Python

[Python] Numpy 배열 합치기(concatenate)

2020. 10. 12. 22:31 수정 삭제 공개

배열의 연결(concatenate)

- hstack (배열을 옆으로 연결)

- vstack (배열을 수직으로 연결)

- dstack (차원으로 연결)

- stack (차원을 수직으로 연결)

- r_ (배열을 하나로 합치기)

- c_ (차원으로 합치기)

- tile(반복을 통해 배열을 만들기)

```
In [22]: def aryInfo(ary):
             print('type : {}'.format(type(ary)))
             print('shape : {}'.format(ary.shape))
             print('dimension : {}'.format(ary.ndim))
             print('dtype : {}'.format(ary.dtype))
             print('Array Data : ₩n',ary)
```

이건 변수의 상태를 보기 위해 만들었다.

hstack(배열을 옆으로 붙인다.)

```
In [29]: # hstack 행의 수가 같은 두 개 이상의 배열은 옆으로 연결 열의 수가 더 많은 배열을 만들 때
         h_arr01 = np.ones((2,3))
         aryInfo(h_arr01)
         print("*"*50)
         h_arr02 = np.zeros((2,2))
         aryInfo(h_arr02)
         print(h_arr02)
         print("*"*50)
         h_arr02=np.hstack([h_arr01,h_arr02])
         aryInfo(h_arr02)
```

```
type : <class 'numpy.ndarray'>
shape : (2, 3)
dimension : 2
dtype : float64
Array Data :
 [[1. 1. 1.]
 [1. 1. 1.]]
**************************************************
type : <class 'numpy.ndarray'>
shape : (2, 2)
dimension : 2
dtype : float64
Array Data :
 [[0. 0.]
 [0. 0.]]
[[0. 0.]
 [0. 0.]]
**************************************************
type : <class 'numpy.ndarray'>
shape : (2, 5)
dimension : 2
dtype : float64
Array Data :
 [[1. 1. 1. 0. 0.]
 [1. 1. 1. 0. 0.]]
```

hstack 행의 수가 같은 두 개 이상의 배열은 옆으로 연결 열의 수가 더 많은 배열을 만들 때 사용한다.
hstack을 이용해 첫 번째 h_arr01과 h_arr02를 합쳤다. 이 때 꼭 행의 개수가 같이야 가능하다. 2,3과 2,2를 합쳐 2,5가 되는 행렬이 된다.

#vstack (수직으로 행을 추가로 연결한다.)

In [30]:
```
# vstack 열의 수가 같은 두 개 이상의 배열은 옆으로 연결 열의 수가 더 많은 배열을 만들 때
h_arr01 = np.ones((2,3))
aryInfo(h_arr01)
print("*"*50)
h_arr02 = np.zeros((3,3))
aryInfo(h_arr02)
print(h_arr02)
print("+"*50)
h_arr02=np.vstack([h_arr01,h_arr02])
aryInfo(h_arr02)
```

```
type : <class 'numpy.ndarray'>
shape : (2, 3)
dimension : 2
dtype : float64
Array Data :
 [[1. 1. 1.]
 [1. 1. 1.]]
**************************************************
type : <class 'numpy.ndarray'>
shape : (3, 3)
dimension : 2
dtype : float64
Array Data :
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
++++++++++++++++++++++++++++++++++++++++++++++++++
type : <class 'numpy.ndarray'>
shape : (5, 3)
dimension : 2
dtype : float64
Array Data :
 [[1. 1. 1.]
 [1. 1. 1.]
 [0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

vstack 열의 수가 같은 두 개 이상의 배열은 아래로 연결 / 행의 수가 더 많은 배열을 만들 때 사용한다.
vstack을 이용해 첫번째 h_arr01과 h_arr02를 합쳤다. 이 때 꼭 열의 개수가 같이야 가능하다. 2,3과 3,3을 합쳐 5,3가 되는 행렬이 된다.

#dstack (차원을 쪼개서 연결한다.)

```
In [33]: # shape(3,4) -> 1개의 3차원 (3,4,2)
h_arr01 = np.ones((3,4))
aryInfo(h_arr01)
print("+"*50)
h_arr02 = np.zeros((3,4))
aryInfo(h_arr02)
print(h_arr02)
print("+"*50)

aryInfo(np.dstack([h_arr01,h_arr02]))
```

```
type : <class 'numpy.ndarray'>
shape : (3, 4)
dimension : 2
dtype : float64
Array Data :
 [[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
**************************************************
type : <class 'numpy.ndarray'>
shape : (3, 4)
dimension : 2
dtype : float64
Array Data :
 [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
**************************************************
type : <class 'numpy.ndarray'>
shape : (3, 4, 2)
dimension : 3
dtype : float64
Array Data :
 [[[1. 0.]
  [1. 0.]
  [1. 0.]
  [1. 0.]]

 [[1. 0.]
  [1. 0.]
  [1. 0.]
  [1. 0.]]

 [[1. 0.]
  [1. 0.]
  [1. 0.]
  [1. 0.]]]
```

dstack은 차원을 늘리는데 사용된다.
위와 같이 3차원에 4행 2열짜리 행렬을 만들어준다. dstack([h_arr01, h_arr02]) 를 통해 차원을 늘려준다. 이건 각자 가지고 있는 것을 쪼개서 만들어준다고 생각하면 된다.

다른 방식의 차원을 늘리는 방법도 있다.

#stack (차원을 수직으로 추가로 연결한다.)

```
In [34]: aryInfo(np.stack([h_arr01,h_arr02]))
```

```
type : <class 'numpy.ndarray'>
shape : (2, 3, 4)
dimension : 3
dtype : float64
Array Data :
 [[[1. 1. 1. 1.]
  [1. 1. 1. 1.]
  [1. 1. 1. 1.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]]
```

np.stack을 사용하여 각기 다른 두개를 아래에 넣어주는 방식으로 차원을 2차원의 3행 4열을 만들어준다.

인덱서(indexer)

r_

```
In [39]: aryInfo(np.r_[np.array([1,2,3]), np.array([4,5,6])])

type : <class 'numpy.ndarray'>
shape : (6,)
dimension : 1
dtype : int32
Array Data :
 [1 2 3 4 5 6]
```

r_ 은 두가지 np를 하나로 합치는 역할을 한다.

c_

```
In [40]: aryInfo(np.c_[np.array([1,2,3]), np.array([4,5,6])])

type : <class 'numpy.ndarray'>
shape : (3, 2)
dimension : 2
dtype : int32
Array Data :
 [[1 4]
 [2 5]
 [3 6]]
```

c_ 같은 경우에는 2가지를 2차원으로 만드는 기능을 가지고 있다.

tile

```
In [44]: aryInfo(np.tile(np.array([[1,2,3],[4,5,6]]),(3,2)))

type : <class 'numpy.ndarray'>
shape : (6, 6)
dimension : 2
dtype : int32
Array Data :
 [[1 2 3 1 2 3]
 [4 5 6 4 5 6]
 [1 2 3 1 2 3]
 [4 5 6 4 5 6]
 [1 2 3 1 2 3]
 [4 5 6 4 5 6]]
```

같은 요소의 배열을 반복해서 사용할 때 사용한다. 행을 3번 열을 2번 반복한다. 그래서 123123으로 열이 두 번 반복되고 이 개수가 3개가 나오는 것이다.

'**Python**' 카테고리의 다른 글

[Python] Numpy를 통한 배열 연산

[Python] Numpy의 배열 행 열 삭제

[Python] Numpy 배열 합치기(concatenate)

[Python] Numpy의 reshape 통한 차원 변경(재배열)

[Python] Numpy에 있는 다양한 함수 사용법 - 2(전치행렬,zeors,ones, iterator,...

[Python] Numpy를 통한 배열 indexing(Boolen indexing, fancy indexing)

Numpy 배열 결합 Numpy 배열 연결 Numpy 배열 합치기

나무늘보스

혼자 끄적끄적하는 블로그 입니다.