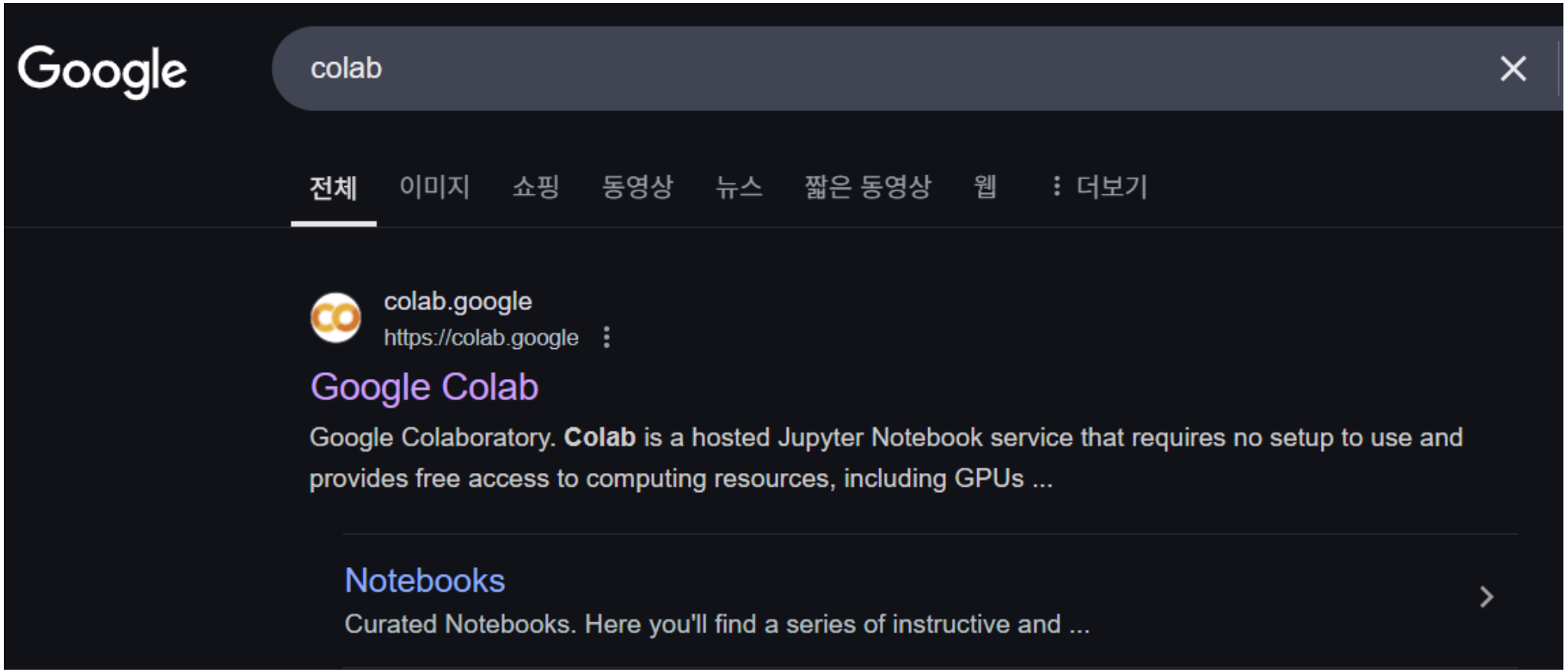
컴퓨팅 사고와 인공지능 실습

튜터 신연우
튜터 송진하
튜터 이채연

Google Colab Usage

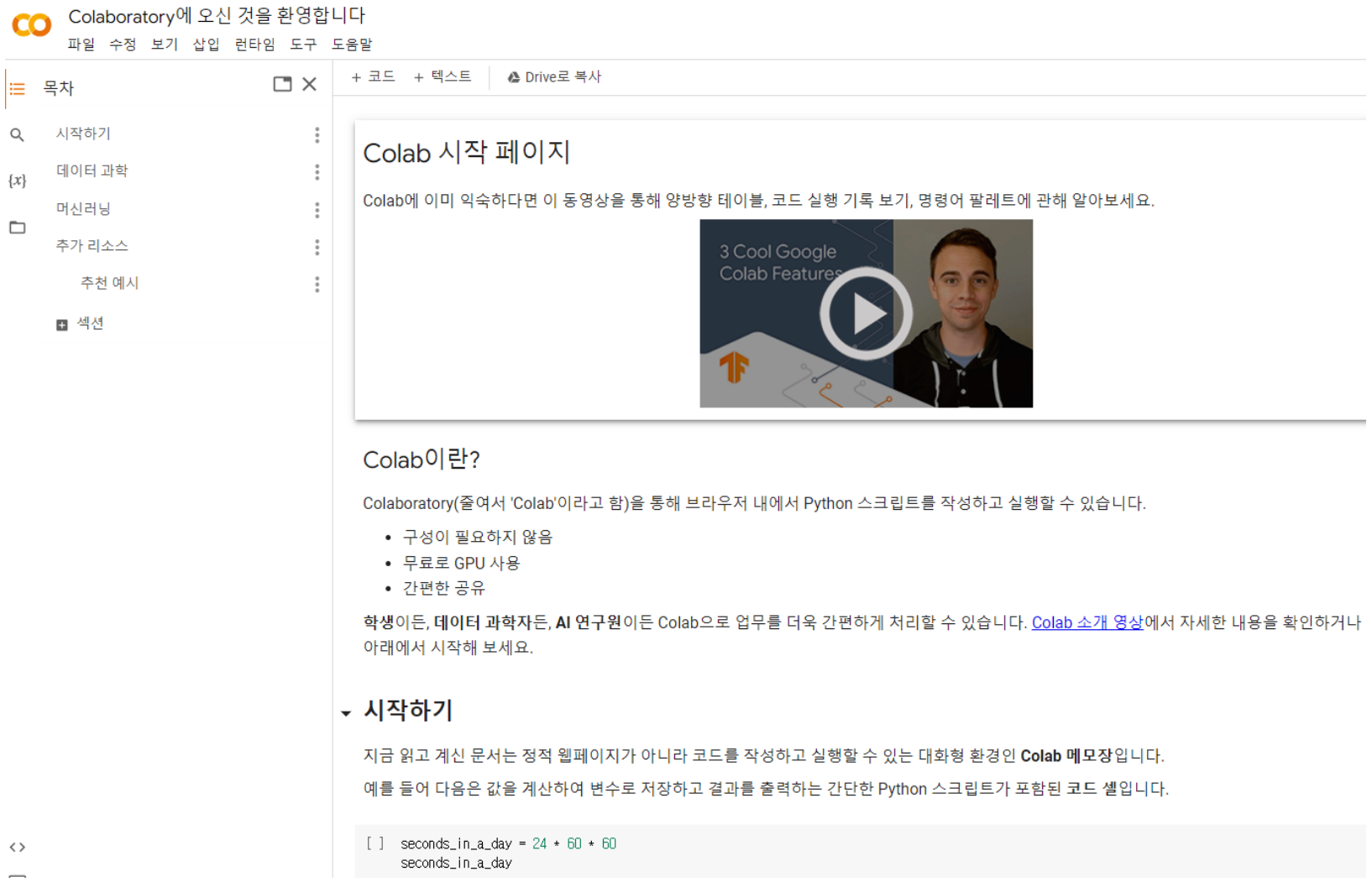
Google Colab Usage



코랩을 쓰는 가장 큰 이유는 프로그램을 따로 설치할 필요 없고,
구글 계정만 있으면 GPU까지 무료로 쓸 수 있다는 장점이 있습니다.

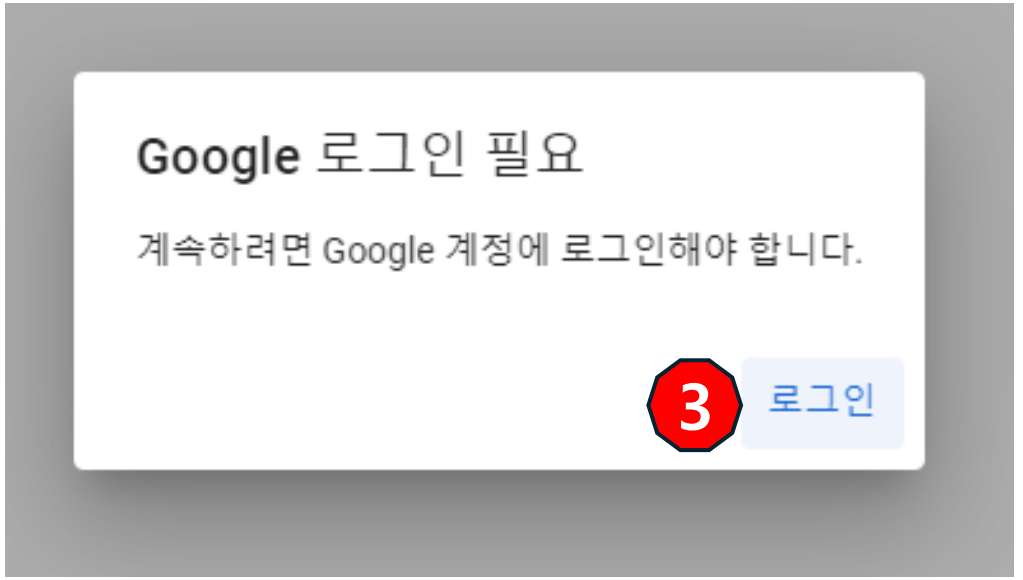
코랩을 실행하는 방법은
(1) 구글에 colab 혹은 코랩을 검색합니다.

Google Colab Usage



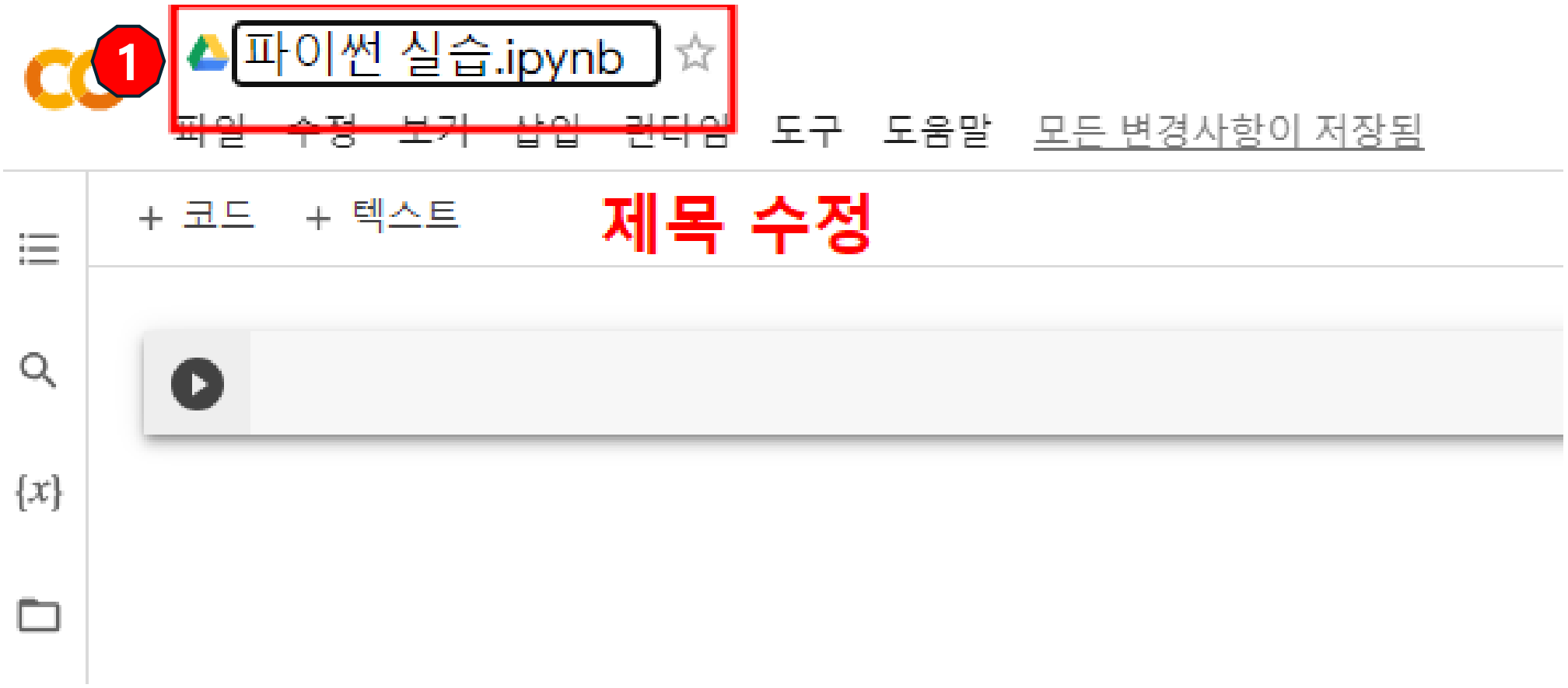
(2) 들어가면 이런 시작 화면이 뜹니다.

Google Colab Usage



(3) 새로운 노트를 만들어야 하는데 만들려면 구글 로그인이 필요합니다.
본인 구글 계정으로 로그인을 하면 됩니다.

Google Colab Usage

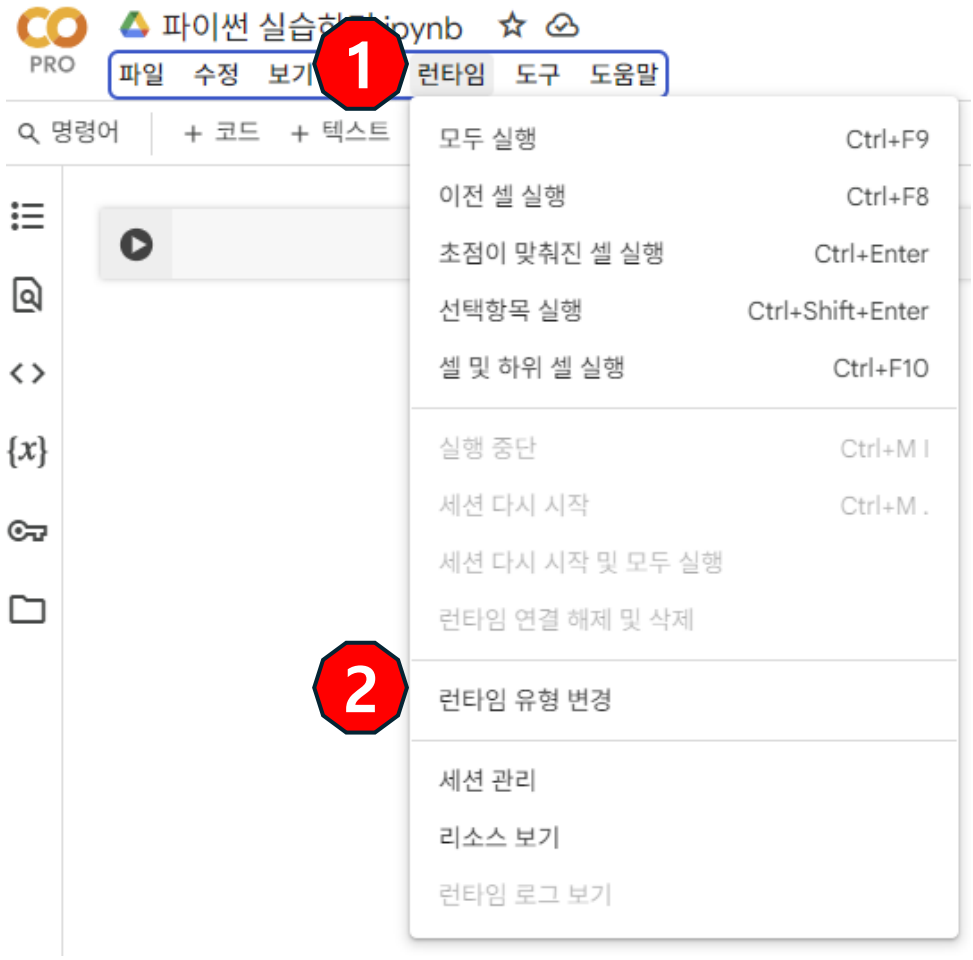


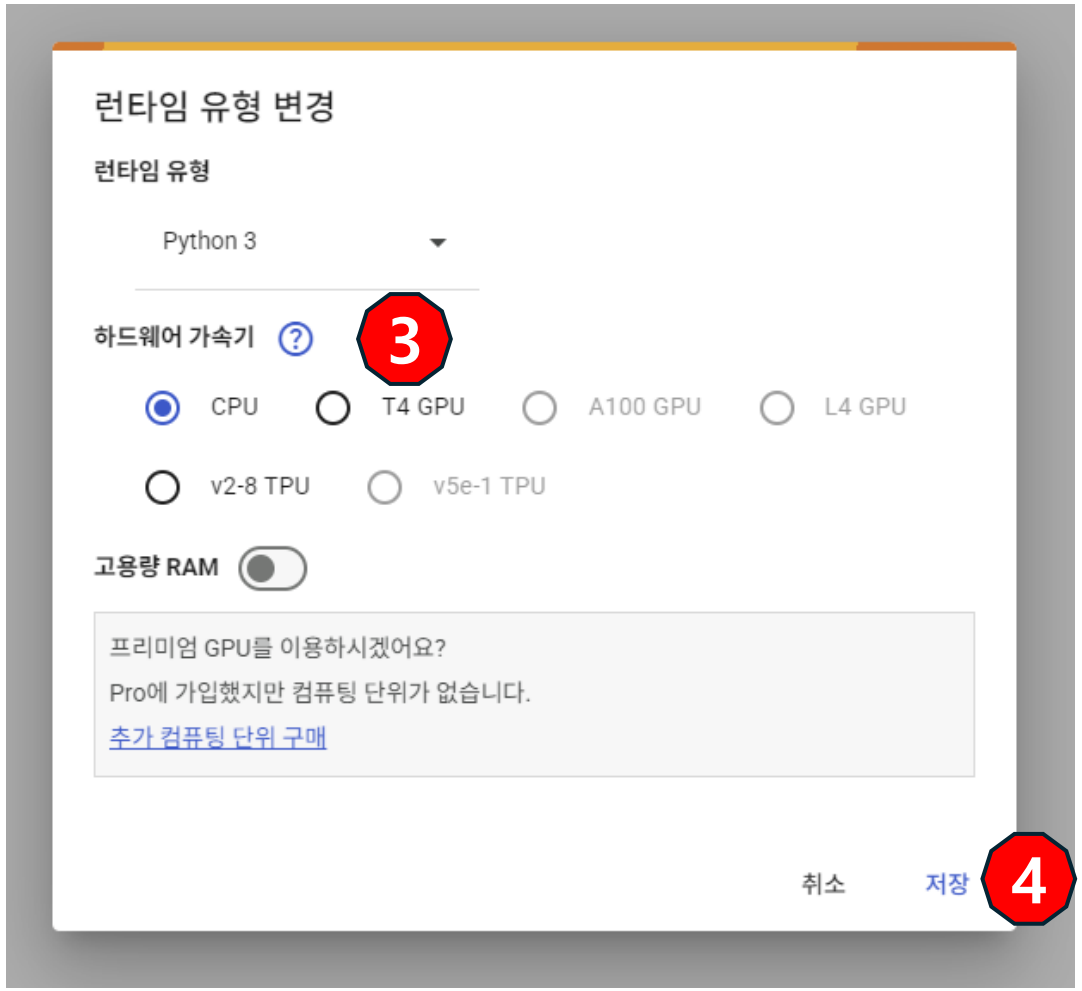
(3) 로그인 후 다시 새 노트를 만들면, 위에 사진처럼 뜨면 된 겁니다.
이때 다시 파일을 구분하기 위해 제목을 수정합니다.

Google Colab Usage

(3) 코랩은 Jupyter notebook과 거의 유사합니다. (1) 그래서 셀 창에 print 구문을 입력하고,
(2) 재생 표시를 누르거나 (shift + enter)를 누르게 되면 실행됩니다.

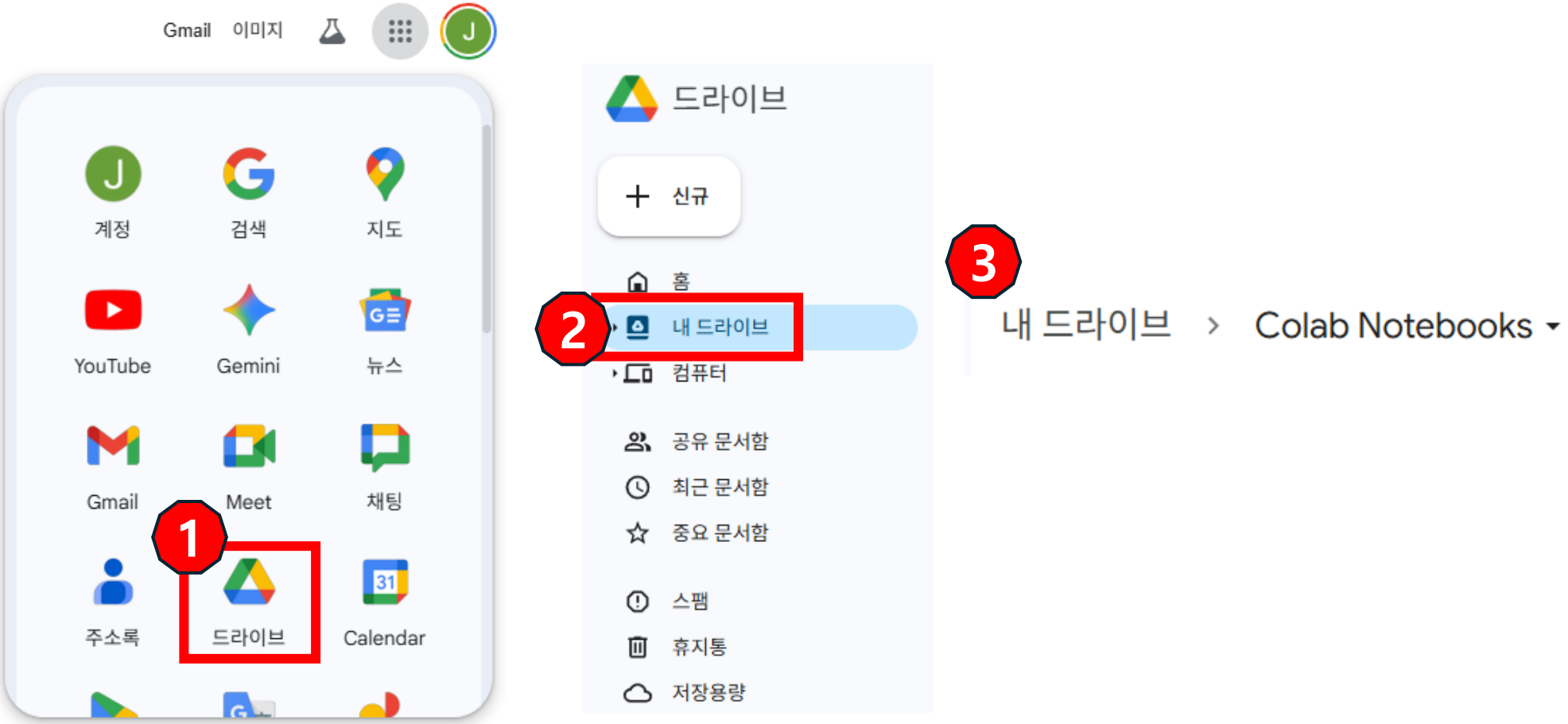
Google Colab Usage





(3) GPU는 딥러닝을 돌릴 때 필요한 자원으로 코랩에서는 무료로 지원해주고 있습니다.
설정은 위와 같이 (1) 런타임을 누르고 (2) 런타임 유형 변경을 누릅니다. 그러면 오른쪽처럼 뜨게 되는데,
(3) 이때 T4 GPU나 v2-8 GPU를 클릭하고 (4) 저장하면 GPU를 간단하게 설정할 수 있습니다.

Google Colab Usage



코랩에서 저장한 파일은 기본적으로 구글 드라이브에 저장됩니다.
(1) 구글 메인 페이지에서 구글 드라이브에 들어가고, (2) 왼쪽에 내 드라이브를 클릭합니다.
(3) 내 드라이브 내에 Colab Notebooks 폴더에 들어가면 저장한 파일이 존재합니다.

Python library 소개

라이브러리란?

- 자주 쓰이는 기능(수학 계산, 시각화, 데이터 처리 등)을 미리 구현해둔 코드 묶음
  - 예시: math, random, os, numpy, pandas, matplotlib

- 표준 라이브러리 vs 외부 라이브러리
  - 표준 라이브러리: Python에 기본 포함 (math, os, datetime 등)
  - 외부 라이브러리: 직접 설치 필요 (numpy, pandas, matplotlib, scikit-learn 등)

외부 라이브러리 설치 방법

설치

```
!pip install numpy pandas matplotlib
```

설치 확인

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
print(np.__version__)      # numpy 버전 확인
```

```
2.0.2
```

pandas와 matplotlib도 같은 방법으로 확인 가능

- 표준 라이브러리 vs 외부 라이브러리
  - 표준 라이브러리: Python에 기본 포함 (math, os, datetime 등)
  - 외부 라이브러리: 직접 설치 필요 (numpy, pandas, matplotlib, scikit-learn 등)

외부 라이브러리 설치 방법

자주 사용하는 라이브러리

라이브러리	주요 용도
numpy	수치 연산, 배열 처리
pandas	표 형태 데이터 분석
Matplotlib	시각화
Scipy	과학 계산, 통계
Scikit-learn	머신러닝 모델
Tensorflow / pytorch	딥러닝 프레임워크

Numpy

Numpy: 수치 데이터를 다루는 파이썬 라이브러리, 선형대수 계산 유리

[1]
✓ 0초

```
import numpy as np
```

약어(어떤 이름으로 불러올 건지)

np.sort()
np.add()
...

import numpy as np 에러가 날 경우
→ pip install numpy 실행 후 다시 import

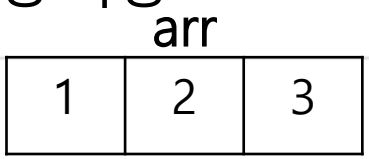
Numpy 배열 생성

array, zeros, ones, arrange 를 통해 배열 생성 가능

[2] ✓ 0초

```
arr = np.array([1, 2, 3])
print(f"arr의 타입: {type(arr)}")
```

```
arr의 타입: <class 'numpy.ndarray'>
```



[3] ✓ 0초

```
print(f"arr의 축의 개수: {arr.ndim}") # 행렬의 차원 출력
print(f"arr의 크기: {arr.shape}") # 행렬의 크기 출력
```

```
arr의 축의 개수: 1
arr의 크기: (3,)
```

```
# 배열의 모든 원소에 0을 삽입
zero_arr = np.zeros((2,3))
print(zero_arr)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

```
# 배열의 모든 원소에 1을 삽입
one_arr = np.ones((2,3))
print(one_arr)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

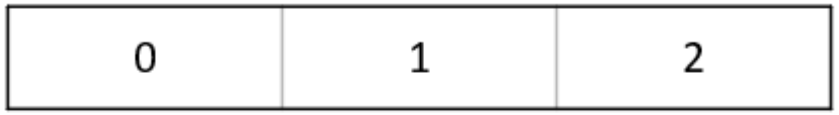
Numpy 배열 생성

array, zeros, ones, arrange 를 통해 배열 생성 가능

```
array_default1=np.arange(3)
```

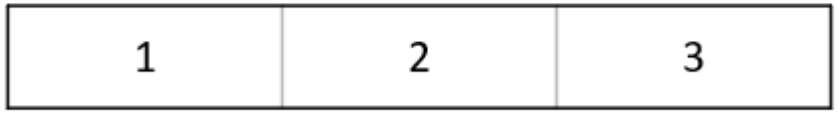
```
array_default1
```

```
array([0, 1, 2])
```



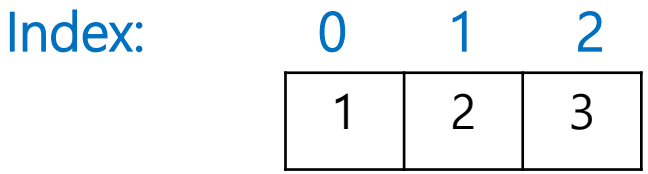

```
array_default1=np.arange(start=1,stop=4)
array_default1
```

```
array([1, 2, 3])
```



Numpy 다차원 배열

1차원 배열



[6]
✓ 0초

```
array1 = np.array([1,2,3])
```

[7]
✓ 0초

```
array1[2]
```

```
np.int64(3)
```

[8]
✓ 0초

```
print('dimension=', array1.ndim, ' ', 'shape=', array1.shape)
```

```
dimension= 1   shape= (3,)
```

Numpy 다차원 배열

2차원 배열

Index:	0	1	2
0	1	2	3
1	4	5	6

[9] ✓ 0초

```
array2 = np.array([[1,2,3], [4,5,6]])
```

이중 리스트 확인! → [[ ]]

[10] ✓ 0초

```
array2[1,2]
```

2번째 행, 3번째 열의 요소 = 6

```
np.int64(6)
```

[11] ✓ 0초

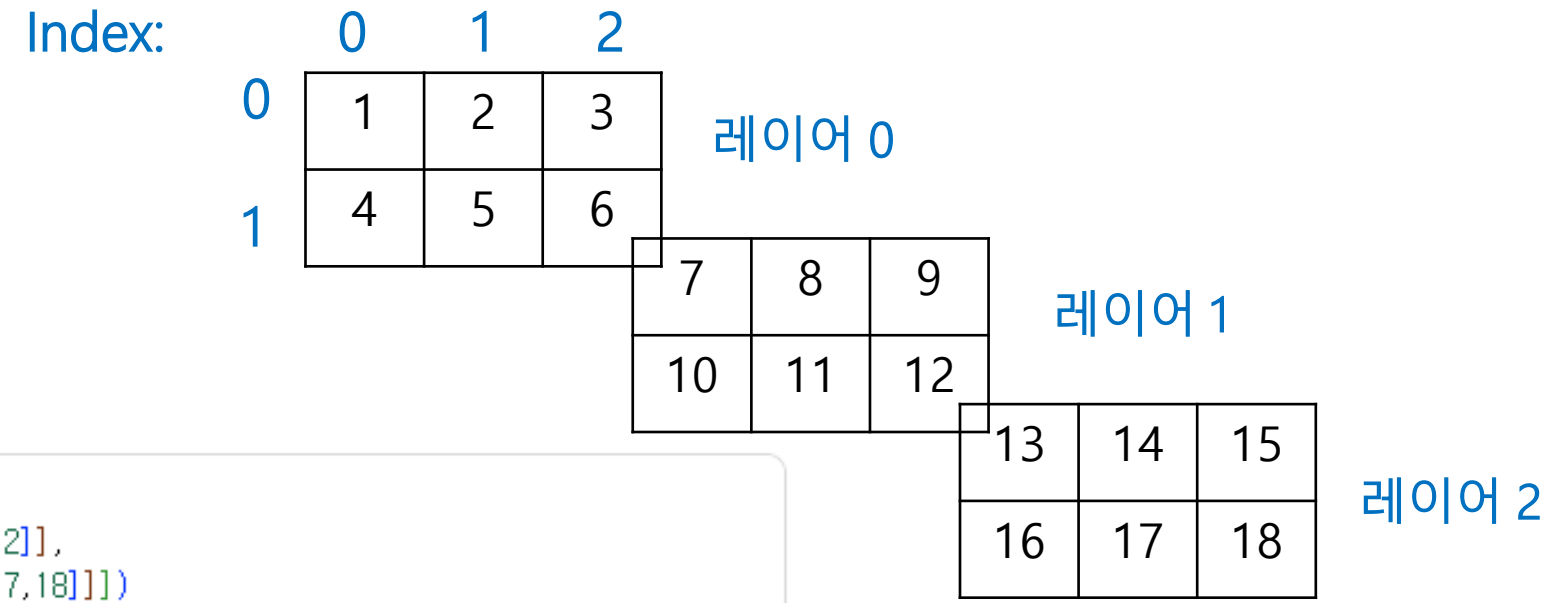
```
print('dimension=', array2.ndim, ' ', 'shape=', array2.shape)
```

```
dimension= 2   shape= (2, 3)
```

2행 3열

Numpy 다차원 배열

3차원 배열



[14] ✓ 0초

```
array3 = np.array([[[1,2,3], [4,5,6]],
                   [[7,8,9], [10,11,12]],
                   [[13,14,15], [16,17,18]]])
```

[15] ✓ 0초

```
array3[1,1,2]
```

레이어 1, 행 1, 열 2 → 두번째 레이어, 두번째 행, 세번째 열 = 12

```
np.int64(12)
```

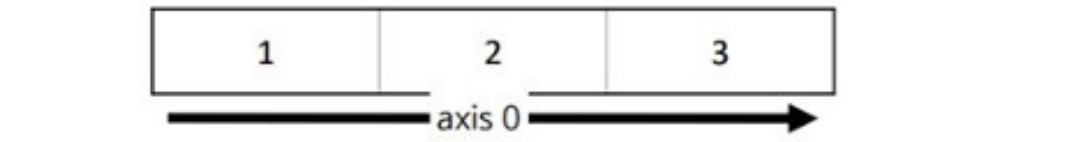
[16] ✓ 0초

```
print('dimension=', array3.ndim, ' ', 'shape=', array3.shape)
```

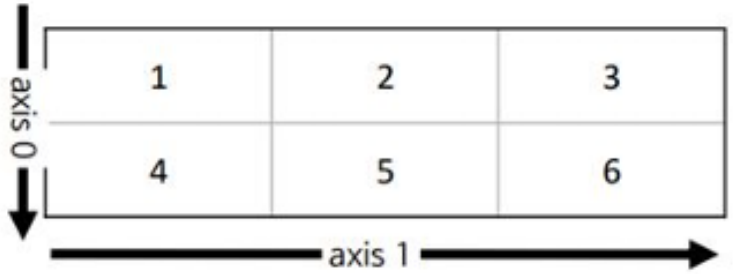
```
dimension= 3   shape= (3, 2, 3)
```

레이어 수, 행 수, 열 수

Numpy 다차원 배열의 축

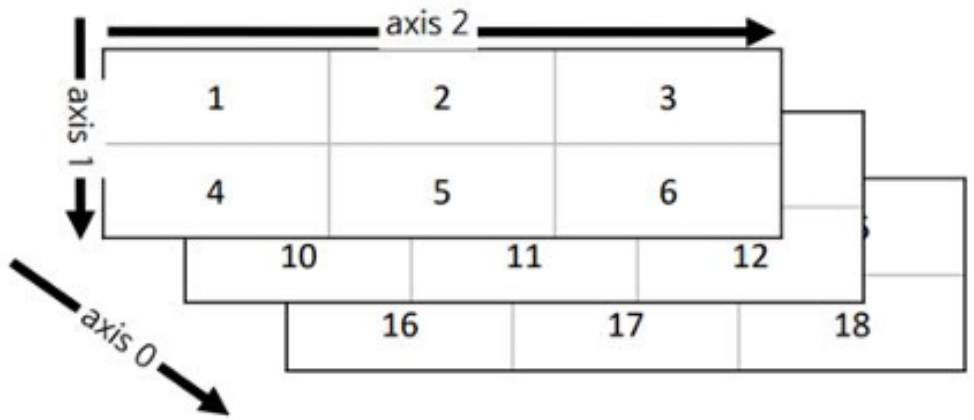


shape: (3,)



shape: (2, 3)

앞에서부터
axis 0, axis 1,
axis 2, …



shape: (3, 2, 3)

Numpy 정렬하기

np.sort(대상 배열, 기준 축)

표의 빈칸을 채워보세요!!

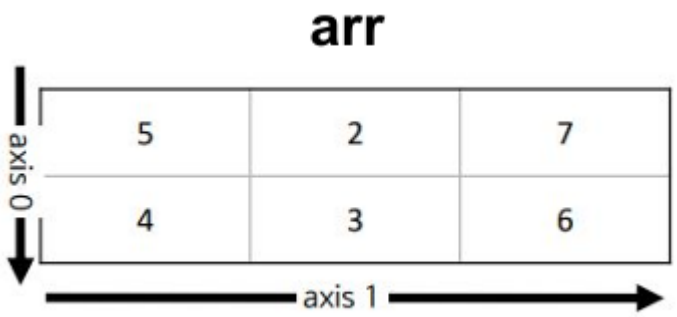
```
arr_sort0 = np.sort(arr1, axis = 0)
arr_sort0
```

Output


```
arr_sort1 = np.sort(arr1, axis = 1)
arr_sort1
```

Output

Numpy 정렬하기



np.sort(대상 배열, 기준 축)

[20]
✓ 0초

```
arr_sort0 = np.sort(arr1, axis = 0)
arr_sort0
```

```
array([[4, 2, 6],
       [5, 3, 7]])
```

axis 0 기준으로 정렬
(행 기준 정렬)

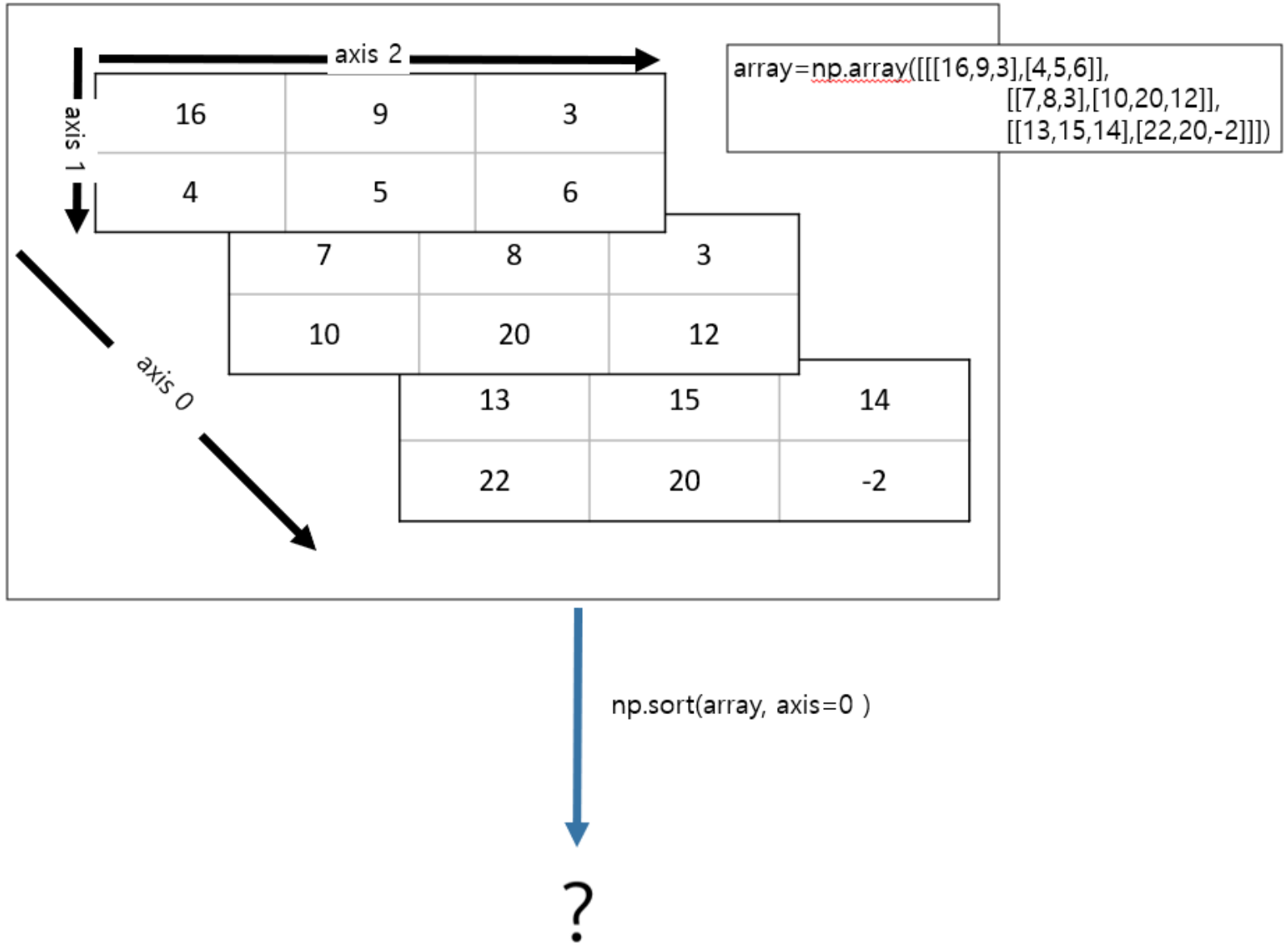
[21]
✓ 0초

```
arr_sort1 = np.sort(arr1, axis = 1)
arr_sort1
```

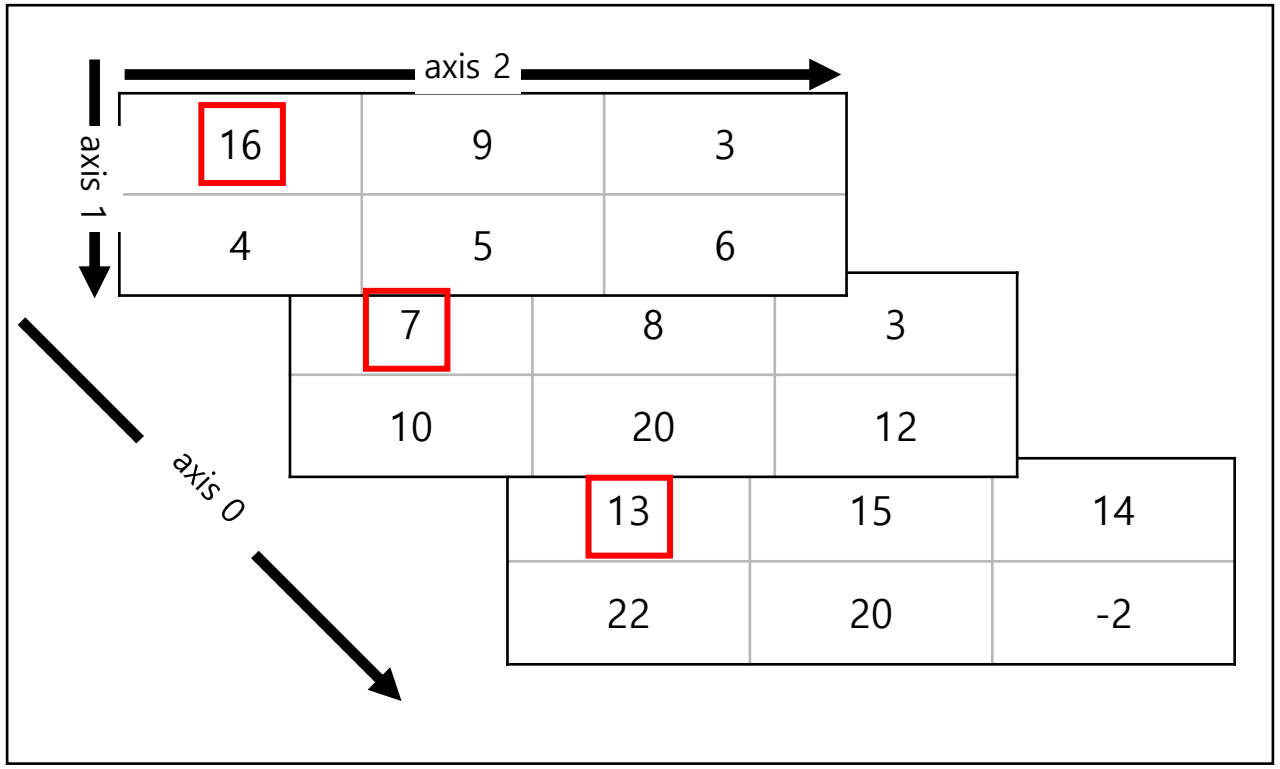
```
array([[2, 5, 7],
       [3, 4, 6]])
```

axis 1 기준으로 정렬
(열 기준 정렬)

Numpy 정렬하기


axis 2
axis 1
axis 0
16
9
3
4
5
6
7
8
3
10
20
12
13
15
14
22
20
-2
array=np.array([[[16,9,3],[4,5,6]],
[[7,8,3],[10,20,12]],
[[13,15,14],[22,20,-2]]])
np.sort(array, axis=0)
?

Numpy 정렬하기



[21]
0초

```
import numpy as np

array = np.array([
    [[16, 9, 3], [4, 5, 6]],
    [[7, 8, 3], [10, 20, 12]],
    [[13, 15, 14], [22, 20, -2]]
])
```

[28]
0초

```
np.sort(array, axis=0)
```

```
array([[[ 7,  8,  3],
        [ 4,  5, -2]],

       [[13,  9,  3],
        [10, 20,  6]],

       [[16, 15, 14],
        [22, 20, 12]]])
```

Numpy Add

두 배열을 요소별(element-wise)로 더할 때 사용

np.add(대상 배열1, 대상 배열2)

[22] ✓ 0초

```
arr1 = np.array([[5,2,7], [4,3,6]])
arr1
```

```
array([[5, 2, 7],
       [4, 3, 6]])
```

[23] ✓ 0초

```
arr2 = np.array([[1,2,3], [4,5,6]])
arr2
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

[24] ✓ 0초

```
np.add(arr1, arr2)
```

```
array([[ 6,  4, 10],
       [ 8,  8, 12]])
```

[5] ✓ 0초

```
arr1 + arr2
```

```
array([[ 6,  4, 10],
       [ 8,  8, 12]])
```

Numpy Subtract

두 배열을 요소별(element-wise)로 뺄셈할 때 사용

np.subtract(대상 배열1, 대상 배열2)

[22] ✓ 0초

```
arr1 = np.array([[5,2,7], [4,3,6]])
arr1
```

```
array([[5, 2, 7],
       [4, 3, 6]])
```

[23] ✓ 0초

```
arr2 = np.array([[1,2,3], [4,5,6]])
arr2
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

[4] ✓ 0초

```
np.subtract(arr1, arr2)
```

```
array([[ 4,  0,  4],
       [ 0, -2,  0]])
```

[6] ✓ 0초

```
arr1 - arr2
```

```
array([[ 4,  0,  4],
       [ 0, -2,  0]])
```

Numpy Arithmetic Operations

연산	예시	설명
np.multiply(a, b)	a * b	두 배열의 원소별 곱
np.divide(a, b)	a / b	두 배열의 원소별 나눗셈
np.power(a, b)	a ** b	거듭제곱
np.mod(a, b)	a % b	나머지

Numpy dot product

두 배열의 내적을 구할 때 사용

np.dot(대상 배열1, 대상 배열2)

[20]
✓ 0초

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

print(np.dot(a, b))  # 1*4 + 2*5 + 3*6 = 32
```

```
32
```

[18]
✓ 0초

```
A = np.array([[1, 2],
              [3, 4]])
B = np.array([[5, 6],
              [7, 8]])

print(np.dot(A, B))
```

```
[[19 22]
 [43 50]]
```

Numpy reshape

배열의 형태를 바꿀 때 사용

0	1
2	3
4	5

```
array_default1=np.arange(6)
array_default1.reshape(3,2)
```

```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

```
array1=np.zeros((3,2),dtype='int32')
```

```
array2=array1.reshape(2,3)
```

```
array1
```

```
array([[0, 0],
       [0, 0],
       [0, 0]], dtype=int32)
```

```
array2
```

```
array([[0, 0, 0],
       [0, 0, 0]], dtype=int32)
```

```
array3=array1.reshape(-1,3)
array3
```

```
array([[0, 0, 0],
       [0, 0, 0]], dtype=int32)
```

np.reshape()에 **-1**을 넣으면 전체 원소 개수를 유지하면서 남은 차원 크기를 NumPy가 자동 계산

Numpy statistical functions

np.mean(): 평균
np.sum(): 합계
np.prod() 곱

1	2	3	4
5	6	7	8
9	10	11	12

[16]
✓ 0초

```
arr = np.array([[1, 2, 3, 4],
                [5, 6, 7, 8],
                [9, 10, 11, 12]])

print("전체 평균:", np.mean(arr))              # 모든 원소의 평균
print("행 기준 평균(axis=1):", np.mean(arr, axis=1))  # 각 행의 평균
print("열 기준 평균(axis=0):", np.mean(arr, axis=0))  # 각 열의 평균
```

```
전체 평균: 6.5
행 기준 평균(axis=1): [ 2.5  6.5 10.5]
열 기준 평균(axis=0): [5. 6. 7. 8.]
```

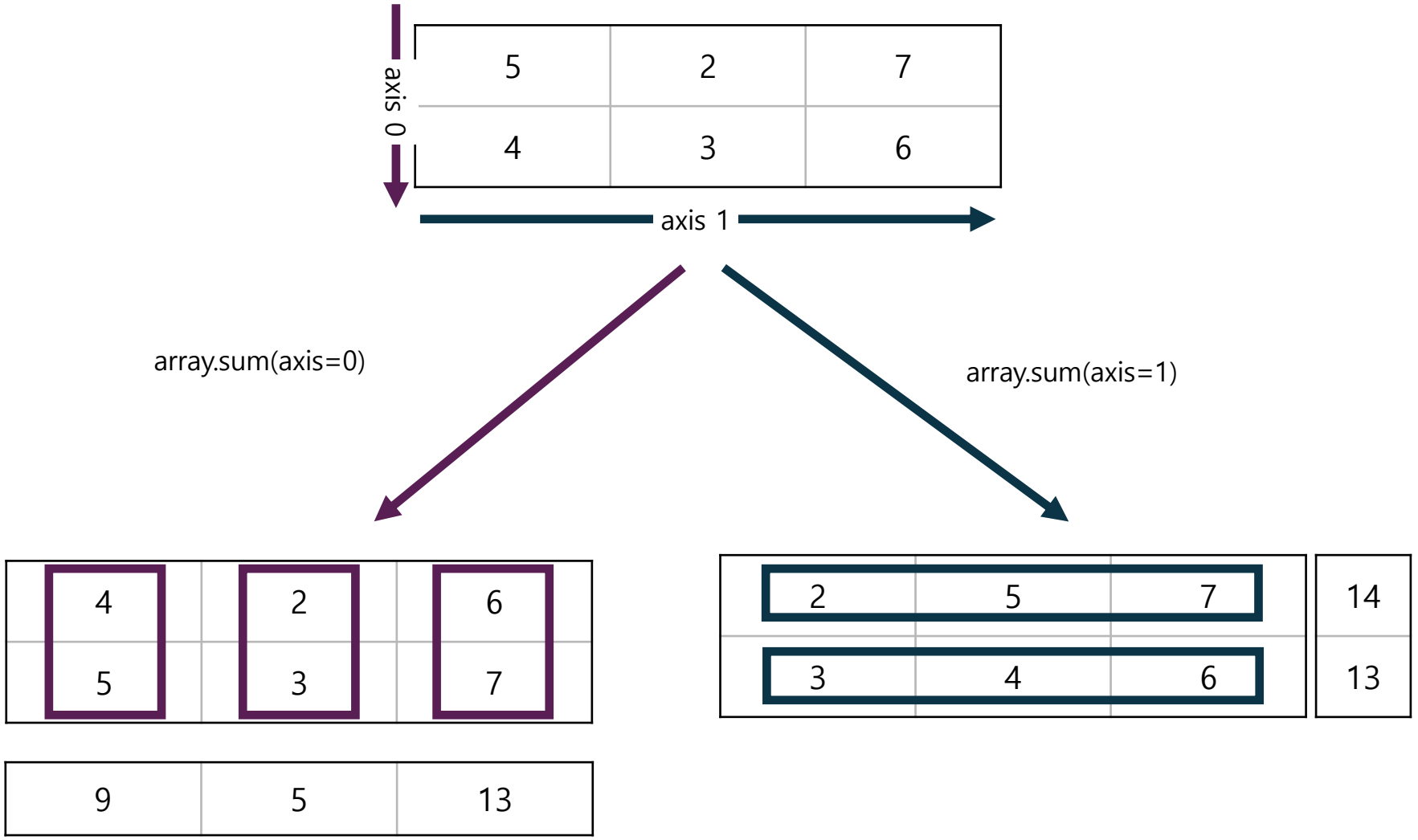
[17]
✓ 0초

```
# 합계와 곱
print("전체 합:", np.sum(arr))
print("각 열의 합(axis=0):", np.sum(arr, axis=0))
print("각 행의 합(axis=1):", np.sum(arr, axis=1))
print("전체 곱:", np.prod(arr))
```

```
전체 합: 78
각 열의 합(axis=0): [15 18 21 24]
각 행의 합(axis=1): [10 26 42]
전체 곱: 479001600
```

axis 옵션을 통해 축을 기준으로 확인 가능
axis=0 → 행을 기준으로 연산
axis=1 → 열을 기준으로 연산

Numpy statistical functions



Practice

예제 3-4 **선형결합**

두 벡터 $\boldsymbol{a}=(1, 0, 3)$, $\boldsymbol{b}=(2, 1, 2)$의 선형결합 $2\boldsymbol{a}+3\boldsymbol{b}$를 구하라.

Practice

예제 3-12 **내적**

$\mathbb{R}^4$의 세 벡터 $\boldsymbol{x}=(1,\ 2,\ 3,\ -1)$, $\boldsymbol{y}=(0,\ 1,\ 2,\ 1)$, $\boldsymbol{z}=(2023,\ 1,\ -1,\ 1)$에 대해 $\boldsymbol{x}\cdot\boldsymbol{y}$와 $\boldsymbol{y}\cdot\boldsymbol{z}$의 값을 구하라.

Practice

오른쪽과 같은 배열을 numpy를 이용해

11	10	3	4
7	1	2	9
6	8	5	12

1. 각 행의 합
2. 각 열의 평균
3. 배열의 모양은 유지한 채, 1~12 크기 순으로 정렬

세 가지 기능을 구현하기