



데사 B0002

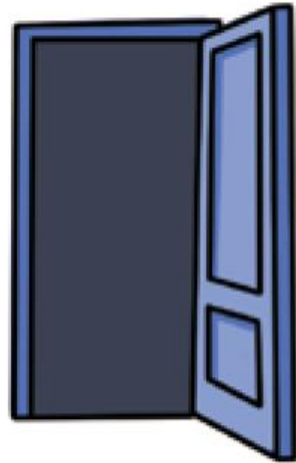
데이터마이닝이해와실습

김 태 완

kimtwan21@dongduk.ac.kr

파일 읽기

- 파일 읽기 3 단계
 - 파일 열기
 - 파일 읽기
 - 파일 닫기



1단계
파일 열기



2단계
파일 읽기



3단계
파일 닫기

파일 읽기

- 파일 읽기 3 단계

- 파일 열기

- 파일을 열기 위해서는 open() 함수에서 파일명을 지정하고, 읽기(Read)를 의미하는 "r"로 설정함
 - 모드(mode) : open() 함수의 마지막 매개변수
 - 파일을 열 때 어떤 용도로 열지 결정함
 - 파일 읽기용은 "r"을 씀

변수명 = open("파일경로/파일이름", "r")

종류	설명
생략	r과 동일하다.
r	읽기 모드, 기본값이다.
w	쓰기 모드, 기존에 파일이 있으면 덮어쓴다.
r+	읽기/쓰기 겸용 모드이다.
a	쓰기 모드, 기존에 파일이 있으면 이어서 쓴다. append의 약어이다.
t	텍스트 모드, 텍스트 파일을 처리한다. 기본값이다.
b	이진 모드, 이진 파일을 처리한다.

- 파일 읽기

- 파일 닫기

파일 읽기

- 파일 읽기 3 단계
 - 파일 열기
 - **파일 읽기**
 - 파일에서 데이터를 읽어올 수 있는 상태
 - **파일 닫기**
 - 파일과 관련된 모든 작업이 끝나면 파일을 정상적으로 닫아줘야 함
 - 파일을 닫기 위해 사용하는 변수는 `open()` 함수로 열었던 변수명임

```
변수명.close();
```

파일 읽기

- 파일에 담아둔 데이터를 읽기 위한 함수
 - readline()

```
f = open("/mnt/disk/aaa.txt", 'r')

while True:
    line = f.readline()
    if not line: # if line == "":
        break
    print(line)

f.close()
```

파일 읽기

- 파일에 담아둔 데이터를 읽기 위한 함수
 - readlines()

```
f = open("/mnt/disk/aaa.txt", 'r')
lines = f.readlines()

for line in lines:
    print(line.strip())

f.close()
```

파일 쓰기

- 원하는 문자열을 파일에 쓰는 함수
 - write()

```
f = open("result.txt", 'w')

for i in range(1, 11):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)

f.close()
```

파일 쓰기

- 파일을 열 때, 모드 값 중 ("w"가 아닌 "a"를 사용하는 경우)
 - 기존 파일을 열어 그 뒤에 문자열 쓰기

```
f = open("result.txt", 'a')

for i in range(1, 11):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)

f.close()
```


파일 닫기를 신경 쓰지 않고 싶을 때.

- with ~ as 구문 사용
 - 기존에는 파일을 열면 항상 close 해주는 것이 좋다.
 - 하지만 with ~ as 구문을 사용하면 들여쓰기가 벗어나는 순간 열린 파일 객체 f가 자동으로 close

```
f = open("foo.txt", 'w')  
f.write("Life is too short, you need python")  
f.close()
```



```
with open("foo.txt", "w") as f:  
    f.write("Life is too short, you need python")
```

문제

- 7명의 이름이 저장된 리스트 (name)에서 길이가 4인 이름만 “name.txt” 파일에 써보자.

```
f = open("name.txt", "w")
```

```
name = ['Alex', 'Emma', 'Smith', 'Jane', 'Ava', 'Charlotte', 'Evelyn']
```



```
f.close()
```

json

- JavaScript Object Notation (JSON)
 - Javascript 객체 문법으로 구조화된 데이터를 표현하기 위한 문자 기반의 표준 포맷
 - 웹 어플리케이션에서 데이터를 전송할 때 일반적으로 사용
 - xml, csv 등과 함께 많이 사용하지만 요즘은 JSON을 많이 사용
 - 우리가 배운 딕셔너리 형태로 구성 : {"name" : "tony", "age": 20}

```
[
  {
    "name": "Tony",
    "age": 20,
    "secretIdentity": "Dan Jukes",
    "powers": ["Radiation resistance", "Turning tiny", "Radiation blast"]
  },
  {
    "name": "Alex",
    "age": 39,
    "secretIdentity": "Jane Wilson",
    "powers": [
      "Million tonne punch",
      "Damage resistance",
      "Superhuman reflexes"
    ]
  }
]
```

[] : 배열 (array)
{ } : 객체 (object)

json

- JavaScript Object Notation (JSON)
 - Json 파일 쓰기 : `json.dump`

```
import json

x = [
    {'name': 'Alex', 'English': 90, 'Korean': 80, 'Math' : 92},
    {'name': 'Tom', 'English': 62, 'Korean': 99, 'Math' : 79},
    {'name': 'Jane', 'English': 78, 'Korean': 93, 'Math' : 90}
]

with open('test.json', 'w') as f:
    json.dump(x, f)
```

json

- JavaScript Object Notation (JSON)
 - Json 파일 쓰기 : `json.dump`

```
import json

d = dict()
d['id'] = 0
d['name'] = 'taewan'
d['mbti'] = 'istj'
d['like'] = ['drive', 'cook', 'travel']
d['check'] = True

print(d)

with open ('test.json', 'w') as f:
    json.dump(d, f, indent=4)
```

json

- JavaScript Object Notation (JSON)
 - Json 파일 읽기 : `json.load`

```
import json

with open('test.json','r') as f:
    info = json.load(f)

for i in info:
    print(i)
```

```
import json

with open('test.json','r') as f:
    info = json.load(f)

for i in info:
    for k,v in i.items():
        print(k,v)
    print('-'*10)
```

모듈

- 모듈(Module)
 - 변수, 함수, 클래스 등을 모아둔 것
 - 다른 프로그램에 호출될 변수, 함수, 클래스 등을 별도 파일로 제작
- 모듈을 불러오는 법
 - 모듈을 불러올 스크립트에 다음과 같은 코드를 추가한다

```
import 모듈이름
```

```
import 모듈이름 as 바꿀이름
```

다른 객체와 이름이 겹칠 경우 **다른 이름으로 바꿔** 불러올 수 있음

모듈

- 모듈(Module)
 - 표준 모듈, 사용자 정의 모듈, 서드 파티 모듈로 구분
 - 표준 모듈 : 파이썬에서 제공하는 모듈
 - 사용자 정의 모듈 : 직접 만들어서 사용하는 모듈
 - 서드 파티(3rd Party) 모듈 : 파이썬이 아닌 외부 회사나 단체에서 제공하는 모듈
 - 파이썬 표준 모듈이 모든 기능을 제공 않음
 - 서드 파티 모듈 덕분에 파이썬에서도 고급 프로그래밍 가능
 - 게임 개발 기능이 있는 pyGame, 윈도우창을 제공 하는 PyGTK, 데이터베이스 기능의 SQLAlchemy 등

모듈

- 파이썬에서 제공하는 모듈 예시

```
import math
```

```
a = math.cos(math.pi/3)  
print(a)
```

```
import random
```

```
x = random.randrange(3,9)  
print(x)
```

모듈

- 사용자 정의 모듈

202309.py

```
import func202309 as taewan  
  
taewan.is_prime_num(10)
```

func202309.py

```
def is_prime_num(n):  
    for i in range(2, n):  
        if n % i == 0:  
            print("소수가 아닙니다.")  
            return 0  
  
    print("소수입니다.")
```

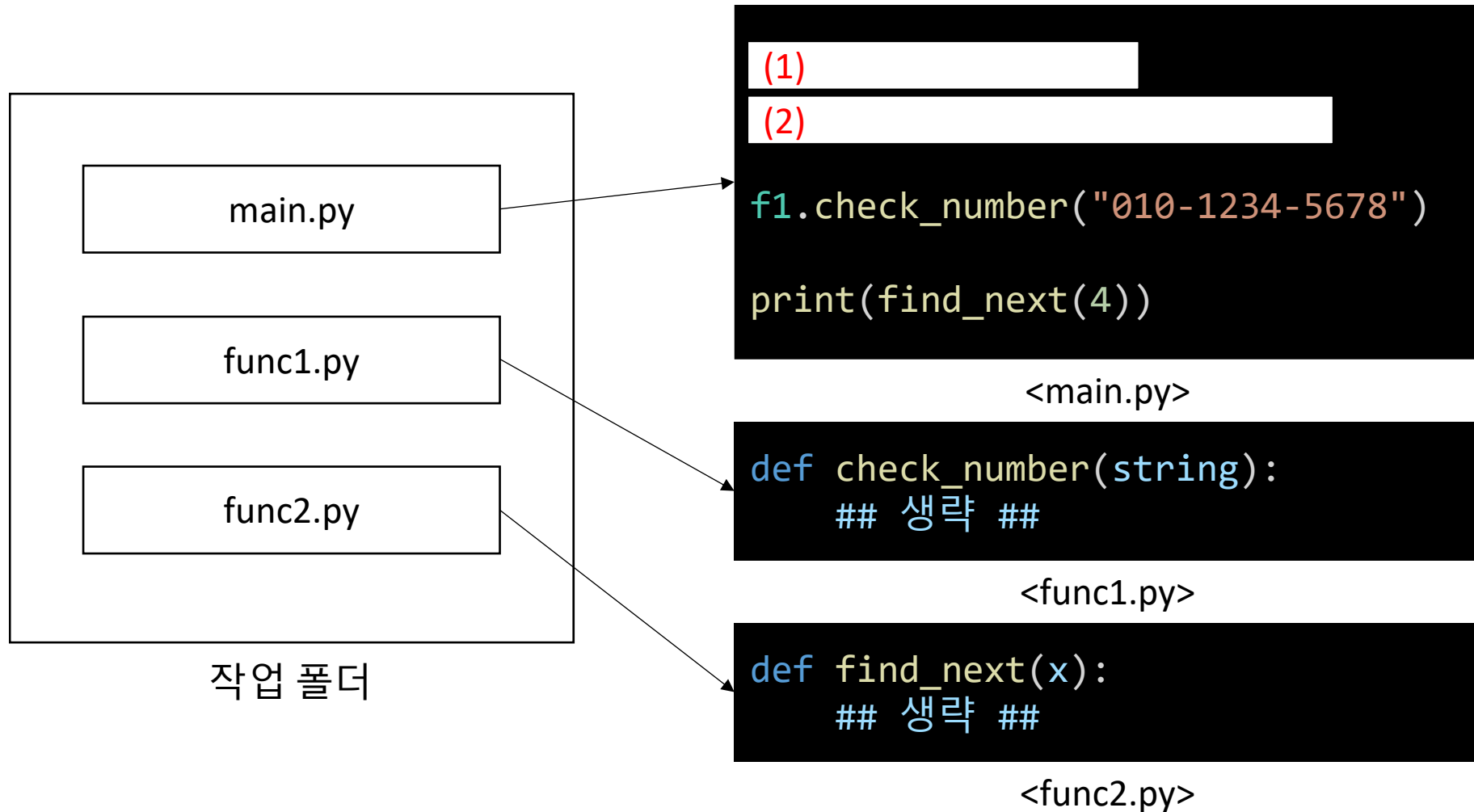
모듈

- 사용자 정의 모듈 : 함수명으로만 호출 가능

```
from func220428 import is_prime_num, pickup_even  
  
is_prime_num(10)  
pickup_even([3,4,5,6,7,8,9,10])
```

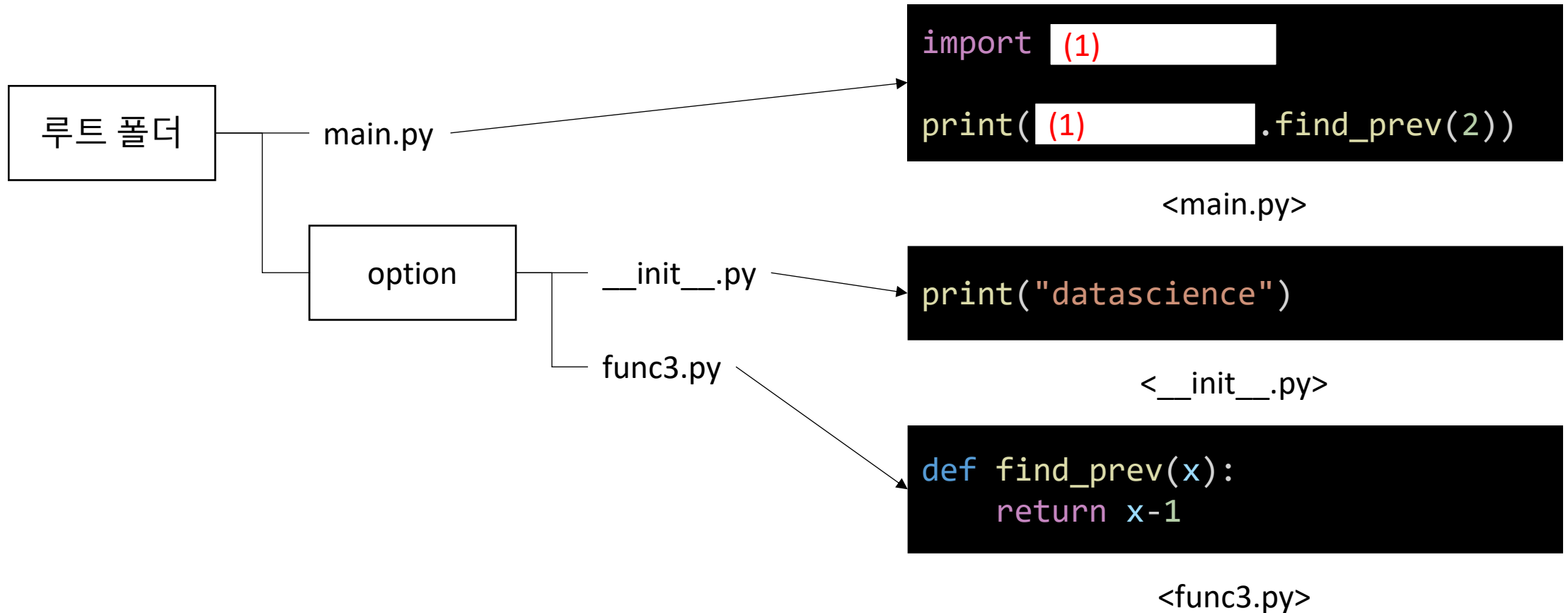
예시

- main.py 에 모듈을 불러오는 부분을 작성하여 코드를 완성해 보자



예시

- 루트 폴더 내 main.py와 "option" 이라는 이름의 폴더에 있는 두 python 파일이 그림과 같이 있을 때, main.py (1)에 들어갈 코드와 main.py를 실행했을 때 출력 결과를 구해보자



외부 라이브러리

- 파이썬 자체에는 없는 기능이지만, 다른 개발자가 좋은 기능들을 만들어 제공하는 묶음을 의미함
- 대부분의 외부 라이브러리는 완전히 무료로 사용할 수 있음



외부 라이브러리



초보 개발자



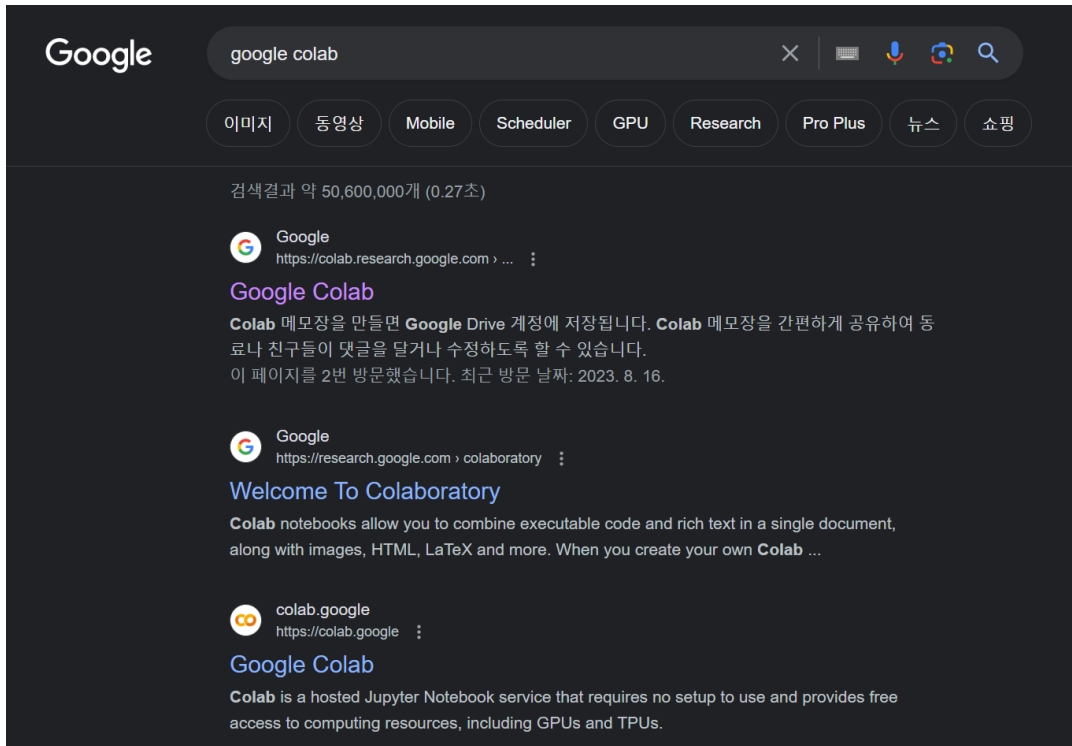
완성된 프로그램

외부 라이브러리

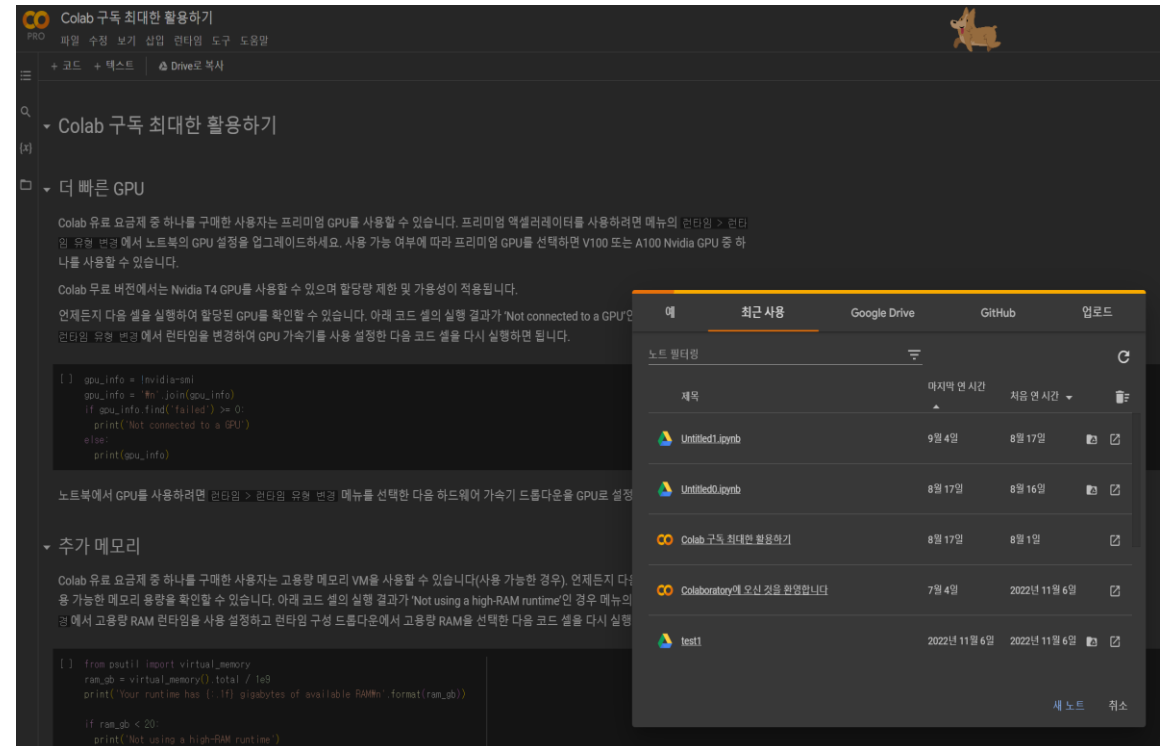
- 이번 학기 우리가 사용할 외부 라이브러리
 - numpy
 - pandas
 - matplotlib
 - scikit-learn

Python

- 이번 학기 이후 파이썬을 사용할 일이 있을 때는 항상 두 가지 방식이 있다는 사실을 알아 두자.
- 첫 번째 방법
 - 지금까지 해온 방법 – pc 에 python 설치 (visual studio code는 optional)
- 두 번째 방법
 - 클라우드 기반 개발 환경 : 구글 코랩 (Google Colab)
 - 인터넷 접속 & 구글 계정만 있다면 사용 가능



Google search results for "google colab". The top result is "Google Colab" with the URL "https://colab.research.google.com". Below it is "Welcome To Colaboratory" with a description: "Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your own Colab ...". The bottom result is "colab.google" with the URL "https://colab.google".



Google Colab interface showing GPU selection and memory usage. The "더 빠른 GPU" (Faster GPU) section is expanded, showing a table of available GPUs. The "추가 메모리" (Additional Memory) section is also expanded, showing a table of available memory options.

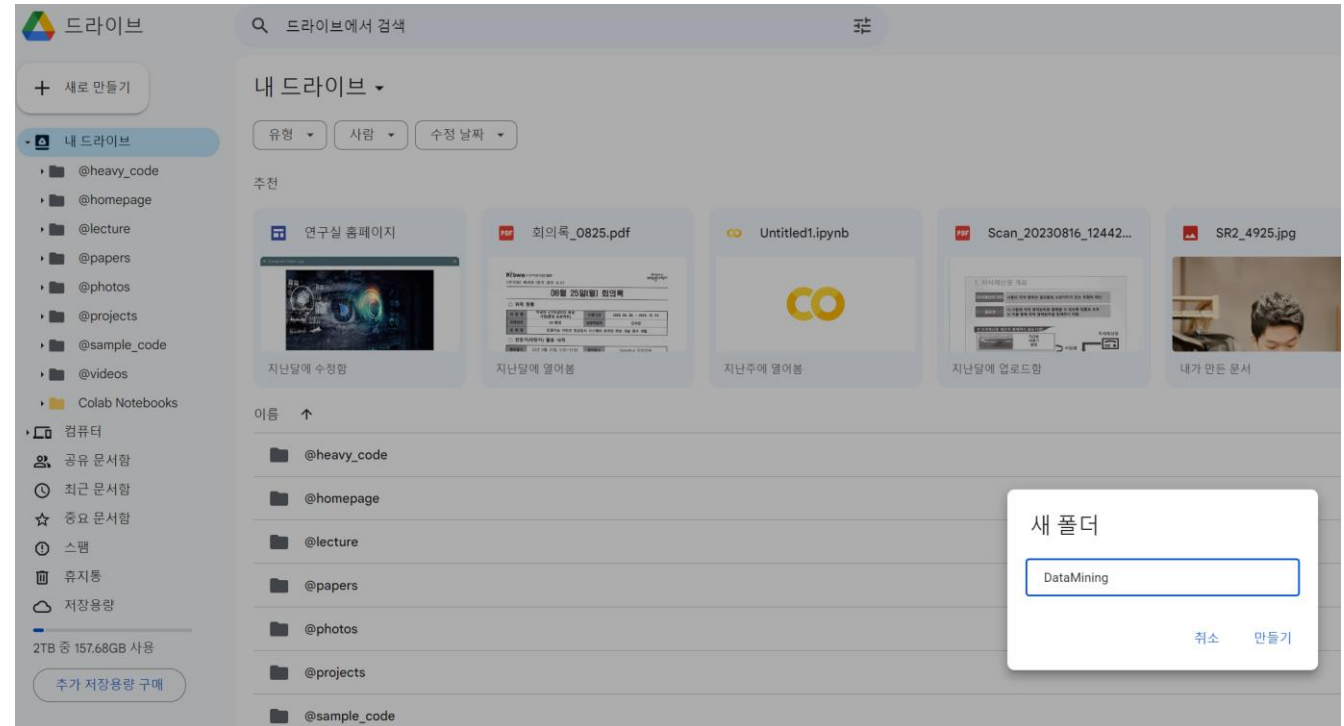
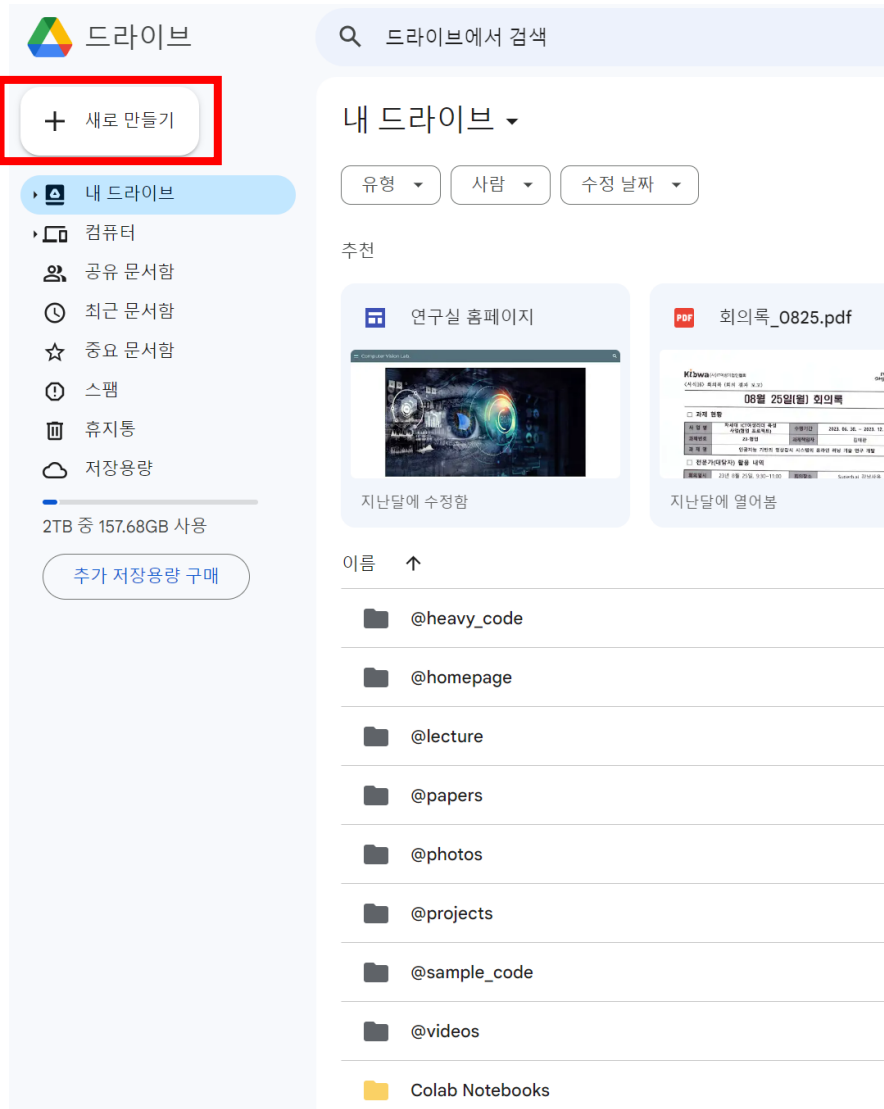
제목	마지막 연 시간	처음 연 시간
Untitled1.ipynb	9월 4일	8월 17일
Untitled2.ipynb	8월 17일	8월 16일
Colab 구독 최대한 활용하기	8월 17일	8월 1일
Colaboratory에 오신 것을 환영합니다	7월 4일	2022년 11월 6일
test1	2022년 11월 6일	2022년 11월 6일

Googel Colab

- Free Jupyter notebook environment
- Free GPU support
- Upload / Download files
- Github Friendly
- Easy Collaboration

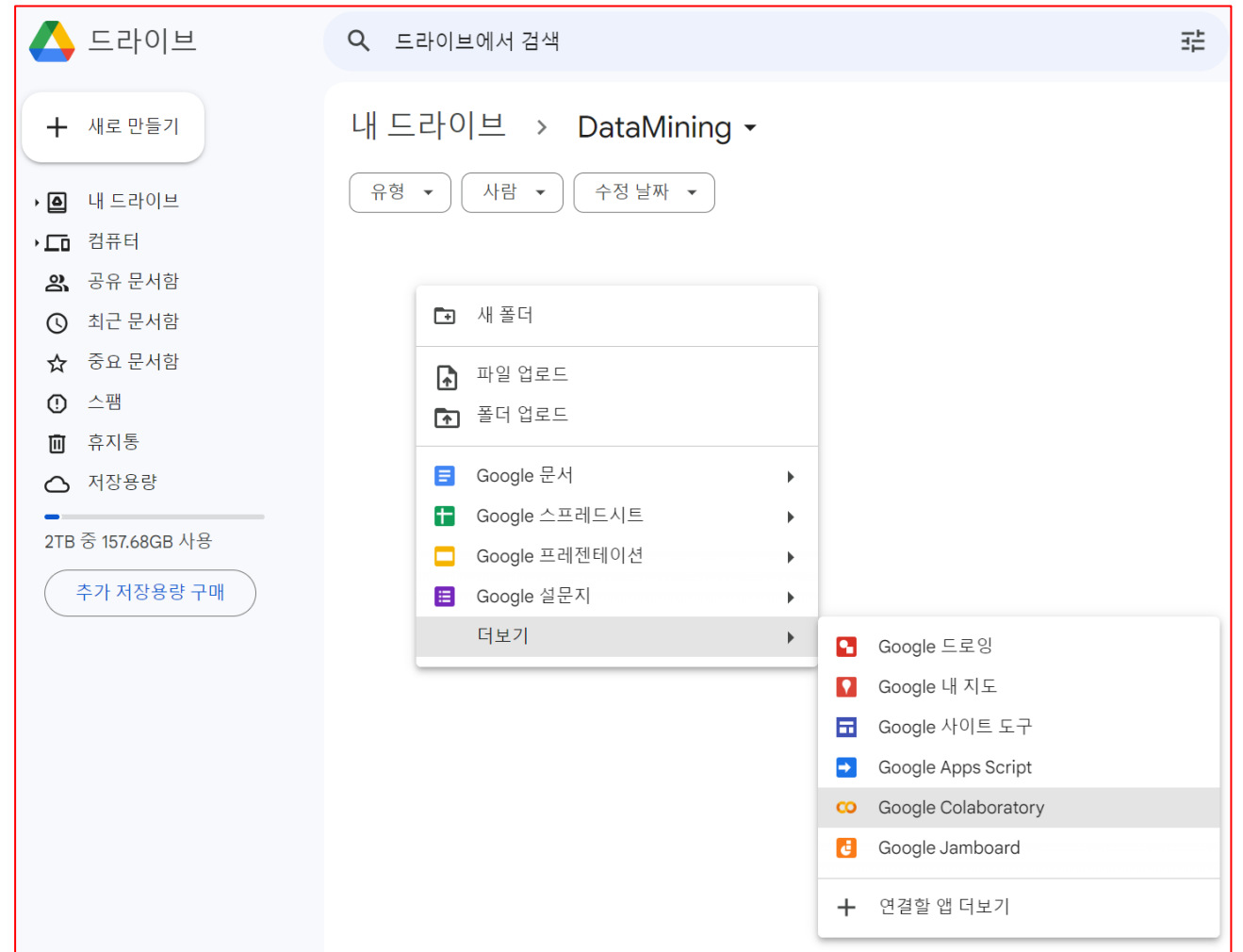
Googel Colab

- Google drive 내 Google Colab 작업용 폴더 만들기



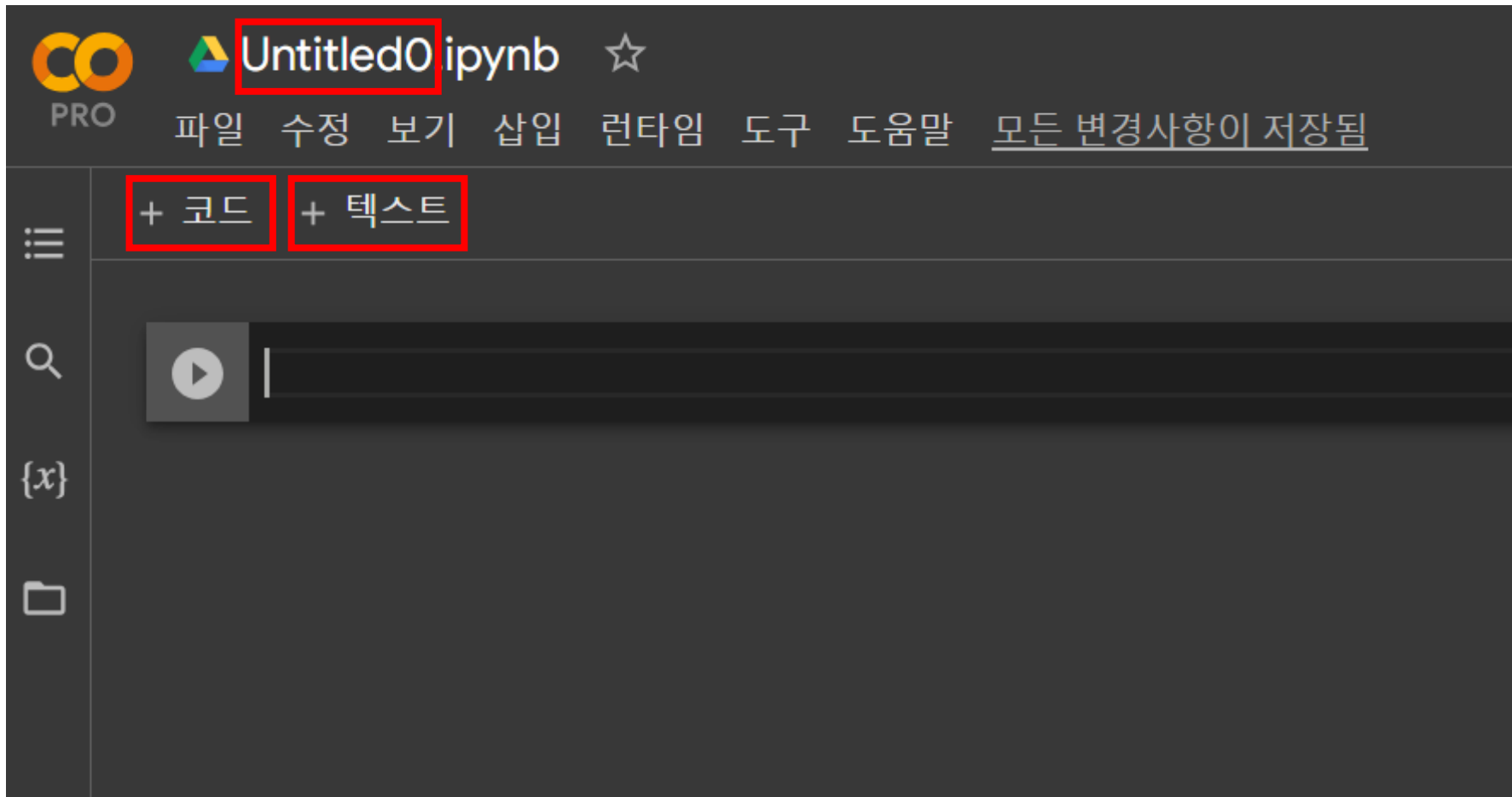
Googel Colab

- 마우스 오른쪽 클릭 후 Google Colaboratory 선택
- .ipynb 파일 : 노트북 (Notebook) 파일
- 주피터 (Jupyter) 노트북에서도 사용 가능



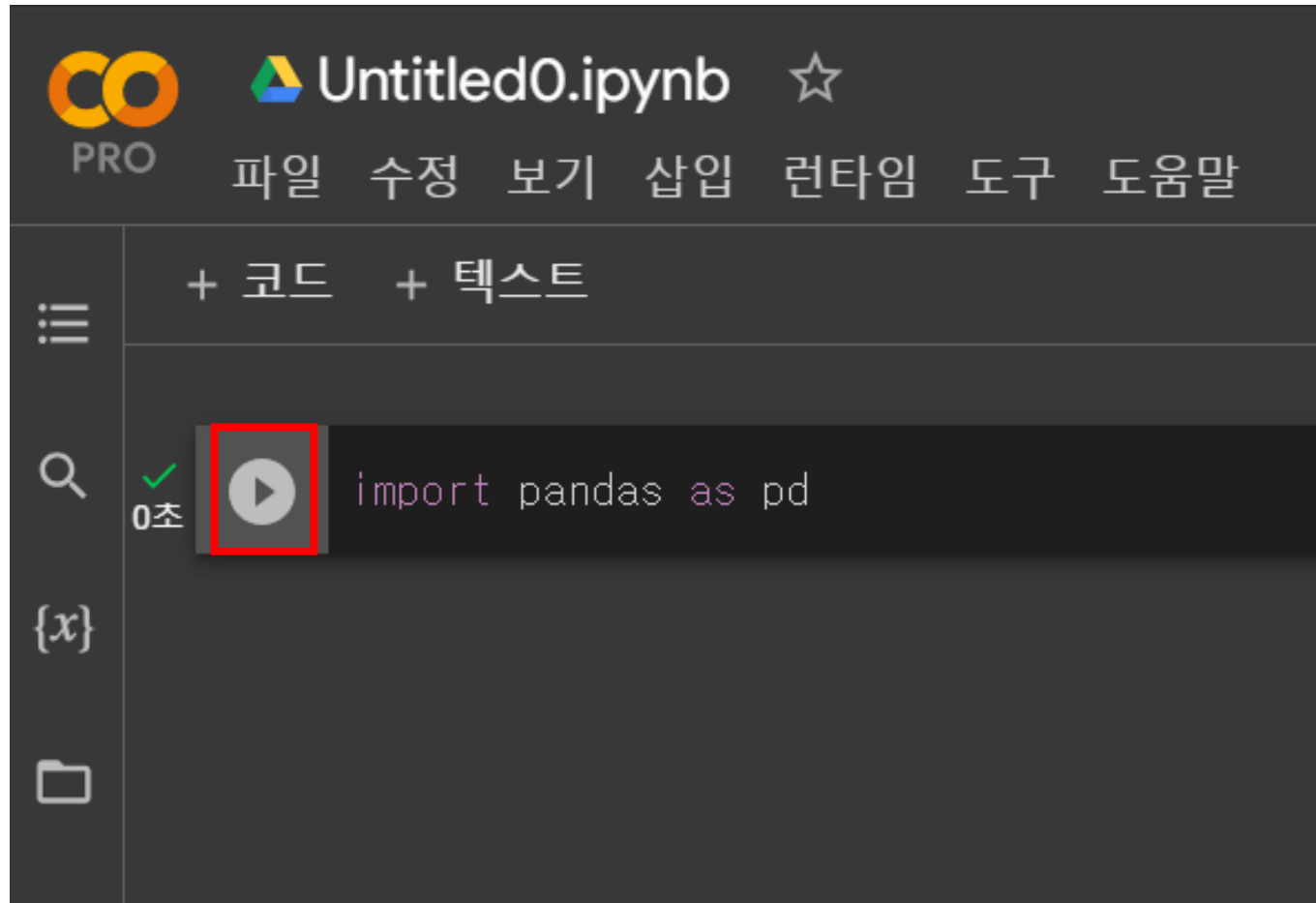
Googel Colab

- Google Colab 문서 만들기
 - 코드 : python code가 작동하는 박스
 - 텍스트 : Markdown 텍스트가 작동하는 박스 → 텍스트 작성용



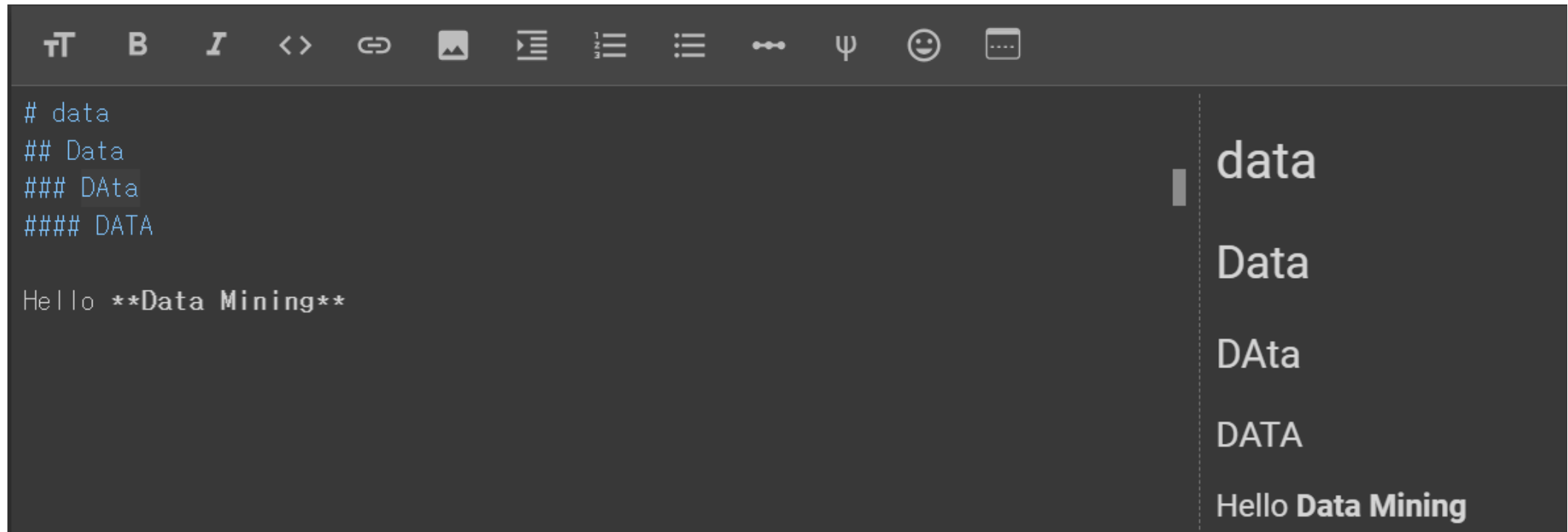
Googel Colab

- 코드 박스
 - 코드 박스안에 작성된 모든 코드는 “python code”라고 보면 됨
 - 재생 버튼 클릭 하거나 “ctrl + Enter”



Googel Colab

- 텍스트 박스
 - Markdown : <https://www.markdownguide.org/getting-started>



The image shows a screenshot of a Google Colab text box. At the top, there is a toolbar with various icons for text formatting, including bold (B), italic (I), code (<>), link, image, list, table, and others. Below the toolbar, the text box contains the following Markdown code:

```
# data
## Data
### DAta
#### DATA

Hello **Data Mining**
```

To the right of the text box, a preview of the rendered Markdown is shown. It displays the following text:

data

Data

DAta

DATA

Hello **Data Mining**

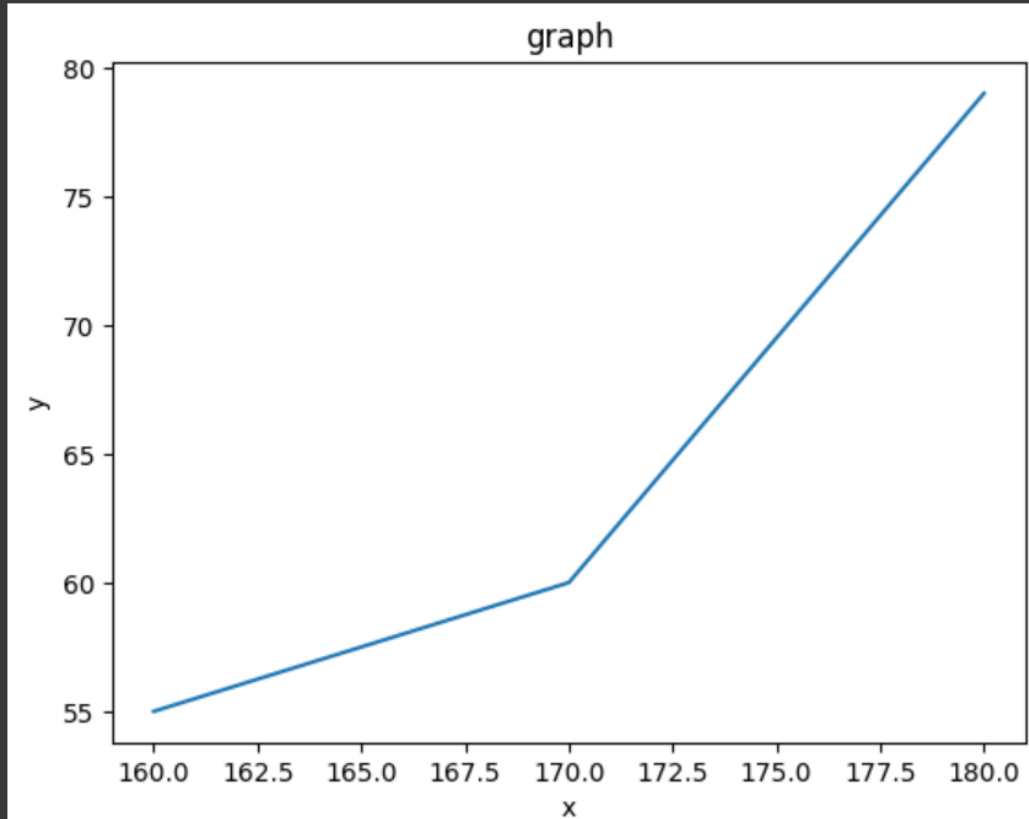
Googel Colab

- 단축키
 - 해당 셀 실행 후 커서는 해당 셀 : Ctrl + Enter
 - 해당 셀 실행 후 커서는 다음 셀 : Shift + Enter
 - 해당 셀 실행 후 커서는 다음에 셀 삽입 : Alt + Enter
- 코드 셀 위에 코드 삽입 : Ctrl/Cmd + M A
- 코드 셀 아래에 코드 삽입 : Ctrl/Cmd + M B
- 코드 셀 삭제 : Ctrl/Cmd + M D
- 코드 셀로 변경 : Ctrl/Cmd + M Y
- 텍스트 셀로 변경 : Ctrl/Cmd + M M
- 단축키 정보 : Ctrl + M H

Googel Colab

- 간단한 코드

```
✓ 1초 ▶ import matplotlib.pyplot as plt  
  
x = [160, 170, 180]  
y = [55, 60, 79]  
  
plt.plot(x,y)  
plt.title("graph")  
plt.xlabel("x")  
plt.ylabel("y")  
plt.show()
```



Googel Colab

- 파일 업로드/다운로드



```
import pandas as pd
df = pd.read_csv('/content/sample_data/california_housing_train.csv')
df.head()
```



	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-114.31	34.19	15.0	5612.0	1283.0	1015.0	472.0	1.4936	66900.0
1	-114.47	34.40	19.0	7650.0	1901.0	1129.0	463.0	1.8200	80100.0
2	-114.56	33.69	17.0	720.0	174.0	333.0	117.0	1.6509	85700.0
3	-114.57	33.64	14.0	1501.0	337.0	515.0	226.0	3.1917	73400.0
4	-114.57	33.57	20.0	1454.0	326.0	624.0	262.0	1.9250	65500.0



감사합니다

kimtwan21@dongduk.ac.kr

김 태 완