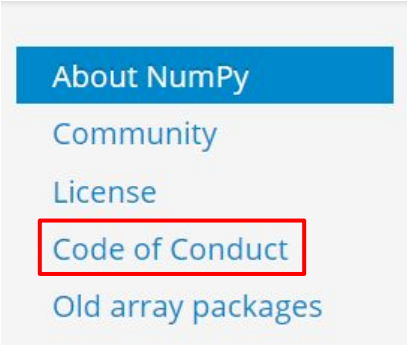
Numpy 특강 part 2

미래연구소 12기 3주차

0. Numpy 등 라이브러리 학습에서 명심할 것

1) 다 외울 수 없습니다.

2) 잘 찾는 능력이 중요합니다.



1> 함수를 처음 봤을 때: shift + tab == np.info('궁금한 함수').

2> 함수 일부 기억 날 듯 말 듯: np.lookfor('궁금한 함수 일부분')

3> 둘 다 하고 싶을 때: numpy 공식 사이트 -> 'code of conduct' 들어가서 검색

4> 정 안 되면 구글링

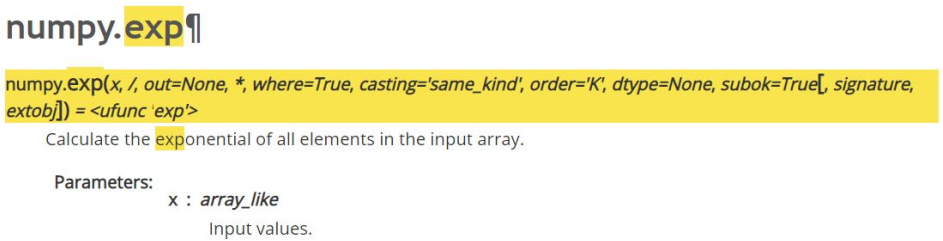
3) 처음 보는 함수를 능숙하게 쓰는 능력도 중요합니다.

1> 찾아서 읽어보고 스스로 사용해보는 습관 (처음이 힘들지 직접 해보면 금방 실력 향상됩니다.)

2> Class, Module, Package등 관련 문법을 잘 모르는 분은 초반에 어려울 수 있습니다. -> 튜터에게 질문

강의자료 ‘Numpy 특강 part 2.ipynb’과 ‘week 2 assignment’ 함께 진행됩니다.

1. Course 1에 나올 numpy 함수 0) Intro



numpy.exp¶

numpy.exp(*x*, /, *out=None*, ***, *where=True*, *casting='same_kind'*, *order='K'*, *dtype=None*, *subok=True*[, *signature*, *extobj*]) = <ufunc 'exp'>

Calculate the exponential of all elements in the input array.

Parameters:

x : *array_like*

Input values.

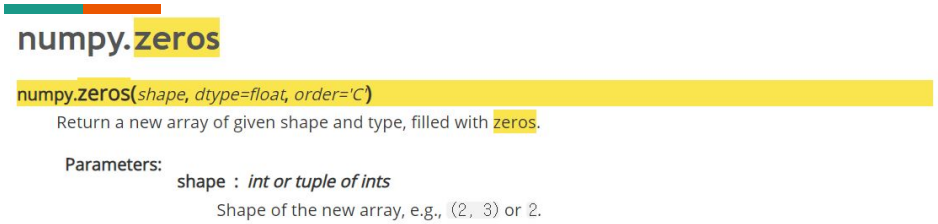
1> 과제를 해보시면서 많은 함수들을 찾아보셨겠지만 검색해보면 그 함수가 복잡해보입니다.

2> 잘 안 쓰니까 **keyword** 방식으로 해놓고 디폴트 값을 사용하게 하는 것입니다.

3> 검색했을 때 나오는 예시 정도만 이해해도 됩니다.

4> 그리고 다 암기하지 않아도 됩니다. 검색해서 보고 사용할 정도만 되면 됩니다.

1. Course 1에 나올 numpy 함수 1) numpy.zeros


numpy.zeros

numpy.zeros(*shape*, *dtype=float*, *order='C'*)

Return a new array of given shape and type, filled with zeros.

Parameters:

shape : *int or tuple of ints*

Shape of the new array, e.g., (2, 3) or 2.

1> np.zeros: 원하는 shape에 맞는 영행렬 생성

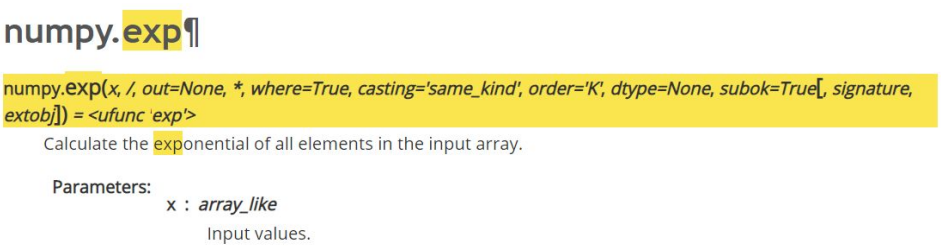
특이사항: shape 부분에

1d array이면 int ex> 3

2d array 이상이면 tuple 자료형을 넣을 것 ex> (3,2)

출처:https://numpy.org/devdocs/reference/generated/numpy.zeros.html?highlight=zeros#numpy.zeros

1. Course 1에 나올 numpy 함수 2) numpy.exp



numpy.exp¶

numpy.exp(*x*, */*, *out=None*, ***, *where=True*, *casting='same_kind'*, *order='K'*, *dtype=None*, *subok=True*[, *signature*, *extobj*]) = <ufunc 'exp'>

Calculate the exponential of all elements in the input array.

Parameters:

x : *array_like*

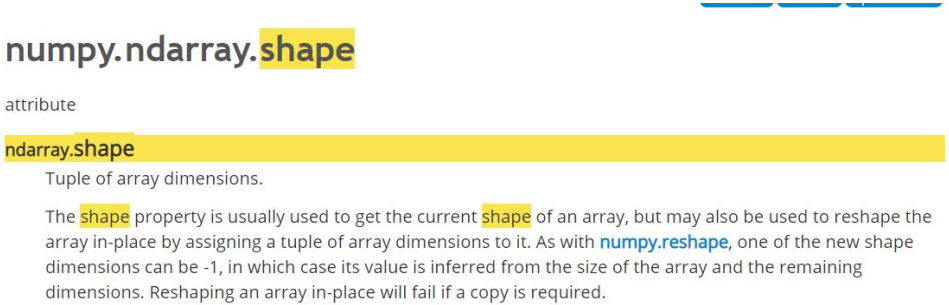
Input values.

1> np.exp: x라는 nd array의 원소들을 e^x로 바꿔줍니다.

2> e란 자연상수를 의미하며 자연 상수의 개념이 크게 중요하지는 않지만 궁금하실 분들을 위해 링크 첨부합니다.
(https://ko.wikipedia.org/wiki/E_(%EC%83%81%EC%88%98))

출처: https://numpy.org/devdocs/reference/generated/numpy.exp.html?highlight=exp#numpy.exp

1. Course 1에 나올 numpy 함수 3) numpy.ndarray.shape



numpy.ndarray.shape

attribute

ndarray.shape

Tuple of array dimensions.

The shape property is usually used to get the current shape of an array, but may also be used to reshape the array in-place by assigning a tuple of array dimensions to it. As with numpy.reshape, one of the new shape dimensions can be -1, in which case its value is inferred from the size of the array and the remaining dimensions. Reshaping an array in-place will fail if a copy is required.

1) X.shape: X라는 nd array의 shape을 return해줍니다.

출처: https://numpy.org/devdocs/reference/generated/numpy.ndarray.shape.html?highlight=shape#numpy.ndarray.shape

1. Course 1에 나올 numpy 함수

4) numpy.arange

numpy.arange

numpy.arange([*start*,]*stop*, [*step*,]*dtype=None*)

Return evenly spaced values within a given interval.

Values are generated within the half-open interval `[start, stop)` (in other words, the interval including *start* but excluding *stop*). For integer arguments the function is equivalent to the Python built-in *range* function, but returns an ndarray rather than a list.

When using a non-integer step, such as 0.1, the results will often not be consistent. It is better to use numpy.linspace for these cases.

Parameters:

start : *number, optional*

Start of interval. The interval includes this value. The default start value is 0.

stop : *number*

End of interval. The interval does not include this value, except in some cases where *step* is not an integer and floating point round-off affects the length of *out*.

step : *number, optional*

Spacing between values. For any output *out*, this is the distance between two adjacent values, `out[i+1] - out[i]`. The default step size is 1. If *step* is specified as a position argument, *start* must also be given.

1> range에서 비롯된 함수입니다.

2> 먼저 **[]**에 대해 배우겠습니다.

3> 그럼 용법이 **3**가지라는 것을 이해할 수 있습니다.

출처: https://numpy.org/devdocs/reference/generated/numpy.arange.html?highlight=arange#numpy.arange

1. Course 1에 나올 numpy 함수 4) numpy.arange

case 1 : np.arange(stop)

0부터 시작해서 stop 직전의 수까지 1d array로 만들어진 array

1> 0부터 시작해서 stop 직전의 수까지 1d array로 만들어진 array

```
np.arange(12)
```

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

case 2 : np.arange(start, stop)

start부터 시작해서 stop 직전의 수까지 1d array로 만들어진 array

2> start부터 시작해서 1> stop 직전의 수까지 1d array로 만들어진 array

```
np.arange(3,12)
```

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

case 3 : np.arange(start, stop, step)

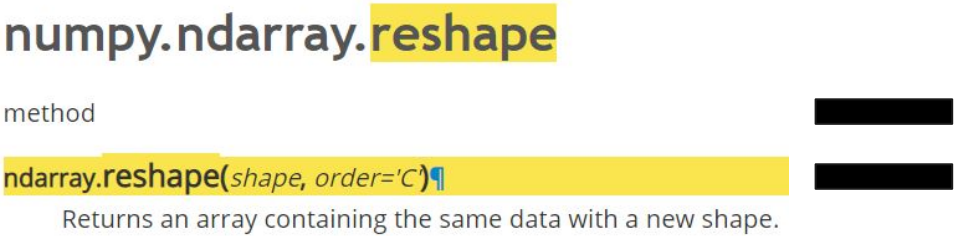
start부터 시작해서 stop 직전의 수까지 step씩 증가하는 수를 1d array로 만든다.

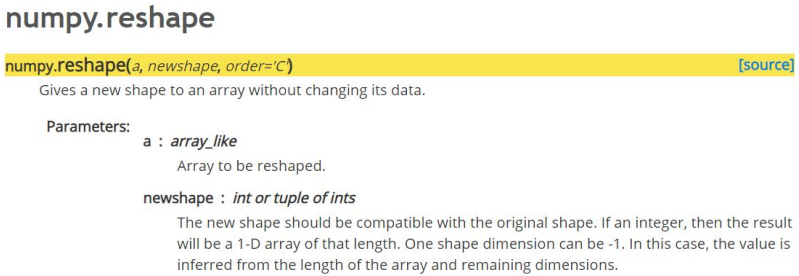
2> start부터 시작해서 1> stop 직전의 수까지 3> step씩 증가하는 수를 1d array로 만든다.

```
np.arange(3,12,3)
```

```
array([3, 6, 9])
```

1. Course 1에 나올 numpy 함수 5) numpy.ndarray.reshape





1> X.reshape: X라는 nd array를 내가 원하는 shape에 맞게 재배열해줍니다.

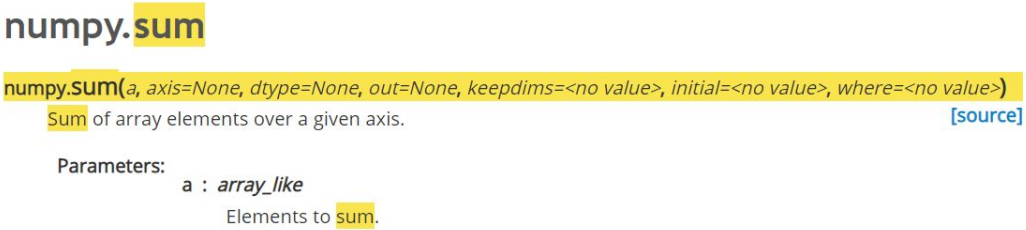
(출처의 설명만으로는 이해하기 어렵지만 출처의 예시까지도 읽어보시면 충분히 이해할 수 있습니다.

2> 첨부된 사진에 있는 두 함수는 같은 함수입니다.
Instance.method(argument) 형식 = Class.method(Instance, argument) 형식
(참고: https://wikidocs.net/28#_6)

3> -1의 의미 = 미지수

출처: https://numpy.org/devdocs/reference/generated/numpy.reshape.html#numpy.reshape

1. Course 1에 나올 numpy 함수 6) numpy.sum



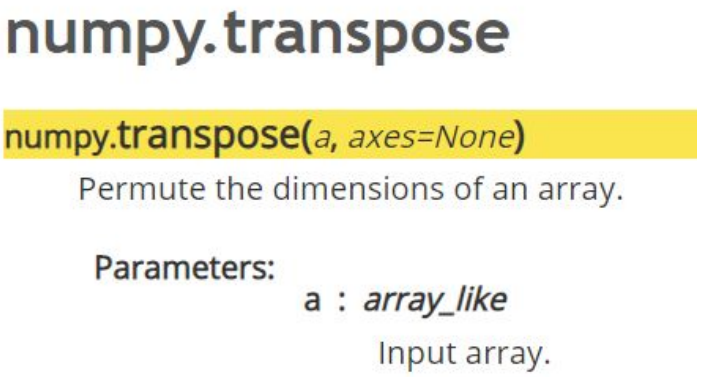
1> np.sum: x라는 nd array의 원소의 합을 구해준다.

2> 하지만 **axis** 개념과 **keepdims** 개념은 중요합니다. 어려운 내용이니 다음 주에 정리하겠습니다.

출처:https://numpy.org/devdocs/reference/generated/numpy.sum.html?highlight=sum#numpy.sum

1. Course 1에 나올 numpy 함수 7) numpy.ndarray.T

=

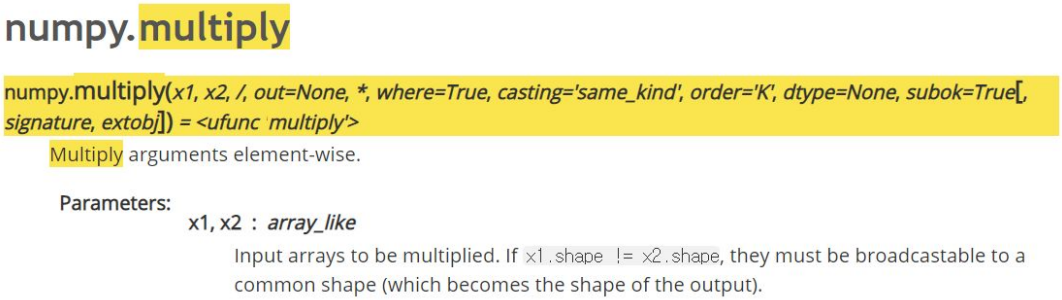


1> X.T: X라는 nd array의 transpose를 구해준다.

2> 보통 X.T 꼴을 많이 사용한다. (np.transpose(X) 대신에)

출처:https://numpy.org/devdocs/reference/generated/numpy.transpose.html#numpy.transpose

1. Course 1에 나올 numpy 함수 8) numpy.multiply



numpy.multiply

numpy.multiply(*x1, x2, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, subok=True[, signature, extobj]*) = <ufunc 'multiply'>

Multiply arguments element-wise.

Parameters:

x1, x2 : *array_like*

Input arrays to be multiplied. If `x1.shape != x2.shape`, they must be broadcastable to a common shape (which becomes the shape of the output).

1> np.multiply: x1과 x2라는 nd array끼리 element-wise product

(위치가 같은 원소끼리 곱셈을 한다.)

출처: https://numpy.org/devdocs/reference/generated/numpy.multiply.html?highlight=multiply#numpy.multiply

2. Broadcasting 0) Intro

1> 많이 나오는 개념은 아니지만

2> 코드를 보고 **broadcasting**이 적용되었는지만 파악할 수 있으면 됩니다.

2. Broadcasting 1) 정의

0> broadcast: ‘퍼뜨리다’ -> ‘방송하다’

1> 행렬 연산 시 shape이 안 맞아도

2> 연산할 때 shape이 작은 array가 알아서 큰 array의 shape에 맞게

3> 확장되는 것(‘퍼뜨리다’라는 어원과 어울린다.)

2. Broadcasting 2) 경우의 수 4가지

		기준 **array**	**확장 array**
1	**조합**	nd array (1이상)	scalar
	shape	?	()
2	**조합**	2d array	1d array
	shape	(a, x)	(x,)
3	**조합**	2d array	2d array
	shape	(a,x) / (x, a)	(1, x) / (x, 1)
4	**조합**	nd array (3이상)	2d array
	shape	(...., x, y)	(x,y)

1> 표에서 하단부는 **shape**을 의미합니다.

2> 사칙 연산에 한해 가능

3> 해당 표보다 강의 자료로 보는 게 이해하기 쉽습니다.

4> 3차원 **broadcasting** 은 거의 다루지 않으므로 **4**번 **case**는 몰라도 됩니다.

5> 2번과 **4**번은 서로의 끝과 앞이 맞으면 됩니다.

2. Broadcasting 3) 특징

1> 사람마다 분류하는 경우의 수는 다르고 이거 말고 다른 경우의 수도 있다.

2> 경우의 수를 다 외우지 않아도 됩니다.

3> 중요한 건 **shape**이 안 맞는데 연산이 되었을 때,
=> broadcasting인지만 알면 됩니다.

4> 사용하는 이유
1 shape 일일이 맞추기 귀찮고
2 코드도 줄어든다.
3 연산할 때만 늘어나 메모리 측면에서도 좋다.