/google-colab/#kaggle-%EC%97%B0%EB%8F%99%ED%95%98%EA%B8%B0)
  + [코기 모드](https://zzsza.github.io/data/2018/08/30/google-colab/#%EC%BD%94%EA%B8%B0-%EB%AA%A8%EB%93%9C)
  + [런타임 연결](https://zzsza.github.io/data/2018/08/30/google-colab/#%EB%9F%B0%ED%83%80%EC%9E%84-%EC%97%B0%EA%B2%B0)

**Google Colab**

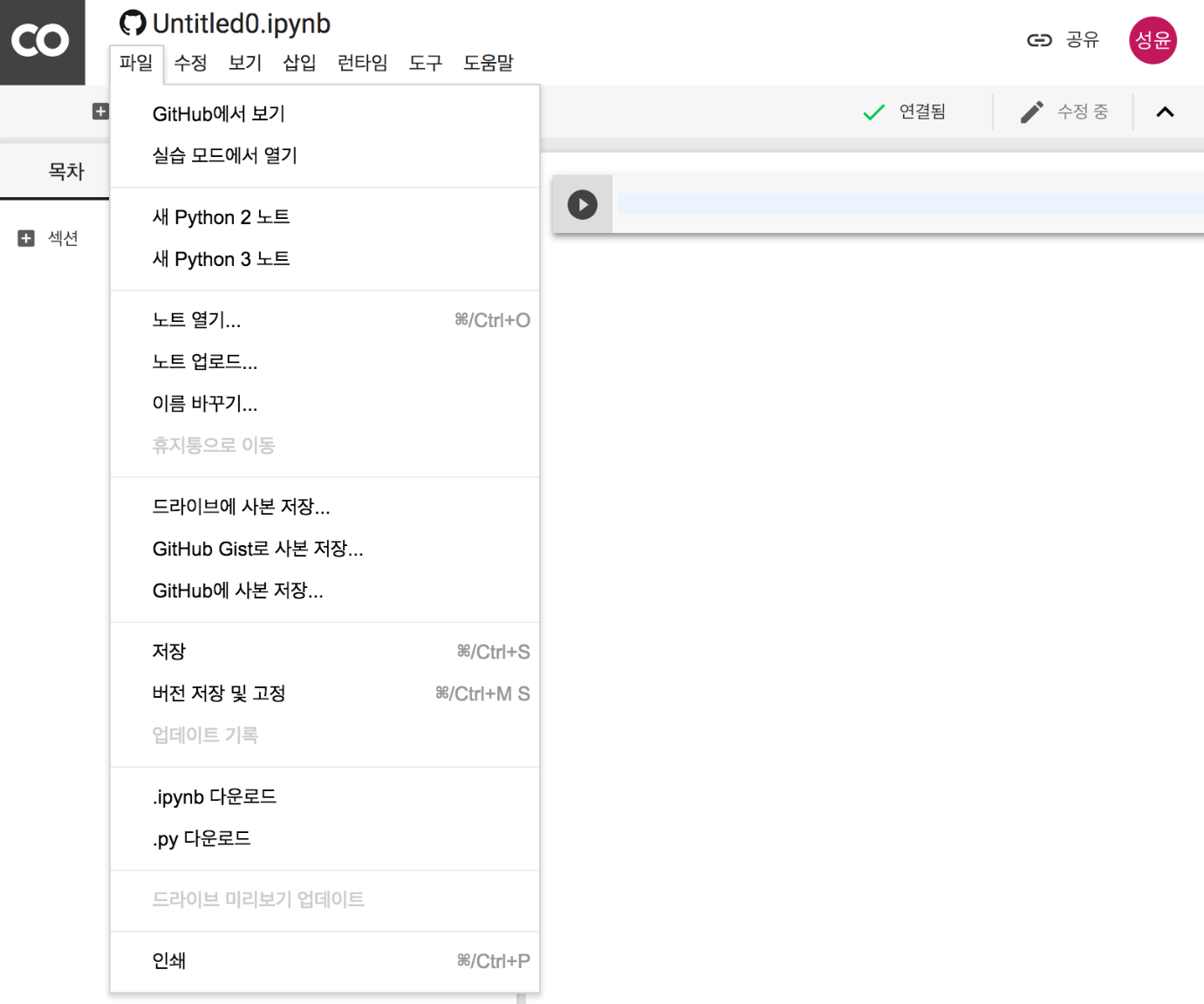
* 풀 네임은 Google Colaboratory
* Google Drive + Jupyter Notebook
  + Google Drive처럼 협업 가능(동시에 수정 가능)
* <https://colab.research.google.com/>로 접속시 사용 가능
* 컴퓨터 사양(18년 8월 기준)
  + Ubuntu 17.10
  + CPU 제논 2.3GHz
  + 메모리 13G
  + **GPU(Tesla K80)** : GPU 없는 제게 희망..
  + TPU도 사용 가능
* GPU 사용시 최대 12시간
* Github의 소스 코드를 Colab에서 사용 가능

**UI**

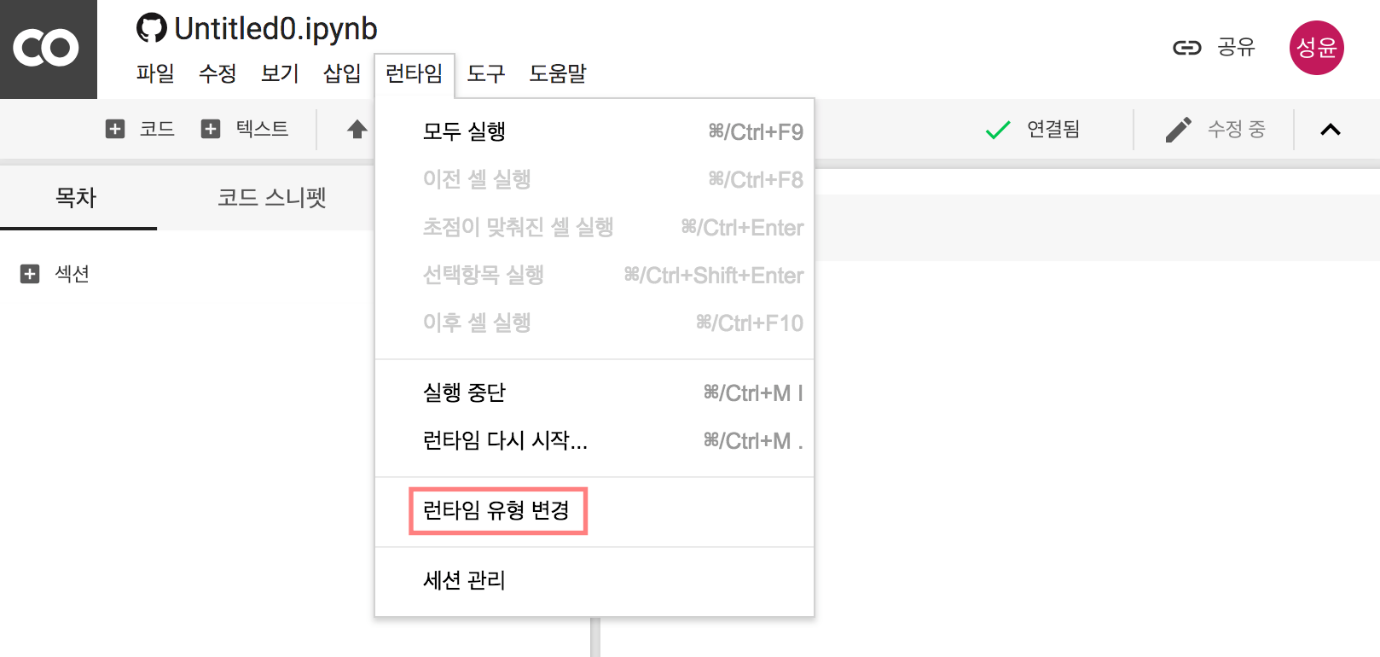


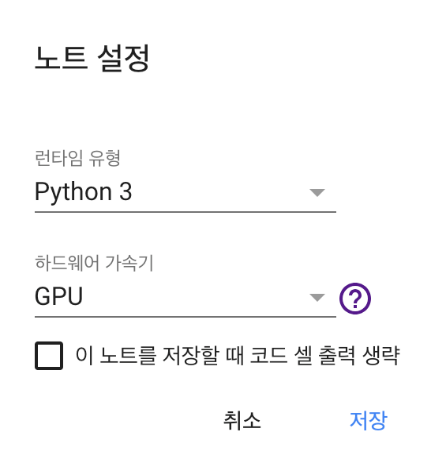
* 붉은 부분
  + 해당 노트북의 목차
  + 코드 스니펫
    - 재사용 가능한 소스 코드로 다양한 예제 코드가 있음
    - Uploading files from your local file system, Using BigQuery, Listing files in Google Drive, Install library, etc
  + 연결된 파일
* 노란 부분
  + 헤더 보이기 유무
* 파란 부분
  + 코드 부분

**상단 설정**



* 새 Python2, 3 노트 : Notebook 파일 생성(GPU 없는 설정)
* 노트 업로드 : 로컬에 있는 노트북 업로드
* Github Gist에 사본 저장 : Gist에 Secret으로 저장(처음에 연동 설정 필요)
* Github에 사본 저장 : Github Repo의 특정 브랜치에 저장



* 런타임 유형 변경 클릭시 GPU 사용 가능
* 

**Colab 사용하기**

* OS 확인
* !cat /etc/issue.net

**하드웨어 사양**

* CPU
* !cat /proc/cpuinfo
* Memory
* !cat /proc/meminfo
* Disk
* !df -h
* GPU
* !nvidia-smi

**구글 드라이브와 Colab 연동**

from google.colab import auth

auth.authenticate\_user()

from google.colab import drive

drive.mount('/content/gdrive')

* colab에서 구글 드라이브 권한 획득
  + 위 명령어 복사 붙여넣기
  + 그 후 나오는 URL로 접속한 후, verification code 입력
  + 단, 매번 이 작업을 해줘야 함….(일정 시간 이후엔 끊김)
* colab에서 drive란 폴더를 만든 후, 우리 구글 드라이브의 root와 drive 폴더를 연결(mount)

!cd gdrive/data; ls-al;

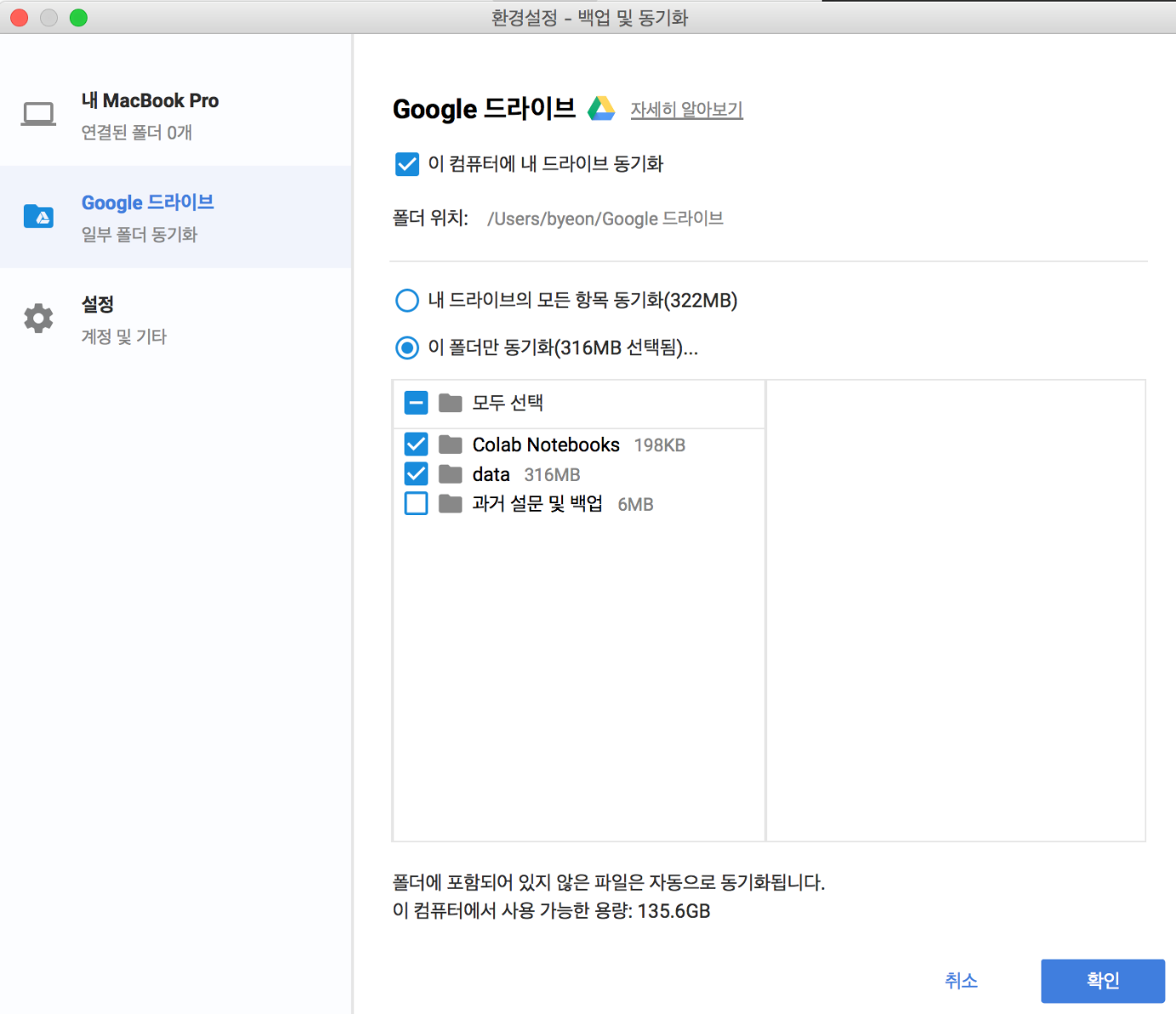
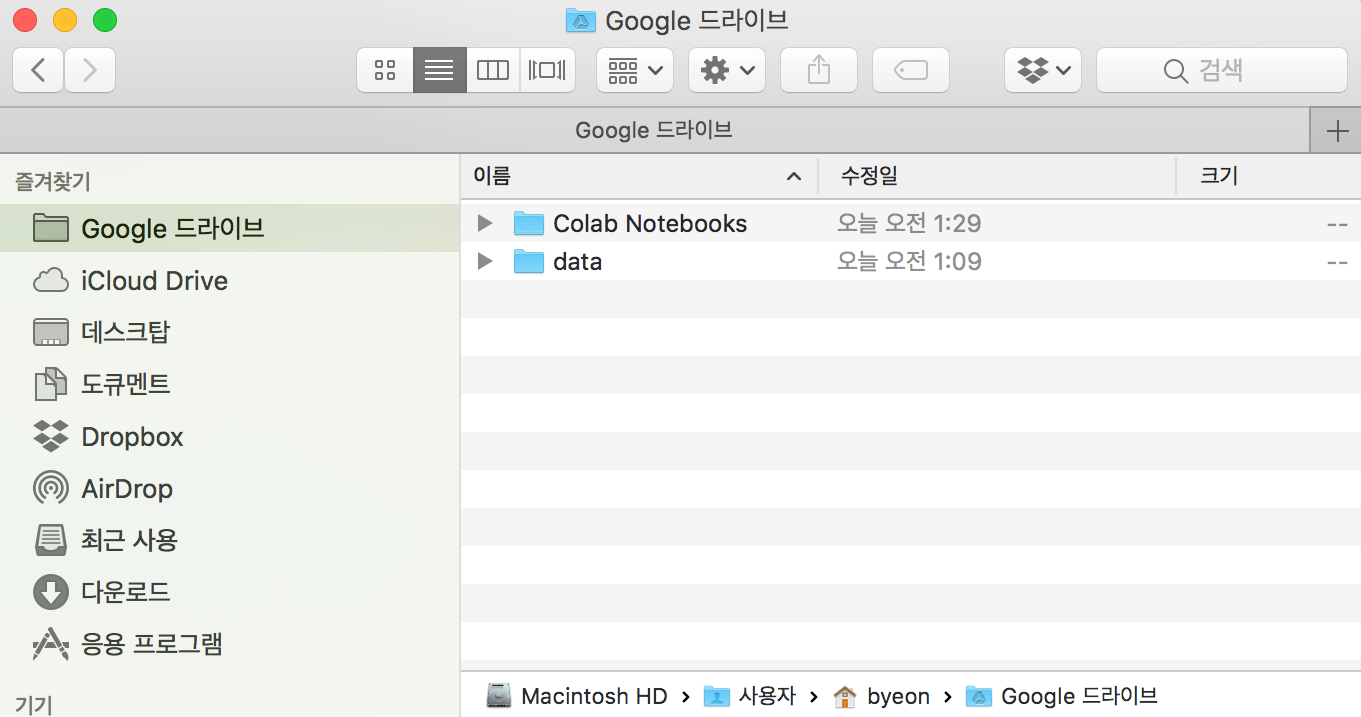
* 구글드라이브의 root에 data란 폴더가 있었음
* train\_activitiy.csv 데이터를 읽어오기

import pandas as pd

df = pd.read\_csv("./gdrive/data/train\_activity.csv")

* 만약 apt-key output should not be parsed (stdout is not a terminal)란 Warning이 나오면 이미 인증이 완료되었다는 뜻이므로 바로 mount하면 됨

**구글 드라이브와 로컬 연동**

* 파일을 하나씩 업로드하지 말고 대량의 파일을 한꺼번에 업로드하고 싶은 경우
* [BackupAndSync](https://www.google.com/drive/download/)를 사용해 로컬과 구글 드라이브를 연동
  + Dropbox처럼 내 로컬의 특정 폴더를 연동
  + 맥북 환경에서 진행
* 위 링크를 클릭해 백업 및 동기화 다운로드
* InstallBackupAndSync.dmg라는 파일을 클릭한 후, (열리지 않으면 우클릭 후 열기) 프로그램 설치
* 맥북 환경이 한글이신 분은 Google에서 백업 및 동기화라는 응용 프로그램이 추가됨(이것도 실행이 안되면 우클릭 후 실행)
* 환경 설정에서 동기화할 폴더 선택
* 
* “폴더 위치”라고 써있는 곳이 이제 Google Drive와 동일하게 연동
* 
* 이제 “폴더 위치”에 원하는 데이터를 저장해두면 Colab에서 사용 가능
  + 단, 크기가 큰 파일은 동기화 시간이 오래 걸릴 수 있음

**Tensorflow 설치하기**

* 2019년 7월 기준 Colab에 설치된 Tensorflow는 1.14.0입니다
  + Tensorflow 2.0을 설치하고 싶으면 아래처럼 입력하시면 됩니다
* !pip install tensorflow==2.0.0-beta1
  + 그 후, 런타임 - 런타임 다시 시작을 누르셔서 런타임을 다시 실행시키면 Tensorflow 2.0.0-beta1 설치된 것을 알 수 있습니다
* import tensorflow as tf
* print(tf.\_\_version\_\_)

**PyTorch 사용하기**

* 2019년 1월 26일부터 Colab에 기본적으로 PyTorch, torchvision, torchtext가 내장되었습니다
* 출처 : [PyTorch 트위터](https://mobile.twitter.com/PyTorch/status/1088882504700968960)

import torch

import torchvision

import torchtext

print(torch.\_\_version\_\_)

**KoNLPy 설치**

* 공식 문서엔 openjdk-7-jdk로 작성되어 있으나, 우분투 17.04에선 ppa를 추가해야 설치 가능
* ppa를 추가하지 않고 8 버전을 설치

!apt-get update

!apt-get install g++ openjdk-8-jdk

!pip3 install konlpy

* 예제 코드
* from konlpy.tag import Twitter
* twitter = Twitter()
* twitter.pos("질문이나 건의사항은 깃헙 이슈 트래커에 남겨주세요")

**Github 코드를 Colab에서 사용하기**

* nbviewer나 htmlpreview처럼 사용 가능
* <https://github.com/~~~> 부분을 <https://colab.research.google.com/github/~~~>로 교체하면 됨
* 예시
  + <https://github.com/zzsza/TIL/blob/master/python/tensorflow-1.ipynb> 를
  + <https://colab.research.google.com/github/zzsza/TIL/blob/master/python/tensorflow-1.ipynb>로 변경

**BigQuery 사용하기**

* google.colab의 auth를 통해 클라우드 권한을 얻은 후, 다양한 라이브러리 사용
* google.cloud.bigquery 사용시
* from google.cloud import bigquery
* from google.colab import auth
* project\_id = '[your project ID]'
* auth.authenticate\_user()
* client = bigquery.Client(project=project\_id)
* for dataset in client.list\_datasets():
* print(dataset.dataset\_id)
* pandas\_gbq 사용시 (개인적으로 이 방법 추천)
* import pandas as pd
* from google.colab import auth
* auth.authenticate\_user()
* query = "SELECT \* FROM <YOUR TABLE> LIMIT 10"
* df = pd.read\_gbq(query=query, project\_id=<your project id>, dialect=‘standard’))

**Matplotlib에서 한글 사용하기**

* 폰트 설치
* !apt-get install fonts-nanum\*
* !apt-get install fontconfig
* !fc-cache -fv
* !cp /usr/share/fonts/truetype/nanum/Nanum\* /usr/local/lib/python3.6/dist-packages/matplotlib/mpl-data/fonts/ttf/
* !rm -rf /content/.cache/matplotlib/\*
* 그래프 그리기
* import matplotlib.pyplot as plt
* import matplotlib as mpl
* import matplotlib.font\_manager as fm
* import numpy as np
* %matplotlib inline
* %config InlineBackend.figure\_format = 'retina'
* mpl.rcParams['axes.unicode\_minus'] = False
* # 그래프에서 마이너스 폰트 깨질 경우 대비
* path = '/usr/share/fonts/truetype/nanum/NanumGothicBold.ttf'
* font\_name = fm.FontProperties(fname=path, size=18).get\_name()
* plt.rc('font', family=font\_name)
* fm.\_rebuild() # 이걸 해줘야 plt.rc가 작동
* plt.plot(np.random.randn(4, 8), np.random.randn(4,8), 'bo--')
* plt.title('타이틀')
* plt.xlabel('X 라벨')
* plt.ylabel('Y 라벨')
* plt.show()

**Tensorboard 사용하기**

* Tensorboard 사용 준비
  + LOG\_DIR를 ‘drive/tb\_logs’로 설정하면 내 구글드라이브 root에 tb\_logs 폴더가 생성
  + 저는 ‘drive/data/tb\_logs’로 지정
* LOG\_DIR = 'drive/data/tb\_logs'
* !wget <https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip>
* !unzip ngrok-stable-linux-amd64.zip
* import os
* if not os.path.exists(LOG\_DIR):
* os.makedirs(LOG\_DIR)
* get\_ipython().system\_raw(
* 'tensorboard --logdir {} --host 0.0.0.0 --port 6006 &'
* .format(LOG\_DIR))
* get\_ipython().system\_raw('./ngrok http 6006 &')
* !curl -s <http://localhost:4040/api/tunnels> | python3 -c \
* "import sys, json; print(json.load(sys.stdin)['tunnels'][0]['public\_url'])"
* 아래 예제 코드 실행 후, Epoch 지나고 위 코드에서 나온 URL 클릭하면 TensorBoard로 이동
* from \_\_future\_\_ import print\_function
* import keras
* from keras.datasets import mnist
* from keras.models import Sequential
* from keras.layers import Dense, Dropout, Flatten
* from keras.layers import Conv2D, MaxPooling2D
* from keras import backend as K
* from keras.callbacks import TensorBoard
* batch\_size = 128
* num\_classes = 10
* epochs = 12
* ### input image dimensions
* img\_rows, img\_cols = 28, 28
* ### the data, shuffled and split between train and test sets
* (x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()
* if K.image\_data\_format() == 'channels\_first':
* x\_train = x\_train.reshape(x\_train.shape[0], 1, img\_rows, img\_cols)
* x\_test = x\_test.reshape(x\_test.shape[0], 1, img\_rows, img\_cols)
* input\_shape = (1, img\_rows, img\_cols)
* else:
* x\_train = x\_train.reshape(x\_train.shape[0], img\_rows, img\_cols, 1)
* x\_test = x\_test.reshape(x\_test.shape[0], img\_rows, img\_cols, 1)
* input\_shape = (img\_rows, img\_cols, 1)
* x\_train = x\_train.astype('float32')
* x\_test = x\_test.astype('float32')
* x\_train /= 255
* x\_test /= 255
* print('x\_train shape:', x\_train.shape)
* print(x\_train.shape[0], 'train samples')
* print(x\_test.shape[0], 'test samples')
* ### convert class vectors to binary class matrices
* y\_train = keras.utils.to\_categorical(y\_train, num\_classes)
* y\_test = keras.utils.to\_categorical(y\_test, num\_classes)
* model = Sequential()
* model.add(Conv2D(32, kernel\_size=(3, 3),
* activation='relu',
* input\_shape=input\_shape))
* model.add(Conv2D(64, (3, 3), activation='relu'))
* model.add(MaxPooling2D(pool\_size=(2, 2)))
* model.add(Dropout(0.25))
* model.add(Flatten())
* model.add(Dense(128, activation='relu'))
* model.add(Dropout(0.5))
* model.add(Dense(num\_classes, activation='softmax'))
* model.compile(loss=keras.losses.categorical\_crossentropy,
* optimizer=keras.optimizers.Adadelta(),
* metrics=['accuracy'])

* tbCallBack = TensorBoard(log\_dir=LOG\_DIR,
* histogram\_freq=1,
* write\_graph=True,
* write\_grads=True,
* batch\_size=batch\_size,
* write\_images=True)
* model.fit(x\_train, y\_train,
* batch\_size=batch\_size,
* epochs=epochs,
* verbose=1,
* validation\_data=(x\_test, y\_test),
* callbacks=[tbCallBack])
* score = model.evaluate(x\_test, y\_test, verbose=0)
* print('Test loss:', score[0])
* print('Test accuracy:', score[1])

**JDK 설치하기**

* Python을 사용해 JDK 설치하기
* 아래 코드 입력
* import os
* def install\_java():
* !apt-get install -y openjdk-8-jdk-headless -qq > /dev/null #install openjdk
* os.environ["JAVA\_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64" #set environment variable
* !java -version #check java version
* install\_java()

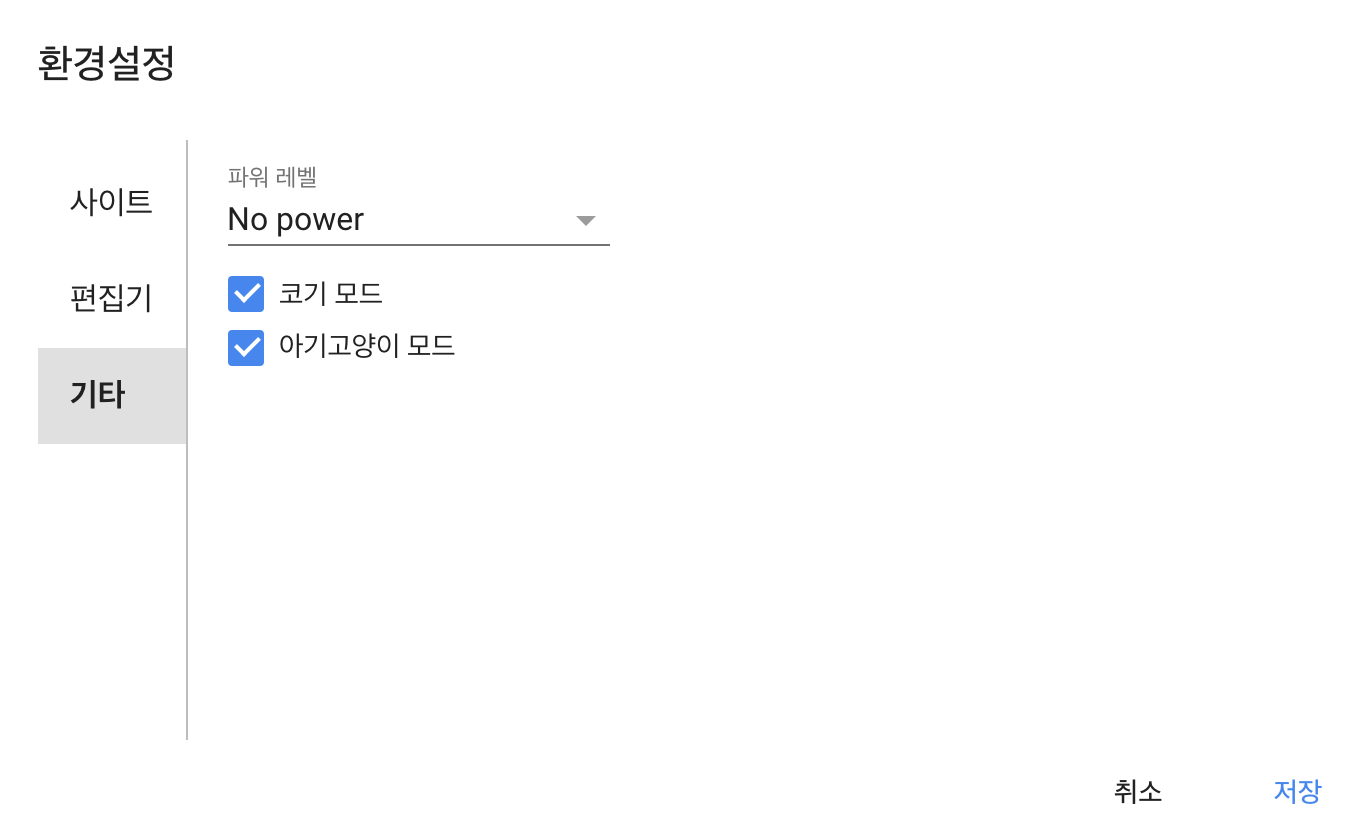
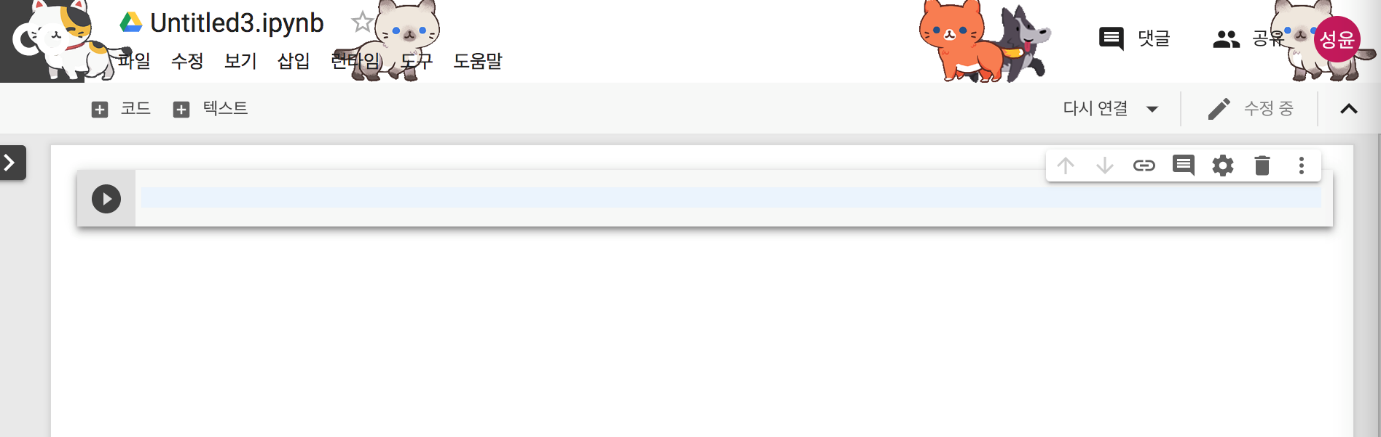
**Google Storage에서 파일 읽기**

* gcsfs를 활용한 방법
  + storage에서 땡겨오기 때문에 colab에 파일을 옮기지 않아도 됨. 그러나 약간 시간 소요됨. 빠른 방법은 아래 gsutil 참고
* !pip3 install gcsfs dask
* import gcsfs
* from google.colab import auth
* auth.authenticate\_user()
* fs = gcsfs.GCSFileSystem(project='project\_name')
* with fs.open('주소') as f:
* df = pd.read\_csv(f)
* 위 방법보다 빠른 방법(gsutil 사용)
  + 단, 세션이 초기화되면 매번 gsutil로 파일을 땡겨와야 합니다
* from google.colab import auth
* auth.authenticate\_user()
* import pandas as pd
* file\_path = 'data.csv'
* !gsutil cp gs://<your\_bucket>/{file\_path} {file\_path}
* df = pd.read\_csv(query\_result\_path)

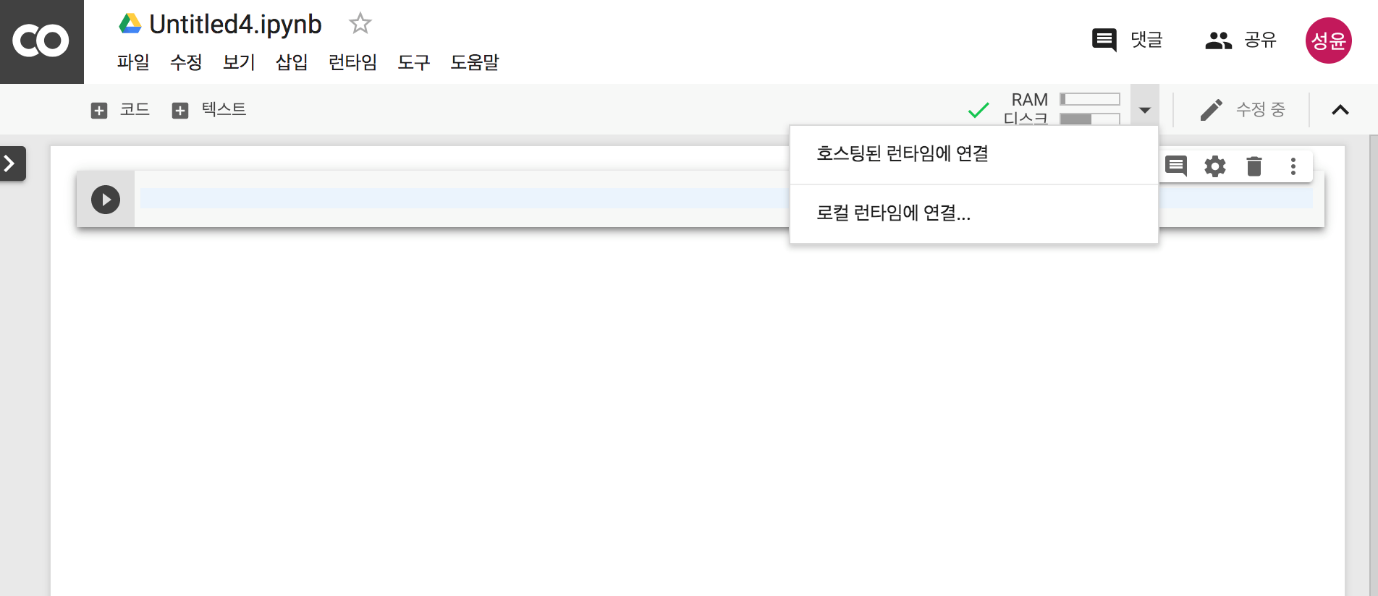
**Kaggle 연동하기**

* 1) Kaggle beta API Json Key 다운
  + Kaggle - My Account - Dataset 옆에 있는 …을 클릭한 후, Account로 이동
  + 하단에 API 부분에 Create New API Token을 클릭하면 Json Key가 다운로드 됨
  + 이 Json 키를 매번 Colab에서 올려서 할 수도 있지만, 더 편하게 사용하고 싶어서 Google Storage에 Json 파일을 올리고, 거기서 키를 복사해오는 방법으로 진행합니다
* 2) Google Storage에 Json Key 저장
  + [Google Storage](https://console.cloud.google.com/storage/browser)로 이동한 후, Storage 버킷 선택 (버킷이 없다면 생성!)
  + Colab에서 아래 명령어 입력
* from google.colab import auth
* auth.authenticate\_user()
* !mkdir -p ~/.kaggle
* !mv ./kaggle.json ~/.kaggle/
* !chmod 600 ~/.kaggle/kaggle.json
* 3) Kaggle 설치 및 데이터 다운로드
* !pip install kaggle
  + Competition 확인하기
* !kaggle competitions list
  + 데이터 다운로드는 각 대회 Data 부분에 나와있는 API 명령어를 입력하면 됨
* !kaggle competitions download -c elo-merchant-category-recommendation
  + 아래 명령어로 압축을 풀면 data 폴더에 파일들이 저장됨
* !unzip '\*.zip' -d ./data
  + 복사하기 편하게 한번에 작성
* from google.colab import auth
* auth.authenticate\_user()
* !pip install kaggle
* !mkdir -p ~/.kaggle
* !mv ./kaggle.json ~/.kaggle/
* !chmod 600 ~/.kaggle/kaggle.json
* !kaggle competitions download -c elo-merchant-category-recommendation
* !unzip '\*.zip' -d ./data

**코기 모드**

* Colab을 조금 더 귀엽게 만드는 방법
* 도구 - 환경설정 - 기타에 코기 모드 / 아기고양이 모드 체크
  + 
* 조금 기다리면 귀여운 친구들이 뿅!
  + 
* 기타에 파워 레벨을 Many power로 설정하시면 코드를 칠 때마다 이팩트가 생깁니다 :)

**런타임 연결**

* RAM, 디스크가 나오는 부분의 우측 삼각형을 클릭하면 아래와 같은 메뉴가 보입니다
  + 
* 호스트된 런타임에 연결(Connect to Hosted Runtime)
  + Google Cloud의 새로운 머신 인스턴스에 연결
  + 라이브러리를 다시 설치해야할 수도 있음
  + Colab의 연결이 끊길 경우 해당 버튼을 클릭해서 다시 연결할 수 있습니다(같은 인스턴스인지는 아직 확인해보지 못했습니다)
* 로컬 런타임에 연결(Connect to Local Runtime)
  + 내 PC(로컬)을 사용

**Reference**

* [Google Colab Free GPU Tutorial](https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d)
* [Google Colaboratory를 활용하여 Keras 개발환경 구축](http://yamalab.tistory.com/80)
* [유재흥님 블로그](https://brunch.co.kr/@jayden-factory)
* [정현석님 블로그](http://jusths.tistory.com/)

이 글이 도움이 되셨다면 추천 클릭을 부탁드립니다 :)