
PYTHON

Python

1991년 귀도 반 로섬이 발표한 고급 프로그래밍 언어

언어 이름은 귀도가 좋아하는 코미디 Monty Python`s Flying Circus에서 따옴

인터프리식 언어, 객체지향적, 동적 타이핑 대화형 언어

비영리 파이썬 소프트웨어 재단이 관리하고 있으며, 사실상 표준은 C언어로 구현된 Cython

귀도 반 로섬

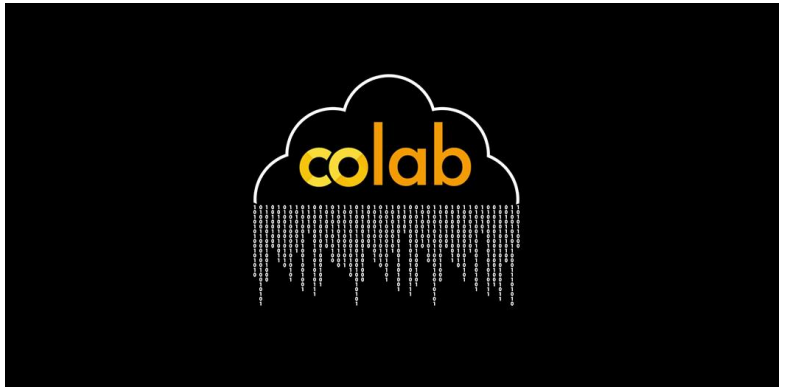
*인터프리식 언어: 코드를 한줄 한줄 읽어가며 명령을 처리

**객체지향적: 파이썬은 Object(객체)로 이루어져 있으며, 객체 지향 프로그래밍을 추구하고 있다.

OOP(Object Oriented Programming, 객체 지향 프로그래밍): 프로그래밍에 필요한 데이터를 추상화하여 상태와 행위를 가진 객체를 만들고, 객체들 간의 유기적 상호작용을 통해 로직을 구성하는 방법

***동적 타이핑 대화형 언어: 파이썬은 변수를 만들 때 변수의 타입을 먼저 지정해주지 않습니다. 그리고, 프로그램 자체가 영어 친화적으로 대화하듯 되어 되어 있습니다.

colab



코랩(colab)은 구글에서 제공하고 있는 클라우드 주피터 노트북입니다.

Jupyter Notebook이 무엇일까요?

주피터 노트북

노트북이라는 말 그대로 공책을 사용하듯 코드를 작성하면서 설명도 넣을 수 있습니다.

코랩을 사용하는 이유?

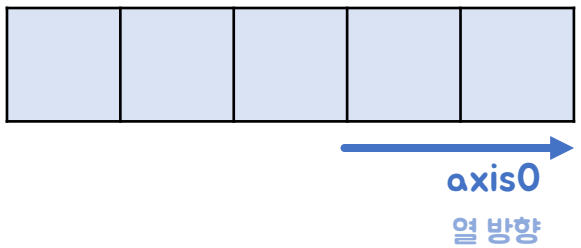
코랩은 클라우드 서비스이기에 내가 사용하는 PC의 CPU나 GPU가 낮아도 괜찮다라는 부분이 이점입니다.

로컬에 프로그램을 따로 설치할 필요가 없으며, 머신러닝과 같이 GPU를 많이 사용할 때 클라우드 서비스를 사용하는데

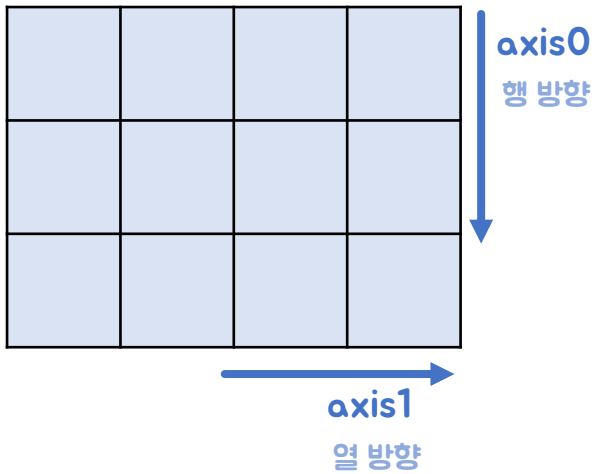
코랩은 이러한 점에서 인공지능을 배우기에 이점이 있습니다.


NUMPY

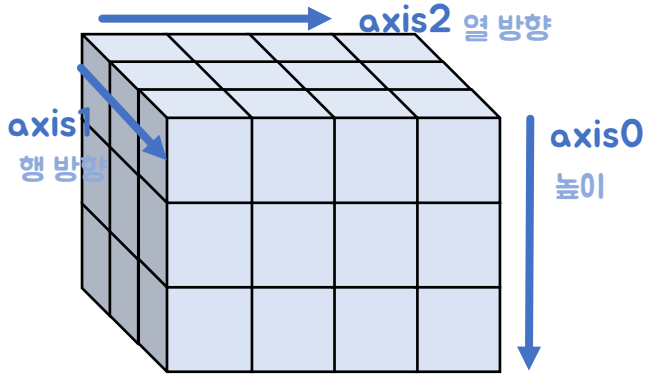
1차원 배열



2차원 배열



3차원 배열



Numpy에서의 차원을 축이라고 합니다.

Numpy

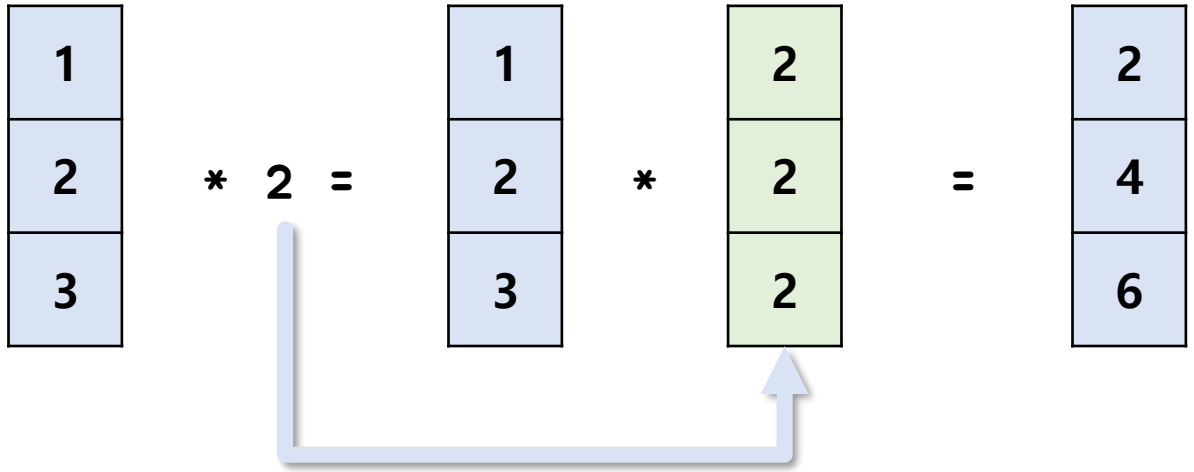
브로드캐스트

일반적으로 배열을 연산할 때 배열들의 shape가 맞아야 연산이 가능할 것이라고 볼 것입니다.

그런데 Numpy에서 이 두 배열의 shape가 다르더라도 연산을 자동으로 맞춰서 해주는 기능이 있습니다.

이것을 브로드캐스트라고 합니다. 예를 들어서 스칼라와 배열 사이의 연산이 가능하죠.

(단, shape가 다 달라도 무조건 가능하진 않습니다. Shape가 같거나 차원 중 값이 1인 것이 존재하면 됩니다.)



Numpy

dot 함수 심화

Array A N차원 $A_1, A_2, A_3, \dots, A_{n-1}, A_n$

Array B M차원 $B_1, B_2, B_3, \dots, B_{m-1}, B_m$

(N, M >= 2)

A_1과 A_2의 차이는 1이 아닙니다.

해당 값들은 미지수입니다.

dot함수를 사용하기 위해서는 n과 m-1이 같아야 합니다.

정확히는 배열 A.dot(배열 B)을 하였으면 A_n 과 B_{m-1} 은 같아야 합니다.

https://numpy.org/doc/stable/reference/generated/numpy.dot.html

- If *a* is an N-D array and *b* is an M-D array (where M>=2), it is a sum product over the last axis of *a* and the second-to-last axis of *b*:

```
dot(a, b)[i,j,k,m] = sum(a[i,j,:] * b[k,:,m])
```

Numpy 정식 문서에 보면 위와 같이 기재되어 있기 때문입니다.

선형대수에 내적이 아닌, 프로그래밍에 정의된 방법에 따라 진행됨을 유의하여 주세요.

Array A N차원 $A_1, A_2, A_3, \dots, A_{n-1}, A_n$

Array B M차원 $B_1, B_2, B_3, \dots, B_{m-1}, B_m$

(N, M >= 2)

A_1과 A_2의 차이는 1이 아닙니다.

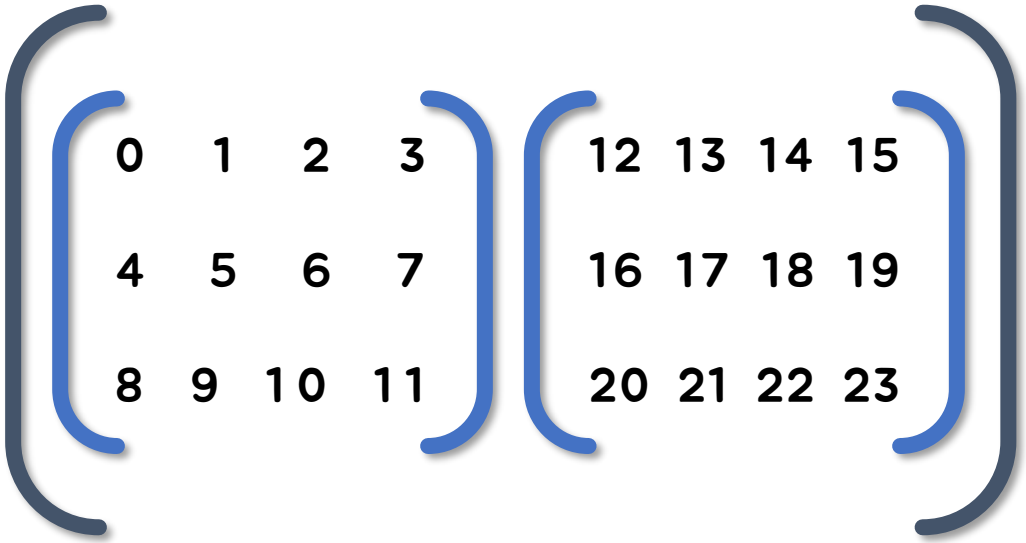
해당 값들은 미지수입니다.

dot 함수의 결과는

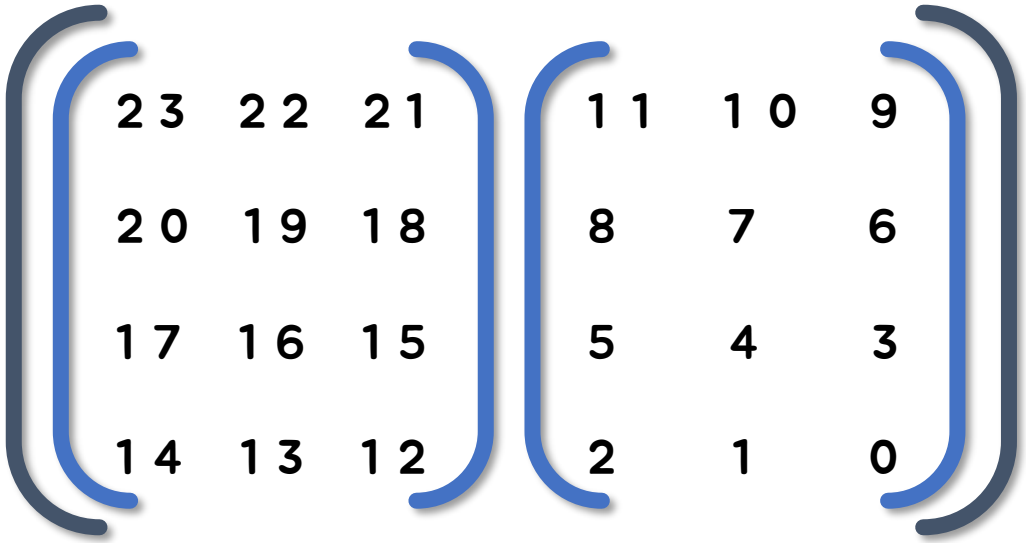
$A_1, A_2, A_3, \dots, A_{n-1}, B_1, B_2, B_3, \dots, B_{m-2}, B_m$ Shape을 가진 배열입니다.

dot 함수

배열 A Shape = (2, 3, 4)



배열 B Shape = (2, 4, 3)



이러한 배열A와 배열B가 있다고 하겠습니다.

해당 배열A.dot(배열B)의 결과를 살펴보도록 하겠습니다.

dot 함수

배열 A Shape = (2, 3, 4)

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

배열 B Shape = (2, 4, 3)

```
[[[23 22 21]
  [20 19 18]
  [17 16 15]
  [14 13 12]]

 [[11 10  9]
  [ 8  7  6]
  [ 5  4  3]
  [ 2  1  0]]]
```

Result 배열

$A_{11}B_{11}$ = (0 * 23) + (1 * 20) + (2 * 17) + (3 * 14)

= 96

```
[[[96  ]]]
```

dot 함수

배열 A Shape = (2, 3, 4)

$$\begin{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} & \begin{bmatrix} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix} \end{bmatrix}$$

배열 B Shape = (2, 4, 3)

$$\begin{bmatrix} \begin{bmatrix} 23 & 22 & 21 \\ 20 & 19 & 18 \\ 17 & 16 & 15 \\ 14 & 13 & 12 \end{bmatrix} & \begin{bmatrix} 11 & 10 & 9 \\ 8 & 7 & 6 \\ 5 & 4 & 3 \\ 2 & 1 & 0 \end{bmatrix} \end{bmatrix}$$

Result 배열

$A_{11}B_{12}$ = (0 * 22) + (1 * 19) + (2 * 16) + (3 * 13)

= 90

$$\begin{bmatrix} 96 & 90 \end{bmatrix}$$

dot 함수

배열 A Shape = (2, 3, 4)

$$\begin{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} & \begin{bmatrix} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix} \end{bmatrix}$$

배열 B Shape = (2, 4, 3)

$$\begin{bmatrix} \begin{bmatrix} 23 & 22 & 21 \\ 20 & 19 & 18 \\ 17 & 16 & 15 \\ 14 & 13 & 12 \end{bmatrix} & \begin{bmatrix} 11 & 10 & 9 \\ 8 & 7 & 6 \\ 5 & 4 & 3 \\ 2 & 1 & 0 \end{bmatrix} \end{bmatrix}$$

Result 배열

$A_{11}B_{13}$ = (0 * 21) + (1 * 18) + (2 * 15) + (3 * 12)

= 84

$$\begin{bmatrix} 96 & 90 & 84 \end{bmatrix}$$

dot 함수

배열 A Shape = (2, 3, 4)

$$\begin{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} & \begin{bmatrix} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix} \end{bmatrix}$$

배열 B Shape = (2, 4, 3)

$$\begin{bmatrix} \begin{bmatrix} 23 & 22 & 21 \\ 20 & 19 & 18 \\ 17 & 16 & 15 \\ 14 & 13 & 12 \end{bmatrix} & \begin{bmatrix} 11 & 10 & 9 \\ 8 & 7 & 6 \\ 5 & 4 & 3 \\ 2 & 1 & 0 \end{bmatrix} \end{bmatrix}$$

Result 배열

$A_{11}B_{21}$ = (0 * 11) + (1 * 8) + (2 * 5) + (3 * 2)

= 24

$$\begin{bmatrix} 96 & 90 & 84 \\ 24 & & \end{bmatrix}$$

dot 함수

배열 A Shape = (2, 3, 4)

$$\begin{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} & \begin{bmatrix} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix} \end{bmatrix}$$

배열 B Shape = (2, 4, 3)

$$\begin{bmatrix} \begin{bmatrix} 23 & 22 & 21 \\ 20 & 19 & 18 \\ 17 & 16 & 15 \\ 14 & 13 & 12 \end{bmatrix} & \begin{bmatrix} 11 & 10 & 9 \\ 8 & 7 & 6 \\ 5 & 4 & 3 \\ 2 & 1 & 0 \end{bmatrix} \end{bmatrix}$$

Result 배열

$A_{11}B_{22}$ = (0 * 10) + (1 * 7) + (2 * 4) + (3 * 1)

= 18

$$\begin{bmatrix} 96 & 90 & 84 \\ 24 & 18 & \end{bmatrix}$$

dot 함수

배열 A Shape = (2, 3, 4)

$$\begin{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} & \begin{bmatrix} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix} \end{bmatrix}$$

배열 B Shape = (2, 4, 3)

$$\begin{bmatrix} \begin{bmatrix} 23 & 22 & 21 \\ 20 & 19 & 18 \\ 17 & 16 & 15 \\ 14 & 13 & 12 \end{bmatrix} & \begin{bmatrix} 11 & 10 & 9 \\ 8 & 7 & 6 \\ 5 & 4 & 3 \\ 2 & 1 & 0 \end{bmatrix} \end{bmatrix}$$

Result 배열

$A_{11}B_{23}$ = (0 * 9) + (1 * 6) + (2 * 3) + (3 * 0)

= 12

$$\begin{matrix} A_{11}B_{1j} \\ A_{11}B_{2j} \end{matrix} \begin{bmatrix} 96 & 90 & 84 \\ 24 & 18 & 12 \end{bmatrix}$$

dot 함수

배열 A Shape = (2, 3, 4)

$$\begin{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} \begin{bmatrix} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix} \end{bmatrix}$$

배열 B Shape = (2, 4, 3)

$$\begin{bmatrix} \begin{bmatrix} 23 & 22 & 21 \\ 20 & 19 & 18 \\ 17 & 16 & 15 \\ 14 & 13 & 12 \end{bmatrix} \begin{bmatrix} 11 & 10 & 9 \\ 8 & 7 & 6 \\ 5 & 4 & 3 \\ 2 & 1 & 0 \end{bmatrix} \end{bmatrix}$$

Result 배열

$A_{12}B_{11}$ = (4 * 23) + (5 * 20) + (6 * 17) + (7 * 14)

= 392

$$A_{11} \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ 96 & 90 & 84 \\ 24 & 18 & 12 \end{bmatrix} \quad A_{12} \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ 392 & & \end{bmatrix}$$

B_{21} B_{22} B_{23}

dot 함수

배열 A Shape = (2, 3, 4)

$$\begin{bmatrix}\begin{bmatrix}0 & 1 & 2 & 3\\ 4 & 5 & 6 & 7\\ 8 & 9 & 10 & 11\end{bmatrix}\begin{bmatrix}12 & 13 & 14 & 15\\ 16 & 17 & 18 & 19\\ 20 & 21 & 22 & 23\end{bmatrix}\end{bmatrix}$$

배열 B Shape = (2, 4, 3)

$$\begin{bmatrix}\begin{bmatrix}23 & 22 & 21\\ 20 & 19 & 18\\ 17 & 16 & 15\\ 14 & 13 & 12\end{bmatrix}\begin{bmatrix}11 & 10 & 9\\ 8 & 7 & 6\\ 5 & 4 & 3\\ 2 & 1 & 0\end{bmatrix}\end{bmatrix}$$

Result 배열 Shape = (2, 3, 2, 3)

$$\begin{bmatrix}\begin{bmatrix}A_{11}\begin{bmatrix}B_{11} & B_{12} & B_{13}\\ B_{21} & B_{22} & B_{23}\end{bmatrix} & A_{12}\begin{bmatrix}B_{11} & B_{12} & B_{13}\\ B_{21} & B_{22} & B_{23}\end{bmatrix} & A_{13}\begin{bmatrix}B_{11} & B_{12} & B_{13}\\ B_{21} & B_{22} & B_{23}\end{bmatrix}\end{bmatrix}\\ \begin{bmatrix}A_{21}\begin{bmatrix}B_{11} & B_{12} & B_{13}\\ B_{21} & B_{22} & B_{23}\end{bmatrix} & A_{22}\begin{bmatrix}B_{11} & B_{12} & B_{13}\\ B_{21} & B_{22} & B_{23}\end{bmatrix} & A_{23}\begin{bmatrix}B_{11} & B_{12} & B_{13}\\ B_{21} & B_{22} & B_{23}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

matmul 함수

배열 A Shape = (2, 3, 4)

$$\begin{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} & \begin{bmatrix} 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \end{bmatrix} \end{bmatrix}$$

배열 B Shape = (2, 4, 3)

$$\begin{bmatrix} \begin{bmatrix} 23 & 22 & 21 \\ 20 & 19 & 18 \\ 17 & 16 & 15 \\ 14 & 13 & 12 \end{bmatrix} & \begin{bmatrix} 11 & 10 & 9 \\ 8 & 7 & 6 \\ 5 & 4 & 3 \\ 2 & 1 & 0 \end{bmatrix} \end{bmatrix}$$

이러한 배열A와 배열B가 있다고 하겠습니다.

해당 np.matmul(배열A, 배열B)의 결과를 살펴보도록 하겠습니다.

matmul 함수

배열 A Shape = (2, 3, 4)

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

배열 B Shape = (2, 4, 3)

```
[[[23 22 21]
  [20 19 18]
  [17 16 15]
  [14 13 12]]

 [[11 10  9]
  [ 8  7  6]
  [ 5  4  3]
  [ 2  1  0]]]
```

Result 배열

```
[[ 96
 [
 ]
 [
 ]]
```

matmul 함수

배열 A Shape = (2, 3, 4)

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

배열 B Shape = (2, 4, 3)

```
[[[ 2  3]  [ 2  2]  [ 2  1]
  [ 2  0]  [ 1  9]  [ 1  8]
  [ 1  7]  [ 1  6]  [ 1  5]
  [ 1  4]  [ 1  3]  [ 1  2]]

 [[ 1  1]  [ 1  0]  [ 9]
  [ 8]     [ 7]     [ 6]
  [ 5]     [ 4]     [ 3]
  [ 2]     [ 1]     [ 0]]]
```

Result 배열

```
[[96 90]
 [  ]
 [  ]]
```

matmul 함수

배열 A Shape = (2, 3, 4)

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

배열 B Shape = (2, 4, 3)

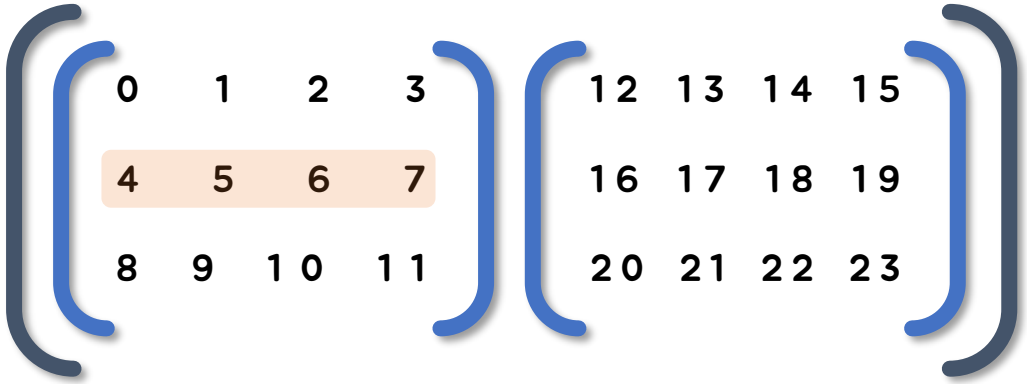
```
[[[23 22 21]
  [20 19 18]
  [17 16 15]
  [14 13 12]]

 [[11 10  9]
  [ 8  7  6]
  [ 5  4  3]
  [ 2  1  0]]]
```

Result 배열

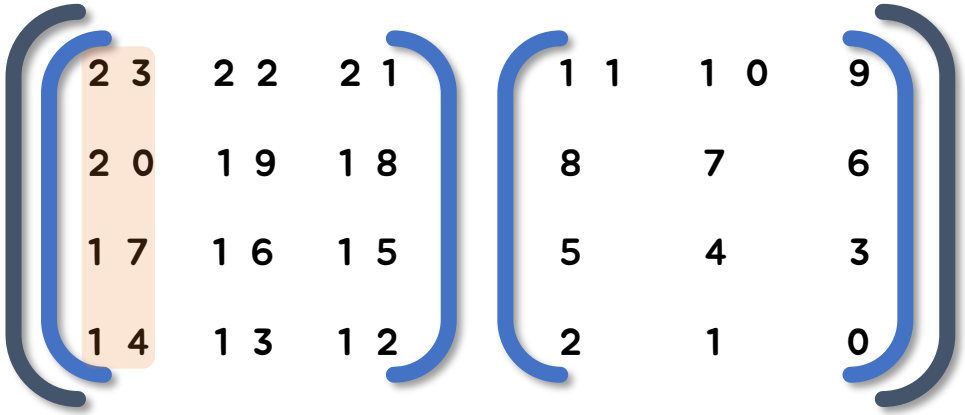
```
[[96 90 84]
 [   ]
 [   ]]
```

matmul 함수

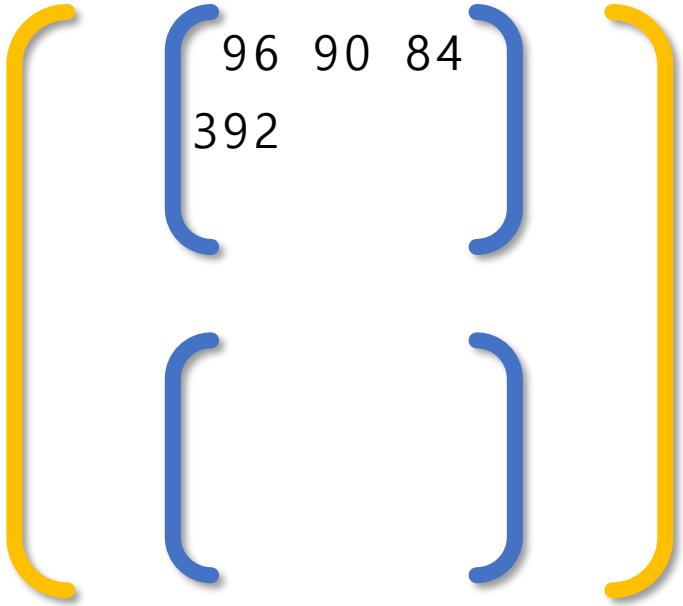
배열 A Shape = (2, 3, 4)



배열 B Shape = (2, 4, 3)



Result 배열



matmul 함수

배열 A Shape = (2, 3, 4)

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

배열 B Shape = (2, 4, 3)

```
[[[ 2  3]  [ 2  2]  [ 2  1]
  [ 2  0]  [ 1  9]  [ 1  8]
  [ 1  7]  [ 1  6]  [ 1  5]
  [ 1  4]  [ 1  3]  [ 1  2]]

 [[ 1  1]  [ 1  0]  [ 9]
  [ 8]     [ 7]     [ 6]
  [ 5]     [ 4]     [ 3]
  [ 2]     [ 1]     [ 0]]]
```

Result 배열

```
[[[ 96  90  84]
  [392 370]]

 [[]]]
```

이러한 방식으로 위에 부분을 채웠다고 한다면,

아랫 부분은 어떻게 계산되는지 알아보도록 하겠습니다.

matmul 함수

배열 A Shape = (2, 3, 4)

$$\begin{bmatrix}\begin{bmatrix}0 & 1 & 2 & 3\\ 4 & 5 & 6 & 7\\ 8 & 9 & 10 & 11\end{bmatrix}\\ \begin{bmatrix}12 & 13 & 14 & 15\\ 16 & 17 & 18 & 19\\ 20 & 21 & 22 & 23\end{bmatrix}\end{bmatrix}$$

배열 B Shape = (2, 4, 3)

$$\begin{bmatrix}\begin{bmatrix}23 & 22 & 21\\ 20 & 19 & 18\\ 17 & 16 & 15\\ 14 & 13 & 12\end{bmatrix}\\ \begin{bmatrix}11 & 10 & 9\\ 8 & 7 & 6\\ 5 & 4 & 3\\ 2 & 1 & 0\end{bmatrix}\end{bmatrix}$$

Result 배열

Shape = (2, 3, 3)

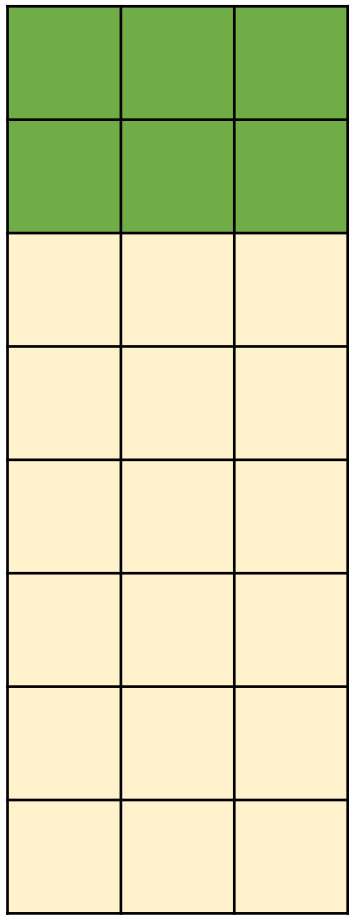
$$\begin{bmatrix}\begin{bmatrix}96 & 90 & 84\\ 392 & 370 & 348\\ 688 & 650 & 612\end{bmatrix}\\ \begin{bmatrix}336 & & \end{bmatrix}\end{bmatrix}$$

아랫 부분은 이렇게 진행이 됩니다.

vsplit

Vertical

Horizontal



arr1에 들어가는 영역

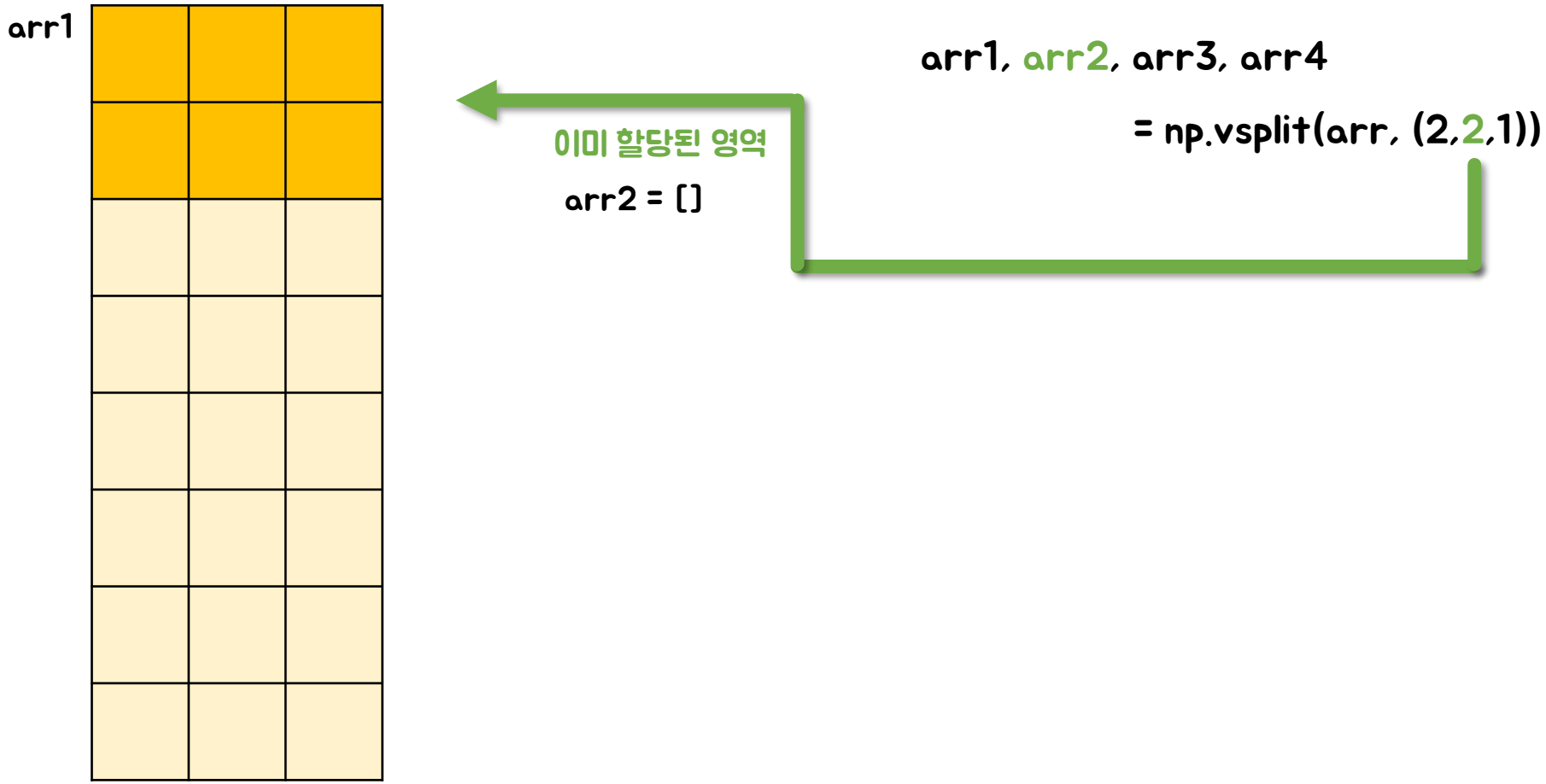
arr1, arr2, arr3, arr4

= np.vsplit(arr, (2,2,1))

vsplit

Vertical

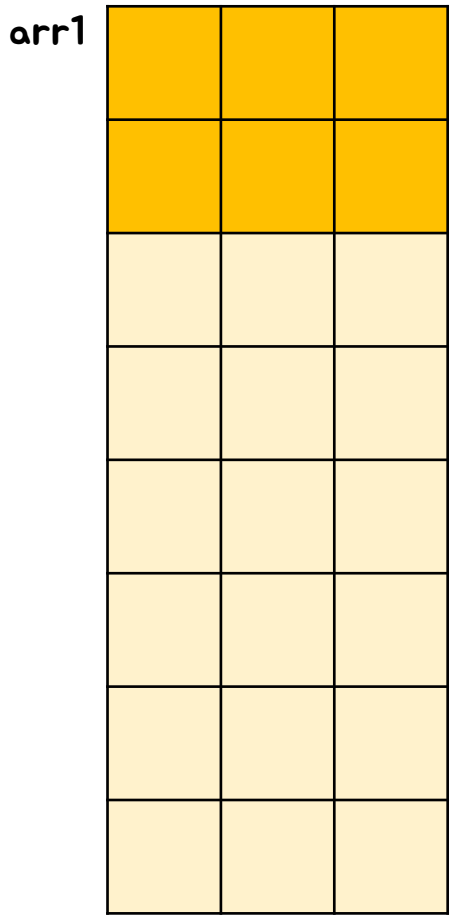
Horizontal



vsplit

Vertical

Horizontal



arr1, arr2, arr3, arr4

= np.vsplit(arr, (2,2,1))

이미 할당된 영역

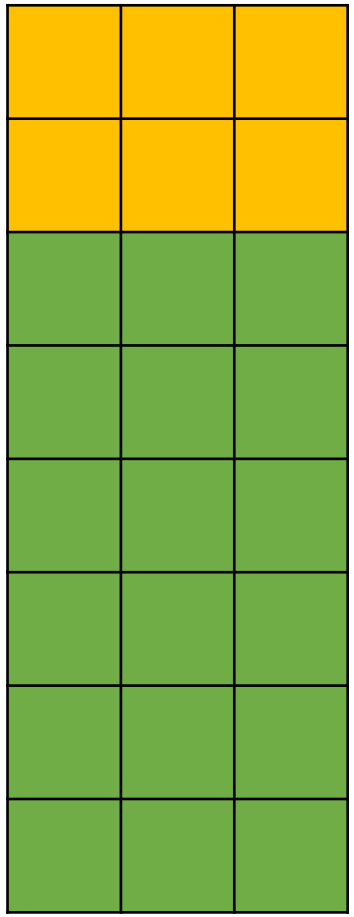
arr3 = []

vsplit

Vertical

Horizontal

arr1



arr1, arr2, arr3, arr4

= np.vsplit(arr, (2,2,1))

할당되지 않은 나머지 영역

arr4

할당 시키기 위해서는 뒤에 인자가 앞에 숫자보다 더 커야함

Numpy

전치행렬

1	2
3	4
5	6

shape(3, 2)

transpose

전치 행렬

1	3	5
2	4	6

shape(2, 3)