# 1 주차 <01. 파이썬 기반의 머신러닝과 생태계 이해(01~10)>

## 01-1. 머신러닝의 개념

### 머신러닝이란?

: 애플리케이션을 수정하지 않고도 데이터를 기반으로 패턴을 학습하고 결과를 추론하는 알고리즘 기법을 통칭

* 데이터 마이닝, 영상인식, 음성 인식, 자연어 처리에서 머신러닝을 적용하면서 급속하게 발전하고 있음
* 기존의 소프트웨어 코드만으로는 해결하기 어려웠던 문제점들을 머신러닝을 이용해 해결해나가고 있음

### 머신러닝, 왜 필요한가?

1. 기존 코드의 문제점
   * 현실 세계에 반영한 매우 복잡하고 방대한 코드
   * 수시로 변하는 환경에 따른 애플리케이션 구현의 어려움
   * 많은 자원과 비용을 통해서 구현된 애플리케이션의 정확성 문제
2. 머신러닝을 통한 복잡한 문제의 해결
   * 복잡한 문제를 데이터 기반으로 숨겨진 패턴을 인지해 해결
   * 통계적인 신뢰도를 강화하고 예측 오류를 최소화하기 위한 다양한 수학적 기법 적용
   * 데이터 내의 패턴을 스스로 인지하고 신뢰도 있는 예측 결과를 도출

### 머신러닝의 분류

1. 지도학습 : Supervised Learning
   * 명확한 결정 값 즉, 라벨링을 단 데이터를 기반으로 학습
   * 예 : 분류, 회귀, 시각/음성 감지/인지
2. 비지도 학습 : Unsupervised Learning
   * 결정값이 주어지지 않는 데이터를 학습
   * 예 : 군집화(클러스터링), 차원 축소
3. 강화 학습 : Reinforcement Learning
   * 주어진 환경에서 보상을 최대한 많이 받을 수 있는 에이전트를 학습하는 것입니다.
   * 해당 강좌에서는 다루지 않음

### 머신러닝 기반의 예측 분석

* 데이터를 관통하는 패턴을 학습, 이에 기반한 예측을 수행해 데이터 분석 영역에 새로운 혁신을 가져옴
* 데이터 관련 전문가들이 머신러닝 알고리즘 기반의 새로운 예측 모델을 이용해 더욱 정확한 예측 및 의사 결정 도출

### 머신러닝 알고리즘 유형

1. 기호주의 : 결정 트리
2. 연결주의 : 인공 신경망, 딥러닝
3. 유전 알고리즘
4. 베이지안 통계
5. 유추주의 : KNN, suport vector machine

### 머신러닝의 단점

* 데이터 의존적 (Garbage in, Garbage out)
* 학습시 최적의 결과를 도출하기 위해 수립된 머신러닝 모델은 실제 환경 데이터 적용 시 과적합 되기 쉬움
* 복잡한 알고리즘으로 인해 도출된 결과는 논리적인 이해가 어려움
* 데이터만 집어 넣으면 자동으로 최적화된 결과를 도출할 것이라는 환상

### 왜 데이터 수집에 열광하는가?

: 다양하고 광대한 데이터를 기반으로 만들어진 머신러닝 모델은 더 좋은 품질을 약속한다.   
앞으로 많은 회사의 경쟁력은 어떠한 품질의 머신러닝 모델을 가지고 있느냐에 결정 될 수 있다.

## 01-2. 파이썬 기반 머신러닝의 특징 및 장점과 구성요소

### R vs. Python - 통계 분석 관점

1. R이란? 통계전용 프로그램 언어
2. 파이썬이란? 객체지향과 함수형 프로그래밍 모두를 포괄하는 유연한 프로그램 아키텍처, 다양한 라이브러리 등의 큰 강점을 가짐
   * 쉽고 뛰어난 개발 생산성으로 전 셰계 개발자들 또는 타 영역의 인재들도 선호
   * 오픈 소스 계열의 전폭적인 지원을 받고 있음
   * 많은 라이브러리 지원
   * 단, 인터프리터 언어의 특성상 속도가 느리지만 유연한 특징으로 인해 다양한 영역에서 사용되고 있음

: 강의자의 개인적인 의견이나, 개발 언어에 익숙하지 않거나, 통계 분석에 능하다면 R이 나음

**그러나 이제 머신러닝을 시작하려는 사람이라면, 특히 개발자라면 파이썬을 더 권하고 싶다!!**

### ML + Python 강점

1. 뛰어난 확장성, 연계, 호환성
   * 분석 영역을 넘어서 머신러닝 기반의 다양한 어플리케이션 개발이 쉽게 가능
   * 기존 어플리케이션과의 연계도 쉬움
2. 딥러닝으로의 진격
   * 유수의 딥러닝 프레임워크가 파이썬 기반으로 작성
   * 텐서 플로우, 케라스, 파이토치 등

### 파이썬 머신러닝 생태계를 구성하는 주요 패키지

1. 머신러닝 패키지
   * scikit learn : 딥러닝 포함 X
   * Tensor flow, keras, pytorch
2. 배열/ 선형대수/ 통계 패키지
   * numpy
   * scipy
3. 데이터 핸들링
   * Pandas
4. 시각화
   * matplotlib
   * seaborn
5. 대화형 파이썬 툴
   * jupyter notebook
   * collab

## 01-3. 파이썬 기반 머신러닝을 위한 SW 설치

### 파이썬 머신러닝을 위한 소프트웨어 설치

: 파이썬 머신러닝을 위한 패키지를 설치하는 가장 쉬운 방법은 아나콘다를 이용하는 것(pip 보다 유용)

### LightGBM을 위한 VisualStudio Build Tools 설치

: window의 경우에는 VisualStudio Build Tools을 설치해야 하나 맥에서는 terminal에서 설치 가능

## 01-4. 주피터 노트북 사용법과 넘파이/판다스의 필요성

### 주피터 노트북

: 대표적인 대화형 파이썬 툴로 특정 코드 영역별로 개별 수행을 지원하여 영역별로 코드 이해가 매우 명확하게 설명

### 머신러닝을 위한 넘파이와 판다스의 중요성

* 머신러닝 애플리케이션 구현에서 다양한 데이터의 추출/가공/변환이 상당한 영역을 차지하고 데이터 처리부분은 넘파이와 판다스의 몫
* 사이킷런이 넘파이 기반에서 작성됐기 때문에 넘파이의 기본 프레임워크를 이해하지 못하면 역시 실제 구현에서 이해 못함
* 사이킷런은 API 구성이 간단하나 넘파이와 판다스는 방대하기 때문에 이를 익히는데 시간이 많이 소모될 수 있음
* : 따라서 넘파이와 판다스에 대한 기본 프레임워크와 중요 API만 습득, 일단 코드로 직접 해보는게 좋음

## 01-5. 강의에 사용될 예제 소스 코드 다운로드 받기

: 실습으로 진행

* 다운 링크 : <https://github.com/chulminkw/PerfectGuide>

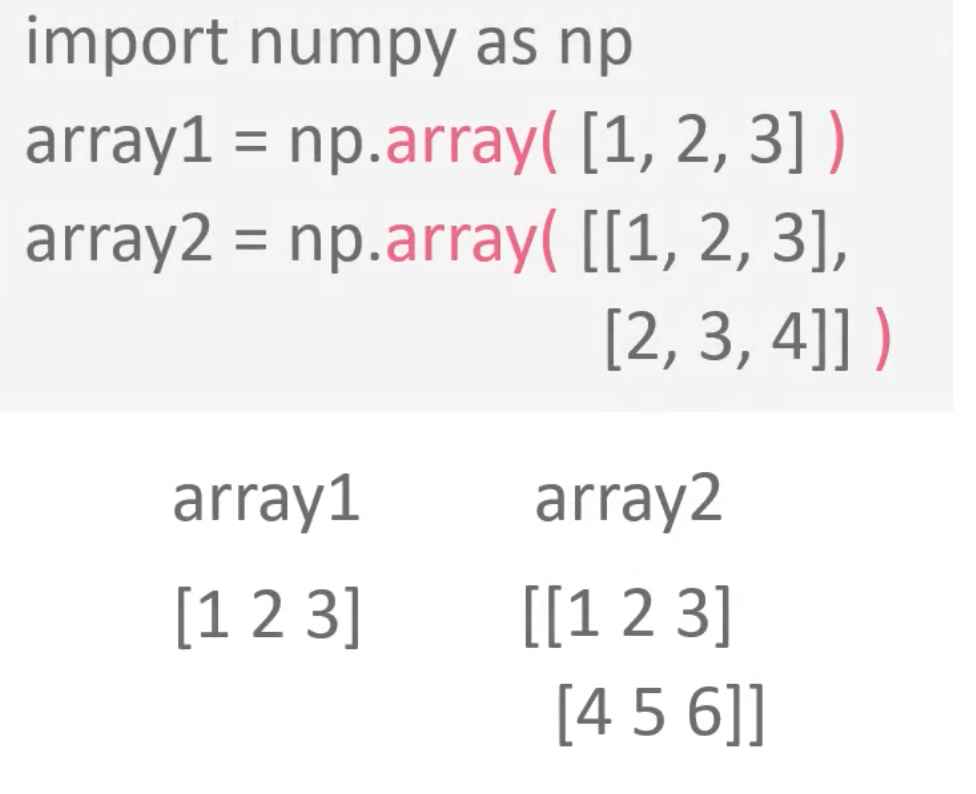
## 01-6. 넘파이 배열 ndarray 소개

### 넘파이 ndarray

* ndarray : N-Dimension Array 객체

### ndarray 생성

: numpy 모듈의 array() 함수로 생성, 인자로 주로 파이썬 리스트 또는 ndarray 입력



### ndarray 형태(shape)와 차원

* ndarray의 shape : ndarray.shape 속성 사용
* ndarray의 차원: ndarray.ndim 속성 사용

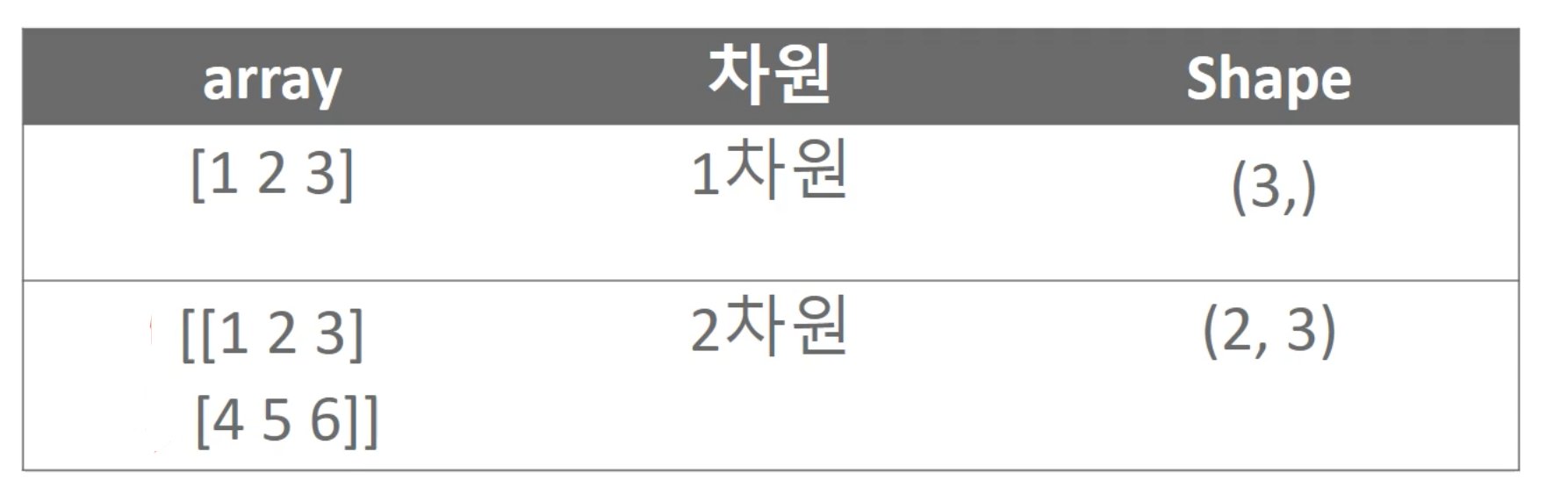


image-20201005140207220

### ndarry type

* ndarry내의 데이터 값은 숫자, 문자열, boolean 값 모두 가능
  + 정수 : int (8/16/32 bit), unsigned int(8/16/32 bit)
  + 실수 : float (16/32/64/128 bit)
  + 이보다 더 큰 숫자 값이나 정밀도를 위해 complex 타입도 제공
* ndarry내의 데이터 타입은 그 연산의 특성상 같은 데이터 타입만 가능
* : 즉, 한 개의 ndarry객체에 int,float가 함께 있을 수 없음
* ndarry내의 데이터 타입은 ndarry.dtype으로 확인할 수 있음

### ndarry 타입 변환

* astype()을 이용하여 변환
  + 변경을 원하는 타입을 해당 함수에 인자로 입력
  + 대용량 데이터를 ndarry 만들 때 메모리를 절약하기 위해 자주 사용
  + 예: 데이터에 주로 0~10 정도의 값만 있다면 64bit float -> 8bit integer로 형을 변환하면 메모리 절약이 많이 됨
* **대용량 데이터를 다룰 시 메모리 절약을 위해서 형 변환을 특히 고려해야 함**

### 넘파이 ndarry의 axis 축

: ndarry는 shape는 행 , 열, 높이 단위로 부여되는 것이 아닌 axis0, axis1,.. 와 같이 axis 단위로 부여됨

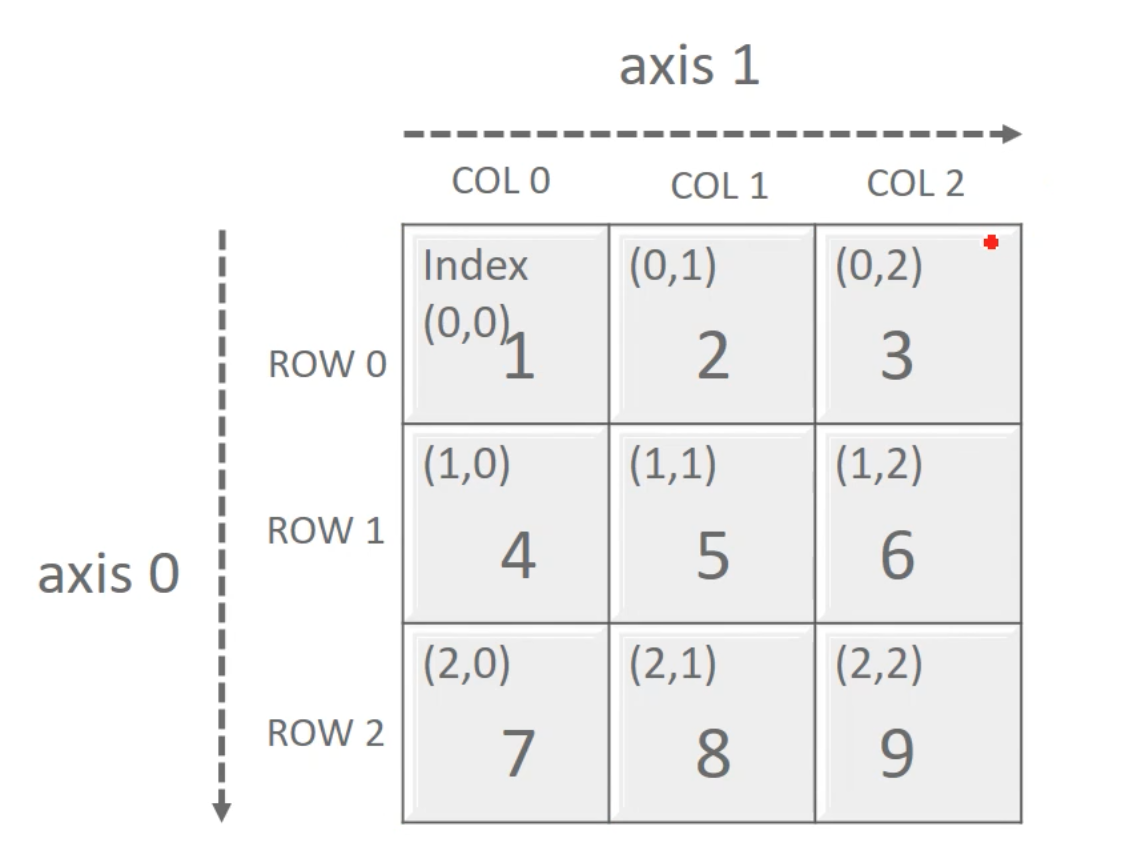


image-20201005141119848

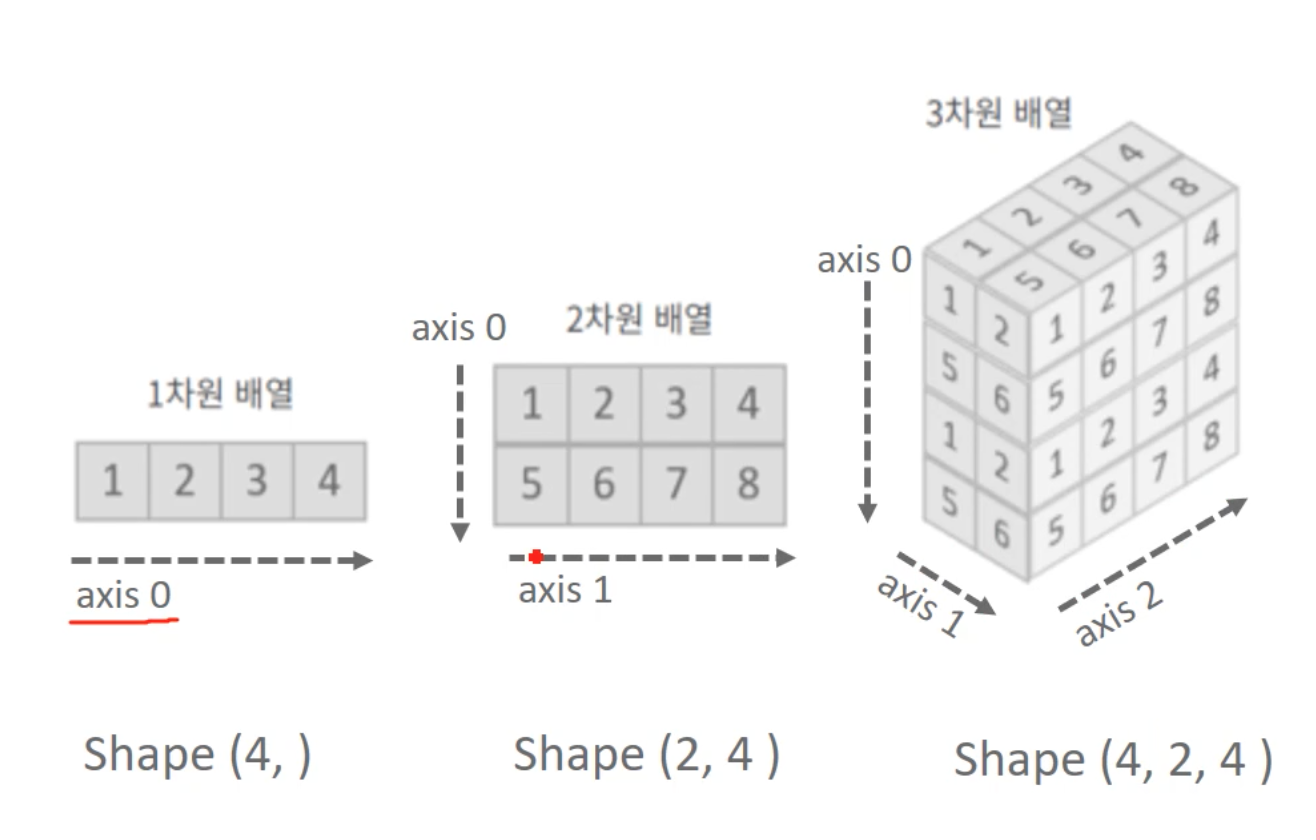


image-20201005141230618

-> 실습

## 01-7. 넘파이 배열 ndarray 초기화 방법과 ndarray 차원과 크기를 변경하는 reshape()의 이해

### ndarray를 편리하게 생성하기

: 특정 크기와 차원을 가진 ndarray를 연속 값이나 0 또는 1로 초기화 생성해야 할 경우 아래의 함수를 이용하여 생성 가능

주로, 테스트 용으로 데이터를 만들거나 대규모의 데이터를 일괄적으로 초기화해야 할 경우 사용

* arrange()
* zeros()
* ones()

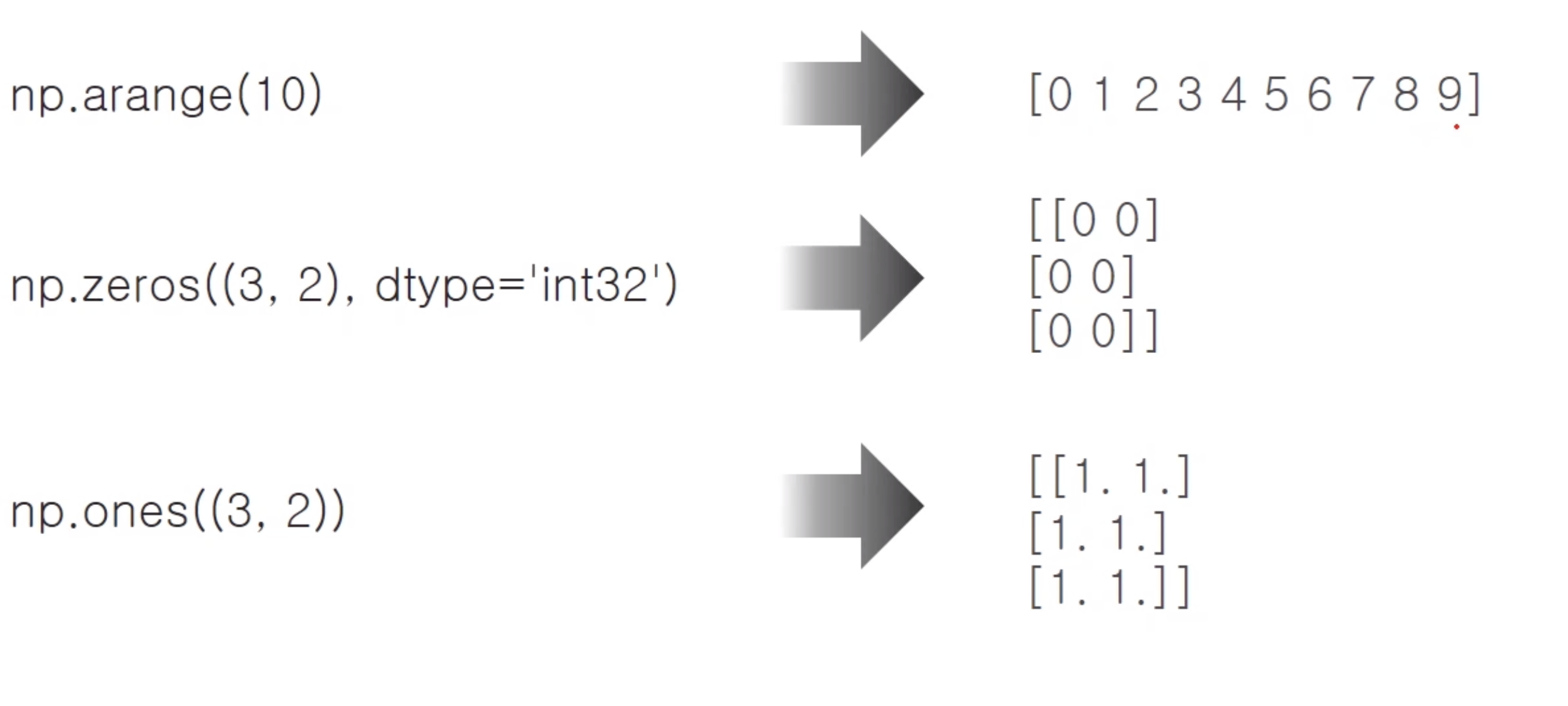


image-20201008194525077

### ndarray의 차원과 크기를 변경하는 reshape()

* ndarray를 특정 차원 및 형태로 변환, 변환 형태를 함수 인자로 부여하면 됨

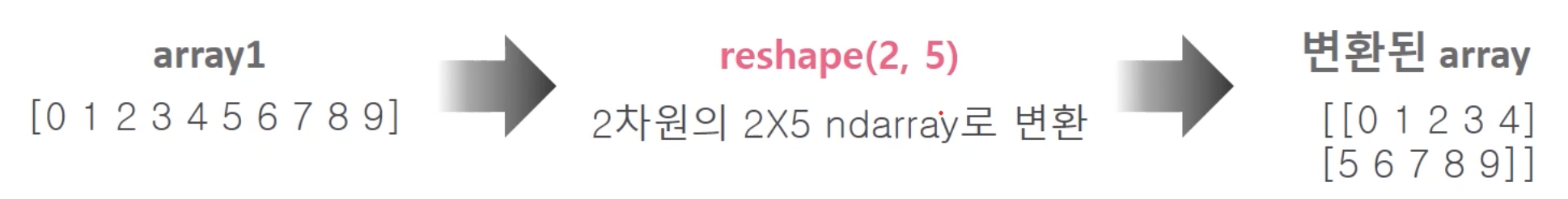


image-20201005165933306

* 인자에 -1을 부여 시, -1 에 해당하는 axis 크기는 가변적이되 -1이 아닌 인자값에 해당하는 axis 크기는 인자값으로 가정하여 ndarray의 shape을 변화시킴

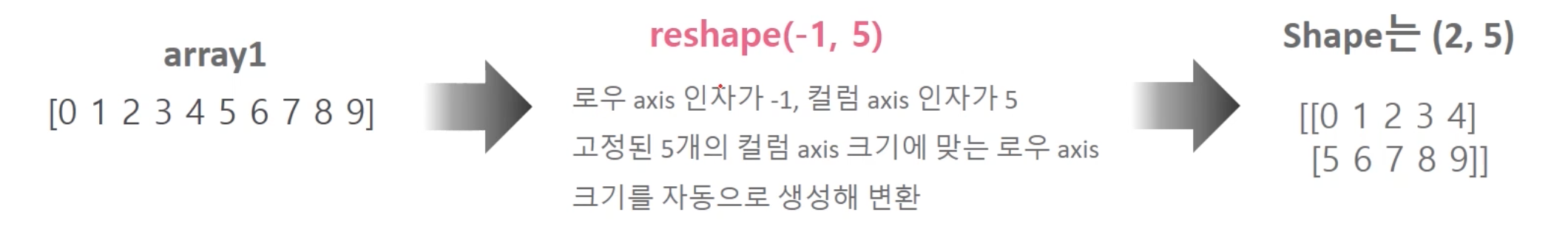
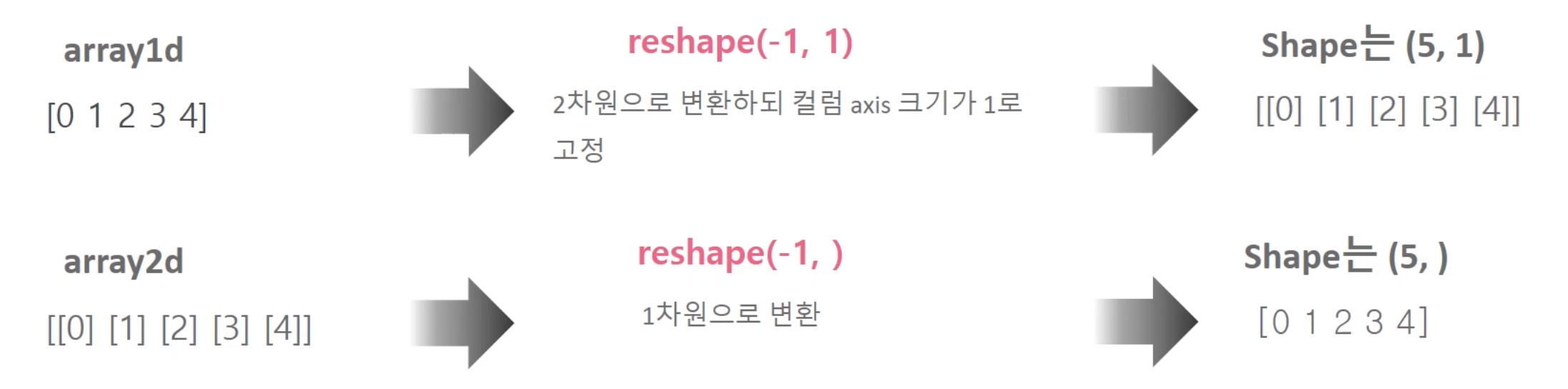


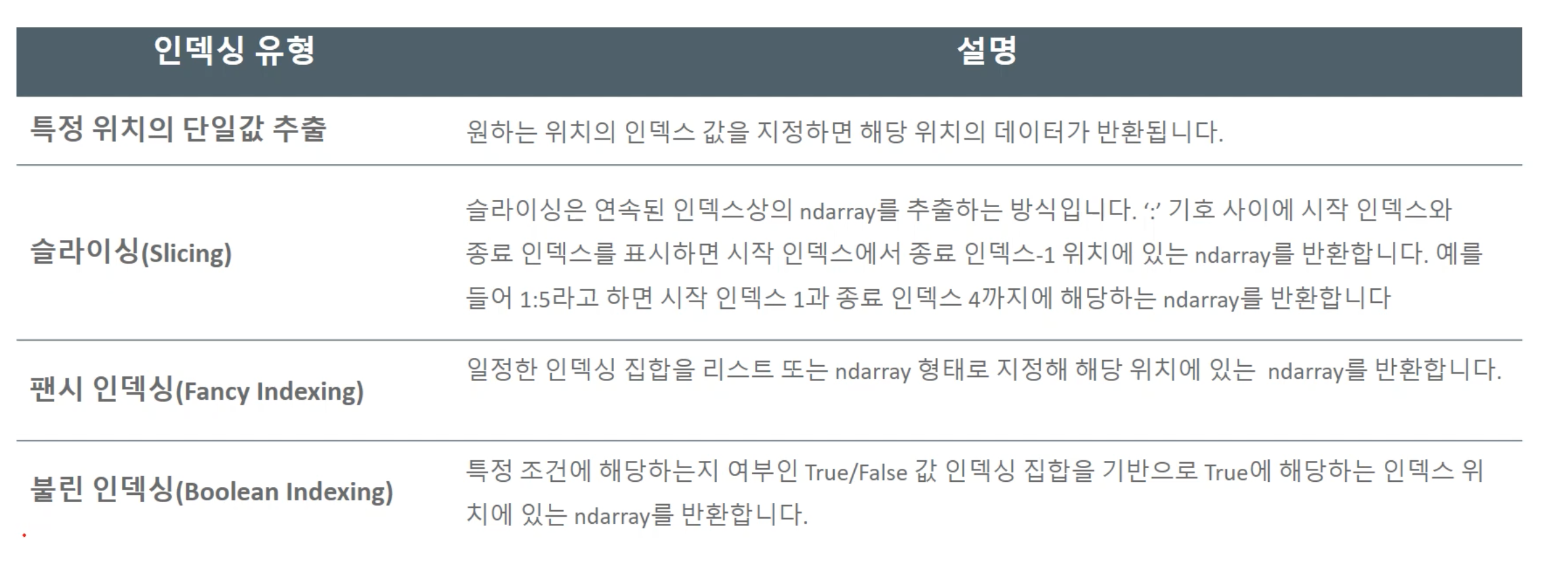
image-20201005170056029

* reshape()는 reshape(-1,1), reshape(-1,) 과 같은 형식으로 변환이 요구되는 경우가 많음
* : 주로 머신러닝 API의 인자로 1차원 ndarray를 명확하게 2차원 ndarray로 변환하여 입력하기를 원하거나 또는 반대의 경우가 있을 경우 reshape()를 이용하여 ndarray의 형태를 변환시켜 주는데 사용
* 
* -> 실습

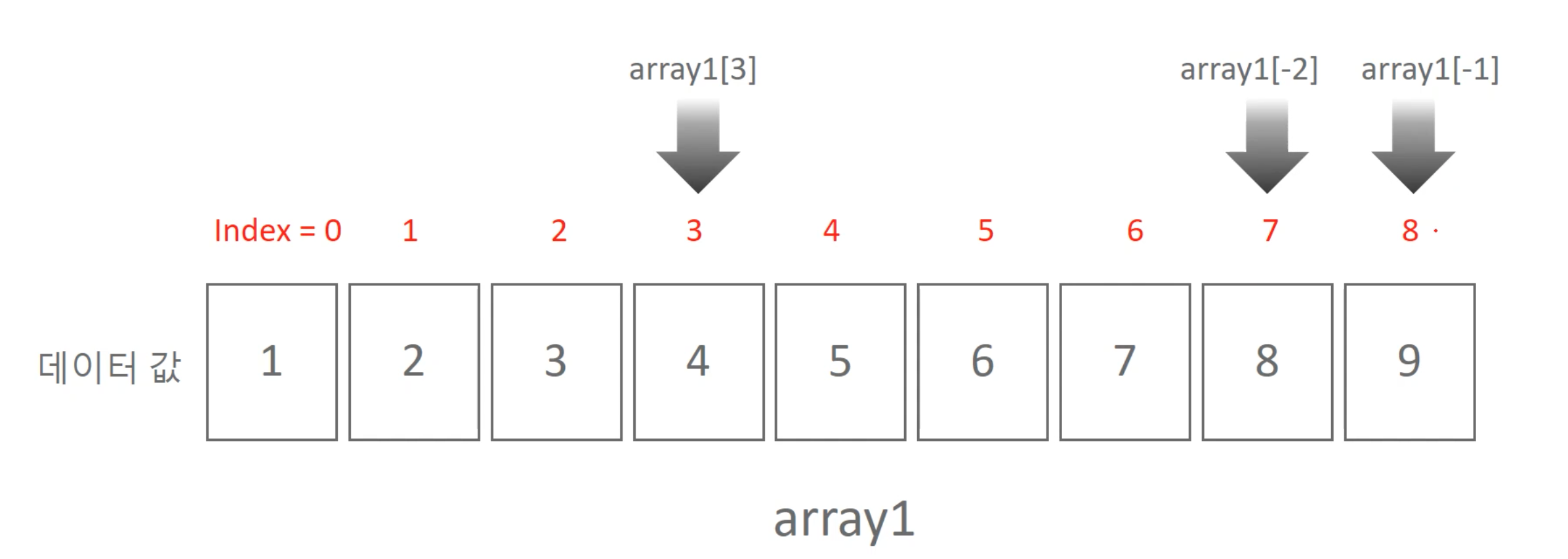
## 01-8. 넘파이 ndarray 의 인덱싱을 통한 데이터 세트 선택하기 - 01

### ndarray의 데이터 세트 선택하기 - 인덱싱

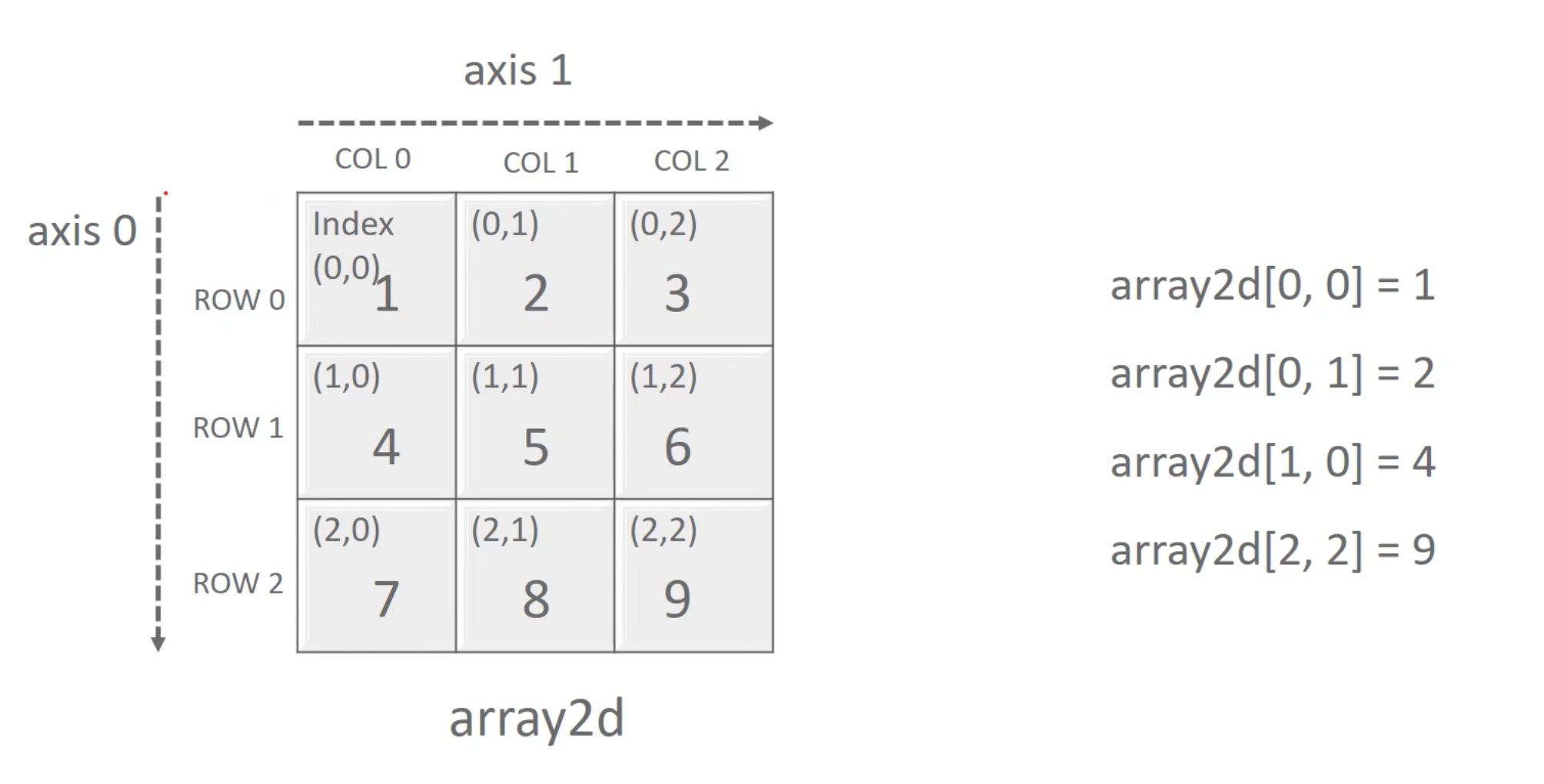
: **불린 인덱싱이 특히 중요**



### 단일 값 추출 - 1차원 ndarray

* ndarray는 axis를 기준으로 0부터 시작하는 위치 인덱스 값을 가지고 있음
  + 해당 인덱스 값을 [ ] 에 명시하여 단일 값을 추출한다.
  + 마이너스가 인덱스로 사용되면 맨 뒤에서부터 위치를 지정함
* 
* image-20201005172034365

### 단일 값 추출 - 2차원 ndarray



### 슬라이싱 - 1차원 ndarray

* 슬라이싱은 :을 이용하여 연속된 값을 선택

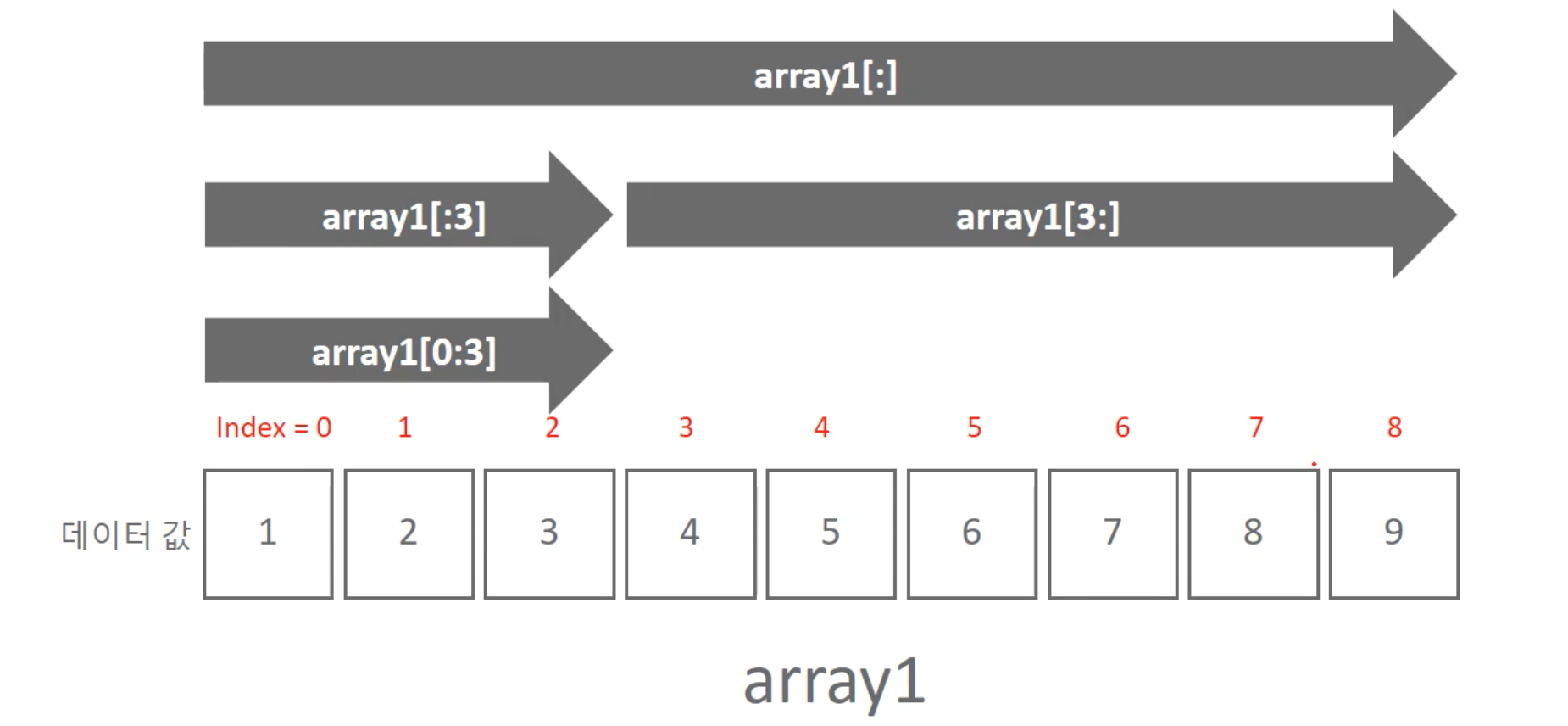
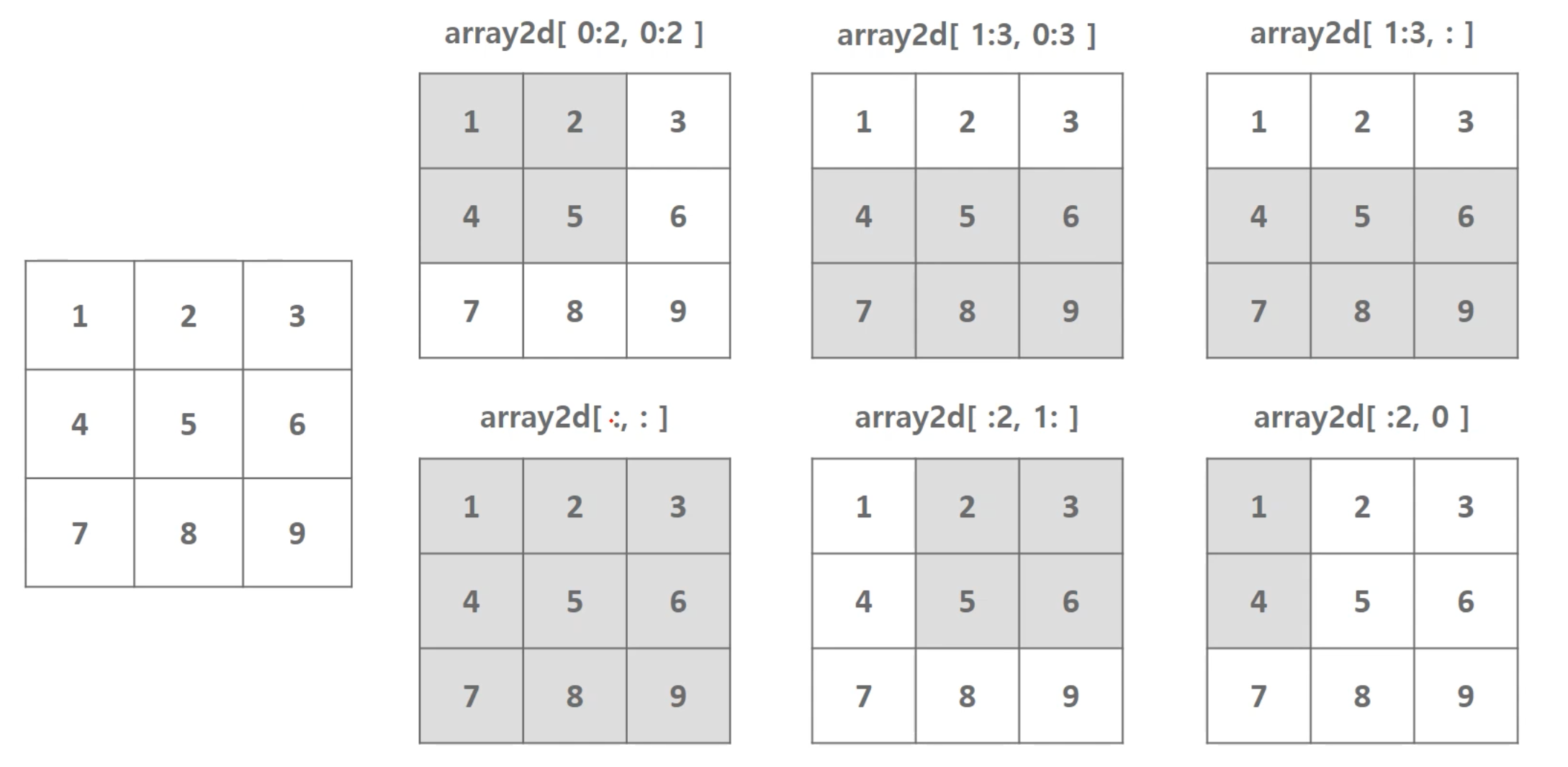


image-20201005172244563

### 슬라이싱 - 2차원 ndarray



### 팬시 인덱싱 - 1차원 ndarray

: 리스트나 ndarray로 인덱스 집합을 지정하면 해당 위치의 인덱스에 해당하는 ndarray를 반환하는 방식의 인덱싱 방식

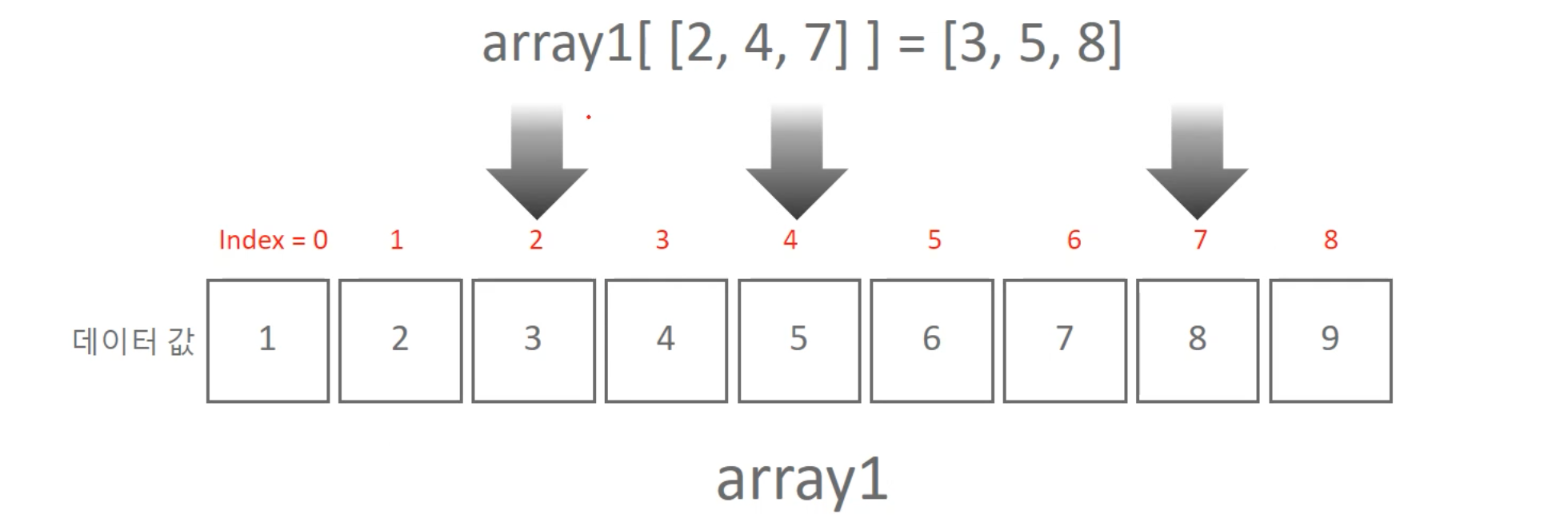


image-20201005172802557

### 팬시 인덱싱 - 2차원 ndarray

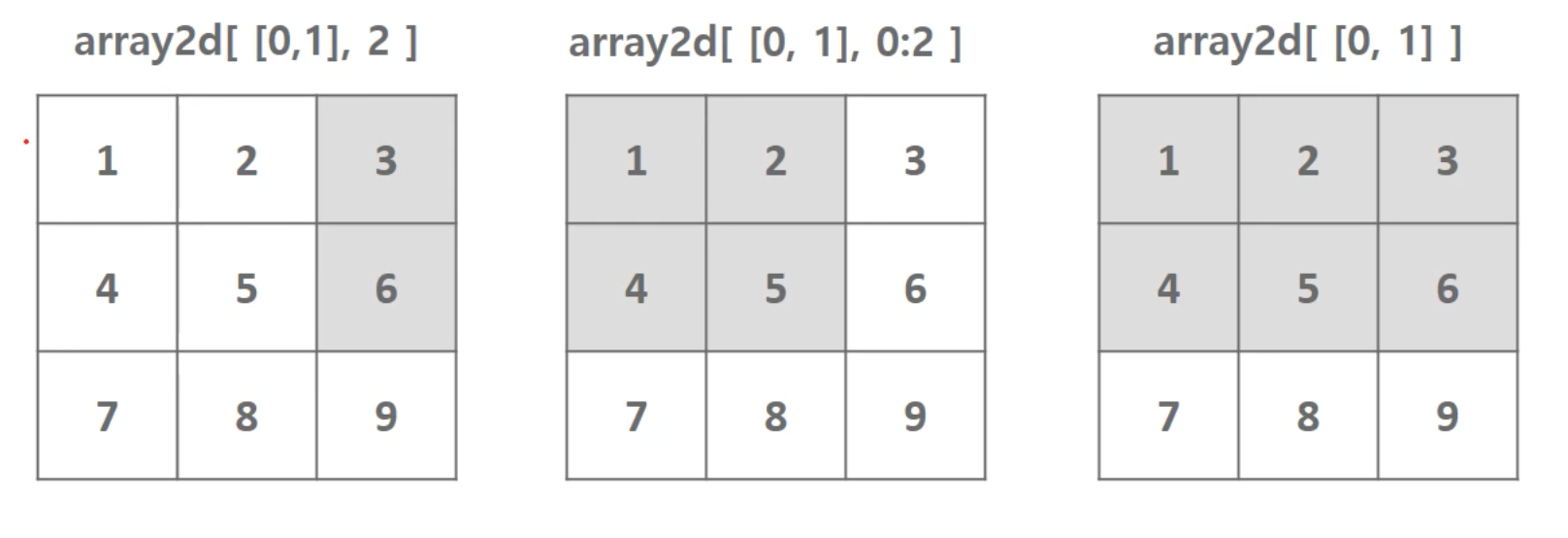
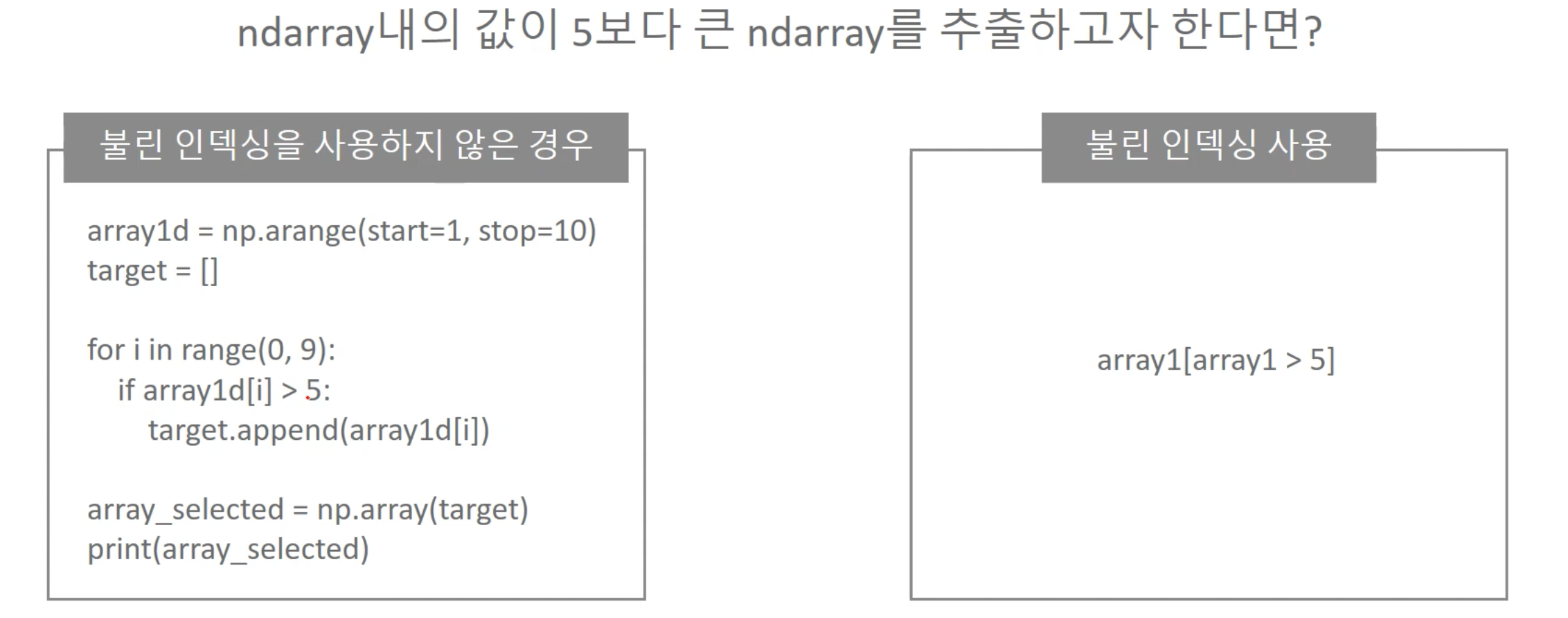


image-20201005172856456

### 불린 인덱싱

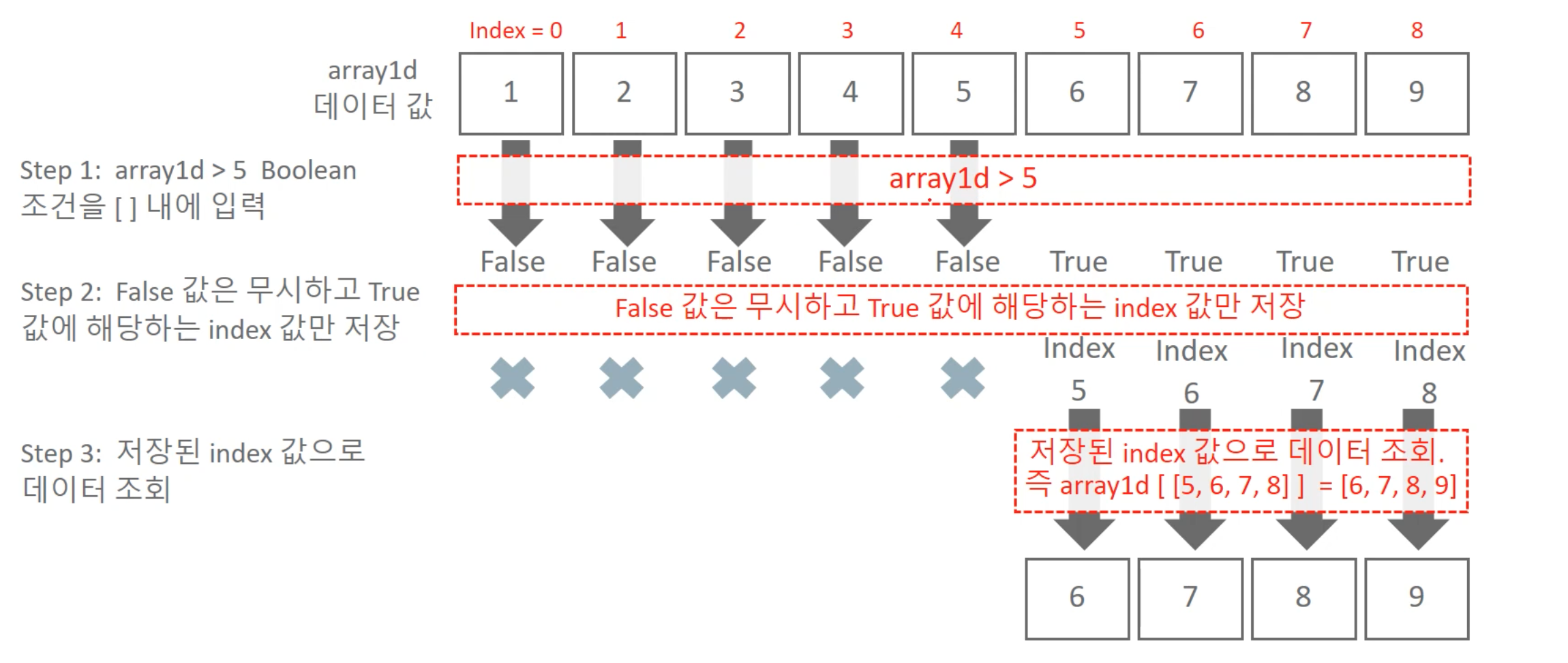
: 조건 필터링과 검색을 동시에 할 수 있기 때문에 **매우 자주 사용되는 인덱싱 방식**



## 01-9. 넘파이 ndarray 의 인덱싱을 통한 데이터 세트 선택하기 - 02

-> 실습

### 불린 인덱싱의 메커니즘



## 01-10. 넘파이 ndarray의 정렬과 선형대수 연산

### 배열의 정렬 -sort(), argsort()

* sort()란?
* : 넘파이 sort( ) 유형
  1. np.sort() : 원 행렬은 그대로 유지한 채 원 행렬의 정렬된 행렬을 반환
  2. ndarray.sort()는 원 행렬 자체를 정렬한 형태로 변환하며 반환 값은 None
  + np.sort나 ndarray.sort()는 모두 기본적으로 오름차순으로 행렬 내 원소를 정렬, 내림차순 정렬은 [::-1] 붙이면 가능
* 2차원 배열에서 axis기반의 sort()

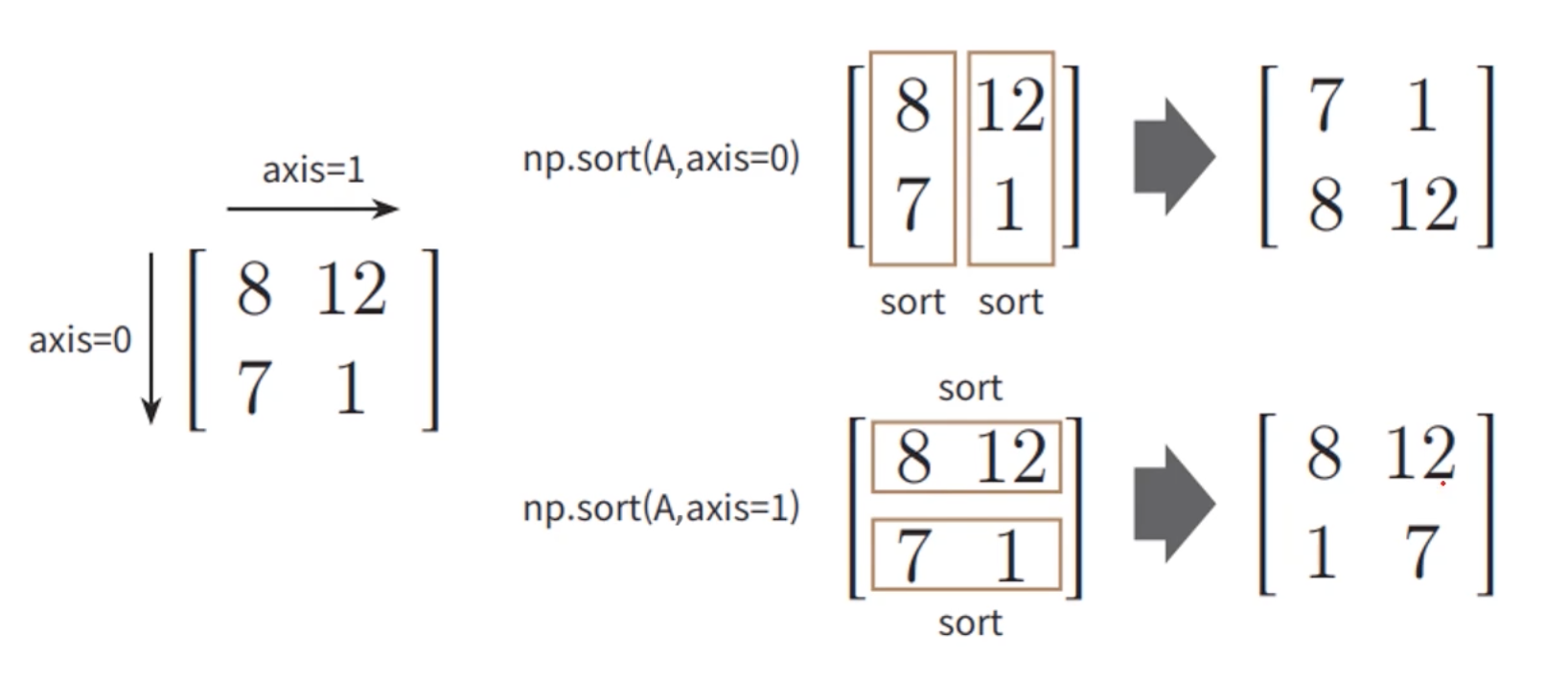
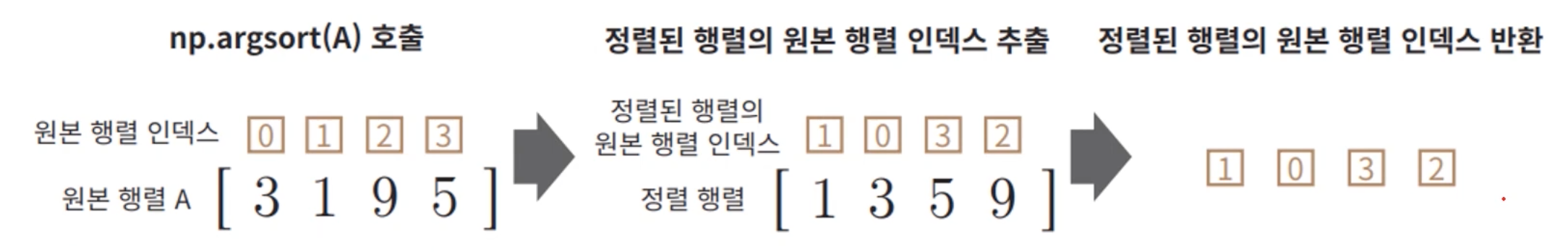


image-20201005174412344

* argsort( )란?
  + 원본 행렬 정렬 시 정렬된 행렬의 원래 인덱스를 필요로 할 때 np.argsort()를 이용
  + 정렬 행렬의 원본 행렬 인덱스를 ndarray 형으로 반환
* 
* image-20201005174543377

### 선형대수 연산 - 행렬 내적

np.dot(A,B) : A와 B를 내적

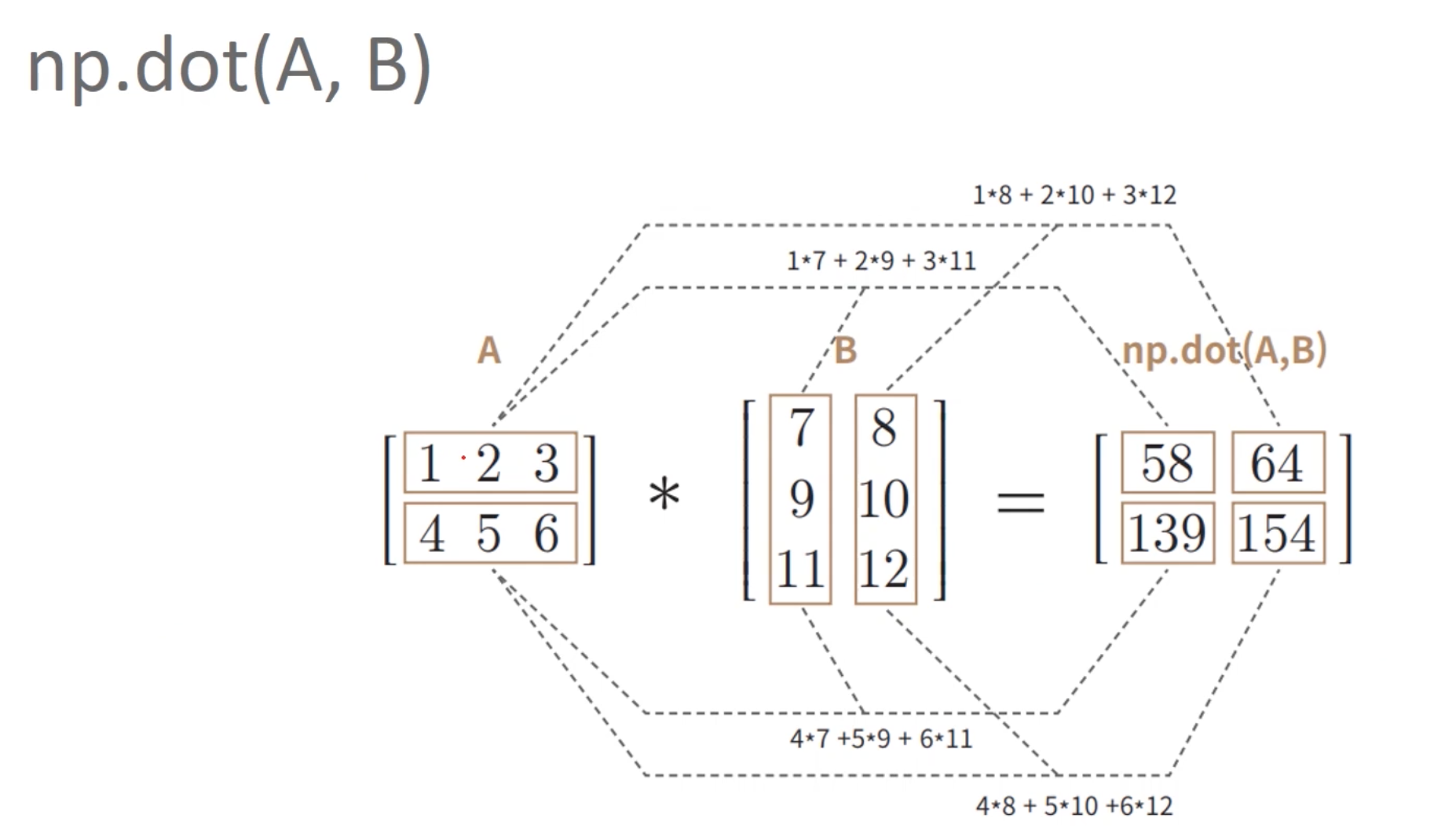


image-20201005174638415

### 선형대수 연산 - 전치 행렬

np.transpose(A) : A의 전치 행렬을 반환

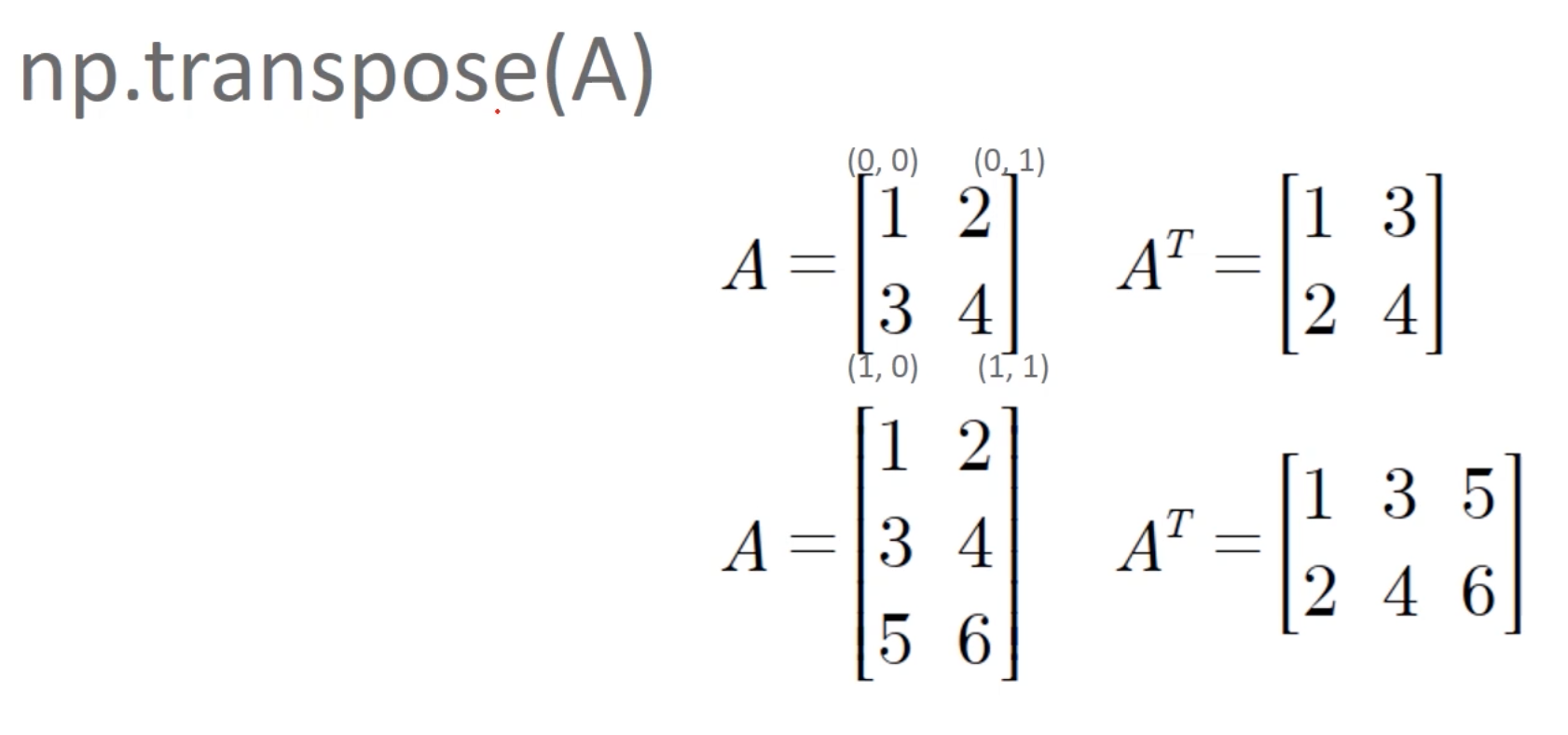


image-20201005174746846

-> 실습

### 넘파이 Summary

* 넘파이는 파이썬 머신러닝을 구성하는 핵심 기반으로 반드시 이해가 필요
* 넘파이 API는 매우 넓은 범위이므로 머신러닝 애플리케이션을 작성할 때 중요하게 활용될 수 있는 핵심 개념 위주로 숙지
* 넘파이는 판다스에 비해서 친절한 API를 제공하지 않음
* : 2차원의 데이터라면 데이터 가공을 위해서 넘파이 보다는 판다스를 이용하는 것이 보다 효율적