

2. 케라스 시작하기

케라스 이야기

- ▶ 케라스는 파이썬으로 구현된 쉽고 간결한 딥러닝 라이브러리입니다.
- ▶ 딥러닝 비전문가라도 각자 분야에서 손쉽게 딥러닝 모델을 개발하고 활용할 수 있도록 케라스는 직관적인 API를 제공하고 있습니다.
- ▶ 내부적으로는 텐서플로우(TensorFlow), 티아노(Theano), CNTK 등의 딥러닝 전용 엔진이 구동되지만 케라스 사용자는 복잡한 내부 엔진을 알 필요는 없습니다.
- ▶ 직관적인 API로 쉽게 다층퍼셉트론 모델, 컨볼루션 신경망 모델, 순환 신경망 모델 또는 이를 조합한 모델은 물론 다중 입력 또는 다중 출력 등 다양한 구성을 할 수 있습니다.

케라스 이야기

▶ 케라스의 주요 특징

- ▶ 케라스는 아래 4가지의 주요 특징을 가지고 있습니다.
 - ▶ 모듈화 (Modularity)
 - 케라스에서 제공하는 모듈은 독립적이고 설정 가능하며, 가능한 최소한의 제약 사항으로 서로 연결될 수 있습니다.
 - 모델은 시퀀스 또는 그래프로 이러한 모듈들을 구성한 것입니다.
 - 특히 신경망 층, 비용함수, 최적화기, 초기화기법, 활성화함수, 정규화기법은 모두 독립적인 모듈이며, 새로운 모델을 만들기 위해 이러한 모듈을 조합할 수 있습니다.
 - ▶ 최소주의 (Minimalism)
 - 각 모듈은 짧고 간결합니다.
 - 모든 코드는 한 번 훑어보는 것으로도 이해가능해야 합니다.
 - 단, 반복 속도와 혁신성에는 다소 떨어질 수가 있습니다.

케라스 이야기

▶ 케라스의 주요 특징

▶ 케라스는 아래 4가지의 주요 특징을 가지고 있습니다.

▶ 쉬운 확장성

- 새로운 클래스나 함수로 모듈을 아주 쉽게 추가할 수 있습니다.
- 따라서 고급 연구에 필요한 다양한 표현을 할 수 있습니다.

▶ 파이썬 기반

- Caffe 처럼 별도의 모델 설정 파일이 필요없으며, 파이썬 코드로 모델들이 정의됩니다.
- 이 멋진 케라스를 개발하고 유지보수하고 있는 사람은 구글 엔지니어인 프랑소와 쉐레(François Chollet)입니다.

케라스 이야기

▶ 케라스의 기본 개념

- ▶ 케라스의 가장 핵심적인 데이터 구조는 바로 모델입니다.
- ▶ 케라스에서 제공하는 시퀀스 모델로 원하는 레이어를 쉽게 순차적으로 쌓을 수 있습니다.
- ▶ 다중 출력이 필요하는 등 좀 더 복잡한 모델을 구성하려면 케라스 함수 API를 사용하면 됩니다.
- ▶ 케라스로 딥러닝 모델을 만들 때는 다음과 같은 순서로 작성합니다.
- ▶ 다른 딥러닝 라이브러리와 비슷한 순서이지만 훨씬 직관적이고 간결합니다.

케라스 이야기

▶ 케라스의 기본 개념

▶ 1. 데이터셋 생성하기

- ▶ 원본 데이터를 불러오거나 시뮬레이션을 통해 데이터를 생성합니다.
- ▶ 데이터로부터 훈련셋, 검증셋, 시험셋을 생성합니다.
- ▶ 이 때 딥러닝 모델의 학습 및 평가를 할 수 있도록 포맷 변환을 합니다.

▶ 2. 모델 구성하기

- ▶ 시퀀스 모델을 생성한 뒤 필요한 레이어를 추가하여 구성합니다.
- ▶ 좀 더 복잡한 모델이 필요할 때는 케라스 함수 API를 사용합니다.

▶ 3. 모델 학습과정 설정하기

- ▶ 학습하기 전에 학습에 대한 설정을 수행합니다.
- ▶ 손실 함수 및 최적화 방법을 정의합니다.
- ▶ 케라스에서는 `compile()` 함수를 사용합니다.

케라스 이야기

▶ 케라스의 기본 개념

▶ 4. 모델 학습시키기

- ▶ 훈련셋을 이용하여 구성한 모델로 학습시킵니다.
- ▶ 케라스에서는 `fit()` 함수를 사용합니다.

▶ 5. 학습과정 살펴보기

- ▶ 모델 학습 시 훈련셋, 검증셋의 손실 및 정확도를 측정합니다.
- ▶ 반복횟수에 따른 손실 및 정확도 추이를 보면서 학습 상황을 판단합니다.

▶ 6. 모델 평가하기

- ▶ 준비된 시험셋으로 학습한 모델을 평가합니다.
- ▶ 케라스에서는 `evaluate()` 함수를 사용합니다.

▶ 7. 모델 사용하기

- ▶ 임의의 입력으로 모델의 출력을 얻습니다.
- ▶ 케라스에서는 `predict()` 함수를 사용합니다.

구글 Colaboratory 사용법

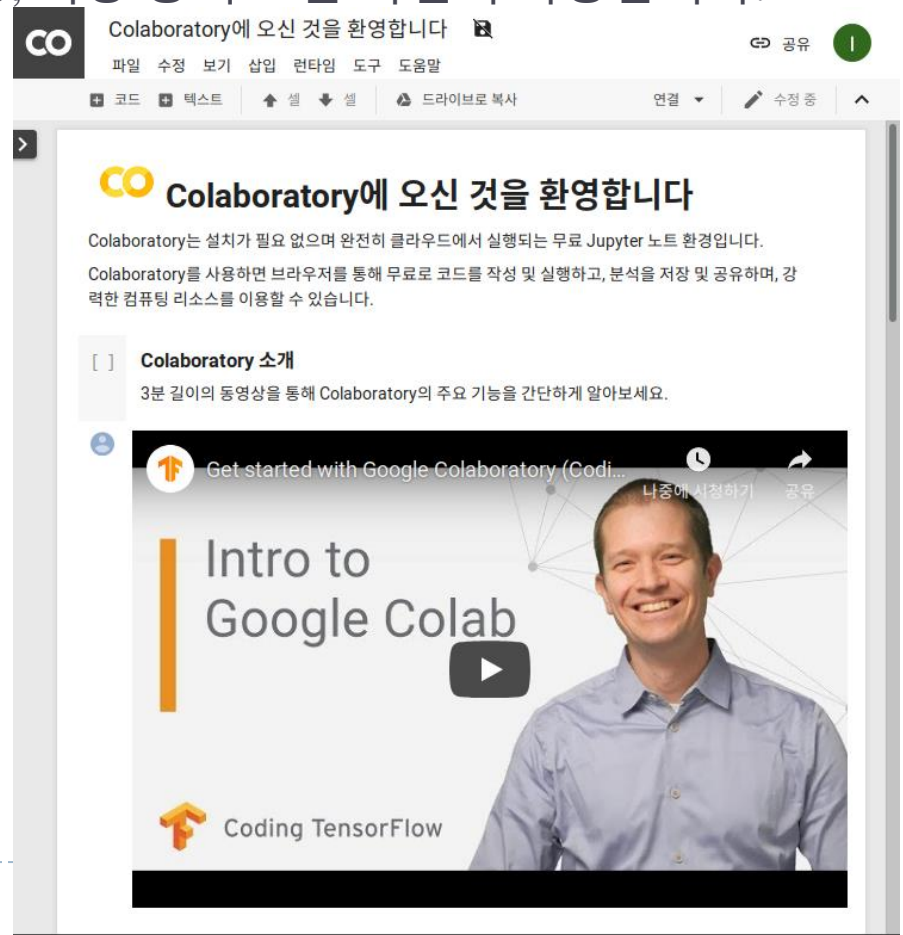
- ▶ Colaboratory 구글에서 제공하는 무료 Jupyter 노트북입니다.
- ▶ 브라우저에서 Python 코드를 실행할 수 있고 무료 GPU와 TPU 등의 컴퓨팅 자원을 함께 사용할 수 있습니다.
- ▶ 예제 코드를 Colaboratory 에서 실행해 보면서, Colaboratory 사용법에 대해서 설명드리도록 하겠습니다



구글 Colaboratory 사용법

▶ Colaboratory 접속

- ▶ 아래의 페이지에 접속하시면, 다음같은 화면이 나타납니다.
- ▶이 곳에서 코드를 작성하고 실행, 저장 등의 모든 작업이 가능합니다.
- ▶ <https://colab.research.google.com/>



구글 Colaboratory 사용법

- ▶ 먼저 예제 파일을 작성하기 위해서, [파일] 메뉴 버튼을 클릭하고 새 Python 노트 메뉴를 선택합니다.



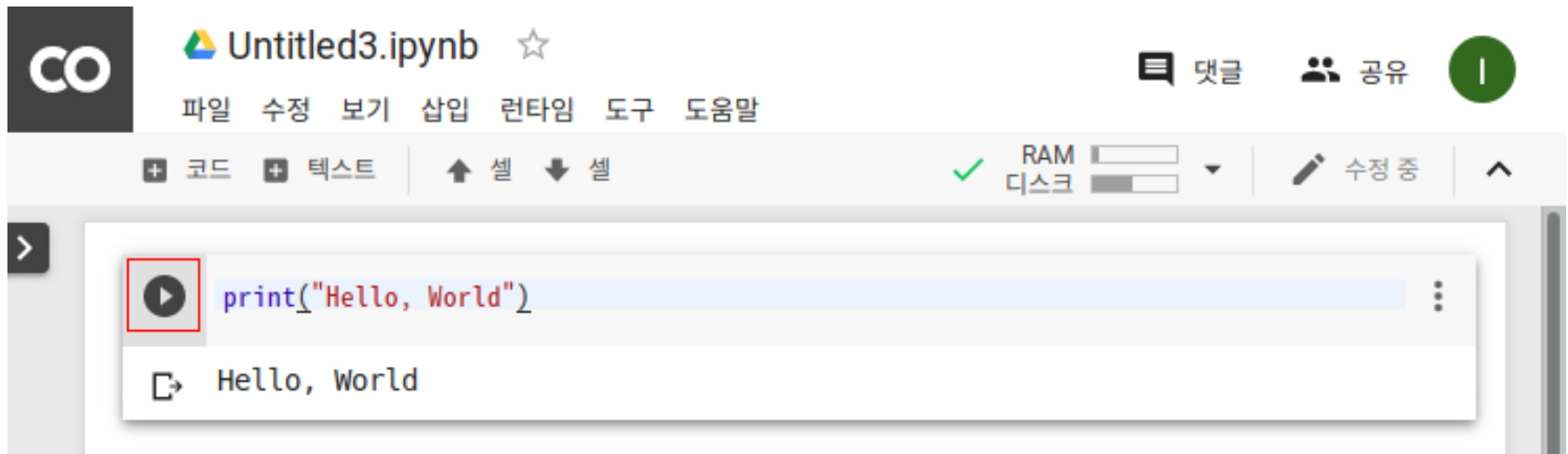
구글 Colaboratory 사용법

▶ Python 코드 실행

- ▶ 새로운 노트가 생성되는데, 아래와 같이 Python 코드를 작성합니다.

```
print("Hello, World")
```

- ▶ 그리고 좌측의 작은 삼각형 모양의 실행 버튼을 클릭하면, 아래와 같이 코드가 실행되게 됩니다.

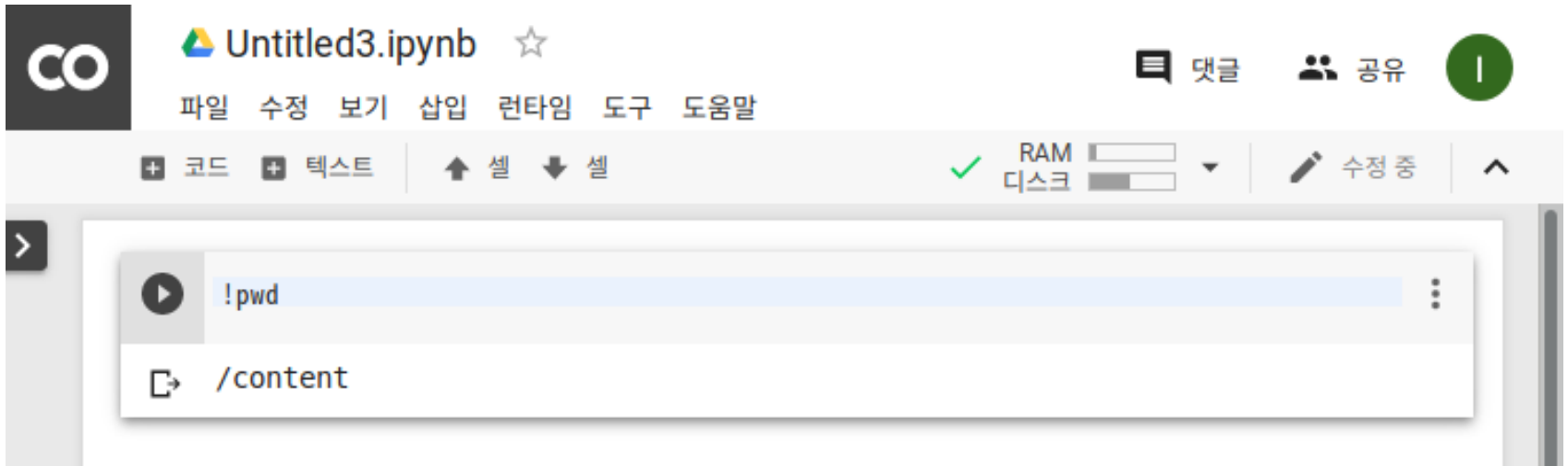


- ▶ 런타임 메뉴를 이용하여 각 셀 별로 코드를 나눠서 실행 할 수 있을 뿐만 아니라, 전체 코드를 한 번에 실행 할 수 도 있습니다.

구글 Colaboratory 사용법

▶ 셸 명령어 실행

- ▶ 편집기 화면에서는 기본적으로 Python 코드를 실행할 수 있지만, 셸 명령어 또한 실행할 수 있습니다. 셸 명령어를 실행하기 위해서는 명령어 앞에 '!' 기호를 입력하여 실행하면 됩니다.

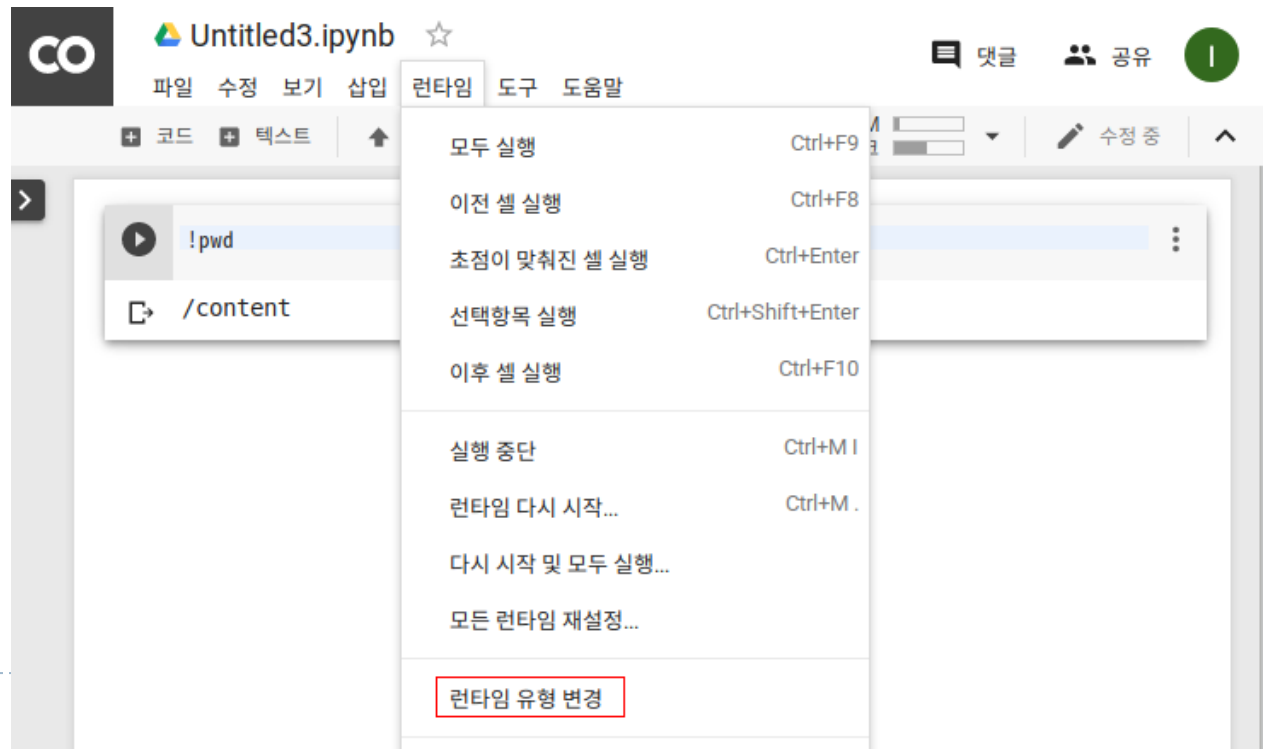


- ▶ 위의 내용은 현재의 디렉터리 위치를 확인하는 명령어인 pwd 명령어를 실행한 결과입니다.

구글 Colaboratory 사용법

▶ 무료 GPU 사용

- ▶ 구글 Colaboratory 에서는 무료 GPU 또한 사용이 가능합니다.
- ▶ GPU 또는 TPU를 사용하기 위해서는 런타임 유형을 변경해야 합니다.
- ▶ [런타임] 메뉴에서 [런타임 유형 변경] 메뉴를 클릭합니다.



구글 Colaboratory 사용법

▶ 무료 GPU 사용

- ▶ 그리고 나타나는 노트 설정 팝업에서 하드웨어 가속기를 GPU 으로 선택합니다.
- ▶ 무료로 TPU 또한 선택하여 사용이 가능합니다.

노트 설정

런타임 유형

Python 3

하드웨어 가속기

GPU

☐ 이 노트를 저장할 때 코드 셀 출력 생략

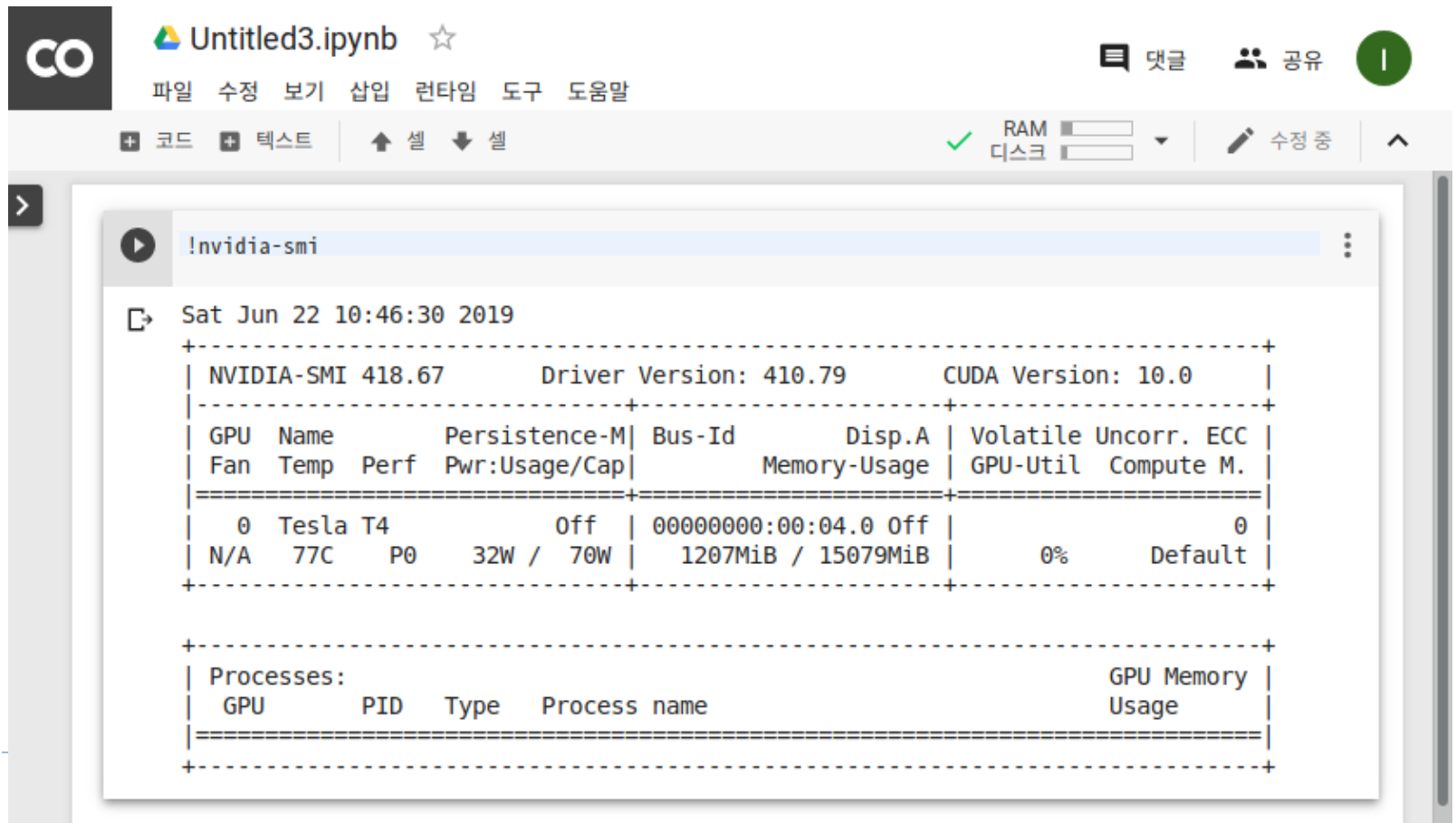
취소

저장

구글 Colaboratory 사용법

▶ 무료 GPU 사용

- ▶ 실제 GPU를 사용할 수 있는지 GPU 정보를 확인 할 수 있는 `nvidia-smi` 명령어를 실행해 보도록 하겠습니다.



CO Untitled3.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말

코드 텍스트 셀 셀

✓ RAM 디스크 수정 중

```
!nvidia-smi
```

Sat Jun 22 10:46:30 2019

NVIDIA-SMI 418.67		Driver Version: 410.79		CUDA Version: 10.0	
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile Uncorr. ECC
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util Compute M.
0	Tesla T4	Off	00000000:00:04.0	Off	0
N/A	77C	P0	32W / 70W	1207MiB / 15079MiB	0% Default

Processes:					GPU Memory
GPU	PID	Type	Process name		Usage

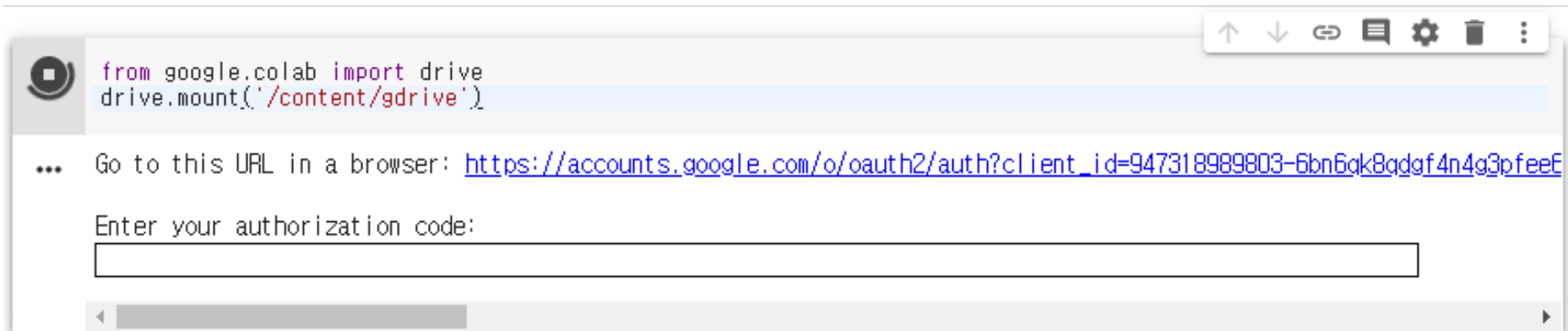
구글 Colaboratory과 구글 드라이브 연동

- ▶ 우선은 구글의 colab에 드라이브를 임포트해줍니다.
- ▶ 두번째로는 구글 드라이브를 마운트를 해줍니다.
- ▶ 여기 까지는 손 댈거 없이 입력해줍니다.
- ▶ 참고로 '/content/gdrive' 이 경로는 '구글 드라이브에 접속하겠다'입니다. (내 구글 드라이버가 아닌 서버접속)

```
from google.colab import drive  
drive.mount('/content/gdrive')
```


구글 Colaboratory과 구글 드라이브 연동

- ▶ 그러면 아래 와 같은 출력창이 나옵니다.
- ▶ 1번째 줄은 링크가능한 url 주소 (클릭시 내 구글드라이버 연동 코드 생성)
- ▶ 그 아래 코드 넣으라고 네모 칸이 있습니다.
- ▶ 위의 링크를 눌러 구글 동의 후 생성된 코드를 복사해서 넣으시면 잠시 뒤에 마운트 되었다고 나옵니다.
- ▶ 이러면 사실상 본인의 구글 웹 드라이브와 구글 Colab와 연동된 것입니다.



구글 Colaboratory과 구글 드라이브 연동

- ▶ 아래는 테스트 코드입니다.

```
with open('/content/gdrive/My Drive/foo.txt', 'w') as f:  
    f.write('Hello Google Drive!')
```

- ▶ 그럼 내 구글 드라이브를 보면 폴더 없이 바로 파일이 생성되어 있습니다.
- ▶ 정리가 필요합니다.
- ▶ 복잡하지 않습니다.
- ▶ 사용할 폴더 하나 만들어서 사용하면 됩니다.
- ▶ 위의 코드 경로구조를 보면
 - ▶ '/content/gdrive/My Drive/foo.txt'
- ▶ 이런 구조입니다.

구글 Colaboratory과 구글 드라이브 연동

- ▶ `/content/gdrive/My Drive/`
- ▶ 여기까지가 딱 내 구글 드라이브 표준 경로 입니다.
- ▶ 윈도우로 따지면 바탕화면이라고 할까요?
- ▶ 단순히 이 경로에 폴더 하나 추가해주면 됩니다.
- ▶ `/content/gdrive/My Drive/Data`
- ▶ 한가지 주의할 점은 미리 사용할 폴더는 구글 드라이브에서 미리 생성해 주세요.
- ▶ 이유는 구글의 colaboratory 작업시 폴더 저장후 대략 5초 - 20초 사이에서 생성됩니다.
- ▶ 그래서 폴더가 없다고 에러를 냅니다.
- ▶ 참고로 이렇게 하는 이유는 구글드라이브의 파일등을 코드상에서 추가 하거나 또는 불러오기 위함입니다.

케라스 사용하기

▶ 케라스의 기본 개념

- ▶ 손글씨 영상을 분류하는 모델을 케라스로 간단히 구현해봤습니다.
- ▶ 가로세로 픽셀이 28 x 28인 이미지를 입력받아 이를 784 벡터로 구성한 다음 이를 학습 및 평가하는 코드입니다.
- ▶ 이 간단한 코드로 93.4%의 정확도 결과를 얻었습니다.
- ▶ 각 함수의 설명 및 인자에 대한 설명은 여러 모델을 실습해보면서 하나씩 설명드리겠습니다.

케라스 사용하기

- ▶ 케라스의 기본 개념
 - ▶ 손글씨 영상을 분류하는 모델 구현

```
# 0. 사용할 패키지 불러오기
from keras.utils import np_utils
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Activation
```

```
# 1. 데이터셋 생성하기
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.reshape(60000, 784).astype('float32') / 255.0
x_test = x_test.reshape(10000, 784).astype('float32') / 255.0
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
```

케라스 사용하기

- ▶ 케라스의 기본 개념
 - ▶ 손글씨 영상을 분류하는 모델 구현

2. 모델 구성하기

```
model = Sequential()  
model.add(Dense(units=64, input_dim=28*28, activation='relu'))  
model.add(Dense(units=10, activation='softmax'))
```

3. 모델 학습과정 설정하기

```
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

4. 모델 학습시키기

```
hist = model.fit(x_train, y_train, epochs=5, batch_size=32)
```

케라스 사용하기

▶ 케라스의 기본 개념

▶ 손글씨 영상을 분류하는 모델 구현

5. 학습과정 살펴보기

```
print('### training loss and acc ###')  
print(hist.history['loss'])  
print(hist.history['acc'])
```

6. 모델 평가하기

```
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=32)  
print('### evaluation loss and _metrics ###')  
print(loss_and_metrics)
```

7. 모델 사용하기

```
xhat = x_test[0:1]  
yhat = model.predict(xhat)  
print('### yhat ###')  
print(yhat)
```

케라스 사용하기

▶ 케라스의 기본 개념

▶ 손글씨 영상을 분류하는 모델 실행결과

Epoch 1/5

60000/60000 [=====] - 2s - loss: 0.6861 -
acc: 0.8214

Epoch 2/5

60000/60000 [=====] - 2s - loss: 0.3472 -
acc: 0.9021

Epoch 3/5

60000/60000 [=====] - 2s - loss: 0.2972 -
acc: 0.9159

Epoch 4/5

60000/60000 [=====] - 2s - loss: 0.2671 -
acc: 0.9243

Epoch 5/5

60000/60000 [=====] - 2s - loss: 0.2444 -
acc: 0.9311

케라스 사용하기

▶ 케라스의 기본 개념

▶ 손글씨 영상을 분류하는 모델 실행결과

```
## training loss and acc ##
```

```
[0.68605235149065658,    0.34716726491451261,    0.29719433775146803,  
0.26708923313419025, 0.24437546383142472]
```

```
[0.82138333333333335,    0.90213333333333334,    0.91591666666666671,  
0.92433333333333334, 0.93113333333333337]
```

```
9728/10000 [=====>.] - ETA: 0s## evaluation
```

```
loss and _metrics ##
```

```
[0.229048886179924, 0.93520000000000003]
```

```
## yhat ##
```

```
[[ 8.87035931e-05  1.47768702e-07  9.51653055e-04  3.75617994e-03  
 2.82607380e-06  4.90140701e-05  2.10885389e-08  9.94055033e-01  
 6.05004097e-05  1.03587494e-03]]
```

Q&A

