

<http://www.numpy.org/>

NumPy Tutorials

2020학년도 2학기

- 📄 NP01_2d_array_creation.py
- 📄 NP02_matrix_mul_1d_array.py
- 📄 NP03_np_array_creation_method.py
- 📄 NP04_array_manipulation.py
- 📄 NP05_random_ndarray_creation.py
- 📄 NP06_basic_operations_functions.py

서경대학교 김진헌

Numpy Quick Start



Table of Contents

- Quickstart tutorial
 - Prerequisites
 - The Basics
 - An example
 - Array Creation
 - Printing Arrays
 - Basic Operations
 - Universal Functions
 - Indexing, Slicing and Iterating
 - Shape Manipulation
 - Changing the shape of an array
 - Stacking together different arrays
 - Splitting one array into several smaller ones
 - Copies and Views
 - No Copy at All
 - View or Shallow Copy
 - Deep Copy
 - Functions and Methods Overview
 - Less Basic
 - Broadcasting rules
 - Advanced indexing and index tricks
 - Indexing with Arrays of Indices
 - Indexing with Boolean Arrays
 - The ix_() function
 - Indexing with strings
 - Linear Algebra
 - Simple Array Operations
 - Tricks and Tips
 - "Automatic" Reshaping
 - Vector Stacking
 - Histograms
 - Further reading

Numpy Dcoumentation

NumPy Documentation

<https://numpy.org/doc/>

[Web](#)

[Reference Guide PDF](#)

[User Guide PDF](#)

[Latest \(development\) documentation](#)

[NumPy Enhancement Proposals](#)

Versions:

[Numpy 1.19 Manual](#)

[\[HTML+zip\]](#) [\[Reference Guide PDF\]](#) [\[User Guide PDF\]](#)

참고: Scipy Lecture Notes

<https://scipy-lectures.org/>

참고:
Scipy에도 Numpy 및
Matplotlib 자료가 있다.

Scipy Lecture Notes

One document to learn numerics, science, and data with Python

Tutorials on the scientific Python ecosystem: a quick introduction to central tools and techniques. The different chapters each correspond to a 1 to 2 hours course with increasing level of expertise, from beginner to expert.

About the scipy lecture notes

[Authors](#) [What's new](#) [License](#) [Contributing](#)

Download

PDF, 2 pages per side

PDF, 1 page per side

HTML and example files

Source code (github)

PDF로
다운로드
가능

1. Getting started with Python for science

- ▶ 1.1. Python scientific computing ecosystem
- ▶ 1.2. The Python language
- ▶ 1.3. NumPy: creating and manipulating numerical data
- ▶ 1.4. Matplotlib: plotting
- ▶ 1.5. Scipy : high-level scientific computing
- 1.6. Getting help and finding documentation

Matplotlib 부분

Numpy 부분

Numpy, The Basics

<https://docs.scipy.org/doc/numpy/index.html>

- NumPy의 목적
 - ▣ 동질의 다차원 배열에 대한 처리 기법 제공
- 동질의 다차원 배열(homogeneous multidimensional array)
 - ▣ 내부 원소를 정수로 인덱싱이 가능하고 내부 요소가 동일한 데이터 타입으로 이루어진 배열 => 행렬(matrix)
- NumPy에서는 차원(dimension)을 축(axis)라고 부르기도 한다.
 - ▣ In the example pictured below, the array has 2 axes.
- The first axis has a length of 2, the second axis has a length of 3.
 - ▣ $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \end{bmatrix}$ # 2x3 행렬(matrix).
 - ▣ 1st axis : row=2, 2nd axis : column = 3

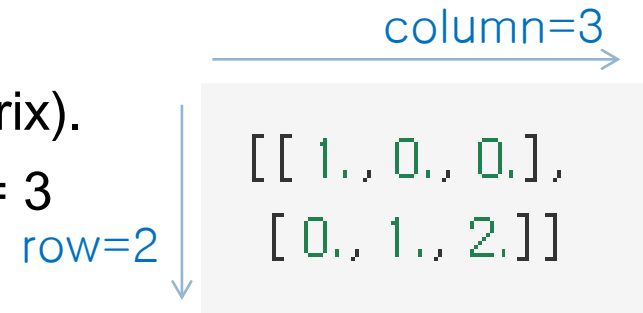


Diagram illustrating a 2x3 matrix A. The matrix is shown as a 2x3 grid of elements: $\begin{bmatrix} 1. & 0. & 0. \\ 0. & 1. & 2. \end{bmatrix}$. A horizontal arrow above the matrix is labeled "column=3", and a vertical arrow to the left of the matrix is labeled "row=2".

ndarray의 생성

NP01_2d_array_creation.py

```
# 1) 2x3 array 선언 =>
```

```
obj1 = [(1, 2, 3), [4, 5, 6]] # a.dtype = int32
```

```
a = np.array(obj1)          # array() 함수로 선언 => ndarray 생성  
print_att('a')
```

```
a=[[1 2 3]
```

```
 [4 5 6]]
```

```
type(a)=<class 'numpy.ndarray'>  numpy로 선언한 객체는 ndarray 클래스이다.
```

```
a.shape=(2, 3)                row x col
```

```
a.ndim=2                      2 dimension
```

```
a.dtype=int32                 각 요소의 데이터 타입. ndarray의 모든 원소(element)는 같은 데이터형
```

```
a.itemsize=4                  한 원소를 표현하는데 소요되는 바이트
```

```
a.size=6                      총 원소의 수
```

ndarray의 속성

- `ndarray.ndim`
 - ▣ the number of axes (dimensions) of the array.
 - ▣ 2x3 행렬의 경우 2 axes(dimensions). `ndim=2`
- `ndarray.shape`
 - ▣ the dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension.
 - ▣ n rows and m columns 의 매트릭스인 경우 `shape = (n,m)`.
 - ▣ The length of the shape tuple is therefore the number of axes, `ndim`.
- `ndarray.size`
 - ▣ the total number of elements of the array. This is equal to the product of the elements of shape.
 - ▣ 2x3 행렬의 경우 `size=6`

ndarray의 속성

- `ndarray.dtype`
 - ▣ 어레이 안의 원소(element)들의 데이터 타입.
 - ▣ Python 표준 : float, int 등
 - ▣ NumPy 제공 : `numpy.int32`, `numpy.int16`, `numpy.float64`, `numpy.uint8` 등
- `ndarray.itemsize`
 - ▣ 요소 한 개의 바이트 길이. float64 타입의 경우 `itemsize = 8(64/8)`,
 - ▣ It is equivalent to `ndarray.dtype.itemsize`.
- `ndarray.data`
 - ▣ the buffer containing the actual elements of the array.
 - ▣ Normally, we won't need to use this attribute because we will access the elements in an array using indexing facilities.

```
a.data= <memory at 0x0000028AADB96708>, type(a.data)= <class 'memoryview'>
```


행렬 곱셈

NP02_matrix_mul_1d_array.py

행렬 $A = \begin{bmatrix} a \\ b \end{bmatrix}$ 와 $B = [c \ d]$ 가 있을 때, 이 둘의 곱셈은 다음과 같다.

$$A \times B = \begin{bmatrix} a \\ b \end{bmatrix} \times [c \ d] = \begin{bmatrix} ac & ad \\ bc & bd \end{bmatrix}$$

$$A^T \times B^T = [a \ b] \times \begin{bmatrix} c \\ d \end{bmatrix} = ac + bd$$

$$B \times A = [c \ d] \times \begin{bmatrix} a \\ b \end{bmatrix} = ca + db$$

$$B^T \times A^T = \begin{bmatrix} c \\ d \end{bmatrix} \times [a \ b] = \begin{bmatrix} ca & cb \\ da & db \end{bmatrix}$$

행렬 $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 와 $B = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$ 가 있을 때, 이 둘의 곱셈은 다음과 같다.

$$A \times B = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

N by M X M by L => N x L

같아야 곱할 수 있다.

ndarray의 속성-예제 프로그램(1/3)

reshape(): NP04_array_manipulation.py

```
a= [0 1 2 3 4]
type(a)= <class 'numpy.ndarray'>
a.shape= (5,) , type(a.shape)= <class 'tuple'>
a.ndim= 1
a.dtype= int32 , type(a.dtype)= <class 'numpy.dtype'>
a.dtype.name= int32, type(a.dtype.name)= <class 'str'>
a.itemsize= 4 각 요소가 4바이트로 구성되어 있다.
a.size= 5 총 요소의 수가 6개이다.
```

```
a = np.arange(5).reshape(5)
# column = 5. ndim= 1
```

```
a= [[0 1 2 3 4]]
type(a)= <class 'numpy.ndarray'>
a.shape= (1, 5) , type(a.shape)= <class 'tuple'>
a.ndim= 2
a.dtype= int32 , type(a.dtype)= <class 'numpy.dtype'>
a.dtype.name= int32, type(a.dtype.name)= <class 'str'>
a.itemsize= 4
a.size= 5
```

```
a = np.arange(5).reshape(1, 5)
# row*column = 1*5. ndim= 2
```

ndarray의 속성-예제 프로그램(2/3)

reshape(): NP04_array_manipulation.py

```
a = [[ 0  1  2  3  4]
      [ 5  6  7  8  9]
      [10 11 12 13 14]]
type(a)= <class 'numpy.ndarray'>
a.shape= (3, 5) , type(a.shape)= <class 'tuple'>
a.ndim= 2
a.dtype= int32 , type(a.dtype)= <class 'numpy.dtype'>
a.dtype.name= int32, type(a.dtype.name)= <class 'str'>
a.itemsize= 4
a.size= 15
```

*a = np.arange(15).reshape(3, 5)
row*column = 3*5. ndim= 2*

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

      [[12 13 14]
       [15 16 17]
       [18 19 20]
       [21 22 23]]]
type(a)= <class 'numpy.ndarray'>
a.shape= (2, 4, 3) , type(a.shape)= <class 'tuple'>
a.ndim= 3
a.dtype= int32 , type(a.dtype)= <class 'numpy.dtype'>
a.dtype.name= int32, type(a.dtype.name)= <class 'str'>
a.itemsize= 4
a.size= 24
```

*a = np.arange(24).reshape(2, 4, 3)
[row*column = 4*3] -> 이런 어레이가 2개. ndim= 3*

ndarray의 속성-예제 프로그램(3/3)

```
a= [[[[ 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]]]
```

```
     [[30 31 32 33 34]
      [35 36 37 38 39]]
```

```
     [[40 41 42 43 44]
      [45 46 47 48 49]]]
```

```
     [[50 51 52 53 54]
      [55 56 57 58 59]]]]
```

```
type(a)= <class 'numpy.ndarray'>
```

```
a.shape= (2, 3, 2, 5) , type(a.shape)= <class 'tuple'>
```

```
a.ndim= 4
```

```
a.dtype= int32 , type(a.dtype)= <class 'numpy.dtype'>
```

```
a.dtype.name= int32, type(a.dtype.name)= <class 'str'>
```

```
a.itemsize= 4
```

```
a.size= 60
```

```
n1=2; n2=3; n3=2; n4=5;
```

```
a = np.arange(n1*n2*n3*n4).reshape(n1, n2, n3, n4)
```

```
# [[row*column = n3*n4] -> 이런 어레이가 n2개 ] ->
```

```
이런 어레이가 n1개. ndim= 4
```