
Introduction To Data Analysis
데이터분석입문
Lecture 23. 과학 컴퓨팅 패키지 numpy
인공지능소프트웨어학과 강환수 교수

학습개요

- 과학 컴퓨팅 패키지 numpy 개요
- 주요 자료형 np.ndarray
- 배열의 속성과 모양 수정
- 배열의 인덱싱과 슬라이싱
- 배열 항목의 조건

학습목표

- 패키지 numpy의 주요 자료형 ndarray를 다양한 방법으로 생성할 수 있다.
- 다차원 배열 array의 모양을 바꿀 수 있다.
- 배열에서 인덱싱과 슬라이싱으로 다른 모양의 배열을 반환 받을 수 있다.
- 배열에서 특정 조건이 만족되는 항목을 반환 받을 수 있다.

LESSON 01

Numpy 개요와 ndarray 자료형

Numpy 개요

Numerical python

- Python의 과학 컴퓨팅을 위한 기본 패키지
  - ✚ 과학 기술을 위한 산술 계산 라이브러리
- 대규모 다차원 배열과 행렬 연산에 필요한 다양한 함수를 제공

제공 기술

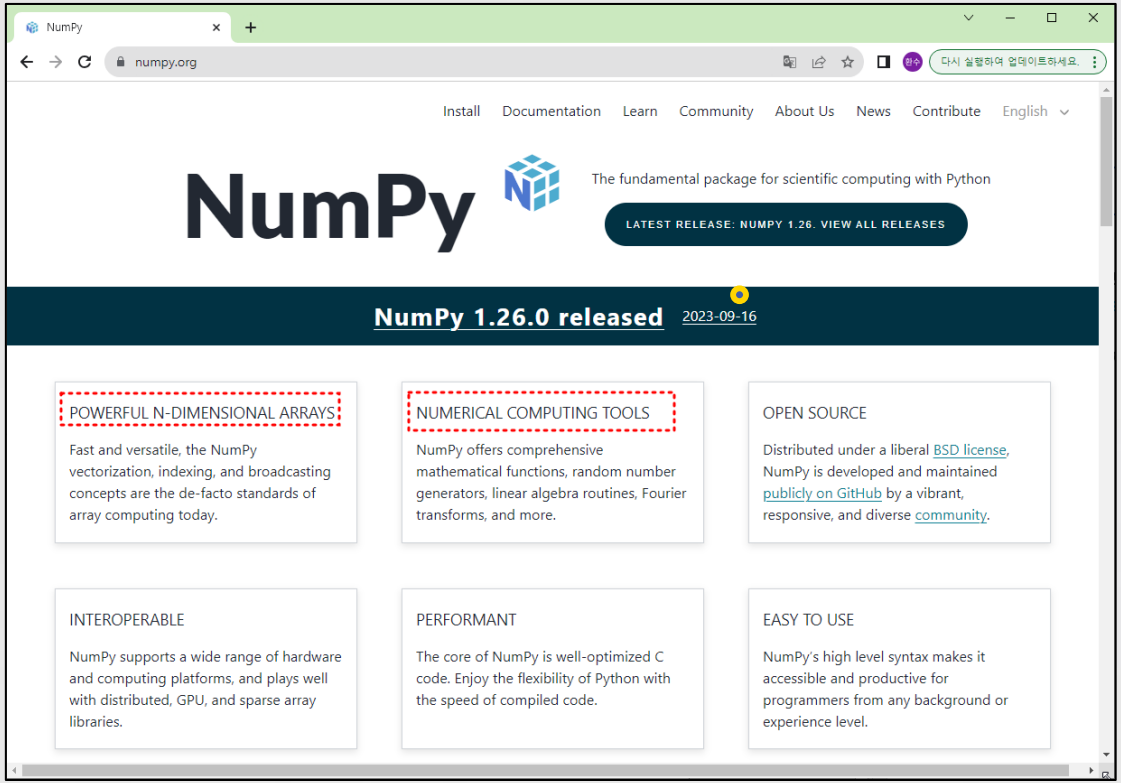
- 다차원 배열 ndarray
- 정교한 브로드캐스팅(Broadcast) 기능
- 전체 데이터 배열을 빠르게 계산하는 표준 수학 함수
- 선형대수, 난수 생성기

장점

- 대용량 배열 데이터를 효율적으로 다뤄 빠르게 처리
- 중요 알고리즘 구현은 C로 작성
- 반복문을 사용하지 않고 빠르게 계산

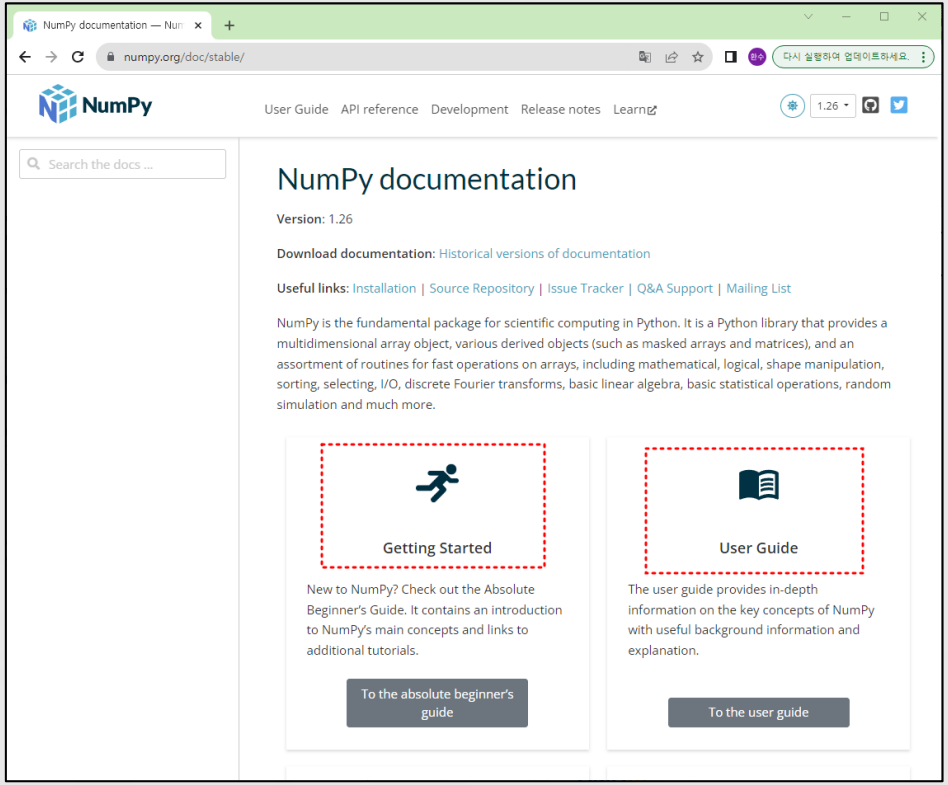
홈페이지

numpy.org



학습

numpy.org


NumPy documentation — Nun
numpy.org/doc/stable/
다시 실행하여 업데이트하세요.
NumPy
User Guide
API reference
Development
Release notes
Learn
1.26
Search the docs ...
NumPy documentation
Version: 1.26
Download documentation: Historical versions of documentation
Useful links: Installation | Source Repository | Issue Tracker | Q&A Support | Mailing List
NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.
Getting Started
New to NumPy? Check out the Absolute Beginner's Guide. It contains an introduction to NumPy's main concepts and links to additional tutorials.
To the absolute beginner's guide
User Guide
The user guide provides in-depth information on the key concepts of NumPy with useful background information and explanation.
To the user guide

패키지 설치

표준 파이썬

- 외부 패키지이므로 설치 필요
  - > pip install numpy

아나콘다

- 기본 설치 패키지

텐서

Tensor

- 넓은 의미로 자료의 모임이 텐서(tensor)
- 작은 의미로 특히 3차원 이상 배열을 텐서(tensor)라고도 부름
  - [[[1, 2], [3, 4]], [[5, 6], [7, 8]]]

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

행렬과 용어

스칼라

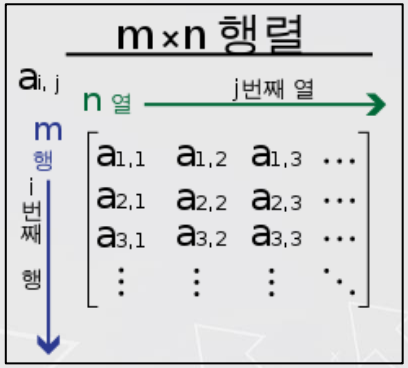
- 스칼라는 하나의 숫자만으로 이루어진 데이터를 의미
  - 3

벡터

- 여러 숫자가 순서대로 모여 있는 것으로, 일반적인 일차원 배열이 벡터
  - [1, 2, 3, 4]

행렬

- 복수의 차원을 가지는 데이터가 다시 여러 개 있는 경우의 데이터를 합쳐서 표기한 것
- 일반적으로 2차원 배열이 행렬
  - [[1, 2], [3, 4]]



자료 유형 ndarray

다차원 배열을 지원하는 유형

- Multi dimensional array
  - Ndarry
- 모든 항목이 동일한 자료형으로 저장되는 구조

Ndarray를 생성하는 함수

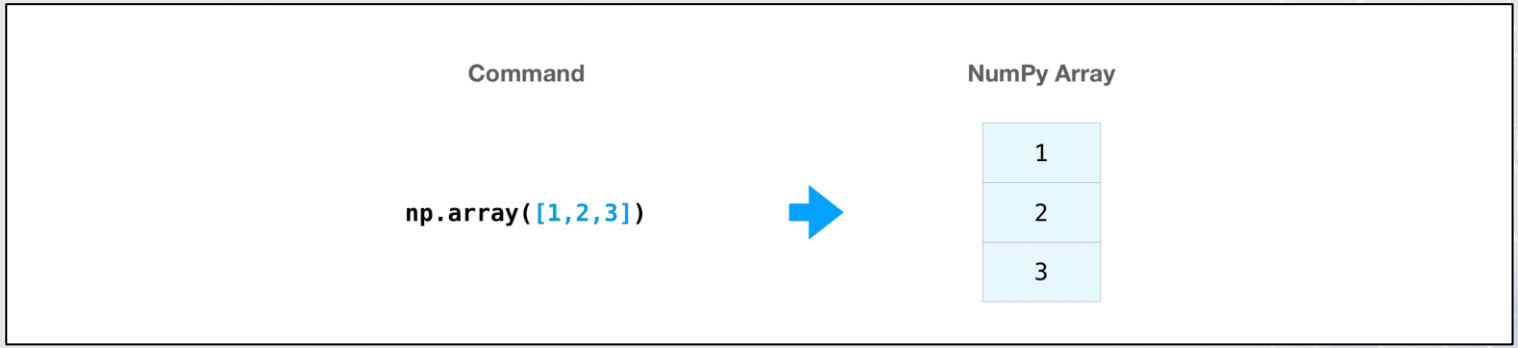
- np.array()
- np.arange(), np.linspace()
- np.zeros(), np.ones()

산점도(Scatter Plot)의 색상

```
import numpy as np
np.array([1, 2 , 3])
```

✓ 0.0s

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



항목의 자료형

기본 데이터 유형

- 부동 소수점(np.float64)

키워드 dtype를 사용하여 원하는 데이터 유형을 명시적으로 지정

- dtype=np.int64

```
a = np.ones(2)
print(a)
print(a.dtype)
```

✓ 0.0s

```
[1. 1.]
float64
```

```
np.ones(2, dtype=np.int64)
```

✓ 0.0s

```
array([1, 1], dtype=int64)
```

항목 정렬

sort()

- 항목 정렬

argsort()

- 항목의 정렬 순번 반환

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

✓ 0.0s

```
array([2, 1, 5, 3, 7, 4, 6, 8])
```

```
arr.sort()
arr
```

✓ 0.0s

```
array([1, 2, 3, 4, 5, 6, 7, 8])
```

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

✓ 0.0s

```
array([2, 1, 5, 3, 7, 4, 6, 8])
```

```
arr.argsort()
```

✓ 0.0s

정렬된 첨자가 각각 1, 0, 3, …

```
array([1, 0, 3, 5, 2, 6, 4, 7], dtype=int64)
```

배열을 만드는 arange와 linspace

np.arange(start, stop, step)

- [start, stop), 구간 길이: step

np.linspace(start, stop, num)

- [start, stop], 구간 길이: (stop-start)/(num-1)

```
print(np.arange(4))
print(np.arange(2, 9, 2))
print(np.arange(1, 5, .7))
```

✓ 0.0s

```
[0 1 2 3]
[2 4 6 8]
[1.  1.7 2.4 3.1 3.8 4.5]
```

```
print(np.linspace(0, 10, 5))
print(np.linspace(0, 10, num=5))
print(np.linspace(0, 10, num=11))
```

✓ 0.0s

```
[ 0.   2.5  5.   7.5 10. ]
[ 0.   2.5  5.   7.5 10. ]
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
```

LESSON 02

배열의 모양과 조건

배열 속성

- arr.ndim
- arr.size
- arr.shape

```
array_example = np.array([[[0, 1, 2, 3],
                           [4, 5, 6, 7]],
                          [[0, 1, 2, 3],
                           [4, 5, 6, 7]],
                          [[0 ,1 ,2, 3],
                           [4, 5, 6, 7]]])
array_example
```

✓ 0.0s

```
array([[[0, 1, 2, 3],
        [4, 5, 6, 7]],

       [[0, 1, 2, 3],
        [4, 5, 6, 7]],

       [[0, 1, 2, 3],
        [4, 5, 6, 7]]])
```

```
array_example.ndim
```

✓ 0.0s

```
3
```

```
array_example.size
```

✓ 0.0s

```
24
```

```
array_example.shape
```

✓ 0.0s

```
(3, 2, 4)
```

배열 모양 수정

arr.reshape

```
a = np.arange(6)
print(a)
```

✓ 0.0s

```
[0 1 2 3 4 5]
```

```
b = a.reshape(3, 2)
print(b)
```

✓ 0.0s

```
[[0 1]
 [2 3]
 [4 5]]
```

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

✓ 0.0s

```
(6,)
```

다차원 모양 수정

```
data = np.arange(30)
data
```

✓ 0.0s Python

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29])
```

```
data.reshape(2, 5, 3)
```

✓ 0.0s Python

```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8],
        [ 9, 10, 11],
        [12, 13, 14]],

       [[15, 16, 17],
        [18, 19, 20],
        [21, 22, 23],
        [24, 25, 26],
        [27, 28, 29]]])
```

```
a = data.reshape(5, 6)
a
```

✓ 0.0s Python

```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11],
       [12, 13, 14, 15, 16, 17],
       [18, 19, 20, 21, 22, 23],
       [24, 25, 26, 27, 28, 29]])
```

차원 추가

np.expand_dims(arr, axis=0 | 1)

```
import numpy as np

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

✓ 0.2s

```
(6,)
```

```
b = np.expand_dims(a, axis=1)
b.shape
```

✓ 0.0s

```
(6, 1)
```

```
b
```

✓ 0.0s

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

```
c = np.expand_dims(a, axis=0)
c.shape
```

✓ 0.0s

```
(1, 6)
```

```
c
```

✓ 0.0s

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

인덱싱과 슬라이싱

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

✓ 0.0s

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

```
data[1]
```

✓ 0.0s

```
2
```

```
data[0:2]
```

✓ 0.0s

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

```
data[1:]
```

✓ 0.0s

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

```
data[-2:]
```

✓ 0.0s

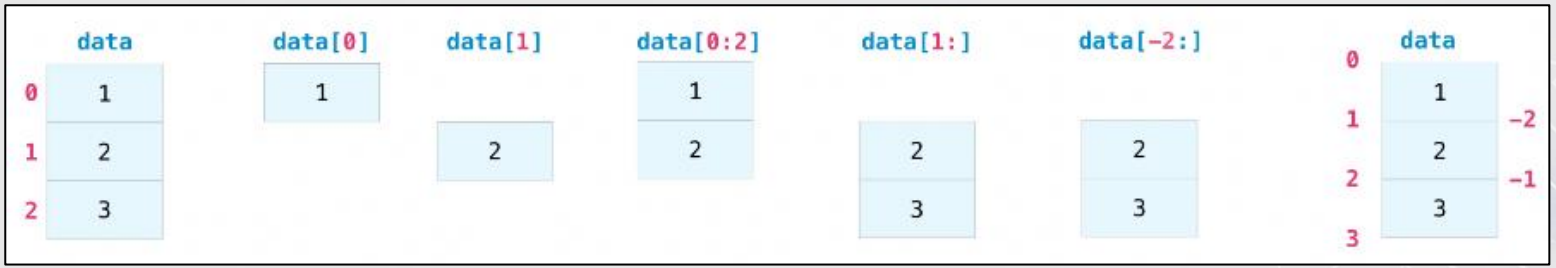
```
array([2, 3])
```

인덱싱과 슬라이싱

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

✓ 0.0s

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



다차원 배열의 인덱싱과 슬라이싱

```
a = np.arange(1, 31).reshape(5, 6)
a
```

✓ 0.0s

```
array([[ 1,  2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11, 12],
       [13, 14, 15, 16, 17, 18],
       [19, 20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29, 30]])
```

```
a[:, :]
```

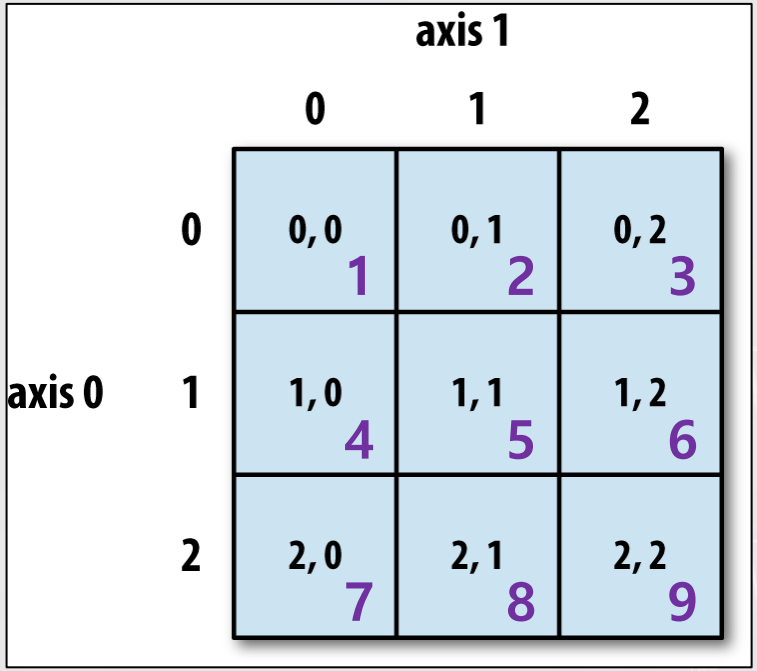
✓ 0.0s

```
array([[ 1,  2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11, 12],
       [13, 14, 15, 16, 17, 18],
       [19, 20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29, 30]])
```

```
a[:2]
```

✓ 0.0s

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



다차원 배열의 인덱싱과 슬라이싱

```
a
```
✓ 0.0s

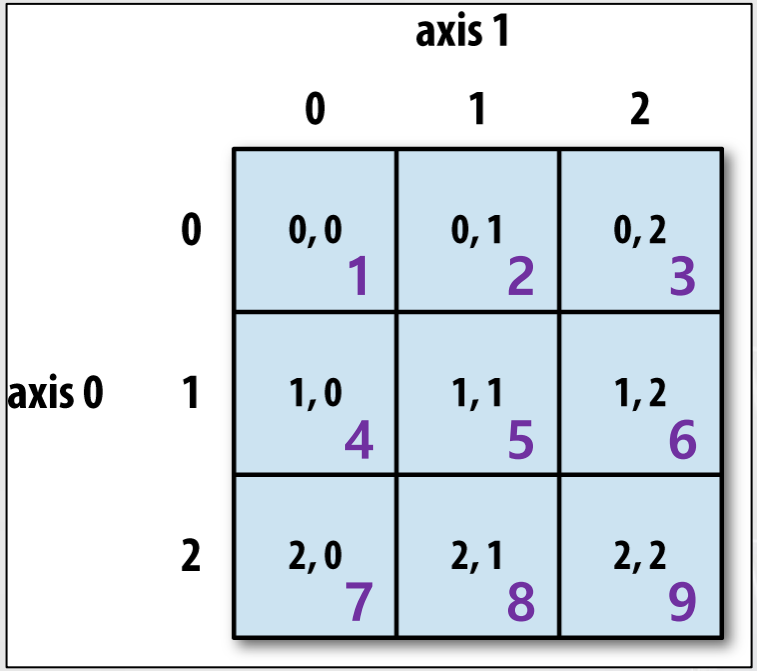
```
array([[ 1,  2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11, 12],
       [13, 14, 15, 16, 17, 18],
       [19, 20, 21, 22, 23, 24],
       [25, 26, 27, 28, 29, 30]])
```

```
a[:, :2]
```
✓ 0.0s

```
array([[ 1,  2],
       [ 7,  8],
       [13, 14],
       [19, 20],
       [25, 26]])
```

```
a[1:3, 2:5]
```
✓ 0.0s

```
array([[ 9, 10, 11],
       [15, 16, 17]])
```



배열 항목의 조건 1/4

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

✓ 0.0s

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

```
a < 5
```

✓ 0.0s

```
array([[ True,  True,  True,  True],
       [False, False, False, False],
       [False, False, False, False]])
```

```
print(a[a < 5])
```

✓ 0.0s

```
[1 2 3 4]
```

배열 항목의 조건 2/4

```
five_up = (a >= 5)
five_up
```

✓ 0.0s

```
array([[False, False, False, False],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True]])
```

```
print(a[five_up])
```

✓ 0.0s

```
[ 5  6  7  8  9 10 11 12]
```

배열 항목의 조건 3/4

```
a
```

✓ 0.0s

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

```
a%2==0
```

✓ 0.0s

```
array([[False,  True, False,  True],
       [False,  True, False,  True],
       [False,  True, False,  True]])
```

```
divisible_by_2 = a[a%2==0]
divisible_by_2
```

✓ 0.0s

```
array([ 2,  4,  6,  8, 10, 12])
```

배열 항목의 조건 4/4

```
a
```

✓ 0.0s

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

```
cond = (a > 2) & (a < 11)
cond
```

```
array([[False, False,  True,  True],
       [ True,  True,  True,  True],
       [ True,  True, False, False]])
```

```
c = a[cond]
c
```

```
array([ 3,  4,  5,  6,  7,  8,  9, 10])
```

SUMMARY

학습정리

배열 Ndarray를 생성하는 함수

- np.arange(), np.linspace()
- np.array(), np.zeros(), np.ones()

배열의 속성과 모양 수정

- arr.ndim, arr.size, arr.shape
- arr.reshape()

배열의 인덱싱과 슬라이싱

- arr[:, :]

배열 항목의 조건

- arr[cond]