Google Colaboratory

Google Colab 특징

- 구글 클라우드 기반의 무료 개발 환경 서비스
- 환경설정 및 실행까지 매우 빠른 환경
- 딥러닝 실행이 가능한 정도의 고사양 환경 제공
- Jupyter Notebook 환경 제공
- 대부분의 패키지들이 이미 설치된 환경 제공
- 여러 사용자와 동시에 사용 가능
- PC, 태블릿, 모바일 상관없이 인터넷 브라우저만 있으면 언제 어디서나 접속 가능
- 목차나 Markdown 미리보기 등 다양한 기능 제공
- 구글 드라이브와 연동이 가능
- Git이나 Github와 쉽게 연동 가능

Google Colab 주의사항

- 구글 계정 필요
- 최대 세션 유지 시간이 존재 (12시간)
- 세션이 종료되면 작업중이던 데이터는 모두 삭제
- 소스 코드는 구글 드라이브에 저장

Google Colab 사양

- 플랫폼

```
import platform
platform.platform()
```

```
'Linux-4.19.104+-x86_64-with-Ubuntu-18.04-bionic'
```

- 운영체제

```
!cat /etc/issue.net
```

```
Ubuntu 18.04.3 LTS
```

- CPU 사양

```
!cat /proc/cpuinfo
```

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 63
model name      : Intel(R) Xeon(R) CPU @ 2.30GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2300.000
cache size      : 46080 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 cl
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs itlb_
bogomips        : 4600.00
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id       : GenuineIntel
cpu family      : 6
model           : 63
model name      : Intel(R) Xeon(R) CPU @ 2.30GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2300.000
```

```
cache size      : 46080 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
apicid          : 1
initial apicid  : 1
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 cl
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs itlb_
bogomips        : 4600.00
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

- 메모리 사양

```
!cat /proc/meminfo
```

```
MemTotal:       13333556 kB
MemFree:        10714340 kB
MemAvailable:   12496256 kB
Buffers:           71392 kB
Cached:          1869456 kB
SwapCached:            0 kB
Active:           706040 kB
Inactive:        1664980 kB
Active(anon):     409692 kB
Inactive(anon):      316 kB
Active(file):     296348 kB
Inactive(file):  1664664 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:              1048 kB
Writeback:             0 kB
AnonPages:        430168 kB
Mapped:           221084 kB
Shmem:               904 kB
Slab:             159552 kB
SReclaimable:     123616 kB
SUnreclaim:        35936 kB
KernelStack:        3344 kB
PageTables:         5268 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     6666776 kB
Committed_AS:    2483344 kB
VmallocTotal:   34359738367 kB
VmallocUsed:           0 kB
```

```
VmallocChunk:          0 kB
Percpu:              912 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:       74940 kB
DirectMap2M:     6215680 kB
DirectMap1G:     9437184 kB
```

- 디스크 사양

```
!df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
overlay         108G   32G   72G  31% /
tmpfs            64M     0   64M   0% /dev
tmpfs           6.4G     0  6.4G   0% /sys/fs/cgroup
shm             5.9G     0  5.9G   0% /dev/shm
tmpfs           6.4G   12K  6.4G   1% /var/colab
/dev/sda1       114G   33G   82G  29% /etc/hosts
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
```

- 파이썬 버전

```
!python --version
```

```
Python 3.6.9
```

Google Colab 런타임

- Colab에서 고성능 하드웨어로 GPU나 TPU 사용 가능
- 런타임 유형 변경(Change runtime type) 필요
  - None: CPU만 사용
  - GPU: 하드웨어 가속으로 GPU 사용
  - TPU: 하드웨어 가속으로 TPU 사용

```
!nvidia-smi
```

```
NVIDIA-SMI has failed because it couldn't communicate with the NVIDIA driver. Make sure that
```

파일 저장 및 다운로드/업로드

- Jupyter Notebook 환경에서 파일 저장 및 다운로드

```
%%writefile test.txt
text
```

```
Writing test.txt
```

```
cat test.txt
```

```
text
```

```
from google.colab import files
files.download('test.txt')
```

```
upload = files.upload()
```

```
파일 선택  선택된 파일 없음
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving iconfinder_Data_analysis_3448017.png to iconfinder_Data_analysis_3448017.png
```

```
!ls
```

```
iconfinder_Data_analysis_3448017.png  sample_data  test.txt
```

Google Drive 연동

- Google Colab은 Google Drive와 `mount`를 통해 쉽게 연동 가능
- Google Drive에 소스 코드 저장 뿐만 아니라 파일을 열거나 저장 가능

```
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=947318989803

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

```
!ls /content/drive
```

```
'My Drive'
```

Jupyter Notebook

- IPython Shell의 브라우저 기반 그래픽 인터페이스
- 파이썬과 IPython 문장 실행
- 서식 있는 텍스트와 정적/동적 시각화, 수학 공식 표현

help()

- 파이썬 객체에 대한 요약 정보와 사용법 보기

```
help(min)
```

```
Help on built-in function min in module builtins:

min(...)
    min(iterable, *[, default=obj, key=func]) -> value
    min(arg1, arg2, *args, *[, key=func]) -> value

    With a single iterable argument, return its smallest item. The
    default keyword-only argument specifies an object to return if
    the provided iterable is empty.
    With two or more arguments, return the smallest argument.
```

?로 설명 보기

- 객체 요약 정보 및 사용법이 있는 docstring 보기

```
max?
```

```
li = ['One', 'Two', 'Three']
li?
```

```
li.count?
```

```
def power(b, n):
  """b의 n승을 반환"""
  return pow(b, n)
```

```
power?
```

```
power??
```

탭(tab) 자동완성

- 객체와 모듈, 인터페이스의 내용을 자동 완성

```
li = ['One', 'Two', 'Three']
li.append
```

```
<function list.append>
```

와일드카드(wildcard) 매칭

- 단어의 중간이나 마지막 글자로 매칭하여 객체나 속성 찾기

```
*Error?
```

```
str.*index*?
```

매직 명령어(magic commands)

- Jupyter Notebook 환경에서 파이썬 파일 저장 및 실행

```
%%writefile test.py
print('Hello Colab')
```

```
Writing test.py
```

```
%run test.py
```

```
Hello Colab
```

- 코드 실행 시간 측정

```
%%timeit?
```

```
%timeit li = [n ** n for n in range(10000)]
```

```
1 loop, best of 3: 5.78 s per loop
```

```
%%timeit
li = []
for n in range(10000):
  li.append(n ** n)
```

```
1 loop, best of 3: 5.67 s per loop
```

입력과 출력 이력

- In과 Out 객체를 통해 이전 명령어와 명령어의 출력 결과 보기

```
In
```

```
['',
 'import platform₩nplatform.platform()',
 "get_ipython().system('cat /etc/issue.net')",
 "get_ipython().system('cat /proc/cpuinfo')",
 "get_ipython().system('cat /proc/meminfo')",
 "get_ipython().system('df -h')",
 "get_ipython().system('python --version')",
 "get_ipython().system('nvidia-smi')",
 "get_ipython().run_cell_magic('writefile', 'test.txt', 'text')",
 "get_ipython().magic('cat test.txt')",
 "from google.colab import files₩nfiles.download('test.txt')",
 'upload = files.upload()',
 "get_ipython().system('ls')",
 "from google.colab import drive₩ndrive.mount('/content/drive')",
 "get_ipython().system('ls /content/drive')",
 'help(min)',
 "get_ipython().magic('pinfo max')",
 "li = ['One', 'Two', 'Three']₩nget_ipython().magic('pinfo li')",
 "get_ipython().magic('pinfo li.count')",
 'def power(b, n):₩n  """b의 n승을 반환"""₩n  return pow(b, n)',
 "get_ipython().magic('pinfo power')",
 "get_ipython().magic('pinfo2 power')",
 "li = ['One', 'Two', 'Three']₩nli.append",
 "get_ipython().magic('psearch *Error')",
 "get_ipython().magic('psearch str.*index*')",
 'get_ipython().run_cell_magic(₩'writefile₩', ₩'test.py₩', "print(₩'Hello Colab₩')")',
 "get_ipython().magic('run test.py')",
 "get_ipython().magic('pinfo %%timeit')",
 "get_ipython().magic('timeit li = [n ** n for n in range(10000)]')",
 "get_ipython().run_cell_magic('timeit', '', 'li = []₩₩nfor n in range(10000):₩₩n
```

```
li.append(n ** n)')",
 'In']
```

In[25]

```
'get_ipython().run_cell_magic(W'writefileW', W'test.pyW', "print(W'Hello ColabW')")'
```

Out

```
{1: 'Linux-4.19.104+-x86_64-with-Ubuntu-18.04-bionic',
 22: <function list.append>,
 30: ['',
  'import platformWnplatform.platform()',
  "get_ipython().system('cat /etc/issue.net')",
  "get_ipython().system('cat /proc/cpuinfo')",
  "get_ipython().system('cat /proc/meminfo')",
  "get_ipython().system('df -h')",
  "get_ipython().system('python --version')",
  "get_ipython().system('nvidia-smi')",
  "get_ipython().run_cell_magic('writefile', 'test.txt', 'text')",
  "get_ipython().magic('cat test.txt')",
  "from google.colab import filesWnfiles.download('test.txt')",
  'upload = files.upload()',
  "get_ipython().system('ls')",
  "from google.colab import driveWndrive.mount('/content/drive')",
  "get_ipython().system('ls /content/drive')",
  'help(min)',
  "get_ipython().magic('pinfo max')",
  "li = ['One', 'Two', 'Three']Wnget_ipython().magic('pinfo li')",
  "get_ipython().magic('pinfo li.count')",
  'def power(b, n):Wn  """b의 n승을 반환"""Wn  return pow(b, n)',
  "get_ipython().magic('pinfo power')",
  "get_ipython().magic('pinfo2 power')",
  "li = ['One', 'Two', 'Three']Wnli.append",
  "get_ipython().magic('psearch *Error')",
  "get_ipython().magic('psearch str.*index*')",
  'get_ipython().run_cell_magic(W'writefileW', W'test.pyW', "print(W'Hello ColabW')")',
  "get_ipython().magic('run test.py')",
  "get_ipython().magic('pinfo %%timeit')",
  "get_ipython().magic('timeit li = [n ** n for n in range(10000)]')",
  "get_ipython().run_cell_magic('timeit', '', 'li = []WWnfor n in range(10000):WWn
li.append(n ** n)')",
  'In',
  'In[25]',
  'Out'],
 31: 'get_ipython().run_cell_magic(W'writefileW', W'test.pyW', "print(W'Hello ColabW')")'}
```

Out[1]

```
'Linux-4.19.104+-x86_64-with-Ubuntu-18.04-bionic'
```

```
print("In[1]: " + In[1] + "WnOut[1]: " + Out[1])
```

```
In[1]: import platform
platform.platform()
Out[1]: Linux-4.19.104+-x86_64-with-Ubuntu-18.04-bionic
```

- _를 이용해 이전 출력값 출력하기

```
print(_)
```

```
Linux-4.19.104+-x86_64-with-Ubuntu-18.04-bionic
```

```
print(__)
```

```
get_ipython().run_cell_magic('writefile', 'test.py', "print('Hello Colab')")
```

```
print(___)
```

```
['', 'import platformWnplatform.platform()', "get_ipython().system('cat /etc/issue.net')", "
```

```
_30
```

```
['',
 'import platformWnplatform.platform()',
 "get_ipython().system('cat /etc/issue.net')",
 "get_ipython().system('cat /proc/cpuinfo')",
 "get_ipython().system('cat /proc/meminfo')",
 "get_ipython().system('df -h')",
 "get_ipython().system('python --version')",
 "get_ipython().system('nvidia-smi')",
 "get_ipython().run_cell_magic('writefile', 'test.txt', 'text')",
 "get_ipython().magic('cat test.txt')",
 "from google.colab import filesWnfiles.download('test.txt')",
 'upload = files.upload()',
 "get_ipython().system('ls')",
 "from google.colab import driveWndrive.mount('/content/drive')",
 "get_ipython().system('ls /content/drive')",
 'help(min)',
 "get_ipython().magic('pinfo max')",
 "li = ['One', 'Two', 'Three']Wnget_ipython().magic('pinfo li')",
 "get_ipython().magic('pinfo li.count')",
 'def power(b, n):Wn  """b의 n승을 반환"""Wn  return pow(b, n)',
 "get_ipython().magic('pinfo power')",
 "get_ipython().magic('pinfo2 power')",
 "li = ['One', 'Two', 'Three']Wnli.append",
 "get_ipython().magic('psearch *Error')",
 "get_ipython().magic('psearch str.*index*')",
 'get_ipython().run_cell_magic(W'writefileW', W'test.pyW', "print(W'Hello ColabW')")',
 "get_ipython().magic('run test.py')",
 "get_ipython().magic('pinfo %%timeit')",
 "get_ipython().magic('timeit li = [n ** n for n in range(10000)]')",
 "get_ipython().run_cell_magic('timeit', '', 'li = []WWnfor n in range(10000):WWn
li.append(n ** n)')",
 'In',
 'In[25]',
 'Out',
 'Out[1]',
 'print("In[1]: " + In[1] + "WWnOut[1]: " + Out[1])',
 'print(_)',
```

```
 'print(__)',
 'print(___)',
 '_30']
```

- `%histroy`를 이용한 입력 이력 살펴보기

```
%history -n 1-7
```

```
   1:
import platform
platform.platform()
   2: !cat /etc/issue.net
   3: !cat /proc/cpuinfo
   4: !cat /proc/meminfo
   5: !df -h
   6: !python --version
   7: !nvidia-smi
```

- `%rerun`을 이용해 이전 입력 이력 다시 실행

```
%rerun
```

```
=== Executing: ===
%history -n 1-7
=== Output: ===
   1:
import platform
platform.platform()
   2: !cat /etc/issue.net
   3: !cat /proc/cpuinfo
   4: !cat /proc/meminfo
   5: !df -h
   6: !python --version
   7: !nvidia-smi
```

셸 명령어

- 텍스트 기반의 셸 명령어 처리
- `!` 문자를 명령어 앞에 붙여서 셸 명령어 사용 가능

- `ls`: 디렉토리 리스트 보기

```
!ls
```

```
drive  iconfinder_Data_analysis_3448017.png  sample_data  test.py  test.txt
```

- `pwd`: 현재 경로 보기

```

```

```
!pwd
```

```
/content
```

- `cd`: 디렉토리 변경
- IPython에서는 임시 셸에서 실행

```
!cd sample_data && ls
```

```
anscombe.json                   mnist_test.csv
california_housing_test.csv     mnist_train_small.csv
california_housing_train.csv    README.md
```

- `%cd`: 지속적인 디렉토리 변경

```
%cd sample_data
```

```
/content/sample_data
```

- `echo`: 화면 출력

```
!echo "Shell"
```

```
Shell
```

- `mkdir`: 디렉토리 생성

```
!mkdir tmp
```

```
mkdir: cannot create directory ‘tmp’: File exists
```

```
!ls
```

```
anscombe.json                   mnist_test.csv          tmp
california_housing_test.csv     mnist_train_small.csv
california_housing_train.csv    README.md
```

- `cat`: 파일 보기

```
!cat README.md
```

```
This directory includes a few sample datasets to get you started.

*   `california_housing_data*.csv` is California housing data from the 1990 US
    Census; more information is available at:
    https://developers.google.com/machine-learning/crash-course/california-housing-data-desc
```

```
*   `mnist_*.csv` is a small sample of the
    [MNIST database](https://en.wikipedia.org/wiki/MNIST_database), which is
    described at: http://yann.lecun.com/exdb/mnist/

*   `anscombe.json` contains a copy of
    [Anscombe's quartet](https://en.wikipedia.org/wiki/Anscombe%27s_quartet); it
    was originally described in

    Anscombe, F. J. (1973). 'Graphs in Statistical Analysis'. American
    Statistician. 27 (1): 17-21. JSTOR 2682899.

    and our copy was prepared by the
    [vega_datasets library](https://github.com/altair-viz/vega_datasets/blob/4f67bdaad10f45e
```

- `cp`: 디렉토리/파일 복사

```
!cp README.md tmp
```

```
!ls tmp
```

```
README.md
```

- `rm`: 디렉토리/파일 삭제

```
!rm -r tmp
```

```
!ls
```

```
anscombe.json                mnist_test.csv
california_housing_test.csv  mnist_train_small.csv
california_housing_train.csv README.md
```

마크다운(Markdown)

- 문법이 간단하고, 사용이 쉬움

제목(Heading) 표시

```
# Heading 1
## Heading 2
### Heading 3
#### Heading 4
##### Heading 5
###### Heading 6
```

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

목록(List) 구성

```
1. 첫째
2. 둘째
3. 셋째
* 1단계
  + 2단계
    - 3단계
```

1. 첫째
2. 둘째
3. 셋째

- 1단계
    - 2단계
        - 3단계

폰트 스타일(Font Style)

```
**bold**
__bold
*italic*
_italic
~~strike~~
<u>underbar</u>
```

bold __bold *italic* _italic ~~strike~~ <u>underbar</u>

인용구

```
> 인용 1
>> 인용 2
>>> 인용 3
```

> 인용 1
>
> > 인용 2
> >
> > > 인용 3

링크

```
[Google](https://www.google.com)
```

Google

이미지

```
![이미지](https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png)
```

수평선

```
---
```

표

```
| C1 | C2 | C3  |
|:---|---:|:---:|
| D1 | D2 | D3 |
```

C1	C2	C3
D1	D2	D3

코드

```
`inline code`
```

```
```block code```
```

`inline code`

```
block code
```
```