NUMPY

Outline：

1. Introduction to Numpy
2. Axes, indexing, slicing
3. “Vectorized” Operations
4. Broadcasting

1 Introduction to Numpy

1.1 Numpy

NumPy is the fundamental **package** for scientific computing with Python.

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

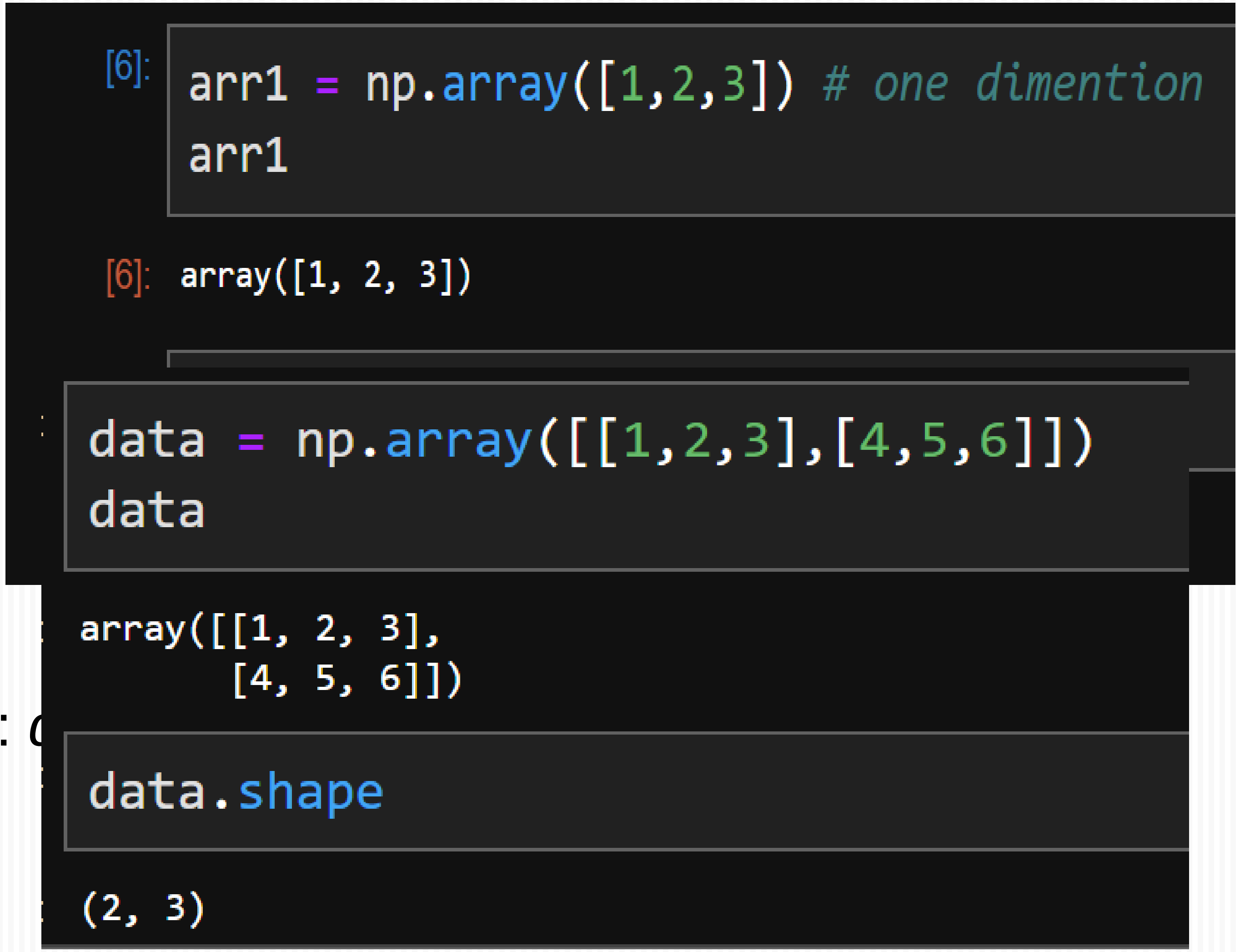
1 Introduction to Numpy

1.2 ndarray

an N-dimensional arra

- multi-dimensional:

- homogeneous data: (



```
[6]: arr1 = np.array([1,2,3]) # one dimention
     arr1
```

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

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

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

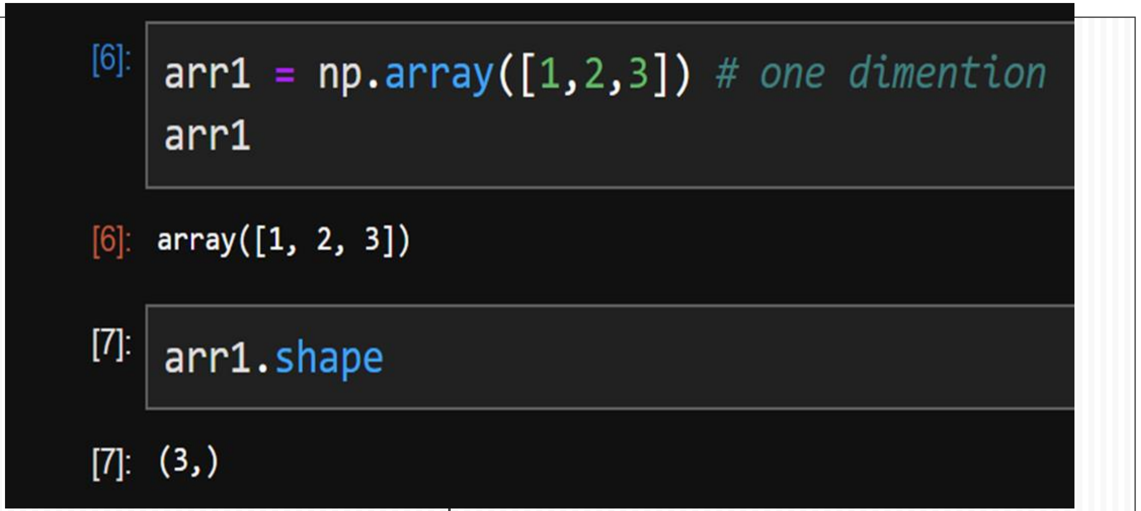
```
data.shape
```

```
(2, 3)
```

Introduction Numpy

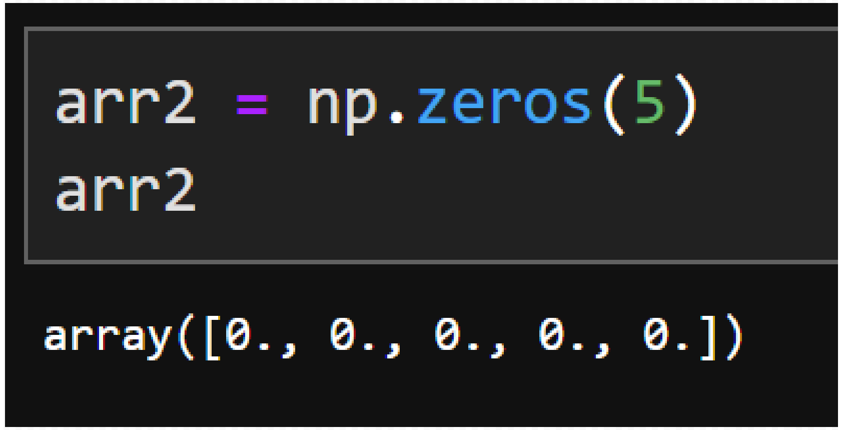
1.3 Create a ndarray object

- np.array()

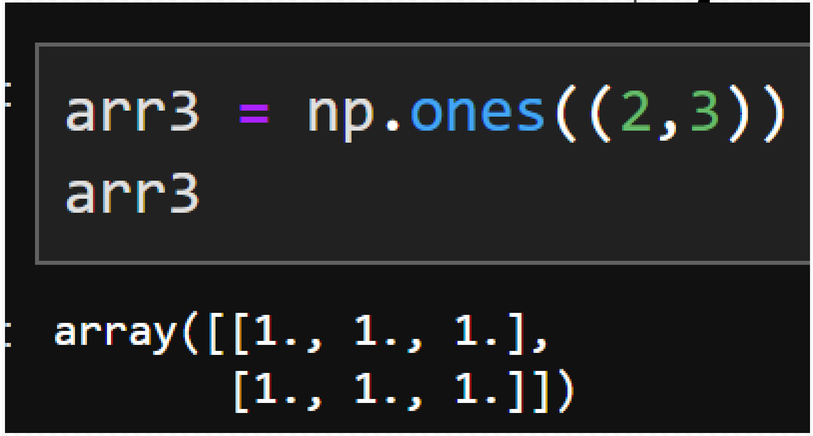


```
[6]: arr1 = np.array([1,2,3]) # one dimention
     arr1

[6]: array([1, 2, 3])

[7]: arr1.shape

[7]: (3,)
```

•np.zeros(); np.ones()



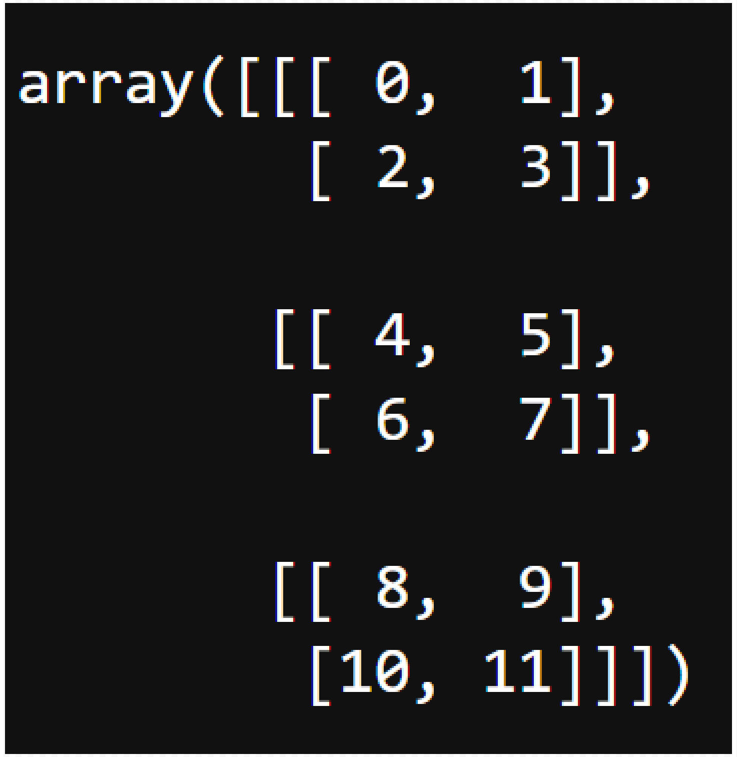
```
arr2 = np.zeros(5)
arr2

array([0., 0., 0., 0., 0.])
```



```
arr3 = np.ones((2,3))
arr3

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

2 Axes, indexing, slicing

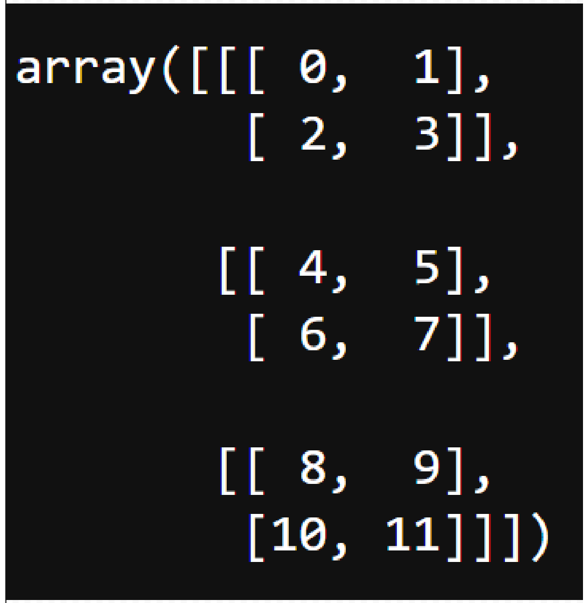
N-dimensional array



```
array([[[ 0,  1],
        [ 2,  3]],

       [[ 4,  5],
        [ 6,  7]],

       [[ 8,  9],
        [10, 11]]])
```

2 Axes, indexing, slicing

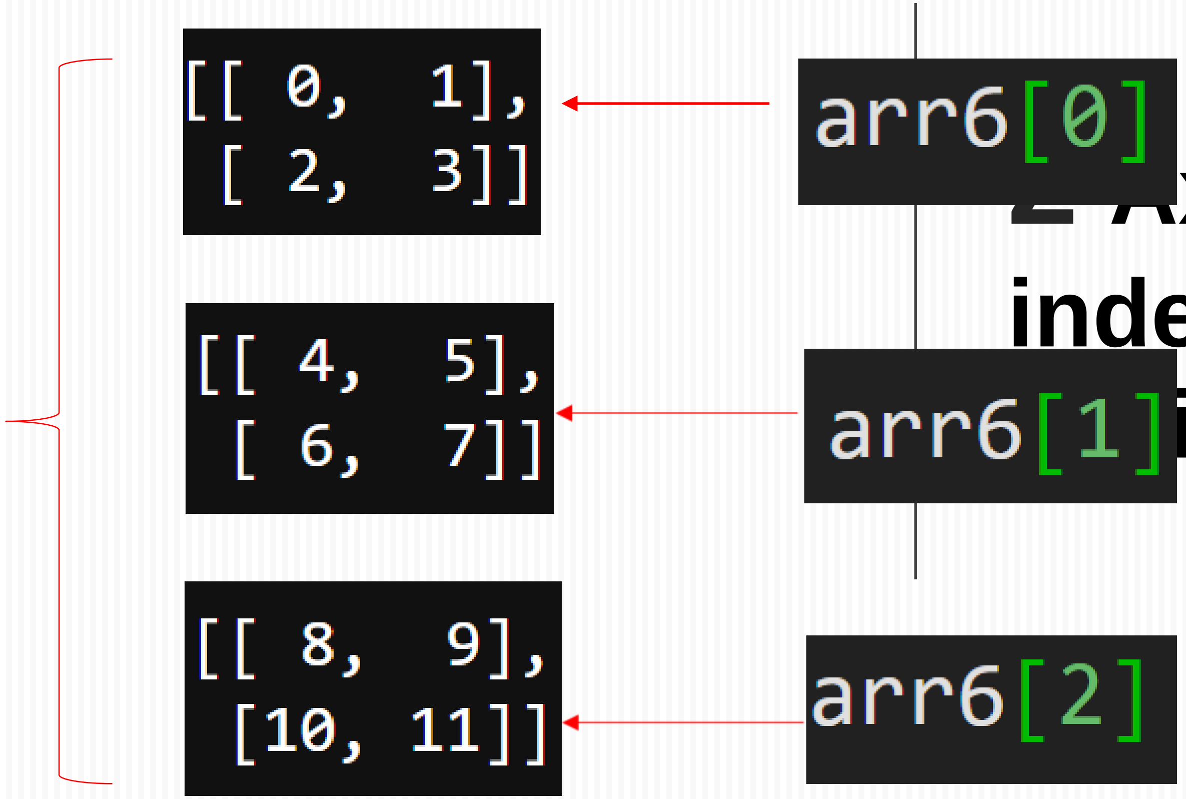
N-dimensional array



```
array([[[ 0,  1],
        [ 2,  3]],

       [[ 4,  5],
        [ 6,  7]],

       [[ 8,  9],
        [10, 11]]])
```

Axis 0

Indexing



```
[[ 0,  1],
 [ 2,  3]]
```

```
[[ 4,  5],
 [ 6,  7]]
```

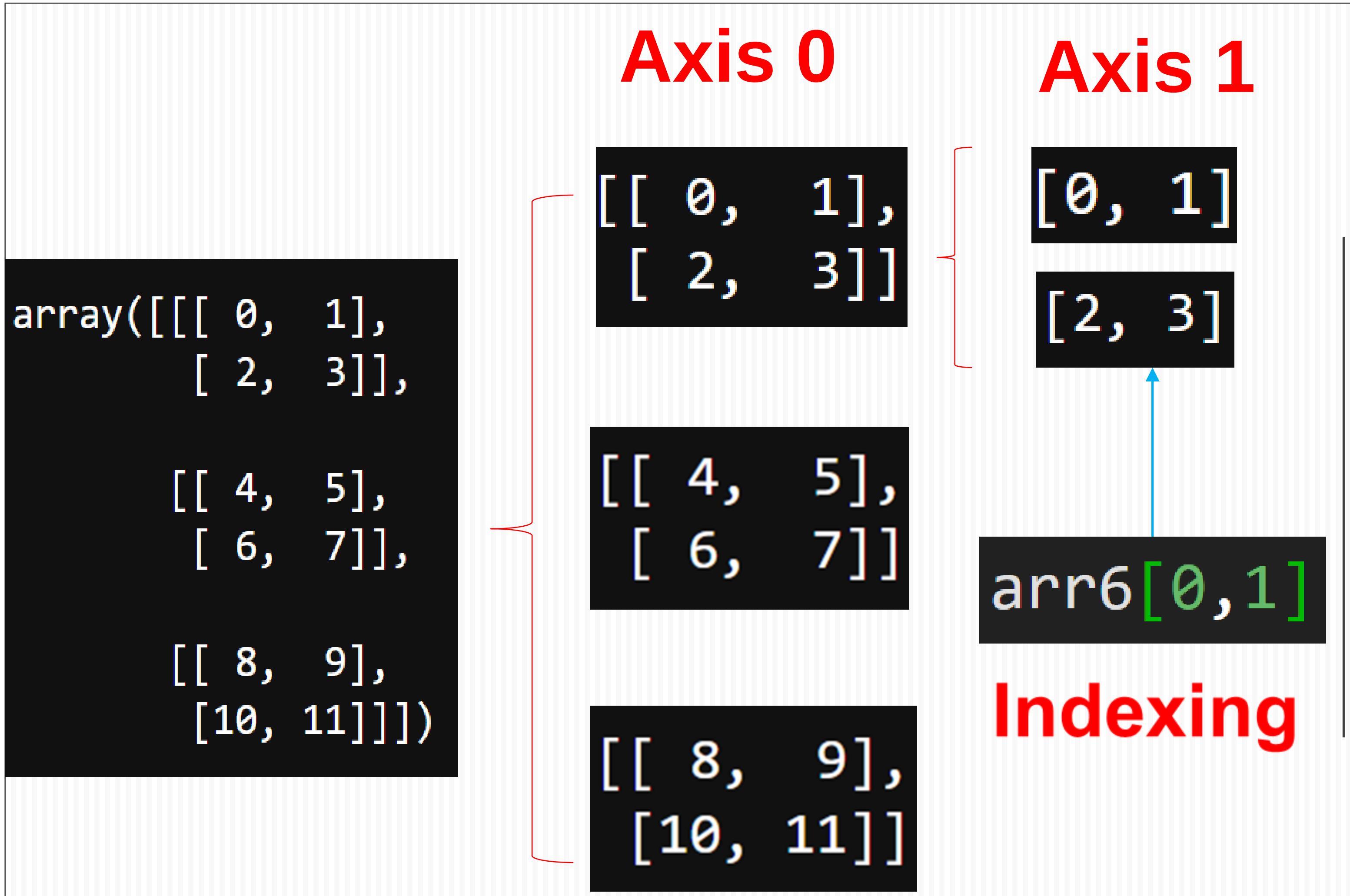
```
[[ 8,  9],
 [10, 11]]
```

```
arr6[0]
```

```
arr6[1]
```

```
arr6[2]
```

2 Axes,
indexing,
ing



```
array([[[ 0,  1],
        [ 2,  3]],

       [[ 4,  5],
        [ 6,  7]],

       [[ 8,  9],
        [10, 11]]])
```

```
[[ 0,  1],
 [ 2,  3]]
```

```
[[ 4,  5],
 [ 6,  7]]
```

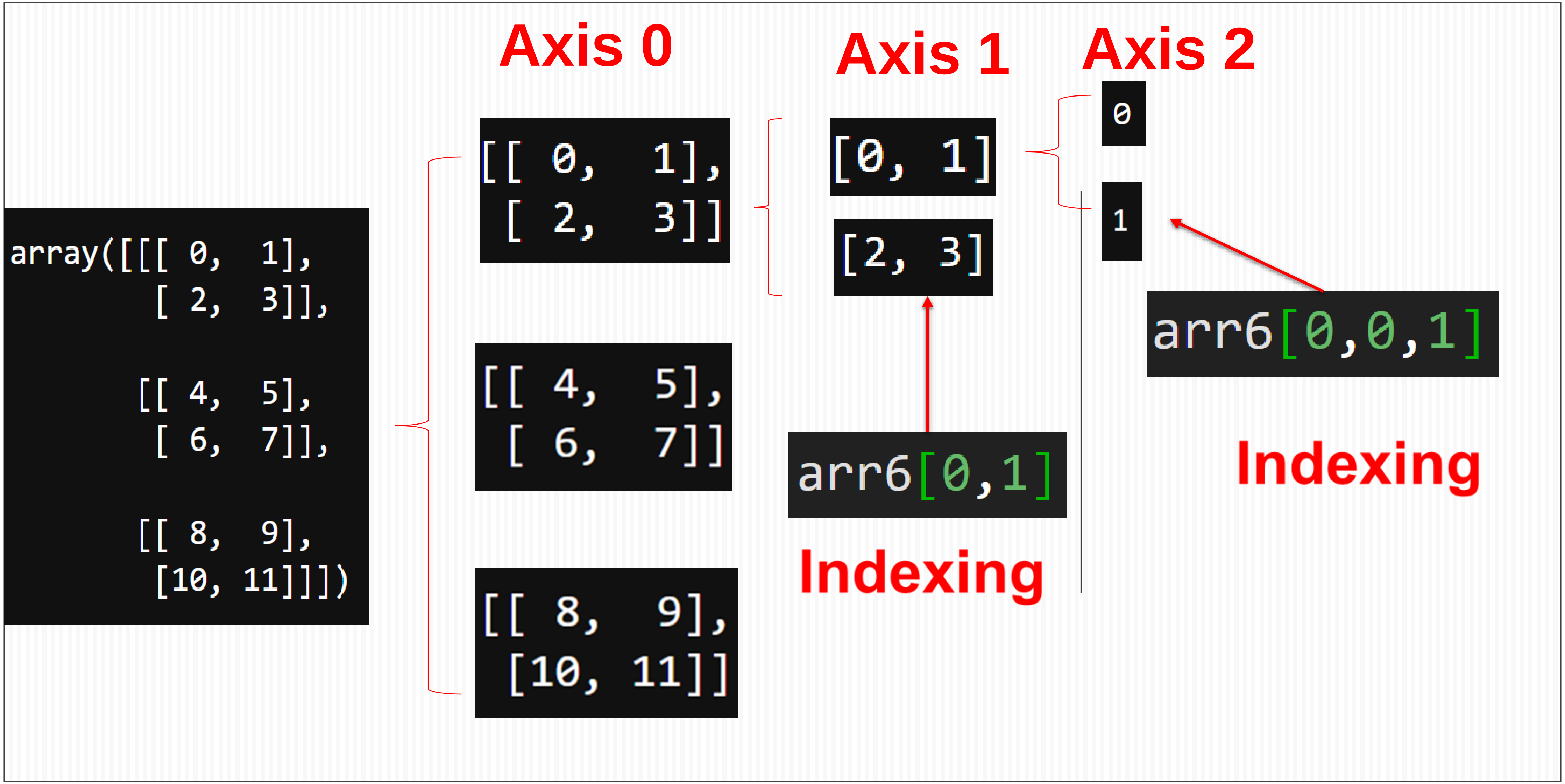
```
[[ 8,  9],
 [10, 11]]
```

```
[0, 1]
```

```
[2, 3]
```

```
arr6[0,1]
```

2 Axes, indexing, slicing



```
array([[[ 0,  1],
        [ 2,  3]],

       [[ 4,  5],
        [ 6,  7]],

       [[ 8,  9],
        [10, 11]]])
```

Axis 0

```
[[ 0,  1],
 [ 2,  3]]
```

```
[[ 4,  5],
 [ 6,  7]]
```

```
[[ 8,  9],
 [10, 11]]
```

Axis 1

```
[0, 1]
```

```
[2, 3]
```

```
arr6[0,1]
```

Indexing

Axis 2

```
0
```

```
1
```

```
arr6[0,0,1]
```

Indexing

3 “Vectorized” Operations

Vectorized Operations

In the context of high-level languages like Python, the term **vectorization** describes the use of optimized, pre-compiled code written in a low-level language (e.g. C) to perform mathematical operations over a sequence of data.

3 “Vectorized” Operations

4 Broadcasting

Array Broadcasting

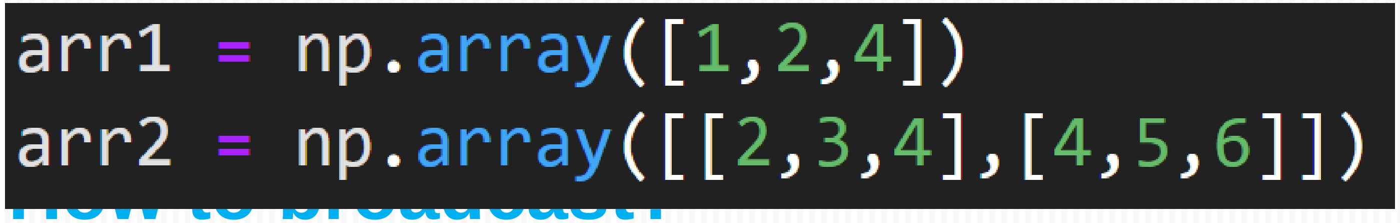
is a mechanism used by NumPy to permit vectorized mathematical operations between arrays of unequal, but compatible shapes.

4
Broadcasting

How to broadcast?

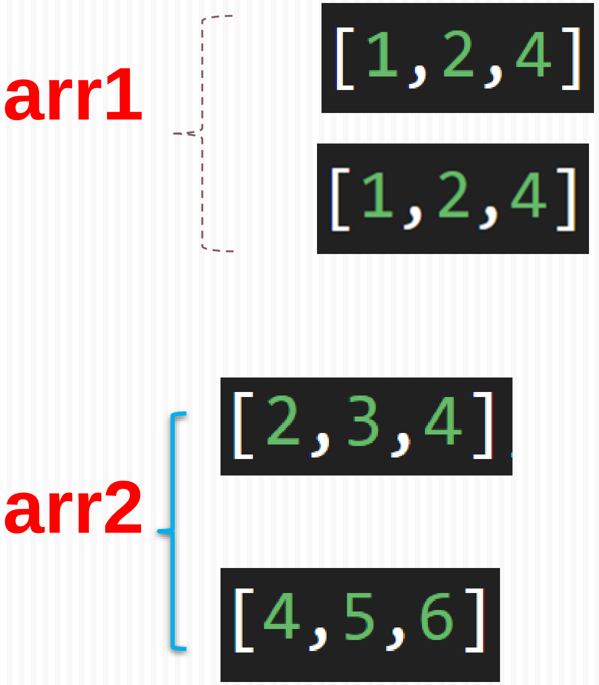
In effect, broadcasting is replicating the smaller array along the mismatched dimension.

4 Broadcasting



```
arr1 = np.array([1,2,4])
arr2 = np.array([[2,3,4],[4,5,6]])
```

How to broadcast?



arr1

```
[1,2,4]
[1,2,4]
```

arr2

```
[2,3,4]
[4,5,6]
```

4 Broadcasting

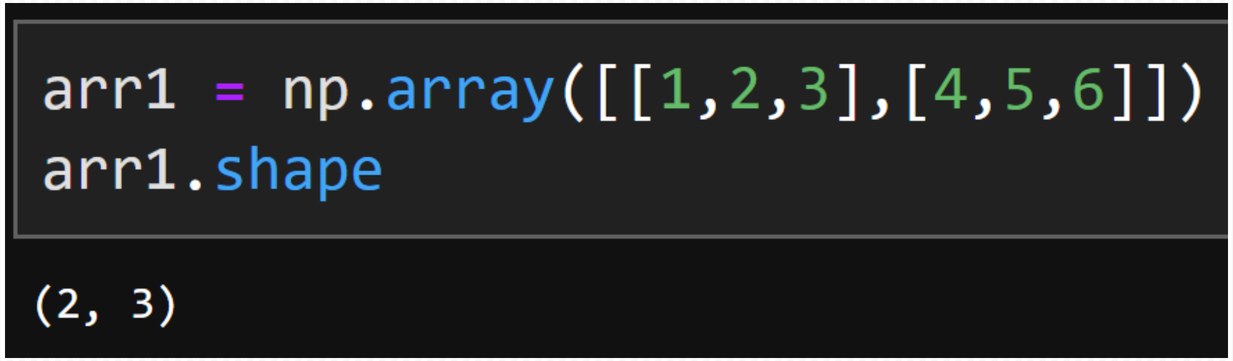
Rules of Broadcasting-- Condition 1

If two arrays **have the same dimensions** but different size, then check that each pair of aligned dimensions satisfy either of the following conditions:

- the aligned dimensions have the same size
- one of the dimensions has a size of 1

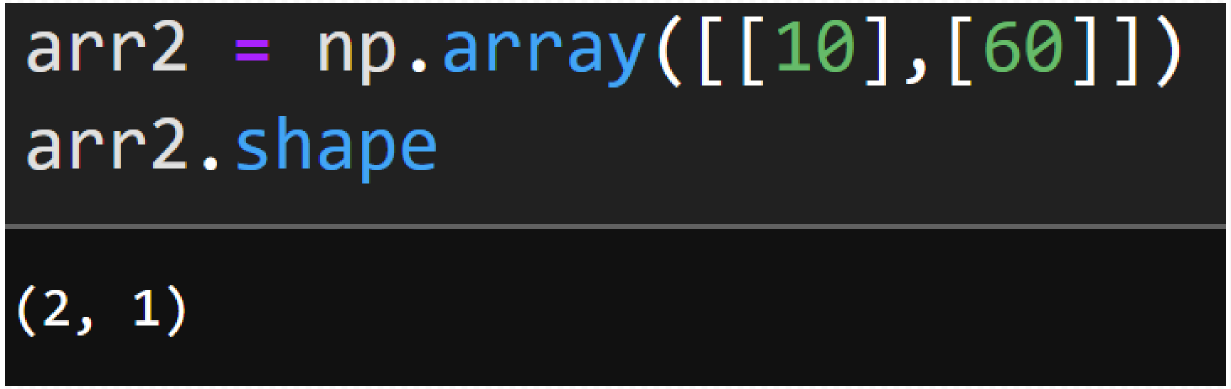
4 Broadcasting

4 Broadcasting



```python
arr1 = np.array([[1,2,3],[4,5,6]])
arr1.shape
```

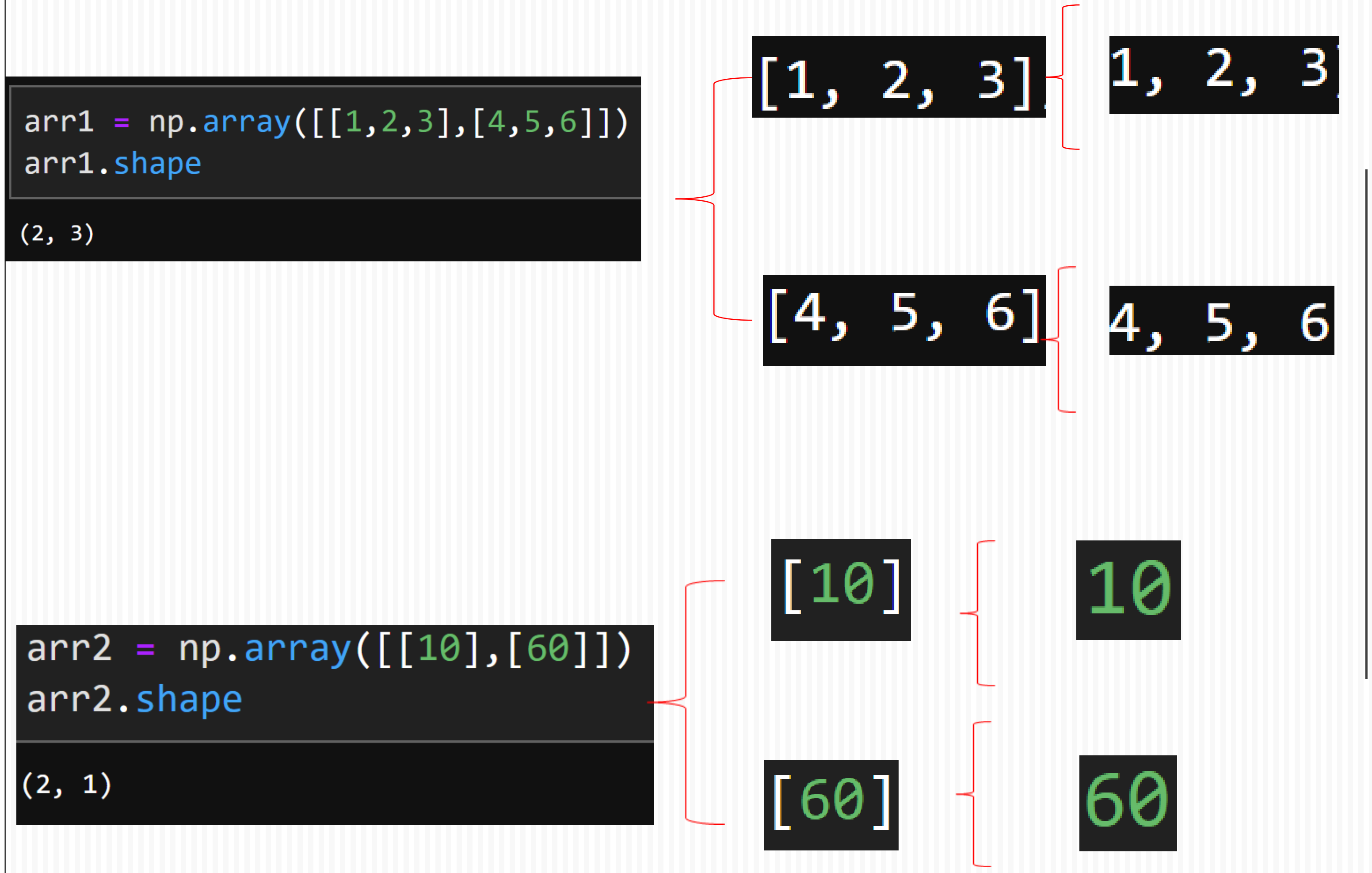
```
(2, 3)
```



```python
arr2 = np.array([[10],[60]])
arr2.shape
```

```
(2, 1)
```

4 Broadcasting

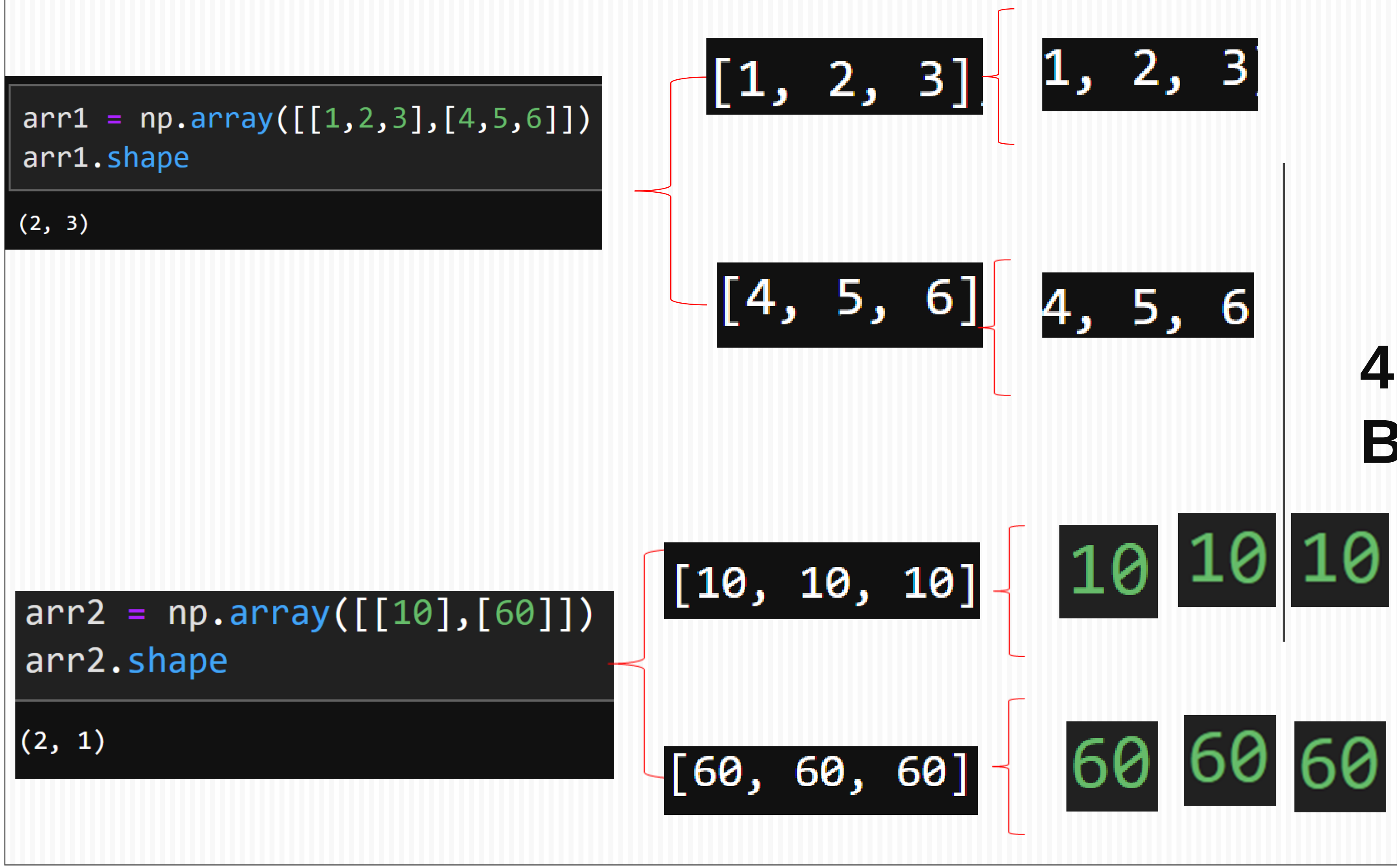


```
arr1 = np.array([[1,2,3],[4,5,6]])
arr1.shape
```

```
(2, 3)
```

```
arr2 = np.array([[10],[60]])
arr2.shape
```

```
(2, 1)
```



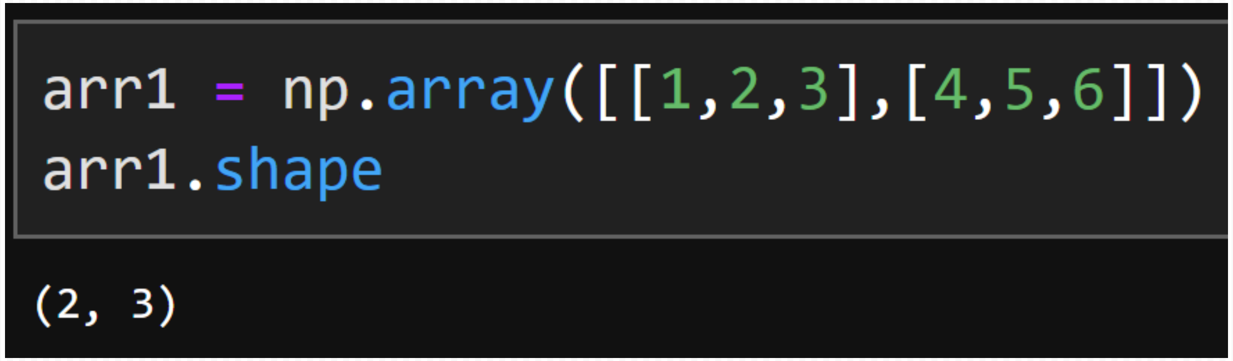
```
arr1 = np.array([[1,2,3],[4,5,6]])
arr1.shape
```

```
(2, 3)
```

```
arr2 = np.array([[10],[60]])
arr2.shape
```

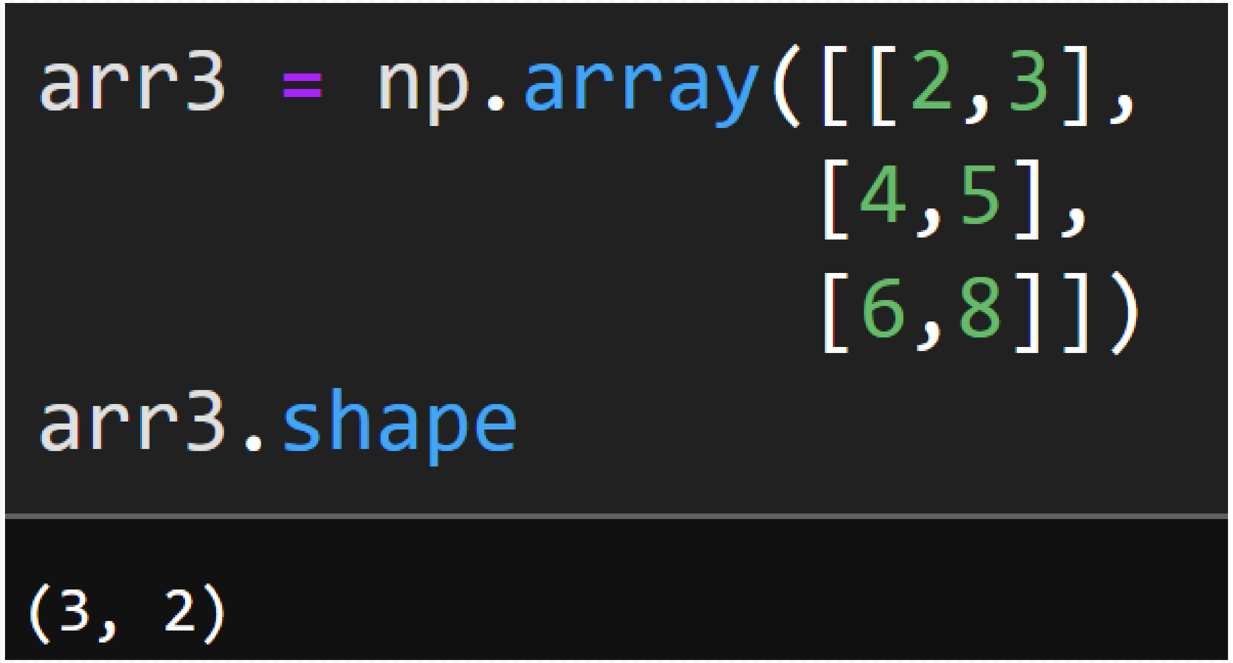
```
(2, 1)
```

4 Broadcasting



```
arr1 = np.array([[1,2,3],[4,5,6]])
arr1.shape
```

```
(2, 3)
```

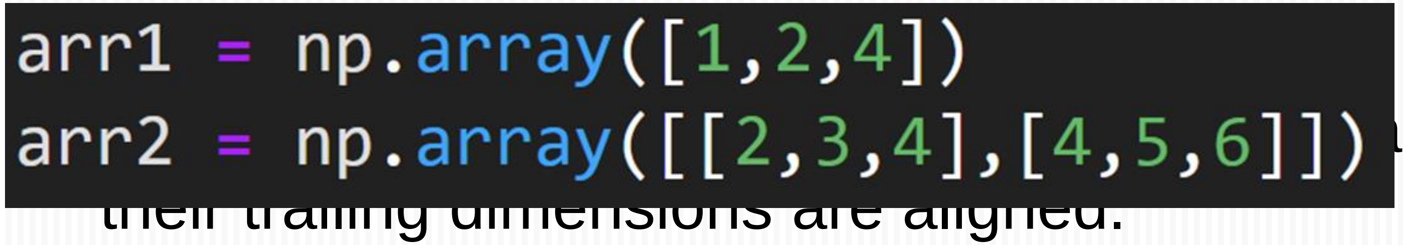


```
arr3 = np.array([[2,3],
                 [4,5],
                 [6,8]])
arr3.shape
```

```
(3, 2)
```

4 Broadcasting

Rules of Broadcasting-- Condition 2



```
arr1 = np.array([1,2,4])
arr2 = np.array([[2,3,4],[4,5,6]])
```

t

their trailing dimensions are aligned.

arr2: 2 , 3

arr1: 3

arr3: 2 , 3

arr4: 3, 2, 1

4 Broadcasting