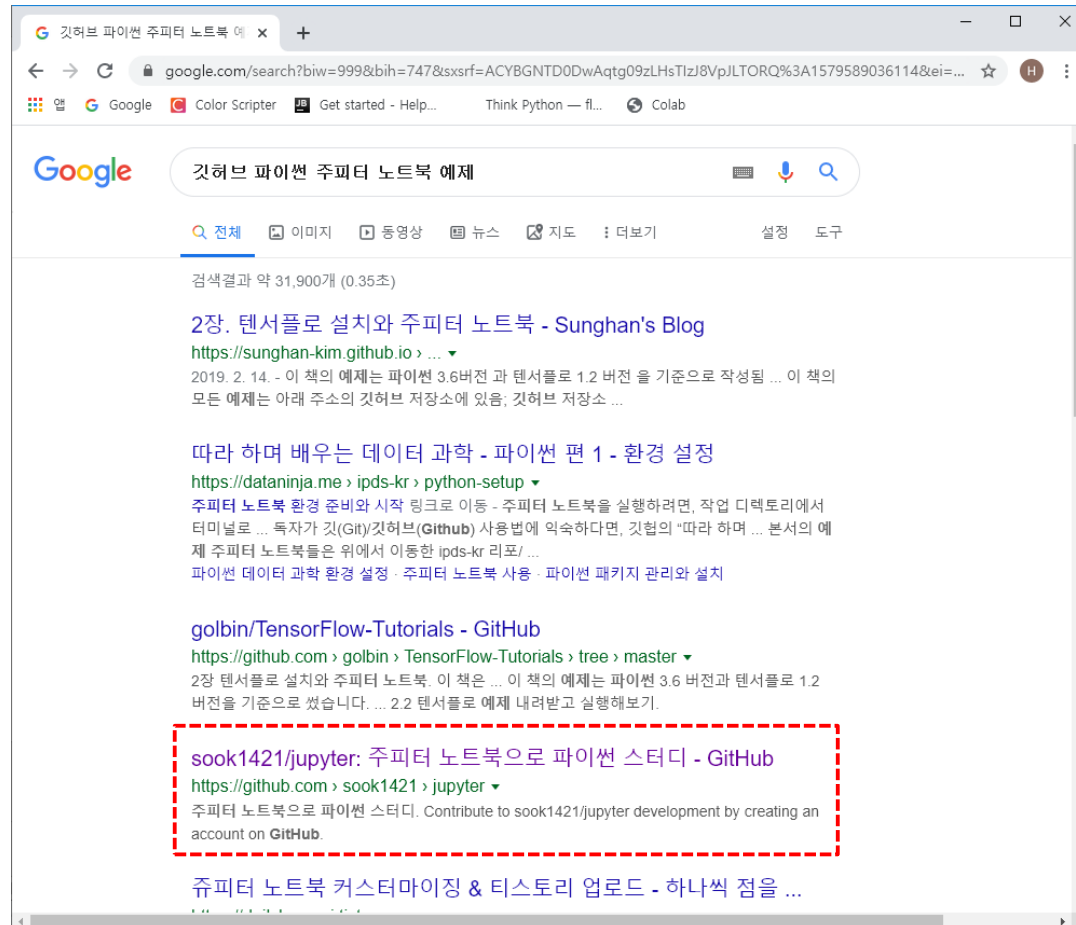


깃허브의 노트 파일을 Colab에서 활용

깃허브 사이트 검색

- <https://github.com/sook1421/jupyter>



노트 열기

- sook1421/jupyter

The screenshot shows the Google Colaboratory web interface. The 'File' menu is open on the left, displaying various options including 'Open from Drive', 'Open from Recent', 'New Python 3 Notebook', 'New Python 2 Notebook', 'Open Notebook...', 'Upload Notebook...', 'Rename...', 'Move to Trash', 'Save to Drive', 'Save to GitHub Gist', 'Save to GitHub', 'Save', 'Save and Commit', 'Update', 'Download .ipynb', 'Download .py', 'Drive Preview Update', and 'Refresh'. The 'Open' dialog is also open, showing a search for 'sook1421/jupyter'. The search results show a list of files, with '기본 문법 알아보기 (파이썬).ipynb' highlighted. The file is located in the 'sook1421/jupyter' repository on GitHub, specifically in the 'master' branch. The file is a Jupyter Notebook file (.ipynb) and is titled '기본 문법 알아보기 (파이썬).ipynb'.

파일 열기 후

모두 실행

The screenshot shows a Google Colab notebook titled '기본 문법 알아보기 (파이썬).ipynb'. The 'Run' (런타임) menu is open, displaying the '모두 실행' (Run All) option with the keyboard shortcut Ctrl+F9. Other visible options include '이전 셀 실행' (Run Previous Cell), '초점이 맞춰진 셀 실행' (Run Current Cell), '선택항목 실행' (Run Selection), '이후 셀 실행' (Run Next Cell), '실행 중단' (Interrupt), '런타임 다시 시작...' (Restart), '다시 시작 및 모두 실행...' (Restart and Run All), '런타임 초기화' (Reset), '런타임 유형 변경' (Change Runtime Type), '세션 관리' (Manage Session), and '런타임 로그 보기' (View Runtime Log).

The notebook content includes:

- 1 세미콜론**
프로그래밍 언어들은
파이썬은 세미콜론을
`[] print('Hello, world!')`
Hello, world!
- 여러 구분을 사용할 때 세미콜론으로 구분**
`[] print('Hello!'); print('JeongSook')`
Hello!
JeongSook
- 2 주석**
주석은 프로그램의 실행에 영향을 주지 않습니다.
한 줄 주석과 범위 주석 두 가지가 있는데,,

내 파일로 저장

• 저장

- '드라이브에 사본 저장...'이 먼저 필요
- 드라이브의 다음 폴더에 저장
 - Colab Notebooks
- 버튼 메뉴
 - 드라이버로 복사 사용 권장

The image illustrates the process of saving a Colab notebook to Google Drive. It shows the Colab interface with a 'Save to Drive' button highlighted in a red dashed box. Below it, the Google Drive interface is shown with the 'Colab Notebooks' folder highlighted in a red dashed box. On the right, a list of files in the 'Colab Notebooks' folder is displayed, with the file '기본 문법 알아보기 (파이썬).ipynb의 사본' highlighted in a red dashed box.

Name	Owner	Last modified
패션 MNIST 이해 시각화.ipynb	me	Jan 1, 2020 me
패션 MNIST 이해 시각화.ipynb	me	Jan 4, 2020 me
방법2.ipynb	me	Dec 8, 2019 me
기본 문법 알아보기 (파이썬).ipynb의 사본	me	4:07 PM me
Untitled2.ipynb	me	Jan 4, 2020 me

실습

- '케라스 창시자에게 배우는 딥러닝 깃허브' 검색
 - <https://github.com/gilbutITbook/006975>

첫 파일 열기

- 2.1-a-first-look-at-a-neural-network.ipynb

에

최근 사용

Google 드라이브

GitHub

업로드

GitHub URL을 입력하거나 조직 또는 사용자로 검색하세요.

☒ 비공개 저장소 포함

gilbutITbook/006975

🔍

저장소:

gilbutITbook/006975 ▼

브랜치:

master ▼

경로

2.1-a-first-look-at-a-neural-network.ipynb

3.4-classifying-movie-reviews.ipynb

3.5-classifying-newswires.ipynb

3.6-predicting-house-prices.ipynb

취소

자신 드라이브에 저장

2.1-a-first-look-at-a-neural-network.ipynb의 사본 ☆

파일 수정 보기 삽입 런타임 도구 도움말 오후 4:43에 마지막으로 저장됨

+ 코드 + 텍스트

```
import keras
keras.__version__
```

Using TensorFlow backend.
'2.2.2'

신경망과의 첫 만남

이 노트북은 [케라스 창시자에게 배우는 딥러닝](#) 책의 2장 1절의 코드 예제입니다. 책에는 더 많은 내용과 그림이 있습니다. 이 노트북에는 소스 코드에 관련된 설명만 포함합니다.

케라스 파이썬 라이브러리를 사용하여 손글씨 숫자 분류를 학습하는 구체적인 신경망 예제를 살펴보겠습니다. 케라스나 비슷한 라이브러리를 사용한 경험이 없다면 당장은 이 첫 번째 예제를 모두 이해하지 못할 것입니다. 아직 케라스를 설치하지 않았을지도 모릅니다. 괜찮습니다. 다음 장에서 이 예제를 하나하나 자세히 설명합니다. 코드가 좀 이상하거나 요술처럼 보더라도 너무 걱정하지 마세요. 일단 시작해 보겠습니다.

여기에서 돌리고 하는 문제는 흑백 손글씨 숫자 이미지(28x28 픽셀)를 10개의 범주(0에서 9까지)로 분류하는 것입니다. 머신 러닝 커뮤니티에서 고전으로 취급받는 데이터셋인 MNIST를 사용하겠습니다. 이 데이터셋은 머신 러닝의 역사만큼 오래되었고 많은 연구에 사용되었습니다. 이 데이터셋은 1980년대에 미국 국립표준기술연구소에서 수집한 6만 개의 훈련 이미지와 1만 개의 테스트 이미지로 구성되어 있습니다. MNIST 문제를 알고리즘이 제대로 작동하는지 확인하기 위한 딥러닝계의 'Hello World'라고 생각해도 됩니다. 머신 러닝 기술자가 되기까지 연구 논문이나 블로그 포스트 등에서 MNIST를 보고 또 보게 될 것입니다.

MNIST 데이터셋은 넘파이 배열 형태로 케라스에 이미 포함되어 있습니다:

```
[ ] from keras.datasets import mnist

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

train_images와 train_labels가 모델이 학습해야 할 훈련 세트를 구성합니다. 모델은 test_images와 test_labels를 사용하여 테스트합니다.

My Drive > Colab Notebooks

Name	Owner	Last modified	
2.1-a-first-look-at-a-neural-network.ipynb의 사본	me	4:43 PM	
기본 문법 알아보기 (파이썬).ipynb의 사본	me	4:07 PM	

노트 파일의 깃허브 URL을
Colab으로 열기

노트 파일을 읽는 방법

- 단일 .ipynb 파일을 clone 하는 방법
 - <https://github.com/~ ~ ~> 에서 ~ ~ ~ 부분을
 - <https://colab.research.google.com/github/~ ~ ~> 로 교체

깃허브 노트 바로 코랩에서 열기

- 내가 알고 있는 노트 파일 url
 - <https://github.com/zzsza/TIL/blob/master/python/tensorflow-1.ipynb>
 - 노트저장소
 - 깃허브 서버 정보 빼고 다음 정보 확인
 - </zzsza/TIL/blob/master/python/tensorflow-1.ipynb>
- 코랩 url 이후 /github 를 붙이고 계속 붙이기
 - <https://colab.research.google.com>
 - /github 추가
 - <https://colab.research.google.com/github>
- 다음 주소로 바로 접근
 - <https://Colab주소/github/노트저장소>
 - <https://colab.research.google.com/github/zzsza/TIL/blob/master/python/tensorflow-1.ipynb>

코랩 접속 화면

The screenshot shows the Google Colaboratory web interface. The browser address bar displays the URL: `colab.research.google.com/github/zzsza/TIL/blob/master/python/tensorflow-1.ipynb`. The notebook title is `tensorflow-1.ipynb`. The code editor contains the following Python code:

```
import tensorflow as tf

[ ] hello = tf.constant("Hi")

[ ] print(hello)

Tensor("Const:0", shape=(), dtype=string)

[ ] # Tensor : 자료형

[ ] a = tf.constant(10)
    b = tf.constant(2)
    c = a + b

[ ] print(c)

Tensor("add:0", shape=(), dtype=int32)

[ ] # 텐서플로우는 그래프 생성 후, 그래프 실행하는 과정을 거쳐야 함. 지연 실행 방법을 사용

[ ] sess = tf.Session()
    print(sess.run(c))

12

[ ] print(sess.run(hello))

b'Hi'
```

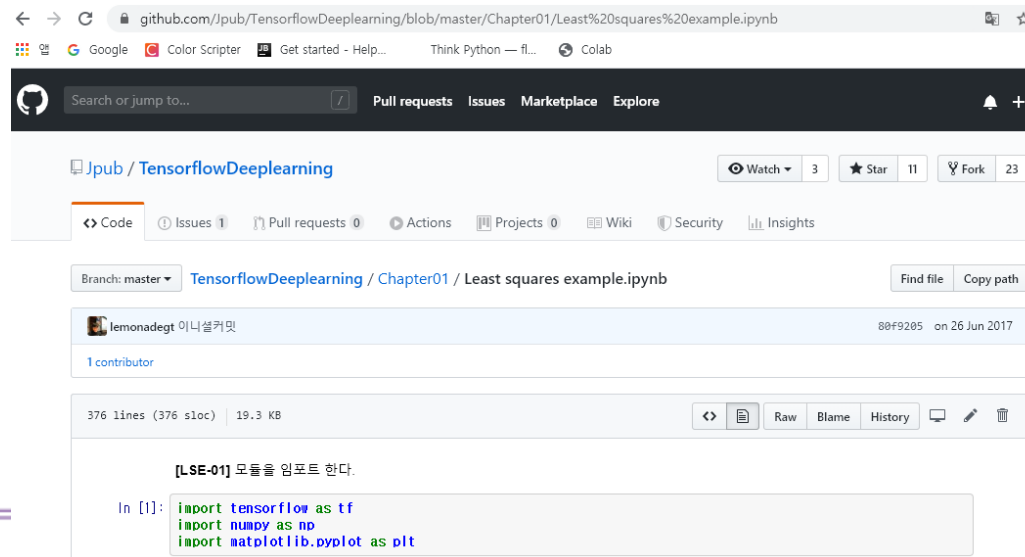
실습

• 텐서플로로 시작하는 딥러닝 깃허브



• 깃허브 페이지

- <https://github.com/Jpub/TensorflowDeeplearning/blob/master/Chapter01/Least%20squares%20example.ipynb>



접속 url

- 코랩 url

- <https://colab.research.google.com>

- 노트 url

- [/Jpub/TensorflowDeeplearning/blob/master/Chapter01/Least%20squares%20example.ipynb](https://colab.research.google.com/Jpub/TensorflowDeeplearning/blob/master/Chapter01/Least%20squares%20example.ipynb)

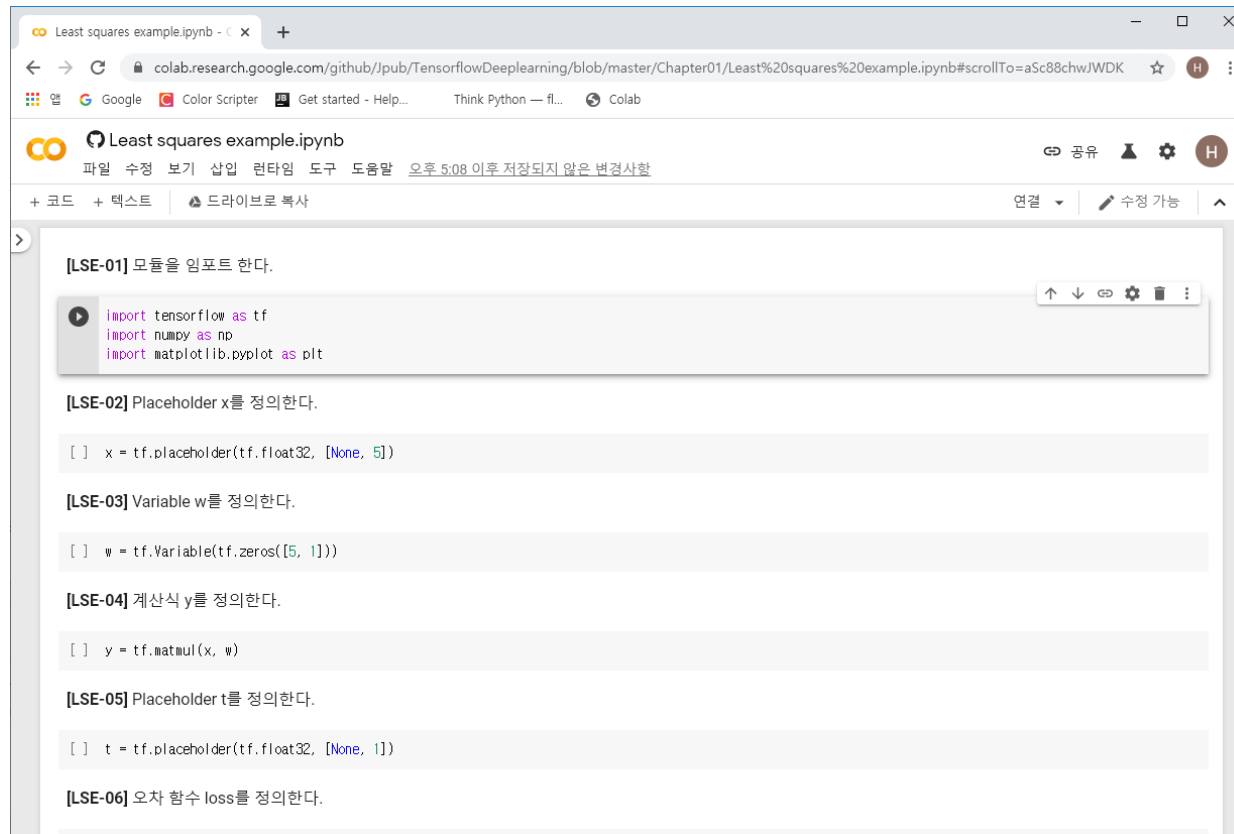
- 결과

- [/github 추가](#)
- <https://colab.research.google.com/github/Jpub/TensorflowDeeplearning/blob/master/Chapter01/Least%20squares%20example.ipynb#scrollTo=2ULb6gM5JWDO>

열기 성공

- 주소

- <https://colab.research.google.com/github/Jpub/TensorflowDeeplearning/blob/master/Chapter01/Least%20squares%20example.ipynb#scrollTo=2ULb6gM5JWDO>



```
[LSE-01] 모듈을 임포트 한다.  
import tensorflow as tf  
import numpy as np  
import matplotlib.pyplot as plt  
  
[LSE-02] Placeholder x를 정의한다.  
[ ] x = tf.placeholder(tf.float32, [None, 5])  
  
[LSE-03] Variable w를 정의한다.  
[ ] w = tf.Variable(tf.zeros([5, 1]))  
  
[LSE-04] 계산식 y를 정의한다.  
[ ] y = tf.matmul(x, w)  
  
[LSE-05] Placeholder t를 정의한다.  
[ ] t = tf.placeholder(tf.float32, [None, 1])  
  
[LSE-06] 오차 함수 loss를 정의한다.
```

구글 Colab 노트
Github에 저장

Welcome에서 Pandas 소개로 이동

- 다양한 자습 리소스 지원

머신러닝 단기집중과정

다음은 Google 온라인 머신러닝 과정에서 가져온 일부 노트입니다. [전체 과정 웹사이트](#)에서 자세한 내용을 확인하세요.

- [Pandas 소개](#)
- [TensorFlow 개념](#)
- [TensorFlow 첫걸음](#)
- [신경망 소개](#)
- [희소 데이터 및 임베딩 소개](#)

가속 하드웨어 사용하기

- [GPU를 사용한 TensorFlow](#)
- [TPU를 사용한 TensorFlow](#)

노트 파일에서 메뉴 선택

- 파일 하부
 - Github에 사본 저장




깃허브의 저장소로 저장

- 깃허브 저장소 입력
 - 인증(로그인) 필요

GitHub으로 복사

저장소: 
hskang7/test ▼

브랜치: 
master ▼

파일 경로

intro_to_pandas.ipynb

변경사항 설명 메시지

Colaboratory를 통해 생성됨

☒ Colaboratory 링크 추가

취소

확인

자신의 저장소 확인

- hskang7/test

hskang7 / test

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

Branch: master test / intro_to_pandas.ipynb Find file Copy path

hskang7 Colaboratory를 통해 생성됨 f57e526 5 minutes ago

1 contributor

656 lines (656 sloc) 19.6 KB

Open in Colab

Copyright 2017 Google LLC.

```
In [0]: # Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

Pandas 간단 소개

학습 목표:

구글 Colab 노트
깃허브 Gist에 저장

Welcome에서 Tensorflow 개념으로 이동

- 다양한 자습 리소스 지원

머신러닝 단기집중과정

다음은 Google 온라인 머신러닝 과정에서 가져온 일부 노트입니다. [전체 과정 웹사이트](#)에서 자세한 내용을 확인하세요.

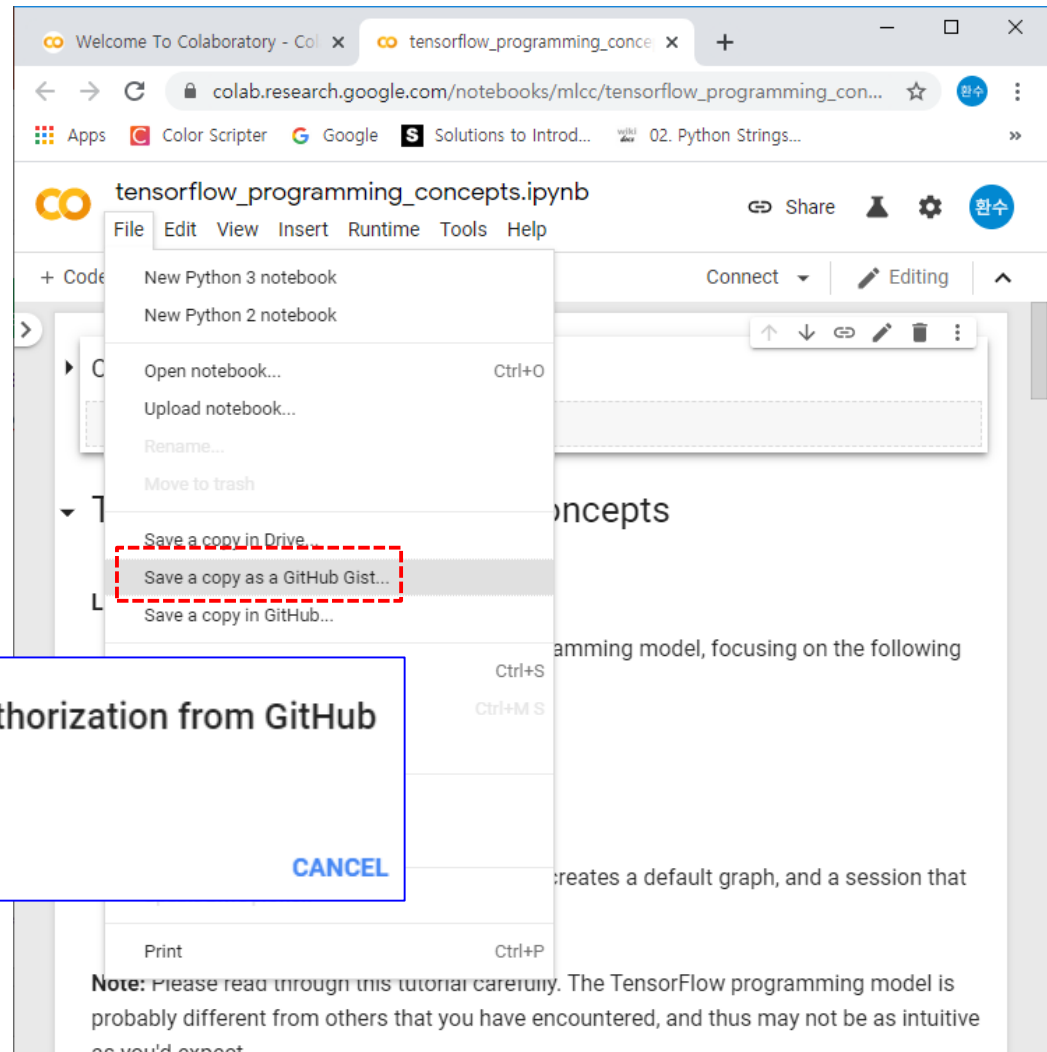
- [Pandas 소개](#)
- [TensorFlow 개념](#)
- [TensorFlow 첫걸음](#)
- [신경망 소개](#)
- [희소 데이터 및 임베딩 소개](#)

가속 하드웨어 사용하기

- [GPU를 사용한 TensorFlow](#)
- [TPU를 사용한 TensorFlow](#)

노트 파일에서 메뉴 선택

- 파일 하부
 - Github Gist로 사본 저장
- 깃허브 인증 필요



Colaboratory is waiting for authorization from GitHub

CANCEL


Note: Please read through this tutorial carefully. The TensorFlow programming model is probably different from others that you have encountered, and thus may not be as intuitive as you'd expect.


깃허브 인증

- 로그인

Sign in to GitHub · GitHub - Google Chrome

github.com/login?client_id=5036cf6d81e65aaa6340&...





Sign in to **GitHub**
to continue to **Colaboratory**

Username or email address

Password [Forgot password?](#)

Sign in

New to GitHub? [Create an account.](#)

자신의 깃허브 Gist에서 확인

- Gist 주소: gist.github.com/사용자id
 – gist.github.com/hskang7

비밀이라 공개는
안됨

hskang7's gists

gist.github.com/hskang7

Apps Color Scripter Google Solutions to Introd... 02. Python Strings... DeepLearningZero... TigerCow.Door :: 텐...

GitHub Gist Search... All gists Back to GitHub

All gists 7 Type: All Sort: Recently created

hskang7 / tensorflow_programming_concepts.ipynb Secret 1 file 0 forks 0 comments 0 stars

Created 6 minutes ago

tensorflow_programming_concepts.ipynb

Open in Colab

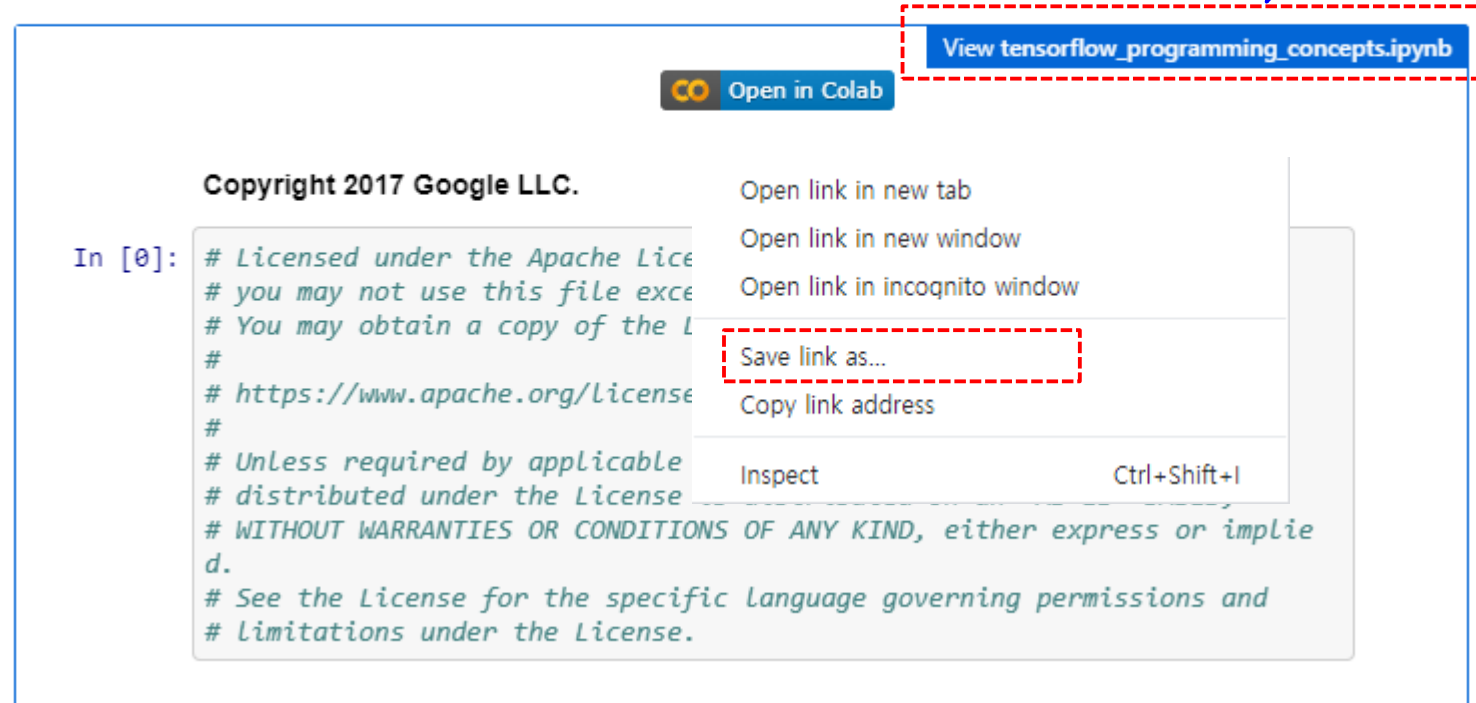
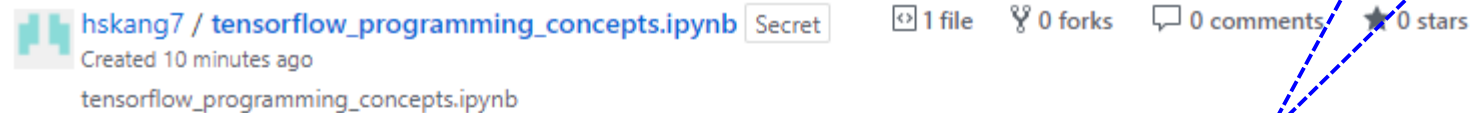
Copyright 2017 Google LLC.

```
In [0]: # Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

Colab에서
소스 열기

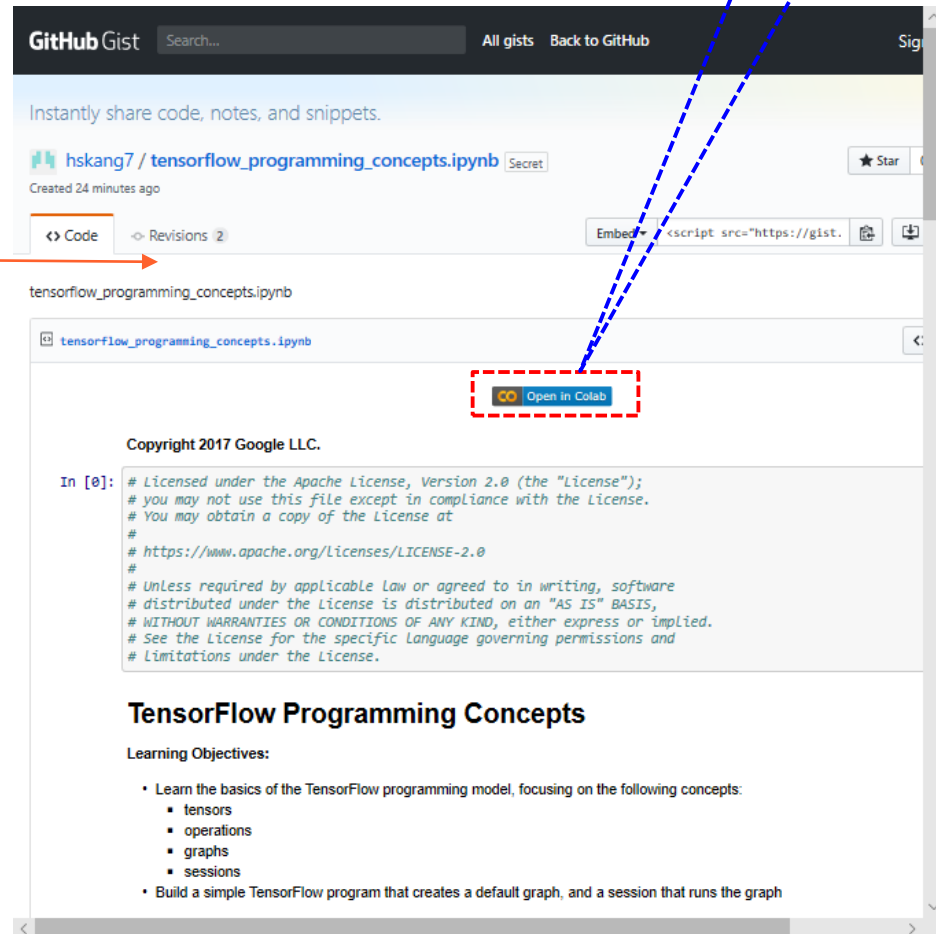
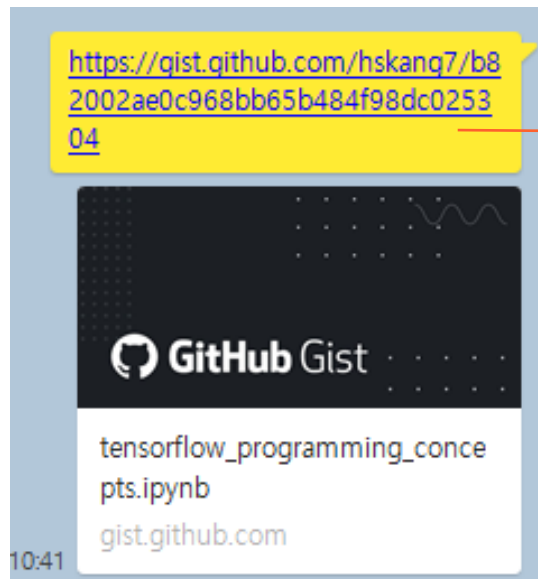
Gist 활용 방법

- 소스에서 오른쪽 팝업 메뉴에서 '링크 주소 복사' 선택



카톡 등에서 복사하면

- 바로 소스로 이동할 수 있는 링크 제공



Github Gist 개념

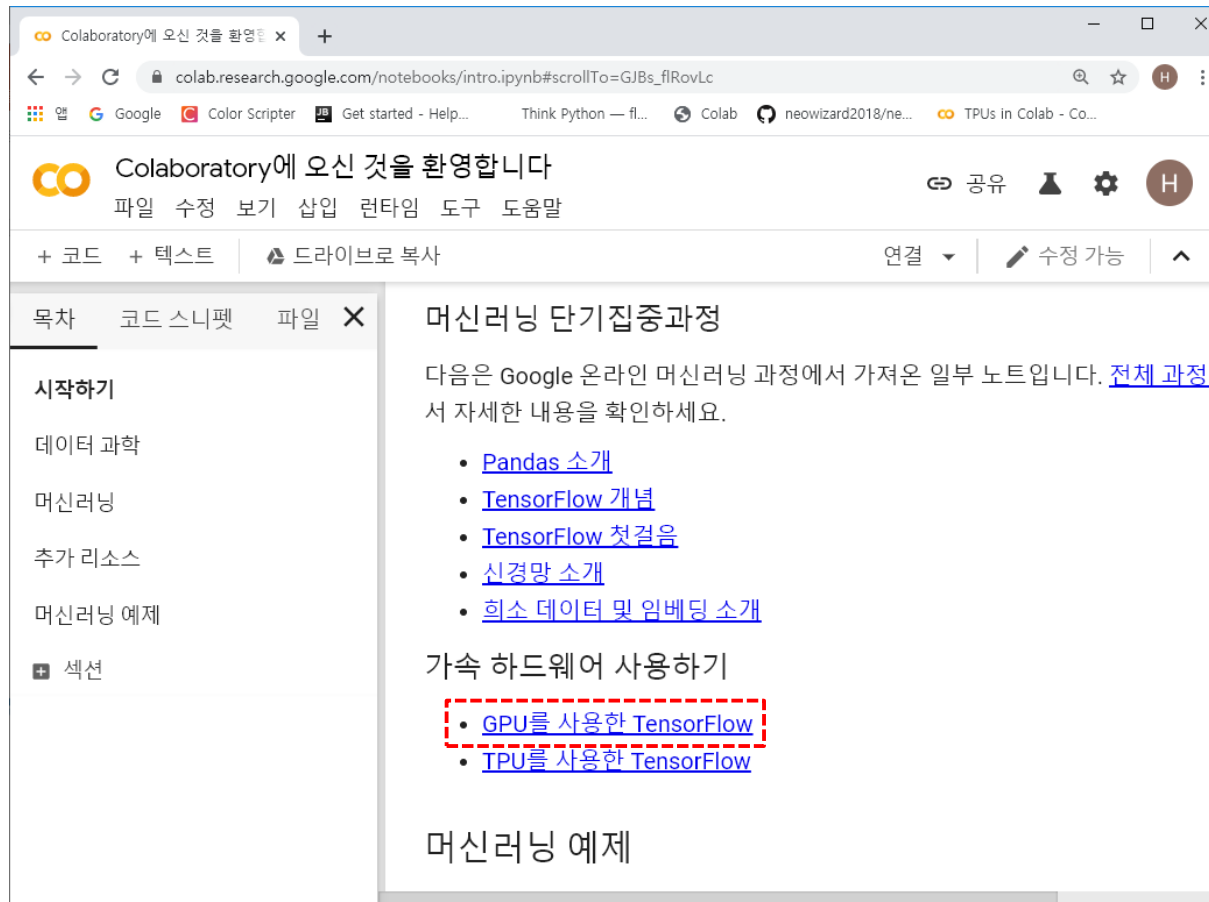
- 깃허브에서 제공하는 링크 공유 서비스
 - 블로그 등에 부분 소스 코드(snippets)로 보기 좋게 붙여 놓는 방법
- Secret과 public
 - Secret
 - 검색 엔진에는 노출이 안되지만 당신이 URL을 알려준 사람에게는 보여주는 수준

구글 Colab GPU 사용

Welcome의 GPU 링크로

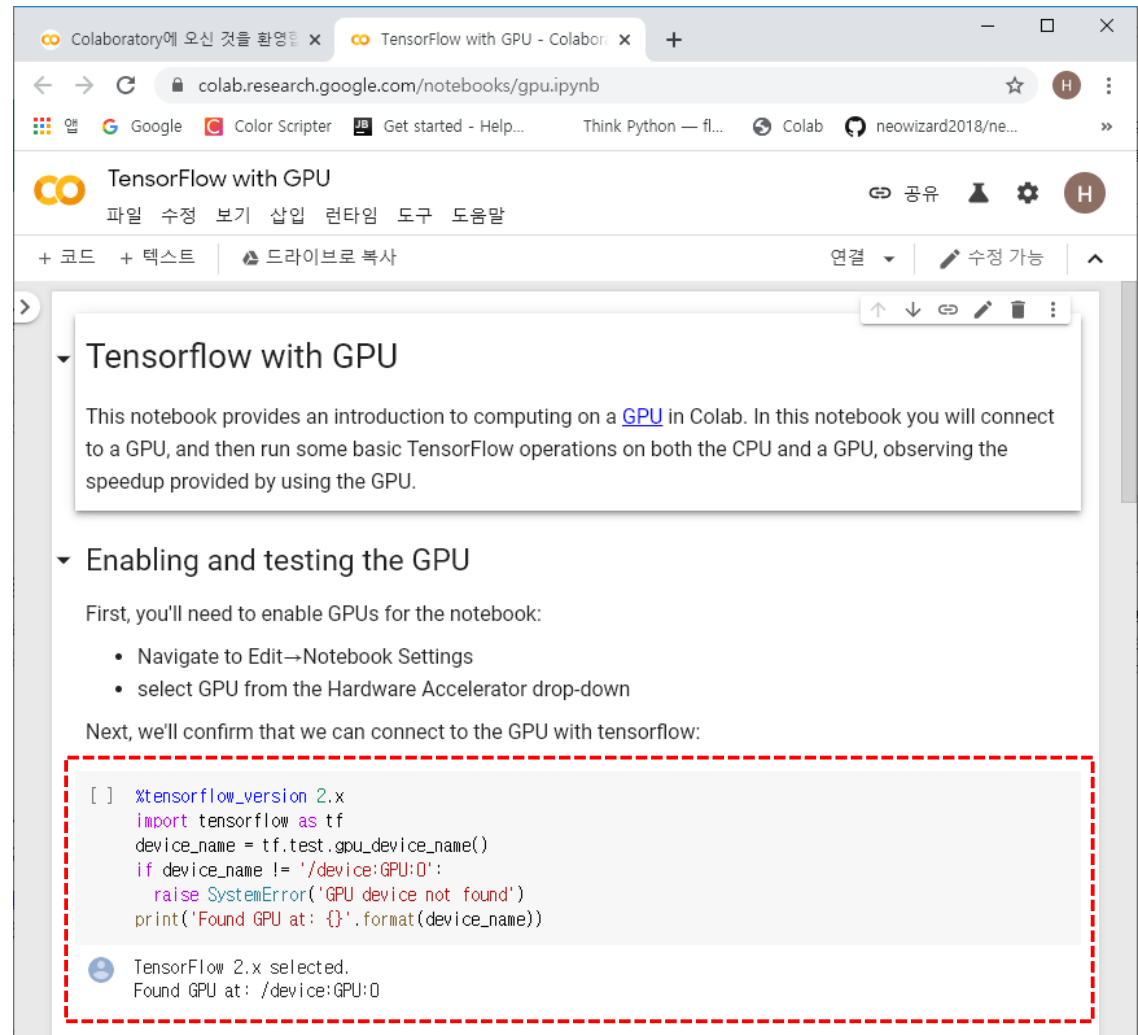
• GPU를 사용한 TensorFlow

- <https://colab.research.google.com/notebooks/gpu.ipynb>



첫 코드 셀 실행


- 내용 읽으면서
- 첫 코드 실행 오류
 - GPU 설정이 안되어 있어



코드 내용

• 텐서플로 코드

```
%tensorflow_version 2.x
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```



```
%tensorflow_version 2.x
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

TensorFlow 2.x selected.

```
SystemError                                Traceback (most recent call last)
<ipython-input-1-d1d5f2f639c4> in <module>()
      3 device_name = tf.test.gpu_device_name()
      4 if device_name != '/device:GPU:0':
----> 5     raise SystemError('GPU device not found')
      6 print('Found GPU at: {}'.format(device_name))

SystemError: GPU device not found
```

SEARCH STACK OVERFLOW

GPU 설정

• 메뉴

- 런타임 / 런타임 유형 변경

노트 설정

런타임 유형

Python 3

하드웨어 가속기

GPU

☐ 이 노트를 저장할 때 코드 셀 출력 생략

취소

저장

런타임 도구 도움말 오후 1:13 이후 저장

모두 실행 Ctrl+F9

이전 셀 실행 Ctrl+F8

초점이 맞춰진 셀 실행 Ctrl+Enter

선택항목 실행 Ctrl+Shift+Enter

이후 셀 실행 Ctrl+F10

실행 중단 Ctrl+M |

런타임 다시 시작... Ctrl+M .

다시 시작 및 모두 실행...

런타임 초기화

런타임 유형 변경

세션 관리

런타임 로그 보기

• 다시 실행

- GPU는 최대 12시간 실행을 지원

```
[1] %tensorflow_version 2.x
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

TensorFlow 2.x selected.
Found GPU at: /device:GPU:0

CPU와 GPU 속도 비교

• CPU 대비 GPU에서 TensorFlow 속도를 관찰

- 이 예는 무작위 이미지에 대한 전형적인 컨볼루션 신경 네트워크 계층을 구성하고 실행 속도를 비교하기 위해 CPU나 GPU에 결과적인 작업을 수동으로 배치

▼ Observe TensorFlow speedup on GPU relative to CPU

This example constructs a typical convolutional neural network layer over a random image and manually places the resulting ops on either the CPU or the GPU to compare execution speed.

```

x tensorflow_version 2.x
import tensorflow as tf
import timeit

device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    print(
        '\n\nThis error most likely means that this notebook is not '
        'configured to use a GPU. Change this in Notebook Settings via the '
        'command palette (cmd/ctrl-shift-P) or the Edit menu.\n\n')
    raise SystemError('GPU device not found')

def cpu():
    with tf.device('/cpu:0'):
        random_image_cpu = tf.random.normal([100, 100, 100, 3])
        net_cpu = tf.keras.layers.Conv2D(32, 7)(random_image_cpu)
        return tf.math.reduce_sum(net_cpu)

def gpu():
    with tf.device('/device:GPU:0'):
        random_image_gpu = tf.random.normal([100, 100, 100, 3])
        net_gpu = tf.keras.layers.Conv2D(32, 7)(random_image_gpu)
        return tf.math.reduce_sum(net_gpu)

# We run each op once to warm up: see: https://stackoverflow.com/a/45067900
cpu()
gpu()

# Run the op several times.
print('Time (s) to convolve 32x7x7x3 filter over random 100x100x100x3 images '
      '(batch x height x width x channel). Sum of ten runs.')
print('CPU (s):')
cpu_time = timeit.timeit('cpu()', number=10, setup="from __main__ import cpu")
print(cpu_time)
print('GPU (s):')
gpu_time = timeit.timeit('gpu()', number=10, setup="from __main__ import gpu")
print(gpu_time)
print('GPU speedup over CPU: {}x'.format(int(cpu_time/gpu_time)))

```

```

Time (s) to convolve 32x7x7x3 filter over random 100x100x100x3 images (batch x height x width x channel). Sum of ten runs.
CPU (s):
3.862475891000031
GPU (s):
0.10837535100017703
GPU speedup over CPU: 35x

```

코드 내용

```
%tensorflow_version 2.x
import tensorflow as tf
import timeit

device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    print(
        '\n\nThis error most likely means that this notebook is not '
        'configured to use a GPU. Change this in Notebook Settings via the '
        'command palette (cmd/ctrl-shift-P) or the Edit menu.\n\n')
    raise SystemError('GPU device not found')

def cpu():
    with tf.device('/cpu:0'):
        random_image_cpu = tf.random.normal((100, 100, 100, 3))
        net_cpu = tf.keras.layers.Conv2D(32, 7)(random_image_cpu)
        return tf.math.reduce_sum(net_cpu)

def gpu():
    with tf.device('/device:GPU:0'):
        random_image_gpu = tf.random.normal((100, 100, 100, 3))
        net_gpu = tf.keras.layers.Conv2D(32, 7)(random_image_gpu)
        return tf.math.reduce_sum(net_gpu)

# We run each op once to warm up; see: https://stackoverflow.com/a/45067900
cpu()
gpu()

# Run the op several times.
print('Time (s) to convolve 32x7x7x3 filter over random 100x100x100x3 images '
      '(batch x height x width x channel). Sum of ten runs.')
print('CPU (s):')
cpu_time = timeit.timeit('cpu()', number=10, setup="from __main__ import cpu")
print(cpu_time)
print('GPU (s):')
gpu_time = timeit.timeit('gpu()', number=10, setup="from __main__ import gpu")
print(gpu_time)
print('GPU speedup over CPU: {}x'.format(int(cpu_time/gpu_time)))
```



Time (s) to convolve 32x7x7x3 filter over random 100x100x100x3 images (batch x height x width x channel). Sum of ten runs.
 CPU (s):
 2.9499288609999894
 GPU (s):
 0.10855397499994979
 GPU speedup over CPU: 27x

PC에 다운로드해 점검

- py 다운로드
- 내 PC에서 해 보니
 - 약 10배 차이
 - 물론 GPU가 있고 환경 설정이 되어야 함

```
Run: tensorflow_with_gpu1 x
[CC.512] Internal: invoking ptxas not supported on windows
Relying on driver to perform ptx compilation. This message will be only logged once.
Time (s) to convolve 32x7x7x3 filter over random 100x100x100x3 images (batch x height x width x channel). Sum of ten runs.
CPU (s):
0.9806504
GPU (s):
0.08204789999999917
GPU speedup over CPU: 11x
```

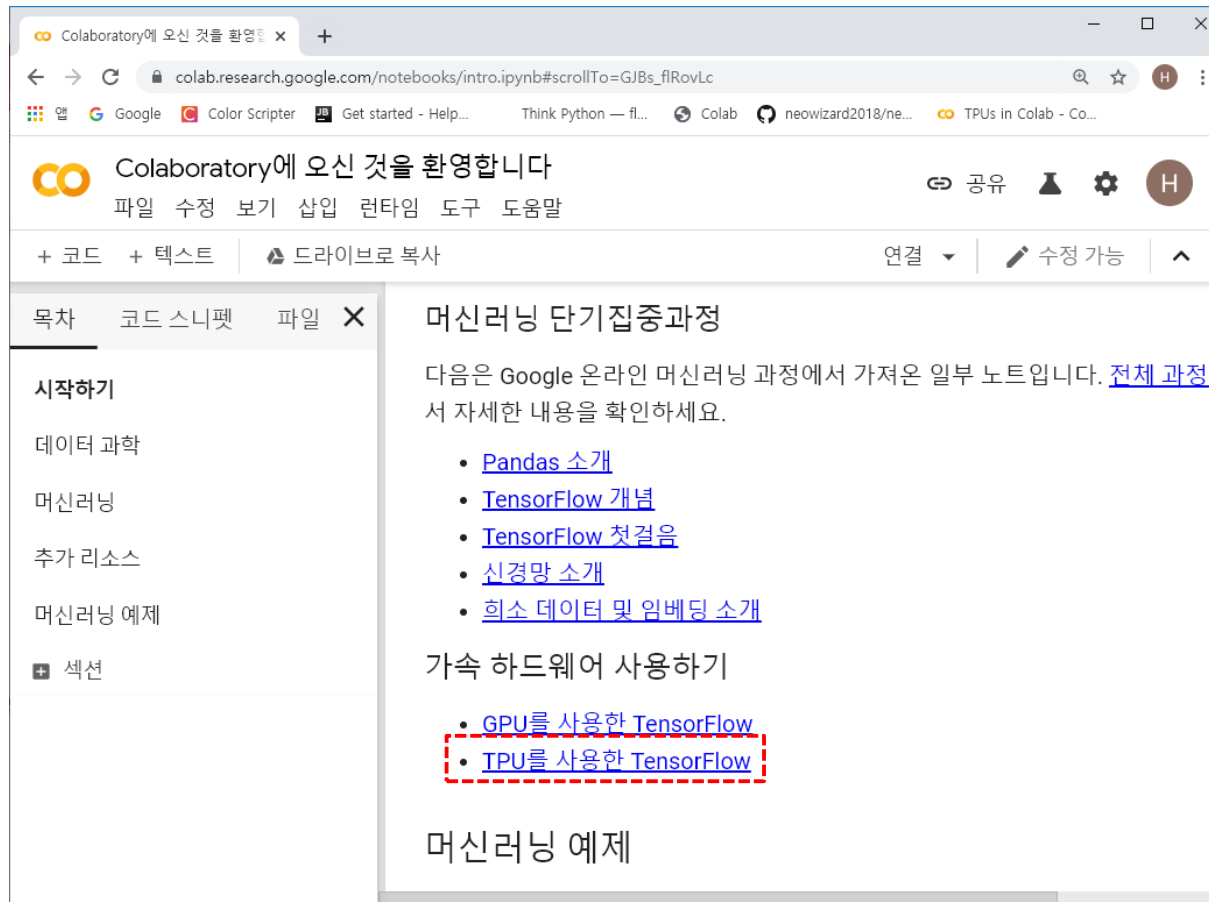
파일	수정	보기	삽입	런타임	도구	도움말
새 Python 3 노트						
새 Python 2 노트						
노트 열기...						Ctrl+O
노트 업로드...						
이름 바꾸기...						
휴지통으로 이동						
드라이브에 사본 저장...						
GitHub Gist로 사본 저장...						
GitHub에 사본 저장...						
저장						Ctrl+S
버전 저장 및 고정						Ctrl+M S
업데이트 기록						
.ipynb 다운로드						
.py 다운로드						
드라이브 미리보기 업데이트						
인쇄						Ctrl+P

구글 Colab TPU 사용

Welcome의 TPU 링크로

- TPU를 사용한 TensorFlow

- <https://colab.research.google.com/notebooks/tpu.ipynb>



바로 페이지에서 TPU 설정

• 메뉴

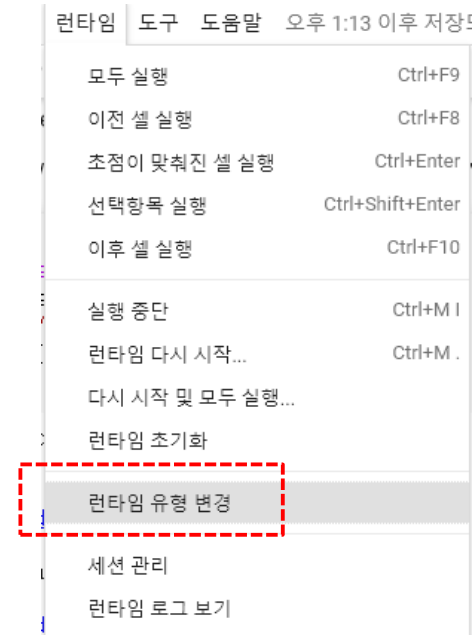
- 런타임 / 런타임 유형 변경

노트 설정



취소

저장



• 셀 실행

실행 결과

• TPU가 여러 개

```
[1] import os
import pprint
%tensorflow_version 1.x
import tensorflow as tf

if 'COLAB_TPU_ADDR' not in os.environ:
    print('ERROR: Not connected to a TPU runtime; please see the first cell in this notebook for instructions!')
else:
    tpu_address = 'grpc://' + os.environ['COLAB_TPU_ADDR']
    print('TPU address is', tpu_address)

    with tf.Session(tpu_address) as session:
        devices = session.list_devices()

    print('TPU devices:')
    pprint.pprint(devices)
```



TPU address is grpc://10.101.106.210:8470

TPU devices:

```
[_DeviceAttributes(/job:tpu_worker/replica:0/task:0/device:CPU:0, CPU, -1, 1364157890708099491),
 _DeviceAttributes(/job:tpu_worker/replica:0/task:0/device:XLA_CPU:0, XLA_CPU, 17179869184, 5185316091243000612),
 _DeviceAttributes(/job:tpu_worker/replica:0/task:0/device:TPU:0, TPU, 17179869184, 16540682973063688061),
 _DeviceAttributes(/job:tpu_worker/replica:0/task:0/device:TPU:1, TPU, 17179869184, 13158371155394370162),
 _DeviceAttributes(/job:tpu_worker/replica:0/task:0/device:TPU:2, TPU, 17179869184, 738041881959218271),
 _DeviceAttributes(/job:tpu_worker/replica:0/task:0/device:TPU:3, TPU, 17179869184, 7350922898889702529),
 _DeviceAttributes(/job:tpu_worker/replica:0/task:0/device:TPU:4, TPU, 17179869184, 4585351899304236492),
 _DeviceAttributes(/job:tpu_worker/replica:0/task:0/device:TPU:5, TPU, 17179869184, 15728817972348570001),
 _DeviceAttributes(/job:tpu_worker/replica:0/task:0/device:TPU:6, TPU, 17179869184, 11834559681570049782),
 _DeviceAttributes(/job:tpu_worker/replica:0/task:0/device:TPU:7, TPU, 17179869184, 11550534067670297373),
 _DeviceAttributes(/job:tpu_worker/replica:0/task:0/device:TPU_SYSTEM:0, TPU_SYSTEM, 8589934592, 2910277370962018695)]
```


두 번째 셀 실행

```
import numpy as np

def add_op(x, y):
    return x + y

# Silence deprecation warnings. TPUs do not yet work with TF 2.0.
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)

x = tf.placeholder(tf.float32, [10,])
y = tf.placeholder(tf.float32, [10,])
tpu_ops = tf.contrib.tpu.rewrite(add_op, [x, y])

session = tf.Session(tpu_address)
try:
    print('Initializing...')
    session.run(tf.contrib.tpu.initialize_system())
    print('Running ops')
    print(session.run(tpu_ops, {x: np.arange(10), y: np.arange(10)}))
finally:
    # For now, TPU sessions must be shutdown separately from
    # closing the session.
    session.run(tf.contrib.tpu.shutdown_system())
    session.close()
```

```
Initializing...
Running ops
[array([ 0.,  2.,  4.,  6.,  8., 10., 12., 14., 16., 18.], dtype=float32)]
```

세 번째 셀 실행: TPU Flops

- benchmark (FLOPS: floating point operations per second)

```

N = 4096
COUNT = 100
import time

def flops():
    x = tf.random_uniform([N, N])
    y = tf.random_uniform([N, N])
    def _matmul(x, y):
        return tf.tensordot(x, y, axes=[[1], [0]]), y

    return tf.reduce_sum(
        tf.contrib.tpu.repeat(COUNT, _matmul, [x, y])
    )

tpu_ops = tf.contrib.tpu.batch_parallel(flops, [], num_shards=8)

session = tf.Session(tpu_address)
try:
    print('Warming up...')
    session.run(tf.contrib.tpu.initialize_system())
    session.run(tpu_ops)
    print('Profiling')
    start = time.time()
    session.run(tpu_ops)
    end = time.time()
    elapsed = end - start
    print(elapsed, 'TFlops: {:.2f}'.format(1e-12 * 8 * COUNT * 2*N*N / elapsed))
finally:
    session.run(tf.contrib.tpu.shutdown_system())
    session.close()

```

Warming up...
Profiling
0.6747391223907471 TFlops: 162.95

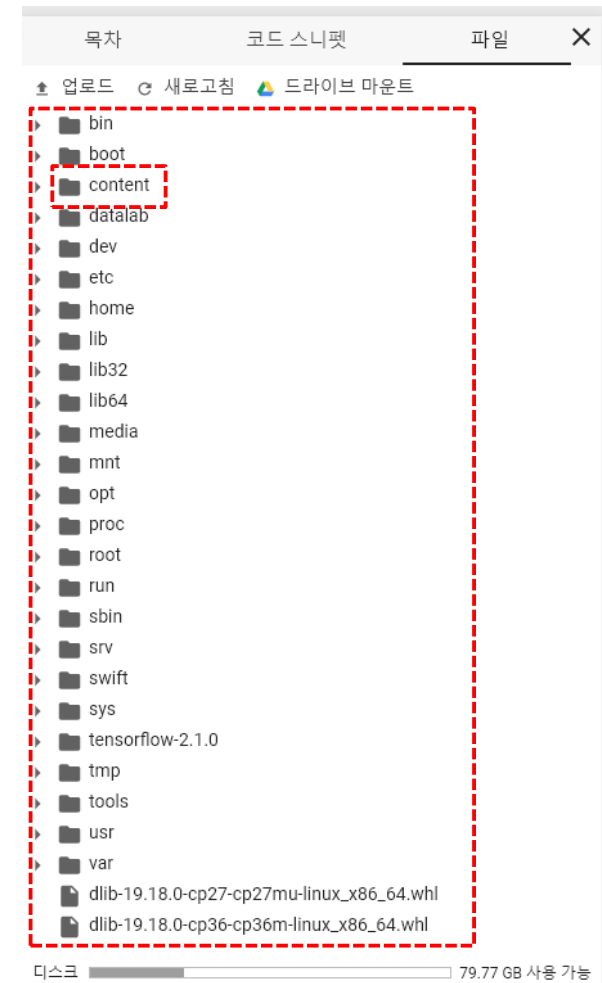
Colab 서버의 폴더에
구글 드라이브를
연결(마운트) 하기

서버 폴더 확인

- 새로운 파일에서 시작
- '파일' 선택
 - 현재 폴더
 - content



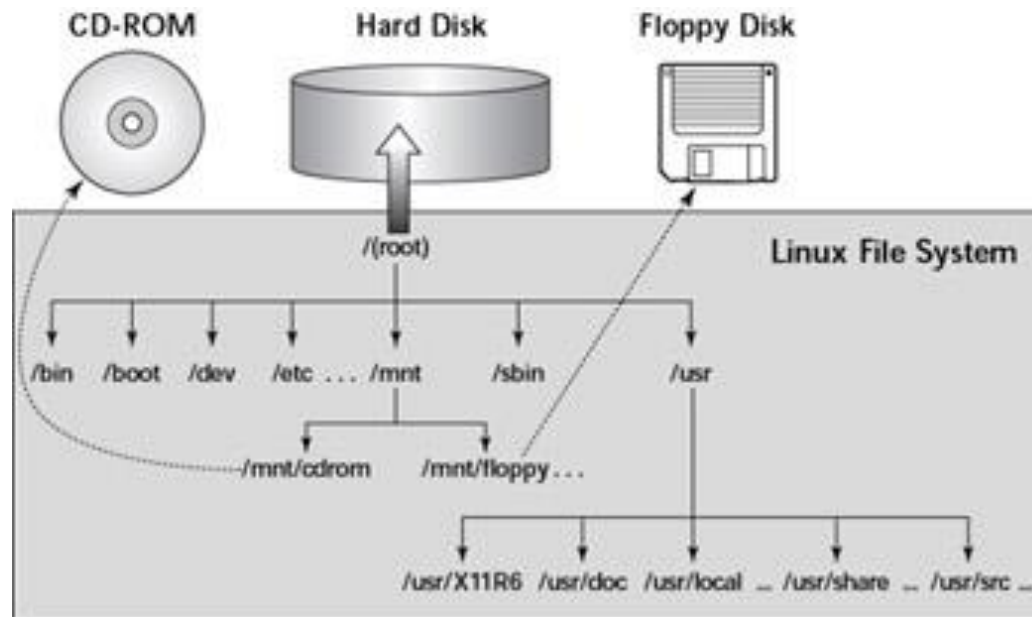
- 상위 폴더로 이동



마운트란?

- **마운트(mount) 의미**

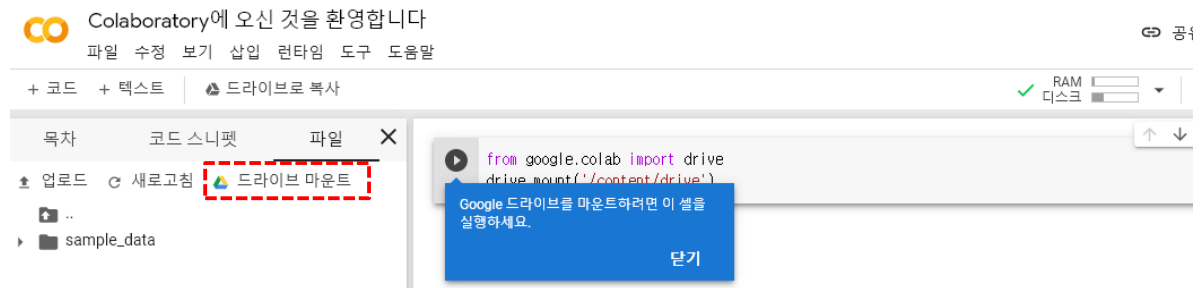
- 물리적인 장치(예로 구글 드라이브)를 특정 위치 즉 디렉터리에 연결시켜주는 것



드라이브 마운트

구글 드라이브와 서버 연결

- 버튼 '드라이브 마운트' 선택
- 마운트 기능의 스니펫 코드가 삽입

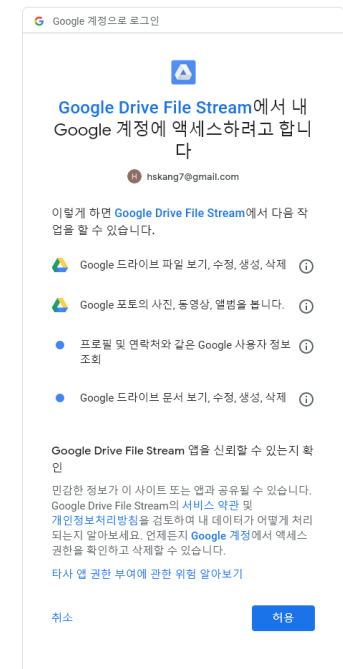
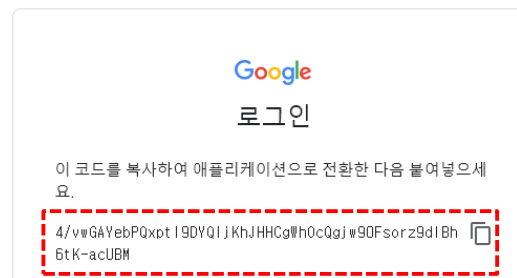
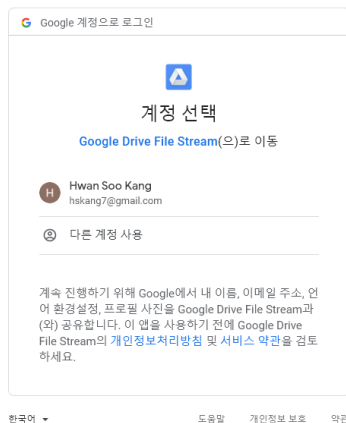


셀 실행 결과

- 인증 코드 복사

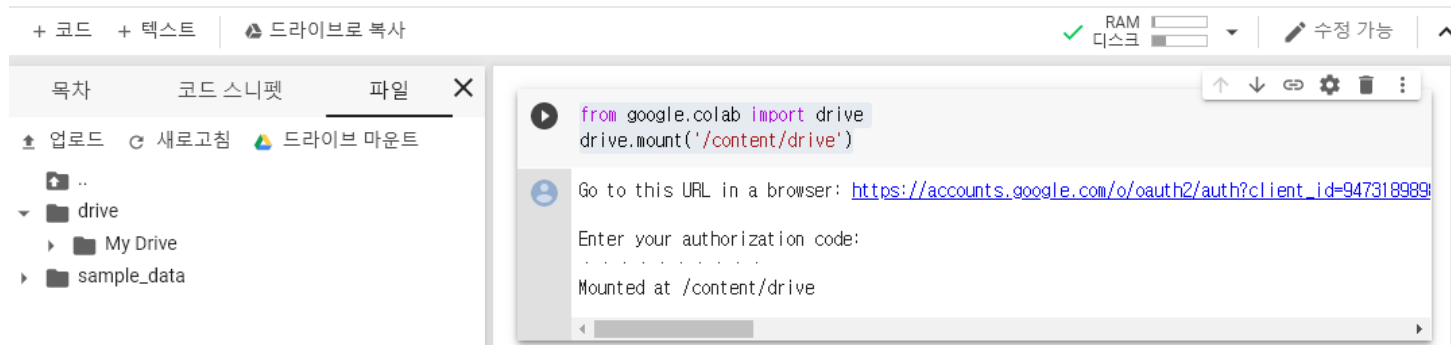
```
from google.colab import drive
drive.mount('/content/drive')
```

원하는 이름으로
수정 가능



연결된 드라이브 확인

- 셀 실행 이후
 - 복사된 인증 코드 붙여 넣고 Enter
- 메뉴 새로 고침 확인
 - 폴더 drive 자동 생성
 - 하부 구글의 My Drive 확인 가능



마운트된 구글 드라이브 활용

- 구글 드라이브의 파일을 서버에서 활용 가능
 - 구글 드라이브의 루트인 My Drive에 저장된 파이썬 소스 *.py 파일 실행 가능

The screenshot shows a Jupyter Notebook with three tabs: '목차' (Table of Contents), '코드 스니펫' (Code Snippets), and '파일' (Files). The '파일' tab is active, displaying a file explorer on the left and a code editor on the right.

File Explorer (Left Panel):

- 업로드 (Upload)
- 새로고침 (Refresh)
- 드라이브 마운트 (Drive Mount)
- .. (Parent Directory)
- drive (Mounted Drive)
 - My Drive (Root of the mounted drive)
 - sample_data (Subdirectory)

Code Editor (Right Panel):

```
[1] from google.colab import drive
    drive.mount('/content/drive')

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989...
Enter your authorization code:
...
Mounted at /content/drive

[2] pwd

'/content'

[5] cd drive/My Drive

/content/drive/My Drive

[7] !dir *.py

stringop.py

[8] cat stringop.py

print("원의 원주율 " + '3.141592')
print("python " + 'programming' + ' language')
print('파이썬 언어는 ' + " 강력하다")
print('파이썬 ' + "언어! " + 3 + "방가 ")

[10] !python stringop.py

원의 원주율 3.141592
python programming language
파이썬 언어는 강력하다
파이썬 언어! 방가 방가 방가
```

The code cell [10] is highlighted with a red dashed box, indicating the execution of the Python script.

Colab 서버의 폴더에
깃허브를 복사 하기

서버에 깃허브 복사

• 명령어

- !git clone https://github.com/neowizard2018/neowizard.git

Colaboratory에 오신 것을 환영합니다
파일 수정 보기 삽입 런타임 도구 도움말 오후 4:58 이후 저장되지 않은 변경사항

+ 코드 + 텍스트 + 드라이브로 복사

RAM 디스크

수정 가능

목차 코드 스니펫 파일

업로드 새로고침 드라이브 마운트

..

drive

neowizard

DataAnalysis

data

DA_LEC_01_Example.ipynb

DA_LEC_02_Example.ipynb

DA_LEC_03_Example.ipynb

DA_LEC_04_Example.ipynb

DA_LEC_05_Example.ipynb

DA_LEC_06_Example.ipynb

MachineLearning

README.md

sample_data

[11] cd ..

/content/drive

[12] cd ..

/content

[14] !git clone https://github.com/neowizard2018/neowizard.git

Cloning into 'neowizard'...

remote: Enumerating objects: 12, done.

remote: Counting objects: 100% (12/12), done.

remote: Compressing objects: 100% (12/12), done.

remote: Total 194 (delta 2), reused 0 (delta 0), pack-reused 182

Receiving objects: 100% (194/194), 269.92 MiB | 35.97 MiB/s, done.

Resolving deltas: 100% (62/62), done.

[15] ls

drive/ neowizard/ sample_data/

[16] cd neowizard

/content/neowizard

%ls

DataAnalysis/ MachineLearning/ README.md

디스크 79.24 GB 사용 가능

마운트된 구글 드라이브 하부 깃허브 복사

• 마운트된 My Drive로 이동

– 복사 명령

- `!git clone https://github.com/hunkim/DeepLearningZeroToAll`

Colaboratory에 오신 것을 환영합니다
파일 수정 보기 삽입 런타임 도구 도움말 오후 4:58 이후 저장되지 않은 변경사항

+ 코드 + 텍스트 + 드라이브로 복사

RAM 디스크

수정 가능

목차 코드 스니펫 파일 X

업로드 새로고침 드라이브 마운트

drive

My Drive

00. 개인

01. 재정

05. 집필

2017 New C로 배우는 프로그래밍 기초

2019 가을 2개의 논문을 위한 설문

Colab Notebooks

DeepLearningZeroToAll

chainer

ipynb

lab-01-basics.ipynb

lab-02-1-linear_regression.ipynb

lab-02-2-linear_regression_fee...

lab-02-3-linear_regression_tens...

lab-03-1-minimizing_cost_sho...

lab-03-2-minimizing_cost_gradi...

lab-03-3-minimizing_cost_tf_o...

lab-03-X-minimizing_cost_tf_gr...

lab-05-1-logistic_regression.ipy...

lab-08-tensor_manipulation.ipy...

디스크 79.23 GB 사용 가능

```
[14] remote: Compressing objects: 100% (12/12), done.
remote: Total 194 (delta 2), reused 0 (delta 0), pack-reused 182
Receiving objects: 100% (194/194), 269.92 MiB | 35.97 MiB/s, done.
Resolving deltas: 100% (62/62), done.

[15] ls

drive/ neowizard/ sample_data/

[16] cd neowizard

/content/neowizard

[17] !ls

DataAnalysis/ MachineLearning/ README.md

[18] cd ../drive/My Drive

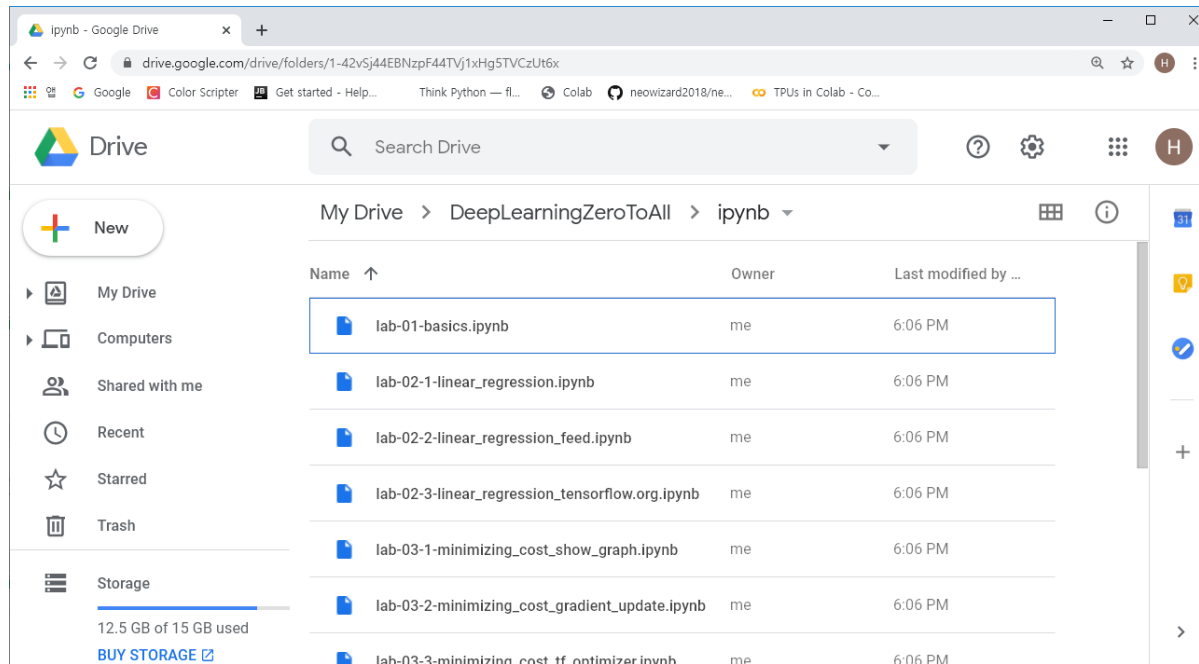
/content/drive/My Drive

[19] !git clone https://github.com/hunkim/DeepLearningZeroToAll

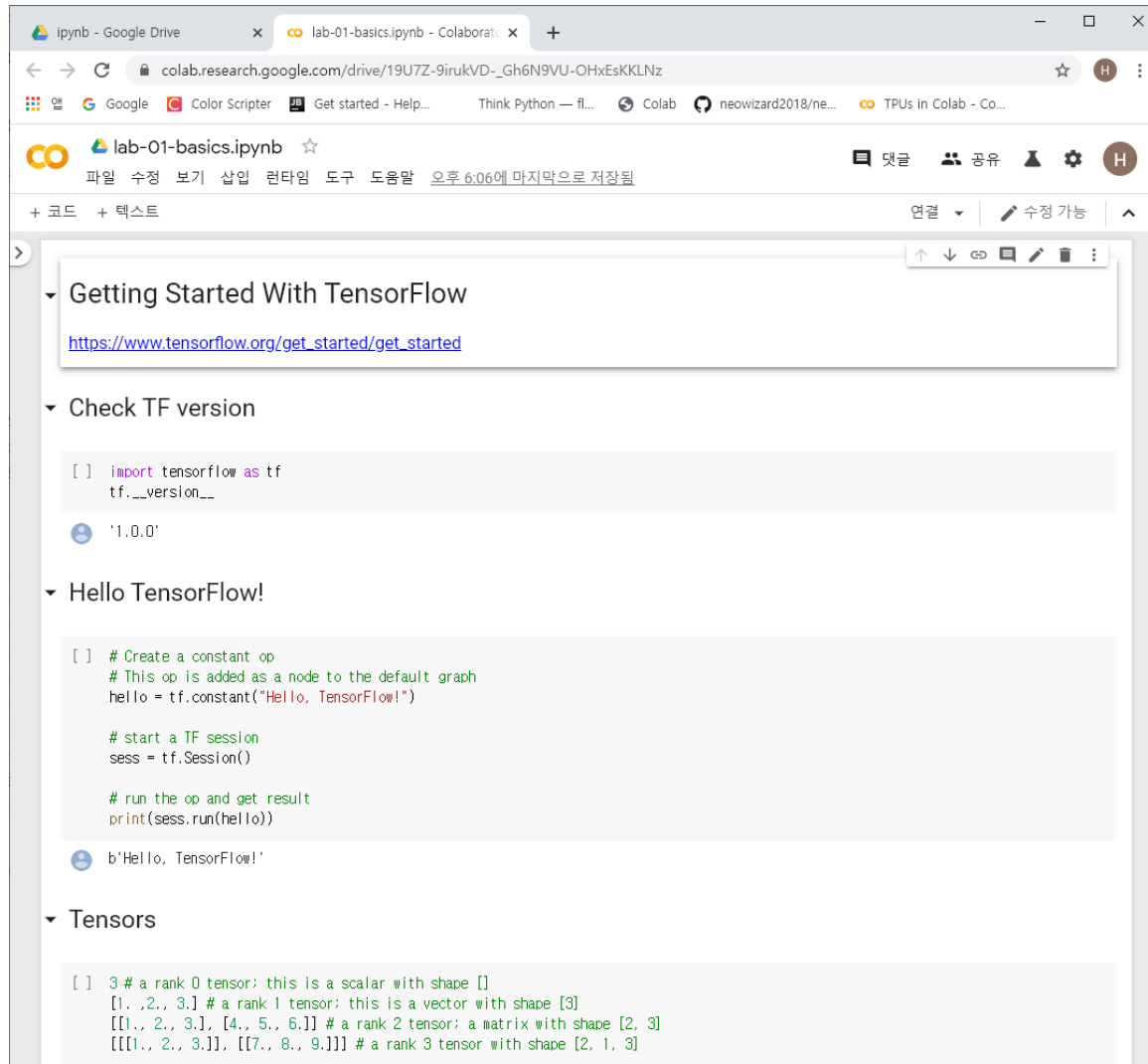
Cloning into 'DeepLearningZeroToAll'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 1688 (delta 2), reused 6 (delta 2), pack-reused 1678
Receiving objects: 100% (1688/1688), 733.45 KiB | 1.74 MiB/s, done.
Resolving deltas: 100% (1101/1101), done.
```

구글 드라이브에서 확인

- 폴더 My Drive/DeepLearningSeroToAll/ipynb
 - 여러 파일 확인
- 파일 lab-01-basic.ipynb
 - Colab으로 열기



Colab으로 연 노트 파일



구글 Colab 서버 정보 확인

서버 정보 확인 명령

[1] `!cat /etc/issue.net`

↳ Ubuntu 18.04.3 LTS

[2] `cat /etc/issue.net`

↳ Ubuntu 18.04.3 LTS

[4] `!head /proc/cpuinfo`

```
↳ processor      : 0
   vendor_id     : GenuineIntel
   cpu family    : 6
   model         : 79
   model name    : Intel(R) Xeon(R) CPU @ 2.20GHz
   stepping      : 0
   microcode     : 0x1
   cpu MHz       : 2200.000
   cache size    : 56320 KB
   physical id   : 0
```

[6] `!head /proc/meminfo`

```
↳ MemTotal:      13335188 kB
   MemFree:      10804652 kB
   MemAvailable: 12509160 kB
   Buffers:      70536 kB
   Cached:       1790100 kB
   SwapCached:   0 kB
   Active:       692488 kB
   Inactive:     1593188 kB
   Active(anon): 402272 kB
   Inactive(anon): 312 kB
```

[8] `!df -h`

```
↳ Filesystem      Size  Used Avail Use% Mounted on
   overlay         108G   29G   75G   28% /
   tmpfs           64M    0    64M    0% /dev
   tmpfs           6.4G    0   6.4G    0% /sys/fs/cgroup
   tmpfs           6.4G   8.0K   6.4G    1% /var/colab
   /dev/sda1       114G   30G   85G   26% /etc/hosts
   shm             6.0G   4.0K   6.0G    1% /dev/shm
   tmpfs           6.4G    0   6.4G    0% /proc/acpi
   tmpfs           6.4G    0   6.4G    0% /proc/scsi
   tmpfs           6.4G    0   6.4G    0% /sys/firmware
```

[9] `pwd`

↳ `"/content"`

[10] `ls -al`

```
↳ total 16
   drwxr-xr-x 1 root root 4096 Jan 13 16:38 ./
   drwxr-xr-x 1 root root 4096 Jan 21 08:43 ../
   drwxr-xr-x 1 root root 4096 Jan 13 16:38 .config/
   drwxr-xr-x 1 root root 4096 Jan 13 16:38 sample_data/
```

[12] `!python --version`

↳ Python 3.6.9

[13] `import tensorflow as tf`
`print(tf.__version__)`

↳ The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
 We recommend you [upgrade](#) now or ensure your notebook will continue to use TensorFlow 1.x via the `!tensorflow_version 1.x` magic: [more info](#).
 1.15.0