numpy-lecture-5

June 1, 2023

0.1 Numpy Lecture - 5

[1]:
```
import numpy as np
```

[2]:
```
a = np.arange(9)
a
```

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

[3]:
```
np.split(a, 3)
```

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

[4]:
```
np.split(a, 4)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [4], in <cell line: 1>()
----> 1 np.split(a, 4)

File <__array_function__ internals>:180, in split(*args, **kwargs)

File /usr/local/lib/python3.9/site-packages/numpy/lib/shape_base.py:872, in␣
 ↪split(ary, indices_or_sections, axis)
    870     N = ary.shape[axis]
    871     if N % sections:
--> 872         raise ValueError(
    873             'array split does not result in an equal division') from No e
    874 return array_split(ary, indices_or_sections, axis)

ValueError: array split does not result in an equal division
```

[5]:
```
np.split(a, [3, 5, 6])
```

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

[6]:
```
a = np.arange(1, 17).reshape(4, 4)
a
```

[6]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

[7]:
```
np.split(a, 2, axis=1)
```

[7]:
```
[array([[ 1,  2],
        [ 5,  6],
        [ 9, 10],
        [13, 14]]),
 array([[ 3,  4],
        [ 7,  8],
        [11, 12],
        [15, 16]])]
```

[8]:
```
np.split(a, 2, axis=0)
```

[8]:
```
[array([[1, 2, 3, 4],
        [5, 6, 7, 8]]),
 array([[ 9, 10, 11, 12],
        [13, 14, 15, 16]])]
```

[9]:
```
np.hsplit(a, 2)
```

[9]:
```
[array([[ 1,  2],
        [ 5,  6],
        [ 9, 10],
        [13, 14]]),
 array([[ 3,  4],
        [ 7,  8],
        [11, 12],
        [15, 16]])]
```

[11]:
```
np.vsplit(a, 2)
```

[11]:
```
[array([[1, 2, 3, 4],
        [5, 6, 7, 8]]),
 array([[ 9, 10, 11, 12],
        [13, 14, 15, 16]])]
```

0.1.1 Stacking

[12]:
```
a = np.arange(5)
```

[13]:
```
np.vstack((a, a, a))
```

[13]:
```
array([[0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4]])
```

[14]:
```
np.vstack()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [14], in <cell line: 1>()
----> 1 np.vstack()

File <__array_function__ internals>:179, in vstack(*args, **kwargs)

TypeError: _vhstack_dispatcher() missing 1 required positional argument: 'tup'
```

[ ]:
```
np.hstack((a, a, a))
```

[15]:
```
a = np.array([0, 10, 5, 3])
np.sort(a)
```

[15]:
```
array([ 0,  3,  5, 10])
```

[16]:
```
a
```

[16]:
```
array([ 0, 10,  5,  3])
```

[17]:
```
a.sort()
```

[18]:
```
a
```

[18]:
```
array([ 0,  3,  5, 10])
```

[19]:
```
np.hstack((a, a, a))
```

[19]:
```
array([ 0,  3,  5, 10,  0,  3,  5, 10,  0,  3,  5, 10])
```

[20]:
```
a = np.arange(5)
a
```

[20]:
```
array([0, 1, 2, 3, 4])
```

[21]:
```
a = a.reshape(5, 1)
a
```

[21]:
```
array([[0],
       [1],
       [2],
       [3],
       [4]])
```

[22]:
```
np.hstack((a, a, a))
```

[22]:
```
array([[0, 0, 0],
       [1, 1, 1],
       [2, 2, 2],
       [3, 3, 3],
       [4, 4, 4]])
```

[23]:
```
help(np.concatenate)
```

```
Help on function concatenate in module numpy:

concatenate(…)
    concatenate((a1, a2, …), axis=0, out=None, dtype=None,
casting="same_kind")

    Join a sequence of arrays along an existing axis.

    Parameters
    ----------
    a1, a2, … : sequence of array_like
        The arrays must have the same shape, except in the dimension
        corresponding to `axis` (the first, by default).
    axis : int, optional
        The axis along which the arrays will be joined.  If axis is None,
        arrays are flattened before use.  Default is 0.
    out : ndarray, optional
        If provided, the destination to place the result. The shape must be
        correct, matching that of what concatenate would have returned if no
        out argument were specified.
    dtype : str or dtype
        If provided, the destination array will have this dtype. Cannot be
        provided together with `out`.

        .. versionadded:: 1.20.0

    casting : {'no', 'equiv', 'safe', 'same_kind', 'unsafe'}, optional
        Controls what kind of data casting may occur. Defaults to 'same_kind'.
```

```
    .. versionadded:: 1.20.0

Returns
-------
res : ndarray
    The concatenated array.

See Also
--------
ma.concatenate : Concatenate function that preserves input masks.
array_split : Split an array into multiple sub-arrays of equal or
              near-equal size.
split : Split array into a list of multiple sub-arrays of equal size.
hsplit : Split array into multiple sub-arrays horizontally (column wise).
vsplit : Split array into multiple sub-arrays vertically (row wise).
dsplit : Split array into multiple sub-arrays along the 3rd axis (depth).
stack : Stack a sequence of arrays along a new axis.
block : Assemble arrays from blocks.
hstack : Stack arrays in sequence horizontally (column wise).
vstack : Stack arrays in sequence vertically (row wise).
dstack : Stack arrays in sequence depth wise (along third dimension).
column_stack : Stack 1-D arrays as columns into a 2-D array.

Notes
-----
When one or more of the arrays to be concatenated is a MaskedArray,
this function will return a MaskedArray object instead of an ndarray,
but the input masks are *not* preserved. In cases where a MaskedArray
is expected as input, use the ma.concatenate function from the masked
array module instead.

Examples
--------
>>> a = np.array([[1, 2], [3, 4]])
>>> b = np.array([[5, 6]])
>>> np.concatenate((a, b), axis=0)
array([[1, 2],
       [3, 4],
       [5, 6]])
>>> np.concatenate((a, b.T), axis=1)
array([[1, 2, 5],
       [3, 4, 6]])
>>> np.concatenate((a, b), axis=None)
array([1, 2, 3, 4, 5, 6])

This function will not preserve masking of MaskedArray inputs.
```

```
>>> a = np.ma.arange(3)
>>> a[1] = np.ma.masked
>>> b = np.arange(2, 5)
>>> a
masked_array(data=[0, --, 2],
             mask=[False,  True, False],
       fill_value=999999)
>>> b
array([2, 3, 4])
>>> np.concatenate([a, b])
masked_array(data=[0, 1, 2, 2, 3, 4],
             mask=False,
       fill_value=999999)
>>> np.ma.concatenate([a, b])
masked_array(data=[0, --, 2, 2, 3, 4],
             mask=[False,  True, False, False, False, False],
       fill_value=999999)
```

[24]:
```
np.concatenate((a, a, a), axis=1)
```

[24]:
```
array([[0, 0, 0],
       [1, 1, 1],
       [2, 2, 2],
       [3, 3, 3],
       [4, 4, 4]])
```

0.2 BroadCasting

[25]:
```
a = np.arange(0, 40, 10)
a
```

[25]:
```
array([ 0, 10, 20, 30])
```

[26]:
```
np.vstack((a, a, a))
```

[26]:
```
array([[ 0, 10, 20, 30],
       [ 0, 10, 20, 30],
       [ 0, 10, 20, 30]])
```

[27]:
```
np.tile(a, (3, 2))
```

[27]:
```
array([[ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30],
       [ 0, 10, 20, 30,  0, 10, 20, 30]])
```

[29]:
```
a = np.tile(a, (3, 1))
```

[30]:
```
a = a.T
a
```

[30]:
```
array([[ 0,  0,  0],
       [10, 10, 10],
       [20, 20, 20],
       [30, 30, 30]])
```

[31]:
```
b = np.tile(np.arange(3), (4, 1))
b
```

[31]:
```
array([[0, 1, 2],
       [0, 1, 2],
       [0, 1, 2],
       [0, 1, 2]])
```

[32]:
```
a + b
```

[32]:
```
array([[ 0,  1,  2],
       [10, 11, 12],
       [20, 21, 22],
       [30, 31, 32]])
```

[33]:
```
c = np.arange(3)
```

[34]:
```
a + c
```

[34]:
```
array([[ 0,  1,  2],
       [10, 11, 12],
       [20, 21, 22],
       [30, 31, 32]])
```

Rule 1 : If two array differ in the number of dimensions, the shape of one with fewer dimensions is padded with ones on its leading(Left Side).

Rule 2 : If the shape of two arrays doesnt match in any dimensions, the array with shape equal to 1 is stretched to match the other shape.

[35]:
```
a = np.arange(8).reshape(2, 4)
b = np.arange(16).reshape(4, 4)
a, b
```

[35]:
```
(array([[0, 1, 2, 3],
        [4, 5, 6, 7]]),
 array([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11],
        [12, 13, 14, 15]]))
```

[36]:
```
a + b
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [36], in <cell line: 1>()
----> 1 a + b

ValueError: operands could not be broadcast together with shapes (2,4) (4,4)
```

[37]:
```
a = np.arange(1, 10).reshape(3, 3)
b = np.array([-1,0, 1])
a + b
```

[37]:
```
array([[ 0,  2,  4],
       [ 3,  5,  7],
       [ 6,  8, 10]])
```

[38]:
```
a = np.arange(1, 10).reshape(3, 3)
b = np.arange(3, 10, 3).reshape(3, 1)
b
```

[38]:
```
array([[3],
       [6],
       [9]])
```

[39]:
```
a + b
```

[39]:
```
array([[ 4,  5,  6],
       [10, 11, 12],
       [16, 17, 18]])
```

[42]:
```
j = 2
a = np.array([[5, 3, 9], [2, 1, 4], [7, 6, 8]])
indices_array = a[:, (j-1)].argsort()
indices_array
```

[42]:
```
array([1, 0, 2])
```

[43]:
```
a[indices_array]
```

[43]:
```
array([[2, 1, 4],
       [5, 3, 9],
       [7, 6, 8]])
```

[44]:
```
a = np.arange(24).reshape(2, 3, 4)
a
```

[44]:
```
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]]])
```

[45]:
```
np.vstack((a, a, a))
```

[45]:
```
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]],

       [[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]],

       [[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

[ ]:

[ ]:

[ ]: